Journal of Machine Learning Research 15 (2014) 3187-3220 Submitted 2/14; Revised 5/14; Published 10/14

Robust Online Gesture Recognition with Crowdsourced Annotations

Long-Van Nguyen-Dinh LONGVAN@IFE.EE.ETHZ.CH
Alberto Calatroni ALBERTO.CALATRONI@IFE.EE.ETHZ.CH
Gerhard Tröster TROESTER@IFE.EE.ETHZ.CH
Wearable Computing Lab
ETH Zürich
ETZ H 95, Gloriastrasse 35, Zürich 8092, Switzerland

Editor: Isabelle Guyon, Vassilis Athitsos and Sergio Escalera

Abstract

Crowdsourcing is a promising way to reduce the effort of collecting annotations for training gesture recognition systems. Crowdsourced annotations suffer from "noise" such as mislabeling, or inaccurate identification of start and end time of gesture instances. In this paper we present SegmentedLCSS and WarpingLCSS, two template-matching methods offering robustness when trained with noisy crowdsourced annotations to spot gestures from wearable motion sensors. The methods quantize signals into strings of characters and then apply variations of the longest common subsequence algorithm (LCSS) to spot gestures. We compare the noise robustness of our methods against baselines which use dynamic time warping (DTW) and support vector machines (SVM). The experiments are performed on data sets with various gesture classes (10-17 classes) recorded from accelerometers on arms, with both real and synthetic crowdsourced annotations. WarpingLCSS has similar or better performance than baselines in absence of noisy annotations. In presence of 60% mislabeled instances, WarpingLCSS outperformed SVM by 22% F1-score and outperformed DTW-based methods by 36% F1-score on average. SegmentedLCSS yields similar performance as WarpingLCSS, however it performs one order of magnitude slower. Additionally, we show to use our methods to filter out the noise in the crowdsourced annotation before training a traditional classifier. The filtering increases the performance of SVM by 20% F1-score and of DTW-based methods by 8% F1-score on average in the noisy real crowdsourced annotations.

Keywords: gesture spotting, crowdsourced annotation, longest common subsequence, template matching methods, accelerometer sensors

1. Introduction

Wearable computing is gaining momentum through the availability of an increasing choice of devices, like smart watches, glasses and sensor-equipped garments. A core component to allow these devices to understand our context is online gesture recognition (*spotting*) in which types of gestures and their temporal boundaries must be recognized in the incoming streaming sensor data. This is carried out using machine learning approaches on different sensing modalities, like acceleration (Bao and Intille, 2004) and video (Elmezain et al., 2009; Yoon et al., 2001).

Training a gesture recognition system requires an annotated training data set that is used to perform supervised learning (Bao and Intille, 2004; Ravi et al., 2005; Aggarwal and Ryoo, 2011; Chen et al., 2012). Specifically, the annotations comprise the start and end times (i.e., temporal boundaries) of gestures of interest and their corresponding labels. Reference data sets are usually annotated by a small number of experts to be as accurate as possible. However, the labeling process is extremely time-consuming: it may take up to 7-10 hours to annotate gestures in a 30-min video (Roggen et al., 2010). Moreover, it is also costly to hire experts to annotate data corpora.

Crowdsourcing has been emerged recently to address these issues (Howe, 2006; Doan et al., 2011). Crowdsourcing is defined as a model that outsources tasks which are traditionally performed by experts to a crowd of ordinary people. Thus, crowdsourcing is promising to reduce the cost and time of labeling. Recently, crowdsourcing has been exploited to get labeling for training data sets for gesture recognition (Nguyen-Dinh et al., 2013c). However, labels obtained from crowdsourcing are provided by low-commitment anonymous workers, thus they are commonly unreliable and noisy (Sheng et al., 2008). In gesture annotation from crowdsourcing, the challenge is to obtain labels matching ground truth, attaining both correct labels and correct temporal boundaries.

Using multiple annotators for the same annotation task by watching videos or audios is a popular strategy to get a good annotation from crowdsourcing (Yuen et al., 2011; Nguyen-Dinh et al., 2013c). However, multiple annotators may not be applicable in some cases, either due to the higher cost or because of some privacy concerns. This latter case occurs when the annotation involves some personal context information, including for example location or other sensitive data. Hence, the annotation is often provided and relied on the crowdsourced user for his recorded data. Moreover, it is very difficult to ask the anonymous low-commitment user to clean his annotation because it is time consuming and he may not remember exactly what he has done. In these cases, the large presence of noise in the training data annotation can degrade significantly the performance.

While other research is focusing on how to improve the quality of crowdsourced annotations, we here point out the need for algorithms that can cope with the kinds of annotation errors that will anyway remain. In this work, we show that our proposed template matching methods (TMMs) based on the longest common subsequence algorithm (known as LCSS or LCS in the literature) are suitable for online gesture recognition in a setting where training data are affected significantly by labeling noise. Additionally, the work targets the recognition of gestures based on acceleration data recorded from only one accelerometer mounted on the user's arm. The reason to just use one sensor is that this setting will be the most common one with smart watches in the close future. Recognizing gestures with just motion data from one sensor is challenging due to the ambiguities in the sensor data, especially with high percentage of *null* class (no gesture of interest).

1.1 Contributions

In this paper, we make the following contributions:

1. We discuss how gesture recognition systems can leverage crowdsourcing to collect annotated data. We address the challenges that arise and then propose a taxonomy

of annotation noise which occur in a crowdsourcing setting. We also give analysis on annotation noise in the real crowdsourced annotated data set.

2. We propose SegmentedLCSS and WarpingLCSS as TMMs for online gesture recognition. These methods were first presented in our previous work (Nguyen-Dinh et al., 2012) and have been shown to perform well in clean annotated gesture data sets both in terms of computational complexity and accuracy. In this work, we show their robustness to the labeling noise from crowdsourcing.

3. We compare the robustness of our gesture recognition methods against three baselines using two variations of dynamic time warping and support vector machines. The algorithms are tested with annotations collected in real crowdsourcing scenarios as well as the synthetic crowdsourced annotations in three data sets recorded from accelerometers on arms. We also investigate the impact of different kinds of noises in crowdsourced annotation on the performance of the gesture recognition methods.

4. We investigate the property of LCSS of being able to select clean templates, which makes it suitable also as a filtering component to select good training examples despite noisy annotations. This filter can be used in combination with other classifiers. We show how inserting this filtering step improves the performance of SVMs and TMMs based on dynamic time warping.

The rest of the paper is organized as follows. In Section 2, we first review existing work in online gesture recognition and crowdsourcing. In Section 3, we discuss crowdsourcing in gesture recognition and propose a taxonomy of annotation noise in gesture labeling by crowdsourcing. Then, in Section 4, we present our proposed SegmentedLCSS and WarpingLCSS methods. The experiments are described in Section 5. We present quantitative results evaluating the robustness of our proposed methods against the baselines in Section 6. Finally, Section 7 concludes our work and gives some potential research directions.

2. Related Work

In this section we discuss related work in the fields of gesture recognition and crowdsourcing, pointing out the lack of an analysis of how noise present in typical crowdsourced annotations impacts gesture recognition algorithms.

2.1 Annotation Techniques

Supervised learning techniques require a set of annotated training samples to build gesture models. Therefore, many annotation techniques have been proposed to collect annotated data. There are *offline annotation* techniques which rely on video and audio recordings (Roggen et al., 2010), subject self-report of activities at the end of the day (Van Laerhoven et al., 2008). *Online annotation* (i.e., real-time) techniques perform the annotation during execution of the activities, like experience sampling (Froehlich et al., 2007) which prompts periodically to a user to ask information about his current activities, or direct annotation in which users responsibly provide a label before an activity begins and indicate when the activity ends (Rossi et al., 2012). There is a trade-off between accuracy of an annotation technique and the amount of time required for annotation (Stikic et al., 2011). For example, offline annotation on video recordings by experts can provide accurate annotations, however it is extremely time consuming (Roggen et al., 2010), and non-scalable to large number of users. In contrast, the self-report of the subject may require less time but the accuracy depends on the subject's ability to recall activities. Therefore, most of the existing works require video annotation by experts to obtain clean and correct annotated data sets (Roggen et al., 2010) or provide a course to teach subjects carefully how they should record and annotate their data correctly (Bao and Intille, 2004).

2.2 Crowdsourcing

Crowdsourcing services, like Amazon Mechanical Turk (AMT)[1] and Crowdflower[2], have emerged recently as a new cheap labor pool to distribute annotation tasks to a large number of workers (Yuen et al., 2011). Crowdsourcing tasks are performed by low-commitment anonymous workers, thus acquired data is commonly unreliable and noisy (Sheng et al., 2008). Therefore, the same task is often redundantly performed by multiple workers and majority voting is a popular decision making method used to identify the correct answers (Yuen et al., 2011). Moreover, in crowdsourcing, malicious workers often take advantage of the verification difficulty (the ground truth is unknown) and submit low-quality answers.

Due to the error-prone nature of crowdsourcing, several strategies were proposed to estimate the quality of workers, in order to reject low-performing and malicious workers. Verifiable questions or pilot tasks for which the requester knows the correct answers is a common empirical strategy to screen workers from crowdsourcing (Kittur et al., 2008; Yuen et al., 2011). Another way to ensure quality is to check the agreement in annotations among workers to detect non-serious workers (Nguyen-Dinh et al., 2013c). Dawid and Skene (1979) proposed a theoretical model that used the redundancy in acquiring answers (i.e., the same task is completed by multiple workers) to measure the labeling quality of the workers. Recently, Raykar et al. (2010) proposed Bayesian versions of worker quality inference. Ipeirotis et al. (2010) improved the method by separating spammers who provide low-quality answers intentionally from biased workers who are careful but biased.

Recently, crowdsourcing has been exploited also in the field of activity recognition to collect annotated training data sets (Rossi et al., 2012; Nguyen-Dinh et al., 2013a,b,c; Lasecki et al., 2013). These works showed that crowdsourced data is erroneous, therefore, filtering strategies such as multiple labelers and outlier removal should be used to reduce labeling noise.

Although many strategies are used to reduce noise in crowdsourced data annotation, there is no guarantee to have a perfect annotation, especially when using multiple labelers can not be applied. Until now, the impact of the noisy annotations in crowdsourcing on the training of gesture recognition methods was not investigated. Furthermore, the nature of the noise that affects the annotations in a crowdsourcing scenario for gesture recognition has not been analyzed yet. These two latter topics are subject of the present paper.

1. The home page for AMT is `http://www.mturk.com`.
2. The home page for Crowdflower is `http://crowdflower.com`.

2.3 Online Gesture Recognition Methods

Signals from body-worn sensors belong to the category of time series data. Suitable machine learning and pattern recognition techniques for online gesture recognition include Hidden Markov Models (HMM) (Lee and Kim, 1999; Deng and Tsui, 2000; Junker et al., 2008; Schlömer et al., 2008), template matching methods (TMM) using mostly dynamic time warping—in short DTW (Ko et al., 2005; Stiefmeier et al., 2008; Hartmann and Link, 2010) and support vector machines (Ravi et al., 2005; He et al., 2008; Wu et al., 2009).

HMMs are not appealing since a large amount of training data is required to get results comparable to other TMMs and SVM. In Vogler and Metaxas (1999) for example, about 1300 instances for 22 classes (i.e., about 60 instances per class) are used to train the HMM, whereas TMMs can work with as little as one training instance per class. The issue of the amount of training data is mentioned also in Cooper et al. (2012), where the authors state, referring to HMMs: "While they have been employed for sign recognition, they have issues due to the large training requirements". In Alon et al. (2009), a variation of HMMs is selected but the parameters could not be learnt because of the scarcity of training data: "We fix the transition probabilities to simplify the learning task, because we do not have sufficient training data to learn more parameters". HMMs remain nevertheless an interesting approach for cases where a large data corpus is available, which is often the case in the field of video-based gesture or sign language recognition, see for example Wilson and Bobick (1999); Lee and Kim (1999); Keskin et al. (2011).

Segmented DTW (Ko et al., 2005; Hartmann and Link, 2010) performs online gesture recognition by first buffering the streaming signals into an observation window. A *t*est segment is a sequence that is examined to classify whether it is an instance of a gesture class. The start and end boundaries of a test segment can vary inside the window. A DTW distance is computed between all templates which represents gesture classes 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 one sample. A *nonsegmented DTW* variation was proposed by Stiefmeier et al. (2008) to reuse the computation of previous readings, recognize gestures and determine their boundaries without segmenting the stream.

Along with DTW, the other commonly used similarity measure for matching two time series is *LCSS* (Fu, 2011). In previous work (Nguyen-Dinh et al., 2012), we introduced two variations of LCSS-based template matching for online gesture spotting and recognition. We applied the methods to accelerometer data. These LCSS-based classifiers (SegmentedLCSS and WarpingLCSS) proved to outperform DTW-based TMMs, both in terms of computational complexity and accuracy (especially for data sets containing high variability in gesture execution as shown in Nguyen-Dinh et al., 2012). Furthermore, our methods were designed with the goal of being robust in case of noisy annotations. The validation of this aspect is the main topic of the present article. The impact of the various kinds of noise occurring in crowdsourced annotations on TMMs has not been investigated in previous literature, to the best of our knowledge.

In sign language recognition literature, we find two other works proposing the use of LCSS as a classifier, applied to video data (Frolova et al., 2013; Stern et al., 2013). In both cases, the methods use a sliding window to set temporal boundaries of a gesture inside the window, similarly to our SegmentedLCSS. With our WarpingLCSS, this need of using a window is removed, reducing the computational complexity. It is interesting to note how Stern et al. (2013) states that "It can then be said that the MDSLCS algorithm can outperform the HMM classifier for both pre-cut and streaming gestures", which supports the idea of using TMMs instead of HMMs to make best use of the available training data. TMMs are competitive with HMMs also with respect to null-class rejection, meaning the ability to spot a gesture within a continuous stream.

Some algorithms present in the literature rely on k-means or spatio-temporal clustering to transform the raw signals into so-called "fenemes", or subunits (Bauer and Karl-Friedrich, 2002; Fang et al., 2004), which allows to reduce the amount of training data, due to the fact that more gestures can contain the same feneme, so that a critical mass can be achieved in terms of amount of training data. We use a similar approach based on k-means clustering to find a quantization of the signals which gives good results.

A large body of literature focuses on a recognition performed on video data, for example for the recognition of sign language (see for example Wilson and Bobick, 1999; Bowden et al., 2004; Alon et al., 2009; Keskin et al., 2011). However, gesture recognition from wearable sensors, e.g., one accelerometer at the wrist, would allow to scale up the recognition system to many users immediately because the system can be deployed easily wherever a user goes with the motion sensor mounted on hand. It does not need any other infrastructure like cameras, which do not follow us everywhere in practice. Of the video-based approaches, the one of Hao and Shibata (2009) captures the videos directly by a moving camera, which could be easily wearable. However, from the practical point of view, such an option has some limitations: first, such a device would be quite costly; second, processing signals from a camera is more computationally intensive than processing those from a motion sensor; third, capturing video data is much more intrusive due to privacy concerns.

2.4 Robustness against Annotation Noise

The impact of noise in annotations on the performance of classifiers has been investigated in the literature (Angluin and Laird, 1988; Amini and Gallinari, 2005; Gayar et al., 2006; Lawrence and Schölkopf, 2001; Stikic et al., 2011). The above cited studies do not concern template matching methods. Moreover, they conducted experiments on synthetic noisy data. Additionally, under "annotation noise", or "class noise", only the case of having wrong labels (i.e., labels are substituted as other classes) was considered. Noise in gestures annotation can nevertheless also mean having labelings with temporal boundaries differing from the ground truth, e.g., a gesture marked as starting earlier and ending later than the ground truth. These other kinds of noise were neglected until now, and they are investigated in this paper in both synthetic and real crowdsourced annotated data.

3. Crowdsourcing in Gesture Recognition

In this section we discuss how gesture recognition systems can leverage crowdsourcing. We outline the challenges that arise and provide a taxonomy of the annotation noises, i.e., the mistakes that affect crowdsourced annotations. We then measure these annotation noises in a real crowdsourced data set.

Gesture recognition systems can take advantages of crowdsourcing in three ways:

1. Crowdsourcing can be used to acquire annotations for an existing gesture data set by asking crowdsourced workers to watch video footage synchronized with the sensor data (Nguyen-Dinh et al., 2013c; Lasecki et al., 2013).

2. Berchtold et al. (2010) proposed a system that asks users to both record and annotate activities. This system can be deployed in a crowdsourcing manner. Users can sporadically select gestures they want to perform and record them with a device (e.g, smart watch, smart phone, etc.). This way, multiple annotated gestures provided by a large user base could contribute to a central repository which grows in time. The data set would capture the variability in gesture execution due to the different people contributing.

3. A more obtrusive crowdsourcing task would ask users to record and annotate as many activities and gestures as possible over a long time span (e.g., weeks). This type of crowdsourced data collection would be useful to gather data for long-term health care monitoring systems.

In any of the previous scenarios, the outcome would be an annotated training data set, with which algorithms can be trained. The benefit of the crowdsourcing setting is that a large data set can be collected quickly, if the crowdsourced user base is large enough.

3.1 Taxonomy of Sources of Annotation Noises

The major challenge in any of the settings outlined above is the quality of the labels obtained, which are unreliable for many reasons. We define the following taxonomy of annotation noises along with examples:

- Some gestures or activities can be understood differently with respect to when they actually start and end. The temporal boundaries of the gesture *drink* can be set from the time when the user picks up a glass to when he or she puts it back to the table. Another variation is that the gesture is annotated only when the person is actually drinking. Both annotations are valid, but this uncertainty of temporal boundaries has an impact on the algorithms that will be trained with the collected annotated data. However, even when we assume the definition of gesture boundary is given, the errors in gesture boundary still happen due to the carelessness of crowdsourced labelers. We call this form of noise *boundary jitter*. We define *boundary jitter* as the presence of a shift in the annotation boundaries, while the label matches the actual gesture (ground truth).

- Some instances of gestures can be wrongly annotated or missed altogether. This can occur for example if the video footage does not have enough resolution to spot subtle manipulative gestures, or more simply if the labeler does not annotate all gestures that are occurring. We use the term *label noise* to denote instances where gestures are associated to wrong labels or to no label at all.

We further categorize *boundary jitter* into four error types, namely *extend, shrink, shift left* and *shift right* according to how the temporal boundary of a gesture is shifted compared to the ground truth. Figure 1a illustrates the subclasses of *boundary jitter*.

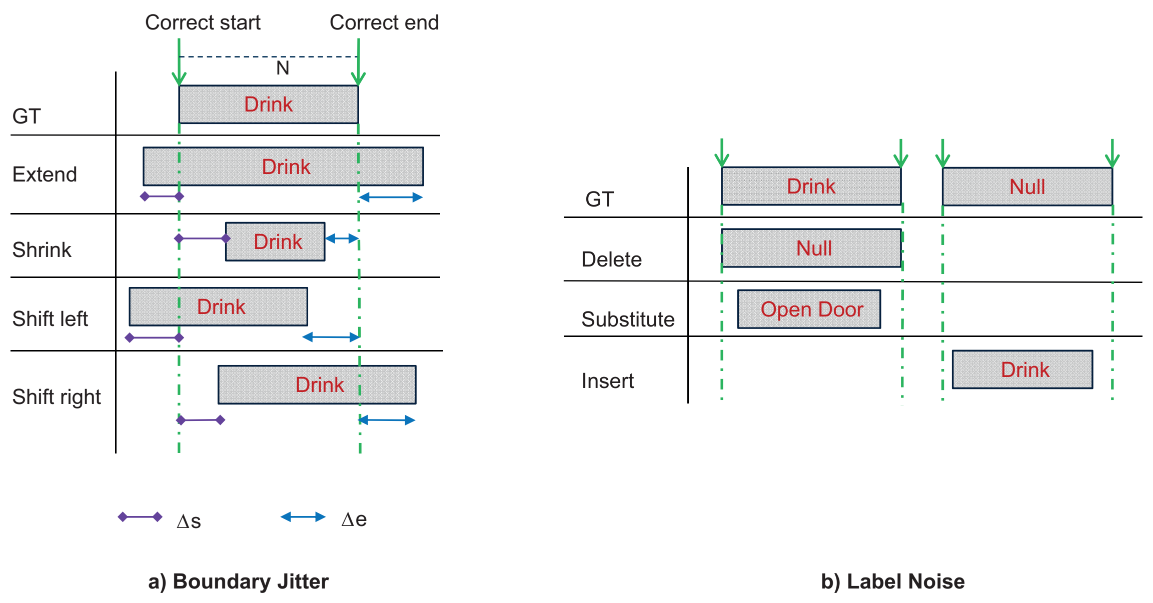

Figure 1: Illustrations of boundary jitter and label noise in crowdsourcing annotation. GT stands for ground truth. The blue dash-dotted lines indicate the correct boundary of a gesture.

- *Extend*: The starting boundary is set earlier and the ending boundary is set later. The information of the gesture instance is preserved, but noise is attached to the gesture instance in the form of samples which belong actually to another gesture class or to no class of interest at all (i.e., *null* class).
- *Shrink*: The starting boundary is set later and the ending boundary is set earlier. In this case, some information of the gesture instance is missed.
- *Shift left*: Both starting and ending boundaries are set earlier. In this case, some information of the gesture instance is missed and noise is added at the end of the gesture.
- *Shift right*: Both starting and ending boundaries are set later. In this case, some information of the gesture instance is missed and noise is added at the beginning of the gesture.

We also categorize *label noise* into three error types, namely *delete, substitute* and *insert.*

- *Delete*: A gesture instance is not annotated. It is automatically marked as *null* class.

- *Substitute*: A gesture instance is labeled as another gesture class.
- *Insert*: A gesture instance is labeled where no gesture of interest actually occurs.

Figure 1b illustrates the subclasses of *label noise.* The subclasses of *label noise* are similar to the definition of classification errors evaluated in performance metrics proposed by Ward et al. (2011). However, in this work, we consider those errors existing in annotations of training data set.

3.2 Annotation Noise Parameters

Along with the taxonomy provided in the previous section, we here list the parameters that quantify the amount of noise in the annotation. Given a gesture instance, let *start* and *end* be the start time and end time of the crowdsourced annotation. Let GT_start and GT_end be the corresponding ground truth boundaries. Let N denote the time length of the gesture ($N = |GT\_end - GT\_start|$). We define Δs as the time difference between the crowdsourced start time and the correct start time ($\Delta s = |start - GT\_start|$). Similarly, we define Δe as the time difference between the crowdsourced end time and the correct end time ($\Delta e = |end - GT\_end|$). Δs and Δe are illustrated in Figure 1a for the different boundary jitter noises.

For *boundary jitter* and for the corresponding subclasses, we define a *jitter level* to quantify the proportion of time that is wrongly annotated in a gesture due to the jitter. The jitter level also indicates how much the boundaries stray from the correct annotation. These jitter parameters are calculated as follows:

$$\begin{aligned}
\text{extend level} &= \text{proportion of time noisy samples added} \\
&= \tfrac{\Delta s + \Delta e}{N}.
\end{aligned}$$

$$\begin{aligned}
\text{shrink level} &= \text{proportion of time good samples missed} \\
&= \tfrac{\Delta s + \Delta e}{N}.
\end{aligned}$$

$$\begin{aligned}
\text{shift-left level} &= \text{proportion of time noisy samples added and good samples missed / 2} \\
&= \tfrac{\Delta s + \Delta e}{2 * N}.
\end{aligned}$$

$$\begin{aligned}
\text{shift-right level} &= \text{proportion of time noisy samples added and good samples missed / 2} \\
&= \tfrac{\Delta s + \Delta e}{2 * N}.
\end{aligned}$$

3.3 Annotation Noise Statistics from A Real Crowdsourcing Experiment

To give a flavor of typical values encountered for the annotation noise levels, we report these levels measured in a real crowdsourcing experiment that we conducted in a previous study (Nguyen-Dinh et al., 2013c). In the crowdsourcing experiment we used video footage belonging to the Opportunity data set (Roggen et al., 2010), which contains gestures of normal daily routines (e.g., drink, open or close doors). We showed each short video to ten workers in Amazon Mechanical Turk (AMT), described the task and collected their annotations. The AMT labelers must annotate the start, end boundaries and the label of all occurrences of gestures of interest in the videos. We applied two strategies to detect and filter non-serious labelers and erroneous labeling (Nguyen-Dinh et al., 2013c). Individual filtering checks the correctness in the answers of each labeler for qualification questions whose answers are known in advance. Collaborative filtering checks the agreement in annotations among workers to detect non-serious labelers. Specifically, the labeler X who has a disagreement score $\mathrm{d(X)} = \frac{\text{Annotation times of X disagree with majority}}{\text{Total annotation times of X}} > \textit{threshold}$ is a spammer. We chose a threshold $= 0.3$, it means if the disagreement score $\mathrm{d} \geq 0.3$ (i.e., less than 70% of annotation of a labeler agrees with the majority), the labeler is a spammer and his annotations are removed. The collaborative filtering is illustrated in Figure 2. After filtering, the majority voting among qualified annotations is performed to generate a final crowdsourced gesture annotation. A more detail on the crowdsourcing experiment is given in Nguyen-Dinh et al. (2013c).

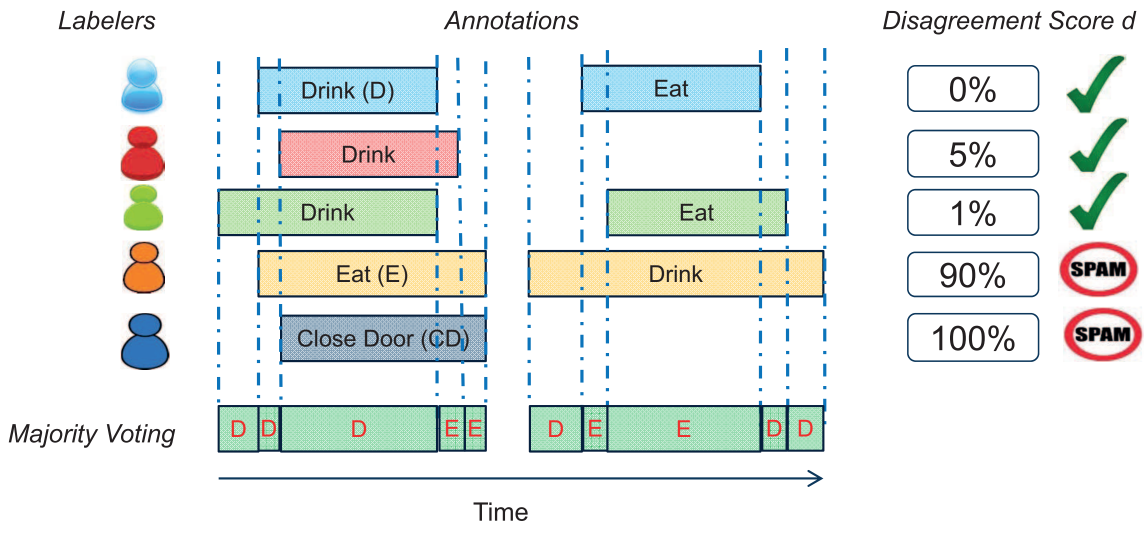

Figure 2: An illustration of the collaborative filtering technique to calculate the disagreement score of each labeler against the majority. The last two labelers are spammers and then their annotations will be removed.

Each video footage of the Opportunity data set was already examined and annotated carefully by one expert (Roggen et al., 2010) and the expert's annotations are used as a ground truth to evaluate our crowdsourced annotation. Here we report the sample-based accuracy (i.e., fraction of correctly labeled samples compared to expert's annotation) for a one-labeler annotation scenario where only one crowdsourced labeler is selected, and for a multiple-labeler scenario where the filterings and majority voting are applied for the ten workers. For a one-labeler annotation, the sample-based accuracy gets as low as 55%. In the multiple-labeler annotation, the accuracy reaches 80%. A breakdown of the types of annotation mistakes, according to the taxonomy introduced in Section 3.1, is shown in Figure 3a. The values for *label noise* and for the *boundary jitter* are shown for one and for multiple labelers. In the scenario of only one labeler, about 52% of the instances are affected by *label noises*, comprising mostly *substitute* and *delete* errors. In the multiple-labeler scenario, *label noise* decreases to 18%. In Figure 3b, we give the average, the min and the max values of jitter level of boundary jitters for one and for multiple labelers. On average, jitter levels ranges from 27% to 60%. However, there are good annotated instances with very low jitter levels (only 2%).

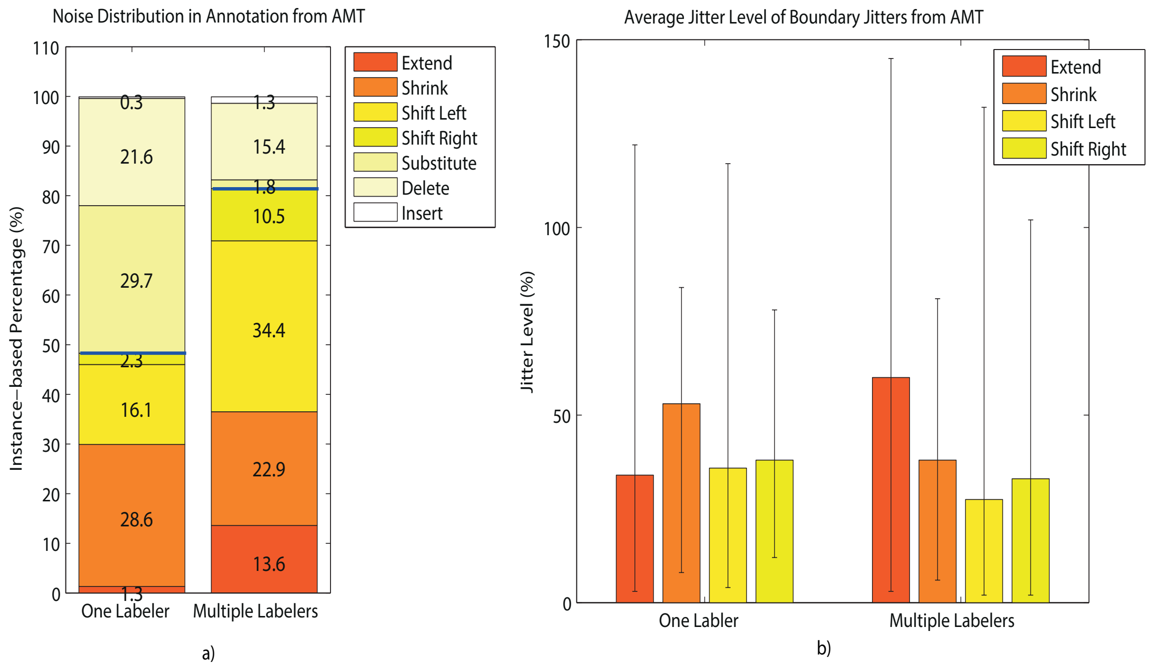

Figure 3: Analysis of crowdsourcing annotation from AMT. Blue lines in the figure a separate boundary jitter part and label noise part. Black lines in the figure b show the minimum and maximum level of jitter in each type of noise.

It can be seen that requesting multiple labelers for an annotation task can reduce labeling errors. However, the result from a one-labeler annotation represents for the scenarios where multiple labelers cannot be applied. Our experiment belongs to the first crowdsourcing category described at the beginning of the present section, i.e., crowdsourcing labeling of data which were previously recorded. The amount and distribution of annotation noises will change depending on the crowdsourcing scenario and on the kind of gesture data, but there is no reason to think that some scenarios will achieve much lower noise levels. On the contrary, in real-time annotation (i.e., providing labels while recording data) , it is more likely that the level of noise increases: more gestures could be forgotten and others would be labeled only after they really occurred, leading to imprecise time boundaries. We therefore argue that annotation noise is a fact that cannot be completely removed and that calls the attention of robust methods when designing gesture recognition systems which use noisy crowdsourced annotations.

In the next sections we present our SegmentedLCSS and WarpingLCSS TMMs which are designed with the aim of being robust to annotation noise for gesture recognition.

4. SegmentedLCSS and WarpingLCSS Gesture Recognition Methods

In this section, we describe in details our proposed methods, Segmented LCSS and WarpingLCSS for online gesture recognition using signals obtained from body-worn sensors.

The methods proposed to recognize gestures are based on template matching (TM). The training phase uses a set of labeled signals to train the gesture recognition algorithm. In the training phase, the sensor signals are quantized and converted into sequences of symbols (strings); furthermore, one template is created for each gesture of interest. When deploying the recognition algorithm, the quantization scheme is again applied to the streaming signals. The strings obtained are then compared with the learned templates by either using the longest common subsequence (LCSS) algorithm in segmented windows (SegmentedLCSS) or using our faster variant of LCSS (namely WarpingLCSS). Figure 4 shows the data flow through different processing components in the training phase and the recognition phase of our proposed system.

The rationale using LCSS is that it gives a measure of similarity between templates and signals to be recognized. Moreover, LCSS is robust to the high variability in gesture execution as shown in our previous work (Nguyen-Dinh et al., 2012) because LCSS can ignore the dissimilarity and accumulate the similarity between two gesture instances.

In the following, we first briefly review LCSS, then we describe the different processing components of the recognition system in Figure 4.

4.1 The Longest Common Subsequence Algorithm (LCSS)

Let s_A and s_B be two strings comprising l_A and l_B symbols respectively. Let $s(i)$ denote the i-th symbol within a string s. For each pair of positions $0 \leq i \leq l_A$ and $0 \leq j \leq l_B$ within the strings, we call $LCSS_{(A,B)}(i,j)$ the length of the longest symbol subsequence in common between the first i symbols of s_A and the first j symbols of s_B. The LCSS between the complete strings is then denoted as $L_{(A,B)}$ or, when the strings are clear from the context, just with L.

$$L_{(A,B)}(i,j) = \begin{cases} 0 \quad \text{, if } i=0 \text{ or } j=0 \\ \\ L_{(A,B)}(i-1,j-1)+1 \quad \text{, if } s_A(i)=s_B(j) \\ \\ \max \begin{cases} L_{(A,B)}(i-1,j) \\ L_{(A,B)}(i,j-1) \end{cases} \quad \text{, otherwise.} \end{cases} \tag{1}$$

Let Ω_A and Ω_B be the sets of indices corresponding to the longest subsequences of s_A and s_B that are matching. The sets $\Omega_A = \omega_A^{(0)} \ldots \omega_A^{(L-1)}$ and $\Omega_B = \omega_B^{(0)} \ldots \omega_B^{(L-1)}$ contain then $L_{(A,B)}$ indices. $L_{(A,B)}$ and the corresponding matching subsequences, hence the sets Ω_A and Ω_B, can be found using dynamic programming (see Cormen et al., 2001).

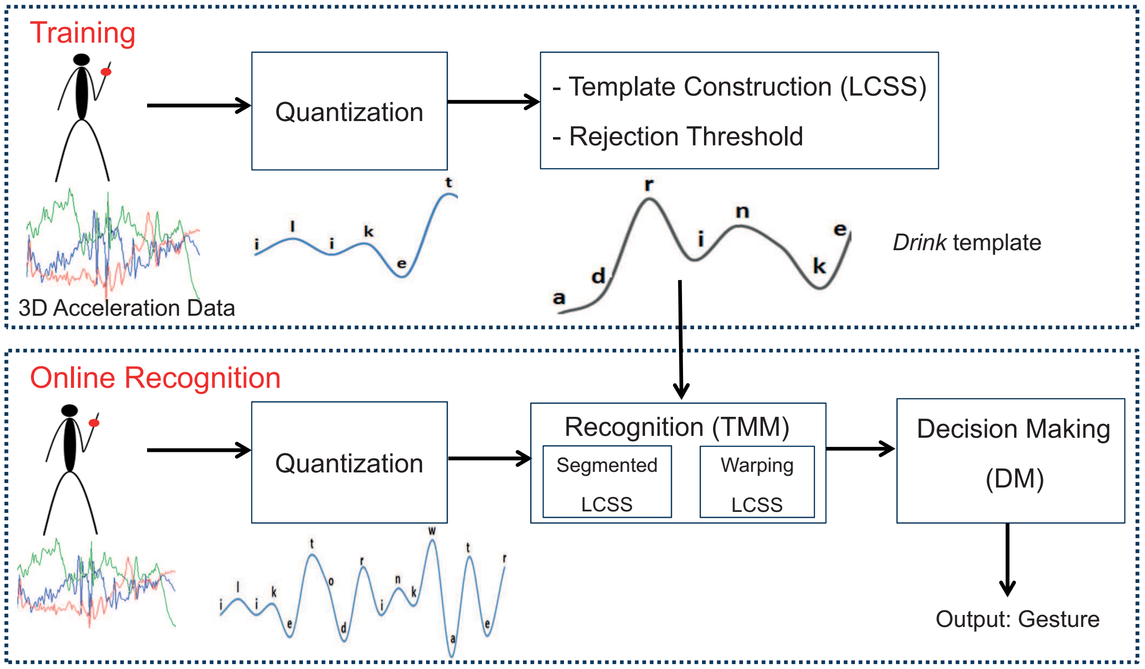

Figure 4: Data processing flow of the proposed LCSS-based template matching methods for gesture recognition.

4.2 Training Phase: Quantization Step

Let n denote the number of signal channels provided by the body-worn sensors (e.g., $n = 3$ for one triaxial accelerometer). Let N be the number of available samples. Let x_i be the time series corresponding to the i-th signal channel, with $1 \leq i \leq n$ and $x_i(t)$ be the value of the time series x_i at time t, with $1 \leq t \leq N$. Let the n-dimensional vector $\mathbf{x}(t) = [x_1(t) \dots x_n(t)]$ denote one sample from all channels at time t.

The quantization step converts the vectors $\mathbf{x}(t)$ into a sequence of symbols (string) $\mathbf{s}(t)$. This is done by performing k-means clustering on the set of n-dimensional vectors $\mathbf{x}(t)$, $\forall t, 1 \leq t \leq N$. The choice of k is performed through cross-validation or empirically. For the gesture data sets used in this paper, $k = 20$ provided a good tradeoff between complexity (k-means' complexity scales linearly with k) and performance. The output of k-means is a set of k n-dimensional cluster centers, $\zeta_\mathbf{0} \dots \zeta_\mathbf{k-1}$, to which k symbols $\alpha_0 \dots \alpha_{k-1}$ are assigned. The quantization procedure then operates on each sample $\mathbf{x}(t)$ to obtain the symbols $s(t)$ as follows:

$$s(t) = \alpha_i | i = \underset{i}{\operatorname{argmin}} \, ||\mathbf{x}(t) - \zeta_\mathbf{i}||_2 \quad .$$

Let us denote with $d(\alpha_l, \alpha_m)$ the distance between two symbols, given by the correspondent distance between their assigned cluster centers, normalized to fall in the interval $[0, 1]$.

$$d(\alpha_i, \alpha_j) = \frac{||\zeta_\mathbf{i} - \zeta_\mathbf{j}||_2}{max_{i,j}||\zeta_\mathbf{i} - \zeta_\mathbf{j}||_2} \quad . \tag{2}$$

4.3 Training Phase: Template Construction

For each labeled gesture in the training data set, a corresponding string is derived used the quantization described in Section 4.2. Denote with $s_i^{(c)}$ the i-th string belonging to class c. The template $\bar{s}^{(c)}$ that represents a gesture class c is then chosen as the string that has the highest average LCSS to all other strings of the same class.

$$\bar{s}^{(c)} = \underset{s_i^{(c)}}{\operatorname{argmax}} \sum_{j \neq i} L_{(s_i^{(c)}, s_j^{(c)})} \quad .$$

4.4 Training Phase: Calculation of Rejection Thresholds

In order to be able to reject signals not belonging to a gesture class upon deployment, a threshold needs to be calculated in the training phase. We define one rejection threshold ϵ_c for each class c. Let $\mu^{(c)}$ and and $\sigma^{(c)}$ be the mean and the standard deviation, respectively, of LCSS values between the template of a class c and any string belonging to the same class. We calculate the rejection threshold to be below $\mu^{(c)}$ by some standard deviations.

$$\epsilon_c = \mu^{(c)} - h * \sigma(c),$$

with h = 0,1,2,...

The rationale is that the good instances belonging to a class should have the similarity with the template around the mean value. ϵ_c is also chosen to be robust with the existence of noisy training instances in gesture class. In our experiments, $h = 1$ provided a good performance.

4.5 Recognition Phase: Quantization Step

In the online recognition, streaming data from a body-worn sensor are quantized to the k-means centroids (i.e., symbols) identified during training, then come to template matching module (TM) which uses either Segmented LCSS or WarpingLCSS to recognize gestures.

4.6 Recognition Phase: SegmentedLCSS

In the SegmentedLCSS approach, the sensor readings $\mathbf{x}(t)$ are first quantized into a string s through the quantization step described in Section 4.5. For each gesture class c, the string s is then segmented into a sliding observation window OW_c. The length of OW_c is chosen as the length of the template $\bar{s}^{(c)}$. A substring of s in OW_c is denoted as s^c_{OW}. Each substring is compared to the template $\bar{s}^{(c)}$ for class c.

The LCSS algorithm is used to calculate $L_{(s^c_{OW}, \bar{s}^{(c)})}$ and the set Ω_s of reference indices of the symbols of s^c_{OW} in the string s matching with symbols in the template. Because the LCSS algorithm can find matching points, the boundaries of the detected gesture can be decided easily. Specifically, if $L_{(s^c_{OW}, \bar{s}^{(c)})} \geq \epsilon_c$, the symbols ranging from $s(\omega_s^{(0)})$ and $s^c(\omega_s^{(L-1)})$ are marked as belonging to class c.

In order to reduce the computational complexity, the next observation window is started at the index $\omega_s^{(0)}$ of the first matching symbol of the previous observation window. In case the set Ω_s is empty, the next observation window is shifted quickly by the window length. Figure 5 illustrates the SegmentedLCSS.

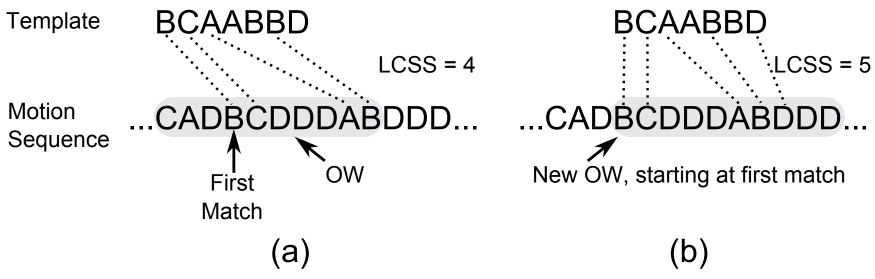

Figure 5: The SegmentedLCSS recognition process. The shaded part represents the observation window OW_c. For each class c, the LCSS is computed between the gesture template $\bar{s}^{(c)}$ and the quantized signal in the window. If the LCSS exceeds the rejection threshold, the samples between the first and the last matching symbols are assigned to class c. The next observation window will start at the first matched point of the previous calculation as illustrated in Figure b.

4.6.1 Computational Complexity of SegmentedLCSS

Let T_c denote the length of a gesture template of class c ($|OW_c| = T_c$). The worst case computational complexity of SegmentedLCSS occurs when new observation windows are shifted by just one sample compared to the preceding ones. In this case, for each class c, the time complexity of SegmentedLCSS is $\mathcal{O}(T_c^2)$. The overall time complexity is then $\mathcal{O}(C * \overline{T}^2)$, where C is the number of classes and $\overline{T}$ stands for the average template length across the classes. The memory usage in SegmentedLCSS is at most $\mathcal{O}(T^2)$, where T is the length of the longest template.

4.7 Recognition Phase: WarpingLCSS

In the SegmentedLCSS, the LCSS must be recomputed every time the observation window shifts, in order to find the beginning and end of each gesture. WarpingLCSS is our variant of LCSS that can find the gesture boundaries without the need of sliding windows, thereby reducing the computational complexity.

In WarpingLCSS, after each new sample of $\mathbf{x}(t)$ is available, the string s is updated by appending the symbol obtained through the quantization of the sample and the LCSS value is recomputed accordingly, relying on the previous values.

Given the gesture template for class c, $\bar{s}^{(c)}$, the WarpingLCSS score $W_{(\bar{s}^{(c)},s)}(i,j)$ between the first i symbols of the template $\bar{s}^{(c)}$ and the first j symbols of the string s is obtained through a modified version of Equation 1 as follows.

$$W_{(\bar{s}^{(c)},s)}(i,j) = \begin{cases} 0 & \text{, if } i = 0 \text{ or } j = 0 \\ \\ W_{(\bar{s}^{(c)},s)}(i-1,j-1)+1 & \text{, if } \bar{s}^{(c)}(i) = s(j) \\ \\ \max \begin{cases} W_{(\bar{s}^{(c)},s)}(i-1,j-1) - p * d(\bar{s}^{(c)}(i), s(j)) \\ W_{(\bar{s}^{(c)},s)}(i-1,j) - p * d(\bar{s}^{(c)}(i), \bar{s}^{(c)}(i-1)) \\ W_{(\bar{s}^{(c)},s)}(i,j-1) - p * d(s(j), s(j-1)) \end{cases} \\ & \text{, otherwise,} \end{cases} \tag{3}$$

where p is a penalty parameter of the dissimilarity and $d(\cdot,\cdot)$ is the distance between two symbols as defined in Equation 2. The rationale of the WarpingLCSS is the following: if the WarpingLCSS algorithm encounters the same symbol in a template and in the current string, W is increased by a reward of 1. Otherwise, W is decreased by a penalty which depends on the parameter p and on the distance between the symbols. Furthermore, if the string s is "warped", that is, it contains contiguous repetitions of a symbol due to a slower execution of a gesture, the penalty is counted only once.

The algorithm starts with an empty string s and $W(0,0) = 0$. As new symbols are appended, W is updated according to Equation 3. If a gesture of a class is performed, it symbols matching the corresponding template are found and W grows, until reaching a local maximum and eventually decreasing again, as soon as the gesture is over. A gesture of class c is recognized for each local maximum of W that also exceeds the rejection threshold ϵ_c. The end point of the gesture is set to the local maximum itself. The start point is 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).

When gestures differ from those encoded by the stored templates, W drops significantly due to the penalty terms. The value of the penalty parameter p depends on the application and can be chosen by cross-validation to maximize the recognition accuracy.

Figure 6 illustrates an example of behavior of W. Figure 7 shows a close-up of W where a gesture was matched to a template. It also shows how the WarpingLCSS detects the temporal boundaries of matched gestures.

4.7.1 Computational Complexity of WarpingLCSS

WarpingLCSS only needs to update the value of W for each new sample. Thus, the time complexity of WarpingLCSS is $\mathcal{O}(T)$. WarpingLCSS has a linear complexity in T compared to SegmentedLCSS, whose complexity grows quadratically in T. The WarpingLCSS maintains at most $\mathcal{O}(T^2)$ memory for the need to trace back the starting boundary of detected gestures.

4.8 Decision Making and Solving Conflicts

The incoming streaming string is concurrently "compared" with templates of all concerned gesture classes in TM module. If a gesture is spotted as belonging to multiple classes (i.e., boundaries of spotted instances are overlapping), the decision making module (DM) will

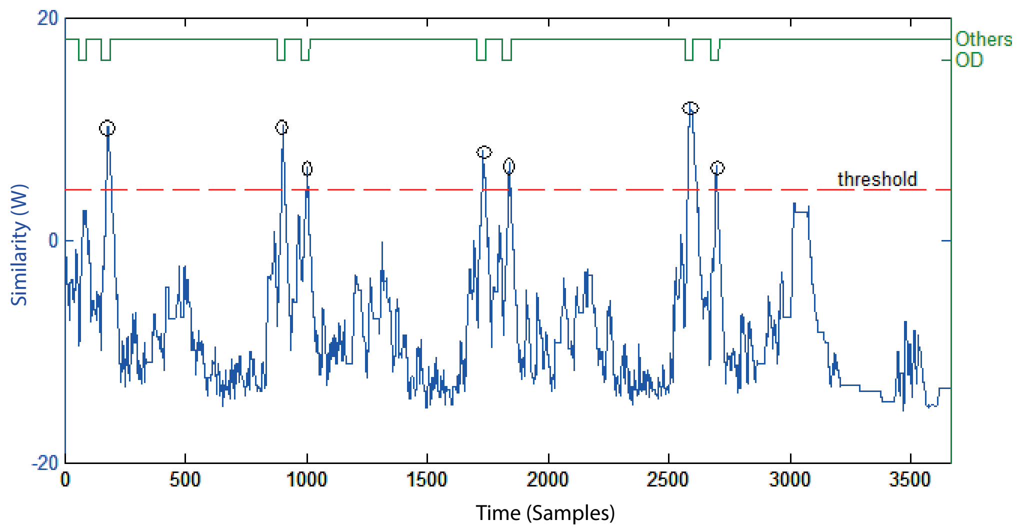

Figure 6: WarpingLCSS between a template of the gesture "open door" (OD) and a streaming string s, p = 3. The value W is updated for each new sample. The line on the top shows the ground truth. The small circles show gesture detection at spotting time.

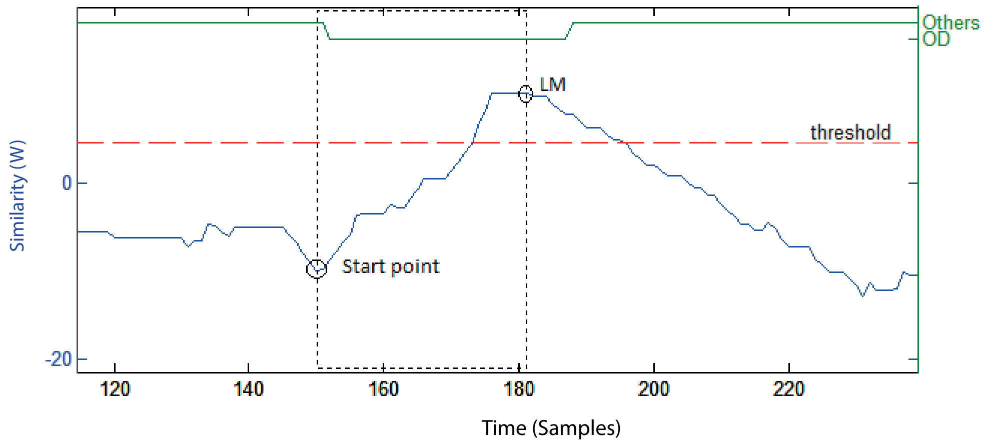

Figure 7: Close-up of the first detected "open door" gesture (OD) in the string s (see Figure 6). The local maximum (LM) marks the end of the gesture, while the start is traced back through the matching symbols.

resolve conflicts (as discussed below) by deciding which class is the best match. If a gesture is classified into only one gesture class, the DM will output the class. Otherwise, if no gesture class is spotted, the DM will output *null*.

Resolving spotting conflicts: We define the normalized similarity between two strings A and

B as 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 SegmentedLCSS and WarpingLCSS.

5. Experiments

To analyze the effect of annotation noise in terms of performance of gesture recognition algorithms, we compare our SegmentedLCSS and WarpingLCSS TMMs against state-of-the-art recognition methods to assess their robustness. We first present three gesture data sets used to evaluate the recognition systems. We then describe how synthetic crowdsourced annotations are obtained. Finally, we discuss baseline methods and evaluation metrics.

5.1 Description of Data Sets

We used three data sets including various gestures which have been labeled manually by experts. The experts' annotation is the ground truth of the data sets. The data sets used also include *null* class, data which do not correspond to any of the gestures of interest. The list of gestures of these data sets are shown in Table 1. In each data set, we use a 3D accelerometer at a subject' dominant (right) lower arm for the evaluations (30Hz sampling rate). Following, we describe briefly each data set[3].

HCI Gestures				
Circle	Triangle	Square	Infinity	Slider
Their Speculars				Null

Opportunity Gestures		
Null	clean Table	open Drawer 1-2-3
close Drawer 1-2-3	open Door 1-2	close Door 1-2
open Fridge	close Fridge	open Dishwasher
close Dishwasher	drink Cup	toggle Switch

Skoda Gestures			
write on notepad	check gaps on the front door	open hood	close hood
open left front door	close left front door	close both left door	check trunk gaps
check steering wheel	open and close trunk	Null	

Table 1: Gestures in Opportunity, Skoda, and HCI data sets.

5.1.1 Skoda

The Skoda data set (Zappi et al., 2008) contains 10 manipulative gestures performed in a car maintenance scenario by one subject. The *null* class takes 23%. Each gesture class has about 70 instances. This data set is characterized as low variant in execution because the subject performed carefully each manipulative gesture in the same manner.

3. Skoda and Opportunity data sets can be downloaded from `http://www.wearable.ethz.ch/resources/Dataset`.

5.1.2 HCI

The HCI data set (Banos et al., 2012) contains 10 gestures executed by a single person. The gestures are geometric shapes executed with the arm in the vertical plane. This data set has a low variability in the execution of gestures and well-defined labeling. The *null* class takes 57% and each gesture class has about 50 instances.

5.1.3 Opportunity

The Opportunity data set (Roggen et al., 2010) is a rich multi-modal data set collected in a naturalistic environment akin to an apartment, where users execute 16 daily gestures. The data set is characterized by a predominance of *null* class (37%) and a large variability in the execution of the daily activities. Each gesture class has 20 instances excepts "Drink Cup" and "Toggle Switch" each having 40 instances. Note that in Opportunity data set, there are three drawers at different heights which makes the recognition more challenging.

5.2 Experiments on Synthesized Crowdsourced Annotation

To analyze how much noise in annotation the gesture recognition methods can tolerate, we conduct experiments with synthesized annotations. We modify clean annotations from the three data sets described above by emulating *label noise* and *boundary jitter* as discussed in the taxonomy in Section 3.1. In order to evaluate the effect of the different types of noise, we run simulations for each type of noise separately.

5.2.1 Label Noise Simulation

In the *label noise* simulation, we assume the label boundaries are perfect. Let α be the label noise percentage in each class. This means that α percent of the instances are selected and their labels are randomly flipped to other classes (including *null class*). Consequently, each gesture class will have $(1 - \alpha)$ percent of clean instances.

5.2.2 Boundary Jitter Simulation

We run different simulations for different error types in boundary jitter. We assume that all gesture instances get affected from boundary jitter. Let β be the *jitter level* defined in Section 3.2. In the *extend* simulation, each gesture instance will have an *extend level* of β, with boundaries extended at both ends equally ($\beta/2$ per side). Similarly, in the *shrink* simulation, each gesture instance will be shrunk at both ends equally by $\beta/2$. In the *shift left* and *shift right* simulations, each gesture instance is shifted to the left or to the right respectively by β compared to the correct starting point.

We assume that all gesture instances have the same jitter level β. This assumption is not realistic however it can show how much jitter level in the training data set the spotting methods can tolerate given the same style of annotation (for example, a labeler always extends all his annotation 20% level). For a more realistic scenario where jitter levels vary from one instance to another instance, the experiment on the real crowdsourced annotation is presented in Section 6.2.

5.3 Evaluation with Baseline Methods

To investigate the effect of noisy crowdsourced data sets on gesture recognition, we compare the performance of recognition methods trained with ground truth annotations against those trained with crowdsourced annotations. With crowdsourcing-based experiments, the recognition system is trained on crowdsourced annotations and tested on clean data (i.e., annotated by experts). For each data set, we perform a 5-fold cross-validation.

We compare our proposed LCSS-based TMMs with three baselines approaches: the Segmented DTW (Ko et al., 2005; Hartmann and Link, 2010), Nonsegmented DTW (Stiefmeier et al., 2008) and support vector machines (SVM). For all TMM methods, we use the same strategy to select templates, i.e., the maximum similarity average for our LCSS-based methods and the minimum distance average for DTW-based ones. They all have the same quantization preprocessing step as presented in Section 4.2. The rejection thresholds are selected as discussed in Section 4.4. For SegmentedLCSS and Segmented DTW, the window length is chosen as the template length.

For SVM, the signals are passed through a sliding window, with 50% overlap. For each window, mean and variance of the signals are calculated and the obtained feature vectors are fed into a SVM classifier. We use RBF kernels and the two RBF parameters are selected by using cross-validation. In this work, we use the LIBSVM library (Chang and Lin, 2011) for training SVM.

5.3.1 Complexity of Baseline Methods

Segmented DTW belongs, like Segmented LCSS, to the category of sliding window based template matching algorithms. Therefore, roughly, they have the same computational cost. However, unlike SegmentedLCSS, in SegmentedDTW the boundaries of the gestures must be swept exhaustively in the observation window and DTW must be recomputed for each choice to find the best match (Ko et al., 2005; Hartmann and Link, 2010). Therefore, when one new sample arrives, the complexity of the SegmentedDTW is $\mathcal{O}(T^3)$ in the worst case. Meanwhile, in SegmentedLCSS the boundary of gesture inside the window can be found easily via matching points and the observation window is shifted to the first matched point in the previous recognition process instead of being shifted forward by only one sample. Thus, SegmentedLCSS has one order of magnitude lower than SegmentedDTW.

Nonsegmented DTW and WarpingLCSS determine gesture occurrences without segmenting the stream. Therefore, they achieve the same computational cost and they are faster than SegmentedLCSS by one order of magnitude.

In the recognition phase, the running time of SVM grows linearly with the length of the window. Hence, SVM has roughly the same computation cost as WarpingLCSS in the recognition phase.

5.4 Evaluation Metrics

The distribution of the gesture classes may be highly unbalanced in real-life data sets. Especially, in our data sets, *null class* is predominant. Therefore, we assess the performance of gesture recognition with the weighted average F1 score. The weighted average F1 score is the sum of the F1 scores of all classes, each weighted according to the proportion of samples of that particular class. Specifically,

$$F1score = \sum_{c} 2 * w_c \frac{precision_c * recall_c}{precision_c + recall_c},$$

where c is the class index and w_c is the proportion of samples of class c; $precision_c$ is the proportion of samples of class c predicted correctly over the total samples predicted as class c; $recall_c$ is the proportion of samples of class c predicted correctly over the total samples of class c.

We present two ways of computing the F1 score, either including (F1-Null) or excluding the *null class* (F1-NoNull). F1-NoNull does not consider the *null class*, but still takes into account false predictions of gesture samples or instances misclassified as *null class*. The recognition system that has high values of both F1-Null and F1-NoNull predicts well both gesture classes and *null* class.

6. Results and Discussion

In this section we present and discuss the results of the experiments conducted with synthesized and real crowdsourced annotations.

6.1 Results on Synthesized Crowdsourced Annotations

We first present the results with synthesized crowdsourced annotations, sweeping the noise levels as described in Section 5. The results show that F1-Null and F1-NoNull have a similar trend of performance as the noise levels increase, therefore we report F1-Null score only.

6.1.1 Label Noise Simulation

Figure 8 shows the results of label noise simulations on the three data sets. WarpingLCSS and SegmentedLCSS are more robust against label noise compared to SVM and DTW-based methods. The performance of LCSS-based methods is stable until a label noise percentage (α) in each class exceeding 70% in Opportunity and HCI data sets and 50% in the Skoda data set. On average, WarpingLCSS outperforms SVM by 22% F1-Null and outperforms DTW-based methods by 36% F1-Null in presence of 60% mislabeled instances. SegmentedLCSS yields similar performance as WarpingLCSS.

SVM performs worse than our LCSS-based methods when α increases. As more label substitutions are added to each class, SVM gets more confused and its performance decreases quickly. The degradation of SVM in performance is expected, since each instance contributes equally to the model building. Hence, wrongly labeled instances can induce the model to choose incorrect support vectors, which severely degrades the performance. Moreover, since the SVM method models *null* class explicitly, it is very sensitive to *delete* noise. Meanwhile, TMMs examine patterns of gesture classes and ignore *null* class in the training phase, thus, TMMs are not influenced with the *delete* noise at all.

The reason why LCSS-based TMMs outperform the ones based on DTW lies in the distance metrics used when selecting the template for each class. Each template is chosen as the one with the highest average similarity to the other instances belonging to the same class. This translates into choosing respectively highest average LCSS and lowest average

DTW distance. While LCSS values between a template and an instance of the same class are bounded between 0 and the length of the template, DTW can grow indefinitely. For this reason, when calculating average DTW distances, mislabeled instances bias the average towards high values, regardless whether correctly labeled instances have a low DTW distance. Consequently, DTW-based TMMs are more likely to pick wrong templates, leading to poor performance when α increases.

The difference between LCSS and DTW in choosing templates can be illustrated with a toy-example. Consider three instances A_1, A_2 and B which are all labeled as belonging to class c_A but let B be mislabeled, that is, B actually belongs to class c_B. To simplify matters, let us assume $LCSS(A_1, A_2) = 1$, $LCSS(A_1, B) = 0$ and $LCSS(A_2, B) = 0$. Similarly, let us assume $DTW(A_1, A_2) = 0$, $DTW(A_1, B) = \infty$ and $DTW(A_2, B) = \infty$. With LCSS, A_1 would have an average similarity of .5 to A_2 and B; A_2 would have an average similarity of .5 to A_1 and B; B would have an average similarity of 0 to A_1 and A_2. Thus, LCSS would pick either A_1 or A_2 as template for the class c_A: both choices would be reasonable. With DTW, A_1 would have an average distance of ∞ to A_2 and B; A_2 would have an average distance of ∞ to A_1 and B; B would have an average distance of ∞ to A_1 and A_2. In this case, the algorithm would not prefer A_1 or A_2 over B, which can lead to choosing as template the mislabeled instance B to represent class c_A. Of course in practice the values of the DTW distance are not infinity, in fact the degradation of DTW-based approaches is not occurring already for a small amount of label noise.

The illustration explains the capability of our LCSS-based methods to pick a good template among noisy instances for a gesture class as long as the number of good instances in a gesture class is still predominant.

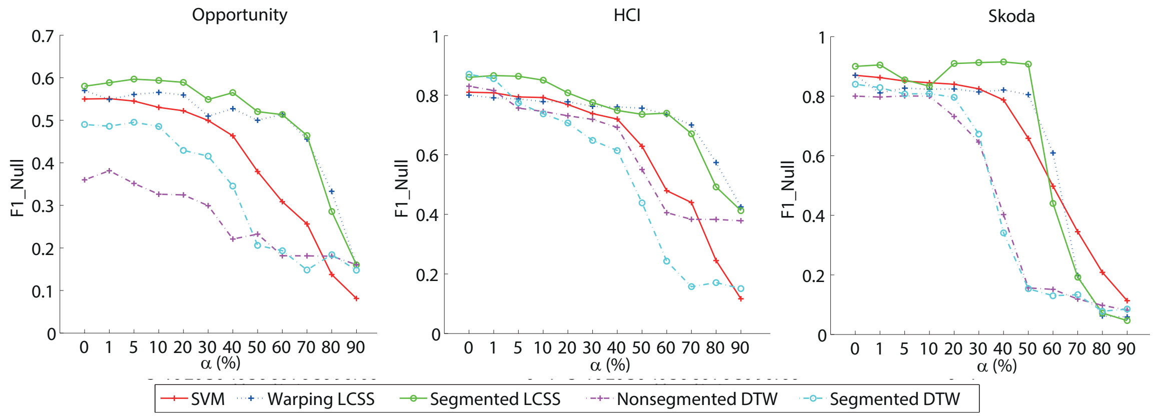

Figure 8: Performance of label noise simulation for the three data sets.

By analyzing the starting points of the curves of Figure 8, obtained with $\alpha = 0$ (no noise), we can conclude that our LCSS-based methods have a similar or better performance compared to the baselines also for the case of clean training data sets.

6.1.2 Extend Jitter Simulation

When temporal boundaries are extended, data belonging to the *null* class (before and after the gesture) are labeled as belonging to the gesture class. This impacts SVM and TMMs differently. In the case of SVM, the *null class* is modeled explicitly. The noisy feature vectors extracted from extended parts are added into the feature space of each gesture class. Besides that, the data really belonging to the gesture are preserved, thus the models of gesture classes maintain good feature spaces correctly. Therefore, the performance of SVM depends on how much the noisy feature vectors added into the model of each gesture class. Accordingly, it relies on the levels of variability of the signals belonging to the *null* class. If the variability of the signals belong to the *null* class is low, even when the extend level is large, the noisy feature vectors in each gesture class does not grow, leading to the stable of SVM performance. In the converse case, the noisy feature vectors in each gesture class will explode as the extend level increases, causing the decrease in the performance of SVM.

For TMMs instead the *null class* is recognized in the test data by means of the rejection threshold ϵ_c and no template is built for it. Thus, if symbols belonging to the *null class* are present in a test sequence, these will be matched to the symbols present in the extended gesture instances, inducing the TMMs to recognize gestures instead of *null class.*

This is confirmed by an analysis of the results, as shown in Figure 9. TMMs can tolerate up to about 40% *extend level* in the Opportunity and HCI data sets and about 10% *extend level* in the Skoda data set. As the extend level is high, the performance of SVM is stable in HCI and Skoda data sets, but degrades quickly in Opportunity data set. As explained above, the reason of the differences among data sets lie in the different levels of variability of the signals belonging to the *null class* in the different data sets.

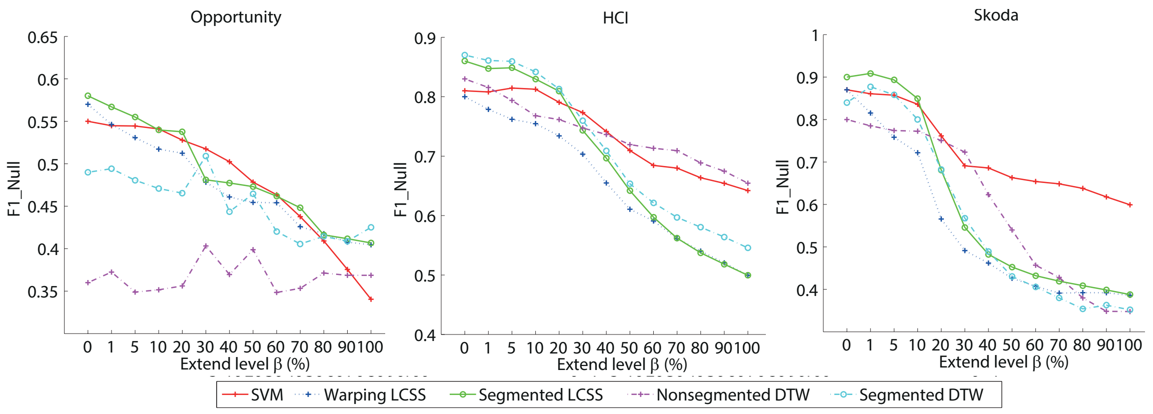

Figure 9: Performance of extend jitter simulation.

6.1.3 Shrink Jitter Simulation

When having a *shrink jitter* noise, the effect is that the methods lose information about the gesture data, since only parts of the gestures are labeled. This has a stronger effect in SVM, since the model is corrupted. For TMMs, subsequences are matched, with the effect that shrunk instances still contain information in form of shorter subsequences that can still be matched to the test data. This is confirmed by the results, shown in Figure 10.

Our proposed LCSS-based methods achieve the best performance in the three data sets. All methods can tolerate about 30% shrink level before a degradation compared to training

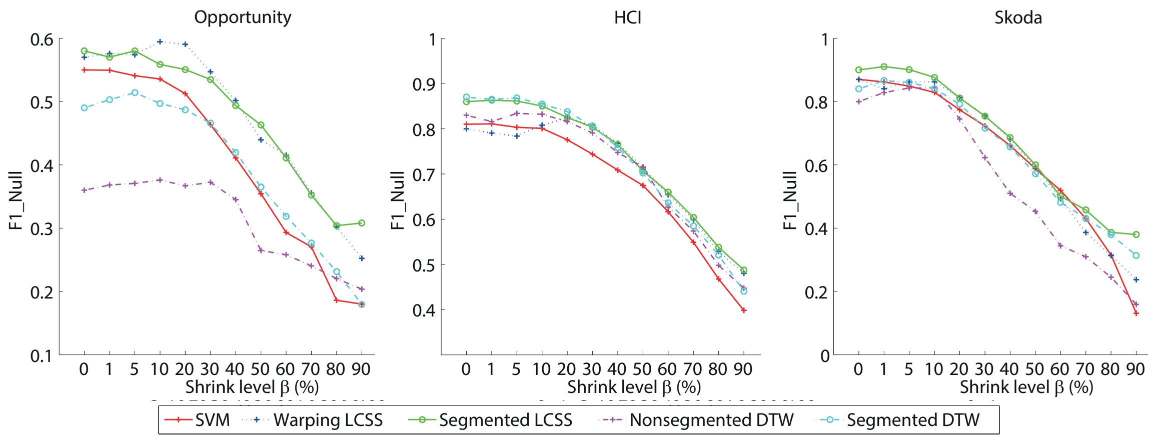

Figure 10: Performance of shrink jitter simulation.

with clean data occurs. The Segmented DTW has a similar results as LCSS-based methods in low-variability data sets (HCI and Skoda). However, Segmented DTW takes a higher computational cost. Moreover, in our experiments, all gesture instances have the same shrink level, i.e., after shrinking, instances of a gesture class are still aligned well and DTW can still achieve a reasonable performance. In a real crowdsourcing annotation setting, different instances may have different shrink levels (see Figure 3b). In that case, DTW will accumulate higher distances due to data misalignment at the beginning and the end of instances (see Nguyen-Dinh et al., 2012 for a more thorough discussion of the weakness of DTW with misalignment in temporal boundaries).

6.1.4 Shift-Left and Shift-Right Jitter Simulation

When annotations are shifted, a mixture of the effects described in Sections 6.1.2 and 6.1.3 are present. Some samples belonging to gestures are lost and some null class samples are labeled as belonging to a gesture. Figure 11 shows the results of *shift-right* jitter simulations (the *shift-left* simulations yield similar results). All methods can sustain about 20% *shift level* before the performance degrades compared to a clean training data set. LCSS-based methods perform often better, or as good as DTW-based methods on the data sets that we examined. TMMs outperform SVM with up to 30% *shift level*.

6.2 Results on Real Crowdsourced Annotation

To further validate the outcome of the previous experiments, we use the real crowdsourced annotations discussed in Section 3.3. The annotations were performed by AMT workers on the Opportunity data set. We use both the annotations obtained in the one-labeler and in the multiple-labeler scenarios. In these annotations, mixtures of all kinds of the errors listed in the taxonomy (Section 3.1) are present and jitter levels are varied from one instance to another instance (see Figure 3).

Figure 12 reports the performance of the different recognition methods on our real crowdsourced annotation. In the clean annotated Opportunity data set, the performance of SVM is slightly lower than that of LCSS-based TMMs (only lower by 3% for F1-Null

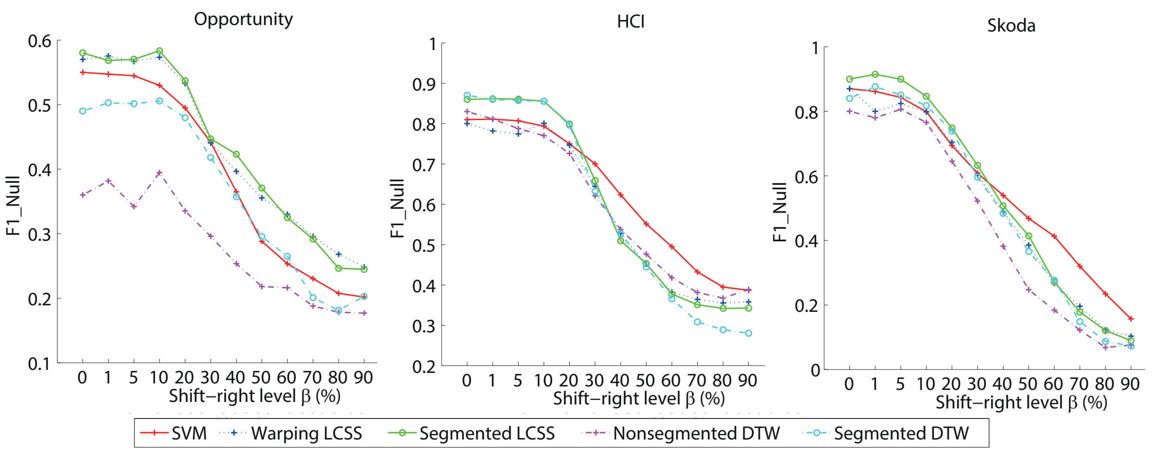

Figure 11: Performance of shift-right jitter simulation.

and by 7% for F1-NoNull). Two DTW approaches underperform the others. The reason is that DTW is very sensitive to high variation in gesture execution (Nguyen-Dinh et al., 2012) and the Opportunity data set contains large variability in the executions of the daily activities.

In the multiple-labeler annotation, labels of 80% of the data samples match the ground truth. Moreover, only 18% of gesture instances are labeled incorrectly and the remainder are correctly labeled with a *jitter level* of at least 2% (see Figure 3). The results show that the performances of all recognition methods are slightly decreased by up to 4% for F1-Null and 6% for F1-NoNull compared to the training with clean training sets. Our LCSS-based TMMs yield the best performance. As stated also in Section 6.1.1, the reason for the robustness of LCSS-based methods lies in their ability to select clean templates also in presence of annotation noise.

In the AMT one-labeler annotation, only 55% samples are annotated correctly. Additionally, about 50% of gesture instances are affected by *label noise*, with many deletions and substitutions. In each gesture class, instances which are labeled correctly are still the majority. The result shows that our LCSS-based TMMs still achieve the best performance. The F1-Null measure decreases by 10% and the F1-NoNull by 16% compared to training with clean annotations.

In the one-labeler annotation, there is a significant difference in performance between TMMs and SVM. The performance of SVM decreases dramatically, down to a F1-NoNull of 5%, which is less than random guessing (which would be around 6% in a 16-class data set like Opportunity). This result confirms what was already measured with the synthetic annotations and discussed in Section 6.1.1.

Additionally, we conduct a 2-sided hypothesis test at the 0.01 level of significance as in Guyon et al. (1998) among the performance of the methods in the three scenarios. The tests showed that the performance differences among the methods are statistically significant except the comparison of the F1-Null between SVM and WarpingLCSS and the comparison of the F1-NoNull between WarpingLCSS and SegmentedLCSS in the multiple-labeler annotation.

The results on the real crowdsourcing annotation confirm that our proposed WarpingLCSS and SegmentedLCSS are robust to noise and yield better performance on crowdsourcing data set. WarpingLCSS is preferable in online recognition, since it has a lower computational cost.

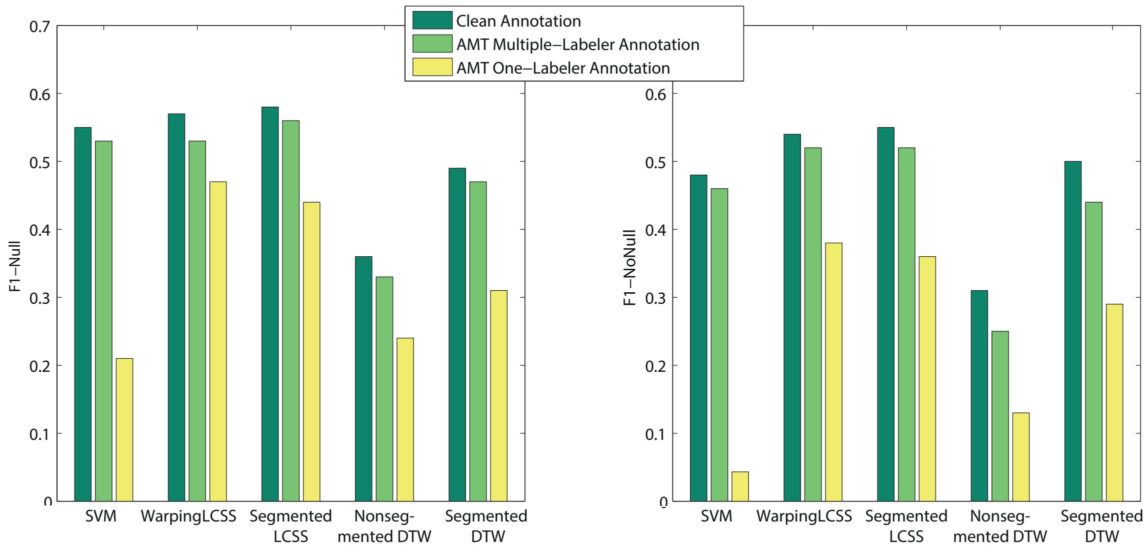

Figure 12: Performance of real crowdsourcing annotation on Opportunity data set.

6.3 A LCSS-based Filtering Component

The results have shown that SVM is very sensitive to the high *label noise* in the training data set. Therefore, a preprocessing component to clean the noisy annotation would be beneficial before using SVM. Given the robustness of our LCSS approaches in selecting templates among noisy instances, as well as in spotting, we further propose a LCSS-based filtering component to filter out noise in crowdsourced annotations before training a SVM. We call this approach FSVM. For each gesture class, the LCSS-based filtering component first computes a LCSS similarity matrix among all pairs of instances in the class. It then keeps only the instances that have an average similarity to other instances of the same class exceeding the average of all the average similarities of all instances in the class. To clean noise inside the *null* instances (e.g., *delete* noise), the filtering component runs the WarpingLCSS on the data annotated as *null* and discards any parts which get classified as any gestures of interest.

For DTW-based TMMs, the performance degrades quickly when the *label noise* percentage in the training data set increases (see Figure 8) because DTW cannot pick a good template among noisy instances. It is interesting to know how templates selected by LCSS perform in the DTW spotting methods. Therefore, we conduct experiments for Segmented DTW and Nonsegmented DTW with templates trained by LCSS. We call these approaches LCSS-SegDTW and LCSS-NonSegDTW respectively. Note that the algorithm running time when the system is deployed remains unchanged: only the training phase is affected.

The performances of FSVM, LCSS-SegDTW and LCSS-NonSegDTW are shown in Figure 13 for the real crowdsourced annotation and in Figure 14 for the synthetic label noise simulation. We present again the performances of the other methods that we discuss above for the sake of comparison.

In the real crowdsourced annotation, the filtering increases the performance of SVM by 20% F1-score and of DTW-based methods by 8% F1-score on average in the one-labeler annotation scenario where high *label noise* exists (see Figure 3). In the clean annotation and multiple-labeler annotation, FSVM performs just slightly worse than SVM (only 2%). This slight decrease can be explained with the fact that the FSVM method decreases the amount training data compared to pure SVM, because the LCSS-based filtering component in the FSVM removes some part of training data, considered noisy. Our proposed LCSS-based methods still outperform FSVM.

The LCSS-NonSegDTW outperforms Nonsegmented DTW in all three scenarios (expert's annotation, AMT multiple-labeler annotation and AMT one-labeler annotation). Similarly, LCSS-SegDTW outperforms SegmentedDTW. The result clarifies that LCSS is capable of picking a better template among noisy instances, compared to DTW. However, LCSS-NonSegDTW and LCSS-SegDTW still underperform compared to our LCSS-based TMMs. The rationale is the same as discussed before. LCSS is more robust to high variation in daily gesture execution, therefore LCSS-based spotting approaches have a better performance than DTW-based ones even with the same templates.

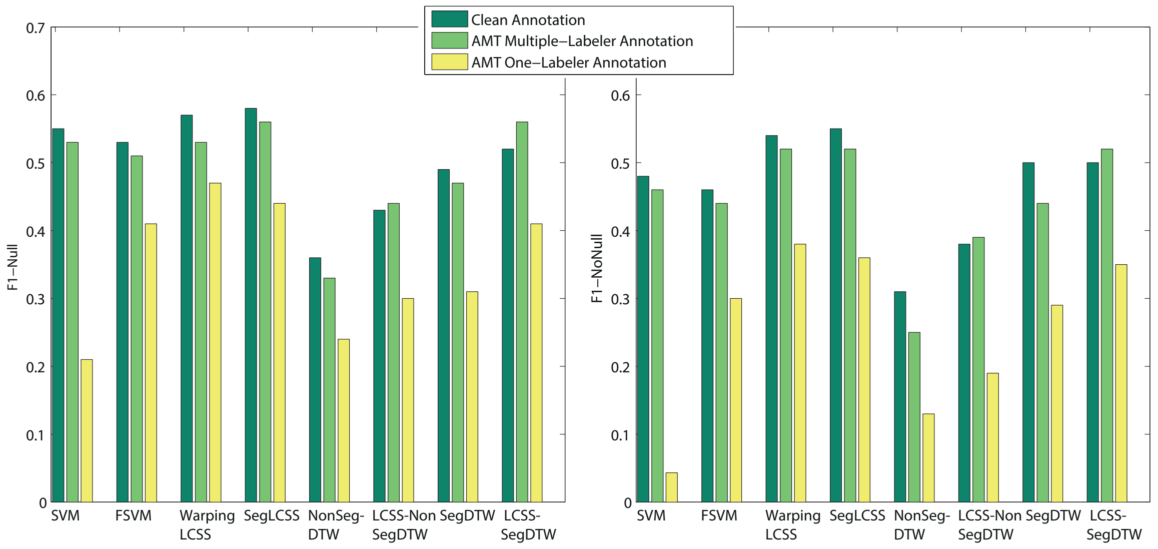

Figure 13: Performance of real crowdsourcing annotation on Opportunity data set for the methods with and without filtering. SegLCSS, NonSegDTW, and SegDTW stand for Segmented LCSS, Nonsegmented DTW and Segmented DTW respectively.

In the synthetic label noise simulation, the FSVM, LCSS-NonSegDTW and LCSS-SegDTW methods outperform SVM, Nonsegmented DTW and Segmented DTW respectively and keep the performance stable much longer when α increases. Our proposed LCSS-based TMMs have similar or better performance than the other methods. Interestingly, with the same templates picked by LCSS, LCSS-SegDTW and LCSS-NonSegDTW have a performance which is similar to our LCSS-based methods in the HCI and Skoda data sets. In the Opportunity data set, the LCSS-NonSegDTW still performs worse than our SegmentedLCSS and WarpingLCSS methods because LCSS is more robust than DTW to high variability in daily gestures (Nguyen-Dinh et al., 2012).

The results show that our LCSS approaches can be used in a preprocessing step for cleaning noisy annotation in the training data for SVM or for selecting templates for DTW-based TMMs.

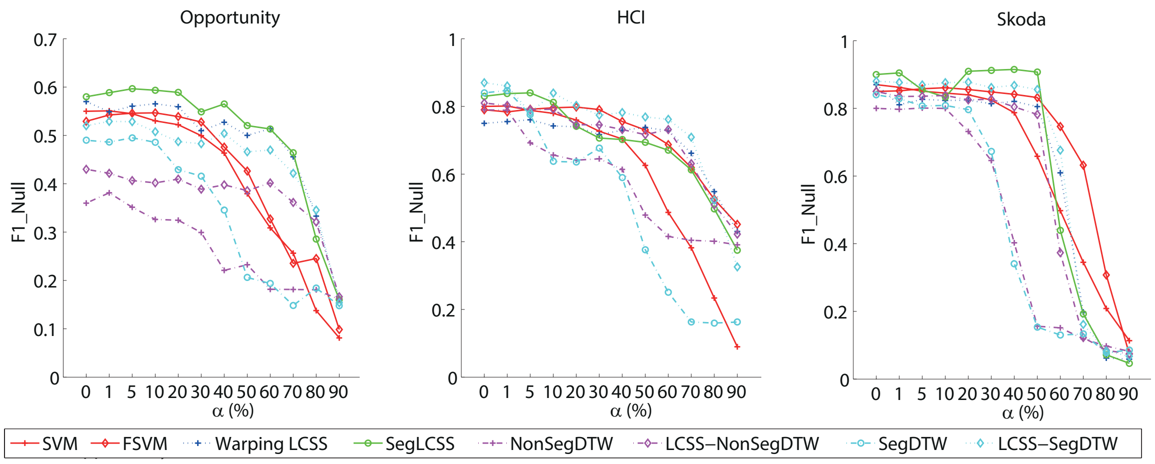

Figure 14: Performance of label noise simulation for the methods with and without filtering.

6.4 Wrapping up

Our LCSS-based TMMs are robust to labeling noise in crowdsourced gesture data sets. Moreover, the LCSS-based TMMs also offer other advantages. (1) They are easy to deploy in online gesture recognition system due to low time complexity. (2) In our systems, signals are converted into symbols, thus SegmentedLCSS lends itself even to embedded implementations. Specifically, string matching in the deployment phase does not involve floating-point operations, thus it can be deployed easily in cheap entry-level microcontroller units. (3) The deployed TMM-based systems are scalable to new gesture classes of interest. After collecting a training data set for a new class, the training phase only works with this class to find a template and the rejection threshold for the class. The template is then integrated directly into the deployed system. Thus, the whole process works smoothly with the new class without interfering with other existing gesture classes.

Our LCSS-based TMMs have been investigated in online gesture recognition with accelerometer data only. Their ability to work with other sensor modalities (e.g., gyroscopes, sound) has been investigated and it has shown promising preliminary results in Nguyen-Dinh et al. (2014).

7. Conclusion and Future Work

In this paper, we investigated the robustness of our proposed LCSS-based TMMs for online gesture recognition on crowdsourced annotated data sets. The results show that SegmentedLCSS and WarpingLCSS are robust to crowdsourced annotation noise and yield better performance than DTW-based methods and SVM. We also introduced a taxonomy of annotation noise in crowdsourcing settings and analyzed the distribution of that noise in real crowdsourced scenarios. Our LCSS-based methods are very robust to label noise because they are capable of selecting a good template among noisy instances for a class. In presence of 60% mislabeled instances, LCSS-based methods outperform SVM by 22% F1-score and outperform DTW-based methods by 36% F1-score on average.

With boundary jitter, the performance of the proposed approaches is comparable to that on clean data sets if annotations can keep most of the information indicating gestures (at most 30%-40% jitter level). In extreme cases when jitter levels go beyond that limit, our LCSS-based TMMS and the other machine learning techniques fail to recognize the complete segment of gestures. This can be the case for example in real-time labeling, where labelers tend to indicate quickly when a gesture occurs with only one time point, without providing the start and end time of the gesture (e.g., the boundary shrinks to a point). Other techniques (e.g., active learning) are necessary to acquire more labels and improve label quality in such cases.

We showed that our LCSS-based methods can be also used as a preprocessing filtering component to clean crowdsourced training data set with severe label noise before feeding the training sets into other learning techniques such as SVM or select templates for DTW. The filtering increases the performance of SVM by 20% F1-score and DTW-based methods by 8% F1-score on average in the noisy real crowdsourced annotations.

In future work, we plan to deploy the system that crowdsources annotated data to a large number of users who record and contribute gestures. Our methods will then be tested on such real large crowdsourced data sets, with the ultimate goal of having a collaborative database of gestures and associated models with direct applications with wearable sensors.

Acknowledgments

The authors would like to thank Dr. Daniel Roggen (University of Sussex) for his useful comments. This work has been supported by the Swiss Hasler Foundation project SmartDAYS.

References

J. Aggarwal and M. Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 43(3):16:1–16:43, April 2011.

J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff. A unified framework for gesture recognition and spatiotemporal gesture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1685–1699, Sept 2009.

R. Amini and P. Gallinari. Semi-supervised learning with an imperfect supervisor. *Knowledge and Information Systems*, 8:385–413, November 2005.

D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, April 1988.

O. Banos, A. Calatroni, M. Damas, H. Pomares, I. Rojas, H. Sagha, J. del R. Millán, G. Tröster, R. Chavarriaga, and D. Roggen. Kinect=imu? learning mimo signal mappings to automatically translate activity recognition systems across sensor modalities. In *Proceedings of the 2012 16th International Symposium on Wearable Computers (ISWC)*, pages 92–99, 2012.

L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *Proceedings of the 2nd International Conference on Pervasive Computing*, 2004.

B. Bauer and K. Karl-Friedrich. Towards an automatic sign language recognition system using subunits. In *International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction*, pages 64–75. 2002.

M. Berchtold, M. Budde, D. Gordon, H. Schmidtke, and M. Beigl. Actiserv: Activity recognition service for mobile phones. In *Proceedings of the 2010 14th International Symposium on Wearable Computers (ISWC)*, pages 1–8, Oct 2010.

R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady. A linguistic feature vector for the visual interpretation of sign language. In *European Conference on Computer Vision*, ECCV '04. 2004.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu. Sensor-based activity recognition. In *IEEE Transactions on Systems, Man and Cybernetics*, 2012.

H. Cooper, E.-J. Ong, N. Pugeault, and R. Bowden. Sign language recognition using subunits. *Journal of Machine Learning Research*, 13(1):2205–2231, July 2012.

T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. 2nd edition, 2001. ISBN 0070131511.

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979.

J. Deng and H. Tsui. An HMM-based approach for gesture segmentation and recognition. In *Proceedings of the International Conference on Pattern Recognition*, ICPR '00, 2000.

A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, April 2011.

M. Elmezain, A. Al-Hamadi, and B. Michaelis. Improving hand gesture recognition using 3D combined features. In *Proceedings of the 2nd International Conference on Machine Vision*, ICMV '09, pages 128–132, Dec 2009.

G. Fang, X. Gao, W. Gao, and Y. Chen. A novel approach to automatically extracting basic units from chinese sign language. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 4, pages 454–457, Aug 2004.

J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay. Myexperience: A system for in situ tracing and capturing of user feedback on mobile phones. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, MobiSys '07, 2007.

D. Frolova, H. Stern, and S. Berman. Most probable longest common subsequence for recognition of gesture character input. *IEEE Transactions on Cybernetics*, 43(3):871–880, June 2013.

T.-C. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, February 2011.

N. Gayar, F. Schwenker, and G. Palm. A study of the robustness of KNN classifiers trained using soft labels. In *Artificial Neural Networks in Pattern Recognition*, volume 4087. 2006.

I. Guyon, J. Makhoul, R. Schwartz, and V. Vapnik. What size test set gives good error rate estimates? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20 (1):52–64, Jan 1998.

J. Hao and T. Shibata. Digit-writing hand gesture recognition by hand-held camera motion analysis. In *Proceedings of the 3rd International Conference on Signal Processing and Communication Systems*, ICSPCS '09, pages 1–5, Sept 2009.

B. Hartmann and N. Link. Gesture recognition with inertial sensors and optimized DTW prototypes. In *Proceedings of the 2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*, 2010.

Z. He, L. Jin, L. Zhen, and J. Huang. Gesture recognition based on 3D accelerometer for cell phones interaction. In *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 217–220, Nov 2008.

J. Howe. The Rise of Crowdsourcing. (accessed July 20, 2010), jun 2006. URL `http://www.wired.com/wired/archive/14.06/crowds.html`.

P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, 2010.

H. Junker, O. Amft, P. Lukowicz, and G. Tröster. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition*, 41(6), 2008.

C. Keskin, A. Cemgil, and L. Akarun. DTW based clustering to improve hand gesture recognition. In *Proceedings of the 2nd International Conference on Human Behavior Unterstanding*, HBU'11, pages 72–81. 2011.

A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the Twenty-sixth SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, 2008.

M. H. Ko, G. West, S. Venkatesh, and M. Kumar. Online context recognition in multisensor systems using dynamic time warping. In *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference*, 2005.

W. S. Lasecki, Y. C. Song, H. Kautz, and J. P. Bigham. Real-time crowd labeling for deployable activity recognition. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1203–1212, 2013.

N. D. Lawrence and B. Schölkopf. Estimating a kernel fisher discriminant in the presence of label noise. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 306–313, 2001.

H.-K. Lee and J. H. Kim. An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973, October 1999.

L.-V. Nguyen-Dinh, D. Roggen, A. Calatroni, and G. Tröster. Improving online gesture recognition with template matching methods in accelerometer data. In *Proceedings of the 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, 2012.

L.-V. Nguyen-Dinh, U. Blanke, and G. Tröster. Towards scalable activity recognition: Adapting zero-effort crowdsourced acoustic models. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, 2013a.

L.-V. Nguyen-Dinh, M. Rossi, U. Blanke, and G. Tröster. Combining crowd-generated media and personal data: Semi-supervised learning for context recognition. In *Proceedings of the 1st ACM International Workshop on Personal Data Meets Distributed Multimedia*, PDM '13, 2013b.

L.-V. Nguyen-Dinh, C. Waldburger, D. Roggen, and G. Tröster. Tagging human activities in video by crowdsourcing. In *Proceedings of the ACM International Conference on Multimedia Retrieval*, ICMR '13, 2013c.

L.-V. Nguyen-Dinh, A. Calatroni, and G. Tröster. Towards a unified system for multimodal activity spotting: Challenges and a proposal. In *Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '14 Adjunct, 2014.

N. Ravi, N. D, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence(IAAI)*, pages 1541–1546. AAAI Press, 2005.

V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11, 2010.

D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Forster, G. Troster, and et al. Collecting complex activity data sets in highly rich networked sensor environments. In *Proceedings of the 7th International Conference on Networked Sensing Systems*. IEEE Press, 2010.

M. Rossi, O. Amft, and G. Tröster. Recognizing daily life context using web-collected audio data. In *Proceedings of the 16th IEEE International Symposium on Wearable Computers (ISWC)*, June 2012.

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*, 2008.

V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, 2008.

H. Stern, M. Shmueli, and S. Berman. Most discriminating segment - longest common subsequence (MDSLCS) algorithm for dynamic hand gesture classification. *Pattern Recognition Letters*, 34(15):1980–1989, 2013.

T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and G. Tröster. Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing Magazine*, 7(2), 2008.

M. Stikic, D. Larlus, S. Ebert, and B. Schiele. Weakly supervised recognition of daily life activities with wearable sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2521–2537, December 2011.

K. Van Laerhoven, D. Kilian, and B. Schiele. Using rhythm awareness in long-term activity recognition. In *Proceedings of the IEEE International Symposium on Wearable Computers (ISWC)*, October 2008.

C. Vogler and D. N. Metaxas. Toward scalability in ASL recognition: Breaking down signs into phonemes. In *Gesture-Based Communication in Human-Computer Interaction*, Lecture Notes in Computer Science, pages 211–224, 1999.

J. A. Ward, P. Lukowicz, and H. W. Gellersen. Performance metrics for activity recognition. *ACM Transactions on Intelligent Systems and Technology*, 2(1), January 2011.

A. Wilson and A. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, 1999.

J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li. Gesture recognition with a 3-D accelerometer. In *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing*, UIC '09, pages 25–38, 2009.

H.-S. Yoon, J. Soh, Y. J. Bae, and H. S. Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34, 2001.

M.-C. Yuen, I. King, and K.-S. Leung. A survey of crowdsourcing systems. In *Social-Com/PASSAT*, pages 766–773, 2011.

P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Tröster. Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection. In *Proceedings of the 5th European Conference on Wireless Sensor Networks*, EWSN'08, pages 17–33, 2008.