Video frame-matching algorithm using dynamic programming

Young-Yoon Lee,[a] Chang-Su Kim,[b] and Sang-Uk Lee[c]
[a]Samsung Electronics Company Ltd., Digital Media R&D Center, Suwon, Korea
[b]Korea University, School of Electrical Engineering, Seoul 136-713, Korea
[c]Seoul National University, School of Electrical Engineering and Computer Science, Seoul, Korea
E-mail: changsukim@korea.ac.kr

Abstract. *We propose a frame-matching algorithm for video sequences, when a video sequence is modified from its original through frame removal, insertion, shuffling, and data compression. The proposed matching algorithm defines an effective matching cost function and minimizes cost using dynamic programming. Experimental results show that the proposed algorithm provides a significantly lower probability of matching errors than the conventional algorithm.* © 2009 SPIE and IS&T. [DOI: 10.1117/1.3092367]

1 Introduction

Video matching techniques are motivated by various applications, including video retrieval, surveillance, and watermarking. In video retrieval,[1,2] frame matching is used to retrieve video clips, similar to a query clip, from a database. In video surveillance using multiple cameras,[3] it is necessary to establish the correspondences in time and space to combine asynchronous multiview sequences. In video watermarking,[4–6] temporal synchronization should be established between the original and watermarked videos before the extraction of frame-dependent watermarks.

Video matching techniques have different accuracy requirements, depending on their target applications. In video retrieval,[1,2] video matching need not be accurate at the frame level. It is acceptable that a retrieved frame does not match a query frame exactly, as long as they contain similar contents. In contrast, in video watermarking, more accurate matching is required, and two video sequences should be synchronized at the frame level, since the matching is a preprocessing step before the extraction of frame-dependent watermarks.[8] For instance, in Ref. 6, the pattern of skipped frames is employed as a watermark, and the accurate frame-level synchronization between an original video and its watermarked version is an essential step in watermark detection.

In watermarking applications, Delannay, Roover, and Macq[4] used an affine model to align video sequences. An affine model, however, cannot effectively describe irregular frame insertion and removal as well as frame shuffling, which are employed to attack watermarked video sequences. Cheng[5] proposed a temporal registration algorithm to correct temporal misalignment, which occurs in frame rate conversion or video capturing. The two-frame integration model was employed to represent temporally overlapping frames in the acquisition procedures.

In video watermarking, the original video can be modified by various temporal attacks, such as frame removal, insertion, and shuffling. To establish frame-level synchronization between original and modified sequences in video watermarking applications, we propose a dynamic programming algorithm that matches each frame in the modified sequence to the corresponding frame in the original sequence. Although Ref. 5 also employed dynamic programming to match two sequences, they used only frame dissimilarity in the matching. On the other hand, we introduce the notion of adaptive unmatched cost in addition to the matching cost to achieve more accurate matching. Moreover, the unmatched cost is used to deal with frame shuffling attacks, as well as frame removal and insertion attacks.

2 Matching Function and Matching Cost

Suppose that a video clip $\mathbf{X}=\{X_1,\ldots,X_{l_X}\}$, where X_j denotes the j'th frame, is modified from the original clip $\mathbf{Y}=\{Y_1,\ldots,Y_{l_Y}\}$. A frame X_j may be different from any frame Y_k in the original clip, since the clip $\mathbf{X}$ may be a compressed version of $\mathbf{Y}$. Moreover, some frames can be removed from or inserted into $\mathbf{X}$, and then the resulting frames can be shuffled. We assume that the number of removed frames is less than a threshold ν_R. Similarly, let ν_I and ν_S denote the maximum numbers of inserted frames and shuffled frames, respectively.

We say that a frame X_j in $\mathbf{X}$ matches a frame Y_k in $\mathbf{Y}$ if X_j originates from Y_k. On the other hand, X_j is called an unmatched frame if X_j does not match any frame in Y. Then, the temporal modification of a video can be represented by a function on the space of frame indices. Specifically, we define the matching function λ as

$$\lambda(j)=\begin{cases} k, & \text{if } X_j \text{ matches } Y_k \\ 0, & \text{if } X_j \text{ is an unmatched frame} \end{cases}, \qquad (1)$$

where j belongs to the frame index set $\{1,2,\ldots,l_X\}$ of $\mathbf{X}$. For example, Table 1 shows a matching function, when $l_X=9$ and $l_Y=10$. In this case, $\mathbf{X}$ is modified from $\mathbf{Y}$ by removing Y_2 and Y_6, inserting new frames before Y_7, and then reversing the order of Y_8 and Y_9.

Let $d_m(j,k)$ denote the matching cost between two frames X_j and Y_k, which measures the dissimilarity between X_j and Y_k. The mean squared error (MSE) is employed as the matching cost.

3 Proposed Matching Algorithm

Given a video clip $\mathbf{X}$ and its original $\mathbf{Y}$, we attempt to find a matching function $\lambda(j)$, such that $Y_{\lambda(j)}$ in $\mathbf{Y}$ is identical or similar to X_j in $\mathbf{X}$ for each $1\leq j\leq l_X$. The simplest matching can be achieved by the local minimization of matching costs (LMMC) via

$$\lambda_{\mathrm{LMMC}}(j)=\arg\min_{1\leq k\leq l_Y} d_m(j,k) \quad \text{for } 1\leq j\leq l_X. \qquad (2)$$

To recover from frame insertion attacks, X_j can be declared to be an unmatched frame if the minimum cost

Paper 08037LRR received Mar. 10, 2008; revised manuscript received Jan. 7, 2009; accepted for publication Jan. 21, 2009; published online Mar. 6, 2009.

1017-9909/2009/18(1)/010504/3/$25.00 © 2009 SPIE and IS&T.

Table 1 A matching function $\lambda(j)$ that describes the temporal alignment between two sequences.

j	1	2	3	4	5	6	7	8	9
$\lambda(j)$	1	3	4	5	0	7	9	8	10

$\min_{1\leq k\leq l_Y} d_m(j,k)$ is larger than a threshold. However, in this LMMC approach, matching functions may not be one to one, and estimation results are very sensitive to the threshold.

We propose a globally optimal matching algorithm that minimizes the total matching cost subject to the one-to-one matching constraint. First, we assume that the original video clip is temporally modified by frame removal and insertion attacks only, and find the best matching function λ^* under this monotonicity assumption. Later, in the second phase, we detect frame shuffling attacks by further matching the unmatched frames of λ^* without imposing the monotonicity constraint.

Assuming frame removal and insertion attacks only, the matching function can be estimated by minimizing the sum of the costs of matched frames and unmatched frames, given by

$$\lambda^* = \arg\min_{\lambda\in\Lambda}\left\{\sum_{j;\lambda(j)>0} d_m[j,\lambda(j)] + \sum_{l;\lambda(l)=0} d_u(l)\right\}, \tag{3}$$

where $\Lambda=\{\lambda\,|\,\lambda(j)<\lambda(j') \text{ if } j<j',\lambda(j)>0,\lambda(j')>0\}$. Note that a function λ in Λ is monotonically increasing on the reduced domain $\{j:1\leq j\leq l_X \text{ and } \lambda(j)>0\}$, which excludes the unmatched frame indices. The unmatched cost of the l'th frame is defined as

$$d_u(l) = \max_{j\in[l-\delta,l+\delta];\lambda(j)>0}\ \min_{1\leq k\leq l_Y} d_m(j,k). \tag{4}$$

Notice that the unmatched cost of the l'th frame is adaptively determined by the matched costs of the neighboring frames inside a temporal window $[l-\delta,l+\delta]$, where δ denotes the window size. In general, a large $d_u(l)$ indicates that the neighboring frames are heavily compressed or they contain complex textures and motions. Thus, a large $d_u(l)$ makes it easier to classify the l'th frame as a matched frame.

In Eq. (3), when i frames, $\max\{l_X-l_Y,0\}\leq i\leq \nu_I$, are inserted into $\mathbf{X}$, there are $\binom{l_X}{i}$ possible choices of unmatched frames and $\binom{l_Y}{l_X-i}$ possible choices of matched frames. Therefore, the complexity of the exhaustive minimization in Eq. (3) is proportional to $\sum_{i=\max\{0,l_X-l_Y\}}^{\nu_I}\binom{l_X}{i}\binom{l_Y}{l_X-i}$, which is too demanding in most applications.

To reduce the complexity, we minimize the total cost function in Eq. (3) based on dynamic programming. The dynamic programming method in this work is similar to that for computing the Levenshtein distance, also called the edit distance, between two text strings.[10,11] The edit distance counts the minimum number of letter substitutions, insertions, or removals to convert a text string to another string. Similarly, the proposed algorithm matches a video sequence to another sequence by considering frame insertions and removals. However, whereas the distance between two letters is binary (identical or not), the distance between two frames should represent the similarity of those two frames. Thus, as compared with letter insertions or removals, it is more difficult to identify frame insertions or removals. To overcome this difficulty, the proposed algorithm defines the matching cost as MSE and the unmatched cost in Eq. (4) and minimizes the total cost in Eq. (3) to achieve reliable video matching.

Suppose that the first j frames of $\mathbf{X}$ are obtained by removing r frames from the first k frames of $\mathbf{Y}$ and then inserting i new frames. Note that $k=j-i+r$, since $(j-i)$ frames in $\mathbf{X}$ match $(k-r)$ frames in $\mathbf{Y}$. Let $s(j;i,r)$ denote the minimum sum of the matching costs for the first j frames in $\mathbf{X}$ when i frames are inserted and r frames are removed.

We compute $s(j;i,r)$ recursively. First, we initialize $s(j;i,r)=\infty$ if $j<1$, $i<0$, or $r<0$ with the exception $s(0;0,0)=0$. Then, $s(j;i,r)$ can be obtained by finding the minimum among three cases

$$s(j;i,r) = \min\left\{\begin{array}{l} s(j-1;i,r)+d_m(j,k), \\ s(j-1;i-1,r)+d_u(j), \\ s(j;i,r-1) \end{array}\right\}, \tag{5}$$

where $k=j-i+r$. When X_j matches Y_k, the matching cost $d_m(j,k)$ is added to $s(j-1;i,r)$, which is the first term in Eq. (5). When X_j is an inserted frame, the unmatched cost $d_u(j)$ is added to $s(j-1;i-1,r)$. If Y_k is a removed frame, $s(j;i,r-1)$ remains unchanged. In this way, we compute all $s(j;i,r)$ inductively. Finally, the minimum sum of costs for the whole sequence is given by $\min_{\max\{0,l_X-l_Y\}\leq i\leq \nu_I} s(l_X;i,l_Y-l_X+i)$. While computing the partial costs in Eq. (5), we also record the minimum conditions, which we can trace back to get the matching function in Eq. (3).

Next, let us consider frame shuffling attacks, in addition to frame removal and insertion attacks. In Eqs. (3) and (5), we impose the monotonicity constraint that the matching function for matched frames is monotonically increasing, and shuffled frames hence can be classified as unmatched frames. Therefore, when X_j is classified as unmatched, it can be further matched to a frame in $\mathbf{Y}$, which is not matched by the function in Eq. (3). More specifically, $\lambda^*(j)$ is refined by

$$\lambda^*(j) = \arg\min_{k\in U} d_m(j,k) \quad \text{if } \min_{k\in U} d_m(j,k) < d_u(j), \tag{6}$$

where U denotes the indices of frames in $\mathbf{Y}$ that are not matched by Eq. (3). Notice that we reduce the total matching cost by changing the category of X_j from an unmatched frame to a shuffled frame when its matching cost is smaller than the unmatched cost.

Table 2 Comparison of matching error probabilities when a video is modified from its original through frame removal (R), insertion (I), shuffling (S), and data compression attacks.

H. 264/AVC		*Temporal attacks* (%)			
Algorithm	QP	R	I	$R+I$	$R+I+S$
Cheng[5]	20	0.00	0.46	0.66	N/A
	35	0.03	1.75	2.12	N/A
LMMC	20	0.27	0.51	0.59	0.59
	35	1.71	3.43	4.11	4.12
Proposed	20	0.00	0.00	0.00	0.00
	35	0.00	0.09	0.10	0.17

4 Experimental Results

The performance of the proposed algorithm is evaluated using five common intermediate format (CIF) sequences Foreman, Paris, Mobile, Tempete, and Carphone, each of which consists of 100 frames. We also use 33,220 video clips, selected from 20 Korean movies.[12] Each clip consists of 100 frames of resolution 360×240, and has a frame rate of either 24 or 30 frames per second. Thus, 3,322,500 frames are used in total.

To match a video clip with another clip, we minimize the total cost in Eq. (3). It can be shown that the dynamic programming method requires also $O(l_X l_Y)$ recursion steps. We use a personal computer with an Intel Pentium D 3-GHz processor for simulations. To match a video clip of 100 frames to another clip, the proposed algorithm requires about 5.32 ms for the dynamic programming method.

Table 2 shows the matching error probabilities. A matching error probability is defined as the rate of an estimated matching function being different from the true matching function. The sequences are compressed by the H.264/AVC standard with QP 20 and 35, which yield about 42.0 and 30.8 dB peak signal-to-noise ratios (PSNRs) on average, respectively. The sequences then go through frame removal attacks (R), frame insertion attacks (I), and frame shuffling attacks (S). The numbers of removed and inserted frames are randomly selected from 1 to 10 and from 1 to 3 ($\nu_R = 10$ and $\nu_I = 3$), respectively. An inserted frame is constructed by averaging two adjacent frames in the original sequences. For shuffling attacks, the number of pairs of adjacent frames is randomly selected from 1 to 3 ($\nu_S = 3$), and each pair of frames swaps their places.

We compare the performance of the proposed algorithm with those of Cheng's algorithm[5] and the LMMC approach. In Ref. 5, frame shuffling attacks are not considered, and the matching fails under these attacks. Since the weights for the two-frame integration model are overparameterized for insertion attacks, matching errors occur even when QP is as low as 20. The LMMC approach is also vulnerable to insertion attacks, since it simply searches the best matching frame for each individual frame. On the other hand, the proposed algorithm provides much better performance by globally minimizing the total matching cost. For example, when QP is 35 and frame removal and insertion attacks are combined ($R+I$), the matching error probability of the proposed algorithm ($=0.10\%$) is about 41 times lower than that of LMMC ($=4.11\%$) and about 21 times lower than that of Cheng's algorithm ($=2.12\%$). As QP increases, the qualities of modified videos degrade and the matching error probabilities get higher. However, the proposed algorithm provides significantly better performance than the conventional algorithm, even in these severe conditions.

5 Conclusion

We propose a temporal alignment algorithm between two video sequences, when a video is modified from the other through frame removal, insertion, and shuffling attacks as well as data compression attacks. We define a cost function for global matching errors and develop a dynamic programming algorithm to minimize cost efficiently. Experimental results show that the proposed algorithm provides a significantly lower probability of matching errors than the conventional algorithm.

Acknowledgment

This work was supported partly by the Ministry of Knowledge Economy, Korea, under the Information Technology Research Center support program supervised by the Institute of Information Technology Advancement (grant number IITA-2008-C1090-0801-0017) and partly by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (number R01-2008-000-20292-0).

References

1. Z. Li, L. Gao, and A. K. Katsaggelos, "Locally embedded linear subspaces for efficient video indexing and retrieval," *Proc. IEEE Intl. Conf. Multimedia Expo*, pp. 1765–1768 (2006).
2. J. Yuan, W. Wang, J. Meng, Y. Wu, and D. Li, "Mining repetitive clips through finding continuous paths," *Proc. ACM Intl. Conf. Multimedia*, pp. 289–292 (2007).
3. E. Grimson, P. Viola, O. Faugeras, T. Lonzano-Perez, T. Poggio, and S. Teller, "A forest of sensors," *Proc. DARPA Image Understanding Workshop* **1**, 45–50 (1997).
4. D. Delannay, C. de Roover, and B. Macq, "Temporal alignment of video sequences for water-marking systems," *Proc. SPIE* **5020**, 481–492 (2003).
5. H. Cheng, "Temporal registration of video sequences," *Proc. ICASSP* 3, 489–492 (2003).
6. Y. Y. Lee, C. S. Kim, and S. U. Lee, "Video fingerprinting based on frame skipping," *Proc. ICIP* 1, 2305–2308 (2006).
8. E. T. Lin and E. J. Delp, "Temporal synchronization in video watermarking," *IEEE Trans. Signal Process.* **52**, 3007–3022 (2004).
10. V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Sov. Phys. Dokl.* **10**, 707–710 (1966).
11. G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.* **33**(1), 31–88 (2001).
12. Y. Y. Lee, "Temporal feature modulation for video watermarking," PhD Thesis, Seoul National University, Korea (2008).