DISS. ETH NO. 23228

Wearable Activity Recognition with Crowdsourced Annotation

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
Long-Van Nguyen-Dinh

MSc. Computer Science, Purdue University, USA

born on 30.10.1984
citizen of Vietnam

accepted on the recommendation of
Prof. Dr. Gerhard Tröster, examiner
Prof. Dr. Aude Billard, co-examiner

2016

Long-Van Nguyen-Dinh
Wearable Activity Recognition with Crowdsourced Annotation
Diss. ETH NO. 23228

First edition 2016
Published by ETH Zurich, Switzerland

Printed by Druckzentrum ETH Zentrum

To my dear and loving husband Minh,
my little daughter Michelle,
and my parents.

Acknowledgments

My deepest thank goes to my thesis supervisor, Prof. Gerhard Tröster, who has offered his time and wisdom in guiding me through my PhD thesis. Especially, without his support I would have not finished the thesis while being a new mom. I also would like to thank Prof. Aude Billard for co-examining this thesis.

A special thank goes to Dr. Alberto Calatroni, from whom I benefited so much. He contributed valuable feedback to my research and shared his knowledge on many aspects such as electronics, signal processing and music that I was very happy to learn. I also thank him for guiding and joining me to fabulous concerts.

Within the SmartDAYS project, I am particularly grateful to Dr. Daniel Roggen for his support during the initial phase of my PhD. To Dr. Ulf Blanke, I am thankful for the fruitful discussions.

Big thanks go to all my colleagues at the Wearable Computing Lab: Alwin, Amir, Andreas H. and M., Bernd, Bert, Burcu, Catherine, Christian, Christina, Christoph, Daniel W., Franz, Fredy, Giuseppe, Julia, Lars, Luisa, Matija, Martin K. and W., Mathieu, Matthias, Michael, Mirco, Niko, Oliver, Paul, Sebastian, Simon, Sinziana, Tobias, Thomas H. and K., Ruth, and Zack. I enjoyed our time together over lunch, ice cream breaks, at Pilates lessons, and during skiing or hiking events. I also want to thank the students I worked with during semester and master theses: Cedric Waldburger, Yifeng Chen, and Frederik Hess.

Beside my work colleagues, my deep gratitude goes to all of my friends who supported me in many ways, and in particular to Tiana Giang Huong, Henri Hien, and Nhung.

Finally, I am thankful to my parents Ba-Long, Ma-Lien, to my sisters Lien-Bao, Lien-Chi, Lien-Phuong, my brother Long-Hai and to my whole family in Vietnam for supporting me all along. Thanks to Michelle, my little daughter for all happy moments she brings to my life. Lastly, thanks to my husband Minh for his love and without him, I would not have made it through the PhD life.

Zurich, December 2015 Long-Van Nguyen-Dinh

Contents

Abstract

The recent ubiquity of wearable sensor technology enables opportunities to monitor daily activities of people. Applications include ambient assisted living, heath monitoring. Activity recognition from wearable sensors is often accomplished by applying supervised learning on sensor signals. An annotated training data set is required to perform supervised learning. So far, most of the existing works on activity recognition require a small number of experts for data annotation regardless of high labeling effort (i.e., time-consuming, tedious). Consequently, labeling by experts provides high quality annotations, but is non-scalable for a large data set.

Crowdsourcing gains popularity recently to distribute data annotation tasks to a crowd of ordinary people. In this work, we investigate the use of crowdsourcing in activity recognition systems to reduce the effort of collecting large-scale training data, but not to sacrifice a high performance from experts' annotations. This work comprises six scientific publications that leverage the two aspects of crowdsourcing: (1) the use of crowdsourcing to annotate a long continuous recording of activities in which start and end boundaries and labels of activities are explicitly specified (2) the use of online crowd-generated sharing databases (e.g., a sound repository Freesound) where contributors sporadically record each individual activity of interest and upload to the shared repository. Those databases can be retrieved to extract the demanding training set for activity recognition.

In the first aspect, we first conduct a case study to collect annotations for the existing activity data sets by using the crowdsourcing service Amazon Mechanical Turk. The crowdsourced annotations can get as high as 80% sample-based accuracy if multiple crowdsourced labelers are applied. Otherwise, the annotations contain a large presence of noises (52% of instances are labeled incorrectly). Crowdsourced annotations suffer from labeling noises such as mislabeling, or inaccurate identification of start and end time of activity instances. We introduce a taxonomy of annotation noises which possibly occur in a crowdsourcing setting and analyze annotation noises in the crowdsourced annotated data set collected from the case study.

The results show that the noisy annotation can degrade the state-of-the-art activity recognition methods significantly. We propose a novel gesture recognition method - WarpingLCSS as a linear-time template matching method that is robust to annotation noises. The method quan-

tizes signals into strings of characters and then applies variations of the longest common subsequence algorithm (LCSS) to spot gestures. The WarpingLCSS is evaluated extensively on both real and synthetic noisy crowdsourcing scenarios on the three existing data sets with various activity classes (10-17 classes) recorded from accelerometers on arms. The WarpingLCSS achieves better performance than the DTW-based methods and SVM, especially with the large presence of noise. With 60% mislabeled instances, WarpingLCSS outperforms SVM by about 22% F1-score and outperforms DTW-based methods by 36% F1-score. Moreover, WarpingLCSS can tolerate 30%-40% jitter level (i.e., a shift in the annotation temporal boundaries). Additionally, we demonstrate the efficiency of WarpingLCSS in both clean expert-annotated data sets as well as in multimodality settings in which a large combination of different multimodal sensors at different on-body positions is deployed. Given the robustness of WarpingLCSS against annotation noises, we demonstrate that WarpingLCSS can be used as a filtering component to discard noisy-annotated samples and select well-annotated ones for other classifiers like SVM to improve their performance.

We further propose a new annotation technique in which labelers do not have to select the start and end time carefully, but mark a one-time point within the time an activity is happening. This one-time point annotation technique is a special case of annotation noise (the boundary shrinks to a point) and it reduces significantly the labeling burden. However, one-time point annotations cannot be used directly for activity modeling. We propose a preprocessing step to correct temporal boundaries for activities given their one-time point annotations. Specifically, we propose the novel BoundarySearch algorithm to search for temporal boundaries of an activity based on data patterns around their one-time point annotations. We evaluate the method on the three existing data sets with 10-17 classes and the performance on the corrected annotations is just lower than the training on well-annotated annotations by 3% F1-score.

In the second aspect of crowdsourcing, crowd-generated shared repositories capture the diversity in user contexts due to contribution from different people. However, crowd based models fail to capture specific data patterns of targeted users. As a result, it is far to reach user-dependent recognition performance. We focus on adaptation techniques that combine crowd-generated data and user-specific data to achieve high performance similar to a user-dependent recognition system, but still minimize the labeling effort. We investigate

two adapting approaches: 1) a semi-supervised learning to combine crowd-sourced data and unlabeled user data, and 2) an active-learning to query the user for labeling samples where the crowd-sourced based model fails to recognize. We extract audio data from the online crowd-generated audio repository Freesound to train a base model for user daily activities. We test our proposed approaches on 7 users using auditory modality on mobile phones with a total data of 14 days and up to 9 daily context classes. Experimental results indicate that the semi-supervised model can indeed improve the recognition accuracy up to 21% but is still significantly outperformed by a supervised model on user data. In the active learning scheme, the crowd-sourced model can reach the performance of the supervised model with only a few label queries.

Our proposed algorithms enable the opportunities to use crowd-sourcing to reduce the labeling effort for activity recognition systems, but still achieve as good performance as experts' annotation. This work is an important step towards a large-scale activity recognition system in which an effort to collect large number of activities on a large number of users can be distributed to crowdsourcing. Therefore, this work provides the fundamentals for the next generation of wearable assistance scenarios that support activity monitoring for everyone everywhere.

Zusammenfassung

Die zunehmende Verbreitung tragbarer Sensoren ermöglicht das Erkennen und Beobachten von Alltagsaktivitäten. Zu den Anwendungen zählen das assistierte Wohnen sowie die Analyse von Gesundheitsdaten. Aktivitätserkennung wird dabei meist durch die Anpassung von Lernalgorithmen auf bestimmte Sensordaten erreicht. In der Regel benötigt man dafür annotierte Traningsdaten. In den meisten bisherigen Arbeiten werden diese Trainingsdaten durch eine geringe Anzahl von Experten zur Verfügung gestellt, was einen hohen Aufwand nach sich zieht (Zeit, Anstrengung). Dieser Ansatz führt zwar zu hochqualitativen Annotationen, ist aber nicht skalierbar für grosse Datensätze.

Sogenanntes Crowdsourcing, wobei eine grössere Menge an Freiwilligen zur Erhebung von Daten mitwirkt, hat in letzter Zeit an Popularität gewonnen bezüglich Datenannotation. In dieser Arbeit untersuchen wir die Anwendung von Crowdsourcing in der Aktivitätserkennung. Dadurch soll sich der Annotationsaufwand für grosse Datensätze begrenzen lassen, aber eine vergleichbare Qualität wie bei Expertenannotation erreicht werden. Die vorliegende Doktorarbeit umfasst sechs wissenschaftliche Publikationen, die zwei Aspekte dieser Zielsetzung genauer untersuchen: 1) die Anwendung von Crowdsourcing bei der Annotation von langen, kontinuierlichen Datenaufnahmen und 2) die Gewinnung von Trainingsdaten aus Online-Datenbänken (z.B. die Tondatenbank „Freesound“), wo Benutzer freiwillig und sporadisch annotierte Trainingsdaten zur Verfügung stellen.

In Hinblick auf das erste Ziel führen wir zuerst eine Fallstudie durch: Wir sammeln Annotationen zu einem existierenden Datensatz mit dem Crowdsourcing-Dienst Amazon Mechanical Turk. Die so gewonnen Aktivitätsdaten stimmen zu 80% mit den Annotationen von Experten überein, sofern die Beiträge mehrerer einzelner Personen kombiniert werden. Die einzelnen Annotationen der Freiwilligen sind durch eine höhere Anzahl inkorrekter Annotationen verfälscht (52% fehlerhafte Eingaben). Wie diese Studie zeigt, enstehen durch Crowdsourcing Fehlannotationen. Ebenfalls möglich sind falsche Identifikation der Start- und Endzeitpunkte von Aktivitätsinstanzen. Wir führen eine Taxonomie für solche Annotationsfehler ein und analysieren diese weiter im vorgenannten Datensatz.

Die Resultate dieser ersten Untersuchung zeigen, dass Fehlannotationen die Aktivitätserkennung für Standard-Algorithmen stark beein-

flussen. Als Alternative schlagen wir mit WarpingLCSS einen neuen, zeitlinearen Gestenerkennungsalgorithmus vor, der robuster ist bei solchen Fehlern. Dieser Algorithms quantifiziert die Eingangsdaten in Zeichenketten und wendet darauf eine Variation des sogenannten Longest Common SubSequence (LCSS) Algorithmus‘ an. Wir evaluieren WarpingLCSS in realen und synthetisch mit Annotationsfehlern behafteten Datensätzen. Dabei fokussieren wir uns auf drei Beispiele zur Erkennungung von Armaktivitäten (10-17 Klassen) aus Beschleunigungssensordaten am Handgelenk. WarpingLCSS erreichte bessere Resultate als andere Standardansätze wie Dynamic Time Warping (DTW) und Support Vector Machines (SVMs), insbesondere bei vielen fehlerhaften Trainingsdaten. Im Fall von 60% falscher Annotationen erreicht WarpingLCSS einen um 22% besseren F1-Wert als SVM und 36% besser als DTW. Ausserdem kann WarpingLCSS Abweichungen von 30%-40% in der Definition von Start- und Endzeitpunkt einer Geste tolerieren. Wir demonstrieren ausserdem die Effizienz von WarpingLCSS in Experten-annotierten Datensätzen sowie in multimodaler Erkennung (Kombination von Sensoren an verschiedenen Körperpositionen). Die hohe Robustheit der Methode gegen fehlerhafte Trainingsdaten ermöglicht es ausserdem, WarpingLCSS als Filtereinheit einzusetzen, wodurch falsche Daten vor dem Training anderer Klassifizierer wie zum Beispiel SVM ausgeschlossen werden.

Des weiteren schlagen wir eine neue Annotationstechnik vor, bei der die Annotierer nur einen Zeitpunkt zur Aktivität zuordnen. Diese einfache Zeitannotation entspricht einem Spezialfall von falscher Zeitmarkierung (Start und Ende identisch) und sie ist wesentlich weniger aufwändig, kann aber auch nicht direkt zur Aktivitätsmodellierung verwendet werden. Wir entwickeln deshalb mit dem BoundarySearch Algorithmus einen Vorverarbeitungsschritt, der die korrekten zeitlichen Grenzen einer Aktivität aus diesen einfachen Zeitpunktannotationen ableitet. Wir evaluieren die Methode in drei Datensätzen mit 10-17 Klassen und erreichen einen um nur 3% schlechteren F1-Wert als in Experten-annotierten Datensätzen mit Start- und Endzeitpunktangaben.

Bezüglich des zweiten Ziels dieser Arbeit, der Verwendung öffentlicher Datenbänke als Trainingsdaten, zeigen unsere Analysen, dass Crowdsourcingdaten die Diversität an Benutzersituationen gut beschreiben können. Gleichzeitig scheitern die Modelle aber dabei, spezifische, benutzertypische Verhaltensmuster zu erkennen. Als Resultat sind die Erkennungsgenauigkeiten bei Crowdsourcing-

daten viel kleiner als bei nutzerspezifische Daten. Wir fokussieren uns deshalb auf Adaptionstechniken, die Crowdsourcing-Daten automatisch mit Benutzerdaten kombinieren. Dadurch sollen ähnliche Erkennungsraten erreicht werden, während sich der Annotationsaufwand für den Nutzer in Grenzen hält. Wir untersuchen zwei adaptive Ansätze: 1) halb-überwachtes Lernen zur Kombination von Crowdsourcing-Daten mit nicht annotierten Benutzerdaten, und 2) aktives Lernen, wobei der Benutzer um Annotationen gebeten wird, wenn das Crowdsourcing-Modell unsicher ist.

Für eine Beispielstudie extrahieren wir dazu Audiodaten von der Onlinedatenbank Freesound und trainieren eine Basismodell für Alltagsaktivitäten. Wir testen diesen Ansatz mit 7 Benutzern und den Mikrophonen ihrer Smartphones, wobei uns insgesamt Daten von 9 Kontextklassen und 14 Aufnahmetagen zur Verfügung stehen. Die Untersuchung zeigt, dass der halb-überwachte Ansatz die Genauigkeit um 21% erhöht, aber immer noch deutlich schlechter funktioniert als wenn nur annotierte Benutzerdaten für das Training des Klassifizierers verwendet werden. Der aktive Ansatz hingegen erreicht schon nach wenigen Rückfragen ähnlich gute Genauigkeiten.

Die in dieser Arbeit vorgeschlagenen Algorithmen verringern den Aufwand zur Annotation von Trainingsdaten in der Aktivitätserkennung. Dies geschieht durch Zuhilfenahme von Crowdsouring, adaptiven Lernalgorithmen und robusten Erkennungsmethoden, weshalb die Genauigkeit gegenüber Expertenannotationen nicht sinkt. Die entwickelten Methoden stellen einen wichtigen Schritt in Richtung skalierbarer Aktivitätserkennung dar, bei welcher der Aufwand des Sammelns von Trainingsdaten durch viele Nutzer geteilt ist. Somit bildet diese Arbeit das Fundament für eine neue Generation von tragbaren Assistenten, welche Aktivitätsverfolgung überall und für jeden ermöglichen.

1

Introduction

Chapter 1 introduces activity recognition using body-worn sensors. Limitations to collect a large-scale training data set for activity recognition are identified and potentials of using crowdsourcing for activity annotation are highlighted. The main objectives addressed in this thesis are then presented together with a list of publications that resulted from this work.

1.1 Activity Recognition from Wearable Sensors

With the increasing ubiquity of wearable sensor-equipped devices like smart phones, smart watches, glasses, it enables opportunities to monitor user's activities in his daily life and then to provide feedback or assistance if necessary. Instead of using external infrastructures (e.g., pre-deployed fixed-location cameras, RFID tags on objects), a wearable system can be deployed easily on user's body wherever a user goes. Among wearable sensors, we find accelerometers, inertial measurement units IMUs (i.e., a combination of accelerometers, gyroscopes), microphones, global positioning systems (GPS), and physiological sensors (e.g., electrocardiogram). A wide range of applications can benefit from activity awareness. Consider for example a health monitoring system that detects changes in user's activity patterns to reveal possibly the progress of diseases [1]. Another example is a system that tracks activities of industry workers and delivers real-time information about tasks to be performed [2].

Activity recognition from wearable sensors is often accomplished by applying supervised learning techniques on sensor signals. Commonly applied supervised learning approaches for activity recognition include Hidden Markov Models (HMM) [3–6], template matching methods (TMM) using mostly dynamic time warping [2,7,8] and support vector machines [9–11]. Supervised learning requires a labeled training data set to model gestures. Annotations comprise the start and end times (i.e., temporal boundaries) of activities of interest and their corresponding labels. The labeling tasks are usually performed carefully by experts to get annotations as precise as possible. This process is extremely time-consuming and tedious: it may require 7-10 hours to annotate activities in a 30-min video [12]. Additionally, it is costly to hire experts for annotation tasks. Consequently, labeling by experts provides high accuracy, but is non-scalable for a large data set.

1.2 Activity Annotation Techniques

Many annotation techniques have been proposed to collect annotated data for activity recognition systems. There are *offline annotation* techniques which rely on video and audio recordings [12], subject self-report of activities at the end of the day [13]. *Online annotation* (i.e., real-time) techniques perform the annotation during execution of the activities, like experience sampling [14] 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 [15]. There is a trade-off between accuracy of an annotation technique and the amount of time required for annotation [16]. For example, offline annotation on video recordings by experts can provide accurate annotations, however it is extremely time consuming [12]. In contrast, the self-report of the subject may require less time but the accuracy depends on the subject's ability to recall activities. Consequently, it may require extra human effort for manually correcting and cleaning the annotation. Therefore, most of the existing works require video annotation by experts to obtain correct annotated data sets [12].

1.3 Crowdsourcing

Crowdsourcing was defined by Jeff Howe [17] as a distributed model in which a large group of people is engaged to solve a large-scale problem through an open call. A wide range of crowdsourcing applications has been developed to make more efficient use of labor and resources and reduce production costs. One example is the creative drawings Sheep Market [1] – a web-based artwork that calls thousands of workers in the creation of a massive database of sheep drawings. A worker creates his drawing of a sheep and receives an incentive of two cents for his work. Another crowdsourcing application is crowd-generated sharing systems in which users contribute various types of information among the crowd. For example, Wikipedia is an online encyclopedia written by any Internet users.

Crowdsourcing gains popularity recently to distribute data annotation tasks which are traditionally performed by experts to a crowd of ordinary people [18]. Crowdsourcing services, like Amazon Mechanical Turk (AMT) [2], crowdflower [3], clickworkers[4] provide a cheap labor pool for performing annotation tasks. Crowdsourcing has been used for labeling data sets in many fields (e.g., natural language processing [19–21], speech recognition [22, 23], multimedia tagging [24–27]). Data acquired from crowdsourcing is usually generated by low-commitment workers [28], thus it is commonly unreliable and

[1] www.thesheepmarket.com
[2] www.mturk.com
[3] http://crowdflower.com
[4] http://clickworkers.com

noisy. Many strategies has been proposed to reject low-performing and malicious workers to attain good annotations. A use of verifiable questions for which the requester knows the correct answers is a common empirical strategy to screen workers from crowdsourcing. Another way is to use multiple labelers for the same annotation task and majority voting is a popular decision making method used to identify the correct answers [20, 28].

Most of the existing works on activity recognition still require expert annotation to obtain correct annotated data sets regardless of high labeling effort. Crowdsourcing emerges as a promising way to reduce the effort of collecting large-scale annotated training data for activity recognition. The goal of this thesis is to investigate the use of crowdsourcing in activity recognition systems. Crowdsourcing can be utilized in two aspects. In one aspect, a long continuous recording of activities is annotated by asking crowdsourced labelers specify explicitly temporal boundaries and labels of activities. Another aspect is the use of online crowd-generated sharing databases (e.g., a crowd generated sound repository Freesound), where contributors sporadically record an activity of interest and upload to the shared repository. Those databases can be available in large quantities and can be retrieved to extract the demanding training set, thus they minimize the cost of collecting the training dataset. In the following subsections, we discuss these two aspects of crowdsourcing for activity recognition and the limitations of the previous works lead to the formulation of the main objectives of this thesis.

1.3.1 Activity Annotation by Crowdsourcing

Labeling activities in a recording requires to specify the temporal boundaries of activities to mark when the activities occur and their corresponding labels. The task can be performed offline in crowdsourcing platforms like Amazon Mechanical Turk by asking crowdsourced workers to label a video footage synchronized with sensor data. A more obtrusive crowdsourcing task would ask users to annotate their own activities over a long time span of recording (e.g., weeks) to capture various contexts in their daily routines. This type of crowdsourced data collection would be useful to gather data for long-term health care monitoring systems and the annotation task can be done in real-time while sensor data are recorded.

Activity annotation by crowdsourcing in which crowdsourced

users provide the temporal boundaries and labels of activities have not been investigated yet. In the work by Zhao et al. [25], individual still images were extracted from a video recording and activities occurring in the image were labeled by crowdsourcing. However, using one video frame to ask for activity may cause ambiguity (e.g., it is hard to distinguish Open Door and Close Door activities with just only one frame of activity). Moreover, it does not work when a video footage of the recording is not available.

Crowdsourcing opens a door for easily obtaining labeling for large-scale training data sets for activity recognition. However, due to the error-prone nature of crowdsourcing [28], the challenge is to obtain both correct labels and correct temporal boundaries for activities. As a video footage is available for labeling, the quality of annotation would be controlled by using multiple labelers and then checking the agreement in the annotations among labelers. However, there is no guarantee to have a perfect annotation, especially when using multiple labelers cannot be applied. Hence, in order to use crowdsourcing for activity labeling, the nature of labeling noises needs to be investigated and the impact of those noisy annotations on the training of activity recognition needs to be analyzed. Moreover, learning methods that can cope with annotation errors are also needed. These topics in activity annotation by crowdsourcing are addressed in details in the first part of this thesis .

1.3.1.1 Robustness against Annotation Noise

The effect of annotation noises on the performance of classifiers has been investigated in the literature [16, 29–32]. However, these studies conducted experiments on synthetic noisy data. Additionally, annotation noises included only the case of having wrong labels (i.e., labels are substituted as other classes). In activity annotation, temporal boundaries may be marked incorrectly (e.g., an activity starts earlier and ends later than the correct boundaries). These other kinds of noise in crowdsourced activity annotation have not been investigated yet.

1.3.2 Activity Crowd-Generated Databases

Many available web repositories contain training data for activity recognition systems. For instance, open user-contributed sound

databases (e.g. the Freesound [5] database) can bootstrap sound-based activity recognition; Youtube [6] provides annotated videos from which activity models based on movement sensors can be derived; geographical locations are mapped to possible activities (e.g. with Google Places [7], Foursquare [8]). The ideas of mining the web crowd-generated databases for relevant training data have been used recently by researchers to reduce the effort to collect and label training data. Perkowitz et. al. [33] presented the first method for web-based activity discovery using text. Rossi et. al [15] used the crowdsourced Freesound database to train sound models which are exploited to recognize activities of daily living on mobile phones.

Consider characteristics of a crowd-generated database, for example, in Freesound sounds are contributed by a very active online community with the rapid increase in the number of sounds available. Each sound clip often belongs to one context and is annotated in free-form style. Moreover, user-contributed sounds are recorded in a wide variety of situations, conditions, motivations, and skills. Roma et. al. [24] addressed the limitations of searching for environmental sounds in the Freesound such as outlier, unbalanced tag distribution, tag synonyms. Rossi et. al. [23] proposed filtering techniques to eliminate web search audio results from Freesound which include sound samples with unexpected acoustic content (i.e. outlier).

The growing crowdsourced repository captures the diversity in user contexts due to contribution from different people. However, crowd based models fail to capture specific data patterns of targeted users. Consequently, it is far to reach user-dependent recognition performance. To enhance the use of crowd-generated databases in activity recognition, the second part of our thesis focuses on adaptation techniques that combine crowd-generated data and user-specific data to achieve high performance similar to a user-dependent recognition system, but still lower the labeling effort to a minimum.

1.3.2.1 Adaptation Techniques in Machine Learning

Semi-supervised learning and active learning are two widely used adaptation techniques in machine learning that minimize the need of

[5] www.freesound.org
[6] www.youtube.com
[7] https://developers.google.com/places/
[8] https://foursquare.com/

labeled training data. Semi-supervised learning make use of both labeled and unlabeled data to train a recognition system. Meanwhile, active learning selectively asks labels of the most informative training instances that can generalize the classifier maximally, and thus reduces user's burden of labeling, but still gets good performance. There are many variations of semi-supervised learning and active learning algorithms. A comprehensive survey can be found in [34] for semi-supervised learning and in [35] for active learning, respectively.

According to the best of our knowledge, there is no previous work that investigated the adaptation techniques to optimally leverage two sources of data: labeled crowd-sourced data and user-centric data. The existing works explored semi-supervised learning and active learning on labeled and unlabeled data that were both from the same data source [36,37]. In contrast to these works, we aim to adapt and improve the recognition of a classifier learned from one free data source – crowd-sourced repository – on another, the user personalized data to achieve high performance but minimize the labeling effort.

1.4 Objectives of the Thesis

This thesis investigates crowdsourcing to reduce the effort of collecting annotated training data for activity recognition, but not to sacrifice a high performance from experts' annotations. Specifically, based on two aspects of crowdsourcing described above, we have identified the following research contributions.

1.4.1 Activity Annotation by Crowdsourcing

This work will investigate the use of crowdsourcing in activity annotation in which labelers provide the temporal boundaries and labels of activities occurring in a long recording. We mainly focus on labeling short actions on arm (i.e., gestures). The following steps constitute the main contributions of this work.

1. A case study to collect annotations for the existing activity data sets by using the crowdsourcing service Amazon Mechanical Turk. Strategies to filter annotation noises as well as malicious workers are investigated. The outcomes illustrate the quality of crowdsourced annotation in both offline and online labeling.

2. Introduction of a taxonomy of annotation noises which possibly occur in a crowdsourcing setting. We give the statistical analysis on annotation noise in the real crowdsourced annotated data set collected from the case study.

3. WarpingLCSS is proposed as a fast, robust template matching method for online gesture recognition. The WarpingLCSS is evaluated extensively on both real and synthetic noisy crowdsourcing scenarios. Additionally, we demonstrate the efficiency of WarpingLCSS in both clean expert-annotated data sets as well as in multimodality settings in which a large combination of different multimodal sensors at different on-body positions is deployed.

4. Given the robustness of WarpingLCSS against annotation noises, we demonstrate that WarpingLCSS can be used as a filtering component to discard noisy-annotated samples and select well-annotated ones for other classifiers to improve their performance significantly.

1.4.2 One-Time Point Annotations

This work is a follow-up of our previous work in activity annotation by crowdsourcing presented in Section 1.4.1. An activity annotation commonly includes the start and end times and the corresponding label in the sensor recording. This work investigates a new annotation technique in which labelers do not have to select the start and end time carefully, but just mark a one-time point within the time an activity is happening. One-time point annotation is likely to happen in real-time labeling, for example, when labelers remember to annotate the start of a gesture but forget to annotate the end. It is a special case of crowdsourced annotation noises in which the boundary of a gesture shrinks to a point. The one-time point annotation technique would reduce the burden in activity annotation considerably. However, the one-time point annotated data set cannot be used directly to model gesture classes. We need an algorithm in the preprocessing step to correct the annotations. Specifically, we need an algorithm that can find the correct start and end time of each gesture around its one-time point annotation. In this work, we make the following contributions.

1. We propose BoundarySearch algorithm to search for the start and end time of a gesture around its given one-time point annotation.

2. The corrected temporal annotations are compared to the ground truth and their performance when used as a training data set for modeling activities is evaluated.

1.4.3 Personalized Adaptation on Crowdsourced-based Models

This part of the thesis focuses on techniques to combine the advantages of crowd-generated data contributed by anonymous (e.g., context diversity, free annotated data) and user-centric data (e.g., individual-specific contexts) to not only obtain a high performance of user-dependent recognition rate, but also minimize the labeling effort. We achieve this goal by adapting a generic model based on crowdsourced data to a personalized model with no to little labeling effort for user data. We consider in particular the Freesound database as a crowd-generated sound repository to extract training data to recognize user daily activities on mobile phones. Two adaptation techniques are applied: semi-supervised learning and active learning.

1. We investigate a semi-supervised learning scheme to combine labeled crowd-generated audio data with unlabeled user-centric data

2. We investigate an active learning scheme to detect the most informative user-specific data samples that the crowd models can not represent well and queries a user to label them.

3. We evaluate the tradeoff between labeling effort and accuracy of those personalized models.

1.5 Thesis Outline and Paper List

This thesis includes six publications to address the objectives as described in Section 1.4. Figure 1.1 gives the link between objectives and the chapters in the thesis by which they are covered.

Table 1.1 lists the publications presented in this thesis by chapter. Chapter 2 provides a summary of this thesis' contributions and lists the main findings, discusses their relevance and limitations, and provides an outlook to future work.

Chapter 3 presents a real crowdsourcing experiment to get activity annotation for the existing recordings of activity data sets.

Chapter 4 introduces the WarpingLCSS algorithm for activity recognition and the evaluation on clean annotated data sets.

Chapter 5 presents a noise taxonomy of crowdsourced annotation and evaluates the robustness of the WarpingLCSS algorithm on both real and synthesis noisy crowdsourced annotation.

Chapter 6 investigates the ability of WarpingLCSS on multimodality.

Chapter 7 presents the one-time point annotation technique and the BoundarySearch algorithm to correct the temporal boundaries of activities.

Chapter 8 investigates adaptation techniques that integrate user data into a crowd-based model.

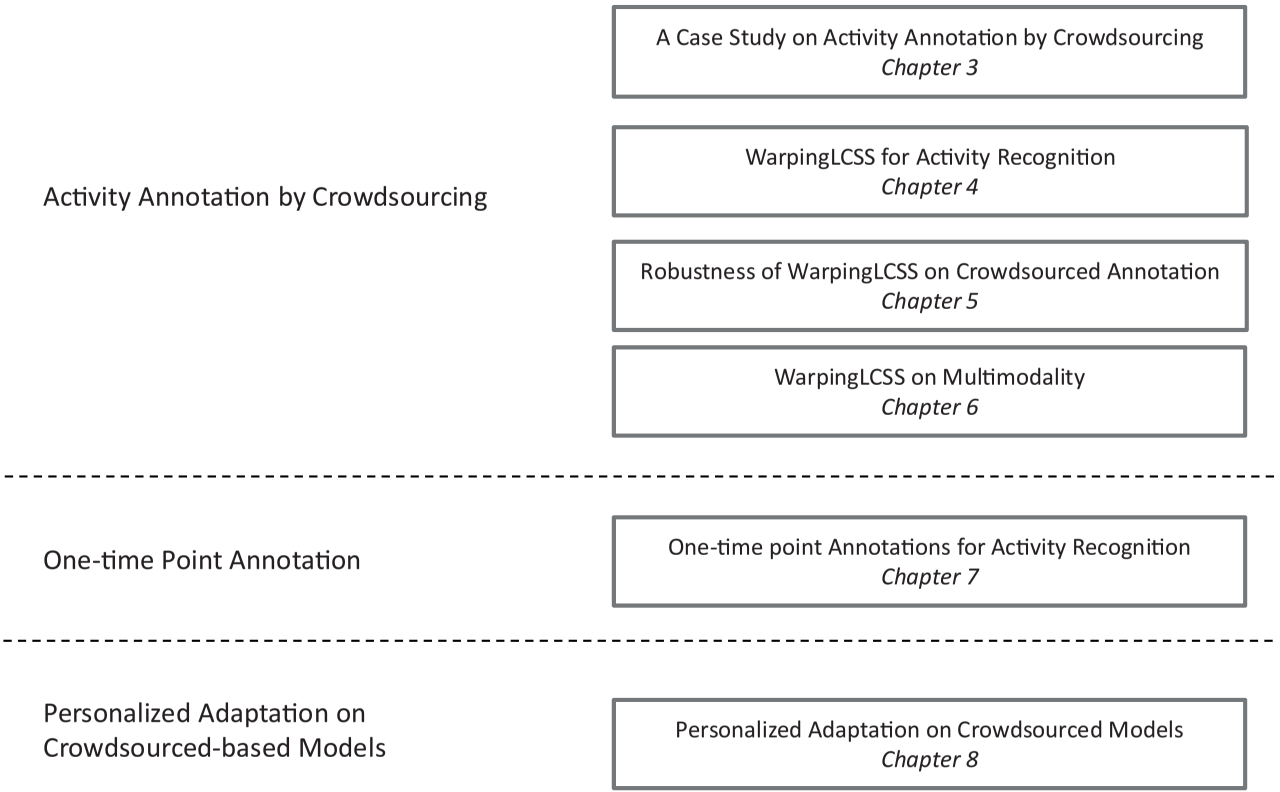

Figure 1.1: Outline of the chapters that are included in this thesis according to the aims presented in Section 1.4 (see Table 1.1 for the corresponding publications).)

Table 1.1: Publications and their corresponding sections in this thesis.

Chapter	Publication
3	**Tagging Human Activities in Video by Crowdsourcing** *L.-V. Nguyen-Dinh, C. Waldburger, D. Roggen, and G. Tröster* Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR), Dallas, USA, 2013, ACM.
4	**Improving Online Gesture Recognition with Template Matching Methods in Accelerometer Data** *L.-V. Nguyen-Dinh, D. Roggen, A. Calatroni, and G. Tröster* Proceedings of the 12th International Conference on Intelligent Systems Design and Applications (ISDA), Kochi, India, 2012, IEEE.
5	**Robust Online Gesture Recognition with Crowdsourced Annotations** *L.-V. Nguyen-Dinh, A. Calatroni, and G. Tröster* Journal of Machine Learning Research (JMLR), Volume 15, pp. 3187–3220, 2014, JMLR.
6	**Towards A Unified System for Multimodal Activity Spotting: Challenges and A Proposal** *L.-V. Nguyen-Dinh, A. Calatroni, and G. Tröster* Proceedings of the International Conference on Ubiquitous Computing (UbiComp Adjunct), Seattle, USA, 2014, ACM.
7	**Supporting One-Time Point Annotations for Gesture Recognition with Wearable Sensors** *L.-V. Nguyen-Dinh, A. Calatroni, and G. Tröster* Transactions on Pattern Analysis and Machine Intelligence (PAMI), pending *(Under review at the time of writing)*, 2015, IEEE.
8	**Towards Scalable Activity Recognition: Adapting Zero-Effort Crowdsourced Acoustic Models** *L.-V. Nguyen-Dinh, U. Blanke, and G. Tröster* Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia (MUM), Luleå, Sweden, 2013, ACM.

1.6 Additional Publications

The following publications have been authored and co-authored in addition to those presented in this thesis:

- **Combining Crowd-generated Media and Personal Data: Semi-supervised Learning for Context Recognition.** *L.-V. Nguyen-Dinh, M. Rossi, U. Blanke, and G. Tröster.* In Proceedings of the 1st ACM International Workshop on Personal Data Meets Distributed Multimedia (PDM), Barcelona, Spain, 2013, ACM.

- **Enhancing Action Recognition through Simultaneous Semantic Mapping from Body-Worn Motion Sensors.** *M. Hardegger, L.-V. Nguyen-Dinh, A. Calatroni, D. Roggen and G. Tröster.* In Proceedings of the International Symposium on Wearable Computers (ISWC), Seattle, USA, 2014, ACM.

- **Limited-Memory Warping LCSS for Real-Time Low-Power Pattern Recognition in Wireless Nodes.** *D. Roggen, L. P. Cuspinera, G. Pombo, F. Ali and L.-V. Nguyen-Dinh.* In Proceedings of the 12th European Conference on Wireless Sensor Networks (EWSN), Porto, Portugal, 2015, Springer [Best Paper Award].

2

Thesis Summary

Chapter 2 summarizes the main approaches and contributions of this thesis, discusses the limitations and presents an outlook with opportunities for future research. Detailed descriptions and discussions of the contributions can be found in the referenced publication chapters.

2.1 Activity Annotation by Crowdsourcing

2.1.1 A Case Study to Acquire Activity Annotation from Amazon Mechanical Turk

We conducted a real experiment at Amazon Mechanical Turk (AMT) to get activity annotations for two existing data sets. The Opportunity data set [12] contains 17 daily gestures (e.g., drink, open or close doors) and the CMU kitchen data set [38] contains 32 gestures for making a brownie (e.g., stir egg, pour water into bowl). The video duration of one recording is about 25 minutes long in the Opportunity data set and about 6 minutes long in the CMU kitchen dataset. For each data set, we selected a video footage of one subject and segmented the long video footage into short videos of about one minute. We showed each short video to workers in 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 video footage. Furthermore, each annotation task was performed by 10 different AMT workers. For more details about the interface design that we used to show the video footages and the task description to AMT labelers, refer to Section 3.3.1 in Chapter 3.

We proposed two strategies to detect and filter non-serious labelers and erroneous labeling. They were individual filtering and collaborative filtering. In individual filtering, we used verifiable questions [22, 23, 39, 40] whose answers are known in advance to screen each individual AMT labeler. Specifically, we asked for the starting time of two activities in the given video sequence and one boolean question about whether an activity occurred in the video. These questions ensured that AMT labelers must watch the given video to answer them correctly. Individual filtering checks the correctness in the answers of each labeler on these verifiable questions and rejects labelers whose answers are wrong. Besides using verifiable questions, in the individual filtering we also used characteristics of the data sets to filter non-serious labelers (e.g., activities are non-overlapping).

Collaborative filtering checks the labeling agreement among labelers to detect spammers. Specifically, a labeler X who has a disagreement score $d(X) = \frac{\text{Annotation times of X disagree with majority}}{\text{Total annotation times of X}} > \textit{threshold}$ is a spammer. Score d is computed as follows.

- Extract the starting and ending times of all tagged activities from all labelers and put into a sorted list.

- Scan through each temporal segment S (i.e., two consequence elements) in the sorted list, and the label of S is the majority voting among all activities containing this segment.
- For each labeler having an annotation for S which disagrees with the majority, the score d is accumulated by the length of S. At the end, d is compared with the threshold and spammers are detected and removed.

We chose a threshold = 0.3. It means the labeler is a spammer and his annotations are removed if less than 70% of his annotations agrees with the majority. The value of threshold was chosen similar to the guidelines for Cohen's kappa coefficient which measures inter-rater agreement [41]. The collaborative filtering process to compute the disagreement score d is illustrated in Figure 2.1. After filtering, the majority voting among qualified annotations is performed to generate a final crowdsourced annotation. A more detail on the crowdsourcing experiment is given in Chapter 3.

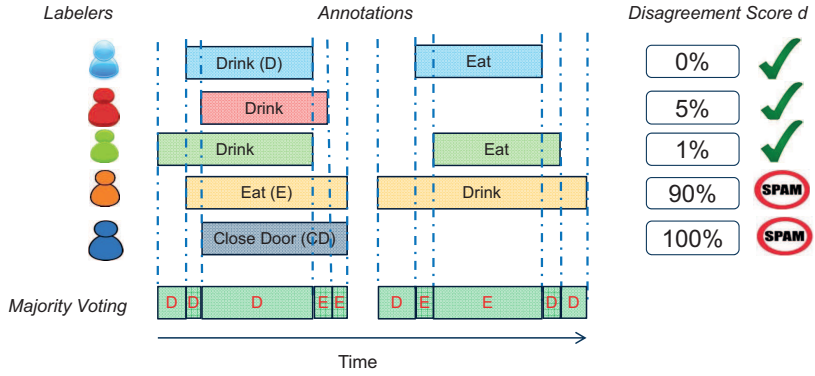

Figure 2.1: An illustration of the collaborative filtering technique to calculate the disagreement score of each labeler against the majority. The last two labelers are spammers and then their annotations will be removed.

In our experiments, we selected labelers in AMT who must have at least 90% approval rate on their work history. However, the results showed that the quality of annotations still needed to be controlled carefully. Specifically, in case of using only one AMT labeler for an

annotation task, the sample-based accuracy (i.e., fraction of correctly labeled samples compared to the ground truth) can get as low as 41%. The result from one labeler illustrates a quality of annotation in real-time labeling where control is difficult. With multiple labelers, our proposed filtering strategies increased the sample-based accuracy of annotations by up to 40%. After filtering, the annotations from AMT achieved high accuracy (76% - 92%) for each short video.

2.1.2 Taxonomy of Annotation Noises

The quality of annotations from crowdsourcing is commonly unreliable. We categorized annotation noises in two types.

- *Boundary jitter:* The annotation boundaries are strayed, but the label is correct. This can happen due to the carelessness of crowdsourced labelers or a vague definition of gesture boundaries.
- *Label noise:* Gestures are associated to wrong labels or to no label at all.

We sub-categorized *boundary jitter* into four error types, namely *extend*, *shrink*, *shift left*, *shift right*, according to how the gesture boundaries are shifted compared to the ground truth. We also sub-categorized *label noise* into three error types, namely *delete* (i.e., a gesture is not annotated), *substitute* (i.e., mislabeled as another gesture class), and *insert* (i.e., mislabeled when no gesture of interest occurs). Figure 2.2 illustrates subclasses of *boundary jitter* and *label noise*.

For *boundary jitter*, we define a *jitter level* to quantify the proportion of data points in a gesture that are wrongly annotated. Let N denote the time length of a gesture. We define Δs be the absolute time difference between the crowdsourced start time and the correct start time. Similarly, we define Δe as the absolute time difference between the crowdsourced end time and the correct end time. Δs and Δe are illustrated in Figure 2.2a for the different boundary jitter noises. For *extend* and *shrink* jitters, the *jitter level* is calculated as $\frac{\Delta s+\Delta e}{N}$. For *shift-left* and *shift-right* jitters, the *jitter level* is $\frac{\Delta s+\Delta e}{2*N}$.

Annotation noise statistics from the real crowdsourcing We summarize annotation noises in the real crowdsourcing case study in the Opportunity data set [12] that we presented in the previous section. The expert's annotations are used as a ground truth to evaluate the

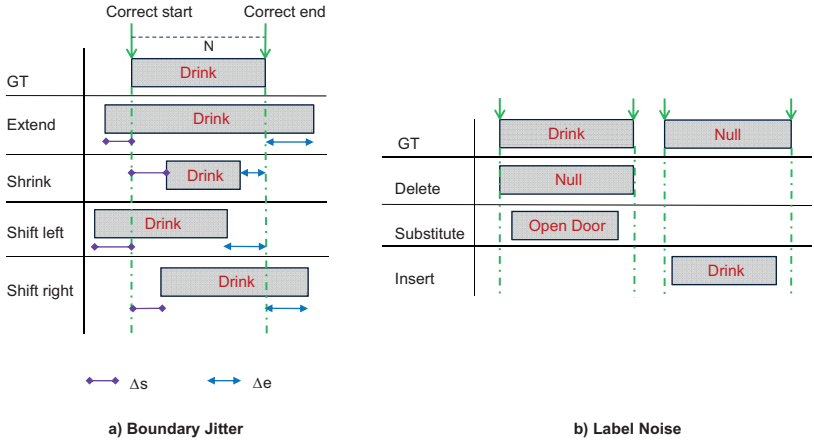

Figure 2.2: Illustrations of boundary jitter and label noise in crowdsourcing annotation. GT stands for ground truth. The blue dash-dotted lines indicate the correct boundary of a gesture. (Figure 5.1, page 107)

crowdsourced annotations. Two crowdsourcing scenarios were analyzed: a one-labeler annotation where only one crowdsourced labeler was selected randomly among the ten workers, and a multiple-labeler annotation where the filtering and majority voting were applied for the ten workers as presented in the previous section 2.1.1. Here we merge the results from all short segmented videos to have a complete annotation for one subject in the Opportunity data set.

The sample-based accuracy is as low as 55% for the one-labeler annotation and it reaches 80% for the multiple-labeler annotation. Figure 2.3 gives a detailed analysis of all noise types. *Label noises* occur often in the one-labeler, take 52% of the instances which are mostly *substitute* and *delete* errors. In the multiple-labeler, *label noises* decrease to 18%. For *boundary jitter*, jitter levels range from 27% to 60% on average. However, there exist good annotated instances with very low jitter levels (only 2%).

This example shows that using multiple labelers for an annotation task can reduce labeling errors. However, it cannot be always applied (e.g., in real-time annotation). The amount and distribution of annotation noises will change depending on the crowdsourcing scenario and on the kind of activity data, but there is no reason to think that some

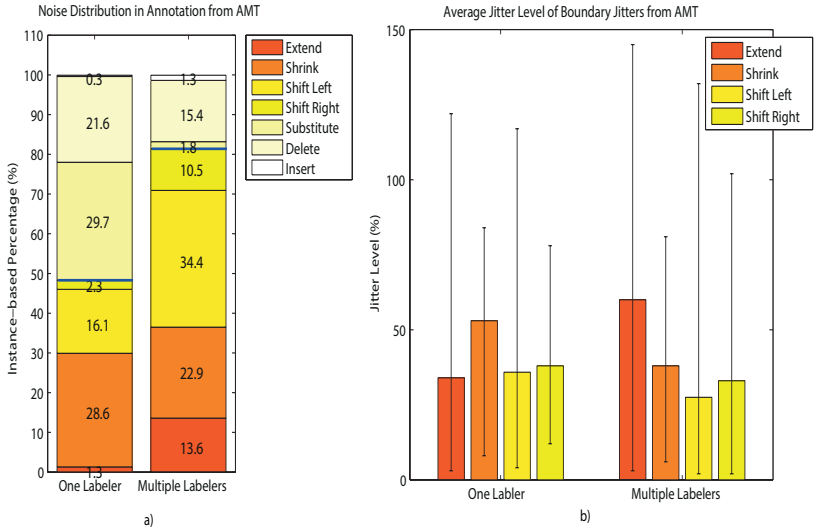

Figure 2.3: Analysis of crowdsourcing annotation from AMT. Blue lines in the figure a separate boundary jitter part and label noise part. Black lines in the figure b show the minimum and maximum level of jitter in each type of noise. (Figure 5.3, page 111)

scenarios will achieve much lower noise levels. It calls the attention of robust activity recognition methods to deal with noisy crowdsourced annotations.

2.1.3 WarpingLCSS - A Template Matching Method

We proposed template matching approaches based on longest common subsequence (LCSS) [42] for online pattern recognition to cope with annotation errors in training data. We focused on short actions performed by hand movements (i.e. gestures) like "open door", "drawing a circle", "drink a cup". Chapters 4,5,6 discuss and evaluate the proposed methods in detail, whereas this section presents a summary and highlight results.

Figure 2.4 depicts the data flow in the proposed template matching approaches. Sensor signals are quantized and converted into a sequence of symbols (i.e., a string). This is done by first performing k-

means clustering on all training data samples recorded from the body-worn sensors in the training phase. The distance between two symbols is the Euclidean distance between their corresponding cluster centers, normalized to an interval [0,1]. Each data sample is then converted to a symbol that indicate the cluster to which a sample belongs. The number of cluster k is chosen empirically or through cross-validation so that the distribution of cluster centroids captures the variation of data samples (i.e., hand movements).

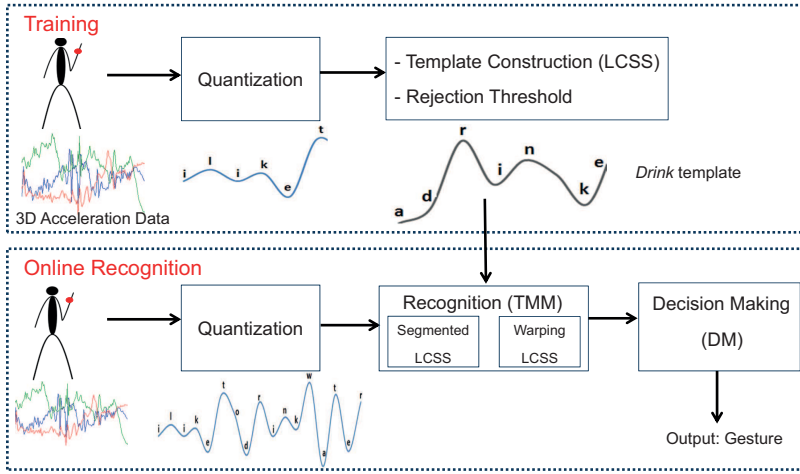

Figure 2.4: Data processing flow of the proposed LCSS-based template matching methods for gesture recognition. (Figure 5.4, page 113)

2.1.3.1 Training phase

From the training data, one template is created for each gesture class to represent the motion pattern for that class. The similarity score between two gesture instances is measured as the length of longest common subsequence (LCSS) between them. The template is chosen as the instance that has the highest average similarity to all other instances of the same class in the training data.

A rejection threshold is a value to decide whether signals belong to a gesture class or not. Instances belonging to a class should have the similarity score with the template of that class greater than the

corresponding rejection threshold. 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 instances belonging to the same class. We calculate the rejection threshold ϵ_c for gesture class c as a value below $\mu^{(c)}$ by some standard deviations.

$$\epsilon_c = \mu^{(c)} - h * \sigma(c), \tag{2.1}$$

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

It implies that any instances in the training data set with the similarity that is lower than h standard deviation from the mean are outliers. This rejection threshold selection is robust with the presence of wrongly annotated instances in a class. The value of h can be selected by running the spotting on training data with a range value of h of between 0 and 5 and selecting the threshold which has the best F1-score (see in Section 5.5.4 for the computation of F1-score metric). In most of our experiments, $h = 1$ provided the best performance.

Note that for noise-free annotated training data, another strategy to select rejection threshold is taking the minimum LCSS between the chosen template and other gesture instances in the same class. The rationale is that in the well-annotated training data all instances in one gesture class are valid (see more in Chapter 4 where we used this strategy to select rejection thresholds).

2.1.3.2 Recognition phase

In the recognition phase, the streaming data from sensor readings is first quantized to symbols and then compared with the templates to recognize gestures.

To detect gestures in the continuous stream, we first proposed SegmentedLCSS approach which was based on a sliding window technique. The sensor signal is first segmented into a sliding window and then the LCSS algorithm computes the similarity score between a string in the window and templates of gesture classes. When a new symbol arrives, the window shifts and the LCSS is repeatedly computed. The SegmentedLCSS algorithm is straightforward, however a drawback is its running time which is quadratic in template length to process a new symbol in the stream. For online recognition, the computational cost is an important factor. Hence, we proposed a linear-time algorithm WarpingLCSS that can detect the occurrences of a template within a stream without pre-segmenting data.

Given a template T and a continuous stream S, the WarpingLCSS algorithm computes the similarity score $W_{(T,S)}(i, j)$ between the first i symbols of T and the first j symbols of S. The algorithm starts with an empty stream. As a j-th symbol is appended to the stream, it is compared with the i-th symbol in the template, and the similarity scores $W_{(T,S)}(i, j)$ at row i and column j is updated instantaneously. If two symbols match, a reward of +1 is added to the similarity score W. Otherwise, W is decreased by a penalty. The penalization takes into account three possibilities: 1) a mismatched alignment between two current symbols in the two strings; 2) the warping between the current symbol with its previous one in the template T; 3) the warping between the current element with its previous one in the stream S. We consider warping two consecutive symbols in the string in case of mismatch so that the penalty is counted only once if the string contains contiguous repetitions of a symbol which occur for example in a slower execution of a gesture. The penalty is computed as $-p * d(\alpha_i, \alpha_j)$, where $d(\cdot, \cdot)$ is the distance between two symbols and p is a scale parameter of the dissimilarity. The similarity scores between the whole template and the stream are stored in the last row of W.

Specifically, $W_{(T,S)}(i, j)$ is computed as follows.

$$W_{(T,S)}(i,j) = \begin{cases} 0 & \text{, if } i = 0 \text{ or } j = 0 \\ W_{(T,S)}(i-1, j-1) + 1 & \text{, if } T(i) = S(j) \\ \max \begin{cases} W_{(T,S)}(i-1, j-1) - p * d(T(i), S(j)) \\ W_{(T,S)}(i-1, j) - p * d(T(i), T(i-1)) \\ W_{(T,S)}(i, j-1) - p * d(S(j), S(j-1)) \end{cases} & \text{, otherwise,} \end{cases} \tag{2.2}$$

Figure 2.5 illustrates how the WarpingLCSS algorithm works. When a gesture of a class is performed, the similarity score W accumulates rewards and grows until reaching a local maximum and eventually decreases again, as soon as the gesture is over. When gestures differ from those encoded by the stored templates, W drops quickly due to the penalty terms. In the last row of similarity scores, when a local maximum exceeds the rejection threshold ϵ_c, it indicates a gesture of class c has been spotted in the sensor signal. The algorithm needs to store only the last column to compute the similarity score of the new symbol. Thus, it requires only a constant amount of memory to update

the similarity score W as well as do backtracking to get the starting boundary of a spotted activity.

The spotting of different templates from all concerned activity classes can be processed concurrently. If a gesture is assigned to multiple classes (i.e., boundaries of spotted instances are overlapping), the decision making module (DM) resolves the conflicts. The class having a highest normalized similarity score (i.e., the similarity score is normalized by template length and spotted gesture length) is chosen (refer to Chapter 4.4.3 for more details about how to compute normalized similarity score between two strings).

2.1.3.3 Evaluation

Data Sets We used three existing data sets including various gestures for evaluation. The list of gestures of these data sets are shown in Table 2.1. The Skoda data set [43] contains 10 manipulative gestures performed in a car maintenance scenario by one subject. Each gesture class has about 70 instances and the *null* class takes 23% of data samples.The HCI data set [44] contains 10 gestures of geometric shapes executed by a single person with the arm in the vertical plane. The HCI data set contains a large number of data samples belonging to *null* class (57%) and each gesture class has about 50 instances. The Opportunity data set [12] contains 17 daily gestures recorded in a naturalistic environment akin to an apartment. In the Opportunity, each gesture class has 20 instances excepts "Drink Cup" and "Toggle Switch" each having 40 instances. 37% of data samples belong to *null* class.

The Skoda and HCI data sets are characterized by a low variability in the execution of gestures, meanwhile the Opportunity data set presents a challenge to detect activities due to their large variability in execution styles. Note that in Opportunity data set, there are three drawers at different heights and two different doors which make the recognition more challenging.

Baseline Methods We compared the WarpingLCSS and the SegmentedLCSS with three other baseline approaches: two existing TMMs based on Dynamic Time Warping (DTW) [2, 7, 8] and support vector machine (SVM). The DTW-based TMMs are the Segmented DTW [7,8] and the Nonsegmented DTW [2] that use DTW distance as a metric to measure the similarity between two gestures. For all TMM methods, we used the same strategy to select templates, i.e., the maximum sim-

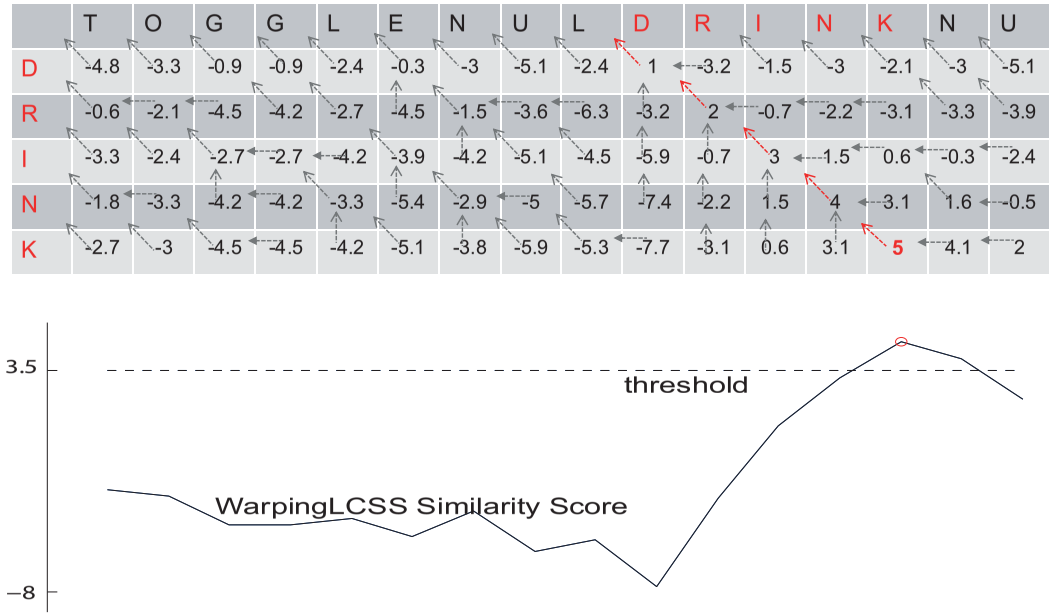

Figure 2.5: WarpingLCSS is computed between a template "DRINK" and a streaming "TOGGLENULDRINKNU" and is shown in the table. We assume the distance between two symbols is their difference in encoded English alphabets (A-Z are converted to 0-25); the penalty p is 0.3; and the rejection threshold for drink gesture is 3.5. Arrows indicate which of the preceding positions are picked to generate similarity scores. The plot of scores in the last row is shown below the table. A red circle at a local maximum greater than the threshold indicates that a drink gesture is detected. The boundary of the spotted gesture can be identified by tracing back the matching path (red arrows).

Table 2.1: Gestures in Opportunity, Skoda, and HCI data sets.

HCI Gestures				
Circle	Triangle	Square	Infinity	Slider
Their Speculars				Null

Opportunity Gestures		
Null	clean Table (CT)	open Drawer 1-2-3 (ODr1-2-3)
close Drawer 1-2-3 (CDr1-2-3)	open Door 1-2 (OD1-2)	close Door 1-2 (CD1-2)
open Fridge (OF)	close Fridge (CF)	drink Cup (D)
open Dishwasher (ODi)	close Dishwasher (CDi)	Toggle Switch (TS)

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

ilarity average for LCSS-based methods and the minimum distance average for DTW-based ones. They all had the same quantization preprocessing step over sensor data. For SegmentedLCSS and Segmented DTW, the window length was chosen as the template length.

For SVM, the signals were segmented into a sliding window, with 50% overlap. For each window, two features - mean and variance of the signals were extracted and then fed into a SVM classifier. We used RBF kernels and the two RBF parameters were selected by using cross-validation.

Among these methods, SegmentedDTW has the worst computational complexity which is cubic in template length to process a new symbol when spotting. The complexity of SegmentedLCSS is quadratic. Meanwhile, WarpingLCSS, Nonsegmented DTW and SVM require only the linear-time complexity.

Experiments We evaluated the LCSS-based TMMs and the baselines in three aspects. Their performances were first evaluated on clean-annotated data sets in which annotations were given carefully by experts. Secondly, their performance on noisy crowdsourced data sets were analyzed. Finally, we evaluated the proposed WarpingLCSS in multimodality settings in which different multimodal sensors are deployed. In the first two evaluations, the 3D accelerometer data on the dominant lower-arm were used to evaluate. In the multimodality experiments, different combinations of motion sensors from different on-body positions were tested.

Activity recognition systems often target user-independent recognition in which a training data set is collected from a few number of representative subjects and then activity models are built with a goal to generalize for all users. As a result, user-independent activity recognition rarely achieves the performance of user-dependent recognition due to the fact that different people may have different styles of activity executions. Due to a large labor pool of crowdsourced labelers, crowdsourcing is potential to give annotations for data recording that is targeted to one specific user. Therefore, with crowdsourcing, the recognition system can be built individually for each user with user-dependent activities of interest. Consequently, we focus on subject-dependent evaluation.

For each subject, we performed 5-fold cross validation. In the WarpingLCSS algorithm, the penalty parameter p was 0.3 for the Opportunity, 0.5 for the HCI, and 0.8 for the Skoda. We evaluated the performance with the sample-based accuracy and the F1-score metrics. The F1 was computed in two ways, either with (F1_Null) or without the Null class (F1_NoNull). The recognition system that has high values of both F1_Null and F1_NoNull predicts well both gesture classes and Null class.

2.1.3.4 Results on clean-annotated data sets

Table 2.2 shows the performances of WarpingLCSS and the four baseline methods on the Opportunity, HCI and Skoda data sets with annotations from experts. In the Opportunity data set, we here report the result from one subject (see Chapter 4.6.2.2 for more results from other subjects). Note that the result discussed in Chapter 4.6.2.2 was for a different strategy to calculate the rejection threshold (the minimum LCSS between the chosen template and other gesture instances in the same class). Here, the rejection threshold is selected by cross-validation with the Equation 2.1 for the consistent comparison between WarpingLCSS and the other methods in the all three aspects that we discussed above.

Our WarpingLCSS and SegmentedLCSS outperform two DTW-based approaches in the Opportunity dataset (up to 21% higher in accuracy and F1 measure). In the HCI data set, all approaches yield comparable results and gain high accuracy. In the Skoda, the SegmentedLCSS is superior to other methods (about 7% higher in accuracy and F1 measure), but they all have good performance (> 80% F1-score). Therefore, in the clean annotated data sets, the results show that our

Table 2.2: The accuracy and F1 measure over sample unit in the three data sets

Opportunity Dataset			
Method	Accuracy	F1_Null	F1_NoNull
WarpingLCSS	0.57	0.57	0.54
SegmentedLCSS	0.59	0.58	0.59
Segmented DTW	0.45	0.49	0.50
Nonsegmented DTW	0.36	0.36	0.31
SVM	0.56	0.55	0.48

HCI Dataset			
Method	Accuracy	F1_Null	F1_NoNull
WarpingLCSS	0.78	0.77	0.69
SegmentedLCSS	0.83	0.83	0.78
Segmented DTW	0.84	0.84	0.79
Nonsegmented DTW	0.79	0.79	0.73
SVM	0.81	0.80	0.63

Skoda Dataset			
Method	Accuracy	F1_Null	F1_NoNull
WarpingLCSS	0.82	0.87	0.83
SegmentedLCSS	0.90	0.90	0.93
Segmented DTW	0.83	0.84	0.82
Nonsegmented DTW	0.79	0.80	0.85
SVM	0.87	0.87	0.87

proposed LCSS-based approaches are competitive to the three state-of-the-art methods in activity recognition, especially for data sets that suffer from high variation in activity execution as in the Opportunity data set. SegmentedLCSS achieves the best performance but WarpingLCSS is more suitable for online recognition due to its small complexity and high accuracy.

2.1.3.5 Results on noisy crowdsourced data sets

We evaluated WarpingLCSS, SegmentedLCSS and the baseline methods on both real and synthetic crowdsourced annotations. To generate the synthetic crowdsourced annotations, we modified clean annota-

tions from the three data sets described above by simulating *label noise* and *boundary jitter* as discussed in the taxonomy in Section 2.1.2. We ran simulations for each type of noise separately in order to understand their effects on performance. The results showed that F1-Null and F1-NoNull scores had a similar trend of performance as the noise levels increase, hence we report F1-Null score only.

Results on Label Noise Simulations Let α be the label noise percentage in each class. In each gesture class, α percent of instances are randomly selected and their labels are randomly flipped to other classes (including *null* class).

Figure 2.6 shows the results of *label noise* simulations on the three data sets over a parameter sweep for α. The performances of LCSS-based methods are stable until a label noise percentage (α) in each class exceeds 70% in Opportunity and HCI data sets and 50% in the Skoda data set. SVM performs worse than our LCSS-based methods when α increases. 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.

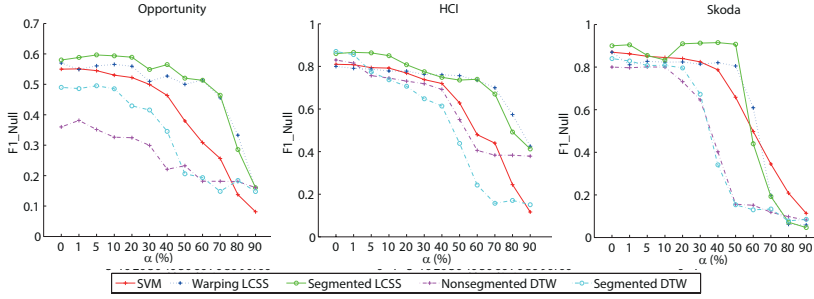

Figure 2.6: Performance of label noise simulation for the three data sets. (Figure 5.4, page 113)

Our LCSS-based methods are robust to *label noise* due to their capability to pick a good template among noisy instances for a gesture class as long as the ratio of good instances in a gesture class is still predominant. Meanwhile, DTW-based methods fail to pick a good template because mislabeled instances bias and dominate the DTW

metrics. SVM performs worse when α increases because both good and bad instances contributes equally to the model building.

Results on Boundary Jitter Simulation In the *boundary jitter* simulations, we assume that all gesture instances have the same jitter level β. Specifically, in the *extend* or *shrink* simulations, all gesture instances are extended or shrunk at both ends equally by ($\beta/2$ per side) respectively. 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.

Figure 2.7 shows the results of extend jitter simulations on the three data sets. In *extend* jitter, data belonging to the *null* or other classes (before and after the gesture) are labeled noisily as belonging to the gesture class. All methods 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. The performance of SVM depends on how much the noisy feature vectors added into the model of each gesture class. In the *extend* case, samples of the null class are partially labeled as a gesture class. Hence, different levels of variability of the signals belonging to the *null class* in the data sets decide the performance of SVM.

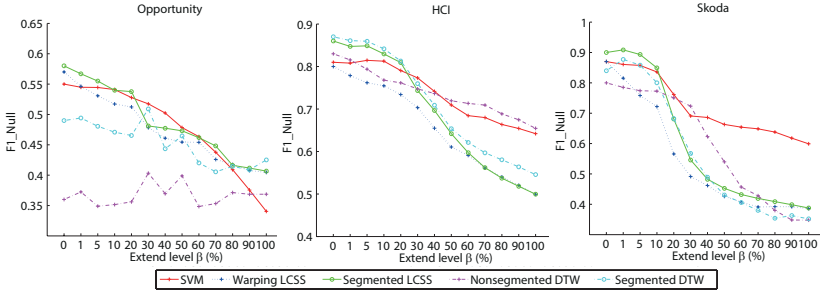

Figure 2.7: Performance of extend jitter simulation. (Figure 5.9, page 127)

The performance of *shrink* simulations are given in Figure 2.8. Our proposed LCSS-based methods (WarpingLCSS and SegmentedLCSS) achieve the best performance in the three data sets. All methods can tolerate about 30% shrink level before a degradation. The performance

of SVM falls faster as a *shrink* level β increases. 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. The SVM model is more likely to be corrupted due to the lack of labeled data when β is large. For TMMs, shrunk instances still form shorter subsequences that can still be matched to the test data. In our simulations, all gesture instances have the same *shrink* level leads to a good alignment between instances in the same class. As a result, the performances of the DTW-based methods degrade slowly as β increases as shown in Figure 2.8. In realistic experiments, when different shrink levels are applied, their performance can fall quickly due to data misalignment.

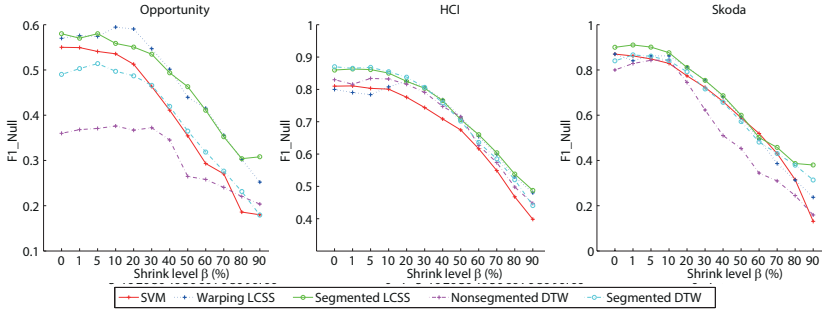

Figure 2.8: Performance of shrink jitter simulation. (Figure 5.10, page 127)

The *shift-right* and *shift-left* simulations have similar results and here we show only the results of shift-right jitter simulations in Figure 2.9. All methods can tolerate about 20% *shift level*. When annotations are shifted, a mixture of the effects on *extend* and *shrink* is present (i.e., some samples belonging to a gesture are lost and some noisy labeled samples are added to the gesture). As can be seen, these effects reflect on the performances of SVM and TMM methods in the three data sets.

Results on Real Crowdsourced Annotation We investigated further the performances of our proposed WarpingLCSS and SegmentedLCSS methods on the real crowdsourced annotations performed by AMT workers for the Opportunity data set that we summarized in Section 2.1.1. Both the annotations obtained in the one-labeler and multiple-labeler scenarios were used. Mixtures of all kinds of the annotation

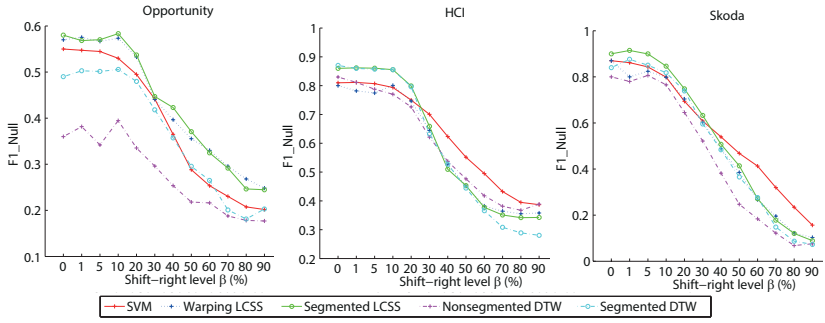

Figure 2.9: Performance of shift-right jitter simulation. (Figure 5.11, page 128)

errors are present and jitter levels are varied from one instance to another instance as shown in Figure 2.3.

The performances on these crowdsourced annotations are shown in Figure 2.10. In the multiple-labeler annotation, 80% of the data samples are annotated correctly. Only 18% of gesture instances are wrongly labeled and the remainder are correctly labeled with a small jitter level of at least 2%. Consequently, the performances of all recognition methods are just slightly decreased by up to 4% for F1-Null and 6% for F1-NoNull compared to the training with clean annotated training sets. However, our LCSS-based TMMs yield the best performance.

In the 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. With the high presence of label noise, the performance of SVM decreases dramatically, down to a F1-NoNull of 5% (i.e., equivalent to a random guess). Our LCSS-based TMMs still achieve the best performance, outperform SVM by about 27% F1-score and outperform DTW-based methods by up to 25% F1-score.

A LCSS-based Filtering Component The results have shown that SVM is very sensitive to the high *label noise* in the training data set (see Figure 2.6 and Figure 2.10). 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 proposed

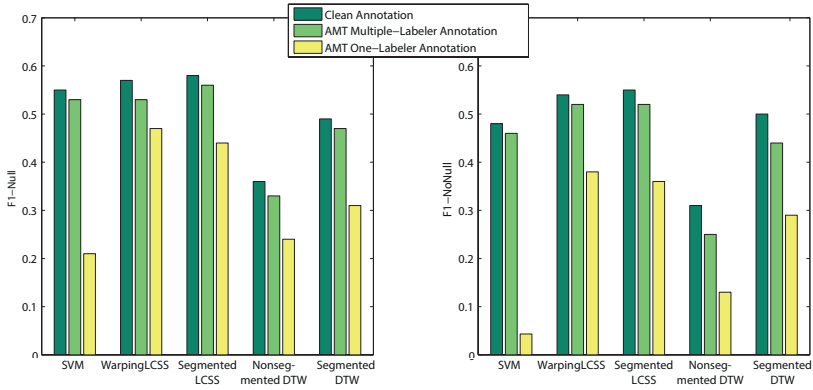

Figure 2.10: Performance of the crowdsourcing annotation from AMT on Opportunity data set. (Figure 5.12, page 130)

a LCSS-based filtering component to filter out noisy labeled data in crowdsourced annotations before being fed into other learning techniques like SVM or the DTW-based TMMs.

Before training a SVM, the LCSS-based filtering component computes a LCSS similarity matrix among all pairs of instances in the class, and keeps only the instances that have a high average similarity to other instances of the same class. To clean noise inside the null instances (e.g., *delete* noise), the WarpingLCSS runs the spotting on the data annotated as null and discards any parts that are classified as any gestures of interest. We call this approach Filtering SVM (FSVM).

For DTW-based TMMs, the filtering component picks a template by using LCSS similarity. Then the template are used in the Segmented and Nonsegmented DTW spotting methods. We call these approaches LCSS-SegDTW and LCSS-NonSegDTW respectively.

The performances of SVM and DTW-based methods with the filtering component are shown in Figure 2.11 for 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.

Similarly, the filtering DTW spotting methods outperform the non-filtering ones. The result clarifies that LCSS is better than DTW in selecting a good template among noisy instances. Importantly, the results show that with the same templates, our LCSS-based TMMs still

outperform LCSS-NonSegDTW and LCSS-SegDTW in spotting.

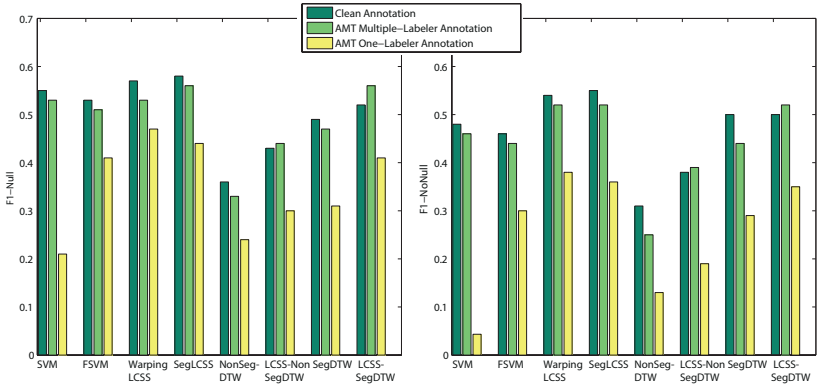

Figure 2.11: Performance of the AMT crowdsourcing annotation on Opportunity for the methods with and without filtering. SegLCSS, NonSegDTW, and SegDTW stand for Segmented LCSS, Nonsegmented DTW and Segmented DTW respectively. (Figure 5.12, page 130)

Figure 2.12 shows the performances of all the methods for the synthetic label noise simulation. The filtering-based methods outperform non-filtering ones and keep the performance stable much longer when α increases. Our proposed LCSS-based TMMs have similar or better performance than the other methods. With the same templates picked by LCSS, the LCSS-SegDWT and LCSS-NonSegDTW have a similar performance as our LCSS-based TMMs in the HCI and Skoda data sets. However, in the Opportunity dataset, our LCSS-based methods are still better. The rationale is that LCSS is more robust than DTW to capture similarity between gestures of the same class that have high-variance in execution (see more in Chapter 4).

2.1.3.6 Results on Multimodality

The results obtained so far show the robustness of our WarpingLCSS and SegmentedLCSS on both clean and noisy annotated data set. However, WarpingLCSS is more appropriate than SegmentedLCSS for online recognition due to its cheap computation cost, therefore we select WarpingLCSS for further exploration. In previous experiments, Warp-

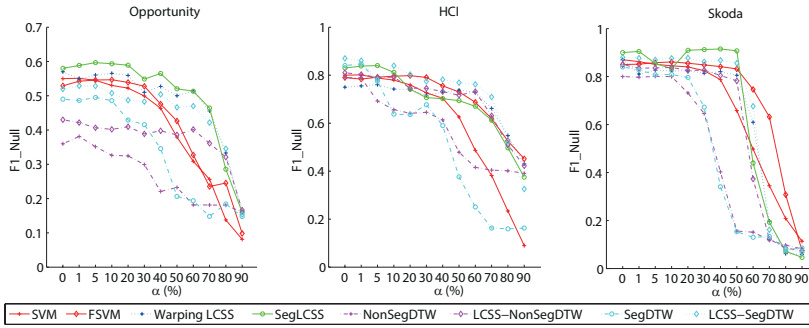

Figure 2.12: Performance of label noise simulation for the methods with and without filtering. (Figure 5.14, page 133)

ingLCSS is tested on single sensor modality (i.e., accelerometer on arm). In this section, we summarize the ability of WarpingLCSS on multimodal activity recognition (see Chapter 6 for more details).

We investigated two multimodal frameworks to fuse different data sources either at the signal level (*signal fusion*) or at a decision level (*classifier fusion*). In the signal fusion, signals from all sensor modalities are combined into one data stream before being processed by the WarpingLCSS. Figure 6.4 shows the signal fusion framework. Let d_i be the dimension of signal data generated from sensor $\mathcal{S}_i \in \Phi$. The combined data stream from the *Signal Fusion* module has a dimension of $\sum_{i=1,\mathcal{S}_i \in \Phi}^{|\Phi|} d_i$. The template matching (TM) module processes data and spots activities with WarpingLCSS. The TM module is already presented in Section 2.1.3 and illustrated in Figure 2.4. Finally, the decision making module handles spotting conflicts and outputs recognized activities.

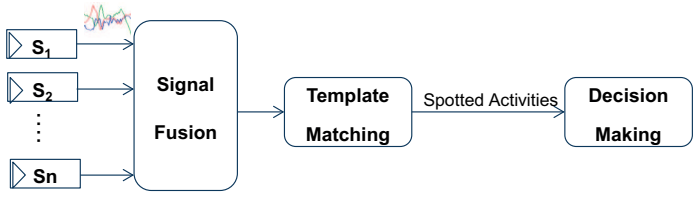

Figure 2.13: Signal fusion framework for sensors $S_1, S_1, \cdots, S_n$

In contrast, in the classifier fusion framework, each sensor is treated uniformly via the same process in the *Template Matching* module. The WarpingLCSS scores from all sensors are combined linearly, weighted by a prior weight to indicate how well a sensor can recognized the specific class. A gesture class with a highest fused score is the best match. Figure 2.14 shows the classifier fusion framework.

Let Φ and $|\Phi|$ be the set of sensors and the number of sensors in the system, respectively. We represent the spotting output from a sensor $\mathcal{S} \in \Phi$ in a spotting matrix $\mathcal{M}(\mathcal{S})$ of size $C * \mathcal{N}$, with C is the number of activity classes of interest and $\mathcal{N}$ is the number of samples processed. $\mathcal{M}(\mathcal{S})(c, i)$ represents the entry at the ith sample and the row of class c in the matrix $\mathcal{M}(\mathcal{S})$. Each row c in the matrix, indicated as $\mathcal{M}(\mathcal{S})(c)$ stores the information of spotted instances of an activity class c from the sensor $\mathcal{S}$. Specifically, if the sensor outputs an activity instance of class c from *start-time* to *end-time* with a similarity score *simScore* (i.e., *[start-time, end-time, c, simScore]*), then $\mathcal{M}(\mathcal{S})(c, i) = simScore$ for all i-th samples in the interval from *start-time* to *end-time* at the row c. Figure 2.15 gives an example of the spotting matrix.

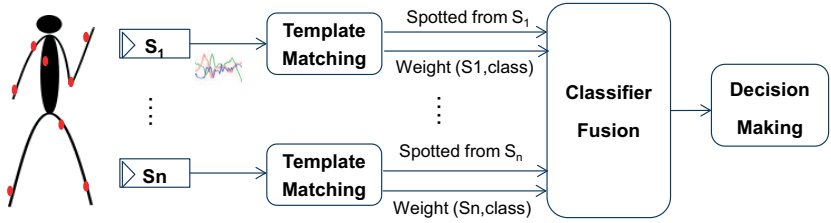

Figure 2.14: Classifier fusion framework for sensors $S_1, S_1, \cdots, S_n$

Let Weight($\mathcal{S}$,c) be a prior weight to indicate how well sensor $\mathcal{S}$ can recognize the specific class c. We set Weight($\mathcal{S}$,c) as the best F1-score performance when selecting the rejection threshold for activity class c in the processing of sensor s (see Section 2.1.3 for more details). The weighted summed spotting matrix is computed as follows.

$$\bar{\mathcal{M}}(c) = \sum_{\substack{i=1 \\ \mathcal{S}_i \in \Phi}}^{|\Phi|} Weight(\mathcal{S}_i, c) * \mathcal{M}(\mathcal{S}_i)(c) \; \forall c. \tag{2.3}$$

The similarity score of an activity in the spotting matrix degrades if

Activity class	samples 1	2	3	4	5	6	7	8	9	10	…
drink	0	0.8	0.8	0.8	0	0	0	0	0	0	…
open_door	0	0	0	0	0	0	0.6	0.6	0.6	0	…
close_door	0	0	0	0	0	0	0	0	0	0	…

Figure 2.15: An example of the spotting matrix with three activity classes (drink, open door and close door) and two spotted activities: [2, 4, drink, 0.8] and [7, 9, open_door, 0.6].

the prior performance of the sensor to recognize the corresponding activity class is low.

We tested the multimodal WarpingLCSS on four subjects in the Opportunity data set (see 2.1.3.3 for detailed description of the Opportunity). Each subject wore 17 sensors of three modalities (3D accelerometer (A), 3D gyroscope (G) and 3D magnetic field (M)) attached at different on-body positions - right upper arm (RUA), right lower arm (RLA), left upper arm (LUA), left lower arm (LLA), back (BACK), right shoe (RSHO) and left shoe (LSHO)). The signals of all sensors were recorded at a frequency of 30Hz.

In the classifier fusion, the number of symbols in quantization was selected empirically k = 20 for each 3D sensor. In the signal fusion framework, the number of symbols was selected much higher k = 200 to capture variants in the combined movements at different on-body positions.

The presence of sensors on shoe may degrade the performance significantly due to the fact that their signals are not distinguishable for different gesture executions. To give a flavor of this hypothesis, we ran the signal fusion with the baseline SVM on 17 sensors. For subject 1, SVM achieves 39% F1-Null and 27% F1-NoNull, degrades the performance of the one-best sensor significantly by 20%. The performances for subject 3 and subject 4 are similar. For subject 3, SVM only obtains 8% F1-NoNull (the result is equivalent to a random guess). The results on SVM demonstrate that adding activity non-distinguishable sensors (e.g., sensors on shoes) into a system may degrade significantly the performance. However, we included them in the multimodality setting to investigate the robustness of WarpingLCSS.

Figure 2.16 shows the performances of WarpingLCSS on each sen-

sor (i.e., number of sensors = 1). As can be seen, the performances of different sensors vary significantly. The sensors on shoes give the worst performance (about 10% F1-NoNull). The results of WarpingLCSS on the use of all 17 sensors in the signal fusion and classifier fusion are shown in Table 2.3. They both achieve good performances for the four subjects (63% to 84% F1 scores). Compared with using only one sensor, using 17 sensors lifts up the average performance by up to 23% F1-Null and 21% F1-NoNull. It also increases the performance of the best one-sensor by up to 11% F1-Null and 15% F1-NoNull. The signal fusion can detect the null class better than the classifier fusion by about 7% F1-Null. The rationale is that the signal fusion of WarpingLCSS has a global view of data from all sensors at once and it spots an activity only when the combined pattern of that activity from different sensors is matched. Conversely, the classifier fusion has only a local view of data from each sensor, and the presence of not-so-distinguishable sensors (e.g., shoe sensors) in the classifier fusion may lead to the spotting of the false detected activities instead of the null class.

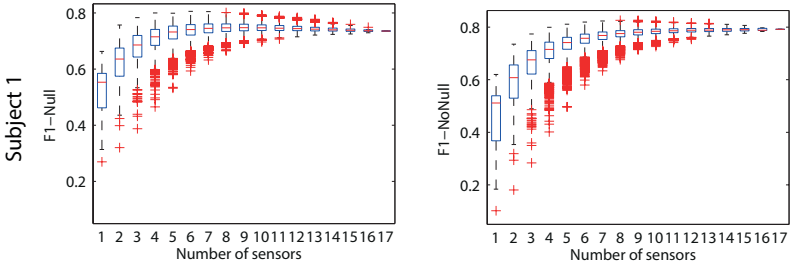

Figure 2.16: Performance of classifier fusion framework with all subset combinations of 17 sensors for subject 1. The red middle lines in boxes show the average performance. (Figure 6.6, page 148)

In Figure 2.16, we also show the performances of different subset combinations in the classifier fusion for subject 1. Other subjects have a similar trend of performance (see Chapter 6 for more details). On average, both F1-Null and F1-NoNull scores grow significantly as the number of sensors increases from 1 to 6 and then keep almost stable. Generally, adding more sensors does not always lead to the better performance (e.g., two accelerometers at lower arm and upper arm

may not improve the detection of open door 1 and open door 2). The observation is similar to the reported findings in [45,46]. We show the best subset of 17 sensors that gets the best result on Table 2.4

Table 2.3: Performance of two frameworks on 17 sensors in the Opportunity dataset. (Table 6.2, page 147)

	Subject 1		Subject 2		Subject 3		Subject 4		Average	
Method	F1-Null	F1-NoNull	F1-Null	F1-NoNull	F1-Null	F1-NoNull	F1-Null	F1-NoNull	F1-Null	F1-NoNull
Classifier Fusion	0.74	0.79	0.63	0.67	0.75	0.80	0.65	0.71	0.69	0.74
Signal Fusion	0.77	0.77	0.67	0.68	0.84	0.81	0.74	0.73	0.76	0.75

The results indicate that our WarpingLCSS performs well in both signal fusion and classifier fusion frameworks. Especially, WarpingLCSS is robust with the presence of not-so-distinguishable sensors in the system (compared with SVM that achieves only 39% F1-Null and 27% F1-NoNull in signal fusion on 17 sensors on subject 1). For SVM and other feature-based machine learning techniques, a careful selection of features for different modalities may be important to improve the performance. Meanwhile, WarpingLCSS treats different modalities, and sensors with the same modalities at different on-body placements in a homogeneous way. Due to its simplicity and effectiveness, WarpingLCSS is useful for multimodal activity recognition systems. For more discussion on trade-off between the sensor fusion and signal fusion, refer to Chapter 6.

Table 2.4: The combination of sensors giving the best performance in the classifier fusion framework. (Table 6.4, page 150)

	Sensors (Number of sensors)	F1-Null	F1-NoNull
Subject 1	BACK_M, RUA_G, RLA_G, RLA_M, LUA_A, LLA_A (6)	0.82	0.83
Subject 2	BACK_G, BACK_M, RUA_G, RUA_M, RLA_G, LUA_A (6)	0.71	0.73
Subject 3	BACK_G, RUA_M, RLA_G, RLA_M (4)	0.87	0.85
Subject 4	BACK_M, RUA_G, RLA_A, RLA_M (4)	0.75	0.74

2.2 One-Time Point Annotations

An activity annotation commonly includes the start and end times and the corresponding label of the activity in the sensor recording. We proposed a new annotation technique in which labelers do not have to select the start and end time carefully, but mark a one-time point within the time an activity is happening. This one-time point annotation technique reduces significantly the labeling burden both in real-time labeling and offline labeling. The work also focused on hand gestures.

One-time point annotation is a special case of *boundary jitter* that we presented previously in Section 2.1.2. It equivalents to a scenario that all gestures are shrunk to minimum, just a one-time point. Our experimental results in Section 2.1.3.5 showed that machine learning methods can tolerate at most 40% jitter level. Consequently, one-time point annotations (almost 100% jitter level) cannot be used directly to model gesture classes. However, we proposed a novel boundary fixing approach to search for the correct start and end time of an activity around its given one-time point annotation.

Figure 2.17 shows the data flow and its illustration through different processing components in the boundary fixing approach. The sensor signals are first quantized and converted into sequences of symbols (strings). The initial start and end boundaries of each gesture are then set loosely around its given one-time point annotation to ensure the correct boundaries fall inside the segment. We consider two ways of boundary initialization, depending on whether the maximum gesture length for each class is known or not, namely MaxLen and NoLen respectively. In the MaxLen case, the initial boundaries are extended around the annotated point by the maximum length. In the NoLen case, we extend the boundaries to the previous and subsequent annotated points. The non-motion removal component shrinks the initialized boundaries to the motion segment containing the annotated point. This non-motion removal step is optional and can be applied only if the signal comes from an accelerometer attached on arm that executes gestures. Finally, a novel boundary searching algorithm (BoundarySearch) seeks for the good boundaries of a gesture around its given one-time point annotation and within the initialized boundaries. Chapter 7 gives a detail about all processing steps in the approach. Here, we summarize the BoundarySearch algorithm and present the highlight results.

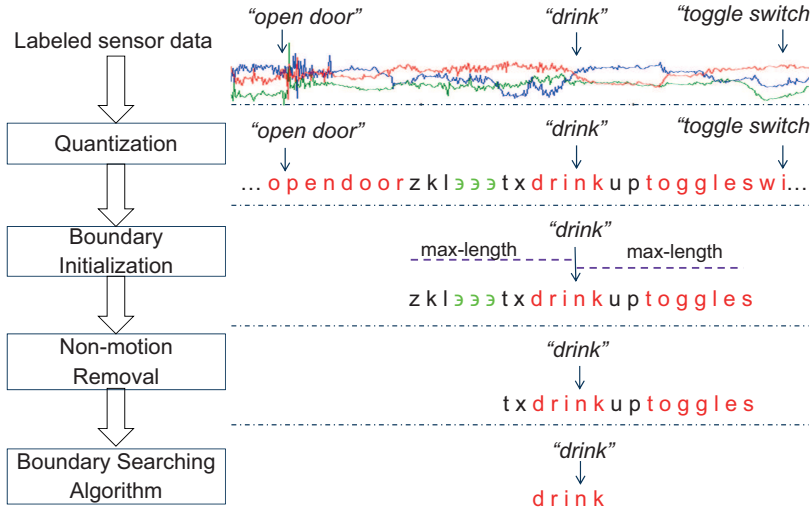

Figure 2.17: Data processing flow of the proposed boundary fixing approach. An example is shown on the right. Three annotated gestures "open door", "drink", and "toggle switch" with their one-time point annotations indicated as black arrows are given in the training data. In this example, the maximum length of "drink" gesture is given and it is used to initialize the boundaries of the "drink" gesture. ə represents non-motion symbols.

2.2.1 BoundarySearch Algorithm

We assume that gestures are performed in a random order in the training data set. The BoundarySearch algorithm searches for the correct boundaries of gestures based on seeking similar data patterns around the annotated points of the same class. Given two gesture instances w_1 and w_2 of the same class whose boundaries are initialized around their one-time point annotations, the BoundarySearch algorithm first detects all pairs of optimal matching substrings within w_1 and w_2. Then a pair of matching substrings that cover the one-time point annotations become new boundaries for w_1 and w_2.

Given two strings w_1 and w_2, we define a warping path to match the two strings by either aligning an element of w_1 to an element of w_2, or warping two consecutive elements of w_1 or of w_2. Figure 2.18

illustrates the warping path to match two strings.

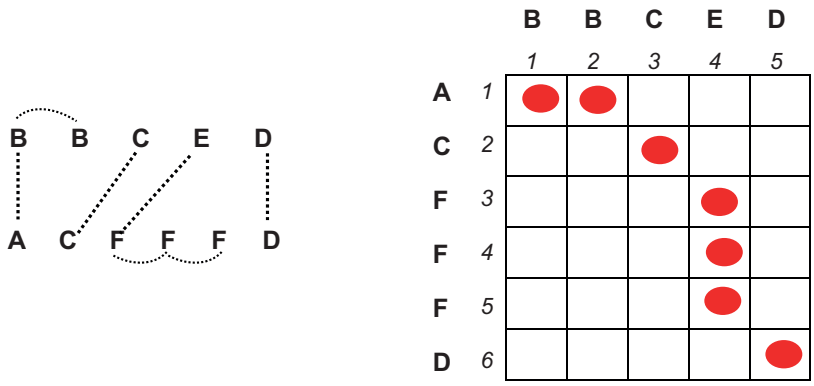

Figure 2.18: Illustration of an alignment between two strings "BBCED" and "ACFFFD" and the corresponding warping path starting from the first indices at the top left to the last indices at the bottom right. In this example, two symbols "B" of the first string are warped and so are three symbols "F" of the second string.

Let $\Psi_A(\cdot,\cdot)$ be a similarity weight of an alignment between a symbol in w_1 and a symbol in w_2 and $\Psi_W(\cdot,\cdot)$ be a similarity weight of a warping between two consecutive elements in w_1 or in w_2. In case of a matched alignment, a similarity weight is a reward $R = 1$. Otherwise, a similarity weight is a negative penalty computed as $-p * d(\alpha_i, \alpha_j)$, where p is a penalty parameter of the dissimilarity and $d(\cdot,\cdot)$ is the distance between two symbols. Specifically, Ψ_A and Ψ_W are defined as follows.

$$\Psi_A(\alpha_i, \alpha_j) = \begin{cases} 1 & \text{, if } \alpha_i = \alpha_j \\ -p * d(\alpha_i, \alpha_j) & \text{, otherwise} \end{cases} \tag{2.4}$$

(2.5)

$$\Psi_W(\alpha_i, \alpha_j) = -p * d(\alpha_i, \alpha_j) \tag{2.6}$$

A warping path has a similarity score which accumulates all similarity weights along the path.

To detect all pairs of optimal matching substrings within two gestures of the same class, we define $B(i, j)$ to be the maximum similarity score attained over all possible warping paths between substrings of

w_1 ending at the i-th position and substrings of w_2 ending at the j-th position. The BoundarySearch algorithm obtains the score $B(i, j)$ as follows.

$$B(i,j) = \begin{cases} 0 & \text{, if } i = 0 \text{ or } j = 0 \\ \max \begin{cases} 0 \\ B(i-1, j-1) + \Psi_A(w_1(i), w_2(j)) \\ B(i-1, j) + \Psi_W(w_1(i), w_1(i-1)) \\ B(i, j-1) + \Psi_W(w_2(j), w_2(j-1)) \end{cases} & \text{, otherwise,} \end{cases} \tag{2.7}$$

The algorithm starts with an empty string and the corresponding B score is 0. A value of B is updated by taking the maximum of the best possible matching (alignment or warping at the current positions) and 0. A negative value of B would not be of interest because we can always choose different starting indices for substrings to get a better value B. $B(i, j) = 0$ also indicates there is no substring matching at position i and j.

All optimal matching substrings between two strings can then be found by tracing back the matching path starting from an element of B greater than 0 and ending with an element of B equal to zero. Figure 2.19 illustrates an example of the similarity score B computed between two "drink" gestures whose boundaries are initialized loosely and their optimal matching substrings. To make the example more interesting, we consider the initialized boundaries wrongly covering some parts of "toggle switch" gestures. As can be seen, the BoundarySearch algorithm can find both a matching between two "drink" gestures and a matching between two "toggle switch" gestures.

Each matching forms a cluster of connected scores (i.e., from the starting vertex of a cluster to the furthest indices the cluster can reach). The BoundarySearch algorithm then selects only the cluster that covers the one-time point annotations of the two gestures w_1 and w_2. In the targeted cluster, we select the highest similarity score $\mathbf{M}$ indexed at (v_1, v_2) such that the path from the starting vertex of the cluster indexed at (u_1, u_2) to (v_1, v_2) still covers the marked annotated points. The new boundaries are set from u_1 to v_1 for the first gesture and from u_2 to v_2 for the second one.

Given an instance k of gesture class c, we run the boundary fixing for this instance paired with all other instances in the same class. Then

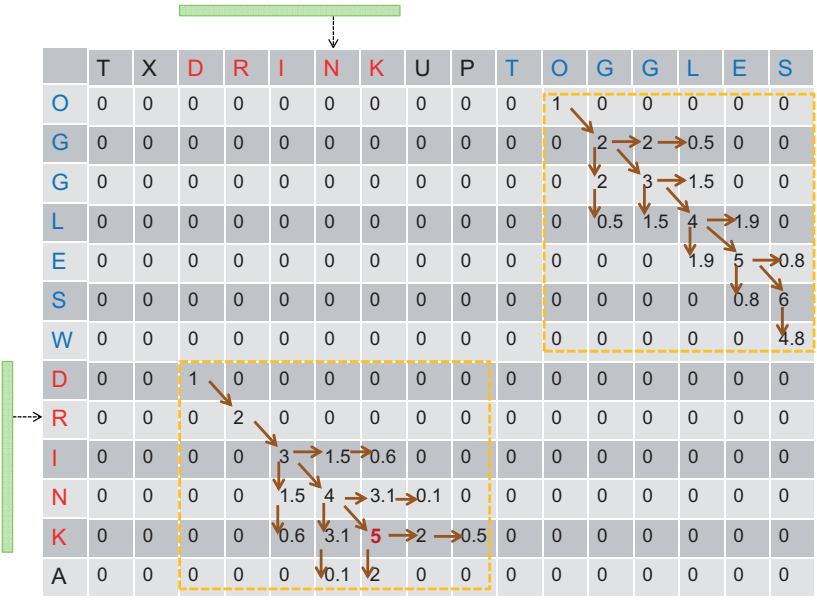

Figure 2.19: Illustration of BoundarySearch algorithm to compute the score B between two "drink" gestures with incorrectly initialized boundaries w_1 = "TXDRINKUPTOGGLES" and w_2 = "OGGLESWDRINKA". In this example, the initialized boundaries wrongly cover some parts of "toggle switch" gesture ("TOGGLES" in w_1 and "OGGLES" in w_2). Here we assume the distance between two symbols is their difference in encoded English alphabets (A-Z are converted to 0-25) and the penalty p is 0.3. Arrows indicate which of the preceding positions are picked to generate score B. Two black arrows outside the table indicate the one-time point annotations of the gestures. Two clusters of connected scores show the longest matching substrings ("DRINKUP" with "DRINKA", and "OGGLES" with "OGGLESW"). The targeted cluster on the bottom left with the highest similarity score **M** = 5 spans the marked annotation. Hence, the corrected boundaries of two gestures are shown in green bars.

the final boundaries of the instance k are the average starting and ending time from all possible fixing boundaries.

2.2.2 Evaluation of Boundary Fixing Approach

We evaluated the boundary fixing approach on the Opportunity, HCI and Skoda data sets. In the Skoda and HCI data sets, we used a 3D accelerometer at the subject's dominant lower arm. In the quantization step, the number of symbols was selected k = 20. The non-motion removal step as presented above and illustrated in Figure 2.17 was applied. In the Opportunity data set, we used 6 different sensors worn at different on-body positions on one subject to achieve the best discrimination among gesture classes. Those sensors were given from the previous evaluation of our WarpingLCSS on multimodality on the Opportunity data set (see Section 2.1.3.6). They were 3D magnetic field on back, 3D gyroscope on right upper arm, 3D gyroscope and 3D magnetic field on right lower arm, 3D accelerometer on left upper arm and 3D accelerometer on left lower arm. The non-motion removal step was not applied for the Opportunity data set since a 3D accelerometer on dominant right arm was not used. The number of symbols was k = 120.

For each data set, we performed a 5-fold cross validation. From the training data set, we generated a one-time point annotation for each gesture by selecting randomly a point inside the ground truth. The boundary fixing approach was then applied to find boundaries for each gesture. The recognition system was trained on the fixed annotations and evaluated on the clean annotated test set. Two spotting techniques WarpingLCSS and SVM were used to evaluate the recognition system. We compared the performance of the spotting methods trained with ground truth annotations against those trained with the fixed annotations. We also evaluated their performances on the initial annotations which were extended loosely around the one-time points but were not corrected by our proposed BoundarySearch algorithm.

2.2.3 Results of Boundary Fixing Approach

2.2.3.1 Quality of Fixed Annotations

To evaluate the quality of fixed annotations compared to ground truth, we used a taxonomy of annotation noises that we summarized in Section 2.1.2. However, here we assume that the labels at one-time point annotations are correctly given, hence *substitute* and *insert* noises do not exist.

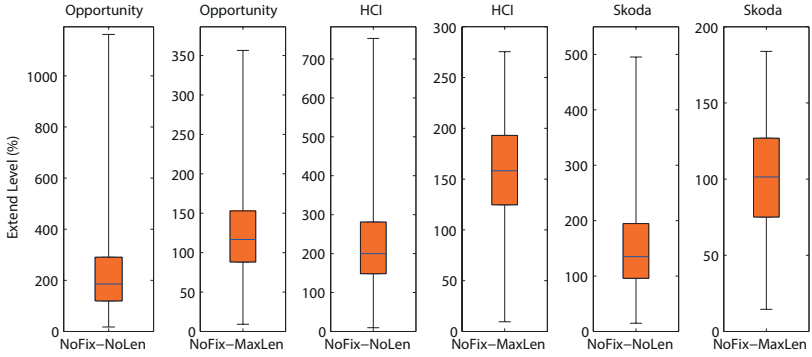

Figure 2.20: Extend levels of gesture boundaries initialized around one-time point annotations before BoundarySearch algorithm is applied. The box in the box plot covers from the 25th percentile to the 75th percentile (i.e., 50%) of the extend levels. Blue lines inside the boxes show the median values. Black lines show the minimum as well as the 98th percentile. Note that the scales are different for the sake of visibility.

Figure 2.20 shows the quality of the initialized boundaries of gestures in case of MaxLen and NoLen extensions before the BoundarySearch algorithm is applied, namely NoFix-MaxLen and NoFix-NoLen. Of gesture instances, at least 50%-75% are extended by more than 100%. There are only few gesture instances (about 4%) suffering less than 30% *extend* level. The *extend* levels of the initial gesture boundaries in HCI and Skoda data sets are much smaller than those in the Opportunity data set due to the use of the non-motion removal component. The prior knowledge of maximum length of each class reduces the *extend* levels significantly. However, the presence of many gesture instances with very large *extend* levels still makes NoFix-MaxLen unacceptable to train gesture models.

Figure 2.21 shows noise distribution and jitter levels of the fixed annotations after the BoundarySearch algorithm is applied in the NoLen case (namely Fixed-NoLen). Interestingly, there are some instances perfectly matched with the ground truth (they are shown as Good in Figure 2.21a) . The BoundarySearch algorithm reduces significantly the jitter levels in the initial boundaries. In the Opportunity data set, the maximum extend level before fixing is above 1000%, meanwhile

it decreases to 200% after fixing. More importantly, after fixing, up to 70%, 83% and 92% of instances in the Opportunity, HCI and Skoda respectively suffer less than 30% jitter levels.

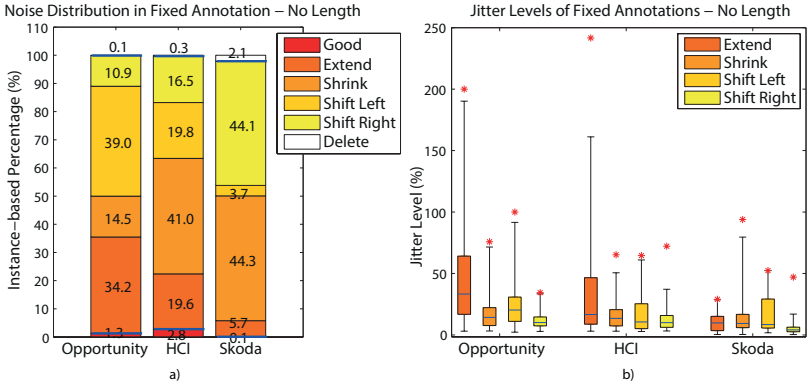

Figure 2.21: Analysis of fixed annotations in case the prior knowledge of maximum gesture length is not available (Fixed-NoLen). Blue lines in Fig. a) on the left split a boundary jitter part from good and delete types. In Fig. b), the description of box plot is the same as that in Figure 2.20. The red star indicates the maximum level of jitter in each type of boundary jitter.

The analysis of the fixed annotations in the case of MaxLen (namely Fixed-MaxLen) is given in Figure 2.22. Similarly, the BoundarySearch algorithm decreases the jitter levels significantly. The ranges of jitter levels in Fixed-MaxLen are smaller than those in Fixed-NoLen. After fixing, the percentages of instances having jitter levels less than 30% in the Opportunity, HCI and Skoda data sets are 78%, 87% and 92% respectively.

2.2.3.2 Recognition Performance on Fixed Annotations

Figure 2.23 shows the performances of WarpingLCSS and SVM on the fixed annotations (Fixed-NoLen, Fixed-MaxLen), on the non-fixing baselines (NoFix-NoLen, NoFix-MaxLen), and on the clean annotated annotation of the three data sets. In the Opportunity data set, in case the maximum gesture length is unknown, the performance of the fixed annotations Fixed-NoLen outperforms the nonfixing NoFix-NoLen by

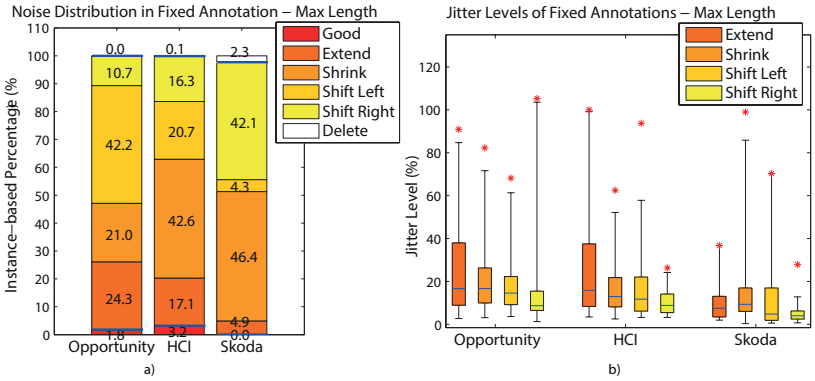

Figure 2.22: Analysis of fixed annotations in case the prior knowledge of maximum gesture length is given (Fixed-MaxLen). Interpretation of notations in this figure is the same as in Figure 2.21.

up to 58% F1-score for SVM and up to 29% F1-score for WarpingLCSS. We achieve the similar results in case of MaxLen. The performance of fixed annotations are decreased by just up to 10% F1-score compared to those of clean annotation for both recognition methods. In the HCI and Skoda data set, the fixed annotations in both cases NoLen and Maxlen can achieve the same performance as the clean annotations. They also outperform the non-fixing baselines by up to 40% F1-Null.

The results indicate that our boundary fixing approach can efficiently correct the boundaries of gestures around the one-time point annotation in both cases, whether the prior knowledge of the maximum length is given or not.

2.3 Personalized Adaptation on Crowdsourced-based Models

We proposed an adaptation on a generic model build on crowd-contributed free-annotated data to a personalized model in Chapter 8 to improve the recognition performance with no to little input from the user. We investigated two adapting approaches: 1) a semi-supervised learning to combine crowdsourced data and unlabeled user data, and 2) an active-learning to detect the most informative user-specific data samples that the crowd-based model fails to recognize and query the

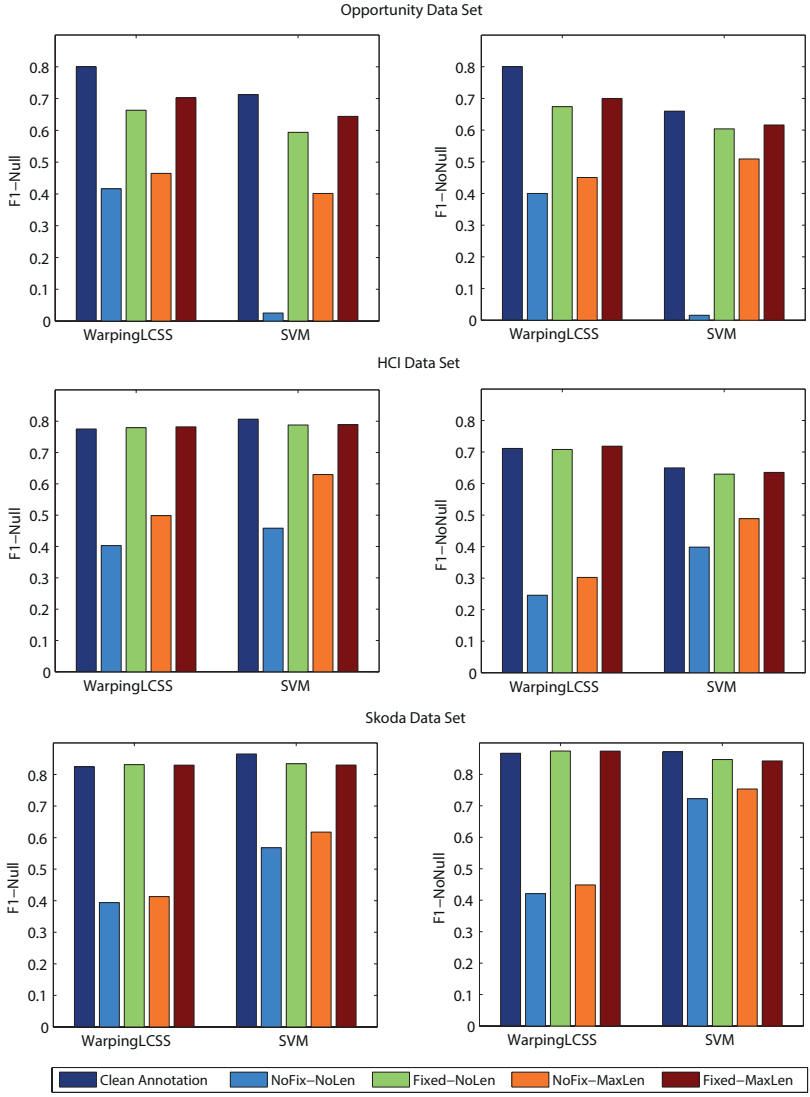

Figure 2.23: Performance of WarpingLCSS and SVM on fixed annotations in both cases of NoLen and MaxLen and on baselines.

user for labeling them. We illustrated the idea on an audio-based activity recognition system with the focus on high-level activities such as "working in the office", "having a conversation".

2.3.1 Personalized Adaptation System

Figure 2.24 shows an overview of a personalized adaptation system. In data preprocessing phase, auditory training data from Freesound [1] and user's mobile phone were collected. We then extracted acoustic features from the collected audio clips. In the learning phase, we applied machine learning techniques to learn and adapt an activity recognition model based on the two sources of data. In the recognition phase, the activity recognition model is used to infer user activity from data recorded on user's mobile phone. We summarize each component in our proposed system in the following.

User-centric Recordings: We recorded continuously two full working days of audio data from users' smartphones with a sampling frequency of 16 kHz and bit depth of 16 bits/sample in WAV format. Activity classes are about working, feeding, transportation and social interaction which are useful in health monitoring [47]. All participants live in Zurich, Switzerland and thus, the transportation includes tram, train, bus and car. Users can perform different set of activity classes, individual to their daily situations. Table 2.5 shows the list of classes provided by 7 participants and the corresponding distribution of classes.

Crowdsourced Repository: Freesound is an online sharing repository of crowd contributed sound data. Sounds are annotated in free-form styles and the tags come from very diverse vocabularies [48]. From the list of activity classes provided by the users, we download free annotated training audio data for those activity classes from Freesound (30 sound clips per activity class). We manually filtered the downloaded audio clips that are irrelevant to the assigned context class. The filtering task can be done quickly by listening the audio clips. Data downloaded from Freesound were also converted to the same format as user-centric data (a sampling frequency of 16 kHz and bit depth of 16 bits/sample in WAV format).

[1] www.freesound.org

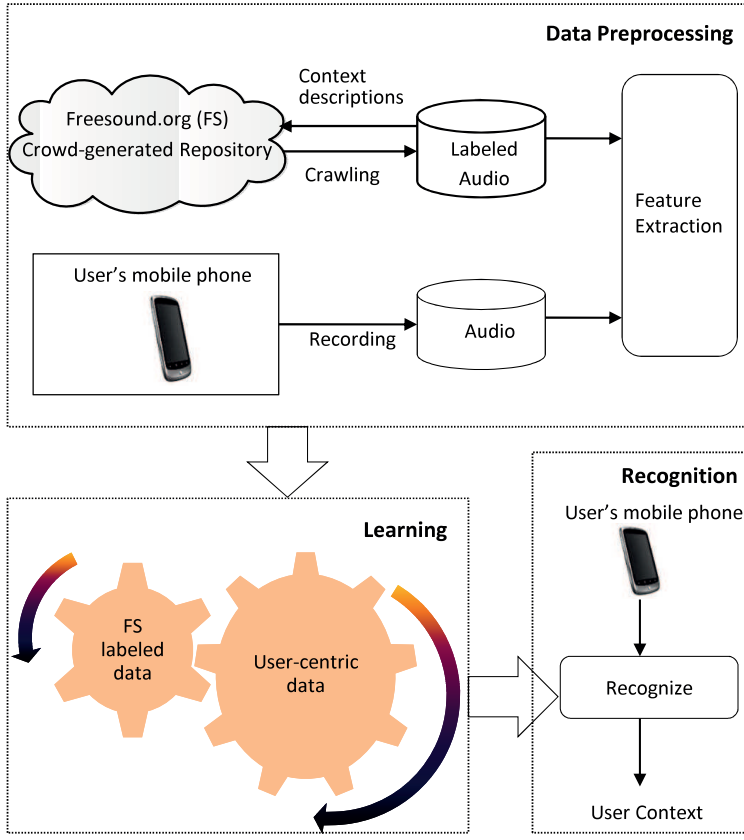

Figure 2.24: The data processing flow of the sound-based activity recognition that combines Freesound data and user-centric audio data from mobile phone

Audio Features: We extracted 12 coefficients mel-frequency cepstral coefficient (MFCC) and log-energy in a sliding window of 32 ms length of audio data.

Learning Approaches: We used semi-supervised learning [34] and active learning [35] schemes based on Gaussian Mixture Model (GMM) [49–53] for the adaptation, namely *Semi-supervised Adaptation* and

Table 2.5: User-dependent activity classes and the corresponding distribution of classes in dataset

	Context Classes and Class Distribution (%)
User 1	office (83), tram (1), train (10), conversation (6)
User 2	toilet (1), office (50), restaurant (5), street (1), conversation (43)
User 3	office (37), restaurant(7), street(12), tram(2), conversation(42)
User 4	toilet (1), office (70), restaurant(2), street (4), tram (1), conversation (22)
User 5	toilet (1), office (63), restaurant (7), street (7), tram (1), train (7), conversation (14)
User 6	toilet (0.4), office(70), restaurant (8), street (4), tram (6), train (5), car (1), conversation (5.6)
User 7	toilet (0.2), office (21), restaurant (9), street (4), tram (5), train (6), car (2), bus (0.2), conversation (52.6)

Active Learning Adaptation. *Semi-supervised Adaptation* combines both Freesound labeled data and user unlabeled data to train activity models. The assumption is that nearby data points are more likely to share a label. Hence, taking user unlabeled data in training can adapt the decision boundaries towards user data points. In the *Active Learning Adaptation*, we first trained a bootstrapped activity model using Freesound labeled data. From that initial model, active learning proceeded and iteratively selected the most informative user-centric samples to query for labels. The activity model was then retrained and adapted with the new user labeled data. Instead of asking a label for only a feature-based point which is extracted from the 32 ms data segment, we ask a label for a longer segment ($\approx$ one minute) around the queried instance.

Recognition Approach: The trained GMM models were used to recognize user activities based on audio data recorded from the smartphone. We constructed a two-level classification. At the low level, audio instances extracted from windows of 32 ms were classified by the GMM models. At the high level, a decision was made on the longer segment (2 seconds) by taking a class with the highest occurrence frequency in the segment as a label.

2.3.2 Evaluation and Results

Each user recorded two full working days of audio data in his ordinary setting with android smartphones. For each recording day, at least 9 hours of audio data were obtained for each user. As can be seen in Table 2.5, users spent most of the time in the office and discussed their works with colleagues. In total, about 130 hours of audio data were collected from mobile phones for the study. The number of activity classes vary from 4 to 9 classes. The audio recording of each subject was divided into two equal halves. The first half was used for training and the second one was used for testing.

We compared our proposed approaches with three baseline non-adapted learning approaches:

- Freesound fully supervised approach (*FS-Supervised*) that used only Freesound data for training
- User fully supervised approach (*User-supervised*) that used only user annotated data for training
- *Active leaning (AL) from User* that initially took one minute of user labeled data for each activity class for training, and then the active learning was applied to query labels for the uncertain samples.

We evaluated them in terms of accuracy and labeling effort. Table 2.6 gives the accuracy of five approaches. For the active learning approaches, we only show here the best performance over the first 20 label queries (see Table 8.7 for the details of which activity classes were queried and Figure 8.3 for the detailed performance of the active learning on each label query).

The *Semi-supervised Adaptation* improves the recognition accuracy of the FS-Supervised up to 21% for six users. However, when the assumption fails (i.e., close data points may not share a label), unlabeled user data in the semi-supervised learning makes the model more uncertain and the performance degrades as shown in scenario of user 3. The results also show that *FS-Supervised* and *Semi-supervised Adaptation* underperform significantly the baseline *User-Supervised* approach. Fully supervised training on user data clearly captures user-specific contexts in the model, meanwhile the crowd-sourced data hardly covers every aspects of user-specific surroundings. For user data points that the crowsourced data can not represent, both *FS-Supervised* and

Table 2.6: Accuracy of learning approaches for 7 users. For active learning (AL), the best performances over 20 label queries are given. (Table 8.5, page 200)

	FS-Supervised	Semi-supervised Adaptation	AL Adaptation	AL from User	User-Supervised
User 1	0.8	0.86	**0.97**	0.93	0.94
User 2	0.5	0.65	**0.94**	0.82	0.9
User 3	0.58	0.43	**0.81**	0.73	0.72
User 4	0.22	0.25	**0.93**	0.86	0.72
User 5	0.35	0.5	0.80	**0.83**	0.82
User 6	0.54	0.61	0.86	**0.87**	0.85
User 7	0.26	0.47	**0.86**	0.76	0.83

Semi-supervised Adaptation fail to model them. However, their accuracy is inverse to the labeling effort on user data. While *FS-Supervised* and *Semi-supervised Adaptation* do not require any effort to label user data, *User-Supervised* requires labels from all user training data.

Both *AL Adaptation* and *AL from User* approaches reach the performance of the *User-Supervised* approach over 20 label queries. However, the active learning technique require a significantly fewer amount of labeled training data (20 queries ≈ 3% of the user training data). Interestingly, the *AL Adaptation* outperforms both *AL from User* and *User-Supervised* learning. The rationale is Freesound contains intra-class diversity, thus it may contain user's unseen contexts in the recognition phase and increase model generalization.

To compare the *AL Adaptation* and *AL from User* in terms of number of label queries, we evaluated how many label queries needed for these two approaches to reach the same performance as the User-Supervised approach. For the *AL from User* approach, we also counted the annotation effort of user to contribute the initial labeled training set to build the initial classifier. The results show that the *AL Adaptation* requires in average five label queries per user (≈ 0.7% of user training data) to get the good performance. Meanwhile, the *AL from User* asks for in average at least 24 label queries per user (≈ 3.6% of user training data).

The result emphasizes that Freesound contains diverse and useful acoustic data that represents partly user data and can be used to recognize user context. To enforce the analysis, we show the confusion

matrix of the approaches for one user in Figure 2.25 . As can be seen, the *FS-Supervised* can recognize well "restaurant" and "tram" classes, and it confuses "office" with "toilet", and "street" with "tram". The *AL Adaptation* needs to ask few labels for the confused classes to catch up the performance difference.

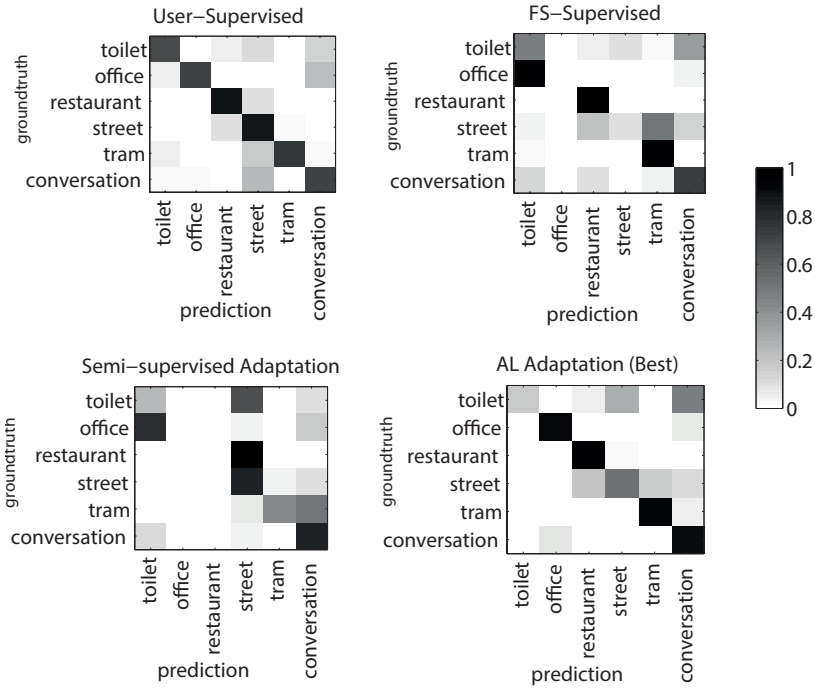

Figure 2.25: Confusion matrix tables of learning approaches from one user. (Figure 8.5, page 206)

2.4 Conclusion

Activity recognition from body-worn sensors provides a valuable tool to monitor user's activities wherever he goes. The state of the art approaches require a clean annotated training data provided by experts to model activities. While it provides high accuracy, it is very time consuming and limits the system to a small data set.

This thesis introduced and demonstrated the use of crowdsourcing to reduce the effort to collect annotation from training data set in activity recognition systems, but still achieve as good performance as experts' annotation. In one aspect of crowdsourcing, a large recording of activities is annotated explicitly by crowdsourced labelers in which temporal boundaries and labels of activities are specified. Another aspect of crowdsourcing is to use online available crowd-generated sharing database to extract the relevant training set and then personalize the general crowdsourced-based models with a small number of user input. From the summary of contributions from Section 2.1 to Section 2.3, we draw the following conclusions:

- Crowdsourced gesture annotation from AMT gets high quality (about 80% sample-based accuracy) if multiple crowdsourced labelers are applied. Otherwise, the annotation contain a large presence of noises (52% of instances are label noise) and the noisy annotation degrades the state-of-the-art activity recognition methods significantly (e.g., the performance of SVM was similar to a random guess).

- Our proposed WarpingLCSS is a linear time method which is suitable for online gesture recognition. The WarpingLCSS is robust to crowdsourced annotation noises and achieves better performance than the DTW-based methods and SVM, especially with the large presence of noise. With 60% mislabeled instances, WarpingLCSS outperforms SVM by about 22% F1-score and outperforms DTW-based methods by 36% F1-score. Moreover, WarpingLCSS can tolerate 30%-40% jitter level. Additionally, WarpingLCSS can be used as a pre-processing filtering component to clean noisy crowdsourced training data for other supervised techniques. Besides the robustness against labeling noises, WarpingLCSS shows their efficiency on clean annotated data sets as well as on multimodal sensor systems.

- We introduced the one-time point annotation technique as a special case of annotation noise (the boundary shrinks to a point), but to reduce the burden in activity annotation significantly. The novel BoundarySearch algorithm searches for good boundaries of an activity based on activity patterns around their one-time point annotations. The performance on the corrected annotations is just lower than the training on well-annotated annotations by

3% F1-score, but it reduces significantly the amount of annotation time.

- We investigated the use of online crowd-generated audio repository Freesound to train a base model for user daily activities. An adaptation with semi-supervised learning improves the accuracy of the crowdsourced models up to 21% without asking labeling on user data, but is still far to achieve the performance of the supervised learning on user data. The active learning adaptation achieves the performance of the user supervised model with only a few label queries.

2.5 Limitations and Outlook

The thesis demonstrates the potential of crowdsourcing to collect a large annotated data set efficiently and introduces robust learning methods to achieve a good performance. However, the following limitations are identified:

- Generalizability of algorithms: The WarpingLCSS and BoundarySearch algorithm were investigated on activity recognition with wearable sensors. However, they are both generic algorithm that can be useful in other applications of pattern recognition. The WarpingLCSS should be investigated for other general time series data. The BoundarySearch algorithm can be generalized to applications that need to find similar patterns inside two strings.

- Noise in one-time point annotation: In the work of one-time point annotation, we assumed that all activities were annotated and each annotated point was associated with a correct label. However, in crowdsourced annotation scenarios, annotation noises are likely to happen. In that case, the outcome of fixed boundaries would be the noisy annotated training data set. Consequently, the WarpingLCSS can be used to filter out those noisy instances before training a classifier. A further study on this direction should be investigated.

- Number of Participants: In the study of personalized adaptation on crowd-based models, while the proposed active learning adaptation achieves good performance, the subject pool was with about 7 subjects rather limited. Moreover, they lived in the same

city, thus the sound-based contexts were constrained on that city only (e.g. user living in Zurich would use public transportation such as tram, train, bus). To draw a more general conclusion, more subjects from different contexts should be considered.

- User Input: In the study of active learning adaptation on crowdsourced models, we assumed the label feedback from users was correct. However, those labels can be verified easily by acquiring labels explicitly for audio data with multiple labelers from a crowdsourcing platform like AMT.

- User-dependent WarpingLCSS: WarpingLCSS was evaluated for user-dependent activity recognition which requires a training template for each activity of interest from each user. Even though it can be considered as a limitation, we believe it is not. In crowdsourcing context, each user can record his own data with his own activities of interest. The corresponding labels can be given by himself or by crowdsourced labelers. The system does not ask for a perfect annotation, but can support a user-dependent recognition which often achieves high performance due to the best match between the training data and testing data. A step further to support the user-independent recognition can be a deployment of a crowdsourced repository to collect many gesture templates contributed from a large amount of users with a goal to generalize the gesture patterns for all users.

3

A Case Study on Activity Annotation by Crowdsourcing

Long-Van Nguyen-Dinh, Cédric Waldburger, Daniel Roggen, Gerhard Tröster

Tagging Human Activities in Video by Crowdsourcing

Proceedings of the International Conference on Multimedia Retrieval (ICMR'13), pp. 263–270, Dallas, TX, USA, 2013.

DOI: 10.1145/2461466.2461508

Abstract

Activity annotation in videos is necessary to create a training dataset for most of activity recognition systems. This is a very time consuming and repetitive task. Crowdsourcing gains popularity to distribute annotation tasks to a large pool of taggers. We present for the first time an approach to achieve good quality for activity annotation in videos through crowdsourcing on the Amazon Mechanical Turk platform (AMT). Taggers must annotate the start, end boundaries and the label of all occurrences of activities in videos. Two strategies to detect non-serious taggers according to temporal annotated results are presented. Individual filtering checks the consistence in the answers of each tagger with the characteristic of dataset to identify and remove non-serious taggers. Collaborative filtering checks the agreement in annotations among taggers. The filtering techniques detect and remove non-serious taggers and finally, the majority voting applied to AMT temporal tags to generate one final AMT activity annotation set. We conduct the experiments to get activity annotation from AMT on a subset of two rich datasets frequently used in activity recognition. The results show that our proposed filtering strategies can increase the accuracy by up to 40%. The final annotation set is of comparable quality of the annotation of experts with high accuracy (76% to 92%).

3.1 Introduction

Activity recognition is useful in many applications such as ambient assisted living, human-computer interaction, video surveillance, or activity life logging. Human activity can be extracted and recognized from video footage, or data streams from on-body sensors, such as inertial measurement units. Regardless of the modality used to recognize activities, a labeled training dataset is required for supervised learning [54–56]. The training dataset must comprise the start and end time of the activities of interest. This is usually obtained by manual inspection of a video footage of an experimental recording where users demonstrate the activities of interest. Even when using on-body sensors for recognition, a video footage is shot for the purpose of labeling. Video labeling is extremely time-consuming and tedious: it may take 7-10 hours to annotate fine-grained activities in a 30-min video [12]. It is also costly to hire experts to do labeling.

In order to reduce cost and time of data labeling, crowdsourcing platforms (e.g., Amazon Mechanical Turk (AMT), Crowdflower) has

become a new trend. Crowdsourcing platforms allow a large number of non-experts from all over the world and without any specific background to solve large-scale tasks for a small financial incentive. Therefore, crowdsourcing is generally employed for tasks that are easy for humans, but hard for computers. Since human activity in a video can be easily recognized by non-experts, we are interested in answering the following question: "Is crowdsourcing an alternative reliable way to get activity labeling from video footage?".

In this work, we investigate the ability of AMT workers and their behavior in annotating temporal boundaries and labels of activities occurring in videos. An approach to achieve good quality for activity annotation in videos through AMT and to handle temporal results from AMT is presented for the first time. Two filtering strategies to detect and remove non-serious taggers according to temporal annotated results are proposed and evaluated. Individual filtering checks the consistence in the answers of each tagger with the characteristic of dataset to identify and remove non-serious taggers. Collaborative filtering checks the agreement in annotations among taggers to detect non-serious taggers. After filtering, the majority voting applied to AMT temporal tags to generate one final AMT activity annotation set. We conduct the experiments to get activity annotation from AMT on a subset of two rich datasets frequently used in activity recognition (CMU and Opportunity [12, 38]). The final AMT annotation set is then compared with the ground truth annotated by experts to evaluate the quality of annotation from crowdsourcing.

3.2 Related Work

Crowdsourcing services (e.g., Amazon Mechanical Turk (AMT), crowdflower[1], clickworkers[2]) has emerged recently as a new cheap labor pool for simple large-scale tasks. Crowdsourcing tasks can typically be accomplished easily and quickly by large number of workers. Crowdsourcing has been characterized in the annotation of datasets in natural language processing [19–21], speech recognition [22, 23], multimedia tagging [24–27]. It has also been proposed in query processing [57] to answer queries that can not be answered by database or search engines. Data acquired from crowdsourcing is generated by

[1] http://crowdflower.com
[2] http://clickworkers.com

low-commitment workers, thus it is commonly unreliable and noisy. 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 [20,28]. Moreover, in crowdsourcing, malicious workers often take advantage of the verification difficulty (the ground truth is unknown) and submit low-quality answers. Hence, it is necessary to include strategies to estimate the quality of workers in order to reject low-performing and malicious workers. The acceptance rate of a worker based on their work history is usually specified as a threshold to allow that worker to participate in the task. Verifiable questions or pilot tasks for which the requester knows the correct answers is a common empirical strategy to screen workers from crowdsourcing [22,23,39,40]. Dawid and Skene [58] proposed a method that used the redundancy in acquiring answers to measure the labeling quality of the workers based on an expectation maximization algorithm. Bayesian versions of worker quality inference were recently proposed by Raykar et al. [59]. Ipeirotis et al. [60] improved the method by separating spammers who provide low-quality answers intentionally and biased workers who are careful but biased. The biased answers can then be recovered and yields much higher quality of results.

Amazon Mechanical Turk: Amazon Mechanical Turk is by far the most popular crowdsourcing platform with almost half a million turkers (i.e., workers) and about 50,000 - 100,000 HITs [3] available to the turkers at any time [61]. Therefore, in our work, AMT is chosen to evaluate activity annotations in videos from crowdsourcing. In AMT, turkers can choose available HITs to complete and submit their results to AMT. The requester of the HIT retrieves all results from AMT after the HIT is accomplished. According to the quality of turkers' answers, the requester approves or rejects their work. A requester design a HIT template to describe the task they would like to distribute. AMT supports a number of template HITs, command line tools and developer APIs. The requester can define how many assignments (i.e., workers) per HIT are needed, duration for each HIT and cost per HIT. A worker is allowed to work only one assignment per HIT. To assure quality of works from turkers, *turker approval rate* which is provided by AMT according to turker's work history can be used as a threshold for eligibility to work. Specifically, turker approval rate is the percentage of turker's works accepted by requesters.

[3] HIT = Human Intelligence Task represents a small task assigned to turkers with an allocated price and completion time

According to the best of our knowledge, there is no previous work that investigates the use of crowdsourcing in activity annotation in videos in which workers provide the starting and ending time of occurrences of activities. In the work by Zhao et al. [25], they extracted individual still images/video frames from the CMU video of activities [38] and then acquired labeling for the activity occurring in the image from crowdsourcing. However, using one video frame to ask for activity may have the limitations of ambiguation. It is more likely for users to recognize the activities by watching the continuous sequence of frames (e.g., it is hard to distinguish Open Door and Close Door activities with just only one frame of activity). It is more natural to ask Amazon Mechanical Turkers to watch a video and annotate activities occurring on the video.

3.3 Crowdsourcing methodology

To get activity labeling from turkers, a HIT interface is needed. Video footages for activity recognition are usually recorded at several perspectives in order to capture all activities. Therefore, the HIT interface should show different video perspectives synchronously. Turkers can navigate through videos and indicate the start, end times and labels of all activity instances occurring in the videos. In this section, we describe our HIT interface, technical details to run our HIT in AMT, and a data processing pipeline to evaluate answers from turkers and fuse their answers to get a final activity annotation.

3.3.1 HIT Interface Design

Figure 3.1 shows our HIT interface to collect activity annotation from different synchronized video perspectives. The interface consists of 4 different parts indicated with a color bar on the left which does not appear in the real interface.
Task Description (blue): At the top, we describe the task that the turkers have to solve in order to get their answers accepted. Turkers have to answer correctly questions in the red section and specify start/end/tag triples for all occurrences of activities performed in the video in the yellow section. We estimate the minimum number of tags (N) labeled from the videos according to the approximation of activity duration. We let turkers know about this number N so that turkers can work carefully on our tasks in order to get all activities. Since we do not

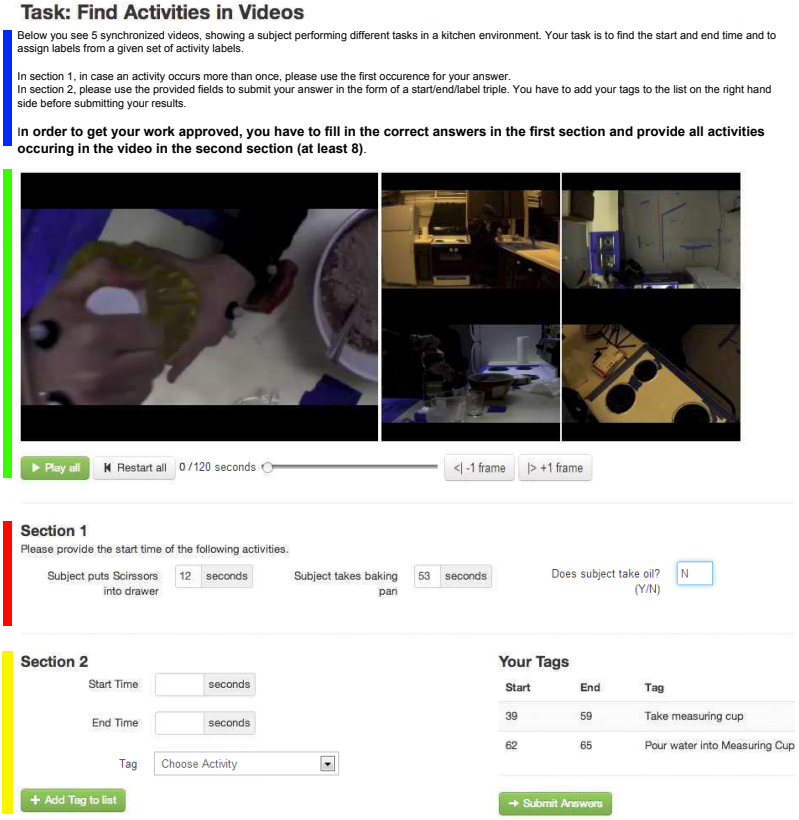

Figure 3.1: The HIT interface to collect activity annotation from different synchronized video perspectives.

know the exact number of activities in each video, therefore, we use this N to verify the quality of turkers. It will be explained in Section 3.3.3.

Video Player (green): This part allows to play all videos synchronously and to pause or restart all of them (turkers can play videos again to check their answers). It also shows the current time in seconds. Moreover, turkers are able to transverse through all videos synchronously. To support turkers to find the exact starting and ending time of an activity, we also support buttons to go forward and backward in videos

frame by frame.
Section 1 - Qualification test (red): As there are low-commitment turkers on AMT, we have to check whether an answer could be taken seriously or if it is just a random submission or spam. We use verifiable questions to screen workers as in [22, 23, 39, 40]. We ask for the starting time of 2 activities in the given video sequence and one boolean question about whether an activity occurs in the video. It does not take much time for requesters to prepare answers for those verification questions but it can ensure that turkers must watch the whole short video to answer correctly this section 1. Different HITs have different verification questions since they contain different videos. The questions also prepare for turkers what they will expect to do in the next section.
Section 2 - Tag Section (yellow): This is where turkers are asked to supply triples of start/end/label of all occurrences of activities in the videos. Possible activities are given in a list. The right-hand side lists all previously given triples for turkers to check and allows them to submit all tags at the end of the task.

We do not receive any negative feedback about our interface during the experiments which assures us that the interface works fine.

3.3.2 Running HIT in AMT

AMT supports a number of template HITs, but unfortunately does not offer a template to show videos synchronously, which is needed in our work to show different video perspectives at the same time. However, AMT can load custom tasks from an external server. External tasks can consist of HTML/Javascript to design a HIT Interface. To ensure that every turker has the same experience when working on our video tagging tasks, the interface should work correctly and look the same in every major browser. To meet this requirement, we use HTML5 and webm and h264 mp4 formats for videos. We test our interface with all major browsers including Chrome, Safari, Firefox, Opera and Internet Explorer. HTML5-Video implementation instead of Flash is also compatible with mobile browsers.

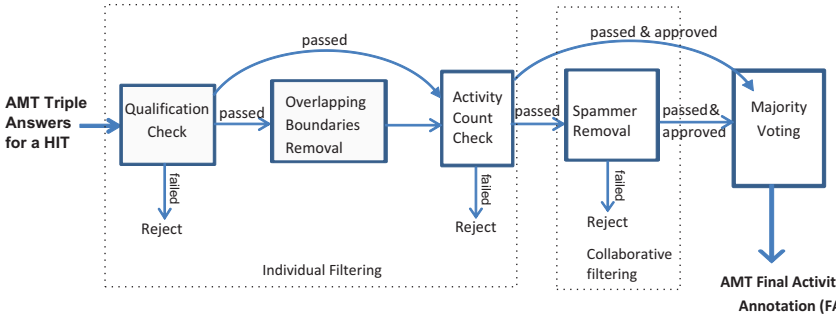

Figure 3.2: A data processing chain taking raw AMT output from turkers and producing a final activity annotation by majority voting. The output from turkers is evaluated with different components enabled or not.

3.3.3 AMT Annotation Post-processing

In this section, we introduce a data processing pipeline to evaluate the quality of answers from turkers and fuse the answers to get a final activity annotation. The quality of answers decides whether we reject or approve an assignment of a HIT. Basically, assignments must fulfill all requirements specified in the HIT in order to get approved. In our work, we have two requirements: 1. Correctly answer three verifiable questions in the section 1 in the interface, and 2. Specify all occurrences of activities in videos. However, since we do not know the exact number of activity instances in each video, we check whether turkers provide at least N activity tags performed by a subject in the video. We propose two kinds of filtering techniques to improve the quality of the final activity annotation: individual filtering and collaborative filtering. Individual filtering examines answers in each assignment to decide whether it should be accepted or not. Collaborative filtering examines answers from all assignments to a HIT to detect and remove spammers, thus reject or accept the work.

Figure 3.2 shows different components of the data processing pipeline. The individual filtering can contains three components: Qualification Check, Overlapping Boundaries Removal, and Activity Count Check. The collaborative filtering contains a component: Spammer Removal. We describe each components as follows.

1. Qualification check: Assignments must answer correctly at least M verifiable questions over three to get into other steps. M = 1,2, or 3. Otherwise, they get rejected. Different turkers may have different decision when an activity starts (e.g., drink gesture can start from the time user picks up a cup or when user starts drinking). Therefore, we allow that the starting time answers can be different from the true answers within 2 seconds.

2. Overlapping Boundaries Removal: Activities of interest are performed and recorded in sequence, and thus non-overlapping. However, we observe that spammers tend not watch the videos and provide randomly overlapping starting and ending times of activities. To increase the quality of the final activity annotation, we remove activities tagged by a turker which are boundary overlapping.

3. Activity Count Check: The number of activities provided in the section 2 must be at least N. Otherwise, it is rejected.

4. Spammer Removal: We define spamming/outlier to be assignments which disagree with the majority most of the time. Specifically, the assignment which have a disagreement score d, $d = \frac{\text{Tagging times disagree with majority}}{\text{Total tagging times}} > \textit{threshold}$ is a spam. Score d is computed as follows.

    - Step 1: We extract starting and ending times of all tagged activities from assignments of a HIT and put into a sorted list.

    - Step 2: We scan through each temporal segment S (i.e., two consequence elements) in the sorted list, and the label of S is the majority voting among all activities containing this segment.

    - Step 3: For each assignment having tags for S which disagree with the majority, the score d is accumulated by the length of S. At the end, d is compared with the threshold and spammers are detected and removed.

5. Majority Voting: At the end, the majority voting among qualified assignments is performed to generate a final list (FA) of good

annotated activities. The list is then compared with the ground truth annotated by the experts for evaluation. The algorithm to get majority voting includes step 1 and 2 in the Spammer Removal section above. Table 3.1 shows an example to get majority among temporal segments of activity annotation.

Table 3.1: An Example of Majority Voting

start	end	tags (label:count)	majority vote
t1	t2	open Drawer: 5, take Scissors:2	open Drawer
t2	t3	open Drawer: 5, take Scissor:1, take Egg:1	open Drawer

3.4 Evaluation

3.4.1 Datasets

The experiments are conducted on video footages from two public datasets for activity recognition: the Carnegie Mellon University (CMU) Kitchen dataset [38] and the Opportunity Dataset [12].

CMU Kitchen Dataset [38]

The CMU Kitchen dataset consists of videos and signals from other modalities (e.g. IMU, eWatch) recorded from different subjects performing different recipes in a kitchen. There are 5 video streams at different locations in the kitchen capturing all activities performed by the subjects. The ground truth labels which are annotated by the authors of the dataset are available for 10 subjects who baked the "brownie recipe". To compare our experiments with the ground truth provided by experts, we use one set of brownie-recipe videos of one subject which is labeled by CMU people to ask for activity annotation from turkers. Activities for a Brownie recipe in CMU datasets are listed in Table 3.3. The video duration is 6 minutes long. We segment the video into 3 short videos of about two minutes. We approximate about 8-14 activities instances occurred in each 2-min video. It is reasonable to ask turkers working on the short videos that contains a small number of activity instances. Five synchronized segments from 5 videos are shown in a HIT for annotation. The HIT interface for the CMU dataset is shown in 3.1.

Opportunity Dataset [12]

The Opportunity Dataset is a rich multi-modal dataset collected in a naturalistic environment akin to an apartment, where users execute daily gestures. There are 3 video streams captured from 3 different positions in the room synchronized with body-worn sensor signals. In our work, we use the videos of one subject performing 20 repetitions of 17 gesture classes as shown in Table 7.1. The video duration is 25 minutes long. We segment the videos into 30 short videos of 50 seconds. We estimate about 8-14 activity instances occurred in each video segment. We publish 30 HITs for the annotation for the Opportunity dataset. Note that in this dataset, there are three drawers at different heights and two different doors. The HIT interface for the Opportunity dataset is similar to 3.1, however we place a map of the kitchen in the task description part to show turkers where the door 1,2 or drawers 1,2,3 are.

Table 3.2: Gestures in Opportunity dataset

drink Cup (D)	clean Table (CT)	open Drawer1 (ODr1)
close Drawer1 (CDr1)	open Drawer2 (ODr2)	close Drawer2 (CDr2)
open Drawer3 (ODr3)	close Drawer3 (CDr3)	open Door1 (OD1)
close Door1 (CD1)	open Door2 (OD2)	close Door2 (CD2)
open Fridge (OF)	close Fridge (CF)	Toggle Switch (TS)
open Dishwasher (ODi)	close Dishwasher (CDi)	

Table 3.4 shows an overview of the experiments we conduct. We publish 3 HITs from the CMU dataset and 30 HITS from the Opportunity dataset. Each HIT has 10 assignments. Each assignment is annotated by a turker who must have at least 90% approval rate on their work history and different assignments of a HIT are completed by different turkers. We pay 30 cents for each assignment.

3.4.2 Evaluation on AMT Final Activity Annotation

To evaluate the quality of the AMT final activity annotation (FA) for a HIT, we compare the FA with the ground truth (GT) using an accuracy metric. The FA and GT are sets of <start,end,tag> triples. Each set contains non-overlapping activities. In our HIT, we do not ask turkers to annotate Null activities, we default the temporal segment without any tag as Null. The accuracy is defined as follows.

$$\text{Accuracy} = \frac{\text{Total match length between FA and GT}}{\text{Total annotation length}}$$

Table 3.3: Brownie-recipe Activities in CMU dataset

walk to Fridge	pour Water into Big Bowl
close Fridge	open Fridge
open Brownie Bag	open Brownie Box
crack Egg	open Cupboard Top Left
open Drawer	pour Big Bowl into Baking Pan
stir Big Bowl	pour Brownie Bag into Big Bowl
stir Egg	pour Oil into Small Measuring Cup
switch oven on	pour Water into Big Measuring Cup
take Fork	put Baking Pan into Oven
read Brownie Box	put Scissors into Drawer
take Oil	put Oil into Cupboard Bottom Right
take Big Bowl	take Brownie Box
take Egg	take Big Measuring Cup
take Baking Pan	take Small Measuring Cup
take Scissors	twist off Cap
twist on Cap	walk to Counter

Table 3.4: AMT Experiment Summary

	# HITs	Assignments per HIT	Length(s)	Price per HIT (USD cent)	Turker Approval Rate
CMU	3	10	120	30	90
Opportunity	30	10	50	30	90

The total annotation length is the length of the HIT's video sequence. The total match length (ML) between the FA and the GT is the total length of temporal segments where both FA and GT agree on the activity tag. The algorithm to compute ML is discussed briefly. We extract start and end times of all triples in both FA and GT and put into a sorted list. We scan through each temporal segment S (i.e.,two consequence elements in the sorted list), if FA and GT agree on the activity tag for this segment, ML is increased by the length of S.

3.5 Results and Discussion

3.5.1 Duration of task completion

Table 3.5 shows the average time that the CMU and Opportunity HITs need to be completed. **HIT Time** measures the time from the moment when a turker starts a HIT until he submits his results. **Completion Time** measures how long it takes to get all assignments on a HIT completed.

Table 3.5: Time Requirement Overview

	HIT length (seconds)	Average HIT Time (mins)	Average Completion Time (hours)
CMU	120	14	62
Opportunity	50	12	71

The Opportunity and CMU HITs takes in average about 2-3 days to complete. Average HIT Time (12 mins for a 50-sec video) is comparable to works done by experts (7-10 hours for a 30-min video), thus turkers annotate data at a similar speed as the experts. The approximate price we pay for 30-min length video is 30 (cents) * 30 (mins) * 5 (accepted assignments for a HIT) $\approx$ 50$, which is significantly less than what we spend for an expert per day to annotate videos.

3.5.2 Turkers Statistics

We recruit totally 136 turkers working on our 33 HITs (118 turkers for the Opportunity HITs and 18 turkers for the CMU HITs). Figure 3.3 shows the histogram of accepted assignments for turkers working on our HITs. There are 38 turkers working on only one HIT and doing well. However, 26 turkers finish more than one HIT correctly. It is interesting that there are 4 dedicated turkers working successfully on more than 10 different HITs. It seems like they find our tasks pleasant.

Figure 3.4 shows the histogram of rejected assignments for turkers working on our HITs. There are 70 turkers getting one assignment rejected (70/136 $\approx$ 51%). Only four workers failed the HITs 4-6 times. Figure 3.5 shows the histogram of turker accepted rate evaluated from the work on our HITs. There are two peaks at 0% accepted rate and 100% accepted rate. Together with Figure 3.3 and 3.4, the results show that there are many turkers work on our HIT once and fail. It could be

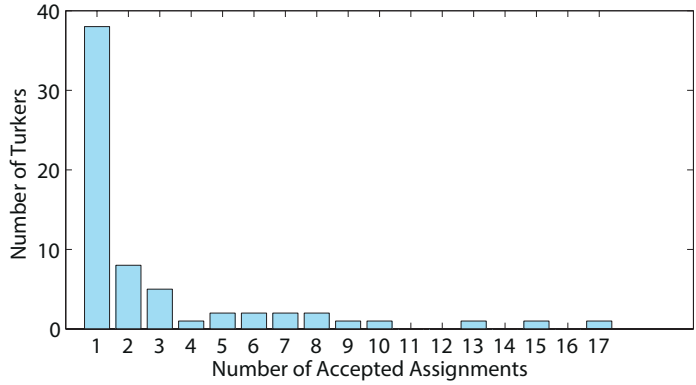

Figure 3.3: The histogram of accepted assignments for turkers working on our HITs

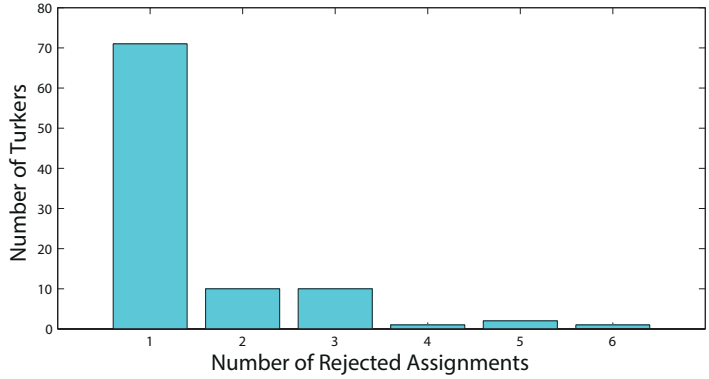

Figure 3.4: The histogram of rejected assignments for turkers working on our HITs

that either they are spammers or they find our HIT difficult. However, many turkers succesfully work on our HITs multiple times.

3.5.3 Accuracy

To understand the quality of annotation from AMT for our HITs, we define six types of post-processing from the raw AMT results to get

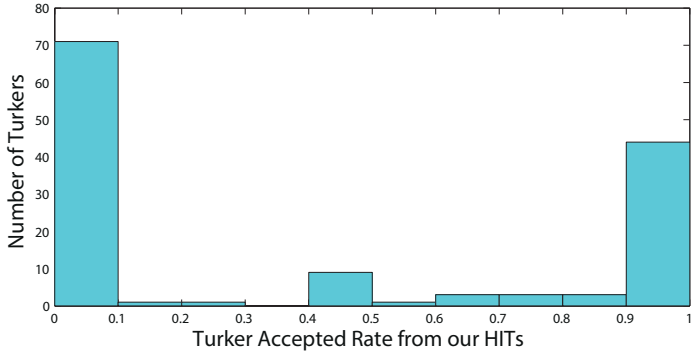

Figure 3.5: The histogram of turker accepted rate evaluated from the work on our HITs

the FA by combining different components in Figure 3.2.

- Type 1 (No Check): In this type, all the answers from all assignments for a HIT are considered in majority voting step to get the FA.
- Type 2 (Qualification Check): In this type, all assignments that pass the qualification check will come to the majority voting step to get the FA.
- Type 3 (Qualification Check + Activity Count Check): Parameter N = 8 for the HITs
- Type 4 (Qualification Check + Overlapping Boundaries Removal + Activity Count Check): Even serious turkers may make a mistake, hence, we accept at most 2 overlapping tags and after overlapping boundaries removal, we decrease the required number of activities by 2. Thus, in this type, parameter N = 6 for the HITs.
- Type 5 (Qualification Check + Activity Count Check + Spammer Removal): Parameter N = 8 for our HITs.
- Type 6 (Check All): All checks are performed. Parameter N = 6 for our HITs.

In both Opportunity and CMU videos, the number of activity instances occurring in a segmented video is about 8-14. Thus, in our evaluation, we choose parameter N = 8 for Activity Count Check.

We find out that even dedicated workers who provided good tags for activities occurring in videos but still made mistakes in answering verifiable questions. Hence we relax the qualification check to remove out only the assignments with at most one question answered correctly (i.e., parameter M = 2). In Spammer Removal, we choose a threshold = 0.3, it means if the disagreement score d >= 0.3 (i.e., less than 70% of annotation of the assignment agrees with the majority), the assignment is a spam and removed. For each HIT, we also compute the accepted rate(i.e., number of accepted assignments/total assignments per HIT) for type 6 only. It shows the real rejected and accepted rate we respond to turkers for each HIT.

Table 3.6 shows the accuracy, the average number of activities provided from accepted assignments for different types of evaluation of CMU HITs and the accepted rates.

Table 3.6: Results for 3 HITs in CMU dataset

Accuracy						
	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6
CMU HIT 1	0.71	0.72	0.72	0.77	0.76	0.77
CMU HIT 2	0.78	0.82	0.82	0.82	0.82	0.82
CMU HIT 3	0.81	0.82	0.82	0.82	0.82	0.82

	Average Number of Activities						Accepted Rate (%)
	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	
CMU HIT 1	7.2	11	10.14	10.14	10.67	10.14	50
CMU HIT 2	6.9	10.2	10.2	10.2	10.2	10.2	50
CMU HIT 3	7.36	10.14	10.14	10.14	10.14	10.14	60

Figure 3.6 shows the detailed accuracy for 30 HITs in the Opportunity dataset. We summarize it in Figure 3.7. Figure 3.8 summarizes the average number of activities provided by qualified assignments in each type in the Opportunity HITs.

Without any checking in Type 1, many Opportunity HITs get very low accuracy, thus even we recruit turkers with the approval rate at least 90%, there are still many non-serious turkers in AMT working in our tasks. It can be explained since every workers who just sign up into AMT system get 100% approval rate, some of them certainly are spammers. With Type 2, the accuracy is generally improved, there is still some HITs get bad accuracy and low average number of activities, it means some turkers may guess section 1 in the interface is qualifica-

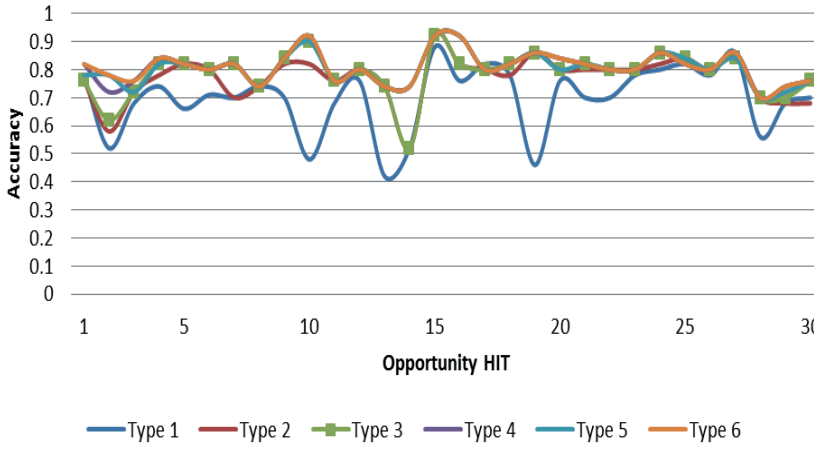

Figure 3.6: Detailed Accuracy of 30 AMT Opportunity HITs

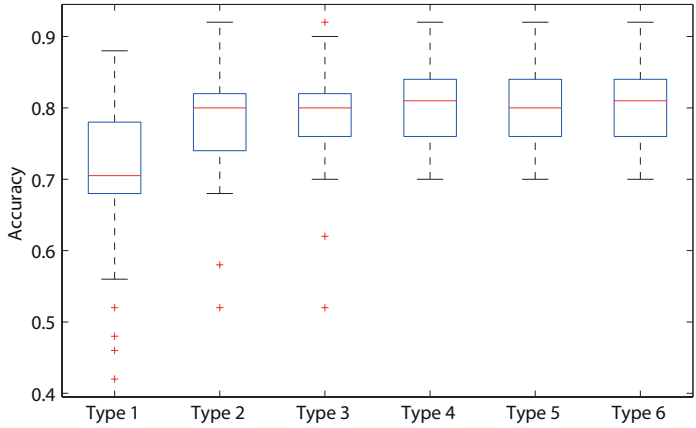

Figure 3.7: Box Plot of Accuracy of AMT Opportunity HITs

tion test and they tried to get good answers for section 1, but not for section 2. With Type 3, the accuracy is slightly better, however with Type 4, after we remove bad tags with the overlapping boundaries, we get very good and stable accuracies. The improvement from Type 3 to Type 4 can be intepreted that some spammers just tried to get a good

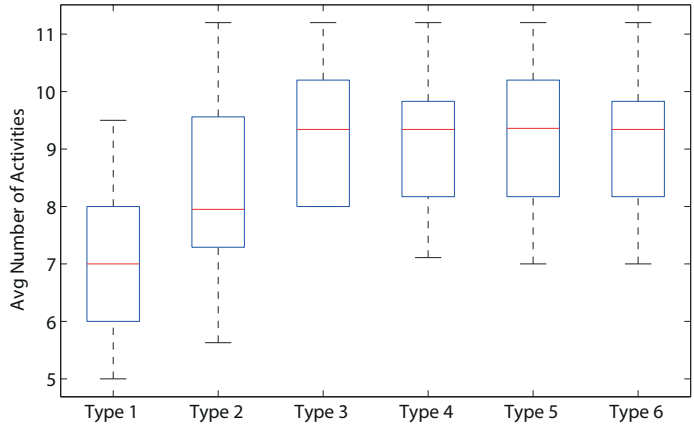

Figure 3.8: Box Plot of Average Number of Activities Tagged by Valid Turkers in Opportunity HITs

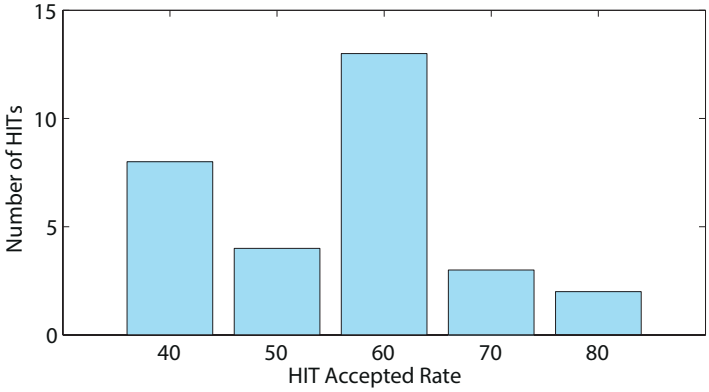

Figure 3.9: Accepted Rate in Opportunity HITs

result for section 1 that they think they can get their work approved and for section 2, they give non-sense or repeated activities which are overlapping many times. Type 4 and Type 5 have similar results. Therefore, the individual check with overlapping boundaries removal and the collaborative check to remove outliers are both good to detect and remove spammers. Type 6 is just a combination of Type 4 and Type

5. For Type 4, 5, 6, the accuracy is achieved from 76% to 92%. Most of the HITs are greater than 80% accuracy. The accuracy is increased by up to 40% compared to Type 1. For HITs that have the accuracy lower than 80%, we check the videos and turkers' answers and see that in the Opportunity videos, the door 2 position can not be seen clearly, so most of turkers can not tag open/close door 2 correctly. The toggle switch activity is very short and there is a lot of disagreement between turkers and experts the starting and ending time of the toggle switch activities. This disagreement is also unavoidable since there is usually no standard definition when the start and end of an activity should be (e.g., drink gesture can start from the time user picks up a cup or when user starts drinking).

The CMU HITs have similar results of accuracy. Type 4, 5, 6 improve the accuracy compared to Type 1. In the CMU dataset, we see that the disagreement between turkers and experts on the boundaries of "crack Egg" activities usually occurs.

Figure 3.9 shows the accepted rate in the Opportunity HITs. Only five Opportunity HITs have the accepted rate between 70%-80%. The rest (25 Opportunity HITs) have high rejected rate between 40-60%. In CMU HITs (see Table 3.6), the accepted rate is about 50-60% which is similar to that shown in the Opportunity HITs. Thus, there are about 40% non-serious turkers with 90% approval rate working in our Opportunity and CMU HITs.

Notably, our first strategy was to publish tasks with qualification test (section 1 in the interface) only. After getting the results from those experiments, we selected the good turkers who answered the test correctly and invited them to a second round where we asked them to work on the actual video annotation (section 2 in the interface). We tried with three HITs of the CMU dataset for this strategy, however, we realized that turkers will not come back to the site soon enough or that they do not want to work our round 2 tasks. Hence, we decided to include the qualification tests into each experiment. However, the results show that many turkers did section 1 very well but did not provide any good tags for section 2 which we need the most. Thus, it does not guarantee that turkers who do well the qualification test will work well on real tasks. Requesters should always have strategies to validate the answers from turkers.

It is interesting that in both Opportunity HITs and CMU HITs, serious turkers usually provide all occurrences of activities in the video even we just ask them to provide at least N tags to get accepted.

3.6 Conclusion and Future Work

In this paper, we conduct experiments to get activity annotation in videos from AMT in which turkers specify the temporal boundary and the label of all occurrences of activities in the videos. We introduce for the first time a methodology to use AMT to annotate activities occurring in video stream, with a data postprocessing stages to improve the annotation quality. The results show that even the HITs are distributed to high quality turkers in AMT (turkers' approval rate is at least 90%), quality of tagging still needs to be controlled carefully. The results also show that using verifiable questions (qualification checking) is not enough to detect good turkers since many non-serious turkers can try to work well only for the qualification checking part. In our work, we propose two filtering strategies to detect non-serious turkers. Both strategies work efficiently in our experiments and increase the accuracy by up to 40%. It takes a similar amount of time for turkers and experts to annotate activities in the videos. The results show that the annotation from AMT has comparable quality to the annotation by experts (76%-92%).

In conclusion, this work shows the feasibility to use crowdsourcing to annotate human activity in videos. However, this work also shows several future research directions:

- We would like to investigate the ability of AMT turkers to annotate activities in the video without the prior knowledge of what kind of activities can occur in the video. This allows for richer description of the activities, but requires more sophisticated filtering and combination strategies taking into account semantic ambiguities.

- Future work must investigate whether the quality of annotation obtained through crowdsourcing is sufficient for the purpose of training activity recognition systems.

- We would like to investigate strategies to stop experiments when we get enough answers from good turkers or extend it otherwise. Thus, the amount of money to pay turkers can be reduced

- Finally we would like to characterize the joint influence on the annotation quality of, e.g. the influence of the user interface, the complexity of the activities in videos, the quality of the videos,

the level of details in the requested annotations (primitive activities (e.g. take a cup) or high level activities (e.g. have breakfast) with/without object labeling). This requires comparative evaluations on more datasets of longer duration.

3.7 Acknowledgment

We would like to thank Zack Zhu (ETH Zurich), and Alberto Calatroni (ETH Zurich) for their assistance and providing the Opportunity videos. This work has been supported by the Swiss Hasler Foundation project Smart-DAYS.

4

WarpingLCSS for Activity Recognition

Long-Van Nguyen-Dinh, Daniel Roggen, Alberto Calatroni, Gerhard Tröster

Improving Online Gesture Recognition with Template Matching Methods in Accelerometer Data

Proceedings of the 12th International Conference on Intelligent Systems Design and Applications (ISDA 2012), pp. 831–836, Kochi, India, 2012.

DOI: 10.1109/ISDA.2012.6416645

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.*

4.1 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 [2]. 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. [2,7,8].

However DTW is shown in time series analysis to be sensitive to outlier noise [62]. 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. 4.9).

To improve online gesture recognition with template matching methods in the face of such noise, we propose two approaches derived Longest Common Subsequence (LCSS) [42]. 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.

4.2 Background and Related Work

Samples from body-worn sensors are categorized as time series data, therefore time series analysis methods are widely used for gesture recognition such as Hidden Markov Models [3–6] and TMMs using DTW [2,7,8].

Segmented DTW [7,8] 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 correspond-

ing 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 [2] 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 [63]. 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 [62]. 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).

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

LCSS and matching points between two sequences can be found efficiently using dynamic programming [42].

4.3 Training process

4.3.1 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 4.1 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.

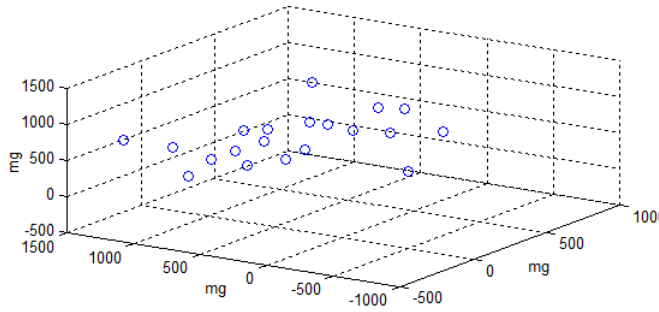

Figure 4.1: The distribution of cluster centroids of one user in Opportunity dataset, k = 20

4.3.2 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.

4.4 LCSS-based online spotting methods

The data processing flow to recognize gestures online is shown in Figure 4.2. 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.

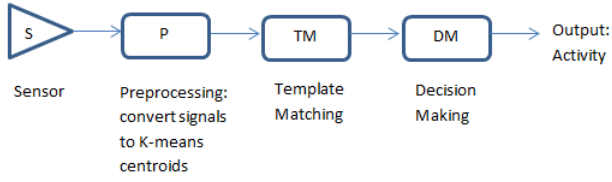

Figure 4.2: The data processing flow at spotting time

4.4.1 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 [7, 8], 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 4.3 illustrates the spotting process of Segmented LCSS.

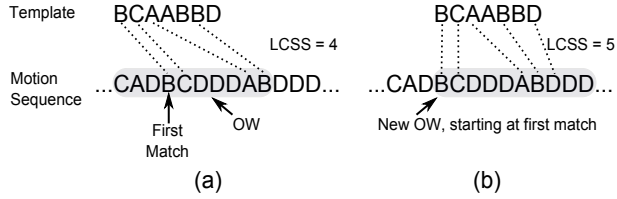

Figure 4.3: The Segmented LCSS spotting process. The shaded part represents the OW. The LCSS is computed between the gesture template and the OW to indicate whether an instance of the gesture class is spotted. The next OW will shift to the first matched point of the previous one.

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 $O(W * T) \approx O(T^2)$. Memory usage in Segmented LCSS is at most $O(T^2)$. 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 $O(T)$ [42]. Thus, the Segmented LCSS requires a constant time and space to process new sample.

4.4.2 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:

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

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 4.3.1. 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 4.4 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 4.5 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.

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 $O(T)$. It is faster than the Segmented LCSS by one order of magnitude. The WarpingLCSS maintains at most $O(T^2)$ memory for the need to trace back the starting boundary of detected gesture.

4.4.3 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

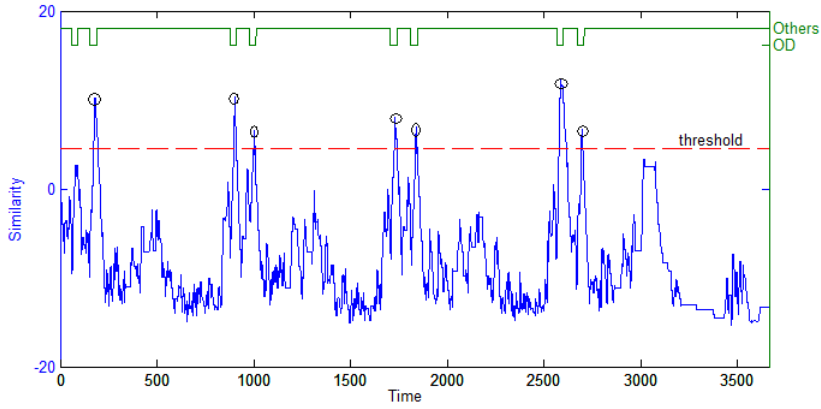

Figure 4.4: WarpingLCSS between an Open Door template and a streaming string, p = 3. The values of similarity W is updated continuously when new samples come. The line on the top shows the ground truth. The small circles show gesture detection at spotting time.

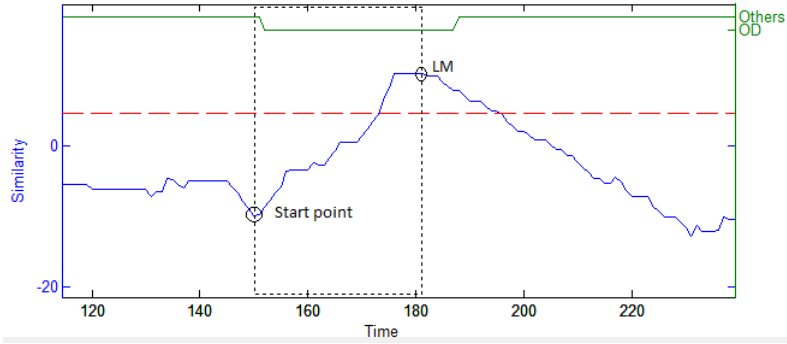

Figure 4.5: The close-up of the first detected OpenDoor in the streaming string

chosen as the best match. This process is the same for both Segmented LCSS and WarpingLCSS.

4.5 Dataset

We evaluate the methods on two datasets. The first is the Opportunity dataset [12] 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 17 gesture classes as shown in Table 4.1. 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. 4.6 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 4.1 are geometric shapes executed with the arm in the vertical plane. Fig. 4.7 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).

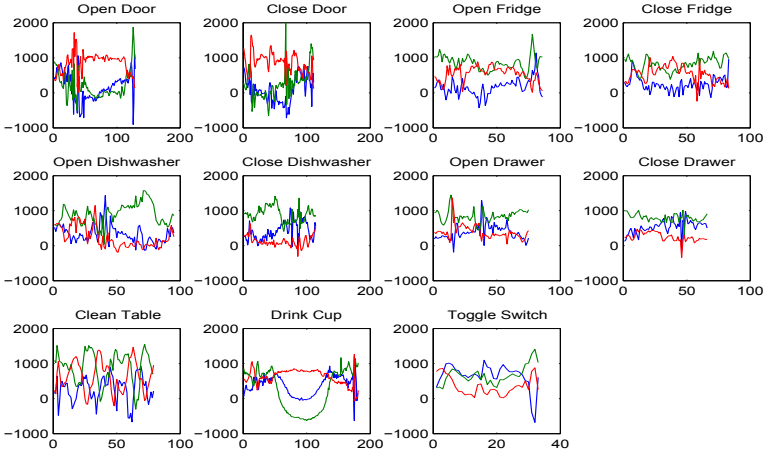

Figure 4.6: 3D Acceleration of the dominant upper arm of one user while executing some of gestures in the Opportunity dataset. Sampling rate is 30Hz. Units: samples on the abscissa and milli-g on the ordinate.

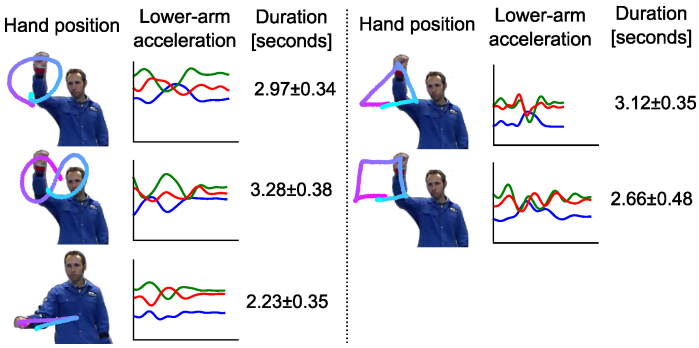

Figure 4.7: 3D Acceleration of the dominant lower-arm of one user while executing gestures in the HCI gesture dataset. Sampling rate is 30Hz. Units: samples on the abscissa and milli-g on the ordinate.

Table 4.1: 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 Door1 (OD1)
close Door1 (CD1)	open Door2 (OD2)	close Door2 (CD2)
open Fridge (OF)	close Fridge (CF)	drink Cup (D)
open Dishwasher (ODi)	close Dishwasher (CDi)	Toggle Switch (TS)

4.6 Experiments and Results

For each subject, we perform 5-fold cross validation. We compare our proposed LCSS methods to the Segmented DTW [7] and the Nonsegmented DTW [2]. All methods use the same strategy to select templates (the best similarity average) and the rejection threshold as discussed in section 4.3.2. 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.

4.6.1 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_NoNull 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.

4.6.2 Results

4.6.2.1 Gesture template similarity

Fig. 4.8 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 DTW is sensitive to boundary noise and variation in gesture execution. A representative case is illustrated in Fig. 4.9 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.

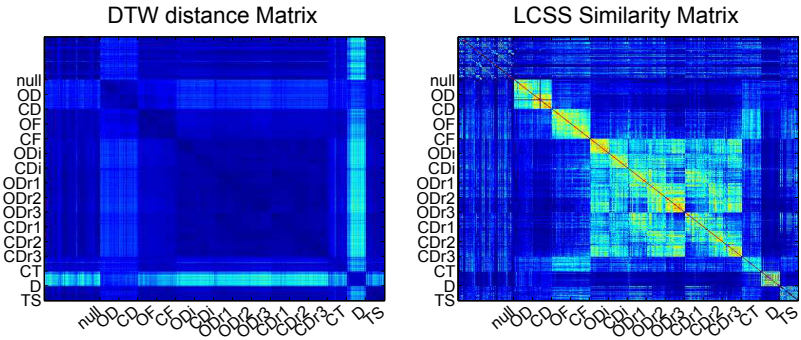

Figure 4.8: DTW distance matrix and LCSS similarity matrix of gestures of one user in the Opportunity training dataset. The LCSS metric allows to better distinguish the gestures of different classes.

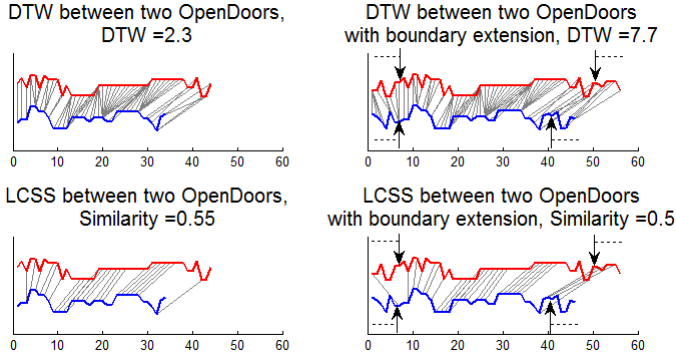

Figure 4.9: The sensitivity of DTW and LCSS to boundary noise. Two string instances of OpenDoor activity are shifted apart to show the patterns and the warping path (DTW distance) or the matching points (LCSS) between them. On the right, both instances are extended by 6 synthetic samples on each side. The arrows point to the positions from which the boundaries are extended.

4.6.2.2 Performance results

Table 4.2 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 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 4.2: The average accuracy and F1 measure over sample unit in two datasets

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

4.7 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

[1] We conducted a 2-sided hypothesis test at the 0.01 level of significance as in [64], the tests showed that the accuracy differences among the four methods are statistically significant.

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.

4.8 Acknowledgment

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.

5

Robustness of WarpingLCSS on Noisy Crowdsourced Annotations

Long-Van Nguyen-Dinh, Alberto Calatroni and Gerhard Tröster

Robust Online Gesture Recognition with Crowdsourced Annotations

Journal of Machine Learning Research, Volume 15, pp. 3187-3220, 2014.

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.

5.1 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 [54] and video [65, 66].

Training a gesture recognition system requires an annotated training data set that is used to perform supervised learning [9, 54–56]. 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 [12]. Moreover, it is also costly to hire experts to annotate data corpora.

Crowdsourcing has been emerged recently to address these issues [17,67]. 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 [68]. However, labels obtained from crowdsourcing are provided by low-commitment anonymous workers, thus they are commonly unreliable and noisy [28]. 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 [18, 68]. 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).

5.1.1 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 [69] 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 5.2, we first review existing work in online gesture recognition and crowdsourcing. In Section 5.3, we discuss crowdsourcing in gesture recognition and propose a taxonomy of annotation noise in gesture labeling by

crowdsourcing. Then, in Section 5.4, we present our proposed SegmentedLCSS and WarpingLCSS methods. The experiments are described in Section 5.5. We present quantitative results evaluating the robustness of our proposed methods against the baselines in Section 5.6. Finally, Section 5.7 concludes our work and gives some potential research directions.

5.2 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.

5.2.1 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 [12], subject self-report of activities at the end of the day [13]. *Online annotation* (i.e., real-time) techniques perform the annotation during execution of the activities, like experience sampling [14] 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 [15]. There is a trade-off between accuracy of an annotation technique and the amount of time required for annotation [16]. For example, offline annotation on video recordings by experts can provide accurate annotations, however it is extremely time consuming [12], 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 [12] or provide a course to teach subjects carefully how they should record and annotate their data correctly [54].

5.2.2 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 [18]. Crowdsourcing tasks are performed by low-commitment anonymous workers, thus acquired data is commonly unreliable and noisy [28]. 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 [18]. 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 [18,39]. Another way to ensure quality is to check the agreement in annotations among workers to detect non-serious workers [68]. [58] 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, [59] proposed Bayesian versions of worker quality inference. [60] 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 [15,68,70–72]. 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`.

5.2.3 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) [3–6], template matching methods (TMM) using mostly dynamic time warping—in short DTW [2,7,8] and support vector machines [9–11].

HMMs are not appealing since a large amount of training data is required to get results comparable to other TMMs and SVM. In [73] 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 [74], 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 [75], 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 [3,76,77].

Segmented DTW [7,8] 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 [2] 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* [63]. In previous work [69], we introduced two variations of LCSS-based template matching for on-

line 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 [69]). 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 [78,79]. 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 [79] 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 [80, 81], 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 [75–77,82]). 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 [83] 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.

5.2.4 Robustness against Annotation Noise

The impact of noise in annotations on the performance of classifiers has been investigated in the literature [16, 29–32]. 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.

5.3 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 [68, 72].

2. [84] 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.

5.3.1 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 5.1a illustrates the subclasses of *boundary jitter*.

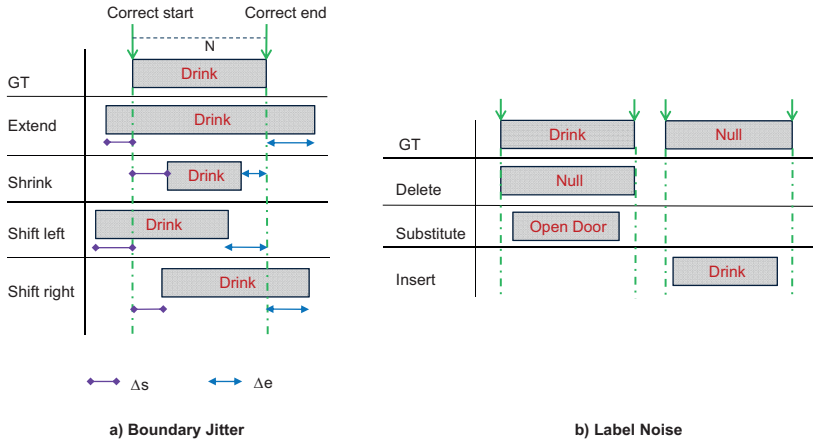

Figure 5.1: Illustrations of boundary jitter and label noise in crowdsourcing annotation. GT stands for ground truth. The blue dash-dotted lines indicate the correct boundary of a gesture.

- *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 5.1b 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 [85]. However, in this work, we consider those errors existing in annotations of training data set.

5.3.2 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 5.1a 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:

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

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

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

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

5.3.3 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 [68]. In the crowdsourcing experiment we used video footage belonging to the Opportunity data set [12], 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 [68]. 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 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 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 5.2. 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 [68].

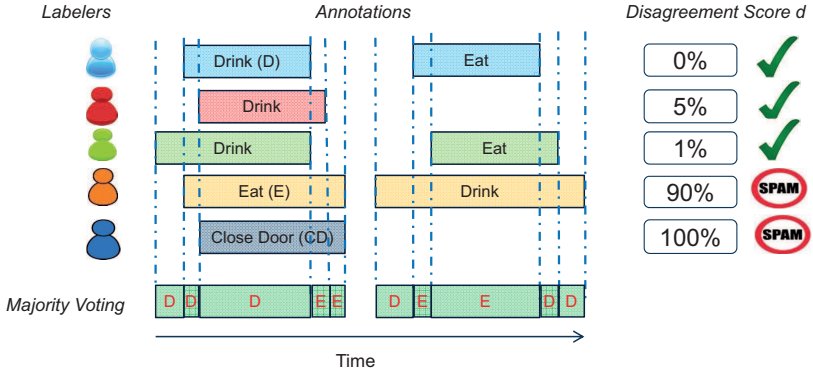

Figure 5.2: An illustration of the collaborative filtering technique to calculate the disagreement score of each labeler against the majority. The last two labelers are spammers and then their annotations will be removed.

Each video footage of the Opportunity data set was already examined and annotated carefully by one expert [12] 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 5.3.1, is shown in Figure 5.3a. 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 5.3b, 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%).

It can be seen that requesting multiple labelers for an annotation

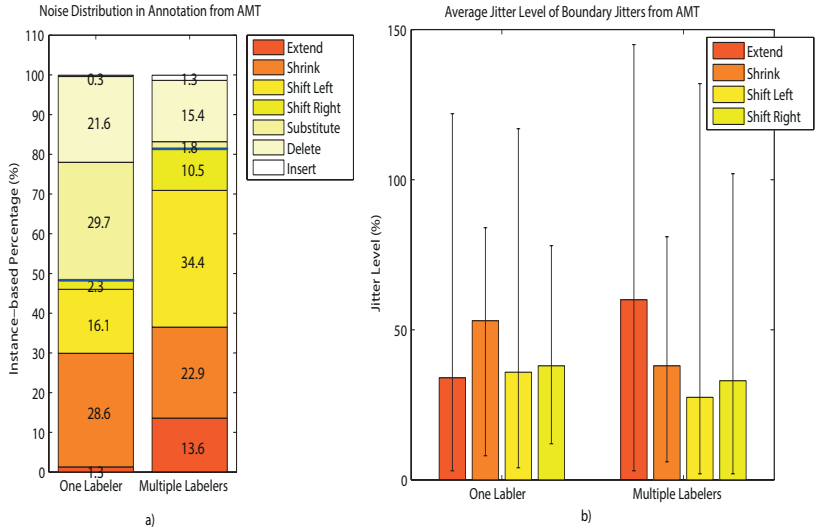

Figure 5.3: Analysis of crowdsourcing annotation from AMT. Blue lines in the figure a separate boundary jitter part and label noise part. Black lines in the figure b show the minimum and maximum level of jitter in each type of noise.

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.

5.4 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 5.4 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 [69] 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 5.4.

5.4.1 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

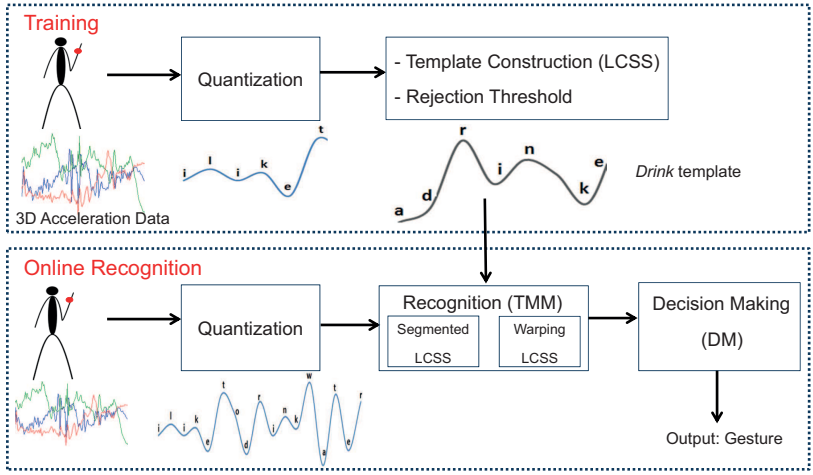

Figure 5.4: Data processing flow of the proposed LCSS-based template matching methods for gesture recognition.

complete strings is then denoted as $L_{(A,B)}$ or, when the strings are clear from the context, just with L.

$$L_{(A,B)}(i,j) = \begin{cases} 0 & \text{, if } i = 0 \text{ or } j = 0 \\ L_{(A,B)}(i-1, j-1) + 1 & \text{, if } s_A(i) = s_B(j) \\ \max \begin{cases} L_{(A,B)}(i-1, j) \\ L_{(A,B)}(i, j-1) \end{cases} & \text{, otherwise.} \end{cases} \tag{5.1}$$

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 $\Omega_A = \omega_A^{(0)} \dots \omega_A^{(L-1)}$ and $\Omega_B = \omega_B^{(0)} \dots \omega_B^{(L-1)}$ 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 [42]).

5.4.2 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, $\zeta_0 \dots \zeta_{k-1}$, 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:

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

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 $[0, 1]$.

$$d(\alpha_i, \alpha_j) = \frac{\|\zeta_i - \zeta_j\|_2}{max_{i,j}\|\zeta_i - \zeta_j\|_2} \quad . \tag{5.2}$$

5.4.3 Training Phase: Template Construction

For each labeled gesture in the training data set, a corresponding string is derived used the quantization described in Section 5.4.2. 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.

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

5.4.4 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.

5.4.5 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.

5.4.6 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 5.4.5. 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 $\vec{s}^{(c)}$. A substring of s in OW_c is denoted as s^c_{OW}. Each substring is compared to the template $\vec{s}^{(c)}$ for class c.

The LCSS algorithm is used to calculate $L_{(s^c_{OW}, \vec{s}^{(c)})}$ 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 $L_{(s^c_{OW}, \vec{s}^{(c)})} \geq \epsilon_c$, 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 5.5 illustrates the SegmentedLCSS.

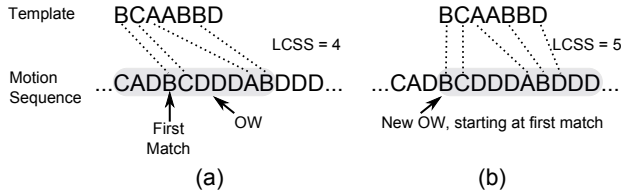

Figure 5.5: The SegmentedLCSS recognition process. The shaded part represents the observation window OW_c. For each class c, the LCSS is computed between the gesture template $\bar{s}^{(c)}$ and the quantized signal in the window. If the LCSS exceeds the rejection threshold, the samples between the first 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.

5.4.6.1 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 $O(T_c^2)$. The overall time complexity is then $O(C * \overline{T}^2)$, 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 $O(T^2)$, where T is the length of the longest template.

5.4.7 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 5.1 as follows.

$$W_{(\bar{s}^{(c)},s)}(i,j) = \begin{cases} 0 & \text{, if } i = 0 \text{ or } j = 0 \\ W_{(\bar{s}^{(c)},s)}(i-1,j-1) + 1 & \text{, if } \bar{s}^{(c)}(i) = s(j) \\ \max \begin{cases} W_{(\bar{s}^{(c)},s)}(i-1,j-1) - p * d(\bar{s}^{(c)}(i), s(j)) \\ W_{(\bar{s}^{(c)},s)}(i-1,j) - p * d(\bar{s}^{(c)}(i), \bar{s}^{(c)}(i-1)) \\ W_{(\bar{s}^{(c)},s)}(i,j-1) - p * d(s(j), s(j-1)) \end{cases} & \text{, otherwise,} \end{cases} \tag{5.3}$$

where p is a penalty parameter of the dissimilarity and $d(\cdot,\cdot)$ is the distance between two symbols as defined in Equation 5.2. 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 **??**. 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 5.6 illustrates an example of behavior of W. Figure 5.7 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.

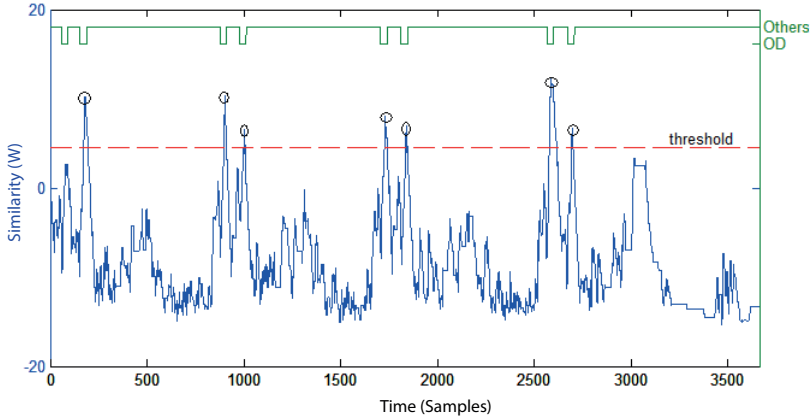

Figure 5.6: WarpingLCSS between a template of the gesture "open door" (OD) and a streaming string s, p = 3. The value W is updated for each new sample. The line on the top shows the ground truth. The small circles show gesture detection at spotting time.

5.4.7.1 Computational Complexity of WarpingLCSS

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

5.4.8 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 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

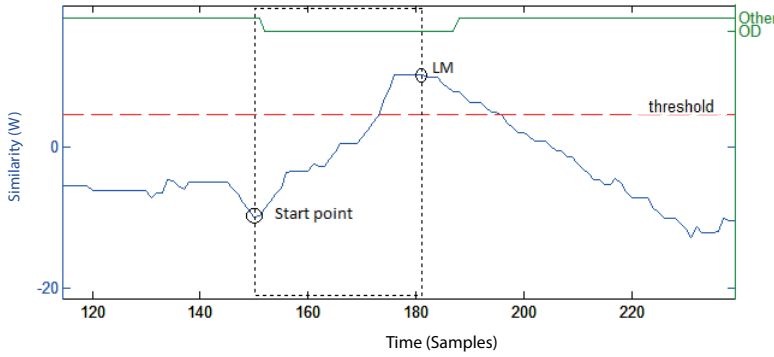

Figure 5.7: Close-up of the first detected "open door" gesture (OD) in the string s (see Figure 5.6). The local maximum (LM) marks the end of the gesture, while the start is traced back through the matching symbols.

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.

5.5 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.

5.5.1 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 5.1. 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].

Table 5.1: Gestures in Opportunity, Skoda, and HCI data sets.

HCI Gestures				
Circle	Triangle	Square	Infinity	Slider
Their Speculars				Null

Opportunity Gestures		
Null	clean Table (CT)	open Drawer 1-2-3 (ODr1-2-3)
close Drawer 1-2-3 (CDr1-2-3)	open Door 1-2 (OD1-2)	close Door 1-2 (CD1-2)
open Fridge (OF)	close Fridge (CF)	drink Cup (D)
open Dishwasher (ODi)	close Dishwasher (CDi)	Toggle Switch (TS)

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

5.5.1.1 Skoda

The Skoda data set [43] 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.

5.5.1.2 HCI

The HCI data set [44] contains 10 gestures executed by a single person. The gestures are geometric shapes executed with the arm in the vertical

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

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.

5.5.1.3 Opportunity

The Opportunity data set [12] is a rich multi-modal data set collected in a naturalistic environment akin to an apartment, where users execute 17 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 make the recognition more challenging.

5.5.2 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 5.3.1. In order to evaluate the effect of the different types of noise, we run simulations for each type of noise separately.

5.5.2.1 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.

5.5.2.2 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 5.3.2. 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 5.6.2.

5.5.3 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 [7, 8], Nonsegmented DTW [2] 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 5.4.2. The rejection thresholds are selected as discussed in Section 5.4.4. 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 [86] for training SVM.

5.5.3.1 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 Segment-

edLCSS, 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 [7,8]. Therefore, when one new sample arrives, the complexity of the SegmentedDTW is $O(T^3)$ 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.

5.5.4 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.

5.6 Results and Discussion

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

5.6.1 Results on Synthesized Crowdsourced Annotations

We first present the results with synthesized crowdsourced annotations, sweeping the noise levels as described in Section 5.5. 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.

5.6.1.1 Label Noise Simulation

Figure 5.8 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.

By analyzing the starting points of the curves of Figure 5.8, 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.

5.6.1.2 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

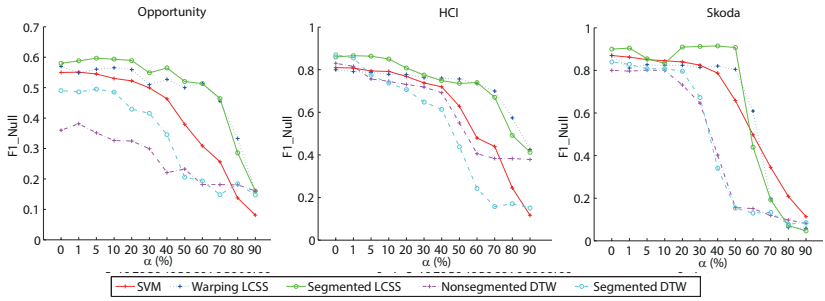

Figure 5.8: Performance of label noise simulation for the three data sets.

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 5.9. 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.

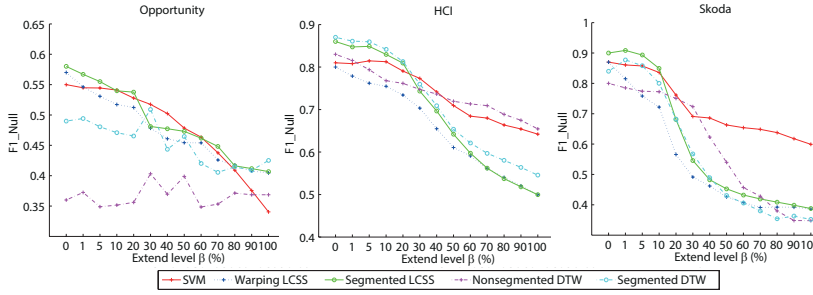

Figure 5.9: Performance of extend jitter simulation.

5.6.1.3 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 5.10.

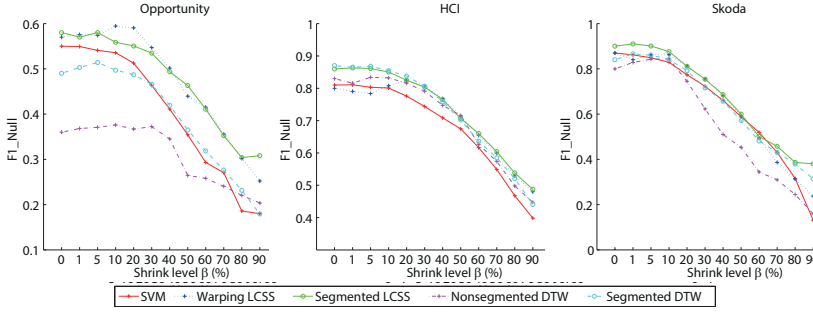

Figure 5.10: Performance of shrink jitter simulation.

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 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 5.3b). In that case, DTW will accumulate higher distances due to data misalignment at the beginning and the end of instances (see [69] for a more thorough discussion of the weakness of DTW with misalignment in temporal boundaries).

5.6.1.4 Shift-Left and Shift-Right Jitter Simulation

When annotations are shifted, a mixture of the effects described in Sections 5.6.1.2 and 5.6.1.3 are present. Some samples belonging to gestures are lost and some null class samples are labeled as belonging to a gesture. Figure 5.11 shows the results of *shift-right* jitter simulations (the *shift-left* simulations yield similar results). All methods can sustain

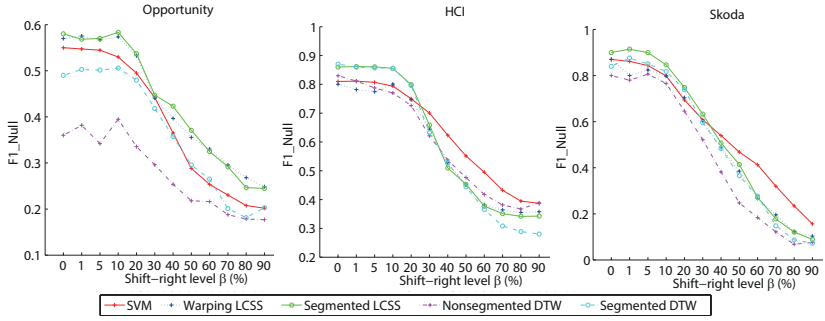

Figure 5.11: Performance of shift-right jitter simulation.

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*.

5.6.2 Results on Real Crowdsourced Annotation

To further validate the outcome of the previous experiments, we use the real crowdsourced annotations discussed in Section 5.3.3. 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 5.3.1) are present and jitter levels are varied from one instance to another instance (see Figure 5.3).

Figure 5.12 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 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 [69] 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 5.3). 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 5.6.1.1, 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 5.6.1.1.

Additionally, we conduct a 2-sided hypothesis test at the 0.01 level of significance as in [64] 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.

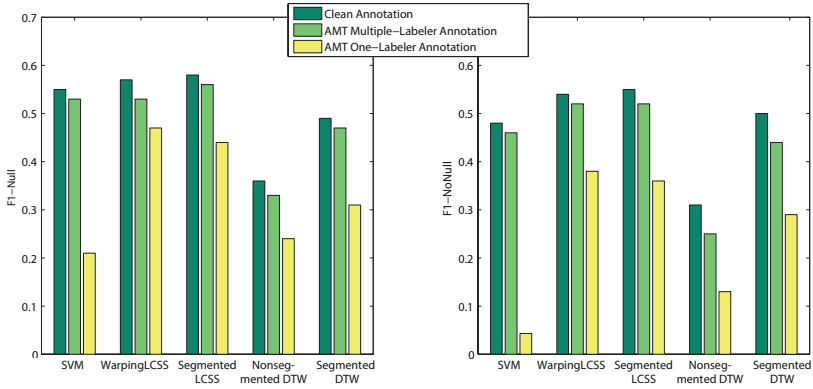

Figure 5.12: Performance of real crowdsourcing annotation on Opportunity data set.

5.6.3 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 Warp-

ingLCSS 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 5.8) 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 5.13 for the real crowdsourced annotation and in Figure 5.14 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 5.3). 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.

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

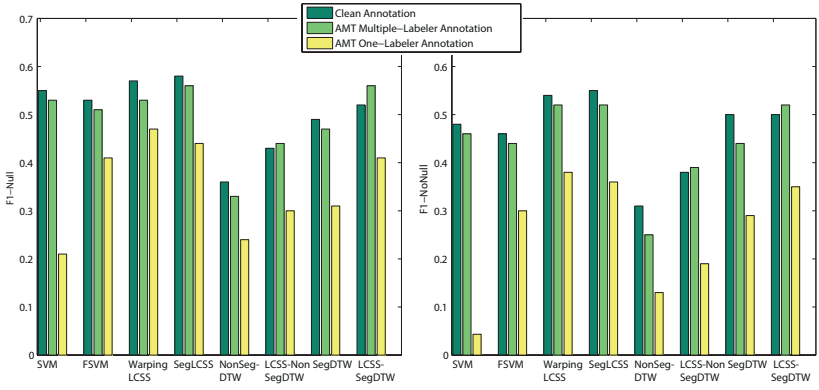

Figure 5.13: Performance of real crowdsourcing annotation on Opportunity data set for the methods with and without filtering. SegLCSS, NonSegDTW, and SegDTW stand for Segmented LCSS, Nonsegmented DTW and Segmented DTW respectively.

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 [69].

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.

5.6.4 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

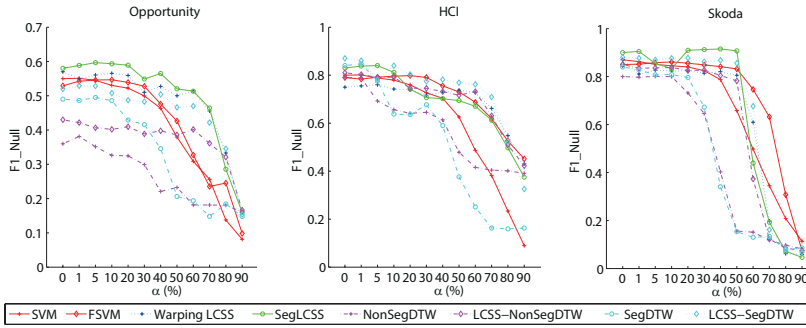

Figure 5.14: Performance of label noise simulation for the methods with and without filtering.

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 [87].

5.7 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.

5.8 Acknowledgment

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 Smart-DAYS.

6

WarpingLCSS on Multimodality

Long-Van Nguyen-Dinh, Alberto Calatroni, Gerhard Tröster

Towards a Unified System for Multimodal Activity Spotting: Challenges and a Proposal

Proceedings of the International Conference on Ubiquitous Computing (UbiComp '14 Adjunct), pp. 807–816, Seattle, WA, USA, 2014.

DOI: 10.1145/2638728.2641301

Abstract

In the existing multimodal systems for activity recognition, there is no single method to process different sensor modalities at different on-body positions. Moreover, sensor types are often selected and optimized so as to accord with the goal of application. The complexity makes those systems infeasible to be deployed for new settings. This paper proposes a unified system which works with any available wearable sensors placed on user's body to spot activities. Each data stream is treated uniformly through our proposed template matching WarpingLCSS to spot activities. With the uniformity in extracting activity-specific patterns from raw sensor signals, our proposed system is compatible with respect to modalities and body-worn positions.

We evaluate our system on the Opportunity dataset of four subjects consisting of 17 hard-to-classify classes (e.g., open/close drawers at different heights) with 17 sensors belonging to three modalities (accelerometer, gyroscope and magnetic field) attached at different on-body positions. The system achieves good performances (63% to 84% in F1 score). Moreover, the robustness and efficiency to addition and removal of sensors as well as activity classes are also investigated.

6.1 Introduction

Continuous activity recognition (*activity spotting*) is a core component in context-aware systems. It enables a variety of applications such as ambient assisted living, human computer interaction. In activity spotting, actions of interest and their temporal boundaries are detected in a continuous data stream in which they are randomly mixed with arbitrary non-interest actions (*null* class).

In the past few years, promising results from body-worn sensors for activity spotting have been presented [2,54]. Many modalities, such as motion-related ones (acceleration, rate of turn, magnetic field) [54,88], temperature [89] or sound [70] have been explored as inputs to activity recognition systems. Nevertheless, with the increasing availability of commercial wearable sensor devices (such as smartphones, watches, glasses and, in a near future, sensor-equipped garments), the multimodal aspect is ready for being fully exploited.

Why are there no multimodal activity spotting systems readily deployed in commercial applications? One challenge is that recognition chains for different sensor modalities need still quite some

hand-crafting for being deployed. Features and classifiers used for accelerometer data differ substantially from the ones needed for gyroscope or compass data, or audio. Even for the same modality, the needed features can differ for sensors mounted on different parts of the body. Furthermore, since each modality allows to recognize some restricted sets of activities, system designers still tend to solve specific activity recognition problems with specific sensors.

Other challenges of a unified multimodal framework that make it easily deployed in any settings are the following:

- If new sensors are worn by the user (e.g., the user wears a new smart watch), these should be integrated into the system smoothly, without asking the user where on the body the devices have been mounted or which classifier should be used for new data from those sensors.

- The system should handle missing sensors in run-time (e.g., the user gets off his sensor-equipped shoes) without interfering with other sensors.

- The system should also adapt new activity classes of interest smoothly without retraining the whole system.

In this paper we make one step towards a unified framework for multimodal activity recognition which attempts to overcome the challenges addressed above. Our system treats different modalities, and sensors with the same modalities at different on-body placements in a homogeneous way. Specifically, each data stream is first quantized into strings of symbols by using k-means (here serving as a vector quantization step) and then substring matching is performed by the fast and efficient template matching method WarpingLCSS [69] to spot activities. Our system recognizes activities by extracting activity-specific patterns from raw sensor signals, hence it is agnostic with respect to modalities and on-body positions. Thus, it can combine any available wearable sensors placed on user's body to spot activities. Besides that, our system takes the benefit of template matching methods in which different classes can be trained and spotted separately so that it can handle new activity classes easily.

We investigate two multimodal frameworks to fuse different data sources either at the signal level (*signal fusion*) or at a decision level (*classifier fusion*). In the classifier fusion framework, a novel fusion technique for template matching is proposed to combine all spotting

results from different sensors. The two frameworks will be compared throughout the paper in terms of recognition performance, speed, ease to add or remove sensors, and ease to add or remove activity classes.

In this work, we test the proposed frameworks with the recognition of hand actions (i.e., gestures), but the frameworks apply with no loss of generality to other activities. The proposed system is evaluated with the complex Opportunity dataset [12], which includes 17 hand actions using 17sensors belonging to three modalities (accelerometer, gyroscope and magnetic field) at different on-body positions. The performances of all subsets of sensors in the *classifier fusion* framework are also given to demonstrate its flexibility to sensor addition and removal.

6.2 Related Work

Various techniques for online gesture recognition can be found in literature; they include Hidden Markov Models (HMM) [90], support vector machines (SVMs) [91], template matching methods (TMM) using dynamic time warping (DTW) [2] or using longest common subsequence (LCSS) [69].

With recent advances in the development of inexpensive wearable sensors, researchers have investigated activity recognition systems using multimodal sensors or multiple single-modal sensors to improve the performance. In [54], five accelerometers were attached at different on-body positions to recognize physical activities. [92] used motion sensors and force sensing resistors to recognize hand actions for quality inspection in car production. The fusion of multiple data sources can be performed either early at signal level, feature level [89], or late at decision level (i.e., classifier fusion) [54, 92].

For each modality, a wide range of features and supervised learning techniques for activity recognition has been explored [93]. As one example, accelerometer data can be classified with Naive Bayes [54], SVMs [91], C4.5 decision trees [54], HMM [90], TMM [69] and a variety of features in both time and frequency domains can be extracted [93]. Due to the diversity of methods and features, the existing multimodal systems selected different methods and features for different modalities or sensors mounted at different on-body positions [2, 92]. For example, in the application of car quality inspection [92], inertial sensors must be attached at arms and torso in order to acquire the trajectories of wrists and elbows, force sensing resistors attached at lower arms to monitor muscle. Four different methods including K-Nearest-

Neighbor (KNN), TMM, k-means classifier, and Bayes classification were used in that work. They also trained different classifiers linked to the different modalities with different set of labels for the best possible performance purpose.

WarpingLCSS was first presented in our previous work [69] as a fast and efficient method to spot gestures using one 3D accelerometer on arm. The method showed robustness against noisy annotations and high variances in activity execution. In the work by Chen et al. [94], they investigated WarpingLCSS with multi-sensor fusion combining 6 different accelerometers at wrists and arms. However, the performance was not improved. According to the best of our knowledge, there is no previous work that investigates the use of WarpingLCSS towards a unified multimodal system, especially with other modalities such as gyroscopes, magnetic sensors.

6.3 Multi-modality System

In our system, we use the recently proposed template matching WarpingLCSS [69] as a core module for data processing, training and activity spotting. Data is recorded continuously and synchronously from multiple sensors. The training data is manually labeled with a list of activity classes of interest. Activities which are not in the list are considered as *null* class.

We propose two frameworks for fusing multimodal sensors at two different processing levels: *classifier fusion* and *signal fusion*. In the *classifier fusion* framework, signal data from each sensor are processed separately by a template matching (TM) module and then the spotting outputs from all sensors will be fused to have a final recognition output. Different TM sessions can be run in parallel. In contrast, in the signal fusion framework, signals from all sensor modalities are combined into one data stream before being fed into the TM module. Figures 6.2 and 6.4 gives an overview of the two frameworks.

6.3.1 Template Matching

The TM module processes input data, generates templates for activity classes in the training phase and recognizes activities in the spotting phase. An overview of the TM module is shown in Figure 6.1.

First, the TM module applies k-means to all training data points and quantizes the signal input to their closest cluster centroids. Thus,

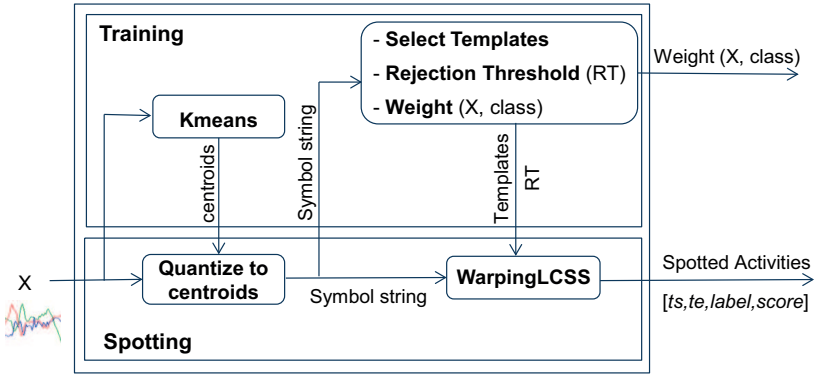

Figure 6.1: Template Matching Module. X is data input from one or multiple combined sensors.

signal data are then represented as a string of symbols (i.e., the indices of the centroids). The number of symbols k depends on the variation of the input signal. Accordingly, the cluster centroids are representative points which can capture body movements at sensor-attached positions regarding to specific activities.

In the training process, one or more templates are created for each activity of interest to represent the typical patterns for that class. The templates are chosen as instances that have the highest average longest common subsequence (LCSS) scores [42] to all other instances of the same class. Additionally, a rejection threshold needs to be calculated for each activity class in the training phase to be able to reject signals not belonging to that class upon recognition. 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: $\mu^{(c)} - h^{(c)} * \sigma^{(c)}$, with $h^{(c)} = 0,1,2,....$ The value of $h^{(c)}$ is determined by testing the recognition of class c on the training data and selected as the one which yields the best F1 score performance. Specifically,

$$F1_c = 2 * \frac{precision_c * recall_c}{precision_c + recall_c}, \tag{6.1}$$

where $precision_c$ 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. Note that the value of $F1_c$ can also be used to indicate how well sensor data fed into the TM module can recognize the specific class c.

When spotting, the same process of quantization is applied to the streaming sensor data, with the cluster centroids identified during training. Then, for each activity class, the WarpingLCSS method [69] is used to match the template within the online string to spot activities belonging to that class. Given the activity 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 as follows:

$$W_{(\bar{s}^{(c)},s)}(i,j) = \begin{cases} 0 & \text{, if } i = 0 \text{ or } j = 0 \\ W_{(\bar{s}^{(c)},s)}(i-1,j-1) + 1 & \text{, if } \bar{s}^{(c)}(i) = s(j) \\ \max \begin{cases} W_{(\bar{s}^{(c)},s)}(i-1,j-1) - p * d(\bar{s}^{(c)}(i-1), s(j-1)) \\ W_{(\bar{s}^{(c)},s)}(i-1,j) - p * d(\bar{s}^{(c)}(i), \bar{s}^{(c)}(i-1)) \\ W_{(\bar{s}^{(c)},s)}(i,j-1) - p * d(s(j), s(j-1)) \end{cases} & \text{, otherwise,} \end{cases} \tag{6.2}$$

where p is a penalty parameter of the dissimilarity and $d(\cdot,\cdot)$ is the normalized Euclidean distance between two symbols (i.e., two corresponding centroids) in a range [0,1] . When a new symbol arrives, the WarpingLCSS processes and updates the score immediately. Hence the computational cost of WarpingLCSS is low for online recognition. The WarpingLCSS score grows (i.e., symbols are matched) when an instance of the examined class is performed and drops significantly if other classes are performed due to the penalty terms. Additionally, the penalty is accumulated in a time-warping manner as in dynamic time warping (DTW) [95], hence WarpingLCSS penalizes the same consecutive symbols which are mismatched only once. An activity of class c is recognized for each local maximum of W that exceeds the rejection threshold.

Outputs of the TM module are spotted activities with a format *[start-time, end-time, label, simScore]* to indicate when the activity occurs and the similarity score (*simScore*) between the activity and the template of that class. In the TM module, the spotting of different activity classes can be processed concurrently in parallel. A more detailed explanation, complexity analysis and illustration of WarpingLCSS can be found in [69].

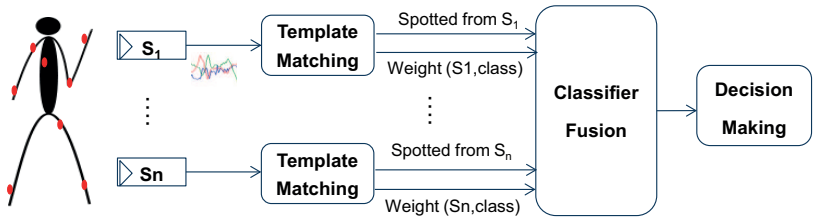

Figure 6.2: Classifier fusion framework

6.3.2 Classifier Fusion Framework

In the *classifier fusion* framework, each sensor is treated uniformly via the same process in the *Template Matching* module. The spotting outputs from all sensors are combined in the *Classifier Fusion* module as shown in Figure 6.2. Then the *Decision Making* (DM) module resolves conflicts for spotted instances belonging to multiple classes and output the results.

Let Φ and $|\Phi|$ be the set of sensors and the number of sensors in the system, respectively. We represent the spotting output from a sensor $\mathcal{S} \in \Phi$ in a spotting matrix $\mathcal{M}(\mathcal{S})$ of size $C * N$, with C is the number of activity classes of interest and N is the number of samples processed. $\mathcal{M}(\mathcal{S})(c, i)$ represents the entry at the ith sample and the row of class c in the matrix $\mathcal{M}(\mathcal{S})$. Each row c in the matrix, indicated as $\mathcal{M}(\mathcal{S})(c)$ stores the information of spotted instances of an activity class c from the sensor $\mathcal{S}$. Specifically, if the sensor outputs an activity instance of class c from *start-time* to *end-time* with a similarity score *simScore* (i.e., *[start-time, end-time, c, simScore]*), then $\mathcal{M}(\mathcal{S})(c, i)$ = *simScore* for all i-th samples in the interval from *start-time* to *end-time* at the row c. Figure 6.3 gives an example of the spotting matrix.

Classifier Fusion We propose Weighted Fusion method to fuse the spotting results from all sensors. Let Weight($\mathcal{S}$,c) be a prior weight to indicate how well sensor $\mathcal{S}$ can recognize the specific class c. We set Weight($\mathcal{S}$,c) as the best $F1_c$ performance (see Equation 6.1) when selecting the rejection threshold for activity class c in the processing of sensor s. The weighted summed spotting matrix is computed as

Activity class	samples 1	2	3	4	5	6	7	8	9	10	…
drink	0	0.8	0.8	0.8	0	0	0	0	0	0	…
open_door	0	0	0	0	0	0	0.6	0.6	0.6	0	…
close_door	0	0	0	0	0	0	0	0	0	0	…

Figure 6.3: An example of the spotting matrix with three activity classes (drink, open door and close door) and two spotted activities: [2, 4, drink, 0.8] and [7, 9, open_door, 0.6].

follows.

$$\bar{\mathcal{M}}(c) = \sum_{\substack{i=1 \\ \mathcal{S}_i \in \Phi}}^{|\Phi|} Weight(\mathcal{S}_i, c) * \mathcal{M}(\mathcal{S}_i)(c) \; \forall c. \tag{6.3}$$

The similarity score of an activity in the spotting matrix degrades if the prior performance of the sensor to recognize the corresponding activity class is low.

Given the fused spotting matrix $\bar{\mathcal{M}}$, for each spotted activity *[t1, t2, c, simScore]*, the similarity score *simScore* is updated as the average score in the interval from the time t1 to the time t2 at the row c in $\bar{\mathcal{M}}$. Specifically, the updated *simScore* is computed as follows.

$$\bar{simScore} = \frac{\sum_{i=t1}^{t2} \bar{\mathcal{M}}(c, i)}{(t2 - t1)\ [\text{samples}]}. \tag{6.4}$$

Consequently, the similarity score of an activity is boosted if more sensors predict that activity performed.

Decision Making If an activity is spotted as belonging to multiple classes (i.e., boundaries of spotted instances are overlapping), the DM module will resolve conflicts by deciding the class with highest similarity score as the best match. If an activity is classified into only one class, the DM will output the class. Otherwise, if no activity class is spotted, the DM will output *null*.

6.3.3 Signal Fusion Framework

In the *signal fusion* framework, the *Signal Fusion* module combines signals from all sensors into one data stream as shown in Figure 6.4.

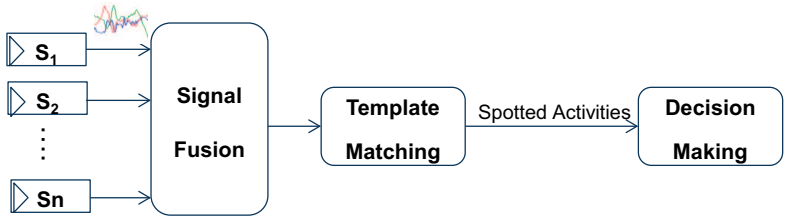

Figure 6.4: Signal fusion framework

Let d_i be the dimension of signal data generated from sensor $\mathcal{S}_i \in \Phi$. The combined data stream from the *Signal Fusion* module has a dimension of $\sum_{i=1,\mathcal{S}_i \in \Phi}^{|\Phi|} d_i$. The TM module then processes data and outputs spotted activities. Finally, the DM module handles spotting conflicts and outputs recognized activities as discussed above.

6.4 Experiments

We present the activity dataset, evaluation metrics and the conducted experiments to evaluate the proposed system in this section.

6.4.1 Dataset

We evaluate the system on the Opportunity dataset [1] [12] which is a rich multimodal multi-sensor dataset collected in a naturalistic environment akin to an apartment, where users execute 17 daily gestures. The dataset contains a large variability in the execution of the activities and *null* class is predominant (37%). We use the subset of recording corresponding to four subjects in which each subject wears 17 sensors belonging to three modalities (3D accelerometer, 3D gyroscope and 3D magnetic field) attached at different on-body positions. Each subject performs 20-40 repetitions of each gesture class. Totally, the dataset contains 1485 activity instances. Table 6.1 shows the list of activity classes in the Opportunity dataset. Note that there are three drawers located at different heights and two different doors in the dataset. Figure 6.5 shows locations of sensors on body (i.e., right upper arm

[1] http://www.opportunity-project.eu/challengeDownload

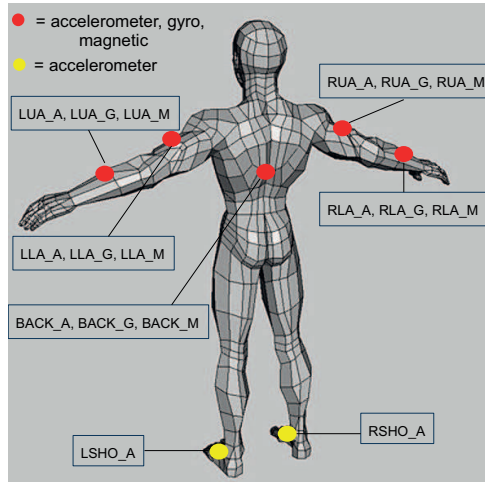

Figure 6.5: Sensors attached at different places on body and their modalities in Opportunity dataset.

Table 6.1: Activities in Opportunity dataset.

Null	clean Table (CT)	open Drawer 1-2-3 (ODr1-2-3)
close Drawer 1-2-3 (CDr1-2-3)	open Door 1-2 (OD1-2)	close Door 1-2 (CD1-2)
open Fridge (OF)	close Fridge (CF)	drink Cup (D)
open Dishwasher (ODi)	close Dishwasher (CDi)	Toggle Switch (TS)

(RUA), right lower arm (RLA), left upper arm (LUA), left lower arm (LLA), back (BACK), right shoe (RSHO) and left shoe (LSHO)) and their modalities. The signals of all sensors are recorded at a frequency of 30Hz.

6.4.2 Evaluation Metrics

Generally, activity classes may occur non-uniformly in real-life datasets. In the Opportunity dataset, *null* class is predominant (37%). Therefore, we use the weighted average sample-based F1 score to assess the performance of activity recognition. It is computed as the sum of the F1 scores of all classes, each weighted according to the proportion

of samples in that particular class. Specifically,

$$F1 = \sum_c w_c * F1_c,$$

where c is the class index, w_c is the proportion of samples of class c, and $F1_c$ is computed as in Equation 6.1.

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* or vice versa. F1-NoNull value represents how well the recognition system detects activity classes of interest. The recognition system that has high values of both F1-Null and F1-NoNull predicts well both activities and *null* class.

6.4.3 Experiments on Multimodal System

For each subject, we perform experiments in 5-fold cross validation. All raw signals (30Hz sampling rate) are down-sampled for a faster computation. Specifically, an average value of each sliding window of size 6 samples and overlap 3 is extracted to represent the corresponding set of data points in the window. In our experiments, the TM module generates only one template for each activity class.

In the *classifier fusion* framework, the number of symbols (i.e., number of clusters in k-means) is selected empirically k = 20 for each 3D sensor. Note that k can be selected by using cross-validation on the training data.

In the *signal fusion* framework, the *Signal Fusion* module combines all 17 sensors into a data stream with a high dimension of 51. Consequently, the number of symbols is selected much higher to capture variants in the combined movements at seven on-body positions (see Figure 6.5). We select empirically k = 200.

6.5 Results and Discussion

6.5.1 Performance of One Sensor

The performance of a sensor reflects how well that sensor recognizes the activities. Figure 6.6 shows the performance of each sensor (i.e., number of sensors = 1) and their combinations in the *classifier fusion*

Table 6.2: Performance of two frameworks on 17 sensors in the Opportunity dataset.

	Subject 1		Subject 2		Subject 3		Subject 4		Average	
Method	F1-Null	F1-NoNull	F1-Null	F1-NoNull	F1-Null	F1-NoNull	F1-Null	F1-NoNull	F1-Null	F1-NoNull
Classifier Fusion	0.74	0.79	0.63	0.67	0.75	0.80	0.65	0.71	0.69	0.74
Signal Fusion	0.77	0.77	0.67	0.68	0.84	0.81	0.74	0.73	0.76	0.75

framework. As seen in Figure 6.6, the performances of different sensors vary significantly. The sensors on shoes (LSHO_A and RSHO_A) give the worst performance since their signals are not distinguishable for different gesture executions (e.g., open doors and open drawers have the similar patterns of foot movements). The accelerometer at lower dominant arm (RLA_A) gives the best performance for subject 1. Meanwhile, the magnetic sensors at lower dominant arm give the best results for subjects 2-4.

6.5.2 Comparison between Two Frameworks

Table 6.2 shows the results on the use of all 17 sensors in the two proposed fusion frameworks. They both achieve a good performance for the four subjects (63% to 84% F1-Null). In average, the performance of the *classifier fusion* framework on 17 sensors increases by 16% F1-Null and 21% F1-NoNull compared with the average performance of one sensor. It also increases by 4% F1-Null and 15% F1-NoNull compared against the average performance of the best one-sensor.

The *signal fusion* outperforms the *classifier fusion* about 7% F1-Null and only 1% F1-NoNull in average. It means the *signal fusion* can detect the *null* class better than the *classifier fusion*. The rationale is that the *signal fusion* has a global view of data from all sensors at once before processing; meanwhile the *classifier fusion* framework has only a local view of data from each sensor. The hand actions of concern and the *null* class may have the similar foot movements (e.g., walking, standing). Hence, data from the shoe sensor may detect the activities when the *null* class actually occurs. Even the other sensors can detect the *null* instance, the classifier fusion still outputs the false detected activities. By contrast, the *signal fusion* framework outputs an activity only when the combined pattern of that activity from different sensors is matched.

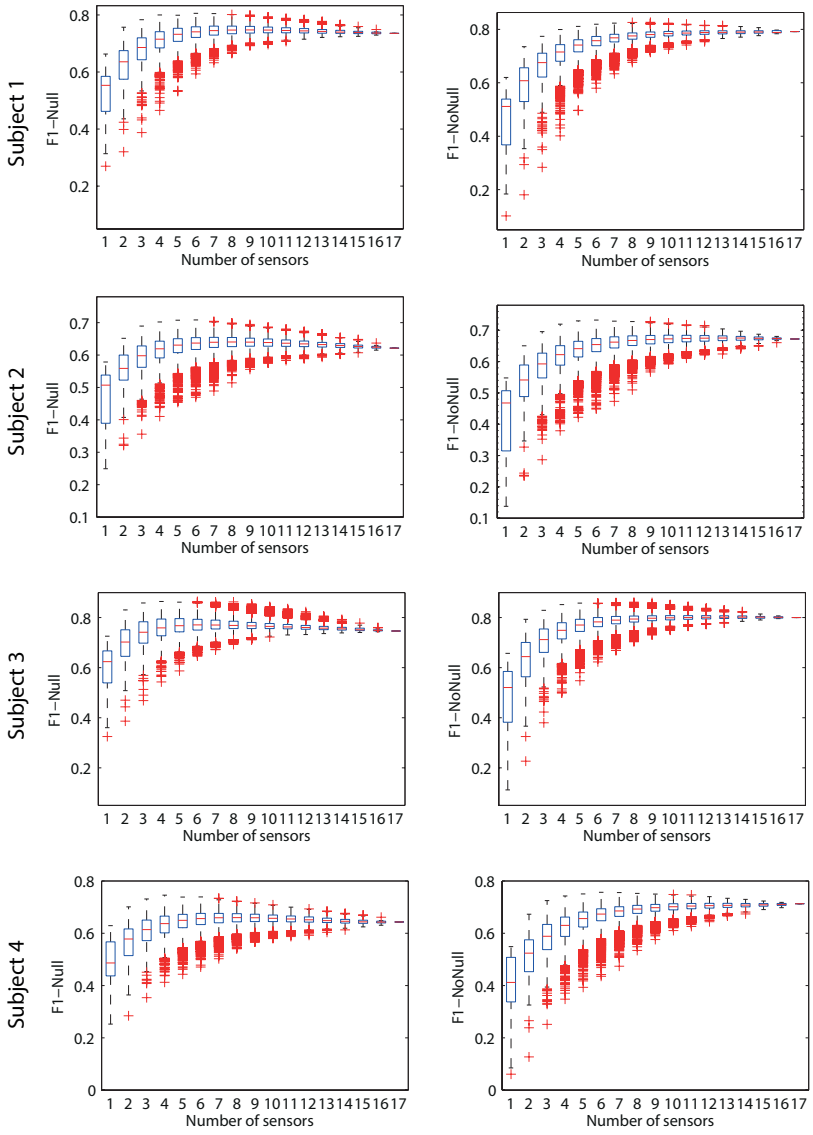

Figure 6.6: Performance of classifier fusion framework with all subset combinations of 17 sensors for subject 1. The red middle line shows the average performance.

Table 6.3: Comparison between classifier fusion framework and signal fusion framework.

	Classifier fusion	Signal fusion
Parallelism at sensor	* Yes	* Do not care
Parallelism at activity class	* Yes	* Yes
Sensor Addition or Removal	* Easy	* Hard
Class Addition or Removal	* Easy	* Easy

Besides the recognition accuracy, we compare the advantages and limitations of the two frameworks with regards to speed, ease to remove or add sensors to the system, and ease to add or remove activity classes. They are summarized Table 6.3.

The running time of our proposed system depends on how many sessions of the TM module run to spot activities. In the *signal fusion* framework, the number of TM session is only one. In the *classifier fusion* framework, it equivalents to the number of sensors deployed in the system. However, those TM sessions for different sensors can be executed in parallel. Therefore, if the parallelism is maximized, the running time of the two frameworks to spot activities is equivalent (i.e., time to run the TM module once).

Each sensor is processed separately in a uniform way in the *classifier fusion* framework. Therefore, it enables easy addition and removal of sensors without interfering with the use of other sensors to spot activities. Meanwhile, in the *signal fusion* framework, the combination of all sensor signals into one data stream before being processed will require the same sensor settings in the training and spotting phases so that the quantization step can proceed properly. Hence, adding or removing sensors in the *signal fusion* framework requires retraining the whole system.

The TM module spots each activity class separately. Therefore, the spotting for different activity classes can be executed independently in parallel. Hence, our proposed recognition system with the core *Template Matching* is very flexible in adding or removing activity classes.

6.5.3 Sensor Combinations in Classifier Fusion Framework

We show the performances of the classifier fusion framework on all subset combinations of 17 sensors in Figure 6.6. The results show that

Table 6.4: The combination of sensors giving the best performance in the classifier fusion framework.

	Sensors (Number of sensors)	F1-Null	F1-NoNull
Subject 1	BACK_M, RUA_G, RLA_G, RLA_M, LUA_A, LLA_A (6)	0.82	0.83
Subject 2	BACK_G, BACK_M, RUA_G, RUA_M, RLA_G, LUA_A (6)	0.71	0.73
Subject 3	BACK_G, RUA_M, RLA_G, RLA_M (4)	0.87	0.85
Subject 4	BACK_M, RUA_G, RLA_A, RLA_M (4)	0.75	0.74

the performances among groups of sensors differ less when the number of sensors increase. This indicates that adding a better performance sensor into a group increases the average performance. The average performance increases significantly in both F1-Null and F1-NoNull as the number of sensors increases from 1 to 6 and then keeps stable. Generally, the combination of more sensors does not always yield the better performance (e.g., two accelerometers at lower arm and upper arm may not improve the detection of OD1 and OD2). The best-combination performance increases dramatically as the group size increases from 1 to 4. F1-NoNull is almost unchanged after reaching the best performance. Meanwhile, the F1-Null of 17 sensors is less than the best performance by 10% in average. As discussed above, the presence of not-so-distinguishable sensors (e.g., shoe sensors) in the *classifier fusion* makes the recognition more confused in detecting the *null* class.

Table 6.4 gives the subset combination of 17 sensors that gets the best result. The orientation sensors on the back and arm (gyroscope and magnetic sensors) distinguish well the hard-to-classify activities in the Opportunity dataset.

6.6 Conclusion and Future Work

We have introduced the unified multimodal system for activity spotting by processing different sensors in a homogeneous way based on the template matching WarpingLCSS. Two fusion frameworks are investigated: the classifier fusion and the signal fusion. The results of the experiments show the flexibility and efficiency of our system in handling multimodal sensors. The more sensors are added into the system, the equal or better performance is achieved in average. Moreover, the

system is flexible in adding or removing activity classes. The classifier fusion framework provides the ease to add or remove sensors. Meanwhile, the signal fusion framework yields the better performance in classifying *null* classes due to a global view of data. In future, we plan to apply sensor selection algorithms in the classifier fusion framework to achieve the best performance. We also plan to investigate other modalities in our system.

6.7 Acknowledgment

This work has been supported by the Swiss Hasler Foundation project Smart-DAYS.

7

One-Time Point Annotations for Activity Recognition

Long-Van Nguyen-Dinh, Alberto Calatroni, Gerhard Tröster

Supporting One-Time Point Annotations for Gesture Recognition with Wearable Sensors

IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), pending *(Under review at the time of writing)*, 2015.

Abstract

This paper investigates a new annotation technique that reduces significantly the amount of time to annotate training data for gesture recognition. Traditionally, the annotations comprise the start and end times (i.e., temporal boundaries) and the corresponding labels of gestures in the sensor recording. In this work, we propose a one-time point annotation in which labelers do not have to select the start and end time carefully, but just mark a one-time point within the time a gesture is happening. The technique gives more freedom and reduces significantly the burden for labelers. To make the one-time point annotations applicable, we propose a novel BoundarySearch algorithm to find automatically the correct starting and ending boundaries of gestures by discovering data patterns around their given one-time point annotations. The corrected annotations are then used to train the gesture recognition system. We evaluate the quality of the corrected annotations and their recognition performance on three public data sets with various gesture classes (10-17 classes) recorded with different sensor modalities. The results show that training on the corrected annotations can achieve the performance close to a fully supervised training on clean annotations (lower by just up to 3% F1-score on average). The BoundarySearch algorithm is also well suited as a generic method to detect similar parts inside two sequences.

7.1 Introduction

Wearable computing has emerged rapidly with the increasing availability of devices, like smart watches, glasses and sensor-equipped garments. A core component to allow these devices to be aware of our context is the recognition of human activities and gestures. By identifying gestures, wearable computing may for instance enable human-computer interaction [96] or enhance social interaction [97]; wearable technologies may support workers in industrial environments [2] or provide health-care monitoring [98].

Online gesture recognition (*spotting*) requires types of gestures and their temporal boundaries recognized in the incoming streaming sensor data. This is accomplished by using supervised learning approaches [9, 54–56, 69] on different sensing modalities, like acceleration, angular velocity or video. To perform supervised learning, an annotated training data set is required to model gestures. Specifically, the annotations comprise the start and end times (i.e., temporal bound-

aries) of gestures of interest and their corresponding labels. Reference data sets are usually annotated carefully by a small number of experts to be as accurate as possible. However, in the labeling process, it is costly to hire experts and extremely time-consuming: it may require 7-10 hours to annotate gestures in a 30-min video [12]. Hence, for large data sets, the labeling cannot be done or requires a huge human effort.

To overcome these issues, we proposed recently a new way to get gesture annotations for training data sets by using crowdsourcing [68,99]. Crowdsourcing is defined as a model that outsources tasks which are traditionally performed by experts to a crowd of ordinary people [17] to reduce the cost and time. Crowdsourcing can be used to acquire annotations offline for an existing gesture data set by asking crowdsourced labelers to watch and label video footages of gesture recording [68]. Another way to exploit crowdsourcing is for instance asking crowdsourced users to record and annotate their own activities and gestures in real-time [99]. However, labels obtained from crowdsourcing are provided by low-commitment anonymous workers, thus they are commonly unreliable and noisy [28, 99]. Some instances of gestures can be associated to wrong labels or not be labeled at all. Other annotation noises can be inaccurate identifications of start and end time of gesture instances. Our previous work [99] showed that machine learning methods can tolerate at most 40% jitter level—the percentage of samples in one instance that are wrongly annotated—in all training instances. In extreme cases when jitter levels go beyond that limit, the performance degrades significantly. Severely noisy annotations are more likely to happen in real-time labeling because it is hard to control the annotation quality. Hence, to improve the performance, it may require costly extra human effort for correcting and cleaning the annotation. However, it is almost impossible to ask anonymous crowdsourced labelers to correct and clean their annotation because it is very time consuming and they may not remember what they have done. Consequently, when jitter levels go beyond the limit, we need novel ways to utilize those noisy annotations and maintain the performance.

In this work, we are especially interested in an extreme case, one-time point annotation (i.e., the start and end time of a gesture are equal). In the one-time point annotation, the boundary of a gesture shrinks to minimum, just a one-time point. One-time point annotation is likely to happen in real-time labeling, for example, when labelers remember to annotate the start of a gesture but forget to annotate the end. In general, if we allow the one-time point annotation, labelers do not have to select

the start and end time, just indicate quickly a one-time point within the time a gesture is happening. Hence, we give more freedom for labelers and reduce significantly the burden in annotation, both in real-time labeling or offline labeling by either experts or crowdsourced labelers. Consequently, the labeling process is considerably faster. However, the one-time point annotated data set cannot be used directly to model gesture classes. We need an algorithm in the preprocessing step to find the correct start and end time of each gesture around its one-time point annotation so that the segment can represent the pattern of gestures. Ideally, the recognition performance obtained from using these corrected/fixed annotations for training should be no worse than that obtained from using the clean annotations.

7.1.1 Contributions

In this paper, we make the following contributions:

1. We propose a new annotation technique for gesture data sets in which a labeler indicates a one-time point within the time a gesture is happening and its corresponding label.

2. We propose a novel algorithm to search for the correct start and end time of a gesture around its given one-time point annotation.

3. We evaluate how good the fixed annotations are compared to the ground truth as well as how well those fixed annotations can be used as a training data set for modeling and then classifying gestures.

The rest of the paper is organized as follows. In the next section, we first review existing work in annotation techniques for gesture recognition and online gesture recognition methods. Then, we present our proposed method to search for the boundaries of a gesture given its one-time point annotation. The experiments are described in Section 6.4. We present quantitative results evaluating the correctness of the fixed boundaries and their performances on training gesture models against the baselines in Section 7.5. Finally, the work is concluded by summarizing the main results and by showing potential research directions.

7.2 Related Work

In this section we discuss related work in the fields of gesture recognition.

7.2.1 Annotation Techniques

Annotated training data are required for supervised learning techniques to build gesture models. Therefore, many annotation techniques have been proposed to collect annotated samples. There are *offline annotation* techniques which rely on video and audio recordings [12], or subject self-report of activities at the end of the day [13]. *Online annotation* (i.e., real-time) techniques perform the annotation during execution of the activities, like experience sampling [14] which prompts periodically to a user to ask information about his current activities, or direct annotation in which users responsibly provide a label when an activity begins and also indicate when the activity ends [15]. There is a trade-off between the accuracy of an annotation technique and the amount of time required for annotation [16]. For example, offline annotation on video recordings by experts can provide accurate annotations, however it is extremely time consuming [12], 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 [12] or provide a course to teach subjects carefully how they should record and annotate their data correctly [54].

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 [18]. Recently, crowdsourcing has been exploited also in the field of activity recognition to collect annotated training data sets for both offline annotation and online annotation scenarios [15, 68, 70–72]. These works showed that crowdsourced data are erroneous, therefore, filtering strategies such as recruiting multiple labelers for the same annotation tasks and applying outlier removal techniques should be used to reject low-performing and malicious workers as well as reduce labeling noise. However, multiple labelers can be applied only in offline annotation

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

when video footage or audio recordings are available. In the online annotation, it is more likely that the level of noise increases and it is also more difficult to clean the annotation. In previous work [99], we analyzed annotation noises from crowdsourcing and investigated the impact of the crowdsourced noisy annotations on the training of activity recognition methods.

These existing annotation techniques for gesture recognition require labelers to indicate the temporal boundaries (start and end time) of gestures as accurate as possible. According to our best knowledge, there is no previous work that supports one-time point annotation in collecting activity data sets.

7.2.2 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) [3–6], template matching methods (TMM) using mostly dynamic time warping—in short DTW [2,7,8] and support vector machines [9–11].

HMMs are not appealing since a large amount of training data is required to get results comparable to other TMMs and SVM. The issue of the amount of training data is mentioned for example in [74], 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 [75], 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 [3,76,77].

Along with DTW, the other commonly used similarity measure for matching two time series is longest common subsequence *LCSS* [63]. In our previous work [69], we introduced two variations of LCSS-based template matching (SegmentedLCSS and WarpingLCSS) for online gesture spotting. These LCSS-based classifiers 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 [69]). We applied these methods to accelerometer data [69] and also other sensor modalities (e.g., gyroscope, mag-

netic field) in a unified multimodal framework [87]. Furthermore, our methods were shown to be much more robust to noisy crowdsourced annotations in training data sets than DTW-based TMM methods and SVM [99]. Our methods can be also used as a preprocessing filtering component to clean training data sets with severe label noise before feeding the training sets into other learning techniques.

A large body of literature focuses on a recognition performed on video data, for example the recognition of sign language (see [75–77,82]). 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 the 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 [83] 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 motion sensors; third, capturing video data is much more intrusive due to privacy concerns.

7.3 Methodology

In this section, we describe our boundary fixing approach to find the start and end boundaries of gestures given their one-time point annotations in signals obtained from body-worn sensors.

The sensor signals are first quantized and converted into sequences of symbols (strings). The initial start and end boundaries of each gesture are then set loosely around its given one-time point annotation to ensure the correct boundaries fall inside the segment. We assume that the executions of gestures of interest are continuous (i.e., no pause within the gesture). Hence, within the initialized boundaries of a gesture, the motion and non-motion parts are detected and the boundaries will be shrunk to the motion segment containing the annotated point. However, our work does not require users to have a pause before and after gesture execution — users can perform all gestures in the natural way and non-motion parts may not occur in the initialized boundaries. Finally, we propose a novel boundary searching algorithm (BoundarySearch) to seek for the good boundaries of a gesture around its given one-time point annotation and within the initialized boundaries such

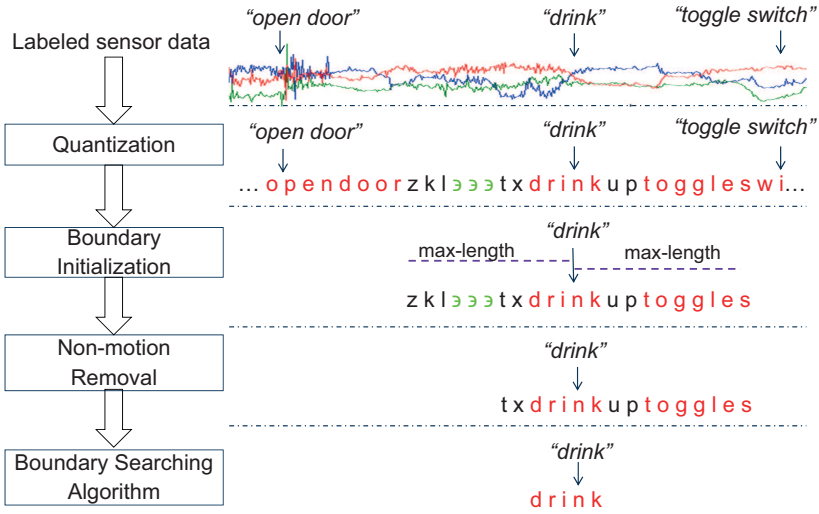

Figure 7.1: Data processing flow of the proposed boundary fixing approach. An example is shown on the right. Three annotated gestures "open door", "drink", and "toggle switch" with their one-time point annotations indicated as black arrows are given in the training data. In this example, the maximum length of "drink" gesture is given and it is used to initialize the boundaries of the "drink" gesture. ə represents non-motion symbols.

that the similar patterns of instances from the same gesture class are found. Figure 7.1 shows the data flow and its illustration through different processing components in the boundary fixing approach. We describe these components in details in the following.

7.3.1 Quantization

The quantization step transforms the raw signals into a one-dimensional (1D) string of symbols. This step reduces data dimensionality and thus it improves computational efficiency.

Let n denote the number of signal channels from 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) $\mathbf{w}(t)$. This is performed by using k-means clustering on the set of n-dimensional vectors $\mathbf{x}(t)$, $\forall t, 1 \leq t \leq N$. The choice of the number of clusters k (i.e., number of symbols) is done through cross-validation or empirically. The output of k-means is a set of k n-dimensional cluster centers, $\zeta_0 \dots \zeta_{k-1}$, 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 $w(t)$ as follows:

$$w(t) = \alpha_i | i = \underset{i}{\operatorname{argmin}} \|\mathbf{x}(t) - \zeta_i\|_2 \quad . \tag{7.1}$$

Let $d(\alpha_i, \alpha_j)$ denote the distance between two symbols α_i and α_j, given by the correspondent distance between their assigned cluster centers, normalized to fall in the interval $[0, 1]$.

$$d(\alpha_i, \alpha_j) = \frac{\|\zeta_i - \zeta_j\|_2}{max_{i,j} \|\zeta_i - \zeta_j\|_2} \quad . \tag{7.2}$$

7.3.2 Boundary Initialization

In this step, we initialize the start and end boundaries of each gesture instance around the one-time point annotation so that the ground truth of the boundaries stays inside the initialized boundaries.

We consider two cases depending on whether there is a prior knowledge of the maximum gesture length for each class of interest or not, namely MaxLen and NoLen respectively. In the MaxLen case, we extend around the annotated point for each gesture instance the maximum length toward the two ends to get the initial start and end time of the gesture. In this case, the boundary of the gesture is extended by at least 100%. However, if the extension with the maximum length around the annotated point goes beyond its previous or subsequent one-time point annotation, we take the previous or subsequent time point as the initial start or end time of the gesture, respectively. In the NoLen case (i.e., the maximum length information is not known), we extend the boundary of each gesture instance to the previous and subsequent time points of the annotation.

7.3.3 Non-Motion Removal

The non-motion removal step is optional and applied only when an accelerometer attached on the arm is used in the system. We detect motion and non-motion parts by using a simple method introduced by Benbasat and Paradiso [100] for accelerometer signals on arm. When the variance of one or more axis in the window exceeds a start motion threshold, we indicate that the motion part begins. The motion part stops when the variances of all of the axes are below a stop motion threshold. We use a 16-sample window with a start motion threshold of 100 milli-g^2 and a stop motion threshold of 50 milli-g^2 as suggested in [101]. After this step, the boundaries are shrunk to the motion segment containing the annotated point.

As we shall show, the BoundarySearch algorithm corrects the boundaries of each gesture by discovering the similar patterns around the one-time point annotations. Therefore, the non-motion part, if existing, should be discarded from the initial boundaries before being fed into the BoundarySearch algorithm to ensure that non-motion signals that are likely to be similar intrinsically will not contribute to the pattern of the gesture class.

7.3.4 Boundary Searching Algorithm

We assume that gestures are performed in a random order in the training data set. If two gestures must always occur together in the same order, they should be grouped into one atomic gesture. Our proposed boundary searching algorithm BoundarySearch searches for the correct boundaries of gestures around their given one-time point annotations based on seeking similar patterns around the annotated points. Therefore, if two gestures always occur together, the BoundarySearch algorithm as described below tends to group them into one atomic gesture.

Up to this point, each gesture instance is a string of symbols with the determined initial boundaries. Let $w(i)$ denote the i-th symbol within a string w of length L. We can write $w = w(1), w(2), ..., w(L)$, or shortly $w = w(1..L)$. A substring $w(i..j)$ of w is a contiguous subsequence of w starting from the i-th position to the j-th position ($1 \leq i \leq j \leq L$).

Let w_1 and w_2 be two strings of gesture instances G1 and G2 of the same class comprising L_1 and L_2 symbols respectively. Gesture G1 has a one-time point annotation at the K_1-th symbol in w_1 ($1 \leq K_1 \leq L_1$) and gesture G2 is annotated at the K_2-th symbol in w_2 ($1 \leq K_2 \leq L_2$). Given

w_1 and w_2, the BoundarySearch algorithm first detects all pairs of a substring $w_1(s_1..e_1)$ of w_1 ($1 \leq s_1 \leq e_1 \leq L_1$) and a substring $w_2(s_2..e_2)$ of w_2 ($1 \leq s_2 \leq e_2 \leq L_2$) such that they are optimized matching substrings (i.e., the similarity score between them as defined below is maximized and positive). It then takes only a pair of matching substrings that cover the one-time point annotations (i.e., $s_1 \leq K_1 \leq e_1$ and $s_2 \leq K_2 \leq e_2$). Subsequently, s_1 and e_1 can become the new start and end boundary of gesture G1; and s_2 and e_2 can become the new start and end boundary of gesture G2.

7.3.4.1 Detecting Optimized Matching Substrings

First, we describe the algorithm to detect all optimized matching substrings inside two strings w_1 and w_2. Given two strings w_1 and w_2, we define a warping path to align the two strings by either

- aligning an element $w_1(i)$ of w_1 to an element $w_2(j)$ of w_2,
- or warping two consecutive elements $w_1(i-1)$ and $w_1(i)$ of w_1,
- or warping two consecutive elements $w_2(j-1)$ and $w_2(j)$ of w_2.

The warping path starts with the alignment between the first element of w_1 and the first element of w_2. The path advances one step at a time, both indices i and j can only increase by at most 1 on each step along the path. Figure7.2 illustrates the warping path to align two strings.

We define a similarity weight for each alignment. If two symbols match, a similarity weight is a positive reward $R = 1$. In case of a mismatched alignment or a warping between two elements α_i and α_j, a similarity weight is a negative penalty computed as $-p * d(\alpha_i, \alpha_j)$, where p is a penalty parameter of the dissimilarity and $d(\cdot,\cdot)$ is the distance between two symbols as defined in Equation 7.2. Let denote $\Psi_A(\alpha_i, \alpha_j)$ and $\Psi_W(\alpha_i, \alpha_j)$ be a similarity weight of an alignment and a warping respectively between two elements α_i and α_j.

$$\Psi_A(\alpha_i, \alpha_j) = \begin{cases} 1 & \text{, if } \alpha_i = \alpha_j \\ -p * d(\alpha_i, \alpha_j) & \text{, otherwise} \end{cases} \tag{7.3}$$

(7.4)

$$\Psi_W(\alpha_i, \alpha_j) = -p * d(\alpha_i, \alpha_j) \tag{7.5}$$

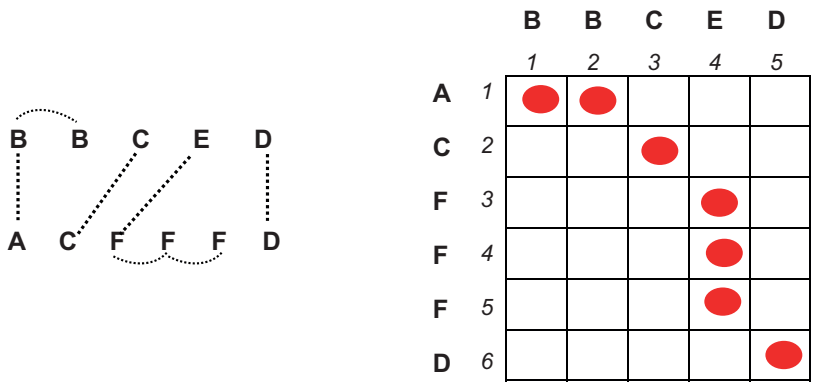

Figure 7.2: Illustration of an alignment between two strings "BBCED" and "ACFFFD" and the corresponding warping path starting from the first indices at the top left to the last indices at the bottom right. In this example, two symbols "B" of the first string are warped and so are three symbols "F" of the second string.

A warping path has a similarity score which accumulates all similarity weights along the path. The rationale behind introducing the warping between two consecutive elements in one string in case of mismatch is that if the string contains contiguous repetitions of a mismatched symbol (e.g., the repetitions are due to a slower execution of a gesture), the optimal warping path with the maximum similarity score should warp those repetitive symbols and the penalty is counted only once. In the example shown in Figure7.2, the mismatch in the alignment between A and B and the mismatch in the alignment between E and F are penalized only once due to the warpings on B symbols and F symbols.

Let $\text{Sim}(w_1, w_2)$ be a maximum similarity score between two strings w_1 and w_2 attained over all possible warping paths between w_1 and w_2. We define $B(i, j)$ to be the maximum similarity score between substrings of w_1 ending at the i-th position and substrings of w_2 ending at the j-th position. Formally,

$$B(i, j) = \max_{\substack{1 \le i' \le i \\ 1 \le j' \le j}} \text{Sim}(w_1(i'..i), w_2(j'..j)) \tag{7.6}$$

The BoundarySearch algorithm obtains the score $B(i, j)$ as follows.

$$B(i,j) = \begin{cases} 0 & \text{, if } i = 0 \text{ or } j = 0 \\ \max \begin{cases} 0 \\ B(i-1, j-1) + \Psi_A(w_1(i), w_2(j)) \\ B(i-1, j) + \Psi_W(w_1(i), w_1(i-1)) \\ B(i, j-1) + \Psi_W(w_2(j), w_2(j-1)) \end{cases} & \text{, otherwise,} \end{cases} \tag{7.7}$$

The algorithm starts with an empty string and the corresponding B score is 0. A value of B is updated by taking the maximum similarity from four possibilities: accepting an alignment between two current symbols in the two substrings; warping the current element with its previous one in the substring of w_1; warping the current element with its previous one in the substring of w_2; and a zero.

The similarity score B is nonnegative. The preceding vaues $B(i-1, j)$, $B(i, j-1)$, $B(i-1, j-1)$ in the formula above would not be of interest if they were negative because we can always get a better value for the similarity score $B(i, j)$. It can be done by treating the negative values as if they were zero and choosing a starting index for the substring i′ = i or j′ = j in Equation 7.6. Thus storing only nonnegative values in B ensures that the BoundarySearch algorithm can find optimal matching substrings. $B(i, j) = 0$ also indicates there is no substring matching up to position i and j.

Our algorithm to compute $B(i, j)$ is similar to the local alignment algorithm proposed by Smith and Waterman [102] to seek matching substrings within two strings. However, we define different similarity weights between symbols and a different type of warping (i.e., we warp similar consecutive symbols in the same string to avoid multiple penalties, while Smith and Waterman used edit operations including insertion and deletion to align two strings).

All optimal matching substrings between two strings can then be found easily by tracing back the matching path starting from an element of B greater than 0 and ending with an element of B equal to zero. Specifically, we define a dependency graph G, a directed graph whose vertices are labeled with the positions (i, j) and each vertex (i, j) has an edge from one of the preceding positions $(i-1, j)$,$(i, j-1)$,$(i-1, j-1)$ whose values contribute to the value $B(i, j)$ and $B(i, j) > 0$. The dependency graph G provides the information of how the B scores are generated and thus all optimized matching substrings can be found.

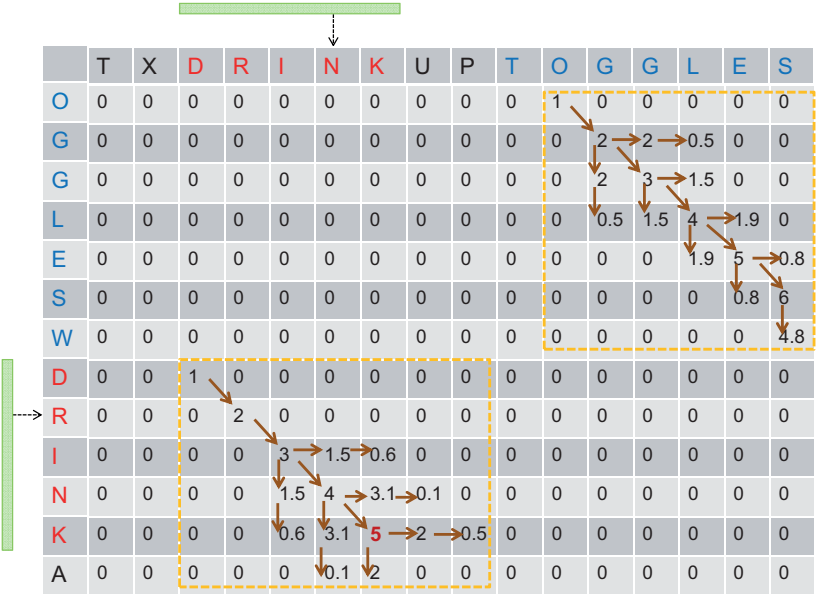

Figure 7.3: Illustration of BoundarySearch algorithm between two "drink" gestures with incorrectly initialized boundaries "TXDRINKUP-TOGGLES" and "OGGLESWDRINKA", and the corresponding dependency graph. The initialized boundaries wrongly cover some parts of "toggle switch" gesture. Here we assume the distance between two symbols is their difference in encoded English alphabets (A-Z are converted to 0-25) and the penalty p is 0.3. In the dependency graph, the horizontal or vertical brown arrows show warpings in one string, meanwhile the diagonal arrows show the alignments between elements of two strings. Two black arrows indicate the one-time point annotations of the gestures. Two clusters of connected scores show the longest matching substrings ("DRINKUP" with "DRINKA", and "OGGLES" with "OGGLESW"). The targeted cluster on the bottom left with the highest similarity score $\mathbf{M} = 5$ spans the marked annotation. Hence, the fixed boundaries of two gestures are shown in green bars.

Figure7.3 illustrates an example of the similarity score B computed between two "drink" gestures and the corresponding dependency graph.

7.3.4.2 Boundary Correction

The dependency graph shows clusters of connected scores which indicate longest matching substrings (i.e., from the starting vertex of a cluster to the furthest indices the cluster can reach). An illustration of these clusters is also shown in Figure7.3. As can be seen, the value of the penalty parameter p decides how much mismatching is acceptable in the longest matching substrings or it can decide the size of the cluster of connected scores (i.e., the cluster can be extended until the value of B is set to 0).

We then select only the cluster that covers the one-time point annotations of the two gestures w_1 and w_2 indexed at the K_1-th and K_2-th symbols respectively. In the targeted cluster, we select the highest similarity score $\mathbf{M}$ indexed at (v_1, v_2) such that the path from the starting vertex of the cluster indexed at (u_1, u_2) to (v_1, v_2) still covers the marked annotated points. Formally, $[u_1, v_1, u_2, v_2]$ and $\mathbf{M}$ are defined as follows.

$$[u_1, v_1, u_2, v_2] = \underset{\substack{i',i,j',j: \\ 1\le i'\le K_1\le i\le L_1 \\ 1\le j'\le K_2\le j\le L_2}}{\operatorname{argmax}} \operatorname{Sim}(w_1(i'..i), w_2(j'..j)) \tag{7.8}$$

$$\begin{aligned} \mathbf{M} &= \max_{\substack{1\le i'\le K_1\le i\le L_1 \\ 1\le j'\le K_2\le j\le L_2}} \operatorname{Sim}(w_1(i'..i), w_2(j'..j)) \\ &= B(v_1, v_2) \end{aligned} \tag{7.9}$$

Finally, the new boundary for gesture G1 is set from u_1 to v_1 and the new boundary for gesture G2 is set from u_2 to v_2. The normalized similarity between the two gestures is $\mathbf{M^{norm}}(G1, G2) = \mathbf{M}/max(\|G1\|, \|G2\|)$, where $\|G1\|$ and $\|G2\|$ are the new lengths of the gestures G1 and G2, respectively. In case the target cluster can not be found, u_1, v_1, u_2, v_2 are set to 0; $\mathbf{M}$ and $\mathbf{M^{norm}}$ are also 0.

Given an instance k of gesture class c, we run the boundary fixing for this instance paired with all other instances in the same class. Let $[s_{k,i}, e_{k,i}]$ be the boundary of the instance k after running the boundary fixing with instance i of class c, and $\mathbf{M^{norm}_{k,i}}$ be the corresponding normalized similarity. Then the final boundaries of the instance k are the

average starting and ending time from all possible fixing boundaries.

$$\begin{aligned} s_k &= \sum_{i=1,i\neq k}^{n} s_{k,i} * \mathbf{M}^{\mathbf{norm}}_{\mathbf{k,i}} / \sum_{i=1,i\neq k}^{n} \mathbf{M}^{\mathbf{norm}}_{\mathbf{k,i}} \\ e_k &= \sum_{i=1,i\neq k}^{n} e_{k,i} * \mathbf{M}^{\mathbf{norm}}_{\mathbf{k,i}} / \sum_{i=1,i\neq k}^{n} \mathbf{M}^{\mathbf{norm}}_{\mathbf{k,i}}, \end{aligned} \tag{7.10}$$

where n is the number of instances in the class c.

If the final fixed boundaries of the instance k are 0 ($s_k = 0$ and $e_k = 0$), it means the BoundarySearch algorithm cannot find any similar substrings/patterns between the gesture instance with all other instances in the same gesture class. In this case, the instance k is discarded by setting its label to *null*.

7.4 Experiments

In this section, we first present three gesture data sets used to evaluate the boundary fixing approach. We then define metrics to quantify the quality of the fixed annotation compared with the ground truth annotation. Next, we present two state-of-the-art gesture recognition methods used to train gesture models with the fixed annotation. Finally, recognition evaluation metrics are discussed.

7.4.1 Description of Data Sets

We used three data sets including various gestures of different length which have been labeled manually by experts. The experts' annotation is the ground truth of the data sets. Data which do not correspond to any of the gestures of interest in the data sets are labeled as *null* class. The signals of all sensors at these data sets are recorded at a frequency of 30Hz. The average durations of gestures of different classes range from about 2 seconds to 32 seconds. Table 7.1 shows the list of gestures of these data sets and their average lengths in sample unit (i.e., lengths of the corresponding sample-based time series). Following, we describe briefly each data set[3].

[3]Skoda and Opportunity data sets can be downloaded from `http://www.ife.ee.ethz.ch/research/groups/Dataset`.

Table 7.1: Gestures in Opportunity, Skoda, and HCI Data Sets

HCI Gestures				
Circle (83 ± 11)	Triangle (81 ± 10)	Square (92 ± 14)	Infinity (95 ± 10)	Slider (61 ± 6)
Their Specular Reflection				Null (106 ± 313)

Opportunity Gestures			
open Drawer 1 (68 ± 19)	open Drawer 2 (69 ± 6)	open Drawer 3 (83 ± 5)	clean Table (120 ± 48)
close Drawer 1 (65 ± 14)	close Drawer 2 (63 ± 5)	close Drawer 3 (77 ± 8)	Toggle Switch (40 ± 11)
open Door 1 (92 ± 11)	open Door 2 (102 ± 6)	close Door 1 (103 ± 10)	close Door 2 (102 ± 21)
open Dishwasher (94 ± 19)	close Dishwasher (86 ± 11)	open Fridge (75 ± 6)	close Fridge (76 ± 6)
Drink (194 ± 40)	Null (81 ± 181)		

Skoda Gestures		
write on notepad (894 ± 80)	check gaps on the front door (702 ± 90)	open hood (955 ± 110)
close hood (940 ± 68)	open left front door (433 ± 81)	close left front door (412 ± 32)
close both left door (785 ± 96)	check trunk gaps (838 ± 55)	check steering wheel (550 ± 44)
open and close trunk (912 ± 124)	Null (218 ± 628)	

Values in parentheses are the average length ± one standard deviation in samples.

7.4.1.1 Skoda

The Skoda data set [43] 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 by a low variability in execution because the subject performed carefully each manipulative gesture in the same manner. We use a 3D accelerometer at the subject's dominant (right) lower arm for the evaluations. The non-motion removal step in the boundary fixing approach is applied. In the quantization step, the number of symbols (i.e., number of clusters in k-means) is selected empirically k = 20.

7.4.1.2 HCI

The HCI data set [44] 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. In this data set, we also use data from one 3D accelerometer at the subject's dominant (right) lower arm. The non-motion removal step in the boundary fixing approach is applied. The number of symbols in the quantization step is selected empirically k = 20.

7.4.1.3 Opportunity

The Opportunity data set [12] consists of 17 daily activities recorded in a naturalistic environment akin to an apartment with various sensor modalities attached at different on-body positions. 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 two different doors and three drawers at different heights which make the recognition more challenging. Hence, in the Opportunity data set we use 6 different sensors worn at different on-body positions to achieve the best discrimination among gesture classes as shown in our previous work [87]. Those sensors are 3D magnetic field on back, 3D gyroscope on right upper arm, 3D gyroscope and 3D magnetic field on right lower arm, 3D accelerometer on left upper arm and 3D accelerometer on left lower arm. Here we do not use a 3D accelerometer on the dominant

right arm for the experiments, thus the non-motion removal step in our fixing boundary approach as discussed in Section 7.3.3 is not applied for this data set. The number of symbols is selected empirically k = 120 in the quantization step.

7.4.2 Fixed Annotation Evaluation Metrics

To evaluate the quality of fixed annotations compared to ground truth, we use a taxonomy of annotation noises similar to the one we proposed in our previous work to evaluate the quality of crowdsourcing annotations [99]. Specifically, we define *boundary jitter* as the presence of a time-shift in the annotation boundaries, while the label matches the actual gesture (ground truth). We 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. Figure7.4 illustrates the subclasses of *boundary jitter*.

- *Extend*: The starting boundary is set earlier and the ending boundary is set later. All information belonging to the gesture instance is preserved, but noisy samples are attached at the two ends of the gesture instance. Noisy samples can belong to another gesture class or to *null* class.
- *Shrink*: The starting boundary is set later and the ending boundary is set earlier. In this case, some part 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 noisy samples are 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.

Beside boundary jitter, fixed annotations can also fall into two other types, namely *good* and *delete*.

- *Good*: Fixed boundaries of a gesture are perfectly matching its ground truth.

- *Delete*: The BoundarySearch algorithm cannot find any similar patterns in a gesture instance as compared with all other instances in the same class. In this case, the fixed boundary is empty and the gesture is then categorized as *delete* and marked as *null*.

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 fixed boundary. The jitter level also indicates how much the boundaries stray from the correct annotation. Given a gesture instance, let *start* and *end* be the starting and ending time of the fixed 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 fixed starting time and the correct starting time ($\Delta s = |start - GT\_start|$). Similarly, we define Δe as the time difference between the fixed ending time and the correct ending time ($\Delta e = |end - GT\_end|$). Δs and Δe are illustrated in Figure7.4 for the different boundary jitter types. The *jitter level* parameters are calculated as follows:

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

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

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

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

In our work, we evaluate the quality of fixed annotations by analyzing the distribution of different kinds of annotation noises and the magnitude of jitter levels in the fixed annotations.

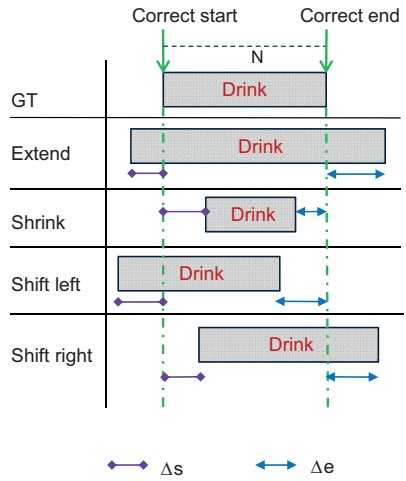

Figure 7.4: Illustrations of boundary jitter in fixed annotations. GT stands for ground truth. The blue dash-dotted lines indicate the correct boundary of a gesture.

7.4.3 Description of Experiments

For each data set, we perform a 5-fold cross-validation (80% of samples for training and the remaining 20% samples for testing). From a training data set, we generate a one-time point annotation for each gesture by selecting randomly a point inside the ground truth. The boundary fixing approach proposed in Section 4.3 is then applied to find the good boundaries for each gesture given its one-time point annotation. The recognition system is trained on the fixed annotations and evaluated on the clean test set (i.e., annotated by experts).

Two spotting techniques used in the recognition system are our template matching method WarpingLCSS [69, 87, 99] and SVM. WarpingLCSS was proposed recently and has been shown to be robust for online gesture recognition, especially in noisy annotated data sets [99]. We describe the WarpingLCSS method briefly below. 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 [86] for training SVM.

To investigate the quality of our fixed annotations on gesture recognition, we compare the performance of the spotting methods trained with ground truth annotations against those trained with the fixed annotations. We also evaluate their performances on the annotations which are extended loosely around the one-time points but are not corrected by our proposed BoundarySearch algorithm.

7.4.3.1 WarpingLCSS

For self containment we will give a brief review of the WarpingLCSS algorithm that we proposed in previous works [69, 87, 99] to spot gestures in a sensor streaming data which are similar to a gesture template. A template is created for each gesture class in the training phase to represent the gesture pattern of that class. In the recognition phase, WarpingLCSS algorithm can automatically detect the gesture boundaries as well as their labels in the online streaming data s without pre-segmenting the data. In our algorithm, sensor signals are also quantized into a sequence of symbols as presented in Section 5.4.2.

If the WarpingLCSS algorithm encounters the same symbol in a template and in the current string, the similarity score W is increased by a reward of 1. Otherwise, W is decreased by a penalty proportional to the distance between the symbols scaled by the constant parameter p. Furthermore, if the string s is "warped" (i.e., contiguous repetitions of a symbol due to a slower execution of a gesture), the penalty is counted only once.

A gesture of class c is recognized for each local maximum of W that also exceeds a rejection threshold ϵ_c which is selected for class c in the training phase [69]. The end point of the gesture is set to the local maximum itself and the start point is found by tracing back the matching path. The WarpingLCSS algorithm can be implemented efficiently by using dynamic programming to constrain the time and memory complexity. The value of the penalty parameter p depends on the application and can be chosen by cross-validation to maximize the recognition performance. See [69, 99] for full details of WarpingLCSS on a single sensor modality and see [87] for WarpingLCSS on multimodality.

Our BoundarySearch algorithm to find the maximum similarity score B between substrings of two strings as shown in Equation 7.7 is a modified version of the WarpingLCSS algorithm. The only difference

is that negative similarity scores are accepted in WarpingLCSS. The reason is in WarpingLCSS the whole template will be compared with the data stream and anything different to some part of the template will contribute the penalties to the score. Meanwhile, in Boundary-Search we find boundaries of two similar gestures (i.e., only matching substrings) inside two strings, hence the negative score should not be considered and it can help to detect all optimized substring matchings.

7.4.4 Recognition Evaluation Metrics

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.

In our data sets, *null class* is predominant. Therefore, 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*. F1-NoNull score represents how well the recognition system detects activity classes of interest. A recognition system that has high values of both F1-Null and F1-NoNull predicts well both gesture classes and *null* class.

7.5 Results and Discussion

In this section we present and discuss the results of the conducted experiments to evaluate our proposed boundary fixing approach. We first present the quality of fixed annotations and then compare the performance of using the fixed annotations on training gesture models with those of baselines.

7.5.1 Quality of Fixed Annotations

To show the efficiency of our BoundarySearch algorithm in searching for good boundaries for gestures, we first analyze the quality of the initialized boundaries of gestures extended around one-time point annotation before fixing — before BoundarySearch algorithm is applied (see the data flow in Section 4.3). We have two cases for boundary initialization as discussed in Section 7.3.2: MaxLen or NoLen, depending on whether there is a prior knowledge of maximum length of each gesture class or not. Accordingly, we name the initialized annotations before fixing as NoFix-NoLen and NoFix-MaxLen. The jitter levels of these annotations are shown in Figure7.5. The initial boundaries are extended to cover the ground truth, hence, only extend jitter occurs. As can be seen, at least 50%-75% of gesture instances are extended by more than 100% for the three data sets in both cases NoFix-NoLen and NoFix-MaxLen. There are less than 2-4% of gesture instances suffering small values of extend level which are less than 30%. Note that the non-motion removal step is applied in HCI and Skoda data sets and thus the initial boundaries of gestures are shrunk significantly and they get closer to the ground truth. Hence, the extend levels of NoFix-NoLen in the HCI and Skoda data sets have a smaller range of values than those in the Opportunity data set. The prior knowledge of maximum length of each class (MaxLen) can help to reduce the extend levels significantly in the three data sets. However, the existence of many gesture instances with large extreme values of the extend levels shows the need of the BoundarySearch algorithm to fix the initialized boundaries in order to get gesture patterns and improve the performance.

After applying the BoundarySearch algorithm to fix the boundaries in the NoLen case (namely Fixed-NoLen), we show noise distribution and jitter levels of the fixed annotations, according to the taxonomy introduced in Section 7.4.2, in Figure7.6. Only few gesture instances are deleted because the BoundarySearch algorithm cannot find the similar pattern around the given one-time point annotation between those instances with all other instances in the same class. There are some instances that are perfectly matched with the ground truth. In each data set, there are about 96% of instances whose fixed boundaries fall into one type of boundary jitters. The values of jitter levels in the fixed annotations compared to the ground truth are shown by means of box-plot graphs in Figure7.6b. Particularly, we show the median, the 25th and 75th percentile, the min, the 98th percentile and the max

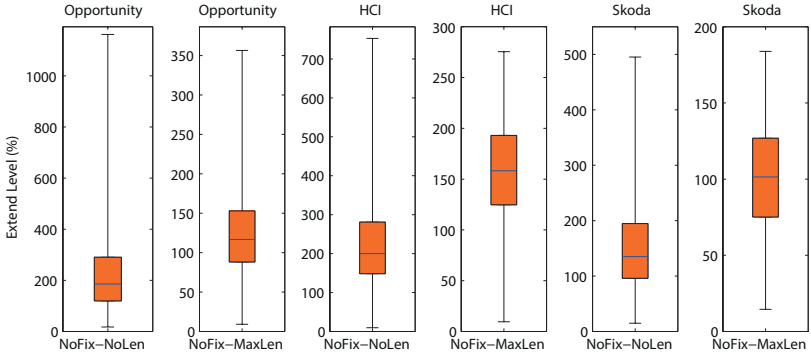

Figure 7.5: Extend levels of gesture boundaries initialized around one-time point annotations before BoundarySearch algorithm is applied. The box in the box plot covers from the 25th percentile to the 75th percentile (i.e., 50%) of the extend levels. Blue lines show the median values and black lines show the minimum as well as the 98th percentile. Note that the scales are different for the sake of visibility.

values of jitter levels of boundary jitters for those three data sets. It can be seen that our BoundarySearch algorithm reduces extremely the jitter levels in the initial boundaries as already shown in Figure7.5 in case of NoFix-NoLen. Extreme values drop dramatically (e.g., in the Opportunity data set, the maximum extend level before fixing is above 1000%, meanwhile it decreases to 200% after fixing). Moreover, up to 70% of instances in the Opportunity data set after fixing suffer only small values of jitter levels which are less than 30%. The corresponding percentage of instances in the HCI and Skoda data sets are 83% and 92% respectively. Hence, the BoundarySearch algorithm can efficiently search around the one-time point annotation and inside the large initial boundaries to discover the pattern of gestures.

Similarly, the analysis of the fixed annotations in the case of MaxLen (namely Fixed-MaxLen) is given in Figure7.7. The results have similar trend as those in Fixed-NoLen. The jitter levels decrease significantly compared to those of the initial boundaries before fixing (NoFix-MaxLen) shown in Figure7.5. The ranges of jitter levels in Fixed-MaxLen are smaller than those in the case of Fixed-NoLen. For example, in the Opportunity data set, the maximum extend level in Fixed-NoLen is 200%, meanwhile in the Fixed-MaxLen it reduces to

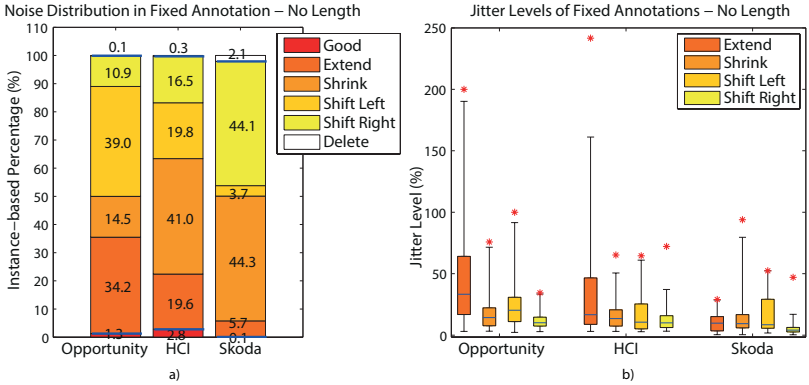

Figure 7.6: Analysis of fixed annotations in case the prior knowledge of maximum gesture length is not available (Fixed-NoLen). Blue lines in Fig. a) on the left split a boundary jitter part from good and delete types. In Fig. b), the description of box plot is the same as that in Figure7.5. The red star indicates the maximum level of jitter in each type of boundary jitter.

90%. The percentages of instances which have jitter levels less than 30% after fixing in the Opportunity, HCI and Skoda data sets are 78%, 87% and 92% respectively.

The quality of the fixed annotations compared with that of the non-fixed annotations show that our boundary fixing approach works well in finding the boundaries of gestures around the one-time point annotation in both cases, whether the prior knowledge of the maximum length is given or not.

7.5.2 Recognition Performance on Fixed Annotations

We compare the performance of using the fixed annotations on training gesture recognition with that of clean annotations (i.e., ground truth). Besides that, to show the robustness of our BoundarySearch algorithm, we also compare the performances of the fixed annotations (Fixed-NoLen and Fixed-MaxLen) with those of baselines which use the NoFix-NoLen and NoFix-MaxLen annotations for training.

Figure7.8 shows the performances of WarpingLCSS and SVM on the fixed annotations and the baselines of the three data sets. In the Op-

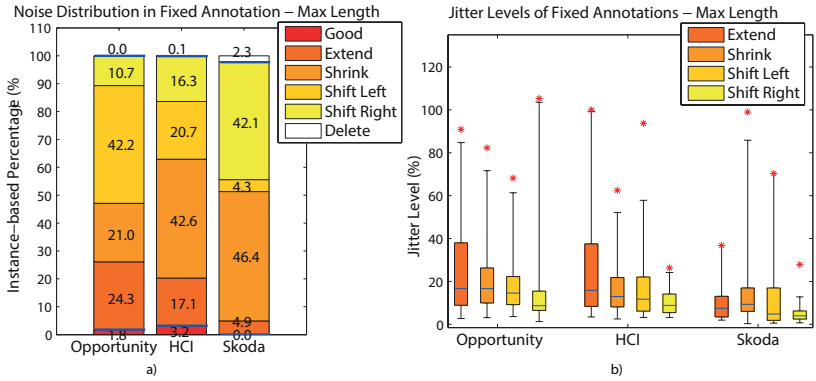

Figure 7.7: Analysis of fixed annotations in case the prior knowledge of maximum gesture length is given (Fixed-MaxLen). Interpretation of notations in this figure is the same as in Figure7.6.

portunity data set, the performances of Fixed-MaxLen are decreased by just up to 10% for F1-Null and F1-NoNull compared to those of clean annotation for both recognition methods. Without the prior knowledge of the maximum length, the performances of Fixed-NoLen are just slightly lower than those of Fixed-MaxLen (only lower by 2-3% F1-scores). However, when comparing between fixing and non-fixing, we can see that there is a huge difference in performance between the Fixed-NoLen annotations and the NoFix-NoLen annotations. Specifically, the performance of the Fixed-NoLen is higher than that of the NoFix-NoLen by up to 58% F1-score with SVM and up to 29% F1-score with WarpingLCSS. Similarly, the Fixed-MaxLen annotation achieves better performance than the NoFix-MaxLen (by up to 25% F1-score in WarpingLCSS and SVM). The performance of SVM for the NoFix-NoLen decreases dramatically, down to a F1-score of 2%, which is less than random guessing (which would be around 5% in a 18-class data set like Opportunity). This can be explained with the fact that the extend levels in NoFix-NoLen in the Opportunity data set are extremely high as shown in Figure7.5. Each gesture instance spans between the previous and subsequent annotated time points of the annotation, therefore, all classes contain a large number of wrongly labeled samples from other classes. Hence, it leads the SVM model to choose incorrect support vectors, which severely degrades the performance. Meanwhile,

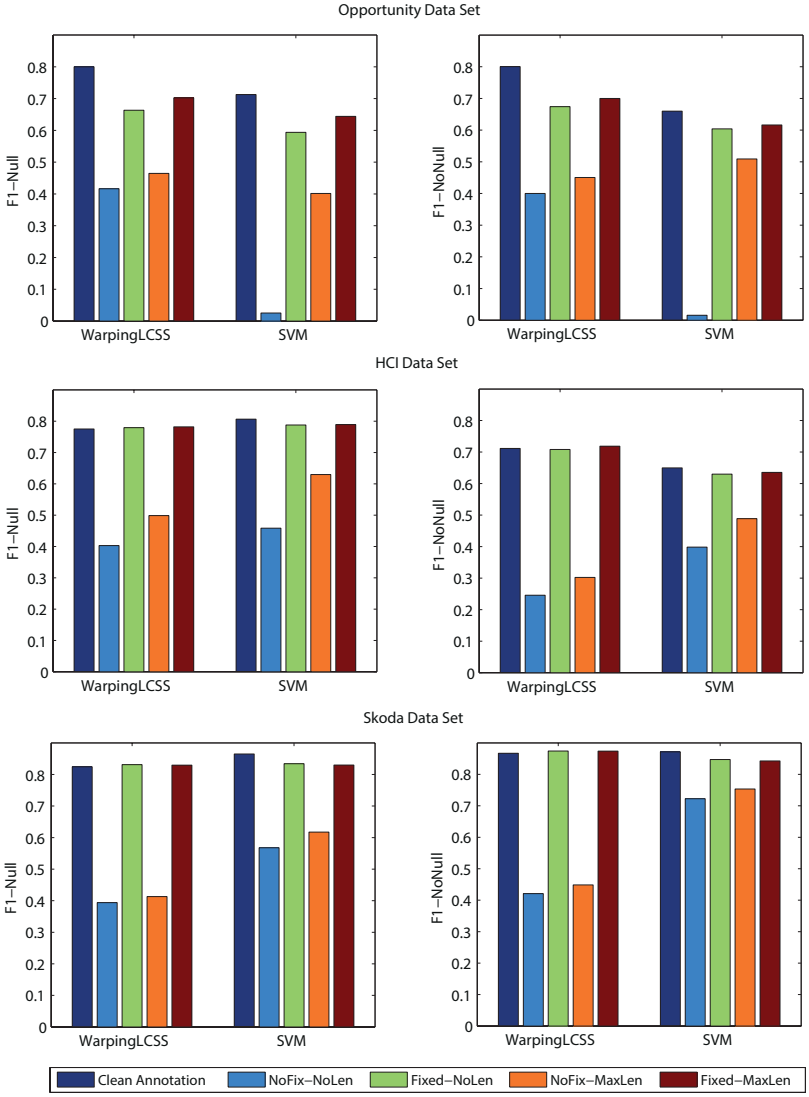

Figure 7.8: Performance of WarpingLCSS and SVM on fixed annotations in both cases of NoLen and MaxLen and on baselines.

after fixing, the Fixed-NoLen annotation of the Opportunity data set contains 70% of instances which have the jitter levels less than 30%. Hence, it reduces significantly the sample noises and improves the performance dramatically. When the maximum length of each gesture class is known, the extend levels of NoFix-MaxLen are reduced significantly as shown in Figure7.5, thus, the performance of SVM in NoFix-MaxLen is much better than that in NoFix-NoLen. WarpingLCSS performs stable in both cases of NoFix-NoLen and NoFix-MaxLen. The rationale is that WarpingLCSS finds the best template to represent each gesture class and then uses that template to spot gestures. It does not take all instances which severely contain wrongly labeled parts from other classes into the gesture models.

In the HCI and Skoda data sets, the performances of our fixed annotations, Fixed-NoLen and Fixed-MaxLen are the same as those of clean annotations and much better than those of the non-fixing baselines. For the non-fixed annotations, SVM performs better in these two data sets than in the Opportunity data set. The extend levels of the non-fixed annotations in the HCI and Skoda data sets are lower than those in the Opportunity data set, especially in the NoLen case, as shown in Figure7.5. In the HCI and Skoda, we apply the non-motion removal step in the non-fixed annotations and it reduces significantly the noisy labeled instances in each class. Moreover, HCI and Skoda data sets contain low levels of variability of the signals belonging to the null class. Thus, the noisy feature vectors in each gesture class are low-variant, leading to the better performance of SVM.

The results comply with our analysis on the quality of the fixed annotations as well as the baseline annotations discussed in Section 7.5.1. It is also consistent with our experiments on crowdsourced annotations shown in [99] where the jitter levels are simulated in a wide range of values. On average, the fixed annotations achieve a similar performance as the clean annotations, lower by just up to 3% F1-scores. The results show the efficiency of our boundary fixing approach in finding good patterns of gesture around the one-time point annotation, with or without knowing the maximum length of each gesture class.

7.5.3 Discussion

In our experiments, each gesture instance is annotated with its correct label and a one-time point within the time the gesture is happening. This new proposed annotation technique reduces significantly labelers'

burden on annotation. The results show the efficiency of our fixing boundary method in finding good boundaries for gestures around their one-time point annotations, and the fixed annotations can be used to train gesture models that achieve performances similar to the ones obtained from clean annotations. In this work noisy annotations are not considered. In the offline annotation by experts, we can assume that there is no *label noise* in the one-time point annotations (i.e., all gestures are annotated and each annotated point is associated with a correct label). However, in the real-time annotation or crowdsourced annotation scenarios, *label noise* is more likely to happen [99]. *Label noise* can contain instances of gestures which are associated to wrong labels or are not labeled. *Label noise* can also appear in the form of gesture instances which are labeled where no gesture of interest (i.e., *null)* actually occurs. If there are wrongly labeled instances in the one-time point annotations, our boundary fixing method can either delete them if there is no similar pattern between them and other instances in the same annotated class, or also correct their boundaries if there are also other wrongly labeled instances which actually belong to the same ground-truth class as them. After fixing the boundaries, if the fixed annotations are still affected by *label noise*, we can use WarpingLCSS to filter out those noisy instances before training a traditional classifier as already shown in our previous work [99]. Particularly, if the majority of instances of one class is correct, WarpingLCSS is able to pick a good template among noisy instances to represent the gesture class. Noisy instances in a gesture class are filtered out by removing instances which have an average similarity to other instances of the same class below a threshold. To clean noise inside the *null* instance, we can run WarpingLCSS with the selected templates on the training data annotated as null and discard any parts spotted as any gestures of interest. We also conducted experiments to investigate the impact of *label noise* in training annotation on the performance of the gesture recognition methods. For more information, see our previous work in [99].

In this work, we evaluated our proposed boundary fixing method on applications from wearable sensing and activity recognition. However, the BoundarySearch algorithm is a generic algorithm that can be useful in other pattern recognition applications, for example supporting one-time point annotation in systems where temporal boundaries must be segmented and annotated in the training data set, or finding similar substrings inside strings. One application could be for exam-

ple finding matching areas inside two images in the field of image processing.

In our work, raw signals are simplified and transformed into a 1D string of symbols with the use of k-means clustering which gives good results. Other quantization techniques such as SAX [103] or temporal clustering [81] can also be applied. Our BoundarySearch algorithm is then applied to these strings of symbols. However, the algorithm can be easily modified to work with higher dimensional time-series data or raw signals. Particularly, in Equation 7.7 to compute B, instead of comparing two symbols for a matching, a distance between two data samples can be computed and compared with a tolerance threshold to define an approximate match.

In our BoundarySearch algorithm, the penalty parameter p must be specified. In this work, we simply used the same penalty values chosen by cross-validation for the WarpingLCSS algorithm on the three data sets in our previous works [99]. In other data sets with only one-time point annotations and without any ground truth, the cross-validation can not be done to pick the best p value which yields the best quality (i.e., jitter levels) of fixed annotation. However, we can run the BoundarySearch algorithm with a range of values of p and the fixed boundaries can be shown and examined quickly by experts to decide which value of p should be chosen.

7.6 Conclusion and Future Work

In this work, we investigated a one-time point annotation technique for gesture recognition systems in which labelers do not have to select the start and end time of each gesture carefully, but just mark a one-time point randomly within its temporal boundaries. The novel BoundarySearch algorithm was proposed to search for good boundaries of gestures based on gesture patterns around their one-time point annotations. The results show that the BoundarySearch algorithm can efficiently fix the boundaries for gestures and the gesture recognition system can use the fixed annotations to train gesture models. Their performance is just lower than the training on clean annotations by 3% F1-scores on average, but it reduces significantly the amount of annotation time. It can allow for example to annotate a video nearly in real time, instead of needing 10 hours for 30 minutes of data. Although our proposed method was investigated on gesture recognition with wearable sensors, the BoundarySearch algorithm is a generic al-

gorithm that can be useful in other applications of pattern recognition in which finding similar patterns inside two strings is necessary. In future work, we plan to test and deploy the one-time point annotation technique and apply the BoundarySearch algorithm within real-time annotation systems.

7.7 Acknowledgment

This work has been supported by the Swiss Hasler Foundation project Smart-DAYS.

8

Personalized Adaptation of Crowdsourced Models

Long-Van Nguyen-Dinh, Ulf Blanke, Gerhard Tröster

Towards Scalable Activity Recognition: Adapting Zero-Effort Crowdsourced Acoustic Models

Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia (MUM 2013), pp. 18:1–18:10, Luleå, Sweden, 2013.

10.1145/2541831.2541832

Abstract

Human activity recognition systems traditionally require a manual annotation of massive training data, which is laborious and non-scalable. An alternative approach is mining existing online crowd-sourced repositories for open-ended, free annotated training data. However, differences across data sources or in observed contexts prevent a crowd-sourced based model reaching user-dependent recognition rates.

To enhance the use of crowd-sourced data in activity recognition, we take an essential step forward by adapting a generic model based on crowd-sourced data to a personalized model. In this work, we investigate two adapting approaches: 1) a semi-supervised learning to combine crowd-sourced data and unlabeled user data, and 2) an active-learning to query the user for labeling samples where the crowd-sourced based model fails to recognize. We test our proposed approaches on 7 users using auditory modality on mobile phones with a total data of 14 days and up to 9 daily context classes. Experimental results indicate that the semi-supervised model can indeed improve the recognition accuracy up to 21% but is still significantly outperformed by a supervised model on user data. In the active learning scheme, the crowd-sourced model can reach the performance of the supervised model by requesting labels of 0.7% of user data only. Our work illustrates a promising first step towards an unobtrusive, efficient and open-ended context recognition system by adapting free online crowd-sourced data into a personalized model.

8.1 Introduction

Human activity recognition is important in developing context-aware systems. Being able to sense a user's routines (e.g. working in the office, staying at home) [104], his physical activities (e.g. standing, walking, cycling) [54], or his social context (e.g. having a conversation) [15], relevant information about the user can be captured. Based on the acquired information, actions can be executed. With the ubiquity of mobile phones, and their growing computational power and sensing capability, they enable new opportunities for developing personal context-aware systems in a large scale to perceive and act on what users are doing or experiencing. One example is the commercial application Google Now [1], which provides location-based reminders

[1] http://www.google.com/landing/now/

or arrival time estimates to reach a destination according to users' daily routines. Many more applications have been investigated in activity recognition research that span across a wide range of domains such as healthcare, sports, or entertainment that can profit from understanding the users' context [105]. Prominent examples include capturing daily life activities for health monitoring motivated by activities of daily living index (ADL) by Katz [106] and Bucks [107] or for the social rhythm measurement (SRM) [47]. Using context recognition systems activities can be detected automatically without burdening the user of keeping track of his activities.

Traditionally, most context recognition systems require a time-consuming preparation in which training data (e.g., sensor readings) is collected and manually annotated to build recognition models. Moreover, the daily life of a user often contains highly individual situations, activities, or environments. Also daily life situations can be expressed in a highly diverse way. For example, the – seemingly simple – activity of *working at the office* can consist of typing at the computer, reading, giving a talk, or attending a meeting. As a result, the recognition system has to be trained individually for a specific user and a large amount of training data has to be collected to cover the variability of the user-specific daily life. Clearly, this laborious and non-scalable requirement impede a real-world deployment in which a user can directly use the system, and prohibit the use in many applications such as in the healthcare-related scenarios mentioned above.

In this work we aim at facilitating the deployment of a context recognition system based on audio on mobile phones. To reduce the effort to collect training data, an alternative approach is to use crowd-sourced sound repositories (e.g., Freesound [2]) from the web [15]. The advantages of using web collected data are their free availability and a rich representation of possibly open-ended categories of sounds. However, crowd-sourced audio data can differ from data obtained on personal mobile phones. Differing characteristics of the user's surroundings or microphone responses can prevent the recognition when using a crowd-sourced model on user-specific data. In contrast, collecting labels of user data is a burden to every user. Table 8.1 shows the trade-off between crowd-sourced audio data and user-centric data recorded from user's mobile phone.

In this work, we take an essential step forward to combine the best properties from crowd-sourced data and user-centric data to obtain a

[2] www.freesound.org

Table 8.1: Comparison between crowd-sourced data and user-centric data

	Crowd-sourced Audio data	**User-Centric data (Mobile phone)**
Annotation Cost	Free	Huge effort (by users or experts)
Length	Short-clips (seconds/minutes)	Long continuous recording (days-months-years)
Location	Unknown, heterogeneous	User's environment surroundings/activities
Device	Unknown, heterogeneous	User's device

high performing and yet scalable recognition system in terms of user labeling effort. We achieve our goal by adapting a generic model based on crowd-sourced data to a personalized model. To this end, we first collect crowd-sourced training data to bootstrap a context recognition system (as in [15]). Then, we refine model parameters with no to little interaction of the user to improve the recognition performance. We contribute an analysis of different methods for the adaptation. In the first approach, a semi-supervised learning scheme is used to combine labeled crowd-sourced audio data with unlabeled user-centric data. In the second approach, we use an active-learning scheme to detect the most informative user-specific data samples that the crowd-sourced model can not represent well and queries a user to label them. We analyze the tradeoff between labeling effort and accuracy of the recognition system. We provide a thorough evaluation on 7 users with a total data amount of 14 days. The results show that combining crowd-sourced data with user-specific data can achieve accuracies similar to a supervised approach built on user data, but lowering the labeling effort to a minimum. Thus, leveraging both crowd-sourced data and user-centric data can open a chance to build a scalable and efficient context recognition system.

The rest of the paper is organized as follows. Section 8.2 offers the literature review on auditory context recognition. Section 8.3 proposes our recognition system to combine the two sources of audio data. In Section 8.4, we discuss the probabilistic learning framework for context recognition used in our paper. The collected datasets are presented in Section 8.5. The proposed research is examined by extensive evaluations in Sections 8.6 and Section 8.7. Section 8.8 concludes our work

and gives some potential research directions.

8.2 Related Work

Environmental sound has been used as a rich source of information to infer person's activities and locations [15,51,53,108–110]. For example, Stäger et al. [109] proposed a dedicated hardware to recognize a set of daily activities based on sound. Lu et al. [110] modeled and recognized sound events on mobile phones. While training data is essential for all these recognition systems, it is time-consuming and non-scalable to obtain sufficient amounts of data with annotations that represent daily life situations. Consequently, most of the previous work are limited to small datasets of sound daily life contexts that are manually collected and labeled under controlled conditions [51,52,108,109].

Although a relatively new idea, mining online multimedia repositories for relevant training data for activity recognition has been proposed by researchers to reduce the effort to collect and label training data as well as increase the number of available context classes. Perkowitz et al. [33] presented the web-based activity discovery using text. Rossi et al. [15] proposed to use the online crowd-sourced Freesound database to obtain a heterogeneous and diverse training data to train sound models to recognize activities of daily living.

Semi-supervised learning and active learning are two different types of techniques in machine learning that minimize the need of labeled training data. Those techniques are highly-motivated where unlabeled data can be easily obtained but annotation is costly or time-consuming to obtain, thus, labels sparse. Semi-supervised learning make use of both labeled and unlabeled data to train a recognition system. Meanwhile, active learning selectively asks labels of the most informative training instances that can generalize the classifier maximally, and thus reduces user's burden of labeling, but still gets good performance. There are many variations of semi-supervised learning and active learning algorithms. A comprehensive survey can be found in [34] for semi-supervised learning and in [35] for active learning, respectively.

According to the best of our knowledge, there is no previous work that investigated adaptation techniques to optimally leverage labeled crowd-sourced audio data and user-centric data recorded from mobile phones to improve the recognition performance but reduce the effort to label user's data. In the work by Rossi et al. [15], Freesound has

been used for context recognition with supervised learning. However, they do not consider user adaptation to improve the performance. Zhang et al. [36] used semi-supervised learning to improve sound event classification. In their work, however, labeled and unlabeled data are of the same data source and they did not work with personalized user context. Stikic et al [37] explored both semi-supervised learning and active learning in physical activity recognition, with focus only on user-centric data record in a highly instrumented home environment. In contrast to these works, we aim to improve the recognition of a classifier learned from one free data source – crowd-sourced repository – on another, the user personalized data on mobile phone.

8.3 Context Recognition System

While the web offers an abundance of labeled data, obtaining labels from a single user is often a tedious and time consuming task. However, with the option of obtaining an abundance of unlabeled data from the user, we employ techniques to optimally use available data. Figure 8.1 shows an overview of our sound-based context recognition system. In data preprocessing phase, we collect auditory training data from Freesound and user's mobile phone. We then extract acoustic features from the collected audio clips. In the learning phase, we apply machine learning techniques to learn and adapt a context recognition model based on the two sources of data. In the recognition phase, the context recognition model will be used to infer user context from data recorded on user's mobile phone. We describe each component in our proposed system in the following.

8.3.1 Data Preprocessing

Freesound Repository. Freesound [111] is an online sharing repository of crowd contributed sound data. Sounds in Freesound are contributed by a very active online community and thus, the number of available sounds has increased rapidly. Currently, the database stores about 170000 samples uploaded by 6000 contributors. Sounds are often annotated in free-form styles and the tags come from very diverse vocabularies. Moreover, crowd-contributed sounds are recorded in a wide variety of situations, conditions, motivations, and skills.

Crawling labeled audio from Freesound. In our system, we focus on everyday situations such as *dining in a restaurant* or *transporting* For

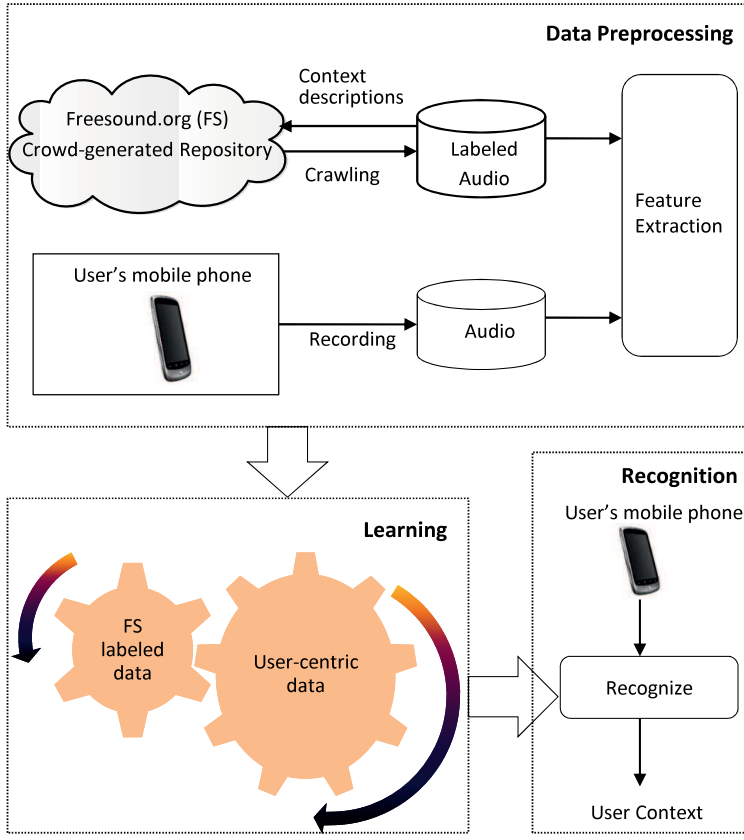

Figure 8.1: The data processing flow of our sound-based context recognition that combines Freesound data and user-centric audio data

each context class, we use its name as a keyword (e.g., "restaurant") to search for the sound clips in Freesound that are tagged with the keyword. The list of context classes can be provided by a user who uses the context recognition system. In health monitoring systems, the list of context classes is usually defined by a specialist beforehand [47]. However, with the diversity of context classes in the crowd-sourced repository, it is easy to extend the recognition system by specifying new context classes and then extracting training data samples for those classes from the crowd-sourced repository. We retrieve only sound clips with

the highest average rating (i.e., high quality) given to the sounds provided by the Freesound community. Multiple sound samples are then labeled with the corresponding context class. All the retrieved audio samples were converted to WAV format with a sampling frequency of 16 kHz and bit depth of 16 bits/sample. We manually filter the downloaded audio clips that are irrelevant to the assigned context class. For automatic filtering techniques see [15]. We do not apply an automatic filtering to remove irrelevant clips because the manual filtering showed the best results [15] and we assume that the small set of short audio clips that we retrieved from Freesound (30 sound clips per context) can be quickly and cheaply filtered by listening.

User Recordings. We record continuously audio data from users' smartphones with a sampling frequency of 16 kHz and bit depth of 16 bits/sample. As with the data from freesound, we store in WAV format.

Extracting audio features. We extract 12 coefficients mel-frequency cepstral coefficient (MFCC) and log-energy on a sliding window of 32 ms length of audio data. The same method are used to extract acoustic features for both audio data from Freesound and the smart phones.

8.3.2 Learning Phase

We propose to use semi-supervised learning and active learning schemes based on Gaussian Mixture Model (GMM) to combine two data sources: Freesound and user-centric data on mobile phones. We name two approaches as *Semi-supervised Adaptation* and *Active Learning Adaptation* respectively.

Semi-supervised Adaptation. Semi-supervised learning is used to combine Freesound labeled data and user unlabeled data.

Active Learning (AL) Adaptation. We train a bootstrapped context classifier using Freesound labeled data. From that initial classifier, active learning proceeds and iteratively selects the most informative user-centric samples to query for labels. The classifier is then retrained and adapted with the new user-centric labeled data.

Semi-supervised Adaptation and *AL Adaptation* are illustrated in Figure 8.2. Details of the semi-supervised learning and active learning algorithms are discussed in Section 8.4.

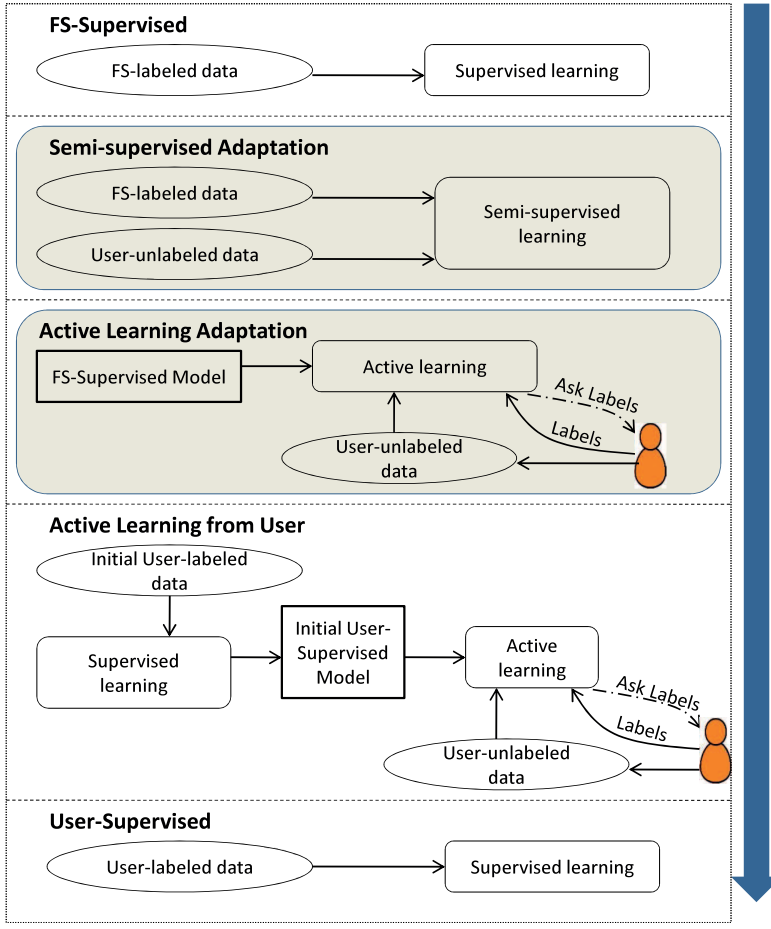

Figure 8.2: Learning approaches using either Freesound (FS) or user-centric data, or both of them to train the context recognition

8.3.3 Recognition Phase

The GMM models which are obtained from the learning phase are used to recognize user daily contexts based on audio data recorded from the smartphone. We construct a two-level classification. At the low level, audio instances extracted from windows of 32 ms are classified by the GMM models. The context class with the highest probability to generate an instance is assigned to that instance. At the high level, a decision is made on the longer segment (2 seconds) by taking a class with the highest frequency in the segment as a label.

8.4 Probabilistic Framework for Context Recognition

As we mentioned before, immediate usage of crowd-sourced data to build a context recognition model is suboptimal due to lack of user-specific training data, therefore we propose two adaptation techniques based on GMM to tailor a context model build from crowd-sourced data to a personalized context model. In this section, we first briefly present the GMM probabilistic framework for context recognition used in this paper. Then the semi-supervised and active learning algorithms built on this framework are presented.

GMM is an effective generative classifier that has been used extensively in acoustic domains (e.g., speaker recognition [49,50], environmental sound [51–53]). Let $\mathcal{D}$ be a set of N observed instances $\mathbf{x}_i \in \mathbb{R}^d$ and Ω be a Gaussian mixture model with K components, $c_1, ..., c_K$. Each component c_k (k = 1,...,K) is a Gaussian density conditional model, i.e., $p(\mathbf{x_i}|c_k) = \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)$, where $\boldsymbol{\mu}_k$ and Σ_k are the mean vector and covariance matrix of the component, respectively. Let us also denote Θ be the set of parameters of the model Ω, $\Theta = \{\boldsymbol{\mu}_k, \Sigma_k, \pi_k\}_{k=1}^{K}$, where π_k is the prior probability of the component c_k.

Given the data $\mathcal{D}$, the maximum log likelihood estimation (MLE) is used as a criteria to define the best model $\widehat{\Theta}$ to fit $\mathcal{D}$, i.e., $\widehat{\Theta} = \underset{\Theta}{\text{argmax}}$ log $p(\mathcal{D}|\Theta)$, with

$$\mathcal{L} = \log \mathrm{p}(\mathcal{D}|\Theta) = \log \prod_{i=1}^{N} p(\mathbf{x}_i|\Theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)$$

In our work, each context class can contain multiple Gaussian components. The probability that an instance x_i belonging to a context class y_i is computed as the sum of the probabilities of the mixture

components belonging to the context class to generate the instance.

$$P(y_i|x_i;\Theta) = \frac{\sum_{j=1}^{K} \mathbb{1}\{c_j, y_i\} P(c_j|\mathbf{x}_i;\Theta)}{\sum_{k=1}^{K} P(c_k|\mathbf{x}_i;\Theta)},$$

with $\mathbb{1}\{c_j, y_i\} = 1$ if class y_i contains component c_j.

8.4.1 Semi-supervised Learning using EM

The goal is to find semi-supervised parameters Θ that maximize $\mathcal{L}$ to fit the labeled (i.e., Freesound data) and unlabeled (i.e., user-centric data) observations. The Expectation-Maximization (EM) approach [112] is a standard procedure to find the locally optimal parameter set $\widehat{\Theta}$. In our work, we consider that each context class can have multiple Gaussian components.

Inputs: Collections X^l of labeled data of l instances and X^u of unlabeled data of u instances. The training set $X = X^l \cup X^u$.

1. Initialization. For each class i, build a GMM model $\widehat{\Theta}_i$ from the labeled data X_i^l of that class. Merge all components of classes to have initial Θ.

2. Loop until converge. (i.e., the change in log likelihood of the training data X is less than 10^{-4}):

E-step: Use the current model to estimate the probability that each mixture component generated each instance (i.e., component membership).

$$\gamma_{ij} = P(c_j|\mathbf{x}_i;\widehat{\Theta}) = \frac{\pi_j \mathcal{N}(\mathbf{x}_i;\boldsymbol{\mu}_j,\Sigma_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)}$$

Restrict the membership probability estimates of labeled instances to be zero for components associated with other classes, and renormalize.

M-step: Re-estimate the GMM model, $\widehat{\Theta}$, given the estimated component membership of all labeled and unlabeled instances.

$$l_j = \sum_{i=1}^{l+u} \gamma_{ij}, \quad \pi_j = \frac{l_j}{l+u}, \quad \boldsymbol{\mu}_j = \frac{1}{l_j} \sum_{i=1}^{l+u} \gamma_{ij} \mathbf{x}_i$$

$$\Sigma_j = \frac{1}{l_j} \sum_{i=1}^{l+u} \gamma_{ij} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T$$

for all j = 1,...,K

8.4.2 Active Learning

Active learning starts with an initial learner trained on a small number of labeled instances in the training set with supervised GMM. The learner then iteratively queries labels for one or more selected instances in the unlabeled training set, learns from the new labeled set, and then updates the learner. In the *Active Learning Adaptation* approach, the initial learner is built on the Freesound labeled data and the unlabeled training set is user-centric data.

In the context recognition on mobile phones, one can imagine an interactive online strategy that asks the user to annotate his current context when the learner confuses how to label the current situation. It is called stream-based active learning [35]. On the other hand, in pool-based active learning [35], the query decisions are made offline after collecting the entire unlabeled training set. The learner evaluates and ranks the entire unlabeled set to select to the best queries. In the auditory context recognition, one can imagine that the audio segments corresponding to the query instances are extracted and given to the user to annotate them.

We use a pool-based algorithm to query labels on the user-centric unlabeled dataset. Specifically, we use entropy [113] as an uncertainty measure to find the most informative unlabeled instance to query a label.

$x^*_{ENT} = \underset{x_i}{\operatorname{argmax}} - \sum_i P(y_i|x_i;\Theta) \log P(y_i|x_i;\Theta)$, where y_i ranges over all possible class labels.

In our work, we use a classification window of 32 ms (see Section 8.3.1). However, everyday context classes defined in this paper often last for at least a few minutes. Therefore, instead of asking the label for the 32 ms data segment only, we extend the labeled segment with a window of one minute around the queried instance.

8.5 Datasets

For our evaluation we collect two datasets: 1) To obtain a user-centric dataset, we collect data recorded from users' smartphones; 2) For the

crowd-sourced dataset we make use of the *Freesound* repository as in [15,71].

User-centric data. We use android-based smartphones (Samsung Galaxy S2) with headset microphones for continuous sound recording. Participants were asked to record two full working days in their ordinary setting. All participants live in Zurich, Switzerland and thus, the transportation includes tram, train, bus and car. The recording application also provides an annotation tool in which user can annotate his current contexts as a ground truth. Specifically, users can indicate when a context class starts/stops happening. We do not ask the user to label fine-grained sound events, but longer lasting everyday contextual situations. In our work, we also want to support the recognition of individual and user-dependent context classes. Therefore, users can annotate different set of context classes, individual to their daily situations. Table 8.2 shows the list of classes provided by 7 participants and the corresponding distribution of classes in the dataset recorded by users on mobile phones. Context classes are about working, feeding, transportation and social interaction which are useful in health monitoring [47]. For each recording day, at least 9 hours of audio data were obtained for each user. As can be seen in Table 8.2, users spend most of the time in the office and discuss their works with colleagues. In total, about 130 hours of audio data has been collected from mobile phones for the study.

Freesound. From the list of context classes provided by the users, we retrieve audio data for those context classes from Freesound. As a result, we download sound clips for 9 context classes from Freesound as shown in Table 8.3. For each class, we retrieve 30 sound clips, tagged with the label of the class, with the highest average rating given to the sounds. Besides the class label, a sound clip also has other tags that usually describe different sound events occurring in the sound clip. Table 8.3 shows the subset of tags in Freesound clips that we download for each context class. As can be seen, each context class contains the heterogeneity of sound events and recording conditions. For example, being in the office consists of multiple sound events such as typing, stapling, printing, etc. After manually filtering for quality, we have 163 audio clips (143 minutes) for 9 context class to train sound models. This data from Freesound is denoted as FS.

Table 8.2: User-dependent context classes and the corresponding distribution of classes in dataset

	Context Classes and Class Distribution (%)
User 1	office (83), tram (1), train (10), conversation (6)
User 2	toilet (1), office (50), restaurant (5), street (1), conversation (43)
User 3	office (37), restaurant(7), street(12), tram(2), conversation(42)
User 4	toilet (1), office (70), restaurant(2), street (4), tram (1), conversation (22)
User 5	toilet (1), office (63), restaurant (7), street (7), tram (1), train (7), conversation (14)
User 6	toilet (0.4), office(70), restaurant (8), street (4), tram (6), train (5), car (1), conversation (5.6)
User 7	toilet (0.2), office (21), restaurant (9), street (4), tram (5), train (6), car (2), bus (0.2), conversation (52.6)

Table 8.3: The heterogeneity of sounds from freesound for each context class

Context Class	Tags of Freesound Clips
Office	office, door-open, typing, locking, coffee-machine, stapler, paper-shuffling, print
Bus	bus, door-open, horn, footstep, air-brake, stop, speeding, air-pressure-release
Car	car, highway, forest, car-door, overtake, start, stop, footstep, brake, snow, rain
Train	train, rail, leaving, accelerating, wheels, door, railway, underground, passing, voice
Tram	tram, door, trolley, passing, beep, creaking, tunnel, bell, announcement, brake
Street	street, pedaling, chatter, people, music, bike, announcement, foot, bell, car, horse
Restaurant	restaurant, chat, drink, eat, pour, liquid, food, ice, dish, nibble, grill, clinking, music
Toilet	toilet, splash, water, scrub, lavatory, sink, shower, brush, urinal, flush, hand-dryer
Conversation	chat, talk, noise, bustle, phone, scream, yell, panic, male, female, English, Spanish

8.6 Evaluation

To evaluate the best use of crowd-sourced data in personalized context recognition, we address the following research questions:

1. Does the *Semi-supervised Adaptation* of a crowd-sourced model improve the recognition of user's contexts on mobile phones?
2. Can the *Active Learning Adaptation* find user data instances that the crowd-sourced data can not represent well to ask for labels and quickly achieve good performance with minimal number of label queries?
3. Does the *Active Learning Adaptation* of a crowd-sourced model perform better than active learning model based on user data only in terms of accuracy and number of label queries?

To answer these question, we compare our proposed approaches with three baseline non-adapted learning approaches. Specifically,

- *FS-Supervised*: A supervised GMM trained on Freesound data only without adaptation with user-collected data.
- *User-Supervised*: A supervised GMM trained on the user-centric annotated training data only.
- *Active Learning (AL) from User*: An active learning scheme on user-centric data only. In this baseline approach, we assume that initially each user can contribute and label randomly one minute of data ($\approx$ 1875 instances) for each context class. Totally, this labeled data takes about 1% of training data. An initial GMM classifier is trained from that labeled data, then the active learning is applied to query labels for the uncertain samples.

Figure 8.2 illustrates the three baseline approaches also with our proposed approaches. They are ordered increasingly in the effort of user data annotation (i.e., (1) *FS-Supervised*, (2) *Semi-supervised Adaptation*, (3) *AL Adaptation*, (4) *AL from User*, and (5) *User-Supervised*). Among them, *FS-Supervised* and *Semi-supervised* do not require any effort to label user data. It is not clear now whether *AL Adaptation* is better than *AL from User* in terms of number of label queries. We will discuss it in detail in the next section. Our goal is to find the best approach in terms of accuracy and labeling effort.

The experiments are performed based on the partitioning of the two-day recording audio data from user's mobile phone into two halves. The first fold (F1, 50% of user data for each class) is used with no labels ($F1_U$) or all labels ($F1_L$) or a small part of these labels ($F1_{1\%L}$ and $F1_{99\%U}$) in training phase. The second fold (F2, another 50% of user data for each class) is used for testing in recognition phase for all five approaches. Table 8.4 shows the usage of two sources of data in learning phase of five approaches.

Table 8.4: The usage of two sources of data in learning phase of five approaches

	Data usage
FS-Supervised	FS
Semi-supervised Adaptation	FS + $F1_U$
Active Learning Adaptation	FS + $F1_U$
Active Learning from User	$F1_{1\%L}$ + $F1_{99\%U}$
User-Supervised	$F1_L$

8.7 Results

Table 8.5 gives the accuracy of five approaches (we only show the best for the active learning). Figure 8.3 plots the detailed performance of the active learning approaches over the first 20 label queries.

Table 8.5: Accuracy of learning approaches for 7 users. For active learning (AL), the best performances over 20 label queries are given.

	FS-Supervised	Semi-supervised Adaptation	AL Adaptation	AL from User	User-Supervised
User 1	0.8	0.86	**0.97**	0.93	0.94
User 2	0.5	0.65	**0.94**	0.82	0.9
User 3	0.58	0.43	**0.81**	0.73	0.72
User 4	0.22	0.25	**0.93**	0.86	0.72
User 5	0.35	0.5	0.80	**0.83**	0.82
User 6	0.54	0.61	0.86	**0.87**	0.85
User 7	0.26	0.47	**0.86**	0.76	0.83

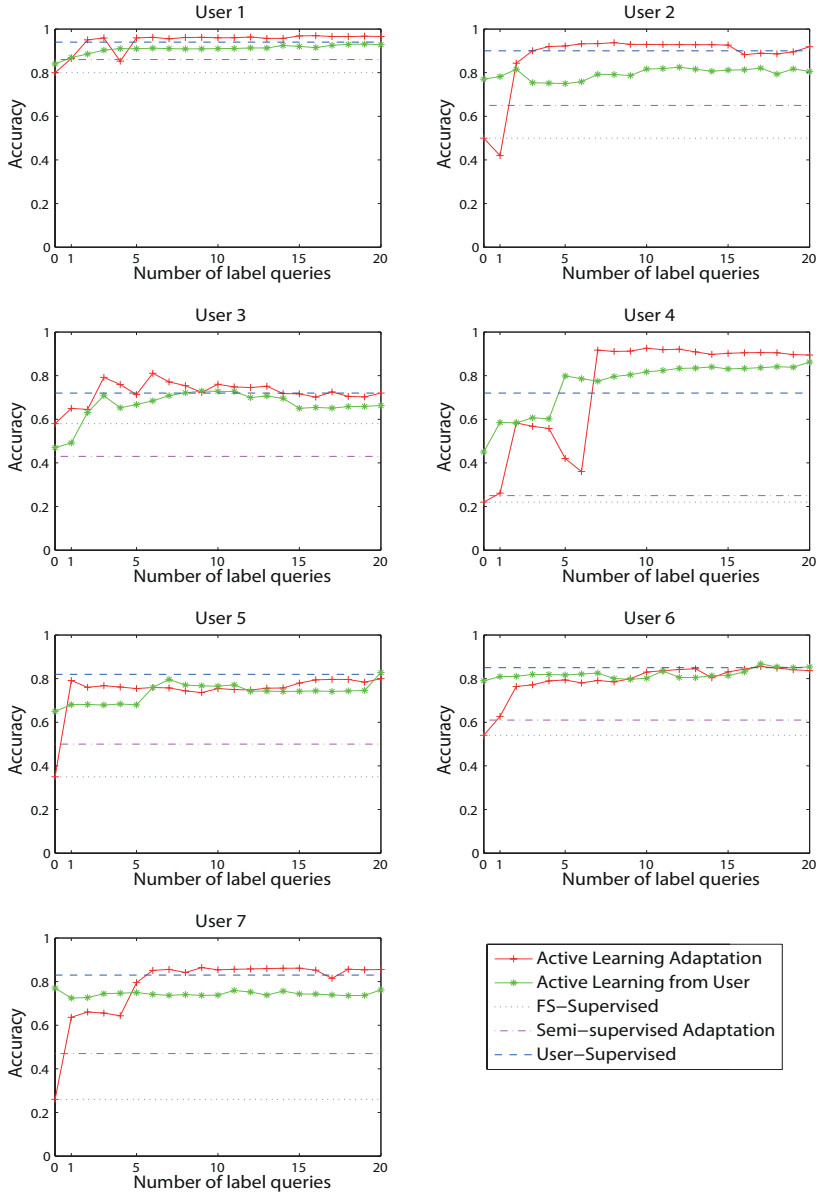

Figure 8.3: Performance of the active learning approaches over the number of label queries for each user

As expected, the *User-Supervised* approach gives much better results than the *FS-Supervised* approach. Supervised training on user data clearly captures user-specific environments in the model (i.e., test and training data tend to be similar for each single user) and thus, recognizes well user context in daily routines. The performance of the *FS-Supervised* model drops significantly since the crowd-sourced data hardly covers all user-specific surroundings. However, *FS-Supervised* model does not require any effort to label user data, meanwhile *User-Supervised* requires huge effort to label them. Note that the accuracy of the supervised Freesound model *FS-Supervised* is at similar range to that reported in the work by Rossi [15] which also used Freesound data to recognize users' contexts.

Semi-supervised Adaptation The results show that the *Semi-supervised Adaptation* approach significantly improves the performance of context recognition compared to the non-adapted *FS-Supervised* (up to 21%) for six users. Unlabeled training data from user can help adapt the crowd-sourced model to the personalized model without asking any labels for user data. Only for user 3, the *Semi-supervised Adaptation* actually decreases the performance. In this case, the contribution of unlabeled user data in the semi-supervised learning makes the model more uncertain. The result in Table 8.5 also shows that *Semi-supervised Adaptation* underperforms significantly the baseline *User-Supervised* approach as can be seen in Table 8.5. Especially from user 4, even the *Semi-supervised Adaptation* can increase the accuracy of *FS-Supervised* by 3%, the accuracies of these two approaches are much lower than that of the *User-Supervised*. Here the Freesound data does not generalize sufficiently to the data recording from that user-specific environments. The visualization of the collected data from the crowd-sourced data and user-centric data in Figure 8.4 supports our explanation (data dimension reduced to 2 using t-SNE stochastic neighborhood embedding method [114]). As you can see, the crowd-sourced data represents some parts of the user data only.

Active Learning Adaptation As can be seen in Figure 8.3, the *AL Adaptation* quickly reaches the performance of the *User-Supervised* approach and even improves further the accuracy over 20 label queries. Freesound contains rich information about the contexts and after asking a few label queries for user data instances that Freesound can not represent well, the *AL Adaptation* approach may generalize better user's

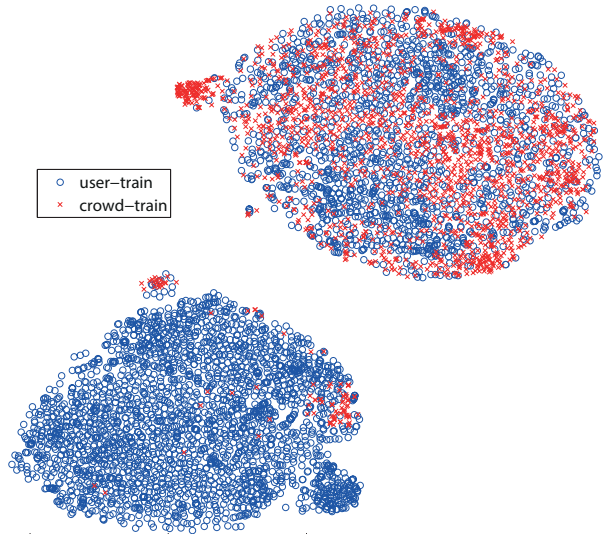

Figure 8.4: A visualization of the collected data from crowd-sourced Freesound and one user's mobile phone data of *train* context class

context. After 20 iterative queries of labels, the *AL from User* approach also gets similar or closely results as of the *User-Supervised* learning. Thus, the active learning technique generally can achieve high accuracy using significantly fewer labeled training data (20 queries $\approx$ 3% of the user training data). In Figure 8.3, we see that the *AL from User* gets better performance than the *AL Adaptation* in initial number of label queries. However, with more queries eventually, the *AL Adaptation* achieves better accuracies. We can explain this with the same reason: Freesound contains intra-class diversity, thus it may contain user's unseen contexts in the recognition phase and increase model generalization.

To compare the *AL Adaptation* and *AL from User* in terms of number of label queries, we evaluate how many label queries needed for these two approaches to reach the same performance as the *User-Supervised* approach. For the *AL from User* approach, we also count the annotation effort of user to contribute the initial labeled training set to build the initial classifier (This effort is equivalent to the total number of context class that the user has). As can be seen, the *AL Adaptation* requires

much less number of queries than the *AL from User* to achieve the same accuracy as the *User-Supervised* approach. It only requires in average 5 label queries per user (≈ 0.7% of user training data) to get the good performance. Meanwhile, the *AL from User* asks for in average at least 24 label queries per user (≈ 3.6% of user training data). Moreover, the number of queries needed in *AL Adaptation* is even much less than the total number of context classes that the user has for most of the user. The Freesound data contains training instances of several classes that describe well user contexts and thus, the *AL Adaptation* does not require to ask labeled instances for those classes.

Table 8.6: Number of label queries needed in two active learning (AL) approaches to reach the performance of *User-Supervised* approach

	AL Adaptation	*AL from User*
User 1	2	23
User 2	3	> 50
User 3	3	8
User 4	7	11
User 5	1	14
User 6	10	24
User 7	6	39

Note that each query requests labels of one minute of data (≈ 1875 data instances)

To analyze in detail which classes for the Freesound-sourced model performs poorly, we investigate the queries of the *AL Adaptation*. We want to reveal which classes are queried for and how much the obtained label improves the classification. For each user, over 20 label queries, we report how many queries performed for each context class y as well as the accumulated improvement of the context class $A(y)$ that these queries contribute to the performance. Specifically,

$A(y) = \sum_{t=1}^{20} \mathbb{1}\{t, y\}(accuracy(t) - accuracy(t-1))$, with $\mathbb{1}\{t, y\} = 1$ if an instance of class y is enquired for a label in the query at time t and accuracy(t) is the accuracy of *AL Adaptation* approach after the t-th query is asked.

Table 8.7 shows the accuracy improvement by *AL Adaptation* over 20 queries. All 7 users need to ask a few queries for office class. Except for

Table 8.7: The accuracy improvement by Active Learning from FS over 20 label queries

	User 1	User 2	User 3	User 4	User 5	User 6	User 7
office	(10, 20%)	(8, 48%)	(5, 27%)	(2, 56%)	(5, 47%)	(2, 1%)	(1, 38%)
restaurant	–	(0, 0)	(0, 0)	(0, 0)	(1, -3%)	(1, 2%)	(0, 0)
street	–	(2, -1%)	(5, -12%)	(4, 30%)	(9, 3%)	(3, 16%)	(7, 0.5%)
tram	(0, 0)	–	(0, 0)	(0, 0)	(0, 0)	(4, 7%)	(2, 1%)
train	(6, 8%)	–	–	–	(0, 0)	(6, -0.3%)	(5, 2%)
car	–	–	–	–	–	(0, 0)	(0, 0)
bus	–	–	–	–	–	–	(1, 15%)
toilet	–	(0, 0)	–	(2, 3%)	(0, 0)	(3, 3%)	(0, 0)
conversation	(4, -11%)	(10, -5%)	(10, -0.9%)	(12, -21%)	(5, -2%)	(1, 0.3%)	(4, 3%)

The first number in parenthesis is the total number of queries requested for the corresponding context class and the second number denotes the accumulated accuracy improvement A. The (0,0) denotes that AL from FS does not acquire a label for any instances from the corresponding context class. A dash – denotes that the user does not have the context class in his ADL.

user 1 for which only little improvement is achieved, the queries significantly improve the recognition performance. Meanwhile, for user 1, queries for street class are executed, which increases the performance significantly. It seems like office context contains a heterogeneous mixture of sound events and some of them are highly user-dependent. Remember that for user 4, the *FS-Supervised* yields a very bad performance (only 22% accuracy) as can be seen in Table 8.5 and Figure 8.3. However, after 2 queries of office class and 4 queries of street class, the performance is dramatically increased. There are several context classes that the Freesound-based model characterizes well in user-specific data. Thus the *AL Adaptation* does not need to query any labels from those classes. For example, for user 4, restaurant context and tram context are not enquired for labels (i.e., (0,0) in Table 8.7). To enforce this analysis, we show the confusion matrix of *FS-Supervised* and *AL Adaptation* for user 4 in Figure 8.5 . As can be seen, the *FS-Supervised* can recognize well restaurant and tram classes, and it confuses office with toilet, and street with tram. That is why the *AL Adaptation* needs to ask labels for office and street classes, but not for restaurant and tram. Therefore, it can be emphasized again that Freesound contains diverse, useful acoustic data that can be used to recognize user context.

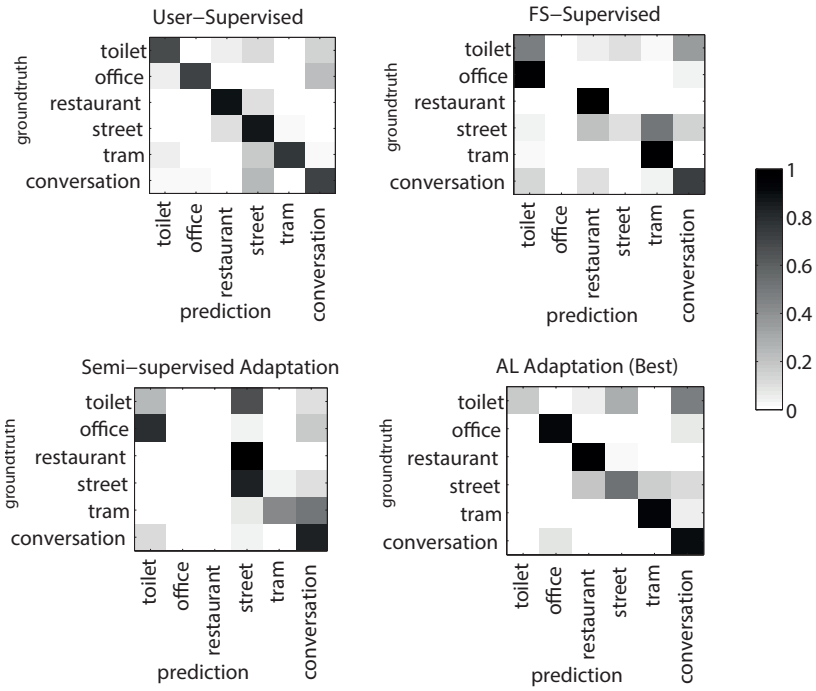

Figure 8.5: Confusion matrix tables of learning approaches for user 4

Discussion Our proposed *Semi-supervised Adaptation* can improve the performance up to 21% from the *FS- Supervised*. However, *Semi-supervised Adaptation* can be very greedy in leveraging the unlabeled data, thus it can mislead and decrease the performance. One solution for this is to lower the emphasis of the unlabeled data by adding a positive weight $\lambda \leq 1$ to the semi-supervised log likelihood [115]. More importantly, it is still far from reaching the performance of the *User-Supervised* baseline approach. While crowd-sourced data indeed helps context recognition, it may not cover exact user-specific context characteristics and all user-specific situations. Thus, both data sources are probably too dissimilar to profit from the unlabeled data from the user data source. *AL Adaptation* can enquire labels for training instances of only user-specific context scenes that the Freesound data can not represent well, and the method can quickly reach the performance of

User-Supervised approach after only a few queries. Interestingly, the *AL Adaptation* can outperform the baseline *User-Supervised* because the *AL Adaptation* can take the advantages of Freesound diversity and variability. Furthermore, *AL Adaptation* is much better than *AL from User* in terms of the number of queries needed to reach the performance of *User-Supervised*, thus the *AL Adaptation* can leverage well both sources of data: Freesound and user- centric data to get a very good accuracy, but reduce significantly user effort to annotate data.

From our evaluations, we state the lessons learned from this work. Context recognition systems which use traditional supervised learning on user training data perform the best, but it requires a huge effort to label a sufficient amount of user training data. The crowd-sourced repositories provide free labeled training data, however the performance is till far from reaching the performance of supervised learning on user-specific data. The semi-supervised learning to combine labeled crowd-sourced data and unlabeled user data is one of the cheapest ways to adapt the system without asking any effort to label user data. It can improve the performance but can not reach the best result. With a few label queries only from user training data, the active learning based on crowd-sourced data can perform a significant improvement and reach the supervised performance with only 0.7% of user data. Hereby the recognition system built on crowd-sourced data can flexibly learn a new class by first extracting training samples for that class from crowd-sourced repositories, and successively improve its performance with another few user queries by using active learning. Therefore, our best recommendation for personal context recognition is to use active learning with crowd sourced data for optimal and efficient learning. Our proposed adaptation techniques which leverages crowd-sourced data can also open a new chance to develop a scalable and open-ended context recognition system.

8.8 Conclusion and Future Work

In this paper, we conducted experiments that combine and leverage complementary properties of two sources of data: the crowd-sourced labeled audio dataset and the user-centric audio recorded from mobile phones, to recognize user daily context. We investigated semi-supervised learning and active learning to adapt a generic model built from crowd-sourced data to a personalize context model. The semi-supervised learning can improve the recognition accuracy up to 21%,

thus, the semi-supervised learning can be used to adapt user-centric data from the crowd-sourced data to build a better context recognition without asking labeling on user data. However, it still underperforms significantly the user supervised model build on user-centric data. The active learning approach that is based on the crowd-sourced labeled data can reach the performance of the supervised model quickly with surprisingly only a few label queries for the user data (in average 5 queries corresponding to 0.7% of the user training data). Furthermore, the active learning approach can even outperform the user supervised model as it leverages diversity and variability of existing crowd-sourced data. Our recommendation for personal context recognition is to use active learning with crowd sourced data for optimal and efficient learning in terms of accuracy and labeling effort. The rich availability of crowd-sourced data in terms of number of classes also open new opportunities to develop open-ended, scalable activity recognition system. In future work, we plan to analyze the influence of unlabeled data in the semi-supervised learning approach by varying their emphasis. We also plan to try stream-based active learning schemes to support interactive online strategy to get annotation from user on mobile phones. Furthermore, we also need to test the algorithms with more number of users and more context classes.

8.9 Acknowledgment

The authors would like to thank Mirco Rossi (ETH Zurich) for useful comments. This work has been supported by the Swiss Hasler Foundation project Smart-DAYS.

Glossary

Notation	Description
1D	one-dimensional
3D	three-dimensional
ADL	Activities of Daily Living
AMT	Amazon Mechanical Turk
Avg	Average
DTW	Dynamic Time Warping
HCI	Human Computer Interaction
HIT	Human Intelligence Task
HMM	Hidden Markov Model
IMU	Inertial Measurement Unit
LCSS	Longest Common Subsequence
LM	Local Maximum
GMM	Gaussian Mixture Model
GPS	Global Positioning System
GT	Ground Truth
kNN	k-Nearest-Neighbor
MFCC	Mel-Frequency Cepstral Coefficient
RBF	Radial Basis Function Kernel

RFID	Radio-Frequency Identification
SAX	Symbolic Aggregate approXimation
SRM	Social Rhythm Measurement
Stdev	Standard deviation
SVM	Support Vector Machine
TM	Template Matching
TMM	Template Matching Method
WAV	WAVeform audio format

Bibliography

[1] A. Salarian, H. Russmann, F. Vingerhoets, P. Burkhard, and K. Aminian, "Ambulatory monitoring of physical activities in patients with parkinson's disease," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 12, pp. 2296 – 2299, 2007.

[2] 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, 2008.

[3] H.-K. Lee and J. H. Kim, "An hmm-based threshold model approach for gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 961–973, Oct. 1999.

[4] 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.

[5] 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, 2008.

[6] 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.

[7] 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.

[8] 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.

[9] 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)*, pp. 1541–1546, AAAI Press, 2005.

[10] 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)*, pp. 217–220, Nov 2008.

[11] 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, pp. 25–38, 2009.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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*, vol. 33, pp. 2521–2537, Dec. 2011.

[17] J. Howe, "The Rise of Crowdsourcing." (accessed July 20, 2010), jun 2006.

[18] M.-C. Yuen, I. King, and K.-S. Leung, "A survey of crowdsourcing systems," in *SocialCom/PASSAT*, pp. 766–773, 2011.

[19] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pp. 254–263, 2008.

[20] P.-Y. Hsueh, P. Melville, and V. Sindhwani, "Data quality from crowdsourcing: a study of annotation selection criteria," in *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, HLT '09, pp. 27–35, 2009.

[21] A. Wang, C. D. V. Hoang, and M.-Y. Kan, "Perspectives on crowdsourcing annotations for natural language processing," 2010.

[22] G. Parent and M. Eskenazi, "Toward better crowdsourced transcription: Transcription of a year of the let's go bus information system data," in *SLT*, pp. 312–317, 2010.

[23] G. Parent and M. Eskenazi, "Speaking to the crowd: Looking at past achievements in using crowdsourcing for speech and predicting future challenges," in *INTERSPEECH*, pp. 3037–3040, 2011.

[24] S. Nowak and S. Rüger, "How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation," in *Proceedings of the international conference on Multimedia information retrieval*, MIR '10, (New York, NY, USA), pp. 557–566, ACM, 2010.

[25] L. Zhao, G. Sukthankar, and R. Sukthankar, "Incremental relabeling for active learning with noisy crowdsourced annotations," in *SocialCom/PASSAT*, pp. 728–733, 2011.

[26] C. Vondrick and D. Ramanan, "Video annotation and tracking with active learning," in *NIPS*, pp. 28–36, 2011.

[27] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *International Journal of Computer Vision*, pp. 1–21, 2012.

[28] 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.

[29] D. Angluin and P. Laird, "Learning from noisy examples," *Machine Learning*, vol. 2, pp. 343–370, April 1988.

[30] R. Amini and P. Gallinari, "Semi-supervised learning with an imperfect supervisor," *Knowledge and Information Systems*, vol. 8, pp. 385–413, November 2005.

[31] 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*, vol. 4087, 2006.

[32] 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, pp. 306–313, 2001.

[33] M. Perkowitz, M. Philipose, K. Fishkin, and D. J. Patterson, "Mining models of human activities from the web," in *Proceedings of the 13th international conference on World Wide Web*, WWW '04, 2004.

[34] X. Zhu, "Semi-supervised learning literature survey," tech. rep., Computer Sciences, University of Wisconsin-Madison, 2005.

[35] B. Settles, "Active learning literature survey," computer sciences technical report, University of Wisconsin–Madison, 2010.

[36] Z. Zhang and B. Schuller, "Semi-supervised learning helps in sound event classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.

[37] M. Stikic, K. Van Laerhoven, and B. Schiele, "Exploring semi-supervised and active learning for activity recognition," in *Proceedings of the 12th IEEE International Symposium on Wearable Computers (ISWC)*, pp. 81–88, 2008.

[38] "http://kitchen.cs.cmu.edu/,"

[39] 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, pp. 453–456, 2008.

[40] M. Soleymani and M. Larson, "Crowdsourcing for affective annotation of video: development of a viewer-reported boredom corpus," in *33th ACM SIGIR, Workshop on Crowdsourcing for Search Evaluation*, 2010.

[41] G. G. K. J. Richard Landis, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.

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

[43] 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, pp. 17–33, 2008.

[44] 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)*, pp. 92–99, 2012.

[45] T. van Kasteren and B. Krose, "Bayesian activity recognition in residence for elders," in *Proceedings of the 3rd International Conference on Intelligent Environments*, 2007.

[46] B. Malhotra, I. Nikolaidis, and M. Nascimento, "Distributed and efficient classifiers for wireless audio-sensor networks," in *Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on*, 2008.

[47] T. H. Monk, E. Frank, J. M. Potts, and D. J. Kupfer, "A simple way to measure daily lifestyle regularity," in *European Sleep Research Society*, 2002.

[48] F. Font and X. Serra, "Analysis of the folksonomy of freesound," in *2nd CompMusic Workshop*, (Istanbul, Turkey), pp. 48–54, 2012.

[49] R. Rose and D. Reynolds, "Text independent speaker identification using automatic acoustic segmentation," in *Interna-*

tional Conference on Acoustics, Speech, and Signal Processing, 1990, pp. 293–296 vol.1, 1990.

[50] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," in *Digital Signal Processing*, p. 2000, 2000.

[51] S. Chu, S. Narayanan, and C.-C. J. Kuo, "Environmental sound recognition with time-frequency audio features," *Trans. Audio, Speech and Lang. Proc.*, vol. 17, pp. 1142–1158, Aug. 2009.

[52] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa, "Computational auditory scene recognition," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, may 2002.

[53] J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music.," *J Acoust Soc Am*, vol. 122, no. 2, p. 881, 2007.

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

[55] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys*, vol. 43, pp. 16:1–16:43, Apr. 2011.

[56] 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.

[57] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "Crowddb: answering queries with crowdsourcing," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD '11, pp. 61–72, 2011.

[58] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the EM algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 20–28, 1979.

[59] 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*, vol. 11, 2010.

[60] 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, pp. 64–67, 2010.

[61] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson, "Who are the crowdworkers?: shifting demographics in mechanical turk," in *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, (New York, NY, USA), pp. 2863–2872, ACM, 2010.

[62] 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*, 2003.

[63] T.-C. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, pp. 164–181, Feb. 2011.

[64] 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*, vol. 20, pp. 52–64, Jan 1998.

[65] 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, pp. 128–132, Dec 2009.

[66] 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*, vol. 34, 2001.

[67] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Communications of the ACM*, vol. 54, pp. 86–96, Apr. 2011.

[68] 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, 2013.

[69] 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.

[70] 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, 2013.

[71] 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, 2013.

[72] 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, pp. 1203–1212, 2013.

[73] 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, pp. 211–224, 1999.

[74] H. Cooper, E.-J. Ong, N. Pugeault, and R. Bowden, "Sign language recognition using sub-units," *Journal of Machine Learning Research*, vol. 13, pp. 2205–2231, July 2012.

[75] 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*, vol. 31, pp. 1685–1699, Sept 2009.

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

[77] 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, pp. 72–81, 2011.

[78] D. Frolova, H. Stern, and S. Berman, "Most probable longest common subsequence for recognition of gesture character input," *IEEE Transactions on Cybernetics*, vol. 43, pp. 871–880, June 2013.

[79] H. Stern, M. Shmueli, and S. Berman, "Most discriminating segment - longest common subsequence (MDSLCS) algorithm for dynamic hand gesture classification," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1980–1989, 2013.

[80] 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*, pp. 64–75, 2002.

[81] 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*, vol. 4, pp. 454–457, Aug 2004.

[82] 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.

[83] 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, pp. 1–5, Sept 2009.

[84] 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)*, pp. 1–8, Oct 2010.

[85] J. A. Ward, P. Lukowicz, and H. W. Gellersen, "Performance metrics for activity recognition," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, Jan. 2011.

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

[87] 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.

[88] K. Kunze, G. Bahle, P. Lukowicz, and K. Partridge, "Can magnetic field sensors replace gyroscopes in wearable sensing applications?," in *International Symposium on Wearable Computers (ISWC)*, 2010.

[89] Y.-S. Lee and S.-B. Cho, "Recognizing multi-modal sensor signals using evolutionary learning of dynamic bayesian networks," *Pattern Analysis and Applications*, 2012.

[90] 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*, TEI '08, pp. 11–14, ACM, 2008.

[91] J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li, "Gesture recognition with a 3-D accelerometer," in *Ubiquitous Intelligence and Computing*, 2009.

[92] G. Ogris, T. Stiefmeier, P. Lukowicz, and G. Troster, "Using a complex multi-modal on-body sensor system for activity spotting," in *IEEE International Symposium on Wearable Computers*, ISWC, 2008.

[93] T. Huynh, M. Fritz, and B. Schiele, "Discovery of activity patterns using topic models," in *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, (New York, NY, USA), pp. 10–19, ACM, 2008.

[94] C. Chen and H. Shen, "Improving online gesture recognition with WarpingLCSS by multi-sensor fusion," in *Computer Engineering and Networking*, Lecture Notes in Electrical Engineering, 2014.

[95] H. Sakoe, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, 1978.

[96] S. Brewster, J. Lumsden, M. Bell, M. Hall, and S. Tasker, "Multimodal 'eyes-free' interaction techniques for wearable devices," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pp. 473–480, 2003.

[97] M. Kanis, N. Winters, S. Agamanolis, A. Gavin, C. Cullinan, and H. C. Group, "Toward wearable social networking with iband," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pp. 1521–1524, 2005.

[98] M. Tentori and J. Favela, "Activity-aware computing for healthcare," *Pervasive Computing, IEEE*, vol. 7, pp. 51–57, April 2008.

[99] L.-V. Nguyen-Dinh, A. Calatroni, and G. Tröster, "Robust online gesture recognition with crowdsourced annotations," *Journal of Machine Learning Research (JMLR)*, 2014.

[100] A. Y. Benbasat and J. A. Paradiso, "An inertial measurement framework for gesture recognition and applications," in *Revised Papers from the International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction*, GW '01, pp. 9–20, 2002.

[101] A. Feldman, E. M. Tapia, S. Sadi, P. Maes, and C. Schmandt, "Reachmedia: On-the-move interaction with everyday objects," in *Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, ISWC '05, pp. 52–59, 2005.

[102] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences.," *Journal of molecular biology*, vol. 147, pp. 195–197, Mar. 1981.

[103] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[104] U. Blanke and B. Schiele, "Daily routine recognition through activity spotting," in *Location and Context Awareness*, vol. 5561 of *Lecture Notes in Computer Science*, pp. 192–206, Springer Berlin Heidelberg, 2009.

[105] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, Sept. 2010.

[106] S. Katz, "Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living.," *Journal of the American Geriatrics Society*, vol. 31, Dec. 1983.

[107] R. S. Bucks, D. L. Ashworth, G. K. Wilcock, and K. Siegfried, "Assessment of activities of daily living in dementia: development of the bristol activities of daily living scale.," *Age and ageing*, vol. 25, Mar. 1996.

[108] A. Eronen, V. Peltonen, J. Tuomi, A. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, 2006.

[109] M. Stäger, N. Perera, and T. V. Büren, "Soundbutton: Design of a low power wearable audio classification system," in *Proceedings of the 7th International Symposium on Wearable Computers (ISWC)*, 2003.

[110] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, pp. 165–178, 2009.

[111] V. Akkermans, F. Font, J. Funollet, B. De Jong, G. Roma, S. Togias, and X. Serra, "Freesound 2: An improved platform for sharing audio clips," in *International Society for Music Information Retrieval Conference (ISMIR 2011), Late-breaking Demo Session*, (Miami, Florida, USA), 2011.

[112] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[113] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, July, October 1948.

[114] L. Van Der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, 2008.

[115] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Mach. Learn.*, vol. 39, May 2000.

Curriculum Vitae

Personal Information

Long-Van Nguyen-Dinh
Born on October 30, 1984, in Da Lat, Vietnam
Citizen of Vietnam

Education

2011–2015	Doctoral studies (Dr. sc. ETH) in Information Technology and Electrical Engineering, ETH Zürich, Switzerland.
2008–2011	Master of Science in Computer Science, Purdue University, USA
2002–2007	Bachelor of Science in Computer Science and Engineering HoChiMinh City University of Technology, Vietnam.

Work Experience

2011–2016	Research and teaching assistant, Wearable Computing Lab, ETH Zurich, Switzerland.
2011	Software engineering intern, Google (Mountain View), USA.
2008–2011	Research assistant, Database Laboratory, Purdue University, USA.
2010–2011	Web programming, Department of Biological Sciences, Purdue University, USA.
2007	Software engineer ELCA Informatics, Vietnam
2006	Software engineering intern FSoft FPT, Vietnam