Logical DP Matching for Detecting Similar Subsequence

Seiichi Uchida[1], Akihiro Mori[2] Ryo Kurazume[1],
Rin-ichiro Taniguchi[1], and Tsutomu Hasegawa[1]

[1] Kyushu University, Fukuoka, 819-0395, Japan,
uchida@is.kyushu-u.ac.jp,
WWW home page: http://human.is.kyushu-u.ac.jp/
[2] Toshiba Corporation, Tokyo, 198-8710, Japan

Abstract. A logical dynamic programming (DP) matching algorithm is proposed for extracting similar subpatterns from two sequential patterns. In the proposed algorithm, local similarity between two patterns is measured by a logical function, called support. The DP matching with the support can extract all similar subpatterns simultaneously while compensating nonlinear fluctuation. The performance of the proposed algorithm was evaluated qualitatively and quantitatively via an experiment of extracting motion primitives, i.e., common subpatterns in gesture patterns of different classes.

1 Introduction

Detection of similar subsequences between two sequential patterns is one of fundamental problems for sequential pattern processing. Especially, this problem is very important when extracting frequent subsequences. For example, frequent subsequences of gesture patterns, called motion primitives, are widely used for analyzing human activities. In addition, frequent subsequences of biological sequences (such as DNA sequence and protein sequence), called motives, also plays quite important role in genome science [1, 2].

In this paper, a new algorithm, called logical dynamic programming (DP) matching, is proposed for detecting similar subsequences. The proposed algorithm employs a logical function $s(i, j)$, called support, to evaluate similarity between the ith frame of $A = a_1, \ldots, a_i, \ldots, a_I$ and the jth frame of $B = b_1, \ldots, b_j, \ldots, b_J$. Specifically, if a pair of frames a_i and b_j are similar, $s(i, j) =$ `true`. Otherwise, $s(i, j) =$ `false`. Similar subsequences between A and B can be determined as sets of consecutive frame pairs having `true` supports. The use of the support allows simultaneous detection of all similar subsequences while compensating nonlinear fluctuations in the subsequences.

The remaining part of this paper is organized as follows. After reviewing related work in in Section 2, the logical DP matching algorithm is described in Section 3. In addition, the logical DP matching algorithm is applied to actual gesture patterns for evaluating its performance on the detection of similar

subsequences among them. In Section 4, the performance of the proposed algorithm is further evaluated quantitatively via an experiment of extracting motion primitives from gesture patterns.

2 Related Work

The algorithms for detecting similar subsequences are classified into two groups. The first group is based on "rigid comparison" between two sequential patterns. In the algorithms of this group, sequential patterns are firstly described by rough representations, such as sequences of a limited number of symbols, by clustering or other quantization technique, and then identical subsequences are detected. The rough representation is necessary for compensating fluctuations between two sequences. Tanaka et al. [3] transformed a gesture sequence into a symbol sequence for extracting motion primitives under an MDL criterion. A similar algorithm can be found in [4]. A problem of this group is that the rough representation will lose exact boundaries of similar subsequences. In fact, a refinement step is introduced in [4] for finding exact boundaries.

The second group is based on "elastic comparison", where two sequential patterns are compared under an optimized nonlinear matching to compensate fluctuations. DP matching algorithm, or dynamic time warping (DTW) algorithm, will be the most popular algorithm in this group. DP matching possesses several useful properties, such as (i) the global optimality of its matching result, (ii) computational feasibility, (iii) high versatility, etc., and thus has been used various sequential pattern processing tasks, such as speech recognition [5] and character recognition [6] and gnome science [1, 2].

The difference between the proposed and the conventional DP matching algorithms is the definition of frame (dis)similarity. Specifically, the proposed algorithm employs the logical support function $s(i, j)$, whereas the conventional algorithm has been employed Euclidean distance between feature vectors of the ith and jth frames. This modification brings the following novel abilities to the DP matching:

- The proposed algorithm can detect *strictly similar* subsequences. This means that all the paired frames in the similar subsequences are similar. In contrast, similar subsequences by the conventional DP matching algorithms are *averagely similar*; that is, several paired frames can be dissimilar if other paired frames are very similar. (This is because of the conventional algorithm is based on accumulated Euclidean distance.)
- The proposed algorithm can detect all similar subsequences *simultaneously*.

3 Detecting Similar Subsequence by Logical DP Matching

This section describes the logical DP matching algorithm and its application to the simultaneous detection of similar subsequences. The performance of the algorithm is also discussed via a detection experiment.

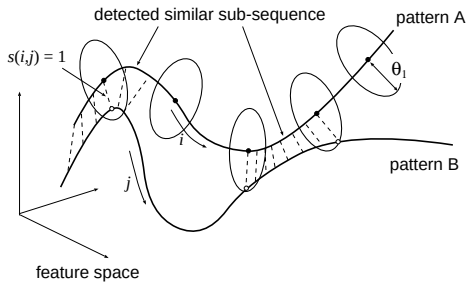

Fig. 1. Similar subsequence detection using support $s(i, j)$.

3.1 Logical DP Matching

Support and supported path Each sequential pattern can be represented as a trajectory in a feature space, as shown in **Fig. 1**. Similar subsequences between two sequential patterns will be observed as regions where two trajectories are close to each other. This closeness is evaluated by the following logical function, called *support*:

$$s(i,j) = \begin{cases} \texttt{true} & \text{if } d(i,j) < \theta_1 \\ \texttt{false} & \text{otherwise,} \end{cases} \tag{1}$$

where θ_1 is a positive constant and $d(i, j)$ is a distance between the ith and the jth frames. In the followings, we will use Euclidean distance $d(i, j) = \|a_i - b_j\|$, unless otherwise noted. If two trajectories are close at the ith and the jth frames (namely, if a_i and b_j are similar), the support $s(i, j) = \texttt{true}$. Otherwise, $s(i, j) = \texttt{false}$. The dashed lines in **Fig. 1** indicates frame pairs where two trajectories are close.

Consider an i-j plane of **Fig. 2** and assume that the support $s(i, j)$ has been already calculated at each (i, j) node according to (1). Since similar subsequences between A and B can be determined as sets of consecutive frame pairs having `true` supports, the problem of finding similar subsequences is treated as the problem of finding paths connecting nodes with `true` supports. Hereafter, this path is called a *supported path*. A supported path from (i_s, j_s) to (i_e, j_e) indicates that subsequences $a_{i_s}, \ldots, a_{i_e}$ and $b_{j_s}, \ldots, b_{j_e}$ are similar throughout, that is, strictly similar.

DP recursion All the supported paths can be found efficiently and simultaneously by a DP-based algorithm. In the algorithm, the following equation, so-called DP recursion, is calculated at each j from $i = 1$ to I:

$$g(i,j) = s(i,j) \wedge \left(\bigvee_{k=1}^{3} g_k \right) \tag{2}$$

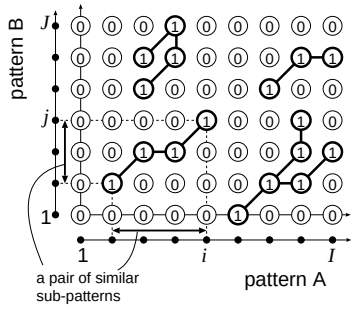

Fig. 2. Representation of similar subsequences by supported paths. Here, "1" stands for `true` and "0" for `false`.

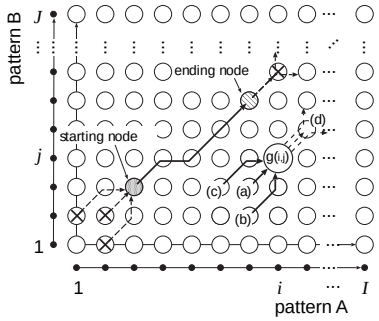

Fig. 3. DP recursion and staring/ending node.

where

$$\begin{cases} g_1 = g(i-1, j-1) \\ g_2 = s(i, j-1) \wedge g(i-1, j-2) \\ g_3 = s(i-1, j) \wedge g(i-2, j-1) \end{cases}$$

and $\vee$ and $\wedge$ denote logical OR and AND operations, respectively. The function $g(i,j)$ is a logical function taking `true` or `false` and indicates whether there is at least one supported path arriving at the node (i,j) or not. The nodes (a), (b), and (c) of **Fig. 3** correspond to the nodes with g_1, g_2, and g_3, respectively. The above recursion represents the trial to extend the supported paths to the nodes (a), (b), and (c) by connecting the node (i,j).

The starting and the ending nodes of each supported path are also detected while calculating of the DP recursion (2). The node (i,j) is marked as a starting node iff $g_1 = g_2 = g_3 =$ `false` and $s(i,j) =$ `true`. Similarly, the node (i,j) is marked as an ending node iff $g(i,j) =$ `true` and $s(i+1, j+1) =$ `false`[3].

Backtracking from every ending node provides all the supported paths simultaneously. The supported paths shorter than θ_2 are eliminated as noise. As shown

[3] Any supported path passing (i,j) will also pass $(i+1, j+1)$. Thus, if $s(i+1, j+1) =$ `false`, no path can be extended from (i,j) and therefore (i,j) is an ending node.

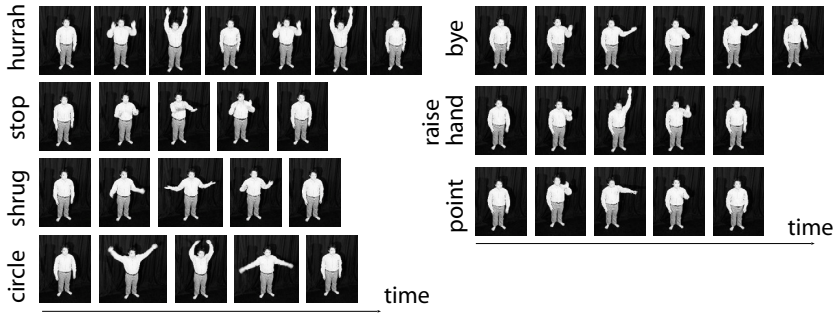

Fig. 4. Snapshot of assumed gestures.

in **Fig. 2**, supported paths often branch and share the same starting/ending node. In the following experiment, those branched supported paths are to be shrunk into a non-branched one by removing less reliable paths. The reliability of each branch is evaluated by the accumulated distance of $d(i, j)$ along the branch. A branch with a higher accumulated distance is considered as a less reliable branch. After this removal, only one supported path will start from each starting node and end in each ending node.

The computational complexity of the above algorithm is $O(IJ)$. This is the same complexity as the conventional DP matching algorithms.

3.2 Experiment of detecting similar subpatterns between gesture patterns

Experimental setup The performance of the proposed algorithm has been evaluated via an experiment of detecting similar subsequences of gesture patterns. Gestures of 18 classes were used in the experiment. **Fig. 4** shows snapshots of 7 gestures of them. Each gesture was performed 6 times by one person and thus 108 gesture patterns were prepared. The average frame length was 89. A six-dimensional feature vector was used for representing the performer's posture at each frame. Specifically, the six elements of the vector were the three-dimensional positions of both hands (relative to the head position), which were obtained by stereo measurement with two IEEE1394 cameras.

The two parameters θ_1 and θ_2 were fixed at 200 and 8, respectively. The condition $\theta_1 = 200$ implies that the support $s(i, j)$ becomes `true` if the difference of two postures, i.e., $\|a_i - b_j\|$, is less than 20cm. The condition $\theta_2 = 8$ implies that similar subsequences whose length is less than 500ms are eliminated as noise.

Detection result of similar subsequences **Fig. 5** shows five detection results. The images at the leftmost column show $d(i, j)$ on the i–j plane and the images at the middle column show the support $s(i, j)$. The images at the rightmost column show the supported paths detected by the proposed algorithm.

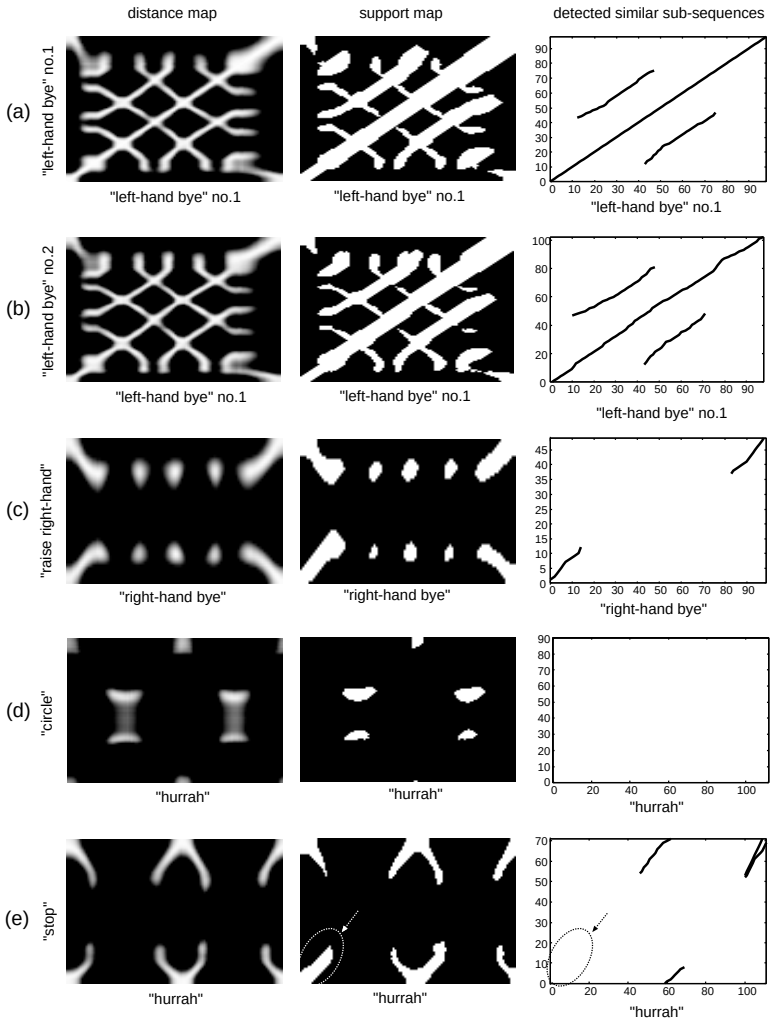

Fig. 5. Detection result of similar subsequences.

Fig. 5 (a) is the result when an identical "(left-hand) bye" pattern was used as both A and B. Since A and B were identical, a long diagonal straight supported path was obtained. The other, more meaningful, two supported paths were obtained because a left-to-right hand motion is repeated twice in "bye" and the first motion and the second motion were correctly detected as similar subsequences.

Fig. 5 (b) is the result when two different "bye" patterns were used for A and B. The repeated hand motions were correctly detected like (a).

Fig. 5 (c) is the result when "bye" and "raise (right-)hand" were used for A and B, respectively. Those gestures share two common motions. One is the

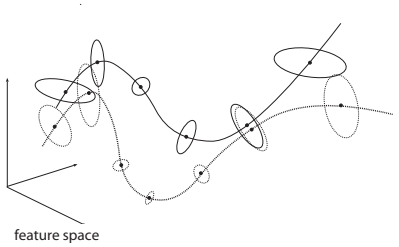

Fig. 6. Statistical extension of support.

beginning motion that the right hand is raised to shoulder height. The other is the ending motion that the right hand is lowered from shoulder height. Those common motions were successfully detected as the two supported paths around the beginning and the end of the gestures.

Fig. 5 (d) is the result when "hurrah" and "circle" were used for A and B, respectively. Those gestures may seem similar but, actually, do not have any common motion, as shown in **Fig. 4**. The proposed algorithm could avoid to detect any false positive, i.e., spurious supported path.

Fig. 5 (e) is a failure result. Two different gestures "hurrah" and "stop" were used and their common motion ("raising both hands to shoulder height") around their beginning part was not detected. This failure results indicates that the beginning parts of those gestures fluctuate more largely than other parts.

One possible remedy to deal with this non-uniform fluctuation range is a statistical extension of $d(i,j)$. Specifically, as illustrated in **Fig. 6**, if $d(i,j)$ is evaluated according to the degree of unstability, the region where $s(i,j) = \texttt{true}$ will be defined adaptively to the fluctuation range and thus it will be possible to improve the detection accuracy. In fact, the above missed common motion was detected correctly in the experimental result using the Bhattacharyya distance as $d(i,j)$. (The detail of this experiment will be presented elsewhere.)

4 Application to Extraction of Motion Primitives

The logical DP algorithm can be applied to the extraction of motion primitives, which are subpatterns constituting gesture patterns. In this section, the performance of the logical DP matching was evaluated qualitatively and quantitatively via an experiment of extracting motion primitives.

4.1 Extracting motion primitives by logical DP matching

Gesture patterns are often decomposed as a sequence of motion primitives for analyzing how gesture patterns are comprised and share common motions. The extraction of the motion primitives can be done by the logical DP algorithm as follows:

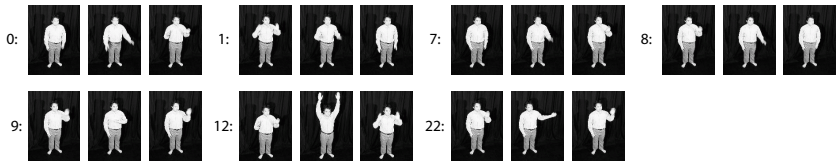

Fig. 7. Snapshot of extracted motion primitives.

1. Apply the logical DP algorithm to a pair of training gesture patterns of the assumed classes.
2. Decompose those two gesture patterns into similar subsequences and the remaining subsequences. All of those subsequences are candicates of motion primitives[4].
3. Repeat the above two steps for all pairs.
4. Unify similar subsequences if their distance [5] is smaller than θ_3.

The subsequences which survive the unification are considered as motion primitives. Any gesture pattern (from the assumed classes) will be approximately represented as a sequence of the resulting motion primitives.

Most past attempts for extracting motion primitives, such as Nakazawa et al. [7] and Fod et al. [8], have assumed that motion primitives can be obtained by segmenting gesture patterns at physically particular points, such as locally minimum speed points and zero-acceleration points.

In contrast, the motion primitives extracted by the procedure have two different properties. First, they are extracted without using any physically particular point, that is, they are free from any assumption (or users' prejudice) on motion primitives. Second, the extracted motion primitives totally depend on the assumed classes. In other words, if the classes of training gesture patterns change, the extracted motion primitives will also change. This property is especially useful for gesture recognition tasks, where the number of assumed gestures is often limited.

4.2 Extraction results of motion primitives

For each of 18 gesture classes, one of 6 patterns was used as a training pattern for the extraction of motion primitives. According to the above procedure, 142 subsequences were firstly detected and then unified into 26 motion primitives. **Fig. 7** shows snapshots of several motion primitives.Note that the parameter θ_3 was determined via a preliminary experiment.

[4] In this paper, we treat subsequences particular to only a certain gesture class as motion primitives. Thus, any gesture pattern can be decomposed into a sequence of motion primitives.

[5] Two subsequences may have different lengths and therefore their distance is evaluated by the conventional DP matching algorithm with Euclidean distance.

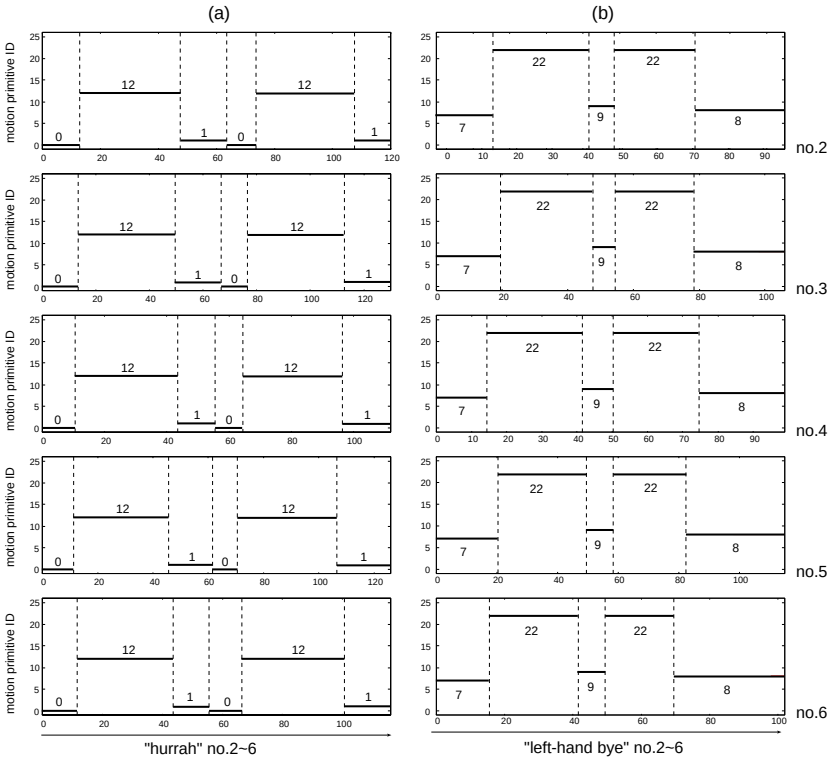

Fig. 8. Segmentation of gesture pattern by extracted motion primitives.

For evaluating the extracted motion primitives, the remaing five gesture patterns of each class were decomposed into the motion primitives. If the five gestures are decomposed into the same motion primitive sequence, the validity of the motion primitives can be shown experimentally. The decomposition was done by a recognition-based segmentation algorithm [5, 9], which performs segmentation and assignment of each segment to a motion primitive in an optimization framework.

Fig. 8 (a) is the decomposition result of a pattern "hurrah." The five gestures were correctly decomposed into the same motion primitive sequence, $0 \rightarrow 12 \rightarrow 1 \rightarrow 0 \rightarrow 12 \rightarrow 1$. (Those numbers correspond to the motion primitives shown in **Fig. 7**.) As shown in **Fig. 8 (b)** the five gestures of "(left-hand) bye" were also correctly decomposed into the same sequence, $7 \rightarrow 22 \rightarrow 9 \rightarrow 22 \rightarrow 8$.

The same decomposition results were obtained at 13 classes among 18 classes. This stable result indicates that the motion primitives extracted by the procedure of Section 4.1 represent common and particular motions validly. Thus, this result also shows resonable accuracy of the logical DP algorithm. The failure results, i.e., different decomposition results of the same class, were mainly due to lax unification on selecting motion primitives. For example, several subsequences of

"lowering both hands from shoulder height" were survived as motion primitives. This fact indicates that some gesture subsequences fluctuate more largely than others and thus the distance between them exceeded θ_3. The statistical extension pointed out in Section 3.2 will be useful also for tackling this fluctuation problem.

5 Conclusion

A logical DP matching algorithm has been proposed for detecting similar subsequences between two sequential patterns, such as gesture patterns. The algorithm was examined via an experiment of extracting motion primitives from a set of gesture patterns. The result of the experiment showed that the proposed algorithm could detect the similar subsequences among gesture patterns successfully and provide stable motion primitives. Future work will focus on (i) a statistical extension of $d(i, j)$, (ii) application to sequential patterns other than gesture, and (iii) utilization of the extracted motion primitives for practical tasks.

References

1. Durbin, R., Eddy, S., Korgh, A., Mitchison, g.: Biological sequence analysis. Camblidge University Press (1998)
2. Mount, D.: Bioinformatics: sequence and genome analysis. Second Eds., Cold Spring Harbor Laboratory Press (2004)
3. Tanaka, Y., Iwamoto, K., Uehara, K.: Discorvery of time-series motif from multi-dimensional data based on MDL principle. Machine Learning **58** (2005) 269-300
4. Zhao, T., Wang, T., Shum, H.-Y.: Learning a highly structured motion model for 3D human tracking. Proc. Asian Conf. Comp. Vis. (2002) 144-149
5. Ney, H., Ortmanns, S.: Progress in dynamic programming search for LVCSR. Proc. IEEE, **88**, 8 (2000) 1224–1240
6. Uchida, S. Sakoe, H.: A survey of elastic matching techniques for handwritten character recognition. IEICE Trans. Inf. Syst., **E88-D**, 8 (2005) 1781–1790
7. Nakazawa, A., K., Nakaoka, S., Ikeuchi, K., Yokoi, K.: Imitating human dance motions through motion structure analysis. Proc. Int. Conf. Intell. Robots Syst., (2002) 2539–2544
8. Fod, A., Mataric, M. J., Jenkins, O. C.: Automated deviation of primitives for movement classification. Autonomous Robots, **12**, 1 (2002) 39–54
9. Casey, R. G., Lecolinet, E.: A survey of methods and strategies in character segmentation. IEEE Trans. Pattern Anal. Mach. Intell., **18**, 7 (1996) 690–706