Real-Time Video Copy-Location Detection in Large-Scale Repositories

Bo Liu
The Hong Kong Polytechnic University

Zhu Li
Huawei Technology

Linjun Yang
Microsoft Research Asia

Meng Wang
National University of Singapore

Xinmei Tian
University of Science and Technology of China

By exploring the temporal relationships inherent in video, a probabilistic framework can help identify and locate copies of query videos.

Video copy-detection, the purpose of which is to find a video copy in a repository, is important for many applications.[1-3] Our research focuses on locating video clips that are copied but maintain frame correspondence. Although this issue has been investigated for several years in multimedia research, efficient and scalable copy-detection is still a challenging problem due to the considerable computational costs. Copy location can be seen as an extension of copy detection. Rather than merely detecting the existence of a query video in a database, the location system must find the query in the database, a process that introduces additional computational costs.

Despite the computational costs, copy location is valuable for many applications. For example, if we would like to see whether a video program, such as an advertisement, is broadcast in a large video program repository, copy location can be used to find the location of the specified program. Also, it can be used in many other applications, including video data mining, copyright protection, and so on.

This article proposes a fast video copy-location method that can find the location of a given query in a large video repository in real time, regardless of the query length. The method proposed here consists of two major contributions. First, it includes a probabilistic model for video copy location to formulate the task as a likelihood maximization problem. Second, it includes a simplified approach to reduce the maximization problem into a set of 0-1 value problems based on the indexing structure.

Probabilistic modeling

For copy location, it's necessary to project the video sequence into a certain feature space and transform video copy location into a feature-matching problem in this feature space. For each video V, it can be first segmented into a set of frames $V = \{v_i, i = 1, 2, \ldots, n\}$, where v_i is the i-th constitutive frame in V. Given a certain feature descriptor D, videos can be mapped into a set of feature vectors, that is,

$$V = \{v_1, v_2, \ldots, v_n\} \xrightarrow{D} F = \{f_1, f_2, \ldots, f_n\}$$

It's expected that for each video frame, the projected points of it and its copies should be nearby in the feature space. Thereafter, we can conclude that the smaller the feature distance, the more likely they are copies. We model this copy relationship with a Gaussian model in the d-dimensional feature space as

$$p(v|q_i) = \frac{1}{\sqrt{(2\pi)^d |\sum|}} exp\left\{-\frac{1}{2}(f-f')^T \sum^{-1}(f-f')\right\}$$

where $\sum$ is a variance matrix. Clearly, with this simple location criterion, the false-frame copy-location rate depends on the feature discriminability. However, the false location likelihood of this approach can be reduced by taking more and more frames into consideration.

Given an input query video Q, which can also be represented by a set of frames $Q = \{q_i, i = 1, 2, \ldots, m\}$, the copy likelihood between Q and the video sequence fragment in video V

1070-986X/11/$26.00 © 2011 IEEE

Published by the IEEE Computer Society

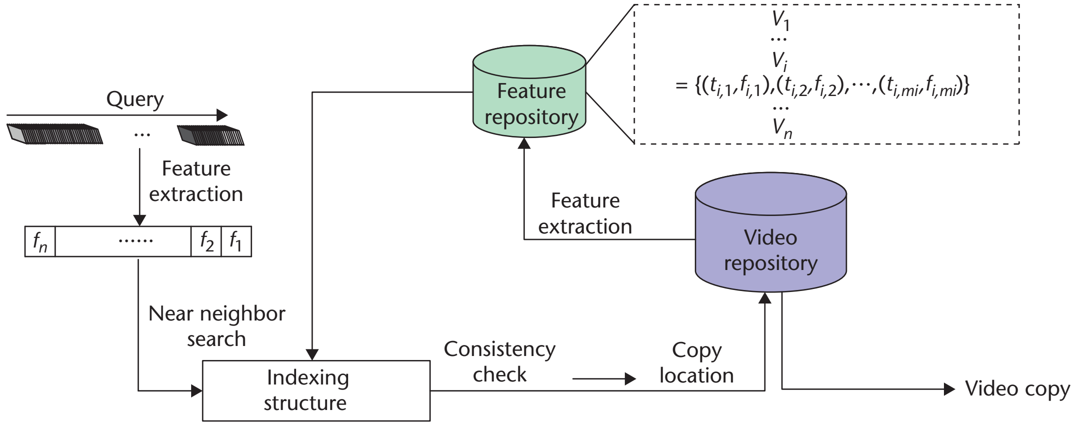


Figure 1. Schematic illustration of the proposed copy-location system.

starting from the k-th frame (with equal frames as in Q) can be formulated as

$$p(v_k, v_{k+1}, \ldots, v_{k+m-1}|Q) = p(v_k, v_{k+1}, \ldots, v_{k+m-1}|q_1, q_2, \ldots, q_m) \quad (1)$$

Then copy location is formulated as the following maximization problem:

$$\mathop{arg\ max}_k p(v_k, v_{k+1}, \ldots, v_{k+m-1}|q_1, q_2, \ldots, q_m) \quad (2)$$

Now the key problem is how to estimate $p(v_k, v_{k+1}, \ldots, v_{k+m-1}|q_1, q_2, \ldots, q_m)$. Here we assume that if two videos are copies, the corresponding frames should also be copies. By assuming that video frames distribute independently of each other, we can derive that

$$p(v_k, v_{k+1}, \ldots, v_{k+m-1}|q_1, q_2, \ldots, q_m) = \prod_{i=1}^{m} p(v_{k+i-1}|q_i) \quad (3)$$

where $p(v_j \mid q_i)$ is the probability of v_j being the copy of q_i.

With Equation 3, the problem in Equation 2 can be reformulated as

$$\mathop{arg\ max}_k p(v_k, v_{k+1}, \ldots, v_{k+m-1}|q_1, q_2, \ldots, q_m) = \mathop{arg\ max}_k \prod_{i=1}^{m} p(v_{k+i-1}|q_i) \quad (4)$$

Hence, in the feature space, Equation 4 can be reformulated as

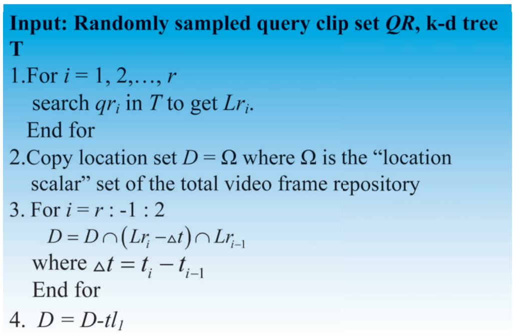
$$\mathop{arg\ max}_k \times \left\{ \prod_{i=1}^{m} exp\left(-\frac{1}{2}(f_{v_{k+i-1}} - f_{qi})^T \sum^{-1} (f_{v_{k+i-1}} - f_{qi}) \right) \right\} = \mathop{arg\ min}_k \left\{ \sum_{i=1}^{m} (f_{v_{k+i-1}} - f_{q_i})^T \sum^{-1} (f_{v_{k+i-1}} - f_{q_i}) \right\} \quad (5)$$

Indexing structure search

We can see that solving Equation 5 is equal to seeking the video sequence that is near to Q in the feature space. Exhaustive search of the minimum for Equation 5 might require a huge computational cost, especially for a large video repository. To address the scalability issue, we must build an efficient indexing for the video repository to facilitate near-neighbor search.

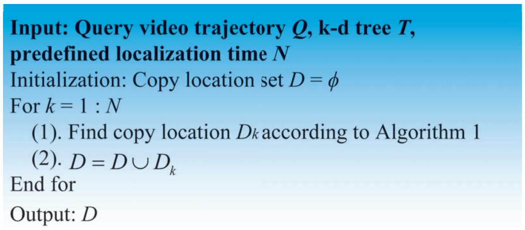
The schematic illustration of our copy-location system is shown in Figure 1. For all of the video frames in the repository, we can adopt nearest-neighbor search to locate the near-neighbor search space. Many indexing methods, such as tree structures and hashing algorithms, have been widely used for such tasks. Here we take the k-dimensional tree example for our framework. After k-dimensional tree construction, the video feature space is partitioned into a series of nonoverlapping bounding boxes. Each bounding box corresponds to a leaf node of the tree and can be regarded as maintaining an inverted list that indicates the video frame points belonging to this leaf node. During each iteration of the k-dimensional tree construction, for all of the points contained in the same node, we calculate variances of the points along each dimension and split the points by the medium according to the dimension, with maximum variance to generate the left and right child nodes. This process stops until a predefined tree height is reached.

Suppose that we build an h-level k-dimensional tree to index the video repository data, the leaf nodes of the k-dimensional tree are represented as $T = \{L_i, i = 1, 2, \ldots, 2^h\}$. L_i is a set of video frames belonging to the leaf node. Each frame p is represented by two variables $p = (t, f)$, where t is a unique location scalar that indicates



```
Input: Randomly sampled query clip set $QR$, k-d tree T
1.For $i = 1, 2,\ldots, r$
    search $qr_i$ in $T$ to get $Lr_i$.
  End for
2.Copy location set $D = \Omega$ where $\Omega$ is the “location
    scalar” set of the total video frame repository
3. For $i = r : -1 : 2$
      $D = D \cap (Lr_i - \Delta t) \cap Lr_{i-1}$
    where $\Delta t = t_i - t_{i-1}$
    End for
4.  $D = D\text{-}tl_1$
```

Figure 2. Algorithm for video copy location through frame-by-frame search pruning.



```
Input: Query video trajectory $Q$, k-d tree $T$,
predefined localization time $N$
Initialization: Copy location set $D = \phi$
For $k = 1 : N$
    (1). Find copy location $D_k$ according to Algorithm 1
    (2). $D = D \cup D_k$
End for
Output: $D$
```

Figure 3. Algorithm for video copy location by multitime search.

to which video p belongs and its exact position in the video, and f is the frame feature vector.

We search the query, frame by frame, according to its feature in the k-dimensional tree. For each query frame, a candidate set can be obtained. We simplify the probability calculation by a 0-1 measure according to whether the test-frame point belongs to the leaf node traversed by the query frame. For example, if leaf node L_i is traversed by query frame q_i, for each of the repository frame point v, copy probability is calculated as follows

$$p(v\,|\,q_i) = \begin{cases} 0 & if \quad v \notin L_i \\ 1 & if \quad v \in L_i \end{cases}$$

Hence Equation 1 can also be valued by a 0-1 measure. That is,

$$p(v_k, v_{k+1}, \ldots, v_{k+m-1}|q_1, q_2, \ldots, q_m) = \begin{cases} 0 \; if \; \exists v_{k+i-1}, \; v_{k+i-1} \notin L_i \\ 1 \; if \; \forall v_{k+i-1}, \; v_{k+i-1} \in L_i \end{cases}, \quad i = 1, 2, \ldots, m$$

The value “1” means the video sequence starting from k is a detected copy for Q while “0” indicates it is not. Rather than using all of the frame points, only part of the query frame points need to be randomly selected to retrieve the k-dimensional tree. The number of points selected is decided as long as it can provide sufficient evidence to eliminate any false copy. Satisfactory results can be obtained even if we select only a few frames during location. Here, the selected frame-point set is denoted as $QR = \{(t_1, qr_1), (t_2, qr_2), \ldots, (t_r, qr_r)\}$. For each frame (t_i, qr_i) in QR, we search in the k-dimensional tree. A candidate copy set can be obtained as Lr_i, with the elements of Lr_i being denoted as $lr_i = (tl_i, fl_i)$.

We fully exploit the video's inherent temporal characteristic and propose an approach to efficiently solve Equation 5. We assume there is no frame loss between the original video and its copy. According to our assumption mentioned previously, if $V = \{v_1, v_2, \ldots, v_n\}$ is a copy for Q and v_i is the frame copy of q_i, v_{i+j} should be the copy of q_{i+j}. This time correspondence is used to develop a copy-location algorithm that is described in the algorithm shown in Figure 2.

Here, $Lr_i - \Delta t$ means for all constitutive frames in Lr_i, their location scalar is subtracted by Δt. $(Lr_i - \Delta t) \cap Lr_{i-1}$ denotes finding the frames in Lr_{i-1} whose location scalar equals that of the elements in Lr_i subtracted by Δt. Owing to the frame transformations, it's possible that some query frame point cannot traverse the leaf node in which its copy is located. This inability to traverse the leaf node can lead to a miss in location according to our algorithm. Our strategy includes methods to reduce location misses. The first algorithm is shown in Figure 3. As the detection time increases, location misses can be reduced.

As for the second algorithm, we partition the feature into n parts, construct a k-dimensional tree set T with n trees, $T = \{T_1, T_2, \ldots, T_n\}$, where T_i is the tree built by the i-th feature segment. Given a query video frame trajectory point $qr_i \in QR$, its feature is partitioned with the same approach as we build the k-dimensional tree set, which is denoted as $qr_i = \{qr_{i,1}, qr_{i,2}, \ldots, qr_{i,n}\}$. For each segment, we search its feature subset $qr_{i,j}$. The leaf node traversed by $qr_{i,j}$ can be denoted as $Lr_{i,j}$. Integrating all of the subsets, the leaf node set of qr_i can be obtained by $Lr_i = Lr_{i,1} \cup Lr_{i,2} \cup \ldots \cup Lr_{i,n}$. The algorithm's location process is summarized in Figure 4.

Dataset and performance metric

We collected a total of 180 video clips (120 hours in time with more than 12 million frames) consisting of diverse content, such as news, interviews, and advertisements (the frame per second of the videos is 29.97). We used the principal component coefficients as an efficient feature descriptor together with the *k*-dimensional tree-indexing structure to test the performance of the proposed framework. The principal component coefficient feature extractor projects the scaled video frame sequence into the principal component space as a trajectory.[4] Specifically, for each frame v_i, we represent it by a *d*-dimensional feature vector x_i derived with principal component analysis (PCA) from the luminance of the scaled frame. That is,

$$x_i = A(S(v_i))$$

where *S* is the scaling operator, which rescales the luminance matrix of v_i to 12 × 16 then reshapes it to a vector. Then, PCA is conducted over the down-sampled frames to get the PCA transformation matrix *A*. This simple PCA-based feature extraction can meet the requirements for real-time feature extraction.

```
Input: Query video trajectory $Q$, k-d tree set $T$
together with the feature partition strategy
Initialization: Randomly select query point set $QR$,
Copy location set $D = \phi$
1. For $i = 1 : r$
    (1).Search the pre-constructed $T$ and get $Lr_{i,j}$
    for all $qr_{i,j}$ , $j = 1, 2, ..., n$
    (2).Calculate $Lr_i$ according to Eqn. (10)
End for
2. Calculate $D$ according to Algorithm 1
Output: $D$
```

Figure 4. Algorithm for video copy location by multitree search.

After scaling and projection, each video sequence is modeled as a temporal trajectory in the principal component space while each frame is represented as a point along this trajectory. Four typical examples are shown in Figure 5 (see http://trace.eas.asu.edu/yuv/index.html

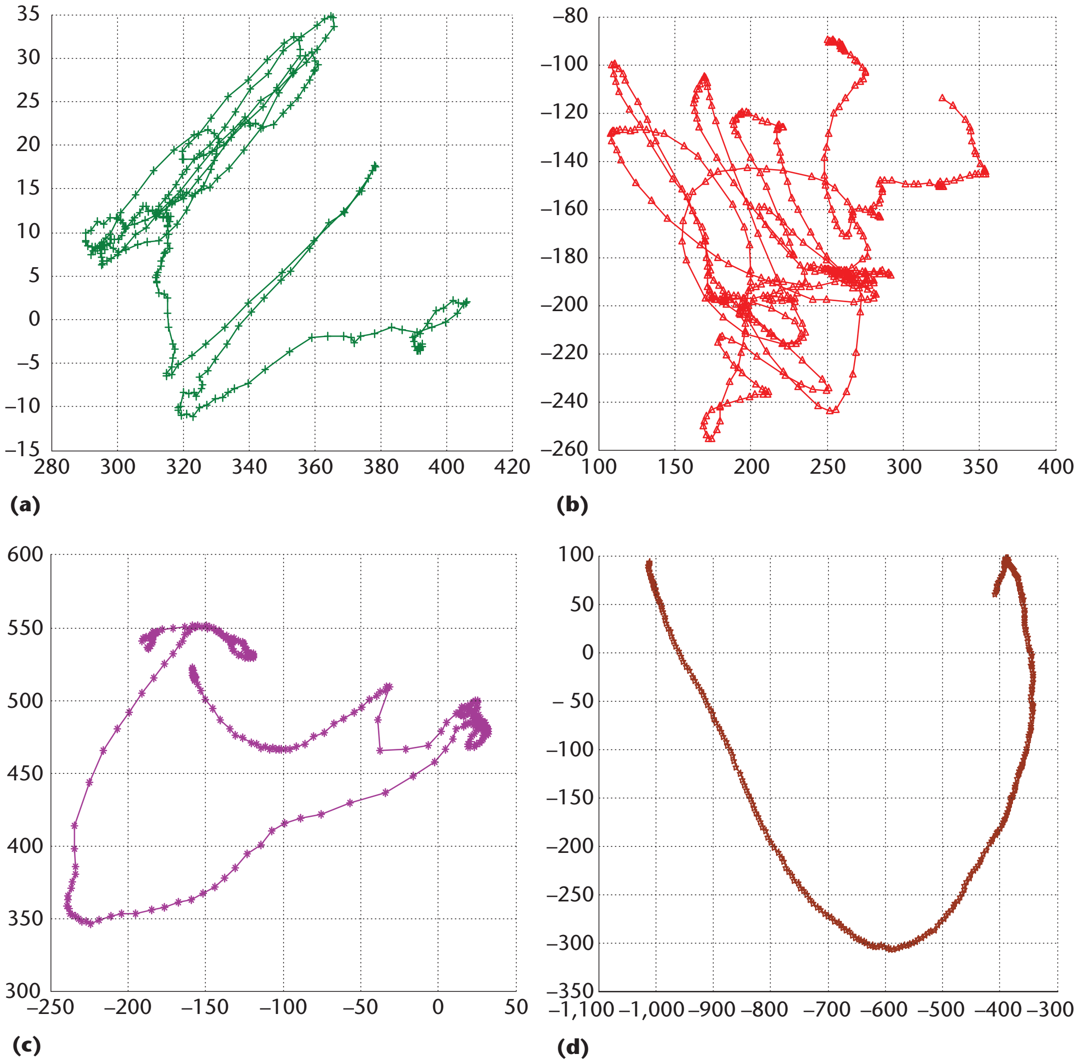


Figure 5. Projection result of the video sequence in the 2D principal component space: (a) Akiyo, (b) Claire, (c) mother-daughter, and (d) mobile.

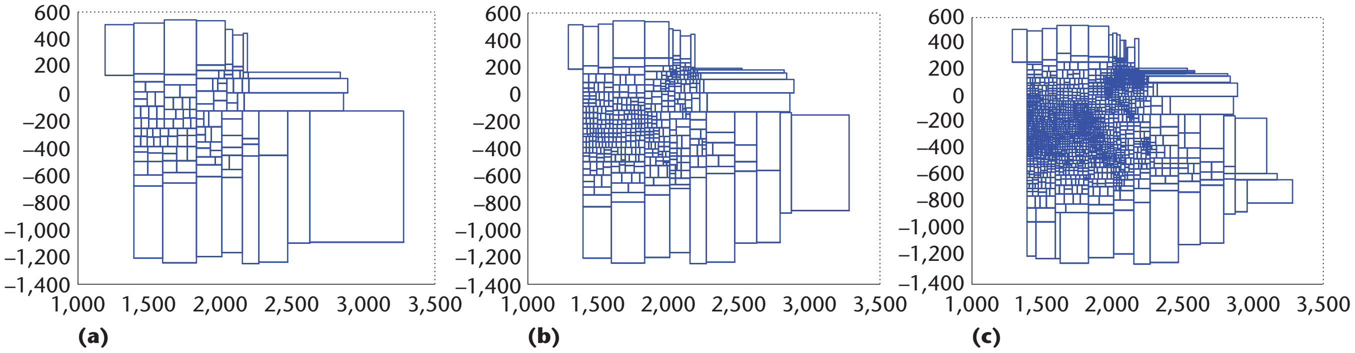


Figure 6. Bounding box of the k*-dimensional tree leaf nodes under different levels for the video repository: (a) 8-level, (b)10-level, and (c) 12-level.*

for more details). The first several principal-component coefficients are chosen as a feature representation for each frame. To test the robustness of the algorithm, we cut 200 query clips with time lengths ranging from 1 to 5 minutes randomly from the repository and for each query clip we simulate a series of transformations as follows:

- additive Gaussian noise measured by peak signal-to-noise ratio at 20 decibels simulated by adding the corresponding noise to the frame luminance component;
- Gaussian filtering with a 5 × 5 template;
- JPEG reencoding with the quality factor of 50;
- a 25 × 25 logo inserted at the top left of each frame; and
- a set of overlap transformations from two of the previous transformations, including JPEG reencoding at quality 50 plus logo, 5 × 5 Gaussian blurring plus logo, 5 × 5 Gaussian blurring plus JPEG reencoding at quality 50, and 20 decibel additive noise plus logo.

The *k*-dimensional tree height is set to achieve a tradeoff between search efficiency and robustness. We build 8-, 10-, and 12-level *k*-dimensional trees with the first two dimension PCA features for the total video repository and calculate the minimum bounding boxes. As shown in Figure 6, for a *k*-dimensional tree with lower level, each leaf node has a larger volume and contains more frame points, which can make it tolerate more complex transformations. But we need more time to eliminate the false copies. For higher level *k*-dimensional trees, there are fewer frame points in each leaf node, so we can get a smaller candidate set for each query frame. But this may lead to location misses because the query frame and its copy are less likely to be in the same leaf node.

Selecting fewer frame points for *k*-dimensional tree search definitely requires less computational cost. Furthermore, the location miss rate can be reduced because it provides a less strict constraint for copy judgment. However, reducing the miss rate can introduce higher false copy location. In other words, videos that are not copied may be judged as copies. In our experiments, we show the effect of frame number choice on the copy-location performance.

For the performance metric, we use F-measure, which takes both precision and recall into consideration. That is,

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

where

$$recall = \begin{cases} 0 & \forall l : |l - l_t| > L_{th} \\ 1 & \exists l : |l - l_t| < L_{th} \end{cases}$$

$$precision = \frac{|l \cap \{c | l_t - L_{th} \le c \le l_t + L_{th}\}|}{|l|}$$

and l indicates the location scalars of detected copy in the video database and l_t indicates the true location scalars of the copy in the video database. L_{th} is set to be 1,000 in our work. All of the experiments were conducted on a computer with a 2.93-GHz CPU and 4 Gbytes of RAM.

Experiment results

In the first experiment, we tested the performance of different frame numbers used for *k*-dimensional tree search on queries with eight or more transformations. We use the first six PCA coefficients as feature representations to build a nine-level *k*-dimensional tree-indexing structure. We randomly select 2, 4, 6, . . . ,

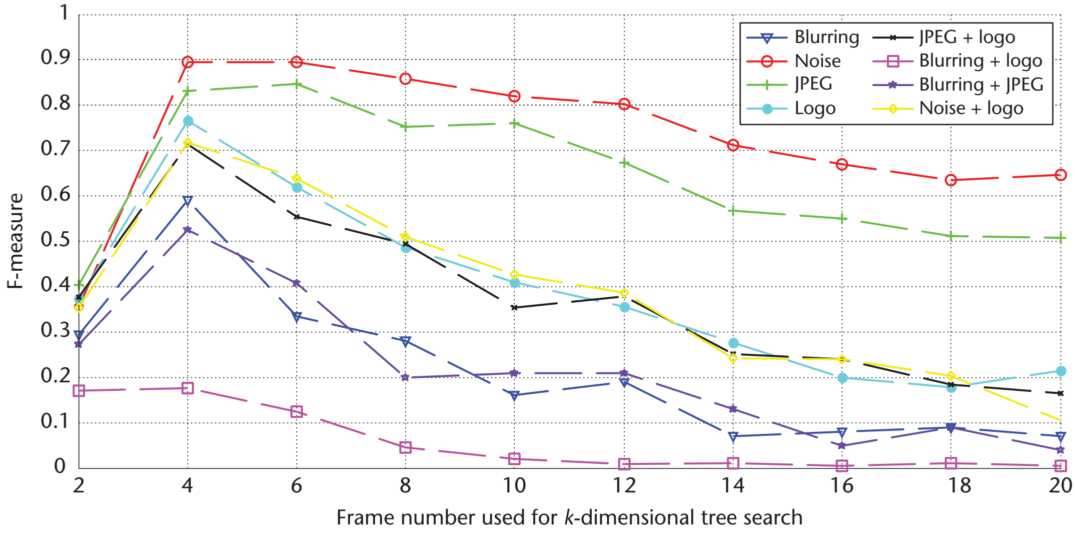


Figure 7. Location performance with different number of query frame.

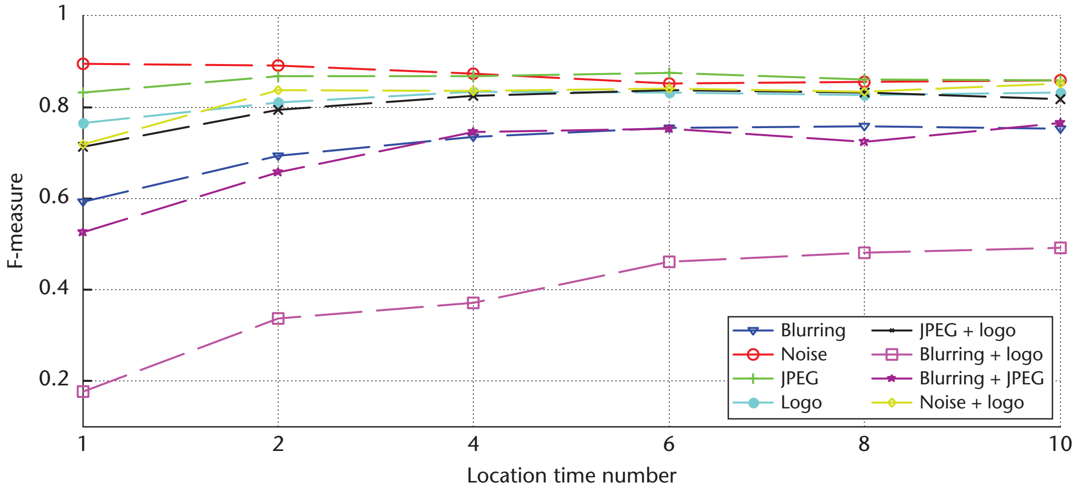


Figure 8. Location performance under different location times.

20 frames each time to locate the video copy. The F-measure with different numbers of frames used for copy location is shown in Figure 7.

We can see from the result that location performance is first determined by feature robustness. For the PCA feature we use here, it describes simple transformations much more effectively than that of more complex transformations. That is why the performance decreases as the transformations get more and more complex. Within the same transformation, selecting fewer frame points for *k*-dimensional tree search definitely can improve recall because it provides a less strict constraint for copy judgment. However, doing so can reduce precision. In other words, videos that are not copied might be judged as if they had been copied. That's why the performance of using only two frames is rather low for all of the eight transformations.

On the other hand, as the frame number increases, the algorithm will produce a more and more strict constraint for copy determination. As a result, copies tend to be missed even if only one query frame does not find the leaf node that contains its copy. We found that using four frames during location produces a better tradeoff between recall and precision. Therefore we fixed the frame number to four for the next set of experiments.

To test the performance of the proposed multitime location strategy, we used the same indexing structure and queries used in the first experiment, and we tested the location performance for queries with different transformations. For each query video, we selected the location time *N* to be 1, 2, 4, 6, 8, and 10, respectively. The results are presented in Figure 8. We can clearly see that for simple transformation such as additive Gaussian noise, one-time location

Table 1. Computational cost under different location times (in seconds).

Location time	1	2	4	6	8	10
Search	0.0003	0.0006	0.0011	0.0013	0.0017	0.0021
Pruning	0.0052	0.0091	0.0177	0.0258	0.0344	0.0455
Total	0.0055	0.0097	0.0188	0.0271	0.0361	0.0476

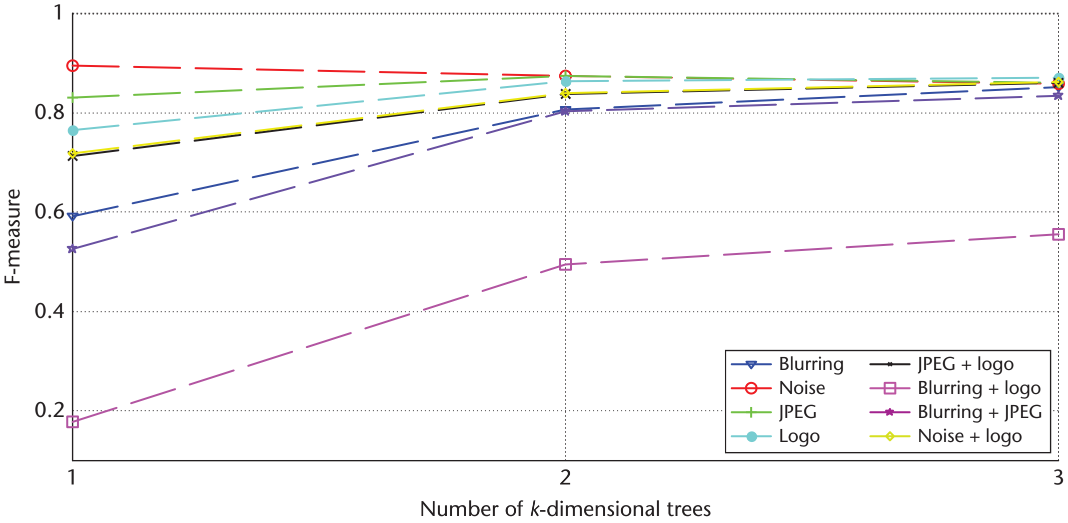


Figure 9. Location performance under different k*-dimensional tree numbers.*

Table 2. Computational cost under different k*-dimensional tree numbers (in seconds).*

k-dimensional tree number	1	2	3
Search	0.0005	0.0012	0.0021
Pruning	0.0055	0.0124	0.0173
Total	0.0060	0.0136	0.0194

can achieve satisfactory results, while increasing the location time may lead to slight performance degradation because of the precision loss.

For most of the transformations, multitime location outperforms one-time location. The location computational cost with different location times is shown in Table 1. It's obvious that the computational cost of the proposed algorithm consists of two parts, one is *k*-dimensional tree search and the other is likelihood pruning. The table shows the time cost of these two parts. In our approach, computational cost is independent of the query video length but only scales linearly with the number of selected frames for *k*-dimensional tree search and likelihood pruning. This is an advantage of our approach, and it makes our approach able to handle videos with different lengths in a sublinear time cost.

We analyzed the performance of the multitree strategy by selecting every sixth frame principal component coefficients successively to build a *k*-dimensional tree structure, using the same 200 queries with various transformations as test samples. Figure 9 illustrates the location performance under 1, 2, and 3 *k*-dimensional trees. The time cost is shown in Table 2. For most of the transformations, we found that multitree can improve the location performance over using just one tree because when we build more trees, more principal components are used and they can provide more detailed frame information for us to locate the copy.

Additionally, for both of the two strategies, parallel- and distributed-computing technologies can be employed to conduct multitime and multitree search at the same time. With this technology, we could reduce the location time further and reduce the computational cost of our algorithm for real applications.

Finally, we tested the performance of our algorithm on eliminating false copies. We generated

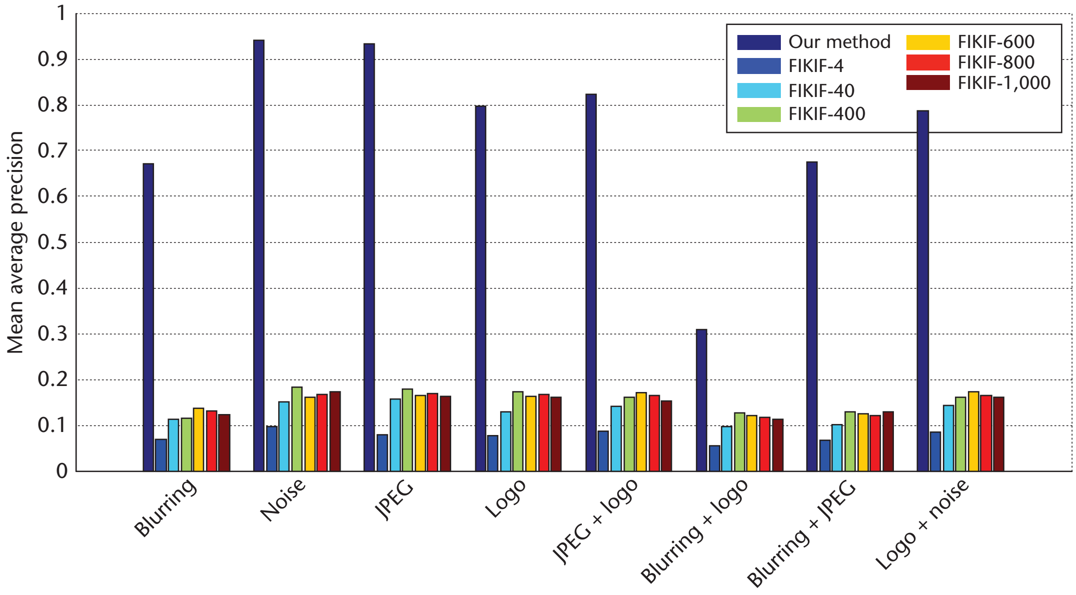


Figure 10. Mean average precision comparison between our method and fast intersection kernel and inverted file.

500 negative queries ranging from 1 to 5 minutes randomly from another 50-hour video dataset with diverse content and tested them with multitime and multitree strategies using the same settings as the positive query. More than 95 percent of the queries declared no matching under all parameter settings, showing that selecting four frames for each *k*-dimensional tree search, along with pruning, can provide enough information for false copy elimination.

Although the widely investigated sequence-matching approaches can achieve frame-level copy location, they suffer from scalability problems. Recently, some fast feature-matching strategies have been proposed for real-time, large-scale, near-duplicate detection. In particular, one recent feature-matching strategy is based on fast intersection kernel and inverted file (FIKIF).[5] In this strategy, under the *k*-dimensional tree-indexing structure scenario, each leaf node would be seen as a visual shingle. Given a query video $Q = \{q_i | i = 1, 2, \ldots, m\}$, the sorted randomly selected frame set $QS = \{qr_i \mid i = 1, 2, \ldots, r\}$. Assume t^{ω}_{QS} is the number of times the ω-th leaf node occurs in the search result using QS, $t^{\omega}_{V_i}$ is the number of frames contained in the ω-th leaf node for database video V_i. Video similarity can be calculated as

$$Sim(Q, V_i) = \frac{\sum_{\omega} min(t^{\omega}_{QS}, t^{\omega}_{V_i})}{M_q + M_i - \sum_{\omega} min(t^{\omega}_{QS}, t^{\omega}_{V_i})} \qquad (6)$$

M_q and M_i are the frame number of Q and V_i, respectively. The speed-up method of another project[5] is also used here to reduce the computational cost.

For our approach, after likelihood pruning, assume $N^{qr_j}_{V_i}$ is the number of frame points contained in the leaf node traversed by qr_j and belonging to V_i. Video similarity can be calculated as

$$Sim(Q, V_i) = \frac{\sum_j N^{qr_j}_{V_i}}{\sum_i \sum_j N^{qr_j}_{V_i}}$$

We evaluated the performance by mean average precision of the top five results, shown in Figure 10. For FIKIF, we tested this performance using 4, 40, 400, 600, 800, and 1,000 frames for copy search (which are denoted as FIKIF-4, FIKIF-40, FIKIF-400, FIKIF-600, FIKIF-800, FIKIF-1,000, respectively). In addition, with our approach, the average detection time is only 0.006 seconds. With FIKIF, it needs 0.002, 0.0127, 0.1015, 0.1376, 0.1632, and 0.1824 seconds respectively for copy detection. Although detection efficiency is comparable, in our experimental setting the performance of FIKIF is inferior to our method.

FIKIF can achieve encouraging performance in detecting video level duplicates, in which case the videos are comparable in length and overall content. It does not performs well in our experimental setting because Equation 6 cannot precisely estimate the similarity if the video copy is only a small part of the target video, which indicates the advantage and application of our proposed approach.

Related Work

Video similarity measurement has been an active area of investigation for many years. Most of the approaches use global feature descriptors to extract video frame features and turn video similarity measurement into a feature sequence-matching problem. For example, in one project, a video sequence is described by a symbol string, where similarity is measured through calculating the edit distance between strings, depending on the number and cost of edit operations needed to transform one string into the other.[1] In another project, a global feature that combines the spatial-temporal and the color range information is explored for query by video clip.[2] In related research, a video signature is discovered through an ordinal measure of resampled video frames.[3]

Kim et al. measure the similarity between two video sequences by calculating the length of the continuously matched substring separated by unmatched strings.[4] In this project, the binary string is generated by matching the features of two video sequences. Bertini et al. use edit distance to compare the video sequence feature taken from MPEG-7 features, such as the scalable color descriptor, color layout descriptor, and the edge histogram descriptor.[5] These methods can be used for video copy detection, but they are only effective on smaller datasets because the sequence-matching is still a computationally expensive process. A comprehensive survey on video copy detection can be found elsewhere.[6]

Recently, many local-feature approaches have been proposed. Local-feature extraction and matching is conducted between the extracted video key-frames in these methods. Zhang et al. proposed an attributed relational graph (ARG) model for keyframe representation. In this approach, the similarity between two ARGs is measured through statistical processes.[7] Scale-invariant feature transform and principal-component-analysis SIFT have been employed for keyframe matching, and many local-points-matching strategies have been proposed.[8,9] In addition, there have been several efforts to use the bag-of-visual-words model for duplicate detection.[10] Although these approaches have proven to be more robust to challenging frame transformations, matching and storing the high-dimensional frame features usually lead to huge cost for computation and storage, which hinders these methods from being used in large-scale, real-time settings.

Today, scalable video near-duplicate detection has gained more and more attention, especially in a Web environment. In 2010, online video was incorporated into Trecvid copy-detection tasks due to the value of multimedia applications (see http://www-nlpir.nist.gov/projects/tv2010/tv2010.html#ccd). Wu et al. proposed a hierarchical method that combines global features as a filter and local features for a fine search.[11] However, this method still cannot meet the efficiency requirements of real-time applications because of the large number of comparisons. Wu et al. use contextual information accompanied with Web video to reduce the

Conclusion

Our experiments on a large-scale dataset demonstrated the effectiveness of our approach. Our proposed scheme is flexible, enabling us to easily integrate other visual features and indexing structures into it. In the future, we plan to investigate different features and indexing methods, and we intend to apply our algorithm to other applications, such as online video search. **MM**

Acknowledgment

This work is partially supported by a faculty grant from Microsoft Research Asia and Hong Kong RGC grant 524509.

References

1. J.L.L. To et al., "Video Copy Detection on the Internet: The Challenges of Copyright and Multiplicity," *Proc. Int'l Conf. Multimedia & Expo,* IEEE Press, 2007.
2. S. Poullot, M. Crucianu, and O. Buisson, "Scalable Mining of Large Video Databases Using Copy Detection," *Proc. ACM Multimedia,* 2008.
3. J.L.L. To et al., "Video Copy Detection: A Comparative Study," *Proc. Int'l Conf. Image and Video Retrieval,* ACM Press, 2007.
4. Z. Li, L. Gao, and A.K. Katsaggelos, "Locally Embedded Linear Spaces for Efficient Video Shot Indexing and Retrieval," *Proc. Int'l Conf. Multimedia & Expo,* IEEE Press, 2006.
5. L.F. Shang et al., "Real-Time Large Scale Near-Duplicate Web Video Retrieval," *Proc. ACM Multimedia,* ACM Press, 2010.

Bo Liu is a research assistant in the department of computing at Hong Kong Polytechnic University. His research interests include computer vision and multimedia information retrieval. Liu has an MEng from the University of Science and Technology of China. Contact him at kfliubo@gmail.com.

comparison time and therefore significantly improve the detection efficiency.[12] This method relies heavily on the video's surrounding text information. Another model, called the Bounded Coordinate System (BCS), has been proposed for video feature representation.[13,14] In this method, video is segmented into clips and then summarized by the BCS. Shang et al. developed a compact spatiotemporal video feature descriptor and proposed a modified inverted file for fast intersection kernel computation for efficient near-duplicate retrieval.[15]

References

1. M.C. Lee, D.A. Adjeroh, and I. King, ''A Distance Measure for Video Sequences,'' *Computer Vision and Image Understanding,* vol. 75, nos. 1-2, 1999, pp. 25-45.
2. J.S. Yuan et al., ''Fast and Robust Short Video Clip Search for Copy Detection,'' *Proc. Pacific-Rim Conf. Multimedia,* Springer, 2004.
3. X.S. Hua, X. Chen, and H.J. Zhang, ''Robust Video Signature Based on Ordinal Measure,'' *Proc. Int'l Conf. Image Processing,* IEEE Press, 2006.
4. Y.T. Kim and T.S. Chua, ''Retrieval of News Video Using Video Sequence Matching,'' *Proc. Int'l Multi Media Modeling Conf.,* IEEE Press, 2005.
5. M. Bertini, A.D. Bimbo, and W. Nunziati, ''Video Clip Matching Using MPEG-7 Descriptors and Edit Distance,'' *Proc. Int'l Conf. Image and Video Retrieval,* Springer, 2006.
6. J.L.L. To et al., ''Video Copy Detection: A Comparative Study,'' *Proc. Int'l Conf. Image and Video Retrieval,* ACM Press, 2007.
7. D.Q. Zhang and S.F. Chang, ''Detecting Image Near-Duplicate by Stochastic Attributed Relational Graph Matching with Learning,'' *Proc. ACM Multimedia,* ACM Press, 2004.
8. X. Wu, W.L. Zhao, and C.W. Ngo, ''Near-Duplicate Keyframe Retrieval with Visual Keywords and Semantic Context,'' *Proc. Int'l Conf. Image and Video Retrieval,* ACM Press, 2007.
9. H.K. Tan et al., ''Accelerating Near-duplicate Video Matching by Combining Visual Similarity and Alignment Distortion,'' *Proc. ACM Multimedia,* ACM Press, 2008.
10. Y.G. Jiang and C.W. Ngo, ''Visual Word Proximity and Linguistics for Semantic Video Indexing and Near-Duplicate Retrieval,'' *Computer Vision and Image Understanding,* vol. 113, no. 3, 2009, pp. 405-414.
11. X. Wu, A.G. Hauptmann, and C.W. Ngo, ''Practical Elimination of Near-Duplicates from Web Video Search,'' *Proc. ACM Multimedia,* ACM Press, 2007.
12. X. Wu et al., ''Real-Time Near-Duplicate Elimination for Web Video Search with Content and Context,'' *IEEE Trans. Multimedia,* vol. 11, no. 2, 2009, pp. 196-207.
13. H.T. Shen et al., ''UQLIPS: A Real-Time Near-Duplicate Video Clip Detection System,'' *Proc. Int'l Conf. Very Large Data Bases,* ACM Press, 2007.
14. Z. Huang et al., ''Bounded Coordinate System Indexing for Real-Time Video Clip Search,'' *ACM Trans. Information Systems,* vol. 27, no. 3, 2009, pp. 17-33.
15. L.F. Shang et al., ''Real-Time Large Scale Near-Duplicate Web Video Retrieval,'' *Proc. ACM Multimedia,* ACM Press, 2010.

Zhu Li is senior staff researcher and leads the Media Analytics & Processing group at Core Networks Research, Huawei Tech. His research interests include audiovisual analytics and machine learning with its application in multimedia. Li has a PhD in electrical engineering from Northwestern University. Contact him at zhu.li@ ieee.org.

Linjun Yang is an associate researcher at Microsoft Research Asia. His research interests includes multimedia information retrieval, with a focus on multimedia search ranking and large-scale multimedia mining on the Web. Yang has an MS in computer science from Fudan University. Contact him at linjuny@microsoft.com.

Meng Wang is a research staff member at the National University of Singapore. His research interests include multimedia content analysis, search, mining, recommendation, and large-scale computing. Wang has a PhD in electronic engineering and information science from the University of Science and Technology of China. Contact him at eric.mengwang@gmail.com.

Xinmei Tian is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China. Her research interests include multimedia analysis and retrieval, computer vision, and machine learning. Tian has a PhD in electronic engineering and information science from the University of Science and Technology of China. Contact her at xinmeitian@gmail.com.