Improving Online Gesture Recognition with Template Matching Methods in Accelerometer Data

Long-Van Nguyen-Dinh, Daniel Roggen, Alberto Calatroni, Gerhard Tröster
Wearable Computing Lab
ETH Zurich
Switzerland
{*longvan,daniel.roggen,alberto.calatroni,troester*}@*ife.ee.ethz.ch*

***Abstract*—Template matching methods using Dynamic Time Warping (DTW) have been used recently for online gesture recognition from body-worn motion sensors. However, DTW has been shown sensitive under the strong presence of noise in time series. In sensor readings, labeling temporal boundaries of daily gestures precisely is rarely achievable as they are often intertwined. Moreover, the variation in daily gesture execution always exists. Therefore, here we propose two template matching methods utilizing the Longest Common Subsequence (LCSS) to improve robustness against such noise for online gesture recognition. *Segmented LCSS* utilizes a sliding window to define the unknown boundaries of gestures in the continuous coming sensor readings and detects efficiently a possibly shorter gesture within it. *WarpingLCSS* is our novel variant of LCSS to determine occurrences of gestures without segmenting data and performs one order of magnitude faster than the Segmented LCSS. The WarpingLCSS requires low-resource settings to process new arriving samples, thus it is suitable for real-time gesture recognition implemented directly on the small wearable devices. We compare our methods with the existing template matching methods based on Dynamic Time Warping (DTW) on two real-world gesture datasets from arm-worn accelerometer data. The results demonstrate that the LCSS approaches outperform the existing template matching approaches (about 12% in accuracy) in the dataset that suffers from boundary noise and execution variation.**

***Keywords*-Online Activity Recognition; Template Matching Method; LCSS; WarpingLCSS;**

I. Introduction

Online gesture recognition (*gesture spotting*) is important in many applications: human computer interaction (HCI) such as gesture-aware gaming, ambient assisted living, rehabilitation, sport, etc. In online recognition, types of gestures and their temporal boundaries must be recognized in the incoming streaming sensor data. Template matching methods (TMM) were shown to be powerful to spot online complex gestures [1]. In TMM, each gesture class is represented by one or only a few number of templates which are gesture instances in training set. Thus, TMM can maintain a certain constant amount of memory to store the templates. The similarity between the templates and the streaming sensor data is computed. High similarity with a template indicates that the gesture class of that template has likely been executed. Dynamic time warping (DTW) has been used in the previous works of TMM. [1]–[3].

However DTW is shown in time series analysis to be sensitive to outlier noise [4]. Nonetheless, imprecision in marking temporal boundaries of gestures is typical for daily activities, as these are often intertwined. In addition, there is usually no standard definition when the start and end of a gesture should be (e.g., drink gesture can start from the time user picks up a cup or when user's lip touches the cup). Therefore, even experts who label the same dataset may disagree on the exact boundaries. Moreover, the variation in daily gesture execution always exists. Therefore, the existing TMM with DTW is weak to both boundary noise and variability in daily gesture execution (see Fig. 9).

To improve online gesture recognition with template matching methods in the face of such noise, we propose two approaches derived Longest Common Subsequence (LCSS) [5]. Segmented LCSS relies on a sliding observation window to segment the incoming streaming sensor data and spots a possibly shorter gesture within it. WarpingLCSS is our variant of LCSS introduced here for the first time to detect occurrences of gestures in streaming data without segmenting data and performs one order of magnitude faster than the Segmented LCSS. The WarpingLCSS requires a small constant time and space to process new samples, hence it is suitable for real-time gesture recognition implemented directly on the small devices such as mobile phones or wearable sensors. In our system, sensor data is preprocessed and quantized to symbols by using k-means.

Our approaches have been tested and compared with the existing TMM approaches using DTW on two real-world gesture datasets comprising arm-worn accelerometer data. The results indicate that the LCSS methods outperform the DTW methods in accuracy in the dataset that suffers from boundary noise and high variability in execution.

II. Background and Related Work

Samples from body-worn sensors is in fact one type of time series data, therefore time series analysis methods are widely used for gesture recognition such as Hidden Markov Models [6]–[9] and TMMs using DTW [1]–[3].

Segmented DTW [2], [3] performs online gesture recognition by first buffering the streaming signals into an observation window. The window length is chosen based on the typical duration of the corresponding gesture class. A *t*est segment is a sequence that is examined whether it is an instance of one interesting gesture class. The start and end boundaries of a test segment can vary inside the window. A DTW distance is computed between all templates and the test segment, and the class of the closest template is eventually selected as label for the test segment if the distance falls below a certain rejection threshold. As the sensor delivers a new reading, the window is shifted by one sample and the process is repeated. Segmented DTW is time consuming since DTW is recomputed to find the best boundaries for the test segment inside the window and it is also recomputed every time the window shifts by 1 sample. A *nonsegmented DTW* variation [1] was proposed to reuse the computation of previous readings, recognize gestures and determine their boundaries without segmenting the stream.

Two most common similarity measures for matching two time series sequences are DTW and LCSS [10]. LCSS is shown in time series as more powerful than DTW in finding the similarity between two time series sequences in the existence of noise in dataset [4]. However, to the best of our knowledge, there is no prior work that utilizes the benefit of LCSS over noise in TMM for online gesture recognition on 3D accelerometer data. Our Segmented LCSS method can find the best boundaries of gesture inside the window efficiently because the LCSS knows which parts in the window contributes to the similarity. Thus, it is superior to Segmented DTW. Our proposed WarpingLCSS that can process one new sample in a small constant time and space is novel.

Longest Common Subsequence: We review LCSS, the classical problem to find the longest common subsequence in two strings. Given two strings A, B of length n, m respectively, LCSS(i,j) is a length of the longest common subsequence between the first i symbols of A and the first j symbols of B. The LCSS between A and B is LCSS(n,m).

$$LCSS(i,j) = \begin{cases} 0 & \text{, if i = 0 or j = 0} \\ LCSS(i-1,j-1)+1 & \text{, if A(i) = B(j)} \\ \max\begin{cases} LCSS(i-1,j) \\ LCSS(i,j-1) \end{cases} & \text{, otherwise} \end{cases}$$

LCSS and matching points between two sequences can be found efficiently using dynamic programming [5].

III. Training process

A. Data preprocessing

At training time, a training set is recorded continuously by 3D accelerometer and manually labeled with a predefined set of gesture classes. Gestures which are not in the predefined list are considered as Null class. To speed up gesture recognition, we extract a vectorial mean feature within sliding windows of size 6 samples and overlap 3, and quantize the resulting values with k-means. Thus, each 3D acceleration vector is quantized to its closest cluster centroid, and the motion sequence is represented as a string of symbols (i.e., the indices of the centroids). The distribution of centroids captures the variation of hand positions of the user. We define the distance d(l,m) between two symbols l and m as the Euclidean distance between the two corresponding centroids. We normalize these distances in a range [0,1]. We selected empirically k = 20, since this gave the best results for the considered datasets. Figure 1 shows the distribution of cluster centroids in 3D space for one user in the Opportunity dataset. When spotting, the same process of feature extraction and quantization is applied to the streaming sensor data, with the cluster centroids identified during training.

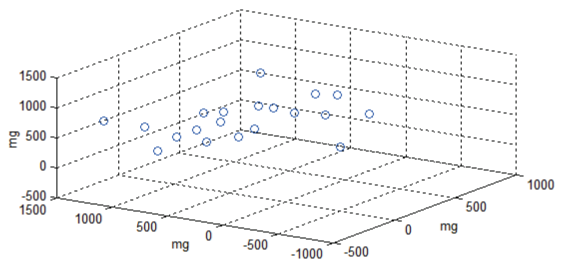


Figure 1. The distribution of cluster centroids of one user in Opportunity dataset, k = 20

B. Training phase: gesture template and rejection threshold

The higher LCSS two gestures have, the more likely they are similar and belong to the same class. For each class, a template represents the typical motion pattern for that class and a rejection threshold decides the minimum allowed similarity between two gestures in the same class. Templates and rejection thresholds are determined at training time. Specifically, for each class of gesture we compute a LCSS similarity matrix among all pairs of instances belonging to that class. The template is chosen as an instance that has the highest average similarity to all other instances of the same class. We choose the rejection threshold as the minimum LCSS between the chosen template and other gesture instances in the same class.

Thus, the training process is done offline in which the cluster centroids, the template and the rejection threshold for each gesture class (except Null) are identified.

IV. LCSS-based online spotting methods

The data processing flow to recognize gestures online is shown in Figure 2. Streaming data from sensor (S) are preprocessed and quantized to the k-means centroids (i.e., symbols) identified during training, then come to template matching module (TM) which uses TMM such as Segmented LCSS, WarpingLCSS to recognize gestures. If a sequence is spotted as belonging to multiple classes, the decision making module (DM) will decide which class is the best match and output the gesture.

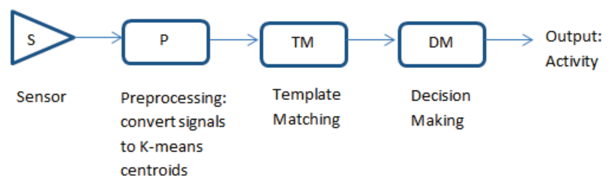
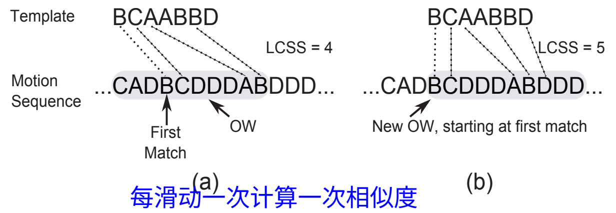


Figure 2. The data processing flow at spotting time

A. *Segmented LCSS*

In the Segmented LCSS approach, sensor readings go through a sliding observation window (*OW*). For each gesture class, the OW length is chosen as the length of the longest instance in that class in the training set. LCSS is computed only once between the gesture template and the whole string sequence in the OW. If the LCSS is greater than the corresponding rejection threshold, this indicates that an instance of that class is spotted. Because the LCSS algorithm can indicate which part in the OW contributes to the similarity (i.e., matched points between two sequences), the boundaries of the detected gesture can be managed from the first matched point to the last matched point with the template in the string sequence in the OW. In Segmented DTW [2], [3], the boundaries of the gesture must vary exhaustively in OW and DTW must be recomputed for each choice to find the best match. Thus, the Segmented LCSS is much faster than the Segmented DTW in finding the boundaries of spotted gestures in OW. When a new sample comes, the window can shift to the first matched point in the previous recognition process instead of shifting forward by only one sample as in the Segmented DTW. Figure 3 illustrates the spotting process of Segmented LCSS.



每滑动一次计算一次相似度

Figure 3. The Segmented LCSS spotting process. The shaded part represents the OW. The LCSS is computed between the gesture template and the OW to indicate whether an instance of the gesture class is spotted. The next OW will shift to the first matched point of the previous one.

Let W denote a window size and T denote length of a gesture template ($W \approx T$). The time complexity of Segmented LCSS to process a new coming symbol in the worst case (i.e., the window is shifted by 1) is $\mathcal{O}(W * T) \approx \mathcal{O}(T^2)$. Memory usage in Segmented LCSS is at most $\mathcal{O}(T^2)$. If the starting boundary of gesture is not needed (i.e., tracing back to find matched points is not needed), the memory complexity can be handled efficiently in $\mathcal{O}(T)$ [5]. Thus, the Segmented LCSS requires a constant time and space to process new sample.

B. *Nonsegmented WarpingLCSS*

In the Segmented LCSS, the LCSS must be recomputed every time the OW shifts. However, without using the OW, the LCSS algorithm can not find the start and end boundaries of gestures within the incoming streaming string. In this section, we introduce for the first time a novel variant of LCSS, the Warping LCSS, that removes the need of a sliding window and reduce the computation cost significantly. This algorithm determines the start and end boundary of gestures automatically.

Let S_t be a gesture template and S_m be a streaming string. S_m continuously adds new symbols when new samples arrive. WarpingLCSS(i,j) represents the similarity between the first i symbols of S_t and a gesture occurring at the position jth in the streaming string S_m. WarpingLCSS(i,j) (shortly, W(i,j)) is defined as follows:

$$W(i,j) = \begin{cases} 0 & \text{,if } i = 0 \text{ or } j = 0 \\ W(i-1,j-1)+1 & \text{,if } S_t(i) = S_m(j) \\ \max \begin{cases} W(i-1,j) - p * d(S_t(i), S_t(i-1)) \\ W(i,j-1) - p * d(S_m(j), S_m(j-1)) \end{cases} & \\ & \text{,otherwise} \end{cases}$$

惩罚项

with p is a penalty parameter of the dissimilarity, and d(l,m) is the distance between two symbols l and m as introduced in section III-A. If the WarpingLCSS algorithm encounters a similar symbol between S_t and S_m, the similarity W is increased by a reward (i.e., reward = 1). Otherwise, if it encounters dissimilar symbols, the similarity W is decreased by penalties. The WarpingLCSS algorithm tries to find instances having high similarity (i.e., more rewards and less penalties) with the template. Similar to the LCSS, the WarpingLCSS can be computed efficiently using dynamic programming.

Figure 4 illustrates the changes of values of similarity W between a template and a streaming string. As gestures different from those encoded by the template are executed, the similarity W is penalized consecutively and drops significantly. It can goes lower than 0. The WarpingLCSS algorithm starts with the similarity W(i,j) for the first row (i.e. i = 0) and the first column (i.e. j = 0) all 0, thus, it control the lower bound of the similarity W. When a matched gesture of many similar symbols occurs, the similarity value can quickly goes up above 0 and continuously accumulates rewards. The WarpingLCSS is also strong to handle a few number of noisy symbols occurring in the matched instance with a small penalty.

The value of the penalty parameter p depends on applications and can be chosen by cross-validation to maximize the recognition accuracy. It controls how the system accepts the dissimilarity between two instances of the same class.

The WarpingLCSS detects matched gestures and their temporal boundaries based on the similarity W and the rejection threshold. For each class, within the stream of values of the similarity W, the local maximum (LM) are detected and compared with the rejection threshold. If LM is greater than the threshold, a matched gesture is recognized with the end point as the position at LM. The start point of the matched activity can be found by tracing back the matching path. The LCSS between the template and the matched gesture is accumulated during the trace-back process if necessary (i.e., when a gesture is spotted as belonging to multiple classes) to make a decision (discussed in next section). Figure 5 show the close-up of the streaming similarity W between a matched gesture and a template. It also show how the WarpingLCSS detects the temporal boundaries of matched gestures.

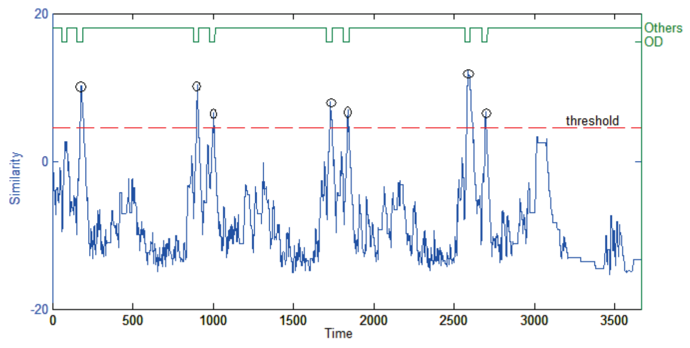


Figure 4. WarpingLCSS between an Open Door template and a streaming string, p = 3. The values of similarity W is updated continuously when new samples come. The line on the top shows the ground truth. The small circles show gesture detection at spotting time.

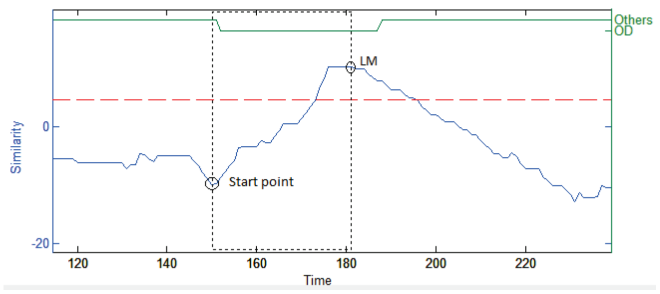


Figure 5. The close-up of the first detected OpenDoor in the streaming string

When a new symbol comes at time t, WarpingLCSS only needs to update the value of W for this new sample and a template. Thus, the time complexity of Warping LCSS is $\mathcal{O}(T)$. It is faster than the Segmented LCSS by one order of magnitude. The WarpingLCSS maintains at most $\mathcal{O}(T^2)$ memory for the need to trace back the starting boundary of detected gesture.

C. *Resolving spotting conflicts*

The incoming streaming string is concurrently "compared" with templates of all concerned gesture classes. If a gesture is spotted as belonging to multiple classes (i.e., boundaries of spotted instances are overlapping), we have to decide which class is the best match. We define the normalized similarity NormSim(A,B) $=$ $LCSS(A, B)/max(\|A\|, \|B\|)$, with $\|A\|$ and $\|B\|$ are the lengths of the strings A and B, respectively. The NormSim between the template and the matched gesture is output to the decision making module (DM). The class with highest NormSim is chosen as the best match. This process is the same for both Segmented LCSS and WarpingLCSS.

V. Dataset

We evaluate the methods on two datasets. The first is the Opportunity dataset [11] which is publicly available for activity recognition. It is a rich multi-modal dataset collected in a naturalistic environment akin to an apartment, where users execute daily gestures. The dataset is characterized by a predominance of Null class (80%) and a large variability in the execution of the activities. We evaluate the method on a subset of 4 subjects comprising 20 repetitions of 15 gesture classes as shown in Table I. Note that in this dataset, there are three drawers at different heights. We use a 3D accelerometer at subjects' dominant (right) arms for the evaluations (30Hz sampling rate). Fig. 6 shows a visual inspection of some instances of all classes.

The second dataset is an HCI gesture dataset executed by a single person. The gestures as shown in Table I are geometric shapes executed with the arm in the vertical plane. Fig. 7 show gestures and their duration. This dataset is characterized by a low execution variability and well-defined labeling. The Null class takes 57%. We use a 3D accelerometer at the dominant right lower arm for the evaluations (30Hz sampling rate).

Table I
Gestures in HCI and Opportunity datasets

HCI Gestures				
Circle	Triangle	Square	Infinity	Slider
Their Speculars				Null

Opportunity Gestures		
Null	clean Table (CT)	open Drawer1 (ODr1)
close Drawer1 (CDr1)	open Drawer2 (ODr2)	close Drawer2 (CDr2)
open Drawer3 (ODr3)	close Drawer3 (CDr3)	open Door (OD)
close Door (CD)	open Fridge (OF)	close Fridge (CF)
open Dishwasher (ODi)	close Dishwasher (CDi)	drink Cup (D)

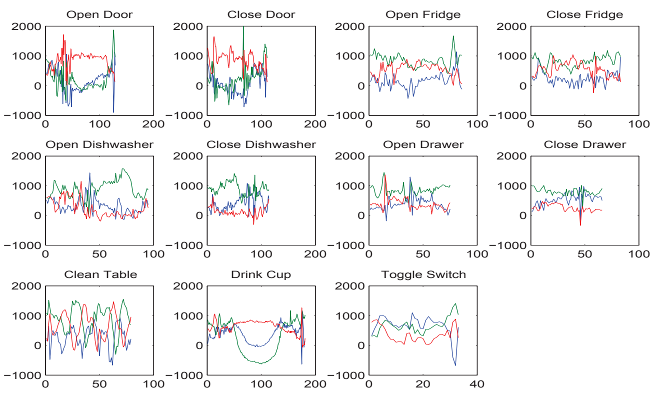


Figure 6. 3D Acceleration of the dominant upper arm of one user while executing some of gestures in the Opportunity dataset. Sampling rate is 30Hz. Units: samples on the abscissa and milli-g on the ordinate.

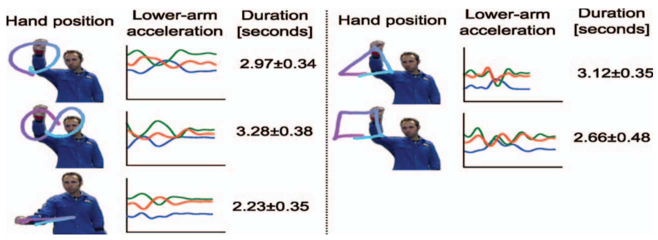


Figure 7. 3D Acceleration of the dominant lower-arm of one user while executing gestures in the HCI gesture dataset. Sampling rate is 30Hz. Units: samples on the abscissa and milli-g on the ordinate.

VI. Experiments and Results

For each subject, we perform 5-fold cross validation. We compare our proposed LCSS methods to the Segmented DTW [2] and the Nonsegmented DTW [1]. All methods use the same strategy to select templates (the best similarity average) and the rejection threshold as discussed in section III-B. However, smaller DTW distance yields higher similarity. Thus, for the DTW approaches, the class template is an instance with the minimum average of DTW distances to all other instances in the same class, and the rejection threshold is the maximum DTW distance between the class template and all other instances in the same class. In the spotting making decision, the DTW between the template and the matched activity is also normalized by the maximum length of the template and the gesture. The instance with the lowest normalized distance indicates which gesture class the system has spotted.

A. Evaluation metrics

We evaluate the performance with the accuracy (accuracy = correct predicted/number of samples). However, the accuracy metric is highly affected by the sample distribution of gesture classes which may be highly unbalanced in real-life datasets. Therefore, we also adopt the F1 measure to give a complementary assessment of performance.

$$F1 = \sum_i 2 * w_i \frac{precision_i * recall_i}{precision_i + recall_i}$$

where i is the class index and w_i is the proportion of samples of class i. $precision_i$ is the proportion of samples of class i predicted correctly over the total samples predicted as class i. $recall_i$ is the proportion of samples of class i predicted correctly over the total samples of class i. We present two ways of computing the F1, either including (F1_Null) or excluding the Null class (F1_NoNull). F1_Null does not consider the null class, but still takes into account false predictions of one gesture class to Null. The recognition system that has high values of both F1_Null and F1_NoNull predicts well both gesture classes and Null class.

B. Results

1) Gesture template similarity: Fig. 8 shows the DTW distance and the LCSS similarity matrices between pairs of gestures in the training set of one user in Opportunity dataset. The LCSS similarity matrix can discriminate gestures of different classes better. It can be explained that

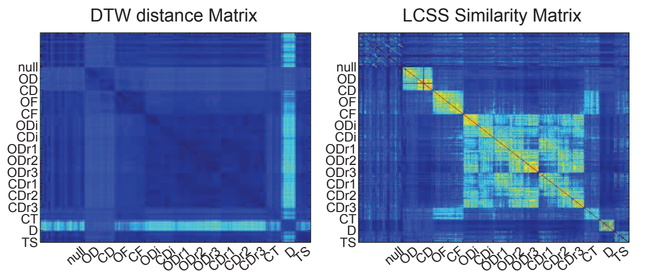


Figure 8. DTW distance matrix and LCSS similarity matrix of gestures of one user in the Opportunity training dataset. We see how the LCSS metric allows to better distinguish the gestures of different classes.

DTW is sensitive to boundary noise and variation in gesture execution. A representative case is illustrated in Fig. 9 that shows the effect of the extension of the boundaries of a gesture on DTW. In this example, the same change to the same patterns on boundaries increases the DTW distance significantly, while the LCSS changes slightly. The DTW is very sensitive due to the dissimilarity accumulating on the noisy parts. The sensitivity of DTW to the variation in gesture execution can be explained similarly.

2) Performance results: Table II shows the results of template matching approaches on the two datasets [1]. The results indicate that both LCSS approaches outperform two DTW approaches in the Opportunity dataset (about 12% higher in accuracy and F1 measure). The Opportunity dataset contains large variability in the executions of the daily activities and naturally contains a large amount of boundary noise. In the

[1]We conducted a 2-sided hypothesis test at the 0.01 level of significance as in [12], the tests showed that the accuracy differences among the four methods are statistically significant.

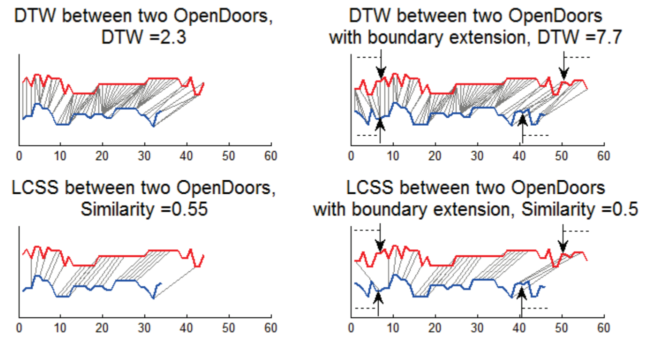


Figure 9. The sensitivity of DTW and LCSS to boundary noise. Two string instances of OpenDoor activity are shifted apart to show the patterns and the warping path (DTW distance) or the matching points (LCSS) between them. On the right, both instances are extended by 6 synthetic samples on each side. The arrows point to the positions from which the boundaries are extended.

HCI dataset, the LCSS and DTW approaches yield similar results and gain high accuracy. The HCI dataset has less variability than the Opportunity dataset as geometric-shape gesture execution contains low variability and the start and end boundaries of gestures are well-defined .

Table II
THE AVERAGE ACCURACY AND F1 MEASURE OVER SAMPLE UNIT IN TWO DATASETS

Opportunity Dataset, Avg $\pm$ Stdev of 4 subjects			
Method	Accuracy	F1_Null	F1_NoNull
Segmented DTW	0.37 $\pm$ 0.02	0.37 $\pm$ 0.03	0.33 $\pm$ 0.03
Nonsegmented DTW	0.38 $\pm$ 0.04	0.34 $\pm$ 0.06	0.27 $\pm$ 0.09
Segmented LCSS	0.48 $\pm$ 0.03	0.48 $\pm$ 0.04	0.44 $\pm$ 0.08
Warping LCSS	0.50 $\pm$ 0.06	0.48 $\pm$ 0.05	0.43 $\pm$ 0.07

HCI Dataset			
Method	Accuracy	F1_Null	F1_NoNull
Segmented DTW	0.76	0.75	0.66
Nonsegmented DTW	0.78	0.78	0.72
Segmented LCSS	0.78	0.77	0.68
Warping LCSS	0.74	0.72	0.63

VII. Conclusion

In this paper, we have introduced two template matching methods derived from LCSS to recognize gestures online. Both Segmented LCSS and WarpingLCSS achieve better accuracy than the existing DTW approaches in the daily-gesture dataset that suffers from boundary noise and execution variation. The WarpingLCSS is faster than the Segmented LCSS by one order of magnitude, but requires to choose an application-specific penalty parameter by cross-validation in training process. The 3D accelerometer data is converted in 1D string thus our spotting process working with symbols is simple and fast. Our novel WarpingLCSS requires a small constant amount of space and time, thus it is appropriate for real-time recognition. In future work, we plan to extend the system by applying sensor fusion and multiple templates for each activity class. We also plan to investigate the WarpingLCSS and the penalty parameter in more general time series data.

Acknowledgement

This work was supported by the EU 7th Framework Program project OPPORTUNITY under FET-Open grant number 225938, and also partially funded by the Hasler Foundation project Smart-DAYS.

References

[1] T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and G. Tröster, "Wearable activity tracking in car manufacturing," *IEEE Pervasive Computing Magazine*, vol. 7, no. 2, pp. 42–50, April-June 2008.

[2] M. H. Ko, G. West, S. Venkatesh, and M. Kumar, "Online context recognition in multisensor systems using dynamic time warping," in *Proceedings of the 2005 Intelligent Sensors, Sensor Networks and Information Processing Conference*, ser. ISSNIP '05, 2005.

[3] B. Hartmann and N. Link, "Gesture recognition with inertial sensors and optimized DTW prototypes," in *SMC*, 2010, pp. 2102–2109.

[4] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '03, 2003, pp. 216–225.

[5] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed., 2001.

[6] H.-K. Lee and J. H. Kim, "An HMM-based threshold model approach for gesture recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 10, pp. 961–973, Oct. 1999.

[7] J. Deng and H. Tsui, "An HMM-based approach for gesture segmentation and recognition," in *Proceedings of the International Conference on Pattern Recognition - Volume 3*, ser. ICPR '00, Washington, DC, USA, 2000.

[8] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recognition*, vol. 41, no. 6, pp. 2010–2024, 2008.

[9] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a Wii controller," in *Proceedings of the 2nd international conference on Tangible and embedded interaction*, ser. TEI '08. ACM, 2008, pp. 11–14.

[10] T.-C. Fu, "A review on time series data mining," *Eng. Appl. Artif. Intell.*, vol. 24, no. 1, pp. 164–181, Feb. 2011.

[11] D. Roggen, A. Calatroni, et al., "Collecting complex activity data sets in highly rich networked sensor environments," in *7th Int. Conf. on Networked Sensing Systems*, 2010, pp. 233–240.

[12] I. Guyon, J. Makhoul, R. Schwartz, and V. Vapnik, "What size test set gives good error rate estimates?" in *IEEE Trans PAMI*, 1996, pp. 52–64.