11-2010

Efficient mining of multiple partial near-duplicate alignments by temporal network

Hung-Khoon TAN

Chong-wah NGO
Singapore Management University, cwngo@smu.edu.sg

Tat-Seng CHUA

Efficient Mining of Multiple Partial Near-Duplicate Alignments by Temporal Network

Hung-Khoon Tan, Chong-Wah Ngo, *Member, IEEE* and
Tat-Seng Chua, *Member, IEEE*

***Abstract*—This paper considers the mining and localization of near-duplicate segments at arbitrary positions of partial near-duplicate videos in a corpus. Temporal network is proposed to model the visual-temporal consistency between video sequence by embedding temporal constraints as directed edges in the network. Partial alignment is then achieved through network flow programming. To handle multiple alignments, we consider two properties of network structure: conciseness and divisibility, to ensure that the mining is efficient and effective. Frame-level matching is further integrated in the temporal network for alignment verification. This results in an iterative alignment-verification procedure to fine tune the localization of near-duplicate segments. The scalability of frame-level matching is enhanced by exploring visual keyword (VK) matching algorithms. We demonstrate the proposed work for mining partial alignments from two months of broadcast videos and across six TV sources.**

***Index Terms*—Partial near-duplicate, temporal graph, keyword matching.**

I. 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 [30], [18] becomes a topic of intensive studies recently, being applied to various emerging applications including copy enforcement [16], news video threading [29], and novelty ranking of web videos [12]. In handling redundant contents, one point of consideration is if overlapping information are useful or negligible. Previous methods [27] have mainly considered near-duplicates as redundant and focus on detecting and removing them from the retrieval list. However, such approach underscores the potential of near-duplicates. In a recent study, [3] further refined the definition of near-duplicate videos through a user-centric survey and concluded that similar clips differing in overlaid or added visual content should not be perceived as redundant. Conforming to this observation, [23] has also utilized the degree of redundancy between the videos to perform automatic refinement of video tags. The degree of overlap were used to define three levels of links, i.e., full-duplicates, part-of and overlap, which were shown to aid video categorization and clustering.

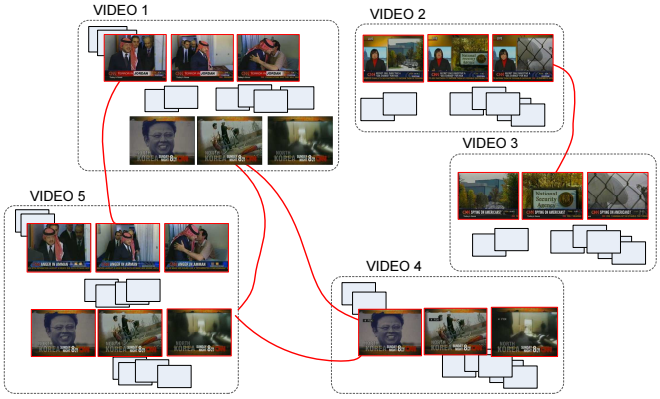

Fig. 1. Partial near-duplicate videos. Given a video corpus, near-duplicate segments create hyperlinks to interrelate different portions of the videos.

In general, there are two scenarios where near-duplicate videos are useful: (1) fully near-duplicate videos which share the same main plot but are supplemented by novel contextual information, or (2) *partial near-duplicate* video where only certain segments of the videos are near-duplicate to each other. In this paper, we focus on the detection of partial near-duplicate video. 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 [17], and video re-ranking by Page Rank like algorithm [12]. 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.

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. While these techniques are sufficient to handle fully duplicate videos, the localization

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 119508).

Hung-Khoon Tan and Chong-Wah Ngo are with the Department of Computer Science, City University of Hong Kong. Tat-Seng Chua is with the School of Computing, National University of Singapore.

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.

of duplicate video segments are often being carried out in a heuristic manner. Typical examples include voting scheme [8] 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 scenario in broadcast news video which contains overlapping fragments of news video. Such scenario are also observed for event-related queries [11] 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.

A. Proposed Work

This paper addresses the problem of partial near-duplicate detection and localization. Given two videos comprising M and N frames respectively, partial near-duplicate is defined as a temporally contiguous set of K near-duplicate frame-pairs across the two videos where K is considerably smaller than either M or N, or both. The partial-duplicate links among videos as shown in Figure 1 are established through partially aligning video content. The partial alignment is modeled as a network flow problem which essentially takes into account the joint visual-temporal consistency among videos. Figure 2 depicts the major flow of the proposed work. Given two videos, a temporal network is constructed to model the visual-temporal consistency among frames. For multiple partial alignment, the network is further split into multiple partitions. Frames in each partition are separately aligned by an efficient network flow algorithm. Further verification is imposed to derive the sets of must-align and cannot-align frames as constraints, which formulate the alignment problem in an iterative manner. Repeating the procedure over all video pairs, threads, which are groups of partial alignments, are then mined to inter-relate a large groups of videos.

The preliminary version of the work was published in [26] which focuses on the modeling of temporal network for localization of a *single* partial near-duplicate segment between two videos. In this paper, we further extend the network to handle *multiple* partial near-duplicate alignments between videos. We consider the *divisibility* and *conciseness* of the temporal network for efficient multiple alignment. The divisibility of a network refers to the feasibility of breaking the network into multiple smaller partitions during alignment. Conciseness, on the other hand, is a measure that evaluates how compact or conversely, the degree of redundancy, in the generated structure, which is related to the placement of frames in the network. Careful consideration of both properties leads to a simpler and concise network structure and more efficient algorithm than [26]. In addition, we consider visual keyword for alignment verification to get rid of exhaustive matching of local keypoints as in [26]. To handle robustness issue, weak geometric and neighborhood constraints are exploited for the matching of visual keywords. Different from [26] which demonstrated single partial alignment for applications such as movie tagging, we apply the work in this paper for mining multiple partial near-duplicates of broadcast videos. Broadcast videos are especially rich with partial duplicate contents which span across multiple channels and time duration. Mining all these links is a fundamental but crucial component for multimedia content analysis such as news story summarization or indexing.

The remaining of the paper is organized as follows. Section II discusses related work on near-duplicate retrieval and detection. Section III presents temporal network modeling to detect near-duplicate segments from two videos. The framework to handle multiple near-duplicate segment detection is then elaborated in Section IV. Section V explores how to efficiently verify the alignment result through frame-level matching by means of visual keyword. The proposed approach is then evaluated on the TRECVID [25] dataset and the results are reported in Section VI. Finally, we summarize our findings in Section VIII.

II. Related Work

Copy or near-duplicate detection and retrieval, as a timely research problem to several emerging application, has been intensively studied recently. Video copies are identical or approximately identical videos with similar appearance but subjected to various degree of transformations. Possible transformations include formatting, photometric and encoding variations, editing, cropping and caption or logo insertion. Previous copy detection [18], [10] focus on developing effective features or matching strategies that are robust towards such transformations. While copies are normally derived from the same source, near-duplicates can be sourced from different recording devices capturing the same scene, object or event. As a result, near-duplicate videos suffer from additional viewpoint distortions and therefore require frame-level matching [33], [30] to achieve desirable results in most circumstances. In this paper, the term 'near-duplicate' shall be used to refer to both categories since near-duplicate can be considered as the superset of copy.

A. Near-Duplicate Detection

Broadly, we can categorize existing works into three main groups: signature-based, keyframe-based and trajectory-based approaches. Signature-based approaches summarize video content into global descriptors for fast retrieval. The signatures can be computed by "averaging" the global color histogram [27], ordinal [10] or temporal-ordinal [2] features of the frames in a video. While being efficient, temporal information is missing in fingerprints and thus retrieval of partial near-duplicates is not supported. In addition, the evaluation conducted in [19] shows that signature-based approaches are effective only for copies with small transformations, which is not surprising owing to the information loss from averaging.

Keyframe-based approaches perform sparse analysis of video content by matching representative frames or keyframes sampled from videos. Different matching strategies have been proposed such as dynamic programming [5], [13], graph-based matching [4], windowing [27] and voting-based approaches [8]. Sliding window is sensitive to temporal resolution and is thus not suitable for retrieving near-duplicates with changes in frame rate. Dynamic programming finds the longest common

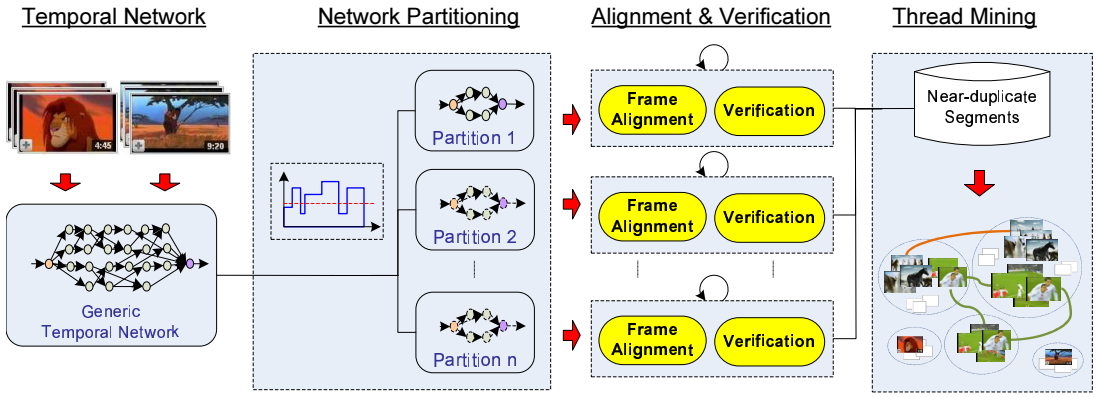

Fig. 2. Partial alignment of videos. Given two videos, a temporal network is constructed to model the visual-temporal consistency between the frames. A divisibility analysis is carried out to split the network into multiple partitions on which partial alignment is performed. The aligned frame pairs are then verified by frame-level matching in an iterative manner. Finally, overlapping near-duplicate segments are bridged to form a media network of videos.

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 visual content matching. In the end, the robustness of the approaches relies heavily on the visual content. Heuristic voting scheme was proposed in [8], [6] where the aggregation of voting from near-duplicate frames makes the near-duplicate shot detection robust to individual near-duplicate false positive frames. 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 [18] and then assign behavioral labels to each local descriptors. In another recent work [28], the whole shot was represented using a bag of trajectories where each trajectory in turn is described as temporal patterns of discontinuities. The extraction of trajectories is an expensive operation. Moreover, trajectory features are sensitive to camera motion and therefore their robustness is limited to copies, but not necessarily robust for near-duplicates, especially when viewpoint changes are involved.

B. Scalable Mining

The growing number of videos in Internet has made both scalable and reliable mining of near-duplicates a timely problem. Frame-level analysis, based on keypoint matching [31], [33], is popularly adopted for its robustness to large transformations and viewpoint distortions. Nevertheless, the analysis is computationally demanding. Various indexing techniques including locality sensitive hashing [16] and random histogram [7] have thus been proposed for fast matching of keypoints. Vector quantization, through clustering of keypoints to visual keywords, is also often adopted. By representing a frame as a sparse vector of keywords, indexing structure such as inverted file can be utilized for direct keyword matching [14]. In [22], a signature-specific indexing structure based on the bucketing of keypoint signature, named Glocal, was also proposed. The approach, nevertheless, is suitable only for large-scale copy detection.

To balance detection accuracy and speed, techniques for distributed and hierarchical mining of near-duplicates have also been proposed. In [32], domain knowledge is exploited for distributed mining. A temporal, semantic and visual partitioning model was proposed to divide video corpus into small overlapping partitions. Mining is performed separately on each partition, and the results are eventually merged by transitivity propagation. In [27], hierarchical mining was proposed by using global signature to detect candidate videos, followed by exhaustive search of near-duplicate keyframe pairs through keypoint matching. However, the speed can still be limited especially if a large number of candidate frame pairs is retained for keypoint matching. In [26], to reduce frame pair comparison, temporal network was proposed to align and verify the most likely frame pairs by imposing visual-temporal consistency. Compared to [27], temporal network is faster by more than 50 times with slight degradation in detection accuracy.

III. Single Partial Alignment

In this section, we briefly describe temporal network which was originally presented in [26]. Conventionally, the mapping of two frame sequences can be framed as an optimization problem to find the set of frame-pairs that maximizes the accumulated similarity while temporality are typically posed as constraints to the objective function. In contrast, in our approach, the temporal constraint is *structurally* embedded into a network structure as directed edges. The proposed structure is referred to as the *temporal network*. This structural embedding novelly converts the alignment problem into a transportation problem, or more specifically a network flow problem, where efficient algorithms are readily available.

A. Temporal Network

Given two videos, a video is designated as the *anchor* video $Q = \{q_1, ..., q_{|Q|}\}$ and the other as the *reference* video

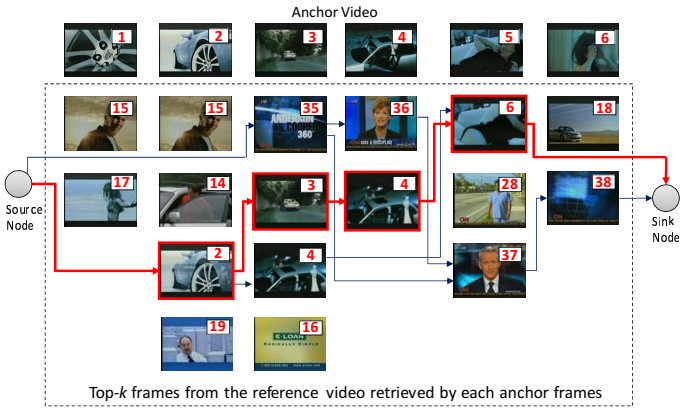

Fig. 3. A temporal network. The columns of the lattice are frames from the reference videos, ordered according to the *k*-NN of the query frame sequence. The label on each frame shows its time stamp in the video. The optimal path is highlighted. For ease of illustration, not all paths and keyframes are shown.

$R = \{r_1, ..., r_{|R|}\}$ where $|\cdot|$ denotes the number of frames in the videos. Temporal network is initially formed by querying the top-k similar frames from R using the query frames q_i. Figure 3 illustrates an example where an anchor video consisting of six frames retrieves six columns of top-k frames from the reference video[1]. Directed edges are established across the frames in the columns by chronologically linking frames according to their time stamp values. For example, the frame in a column with time stamp value t can link to another frame in a right-hand side column with time stamp larger than t. In other words, when tracing the list of connected edges from left to right, the time stamp values are monotonically increasing. 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. The set of all possible paths in the temporal network from the source to the sink node encompasses all possible frame alignments between videos Q and R which follow strict temporal coherency.

Denote the temporal network as $G = (\mathbf{N}, \mathbf{E})$ where $\mathbf{N} = \{N_1, \ldots, N_{|Q|}\}$ are columns of frames from R, where each column $N_i = [n_1, ..., n_k]$ is the retrieval result using $q_i \in Q$ as the query, while $\mathbf{E} = \{e_{ij}\}$ is the set of all edges where e_{ij} represents a weighted directed edge linking any two nodes from column N_i to N_j, respectively. Each edge is characterized by two terms: weight $w(\cdot)$ and flow $f(\cdot)$. Given an edge e_{ij}, its weight 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

$$w(e_{ij}) = Sim(q_j, n_j) \tag{1}$$

where n_j is the node in N_j and q_j is the query frame which retrieves n_j. Note that the weight does not depend on n_i which links to n_j. For any edge terminating at the sink node, the weight is assigned to zero. The flow $f(e_{ij})$, under our problem definition, is a binary indicator with value equal to 1 or 0. A valid solution is an unbroken chain of edges forming a path from the source node to the sink node where the flows at the edges traversed by the path is 1 while for all other edges, the flow value is 0. 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.

[1]In practice, a similarity threshold can be set to further prune highly dissimilar candidates from the top-k list.

B. Frame Aligment

The optimization is indeed an equivalent of the classical network maximum flow problem in operations research [1]. The objective is to find the optimal values of the flow variables $f(\cdot)$ that maximizes the total accumulated weight. In addition, the solution should obey the equilibrium requirement so 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. The frame alignment, based on network flow optimization, is thus formulated as:

$$\text{maximize} \sum_{e_{ij} \in \mathbf{E}} f(e_{ij}) w(e_{ij}) \tag{2}$$

subject to

$$\sum_{e_{in} \in E_{in}(n)} f(e_{in}) - \sum_{e_{out} \in E_{out}(n)} f(e_{out}) = 0, \quad \forall n \in \mathbf{N} \tag{3}$$

$$\sum_{e_{out} \in E_{out}(n_{src})} f(e_{out}) = 1 \tag{4}$$

$$\sum_{e_{in} \in E_{in}(n_{sink})} f(e_{in}) = 1 \tag{5}$$

$$0 \leq f(e_{ij}) \leq 1, \quad \forall e_{ij} \in \mathbf{E} \tag{6}$$

where $E_{in}(n)$ and $E_{out}(n)$ denote the set of in-coming edges and out-going edges of node n respectively while n_{src} and n_{dst} denote the source and sink nodes respectively. Equations 3, 4 and 5 impose the *flow conservation* constraints to control a well-behaved weight transfer from the source to the sink node. The network flow formulation is a special constrained linear program which exhibit the *unimodality* property [1]. The property ensures that the solution (the values of f) must also be binary when the right hand sides of the constraint equations are binary. This makes network flow optimization suitable to handle assignment problems. Thus, the set of nodes traversed by the optimal path indicated by $f(\cdot) = 1$ constitutes the solution and only edges along the selected path contribute to the accumulated weight. The temporal network always has a feasible solution since there always exists at least a valid path from the source node to the sink node and there are no dangling nodes in the structure. Therefore, by the convexity property, Equation 2 will always converge into a global solution [1].

IV. Multiple Partial Alignment

We consider different configurations of partial alignments frequently observed in broadcast news videos. Figure 4 depicts

three typical configurations, namely sequential, cross and self alignment. News stories can span across a certain period of time as more updates on a particular story are reported as it unfolds. In addition, important news can be broadcasted by various broadcast sources where the placement and visual editing of the news story are performed independently. In addition, different stations may also use different footages captured by different sources of the same scene.

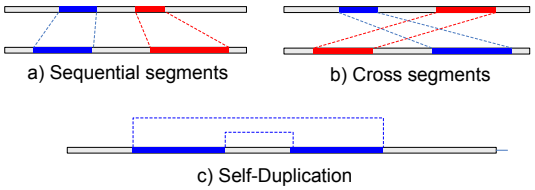

Fig. 4. Different configurations of multiple partial near-duplicate.

To handle multiple partial alignment, single alignments are performed repeatedly on the temporal network where nodes from previously detected segments are removed from the structure after each iteration. The step is carried out until no more new near-duplicate sequence can be found. However, the length of near-duplicate segment is typically only a fraction of the query video length and extracting small segments from a huge temporal network is not desirable. In this respect, two properties of the network are helpful, i.e., (a) *divisibility* which analyzes the feasibility of partitioning the network into multiple sub-networks, and (b) *conciseness* which measures the compactness of the structure in terms of the placement of positive keyframes in each column of the temporal network.

A. Divisibility and Early Partitioning

The divisibility of the network refers to the feasibility of the network to be cut into multiple non-interacting partitions so that near-duplicate detection can be performed independently on each partition. The boundaries of the partitions are the locations where there do not exist any valid sequences that cross and bridge two neighboring partitions. To find the partitions at such an early stage, the criterion that we use is sequence length where sequences which do not meet the required minimum length L_{min} are discarded. In essence, L_{min} specifies the minimum length of the near-duplicate segments that are considered as valid. Since no matching results are available at this point, a pessimistic model is employed. Given a column, the largest *maximal length* for all nodes in the column is used to determine its divisibility factor. The maximal length of a node is defined by the length of the longest path in the structure that traverses the node. It can be efficiently found through a variant of the forward-backward algorithm [26].

Denote the maximal length for node n_i as $L(n_i)$. The maximal length for a column N_k is $L(N_k) = max\{L(n_i)\}$ where $n_i \in N_k$. A new partition starts once a valid column p with $L(N_p) \geq L_{min}$ is detected, and terminates at column k only when the value of L drops below L_{min} and remains so over a period specified by the temporal window wnd,

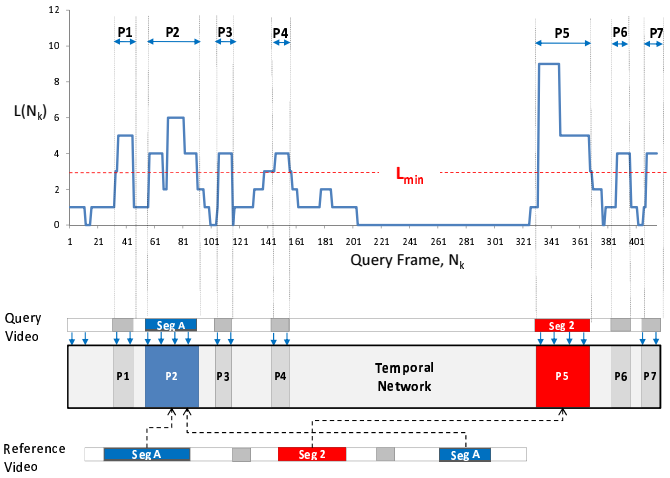

Fig. 5. Early partitioning of a temporal network. The largest value of the *maximal length* (y-axis) for all nodes in each column (anchor frames or the x-axis) is shown. A total of seven partitions are found and near-duplicate segment detections are then conducted separately on each partition. Ideally, the frames that make up the nodes in each partition are from the corresponding segments in reference video, as shown at the bottom of the figure. Thus, near-duplicate segment extraction can be carried out separately on each partition.

or equivalently $L(N_i) < L_{min}, i = \{k, \ldots, k + wnd\}$. Figure 5 shows the maximal length $L(N_k)$ for the columns of a temporal network constructed from two videos in one of our experiments. In this example, L_{min} is set to 3 and the network is partitioned into a total of 7 non-interacting partitions. Each partition is constructed using the frames from various locations of the reference videos, ideally from the corresponding near-duplicate segments, based on their similarity to the query frames in the partition. Thus, each partition is essentially self-sufficient where near-duplicate extraction can be carried out separately, without much interference from each other.

Performing multiple smaller network flow problems is significantly faster than running a large network flow problem as highlighted in [26]. This independence property is desirable since it supports *parallelization* where the processing of each partition can be safely delegated to different cores of a multi-core processor, a powerful feature when dealing with extremely long video sequences. Moreover, the cost to construct the sub-networks is almost negligible since they can be built incrementally from the generic structure.

B. Network Conciseness

The conciseness of the temporal network refers to the size of the network in each partition that are required to successfully detect the set of all near-duplicate segments for each partition where a smaller network is desirable. In other words, it evaluates the degree of non-redundancy in the constructed structure. The factor that directly contributes to conciseness is the choice of the signature used to retrieve and sort the reference keyframes in each column. Ideally, the top frames should be devoid of as many non-related frames as possible.

Global dense signatures, e.g., edge histogram and color moment, are less discriminative when handling near-duplicates that contain medium to large transformations such as cropping

and picture-in-picture. As a result, a deeper network with longer columns becomes necessary to ensure no near-duplicate segments are missed. On the other hand, local-point signature such as visual keywords [24] can generate a more relaxed list. Visual keywords (VK) are generated by quantizing keypoints into a visual dictionary through a clustering process and each entry in the dictionary, or the centroid of a cluster, corresponds to a word.

The scalability of VK for large-scale retrieval of near-duplicate frames has been investigated in [14]. In our case, employing VK to retrieve the neighbor lists for each query frame should translate to a sparser temporal network. Due to the sparse nature of VK, hashing technique such as inverted file indexing can be adopted to speed up the entire retrieval process. In the inverted file, the index stores the keyword-frame relationship, where each entry corresponds to a keyword and links to the list of keyframes which contains the word. Given a frame, the words are hashed into the index and the matched keyframes can be retrieved rapidly.

V. Efficient Alignment Verification

The joint consideration of visual-temporality, with conciseness and divisibility properties, increases the reliability and speed of temporal network in multiple partial alignment. Nevertheless, in face of large transformation or view-point distortion especially when precise localization is desired, frame-level matching becomes necessary to isolate random noisy sequences from genuine ones. In this respect, each initial alignment can serve as a filtering step to produce the set of most likely frame-pair candidates for matching. The role of matching is to verify the near-duplicate identity of the frame-pairs in the extracted sequences and partly to refine their boundary.

Interestingly, the result from frame-level matching can be re-used as the *a priori* information to perform refinement of the alignment result. Figure 6 illustrates how frame-level verification is performed in our framework. The results of verification is utilized to derive the set of *must-link* and *cannot-link* keyframes between two initially aligned sequences. A node is marked as a must-link node if frame-level matching between the reference and query frames referenced by the node identifies them as near-duplicate pair. Otherwise, it is marked as a cannot-link node. The must-link nodes are novelly used as stubs which further cut the partition into a series of smaller sub-networks on which network flows become increasingly efficient. For each sub-network, the must-link nodes, which define the boundaries between sub-network, will play the new role as the source and sink nodes. Cannot-link nodes, on the other hand, are removed from the sub-networks together with their edges. The construction of sub-networks is incremental in nature and therefore the process is an efficient one.

A. Keyword versus Keypoint Matching

In [26], verification is conducted by exhaustive pair-wise matching among *keypoints* extracted from two keyframes. Keypoint matching is a computationally expensive process due to the huge number of keypoint permutations between two

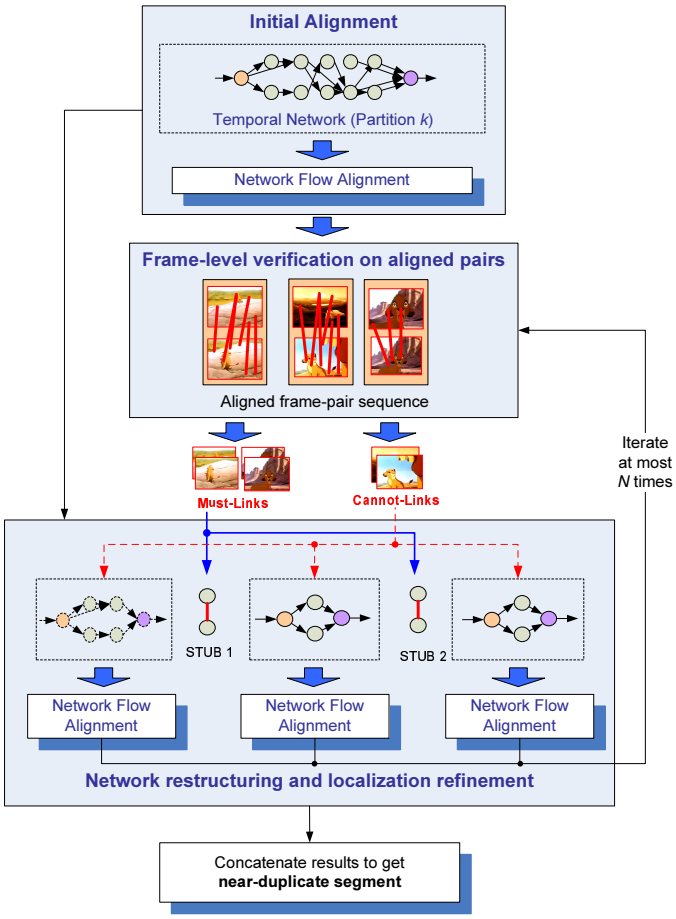

Fig. 6. Simultaneous verification and alignment. The sequence of candidate frame-pairs from the initial alignment result is subjected to frame-level verification. Near-duplicate frame-pairs (must-links) are used to cut the network into multiple sub-networks while non near-duplicate pairs (cannot-links) are removed from the structure. The process is iterated until the alignment converges. The alignments from sub-networks are then concatenated to form the near-duplicate segment.

frames. To speed up verification, we adopt visual *keywords* (VK) as an alternative to keypoints. This allows us to skip the expensive mapping process since the correspondences between local points are readily available when keypoints are quantized into their respective keywords. Thus, two local points p and q from two keyframes are assumed to match if they are assigned to the same keyword index through the Kronecker delta

$$f(p,q) = \delta_{k(p),k(q)} \tag{7}$$

where $k(\cdot)$ denotes the keyword index of a local point.

Unfortunately, direct matching reduces the discriminative power of keypoints due to lost in keypoint quantization. As a result, mapping based on visual keywords are coarse and noisy at best. Figure 7 (first column) shows the result of VK matching and as expected, the matching quality is poor and unacceptable. To overcome noise, we consider the underlying geometric and neighborhood consistency by two techniques named scale-rotation intersection (SRI) and neighborhood intersection (NI).

(a) VK (b) VK+SRI (c) VK+SRI+NI

Fig. 7. Keyword matching. The correspondences of local points based on (a) Visual Keyword (VK) mapping, (b) VK + Scale-Rotation Intersection (SRI) and (c) VK + SRI + Neighborhood Intersection (NI) are shown for a positive (top row) and negative example (bottom row).

B. Scale-Rotation Intersection (SRI)

Recently, [14] has examined the geometric consistency among the set of weakly corresponding point-pairs to prune keywords which are falsely matched. Two geometric parameters, scale or angle, are used as the basis to perform pruning of dissimilar local points that deviate from the dominant transformations as registered by majority of the points. The parameters are not estimated explicitly but are derived directly from the local patches. For example, the scale s_p of a local point p can indicate the level where p resides in the Difference of Gaussian (DoG) pyramid while the angle θ_p is the dominant orientation of the gray-level intensity of the patch, both of which are generated during keypoint detection [20]. Given the initial set of corresponding local points from VK matching M, for all keyword-pairs $(p, q) \in M$, the difference in log-scale $s_{p,q}$ and orientation $\theta_{p,q}$ are computed as follows

$$s_{p,q} = \log(2^{s_q - s_p}) \quad (8)$$

$$\theta_{p,q} = \theta_q - \theta_p \quad (9)$$

Considering the two parameters separately, the values are binned into the log-scale histogram h^s and orientation histogram h^θ respectively. Each bin reflects one kind of transformation performed by a group of keywords.

Ideally, the histogram of two near-duplicate frames should display a low entropy value and there exists prominent peaks in which most positive keyword-pairs can be found. By retaining only correspondences that are consistent with both the dominant scale and orientation, a more reliable correspondence set can be retrieved. Conversely, the histogram for two non near-duplicate frames should be relatively flat and has a high entropy score since the transformation is random and less focused. As a result, scale and orientation consistency will eliminate majority of the correspondences. Figure 8 depicts the mechanism of our approach. The dominant scale $\hat{s}$ and angle $\hat{\theta}$ are approximated using the largest bins in the two histograms and only keywords that are consistent with both $\hat{s}$ and $\hat{\theta}$ are retained, resulting in a reduced keyword set M_{SRI} as follows

$$M_{SRI} = h^s(\hat{s}) \cap h^s(\hat{\theta}) \quad (10)$$

where $M_{SRI} \subseteq M$ while $h^s(\hat{s})$ and $h^s(\hat{\theta})$ are the bins which contain the set of matched keywords that contribute to the dominant scale and angle respectively. Figure 7 (second column) illustrates the set of keyword-pairs in M_{SRI} and clearly SRI manages to prune away a considerable number of noisy correspondence set from the original mapping produced by VK alone.

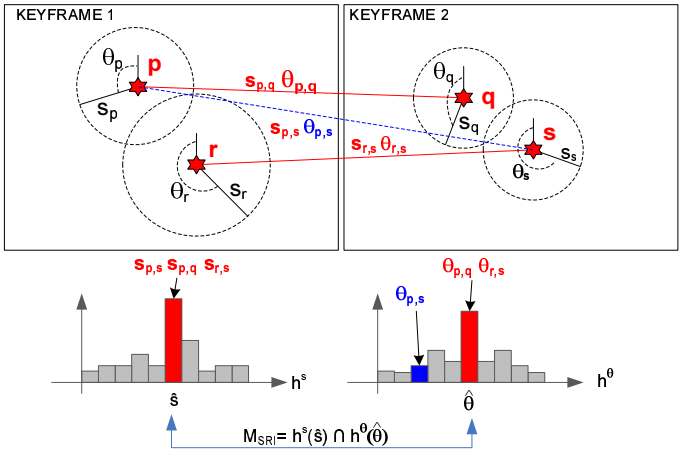

Fig. 8. *Scale Rotation Intersection*. The difference in the characteristics scale and orientation between two corresponding local points are computed and binned into the scale histogram h^s and orientation histogram h^θ respectively. Only correspondences that are consistent with both the dominant scale and orientation, i.e. binned into both $h^s(\hat{s})$ and $h^\theta(\hat{\theta})$ respectively, are retained. In this example, the correspondence (p, s) is considered a noise and is not included in M_{SRI}.

C. Neighborhood Intersection (NI)

SRI-based pruning alone is not sufficient as there still exists a non negligible number of false keyword-pairs that exhibits similar geometric transformation by chance. To further refine the matching quality, the neighborhood consistency of the corresponding keyword-pair can be employed. Figure 9 depicts how counting the number of common neighbors can segregate positive keyword-pairs from noisy ones. Given two near-duplicate keyframes, the neighbor set of two corresponding keywords are expected to be stable regardless of the overall global transformation between the two point set. Conversely, neighborhood consistency will tend to break apart for non near-duplicate pairs. Thus, noisy correspondences can be pruned by removing keyword-pairs which do not contain any

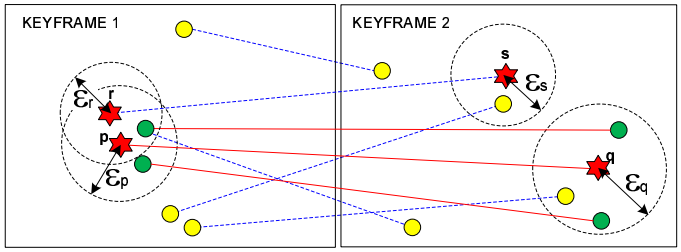

Fig. 9. *Neighborhood Intersection*. The ϵ-neighborhood for the positive correspondence (p, q) contains two neighbors (solid red line). Under our definition, the noisy correspondence (r, s) is not a neighbor (p, q), and is pruned since it does not contain any neighbors.

keyword-pair neighbors.

Given a point p, its neighbor set $N_\epsilon(p)$ constitutes all points that fall within a neighborhood of radius ϵ_p, i.e., $N_\epsilon(p) = \{n_p : d(n_p, p) < \epsilon_p\}$. Given two corresponding keyword-pairs (p, q) and (r, s) from M_{SRI}, with reference to Figure 9, (r, s) will only be considered to be a neighbor of (p, q) if (r, s) preserves a neighborhood relationship to (p, q) at both ends, i.e., $r \in N_\epsilon(p)$ and $s \in N_\epsilon(q)$. In other words, the keyword correspondence set is returned through the intersection between the ϵ-neighborhoods of the two points as follows:

$$M_{NI} = \{(p, q) : N_\epsilon(p) \cap N_\epsilon(q) \neq \emptyset,\ (p, q) \in M_{SRI}\}. \quad (11)$$

The approach is robust to scale change since the size of the neighborhood is automatically adjusted based on the statistics of the corresponding keywords in the frame-pairs. In fact, the appropriate value of ϵ_p varies for each point p where it is derived locally from the density of its k nearest neighbors $N_k(p)$ as follows:

$$\epsilon_p = \min(\mathcal{D}(n_p)),\ n_p \in N_k(p) \quad (12)$$

$$\mathcal{D}(n_p) = \text{median}(d(r, n_p)),\ r \in N_k(n_p) \quad (13)$$

where $d(\cdot, \cdot)$ is the $L2$-distance between two points in the frame and the density $\mathcal{D}(\cdot)$ of a point is computed from the median distance of its k-nearest neighbors. To speed up NI, kd-tree [9] can be used to retrieve the neighbors of the points where the time complexity to complete a search is proportional to $O(\log(n))$ where n is the number of points in a frame. Figure 7 (third column) shows the final correspondence set M_{NI}. Clearly, the quality of the matching pattern are significantly enhanced through successive prunings using SRI and NI.

VI. Experiment-I: Near-Duplicate Detection

We first verify the performance of keyword-based matching, specifically the performance of VK+SRI+NI, for near-duplicate keyframe detection. The following setup is used. Local points are detected using the combination of two detectors DoG [20] and Hessian Laplacian [21], and are described using the SIFT [20] descriptor. A total of 800K local features is randomly collected from the dataset. CLUTO [15], a publicly available clustering algorithm, is employed to cluster these samples into a visual dictionary. To improve efficiency, a hierarchical representation is adopted where the first layer contains 2000 clusters, which in turn are further clustered into 10 sub-clusters, resulting in a visual dictionary of 20,000 keywords. All experiments are conducted on an Intel Core2 Duo E8500 3.16GHz CPU with 3GB of RAM. For abbreviation, we refer our approach as VSN (VK+SRI+NI).

A. Dataset and Evaluation

We use the dataset in [30] for evaluation. The dataset contains a total of 7,006 shots collected from the TRECVID 2003 [25] dataset which covers 52 broadcasts of CNN and ABC channels in March of 1998. There is a total of 24,538,515 candidate keyframe-pairs and among them, only 3,384 pairs are near-duplicates. By VK retrieval using an inverted table, a total of 251,776 keyframe-pairs are returned. Among the returned list, 2591 positive frame-pairs or 76% true positives exist in the list and are subjected to keyword or keypoint matching.

We compare VSN to OOS (one-to-one symetric) in [31]. OOS employs exhaustive pair-wise keypoint matching to find the best possible correspondence, and then assesses the spatial regularity of point correspondence using entropy measure. Due to the nature of brute-force matching, OOS can basically achieves the best performance for near-duplicate detection despite extremely slow in speed. For both VSN and OOS, we employ SR-PE [30] to determine whether a frame pair is near-duplicate. SR-PE is a measure which assesses the spatial regularity of point correspondence using entropy measure. For evaluation, three measures are used, i.e., recall, precision and f-measure which are formulated as follows:

$$\text{Recall} = \frac{\#\text{Correctly detected near-duplicate frames}}{\#\text{Total near-duplicate pairs}} \quad (14)$$

$$\text{Precision} = \frac{\#\text{Correctly detected near-duplicate frames}}{\#\text{Detected near-duplicate pairs}} \quad (15)$$

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

Recall measures the completeness of the near-duplicates in the returned list with respect to the ground-truth while precision assesses the accuracy of the detected positives. F-measure calculates the fitness of ground-truth and detected ND pairs by jointly considering recall and precision.

B. Comparison between Keyword and Keypoint Matching

Table I shows the comparison result between VSN and OOS for near-duplicate detection. VSN manages to maintain a similar precision (above 0.9) as OOS. This confirms that context information, namely geometric and neighborhood constraints, is capable of compensating the losses in keyword quantization. However, a slight drop (around 5%) is observed in recall performance when keywords are mismatched due to quantization error during VK mapping and SRI and NI pruning.

Table II shows the contribution of each component towards VSN. Clearly, the performance of VK matching alone is not satisfactory. With SRI and NI, the F-measure is improved by 22% and 25% respectively. VK alone produces the best recall

TABLE I
COMPARISON BETWEEN KEYWORD AND KEYPOINT.

	VSN (Keyword)	OOS [30] (Keypoint)
Recall	0.63	0.68
Precision	0.92	0.93
F-Measure	0.75	0.78
VK retrieval time[a]	2 min 26 sec	
Total matching time[b]	1h 40s	6h 48m
Average matching time	14ms	97ms

[a]Total runtime to retrieve the set of 251,776 keyframe pairs through the inverted table.
[b]Total runtime for VSN to match all keyframe-pairs.

TABLE II
PERFORMANCE CONTRIBUTION FROM SRI AND NI.

	VK	VK+SRI	VK+SRI+NI
Recall	0.70	0.68	0.63
Precision	0.52	0.79	0.92
F-Measure	0.60	0.73	0.75
Improvement	-	22%	25%

but unfortunately it has a low precision performance, which makes it unreliable for near-duplicate detection. When SRI and NI pruning are imposed, recall drops by an acceptable rate of 10% but there is a significant increase (77%) in precision which approaches the performance of keypoint matching (OOS). In average, referring to Table IV, SRI prunes 54% of the 180 correspondences for a frame-pair comparison while NI prunes an additional 29%. For negative correspondences, the pruning removes almost all the correspondences. Table III further shows the impact of vocabulary sizes towards VK and VSN in terms of F-Measure. As the vocabulary size increases from 20K to 40K, the performance of VK improves, mainly due to better precision, before stabilizing at 35K. On the other hand, VSN maintains a consistently superior performance to VK with F-Measure within the range of 0.72 to 0.75.

For speed efficiency, VSN delivers a significant 6 times speed-up (in average 14ms per frame-pair) compared to OOS, as shown in Table I. For VSN, a break-up of the execution time is shown in Table IV for both the positive and negative keyframe-pairs. Due to the use of k-d tree, the time required to perform NI-pruning is surprisingly not as large as expected compared to SRI-pruning despite involving a retrieval process to determine the neighbors for each point. For the negative set, the time to perform NI is basically negligible since most correspondences can be successfully filtered out by SRI.

VII. EXPERIMENT-II: MULTIPLE PARTIAL ALIGNMENT MINING IN BROADCAST VIDEOS

In this section, we experiment the performance of temporal network for mining multiple partial alignments from broadcast videos. For evaluation, we use the TRECVID 06 corpus [25] which is composed of news videos from different TV channels including CNN, MSNBC, LBC, HURRA, CCTV and NTDTV. The total duration of these videos is 165 hours, spanning over a period of 2 months (59 days).

TABLE III
F-MEASURE FOR DIFFERENT VOCABULARY SIZE.

	20K	25K	30K	35K	40K
VK	0.60	0.62	0.63	0.65	0.65
VK+SRI+NI	0.75	0.74	0.72	0.75	0.74

TABLE IV
AVERAGE RUNTIME AND NUMBER OF CORRESPONDENCES.

	VK	SRI	NI	SR-PE	Others[a]
Positive frame-pairs					
#Correspondences	180	83	33	-	-
Runtime (ms)	0.5	1.9	2.6	4.1	6.7
Negative frame-pairs					
#Correspondences	30	3	0	-	-
Runtime (ms)	0.5	2.3	0	0	10.7

[a]Includes the duration to read input files, I/O operations, etc.

A. Setup and Evaluation

A total of 391,456 keyframes are extracted from the whole corpus. For each keyframe, keypoints are extracted. A visual dictionary of size 20K are constructed by randomly selecting 700,000 keypoints. We follow the setup in [32] for performance evaluation. The dataset is temporally divided into multiple groups. Each temporal group is handled separately and the final results are retrieved by linking the detected near-duplicate threads in each group via transitivity propagation. Since not all videos are compared, the experiment could complete in a shorter time, without much impact to detection performance[2]. The dataset is divided into 10 groups where each partition spans a duration of 8 days and there is an overlapping of 2 days for adjacent groups.

We use two-level of evaluation, at the thread and segment levels. Threads are the partial alignments found on different videos across time and TV channels while segments[3] refer the locations of the threads in each video.

- $Precision_{thd}$ and $Recall_{thd}$ are computed at the thread-level granularity where a ground truth thread is considered to be detected if at least two segments from the thread are found to be true positives.
- $Precision_{seg}$ and $Recall_{seg}$ consider the number of segments in the detected threads. If the ground truth thread is split into multiple threads, only the largest thread would be considered to compute recall.

For recall, we use the ground truth in [32] where neighboring positive keyframe-pairs are grouped into a total of 107 threads across videos. The ground truth covers only live news videos and excludes other contents such as anchor person and news openings. For precision, we manually annotate the set of *all* detected threads, including those not included in the ground truth. To avoid ambiguity, the following guidelines are used for annotation. First, the boundary of partial near-duplicate segments should include no more than three falsely aligned

[2]YanTao et al. [32] reported that temporal grouping maintains a 96.7% recall or completeness of near-duplicate pairs on the same data set
[3]The left and right boundaries of a near-duplicate segment are derived from the shot boundaries nearest to the first and the last keyframes of the segment respectively.

keyframes. Second, a thread is allowed to contain multiple near-duplicate scenes if they always co-occur together, e.g., the scene of an anchor person followed by a live news footage. Lastly, the number of noisy segments in a thread must not be larger than one-fifth of the thread size.

B. Performance Comparison

We compare the proposed approach against Hough Transform (HT) [8]. HT weakly measures the temporal alignment between videos by projecting the matched keyframes into a histogram of time lags. In other words, each bin in the histogram accumulates the number of keyframe matches with similar time lags. Peaks in the histogram indicate potential alignments between two videos. To detect near-duplicate segments of arbitrary time lag, contiguous frame-pairs from each bin are linked to form segments [22]. HT is popularly adopted in the literature due to its simplicity and efficiency. In the implementation, we use the same technique as presented in Section IV-B to retrieve the set of all corresponding frame-pairs. Then, weak geometric consistency (WGC) analysis [14] is performed to verify keyword matching before conducting HT. In practice, thresholding on the WGC score is necessary to decide whether there is a match between two frames. The setting of threshold is sensitive to the mining performance. We only report the best possible results in this section, with the threshold set to 0.7.

We also compare the performance to the original version of temporal network, named TN, in [26]. TN uses edge histograms (EH) as global signatures without indexing support to construct the temporal network. The network is denser with excessive number of nodes due to the use of EH. As a result, a slight modification is required for TN to achieve a manageable runtime, where early partitioning as presented in Section IV-A is also applied to TN. In addition, TN adopts OOS for frame-level verification, which can produce more reliable keyframe matching but at the expense of speed efficiency. For abbreviation, we name our proposed approach as TN-E, namely the efficient version of TN. For both approaches, we limit the number of nodes in each column of the network to a maximum of 200. To avoid redundant computation, keyframe-pairs that are highly unlikely to be near-duplicate are filtered out through thresholding. The threshold value is determined empirically and is set as low as possible to eliminate only highly dissimilar keyframe-pairs. For the EH signature used in TN, the minimum threshold is set to 0.70 and for the VK signature used in TN-E, the value is set to 0.01.

Tables V and VI show the performance comparison. TN-E and TN outperform HT in both recall and precision at thread and segment levels. For HT, the result indicates that there is a trade-off between recall and precision. HT can achieve either competitive recall or precision to TN-E, but not both, by varying the WGC threshold. In addition, TN-E extracts 89% and 157% more true positive threads and segments than HT, respectively. This is due to the effective frame-level verification framework adopted by TN-E and TN, which is lacking in HT. The boundaries of HT localization is coarse, leading to neighboring segments being erroneously joined when grouping segments into threads.

TABLE V
PRECISION OF PARTIAL NEAR-DUPLICATE ALIGNMENT.

	HT	TN[26]	TN-E
Detected Threads	829	941	1492
Positive Threads	767	941	1449
$Precision_{thd}$	0.93	0.94	0.97
Detected Segments	3474	4457	7031
Positive Segments	2262	3881	5833
$Precision_{seg}$	0.65	0.87	0.83

TABLE VI
RECALL OF PARTIAL NEAR-DUPLICATE ALIGNMENT.

	HT	TN	TN-E
Detected Positive Threads	83	101	107
$Recall_{thd}$	0.77	0.94	1.00
Detected Positive Segments	496	768	737
$Recall_{seg}$	0.43	0.67	0.64

TN-E delivers almost similar performance to TN but in addition, manages to detect 54% more positive threads despite having a similar setting in terms of the maximum number of nodes per column in the network (set to 200). This shows that the list of frames retrieved by VK is more precise than EH. As a result, more near-duplicate segments are embedded into the temporal network for TN-E. In TN, EH signatures construct a more complicated temporal structure with more redundant conductivities compared to VK signatures.

Figure 10 shows four positive threads that mined from broadcast videos by TN-E in our experiment. Three news topics are covered by the four threads and they span across three different TV channels (thread 2) and spread across three days (thread 1). Threads 2 and 3 are cross-alignment while threads 2 and 4 are sequential-alignment. Detecting the near-duplicate segments reveals the co-related parts in the videos where additional novel information can be found, as shown by the neighboring frames adjacent to the near-duplicate segments.

C. Speed Efficiency

Table VII lists the speed for various approaches. Compared to TN and TN-E, due to the use of binning technique, HT is highly efficient and requires only 9 minutes to complete frame alignment. However, HT depends on WGC thresholding to achieve acceptable alignments and when thresholding runtime is included, the total runtime for HT is almost similar to TN-E. In comparison, both TN and TN-E requires a longer time to match video-pairs for involving optimization to align partial near-duplicate segments. The average runtime to align two videos is 55 seconds for TN and around 8 seconds for TN-E, where TN-E is faster than TN by around 6.8 times. The improvement for partial alignment alone is more significant, namely around 10 times, while frame-verification in TN-E is 5.4 times faster than TN.

Table VIII shows the number of nodes, edges and partitions processed by TN and TN-E. In order to evaluate the conciseness of TN and TN-E, we compute the size of the constructed network as $B \times C \times N$ where B and C are the average number of nodes and edges in a partition, respectively, and N is the

Fig. 10. Four threads detected by TN-E for 5 videos from 24 to 26 December 2005 from 3 different stations, NBC (English news station), Phoenix (Chinese news station) and LBC (Arabic news station). The threads link the related segments across multiple channels and time on three stories, i.e., 'Iraq War' (Thread 2 and 3), 'One Year after the 2004 Tsunami' (Thread 1) and 'Christmas message by the new Pope' (Thread 4).

number of partitions in the temporal network. The size of the structure for TN (constructed using EH signature) is at least 27 times larger than TN-E (constructed using VK signatures) although from Table V, TN-E could detect 58% more near-duplicate segments from the structure. This shows that TN-E generates a more concise structure compared to TN.

D. Discussions

In the news domain, broadcast videos contain not only copies but also near-duplicates where the latter cover a wider

TABLE VII
TOTAL RUNTIME IN MINING NEAR-DUPLICATE SEGMENT FROM 165-HOUR VIDEOS.

	HT	TN	TN-E
Frame retrieval	3h 48m	-	3h 48m
WGC thresholding	8h 24m	-	-
Partial alignment	9m	3d 12h 56m	7h 50m
Frame-level verification	-	8h 46m	1h 38m
Total	12h 21m	3d 21h 42m	13h 16m

TABLE VIII
STATISTICS ON THE SIZE OF TEMPORAL NETWORK.

	TN	TN-E
Average #Nodes	1185	329
Average #Edges	1937	575
Average #Partitions	23	10
Network size	52,792,953	1,891,750

range of variations especially when derived from different capturing devices. Figures 11(a)-(c) give some examples of the near-duplicate segments detected by TN-E in our experiments. Figures 11 (b) and (c) show two difficult examples due to moving artefact and picture-in-picture transformation respectively. The ability to detect such sequences is attributed to the capacity of temporal coherence. Although individual frame-pair may exhibit low similarity scores, temporal network successfully gives precedence to the sequence of frames by compromising temporal consistency and visual similarity.

However, TN-E encounters difficulty when two unrelated scenes resemble each other. Figures 11(d)-(f) show some false positives which are erroneously identified by TN-E as near-duplicate segments. Most of the false positives are caused by the use of similar template by the same broadcast channel to report different news events. Another category of error is similar scene, for example, when both segments show a person with similar pose and covering a major portion of the keyframes, a scenario which is common for interview scenes. Error can also happen when matching textual footages. These examples, although not considered as near-duplicates, are indeed visually similar.

While TN-E is competent for matching segments, the algorithm is not suited to detect isolated near-duplicate frames which do not display any temporal relationship with their neighbors. In our experiments, a conservative value of $L_{min} = 3$ results in significant improvement in speed but near-duplicate segments shorter than L_{min} would be missed. Another possible limitation with TN-E is the handling of segments containing reverse scenes where the same scene is depicted in reverse of the original scene. Such scenes are rare in general although they can still be found in advertisement and movie videos. TN-E can handle this case by a slight modification as follows. An additional temporal network needs to be constructed differently where the edges are linked based on a descending instead of an ascending chronological manner. Since only few valid sequences are expected to be chained, the network constructed in a reverse order will be sparse. In other words, the optimization can be efficiently carried out while incurring few computational overhead.

(a) (b) (c)

(d) (e) (f)

Fig. 11. Six near-duplicate segments detected by TN-E. Only one corresponding frame-pair is shown for each near-duplicate segment. (a)-(c) shows some correctly detected near-duplicate segments. The detected segments contain various degree of variation from editing effect such as logo insertion, photometric and view point variations, moving artefact and picture-in-picture transformation. (d)-(f) shows several incorrectly detected near-duplicate segments. The errors are caused by the use of similar templates (d, e and f), similar scenes (e) and textual footages (f).

VIII. CONCLUSION

We have presented temporal network for matching partial near-duplicate videos where threads or groups of near-duplicate segments are mined from news videos. Temporal constraints are embedded into the network and partial alignment is novelly posed as a network flow problem. The framework uses inverted file indexing of visual keywords to ensure a sparser structure and in addition, partitions the network into smaller independent parts for multiple partial alignments. Verification is further conducted by visual keyword matching, together with scale-rotation and neighbourhood intersection, resulting in an iterative partial alignment framework. In our approach, the proposed temporal network is 6.8 times more efficient than the original version in [26] when performing mining on a large video corpus of 165 hours. In addition, compared to Hough Transform (HT) technique, temporal network is capable of keeping a balance between recall and precision. For future work, we plan to apply the proposed technique for applications such as news topic discovery, topical video browsing and summarization. In addition, the performance of TN-E can be further improved by considering two-tier temporal filtering. Hough Transform can be utilized to perform an initial coarse temporal filtering and the precision of TN-E can be employed to optimize the sequence extraction process.

REFERENCES

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

1993.

[2] L. Chen and F. W. M. Stentiford. Video sequence matching based on temporal ordinal measurement. *Pattern Recognition Letters*, 29(13):1824–1831, Oct 2008.

[3] M. Cherubini, R. de Oliveira, and N. Oliver. Understanding near-duplicate videos: A user-centric approach. *ACM International Conference on Multimedia*, pages 34–44, 2009.

[4] C.-Y. Chiu, C.-S. Chen, and L.-F. Chien. A framework for handling spatiotemporal variations in video copy detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(3):412–417, 2008.

[5] C.-Y. Chiu, C.-H. Li, H.-A. Wang, C.-S. Chen, and L.-F. Chien. A time warping based approach for video copy detection. *International Conference on Pattern Recognition*, 3:228–231, 2006.

[6] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. *ACM International Conference on Image and Video Retrieval*, pages 549–556, 2007.

[7] W. Dong, Z. Wang, M. Charikar, and K. Li. Efficiently matching sets of features with random histograms. *ACM International Conference on Multimedia*, pages 179–188, 2008.

[8] M. Douze, A. Gaidon, H. Jegou, M. Marszatke, and C. Schmid. Inria-lear's video copy detection system. *TRECVID*, 2008.

[9] Friedman, Bentley, and Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[10] A. Hampapur, K. Hyun, and R. M. Bolle. Comparison of sequence matching techniques for video copy detection. *SPIE: Storage and Retrieval for Media Databases*, 4676(194):194–201, Jan 2002.

[11] R. Hong, J. Tang, H.-K. Tan, S. Yan, C.-W. Ngo, and T.-S. Chua. Event driven summarization for web videos. *The First SIGMM Workshop on Social Media*, pages 43–48, 2009.

[12] W. H. Hsu, L. S. Kennedy, and S.-F. Chang. Video search reranking through random walk over document-level context graph. *ACM International Conference on Multimedia*, pages 971–980, 2007.

[13] X.-S. Hua, X. Chen, and H.-J. Zhang. Robust video signature based on ordinal measure. *International Conference on Image Processing*, 1:685–688, 2004.

[14] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. *European Conference on Computer Vision*, pages 304–317, 2008.

[15] G. Karypis. http://glaros.dtc.umn.edu/gkhome/views/cluto/.

[16] Y. Ke, R. Sukthankar, and L. Huston. Efficient near duplicate detection and sub-image retrieval. *ACM International Conference on Multimedia*, pages 869–876, 2004.

[17] L. Kennedy and S.-F. Chang. Internet image archaeology: Automatically tracing the manipulation history of photographs on the web. *ACM International Conference on Multimedia*, pages 349–358, 2008.

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

[19] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford. Video copy detection: a comparative study. *ACM International Conference on Image and Video Retrieval*, pages 371–378, 2007.

[20] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[21] K. Mikolajcyk and C. Schmid. An affine invariant interest point detector. *European Conference on Computer Vision*, pages 128–142, 2002.

[22] S. Poullot, M. Crucianu, and O. Buisson. Scalable mining of large video databases using copy detection. *ACM International Conference on Multimedia*, pages 61–70, 2008.

[23] S. Siersdorfer, , J. S. Pedro, and M. Sanderson. Automatic video tagging using content redundancy. *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 395–402, 2009.

[24] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. *IEEE International Conference on Computer Vision*, 2:1470–1477, Oct 2003.

[25] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVID. *ACM international workshop on Multimedia information retrieval*, pages 321–330, 2006.

[26] H.-K. Tan, C.-W. Ngo, R. Hong, and T.-S. Chua. Scalable detection of partial near-duplicate videos by visual-temporal consistency. *ACM International Conference on Multimedia*, pages 145–154, 2009.

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

[28] X. Wu, M. Takimoto, S. Sato, and J. Adachi. Scene duplicate detection based on the pattern of discontinuities in feature point trajectories. *ACM International Conference on Multimedia*, pages 51–60, 2008.

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

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

[31] W.-L. Zhao, C.-W. Ngo, H.-K. Tan, and X. Wu. Near-duplicate keyframe identification with interest point matching and pattern learning. *IEEE Transactions on Multimedia*, 9(5):213–238, Aug 2007.

[32] Y.-T. Zheng, S.-Y. Neo, T.-S. Chua, and Q. Tian. The use of temporal, semantic and visual partitioning model for efficient near-duplicate keyframe detection in large-scale news corpus. *ACM International Conference on Image and Video Retrieval*, pages 409–416, 2007.

[33] J. Zhu, S. C. H. Hoi, M. R. Lyu, and S. Yan. Near-duplicate keyframe retrieval by nonrigid image matching. *ACM International Conference on Multimedia*, pages 41–50, 2008.