University of Sussex

Movement recognition from wearable sensors data: power-aware evolutionary training for template matching and data annotation recovery methods

Mathias Ciliberto

Submitted for the degree of Doctor of Philosophy

University of Sussex

April, 2021

Declaration

I hereby declare that this thesis has not been and will not be submitted in whole or in part to another University for the award of any other degree.

Signature:

Mathias Ciliberto

UNIVERSITY OF SUSSEX

Mathias Ciliberto, Doctor of Philosophy

Movement recognition from wearable sensors data: power-aware evolutionary training for template matching and data annotation recovery methods

Summary

Human activities recognition finds numerous applications for example in sport training, patient rehabilitation, gait analysis and surgical skills evaluation. Wearable sensing and Template Matching Methods (TMMs) offer significant advantages compared to manual assessment methods as well as to more cumbersome camera-based setups and other machine learning (ML) algorithms.

TMMs require less data for training than other ML methods, they are low-power and therefore suitable for integration on wearable sensor. They compute a sample-by-sample distance between two time series. When applied to gestures sensors data, this even enables a richer and more movement-specific assessment and feedback. However, TMMs lack of a standard training procedure.

In this thesis, we introduce an innovative evolutionary training algorithm for TMM that not only can maximize recognition performance, but it can also prefer power-minimisation by reducing the TMM's computational cost with a configurable trade-off. We exhibit that a reduction is even possible without sacrificing recognition performance by exploiting the long-established concept of "time warping". We demonstrate that our method is suitable for a wide variety of raw data as well as processed, fused and encoded sensor data.

We present a new original multi-modal, multi-user dataset of beach volleyball movements that allowed to evaluate our training methods on a real-case of sport training actions. Moreover, the collection of this dataset helped to generate a set of guidelines for the collection of movement data in the wild, using wearable sensors.

We introduce a 3D human model that can be animated through inertial wearable sensors data for troubleshooting, movement analysis and privacy-safe annotation of human activities. Finally, through a case study on a dataset of drinking actions, we demonstrate how TMM can improve the quality of a badly annotated but highly valuable dataset.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Daniel Roggen for the guidance, the support, the lessons, the great opportunities and the challenges he provided over the years to make me the researcher I am now.

I owe thank to Dr. Luis Ponce Cuspinera for the support and the help, especially with his knowledge and connections to setup the beach volleyball data collection.

I also would like to thank the amazing people I had the chance to work with in the Wearlab and more in general in the Sensor Technologies Research Centre: Francisco Javier Ordoñez, Hristijan Gjoreski, Lin Wang, Sebastien Richoz, Julio Costa, Arash Pouryazdan, Leonardo Garcia, Pasindu Lugoda and Filippo Spina. Working with all of these great scientists gave me motivation, invaluable lessons and it made me a better person.

Most importantly, I want to thank my mum and my dad for the incredible support, understanding and patience, even more so considering the distance and my rare trips to Italy. I know I have not been home as much as you would like, and I love you for always being there, no matter what. A big special thanks goes also to my brother that has always been there for me, and especially for mum and dad while I was abroad.

I also want to express my greatest gratitude to Elsa, who has always been there for me with her wisdom, her positivity and her understanding, to Simone whose integrity, respect and frankness have always been an example, and to Martina whose incredible strength and sincere emotions have always been of great support. I am very proud of having you in my life, no matter how far we are and will be.

I would also to thank all the amazing people I met over the years in Brighton. In particular, thanks to Flavia for all the support and the long conversations, but also for the beers and the incredible nights out. Thanks to my amazing flatmates Julio and Isabelle, who greatly helped making our flat my home. You all made Brighton my city for more than 5 incredible years that I will always bring you with me.

Last but not least, I want to thank Salvatore, Valeria and Federica for being the incredible long-distance friends that you are. I know I have not always been there, but I cherish your friendship more than you know.

Contents

List of Publications

This thesis is based on the content of the following original publications and resources, which are referred as Article I-IX. The articles are re-printed as Apprendix D.

[**I**] Mathias Ciliberto, Daniel Roggen and Francisco Javier Ordóñez Morales. 'Exploring human activity annotation using a privacy preserving 3D model'. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. ACM, Sept. 2016. DOI: 10.1145/2968219.2968290. URL: https://doi.org/10.1145/2968219.2968290.

[**II**] Daniel Roggen, Arash Pouryazdan and Mathias Ciliberto. 'BlueSense: designing an extensible platform for wearable motion sensing, sensor research and IoT applications'. In: *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*. ACM, 2017.

[**III**] Mathias Ciliberto, Luis Ponce Cuspinera and Daniel Roggen. 'Complex human gestures encoding from wearable inertial sensors for activity recognition'. In: *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*. ACM, 2018. ISBN: 978-0-9949886-2-1.

[**IV**] Mathias Ciliberto et al. 'A Case Study for Human Gesture Recognition from Poorly Annotated Data'. In: *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. ACM, Oct. 2018. DOI: 10.1145/3267305.3267508. URL: https://doi.org/10.1145/3267305.3267508.

[**V**] Mathias Ciliberto et al. 'Drinking Gesture Recognition from Poorly Annotated Data: A Case Study'. In: *Human Activity Sensing*. Springer International Publishing, 2019, pp. 71–89. DOI: 10.1007/978-3-030-13001-5_6. URL: https://doi.org/10.1007/978-3-030-13001-5_6.

[**VI**] Mathias Ciliberto and Daniel Roggen. 'WLCSSCuda: a CUDA accelerated template matching method for gesture recognition'. In: *Proceedings of the 23rd International Symposium on Wearable Computers*. ACM, 2019. DOI: 10.1145/3341163.3347718. URL: https://doi.org/10.1145/3341163.3347718.

[**VII**] Mathias Ciliberto, Luis Ponce Cuspinera and Daniel Roggen. 'WLCSSLearn: Learning Algorithm for Template Matching-based Gesture Recognition Systems'. In: *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. IEEE, May 2019. DOI: 10.1109/iciev.2019.8858539. URL: https://doi.org/10.1109/iciev.2019.8858539.

[**VIII**] Mathias Ciliberto and Daniel Roggen. 'WLCSSLearn: Evolutionary training of template matching methods.' In: *Pattern Recognition* (2021).

[**IX**] Mathias Ciliberto, Luis Ponce Cuspinera and Daniel Roggen. '"Collecting a dataset of gestures for skill assessment in the field: a beach volleyball serves case study"'. In: *International Joint Conference and International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. ACM. 2021.

[**X**] Mathias Ciliberto. *"How good are you in beach volleyball?"* https://www.britishsciencefestival.org/event/how-good-are-you-at-beach-volleyball/. [accessed 28-November-2017]. 2017.

List of Tables

List of Figures

Chapter 1

Introduction

Activity recognition is the procedure of spotting and identifying human activities from a certain source of data. We consider activity anything performed for a purpose by a person. Example of activities can be "consuming a meal", "performing a surgery", "completing a workout". Often, an activity is defined as composition of a finite set of shorter actions performed by a person. We consider an action or a movement anything that is performed by a user between two clear starting and ending moments. Example of actions (or movements) are "drinking from a cup", "blinking" or "performing a push-up". When an action has communicative purposes is called a gesture.

The evolution of wearable and ubiquitous computing has enabled the sensing and recognition of human activities and actions through small and low powered sensors. Sensing and automatically recognize human actions has important applications in healthcare, sports, fall detection, just to cite a few [1].

Human activity recognition (HAR) and more specifically action recognition is a problem of pattern recognition where patterns typical of movements of interest need to be identified in time series data originating from sensors [2]. This is an important field with a large number of applications with high societal benefits, such as support for impaired people [3], skill assessment and behaviour analysis [4, 5, 6], human-computer interactions [7, 8].

Activity recognition from wearable sensors data has also been deployed in sport movement analysis [9] and athletes training [10], surgical skills evaluation [11, 12] and training [13], in patients rehabilitation [14], in human motion analysis [15], and in gait assessment [16].

Templates matching methods (TMMs) are algorithms compares a prototypical template with an incoming stream of data in order to find a match and therefore they are a

very suitable solution for HAR. They can have comparable performance to other machine learning approaches with the advantage of requiring less training data [17]. Moreover, they can be used to extract richer information during the classification: for example, it is possible to use them to measure the distance in centimetres between two movements by using the right encoding [**III**]. This is important in sports performance assessment or skills assessment.

In addition, TMMs can have low computational complexity thanks to dynamic programming implementation [18], which enables their implementation in devices with limited computational capabilities, such as battery operated devices and miniature embedded systems. They have been integrated in a wide variety of low-power applications: healthcare [19, 20], Internet-of-Things embedded systems [21], anomaly detection in manufacturing [22], low-resource speech recognition [23], wearable computing [24], and human activity recognition (HAR) [17], of which the last two are the applications we consider in this thesis.

In wearable computing, TMMs have been used for human activity recognition, both to performing crisp detection of pattern [25, 26], as well as quantifying similarity between patterns. The latter can be used for skill assessment as measure of the the ability of an entity (e.g. a person or a robot) at performing a specific assignment.

1.1 Motivations

In this thesis, we are interested in activity and movements recognition using wearable computing and, more specifically, we look at the challenges of power-aware action recognition using Template Matching Methods (TMMs).

Even though other machine learning approaches, such as deep learning, have showed to achieve excellent performance in many wearable computing scenarios [27, 28, 29], the nature of this field, which relies on miniature battery operated embedded devices [30, 31, 32, 33], puts an onus on advancing low-power yet robust pattern recognition algorithms suitable for embedded devices.

Moreover, methods such as deep learning provide only a classification of activities without being able to explain the difference between two movements. Instead, TMMs compare a prototype signal to the incoming data stream such as time series originating from motion sensor and provide a matching score at each time step. This sample by sample similarity measure enables a more granular analysis of the movements enabling additional application of the recognition, for example for skill assessment.

TMMs have been demonstrated effective for action recognition [34]. More specifically, in this thesis, we selected Warping Longest Common Subsequence (WLCSS) as TMM for its robustness against movements variation and low-complexity [26], which make it suitable candidate for embedded application such as wearable computing platforms [35].

However, WLCSS, and more generally any TMMs presented to date, presents the main challenge of lacking of a standard training procedure.

1.2 Aims

- We aim at creating a training methods for TMMs that would maximize recognition performance. Because of the nature of the field of wearable computing, we also want this algorithm to be power-aware in order to reduce the computational cost of template matching based recognition whenever the application warrants it (e.g. need for long battery life outweighs needs for highest accuracy)

- While TMMs have been selected for their runtime low-complexity, the training process can be performed offline and therefore it can leverage high performance architectures, such as modern multi-core CPU and GPU. We aim at accelerate the training procedure by exploiting such architectures.

- We aim at evaluating our methods on a dataset of activities and movements that would ultimately profit from skill assessment, such as a sport application. We focus primarily, but not only, on beach volleyball serve movements. Beach volleyball is a sport that comprises a limited set of very different basic movements (serving, passing, hitting) that can be combined during a game between two teams of two players. Every basic movement can then have multiple variation resulting in different outcomes in the game. In this work, we focus on the serving action because i) it is the first movement of every rally which means it can greatly affect every consecutive action; ii) it is not affected by any previous action of the game, allowing players to have full control and freedom in performing the movement. Performing a good serve can make a significant difference between winning and losing a point, by putting the pressure on the opponent from the first movement of the rally. Despite being a popular sport, there are no publicly available dataset of beach volleyball movements collected using wearable sensors.

- We also aim at extensively evaluate our methods on different set of actions and sensors modalities, therefore we also need additional annotated datasets. We aim

at identifying datasets suitable for activity and movement recognition, by selecting some of those proposed in the wearable and ubiquitous computing community. We aim at selecting several application scenarios, different sensor modalities and diverse sensor fusion methods.

- Due to possible quality issues that can affect datasets such as bad and missing annotation, we aim at developing methods to help identify, correct and recover dataset with such problems before to apply our methods. Hence, we aim at creating visualisation tools that would allow us to analyse the movements data in order to possibly fix these issues. When it is not possible to deploy visualisation tool, e.g. because the inadequacy of some sensor modalities, we need methods to automatically recover the correct annotations for a dataset.

1.3 Contributions

As result of our motivation and aims, in this thesis, we present five contributions:

Beach volleyball dataset We collected a new dataset of beach volleyball movements recorded using wearable inertial sensors. The dataset includes a total of 10 users wearing sensors in 4 positions, divided on 3 sessions. We recorded this dataset with a focus on the serve actions, but also including game rallies between players. It is freely available for the community and to best of our knowledge, it is the only publicly available dataset of beach volleyball movements comprising inertial data from wearable sensors.

3D Human model for movement animation and annotation We developed a 3D human model for movement analysis and annotation. Using the orientation data provided by wearable inertial sensors, this 3D model can be animated in real time or offline for troubleshooting issue in the movement data and annotations. We present a study where this model was demonstrated to be suitable for privacy-preserving labelling of data.

WLCSSLearn training algorithm We created the first automatic training procedure for the Warping Longest Common Subsequence (WLCSS) template matching method. Based on evolutionary algorithm, WLCSSLearn comes in two versions: WLCSSLearn_p trains the parameters of WLCSS to maximize the movement recognition performance. WLCSSLearn_t generates variable templates in length and shape with a configurable trade-off between recognition performance and power-saving through computational costs reduction. To the best of our knowledge, WLCSSLearn is the first training methods that i) allows such trade-off, ii) is application independent, iii) can be easily applied to other

TMM.

WLCSSCuda In order to support the evolutionary training and to accelerate the template matching with WLCSS, we introduce the first GPU based implementation of WLCSS. WLCSSCuda exploits General-Purpose Computing on Graphics Processing Unit (GPGPU) in order to match multiple templates and segments in parallel. We demonstrate that not only WLCSSCuda is faster than a standard CPU only based implementation, but it also scales better with the increase of the number of templates and segments.

Segments extraction from poorly annotated data We demonstrate how WLCSS and WLCSSLearn can be deployed to recover a poorly annotated but high value datasets for action recognition with a case study of drinking actions. A dataset of more than 8800 events recorded using specifically instrumented mugs and loosely annotated by the participants is relabelled using TMMs. We demonstrated that the quality of the dataset can be highly improved through automatic methods.

1.4 Thesis outline

This thesis develops in three parts: Chapter 2 and 3 deal with the problem of collecting a datasets and the tools required for checking the quality of the data. Chapters 4 to 7 constitute the main corpus of this thesis, introducing the problem of activity and movement recognition using TMM and presenting our innovative algorithm for the training of our selected template matching method, WLCSS. Finally, Chapter 8 presents a case study of how our methods can be used for recovering a datasets of badly annotated sensors events. More specifically, each chapter develops as follow:

Chapter 2 describes the collection of the dataset of beach volleyball movements. We present the sensors, the data collection protocol and the data annotation procedure. We also presents the challenges we encountered during the annotation phase due to missing videos for one of the recording session.

Chapter 3 presents the 3D human model for movement animation, analysis and annotation. We describe the 3D model, as well as the pipeline for real-time and off-line movement animation. Finally, we illustrate how this model can be used for data annotation while also preserving the privacy of the participants involved in a data collection.

Chapter 4 presents the topic of activity and movement recognition using TMM, introducing the concepts and the notations, as well as providing a description of the main TMMs. This is useful to explain the motivation for choosing WLCSS as TMM, the activity recognition pipeline used in this thesis as well as helping to discuss the applicability of

our training algorithm to other TMMs.

Chapter 5 describes the training methods for the parameters of WLCSS, named WLCSSLearn_p. Chapter 6 presents the training methods for generating variable length templates in a power-aware manner, named WLCSSLearn_t. The respective chapter for each version presents a review of the very few and very specific training procedures proposed in the literature and their limitations. Then we detail the algorithms, the main parameters and their evaluation. We also discuss how the underlining concepts of WLCSSLearn are generic enough to be applied to other TMM.

Chapter 7 introduces WLCSSCuda, a GPU implementation of WLCSS aimed at hastening the training and matching speed by exploiting the highly parallelized computation of General Purpose GPU (GPGPU).

Chapter 8 illustrates a case study of recovery of a highly valuable dataset comprising 8800 poorly annotated segments of drinking/no-drinking actions using TMMs. We analyse the provided annotations recorded through experience sampling (ES) and show the low reliability for supervised learning. We demonstrate how TMM can be used together with our training method to more accurately select and re-annotate drinking and no-drinking actions.

Finally, Chapter 9 concludes this thesis with a summary of our achievements, their potential limitations and some prompts for future work.

Chapter 2

Data collection in the wild: A beach volleyball case study

In the field of activity recognition, the first need is data. In this chapter, we present the collection of a dataset of beach volleyball movements. Starting from the motivation, the chapter presents the sensors setup, the data collection protocol, the annotation process and the challenges of collecting such dataset, with a case study of missing videos due to a failure in the cameras recordings.

*The content of this chapter has been published in [**IX**]. The sensors used during the data collection was described in [**II**]. I contributed in the development and debugging of the sensor's firmware.*

2.1 Introduction

A dataset of well annotated activities is fundamental for training and evaluation of automatic recognition methods [2]. As presented in Section 4.3, several datasets have been presented in the community for this goal. However, in order to evaluate our methods we decided to also collect a new dataset of actions specifically recorded in an environment that would benefit from skill assessment. We chose beach volleyball as domain for our dataset.

Beach volleyball is a sport that comprises a limited set of very different basic movements (serving, passing, hitting) that can be combined during a game between two teams of two players. Several studies explored the technical aspect of beach volleyball actions and their effectiveness on the game [36, 37] through the visual and manual analysis of videos in what is called *notational analysis*. This method has been used to investigate

the variations of techniques among different kind of players [38], the dependency of the different techniques to the success of the game [39], to evaluate a single technique [40], to understand the biomechanics of a specific technique [41, 42].

From the literature, it has also emerged the high correlation between a good quality in the basic techniques and the success in the game [37]. For this reason, it is important for the athletes to achieve the best quality in the performing of the game technique.

The analysis of the beach volleyball games has been performed used mainly the video recording and the tracking of the players [43, 44]. A first attempt to use wearable sensors to autonomously recognize the beach volleyball serves and other movements has been studied in [45, 46]. However, despite a wide set of videos publicly available for beach volleyball training and games, we were unable to retrieve a dataset of playing actions recorded using wearable sensors. The few studies investigating wearable sensors for beach volleyball application did not make their datasets available to the community or they did not include a set of sensors that could enable movements encoding.

We collected our new dataset of beach volleyball movements using wearable inertial sensors. Recording a new dataset allowed us to:

- have full control on the set and positioning of the sensors, enabling, for example, the encoding of the movements for recognition (see Section 4.2);
- customize the data collection protocol in order to focus on specific actions. We focus on the serving movements because i) it is the first movement of every rally which means it can greatly affect every consecutive action; ii) it is not affected by any previous action of the game, allowing players to have full control and freedom in performing the movement.

In this chapter, we describe our data collection and the resulting dataset. We introduce the equipment, the data collection protocol and data annotation process. We also present a possible method for dealing with possible issues during the data collection, with a case study. Finally, we conclude with the discussion of a set of guidelines for the collection of datasets using wearable sensors.

2.2 Data collection equipment

The data from the players were recorded using a set of wearable in-house developed inertial sensing platform, called BlueSense. The entire data collection was recorded using cameras for a-posteriori annotation. Both sensor modalities are explained in the following.

Figure 2.2.1: *BlueSense AHRS.*

2.2.1 BlueSense

BlueSense is a sensor research platform developed for wearable and IoT application [**II**]. Its small size of just 30x30mm allows to be worn with minimal to none hindering of the user while performing movements. The platform is instrumented with a 3-axis inertial unit (IMU) including an accelerometer, a gyroscope and a magnetometer. Thanks the IMU and its microcontroller, BlueSense is capable of computing 3D orientation on-board, making it what is called an Attitude and Heading Reference System (AHRS). BlueSense can sample data up to 1 KHz, and provided orientation data up to 500Hz, while also logging the raw motion data on a locally stored microSD. The orientation in the form of quaternions or Euler angles is computed combining the raw data of the IMU using a variation of Madgwick's algorithm [47].

BlueSense includes a Bluetooth module for streaming of commands (e.g. start/stop recording) and data. The platform also includes an on-board real time clock (RTC) with minimal time deviation of 0.6ppm. This enables to synchronise multiple BlueSense platforms with a single timestamps and having them synchronised for the entire data collection, enabling an easier merge of data and annotations once the collection is complete.

For this data collection, we set the sampling rate to 500Hz as we recorded both raw and orientation data. We also used the integrated microSD as streaming data over Bluetooth would results in possible loss of samples, as well as lower sampling rate. The commands were sent to the sensors using a specifically developed Android application [1]. This app allowed not only to start/stop the recording of the data, but also the synchronization of the timestamps for multiple sensors simultaneously.

[1] https://play.google.com/store/apps/details?id=net.danielroggen.bs2mgr

Figure 2.2.2: *Placement of the BlueSense sensor platform on player.*

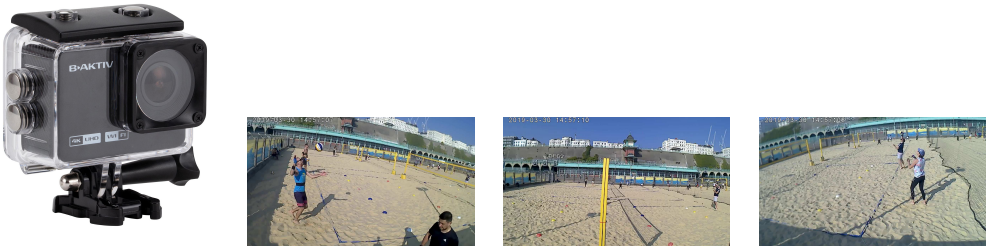
2.2.2 Sensor placement

A set of 4 BlueSense per player was placed on the torso, the dominant upper arm, lower arm and hand of the players, thanks to a set of strap bands as shown in Figure 2.2.2. The straps were made by using an an elastic band and industrial velcro. This ensured a tight placement of the sensor on the body of players minimising accidental vibrations of the platform.

By including the torso and the dominant arm, the orientation data can be used to encoded movements as explained in Section 4.2 or to animate a 3D model like the one described in Section 3.3.1.

2.2.3 Cameras

Video cameras were used to record every session in order to annotate the data at a second time. Three action cameras were placed as shown in Figure 2.3.1. These cameras were chosen because of their wide field of view, allowing a comprehensive recording of the entire court (see Figure 2.2.3). They were set to record at 1280x720 pixels and 120fps. While they would have been able to record at higher resolution, we opted to reduce the frame size in exchange for a higher framerate that could be more useful when looking a fast movement as beach volleyball serves.



(a) Action camera **(b)** Left view **(c)** Center view **(d)** Right view

Figure 2.2.3: *Action camera for annotation and camera views, respectively for left, center and right cameras.*

2.3 Data collection

A data collection can be a time and effort consuming task, and therefore must be planned in details in advance. For this reason, we accurately defined the data collection protocol for the participants as well as the placement of the sensors, and the organization of the court.

2.3.1 Participants recruitment

The first step of the collection was the choice of the participants. In order to minimise errors while performing the protocol, we needed players that would be able to serve and perform specific game actions on command. The participants were selected according with their level of expertise from the community of players at our closest beach volleyball venue. The players's level variated between semi-professional to former professional players. We did not constrain gender or physicality. We selected 10 players, 5 females and 5 males, divided in 3 sessions of 4 players. 2 female players were present in two sessions. 9 players were right-handed and 1 was left-handed. More about the participant recruitment can be found in Appendix A.

2.3.2 Court Setup

The court was setup accounting for one serving areas and two landing areas on each side of the net. The two landing areas were delimited by little cones and differentiated between short and long serves. We empirically set the landing areas to 1.5m deep. They were delimited by little cones that were removed for the gaming phase of the data collection.

The net height was set to 2.33m as average between men (2.43m) and women (2.24m) official heights.

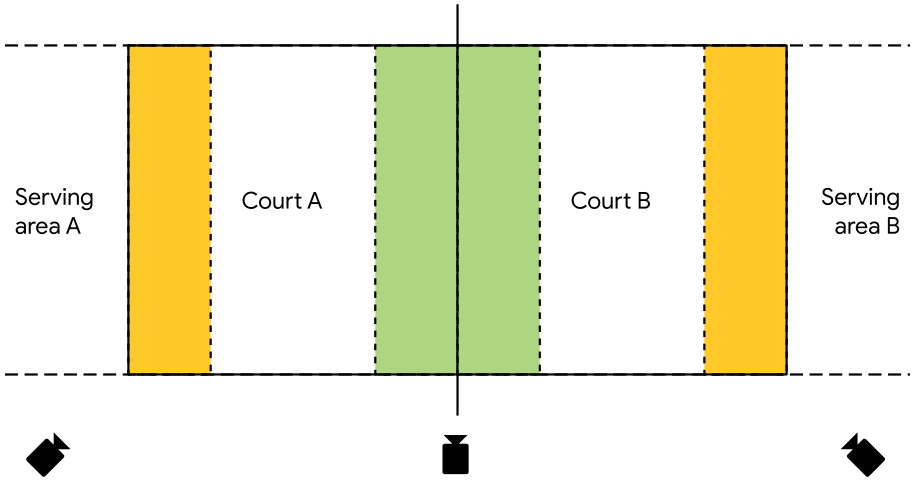


Figure 2.3.1: *Setup of the court during data collection. The green areas are the aimed landing area for short serves, while the orange areas are for long. Players can serve from any point in the serving area of each court, aiming at the landing areas of the opposite side. The three wide angle cameras, placed as shown by the black symbol, offer a comprehensive coverage of the entire court for annotation.*

2.3.3 Data collection protocol

The players were instructed to follow a precise but flexible protocol during each session of the data collection. The protocol was developed in order to record more controlled serves actions as well as free in-game movements. 4 players would perform such protocol in a 2 hours session, excluding some time for setup and synchronisation.

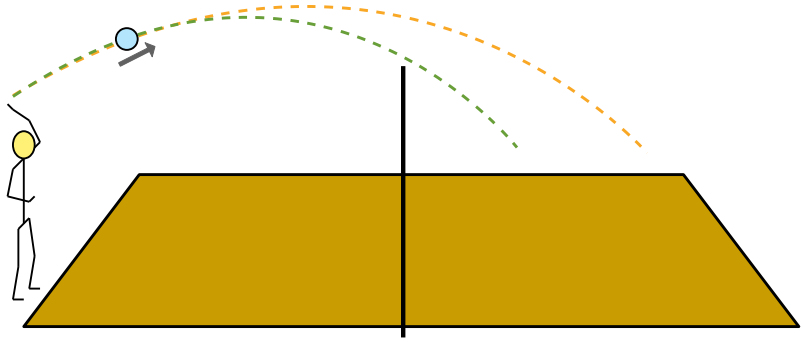
The protocol was divided in two main parts: the more controlled serving part and the free game part. In the former, the participants were asked to perform specific serves individually, as they were in a training session. In the latter, the participants were allowed to freely play few sets following the standard 2-vs-2 rules of beach volleyball.

We identified 2 main type of serves to consider, graphically presented in Figure 2.3.2 and described in the following. According with the force applied by the player and therefore the length of the trajectory, each type can be short or long:

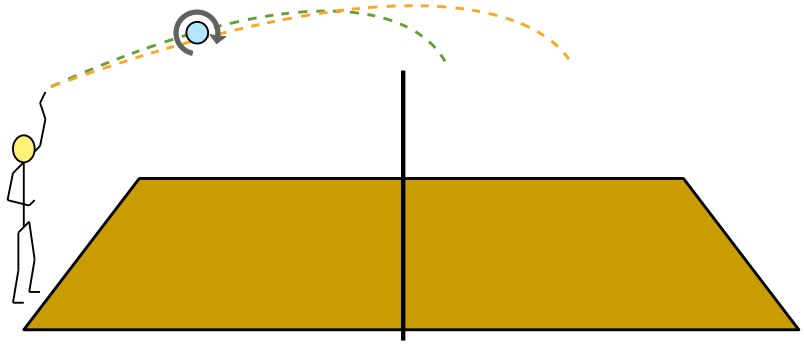
Float serve In this serve, the player hits the ball at the centre with a flat and stiff hand. This cause the ball to float in the air with a small unpredictable side-to-side movements.

Top spin In this serve, the player perform a wider movement with the arm, compared to the float serve, and he hits the ball at the top-centre with a spinning forward movement of the hand. The ball to rotate forward during the flight and the trajectory

result more curved compared to a float serve.



(a) Float serves



(b) Top spin serves

Figure 2.3.2: *Example of float and top spin serves with trajectories, for long (orange) and short (green) serves.*

Once the players were instrumented and the court was prepared, the recording on the sensors and the cameras started. Then the protocol begin with a synchronisation step comprising 3 hand claps in front of the cameras. This allowed the synchronization of all the involved data sources for an accurate annotation (see Section 2.4). The protocol proceeded as presented in Table 2.3.1.

Finally, the players were asked to repeat the synchronisation step before to be de-instrumented from all the sensors.

Players were asked to follow the protocol at their best. However, we left room to correct possible mistakes during the data collection such as the rest breaks and additional time in between the two parts of the protocol.

2.4 Annotation

The annotation was performed off-line using a previously in-house developed software. The software supports multiple video and data sources, as well as multiple annotation

Step No.	Rep./Dur.	Activity	Description
1	8 mins	Warm up	The participants could warm up freely with the ball. This was necessary to reduce the risk of injuries and it also allowed the players to get familiar with the sensors while playing.
2	12x	Long float serves	The players placed themselves at the end of the court in the serving area. They were asked to serve 12 long float serves. Each player must count its own serves.
3	1 min	Rest	
4	12x	Long top spin serves	The players placed themselves at the end of the court in the serving area. They were asked to serve 12 long float serves. Each player must count its own serves.
5	1 min	Rest	
6	12x	Short float serves	The players placed themselves at the end of the court in the serving area. They were asked to serve 12 long float serves. Each player must count its own serves.
7	1 min	Rest	
8	12x	Short top spin serves	The players placed themselves at the end of the court in the serving area. They were asked to serve 12 long float serves. Each player must count its own serves.
9	1 min	Rest	
10	Rest of time	Free game	The players were asked to freely play for the remaining time of the 2 hours session. The free game respected the standard 2-vs-2 rules, with sets up to 21 points and changing side every 7-points. The players teams stayed the same for this entire part.

Table 2.3.1: *Data collection protocol. The columns indicates in order: the step number, the duration of the step in number of repetitions or minutes, the activity required in each step and a brief description.*

tracks. All the data sources can be synchronised within the application. A screenshot of the annotation software is shown in Figure 2.4.1. The annotation was performed by a single annotator recruited amongst the PhD students in our laboratory. The annotator was carefully instructed by the main author of this data collection, who was also present in order to solve possible ambiguities.

For this first iteration of the dataset, we labelled only the part of the data collection including the controlled serves of the players. The labels for each of the four serve types were introduced manually for each player. The manual annotation allowed to correct possible mistakes between serve requested by the protocol and the one actually performed by the players.

The annotations were then exported from the software and merged with the data of

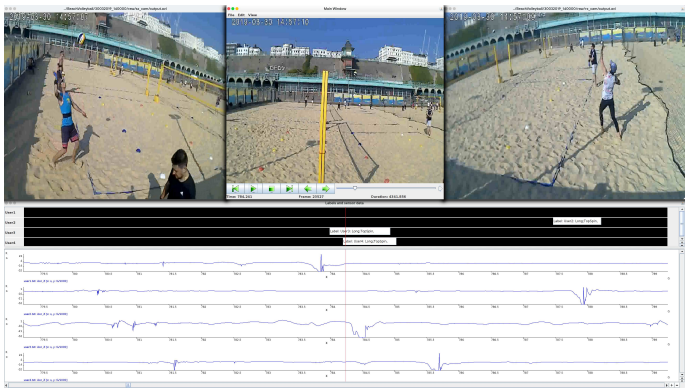


Figure 2.4.1: *Annotation tool used to label the data. The videos on the top are used to synchronise all the source and to analyse the movements. The annotation are then inserted in the black tracks (one track per player), in the centre of the screen. On the bottom, a downsampled signal from one channel of the BlueSense's IMU is shown to check the consistency of the sensor data with the movements. One signal is displayed for each player.*

each player singularly. The data from the multiple sensors, torso, upper arm, lower arm and hand, for each player were synchronized thanks to the timestamps associated with the sampled data from the BlueSense.

2.5 Dataset issues

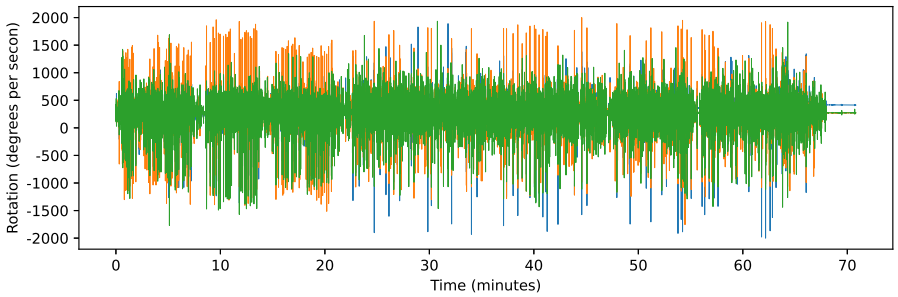
Unfortunately, during the data collection we encountered two main issues that could affect the quality of the data: i) the missing videos for one of the session due to a failure of the cameras, ii) the malfunctioning of two sensors for one participant resulting in difficulties in annotating such data.

2.5.1 Missing videos

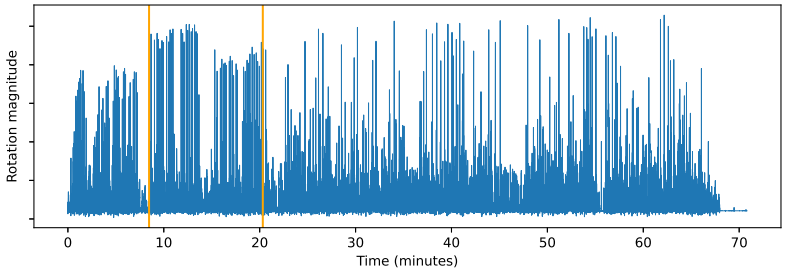
During one of the sessions, the cameras failed after few seconds of recording. This resulted in the impossibility of annotating the data as described in Section 2.4. While we could not figure out the reason for the failure, we managed to recover the annotation thanks to the precise protocol.

We could indeed analyse the data knowing the order of the steps each player followed and therefore annotate the serving segments net of possible mistakes made by the players in the serving.

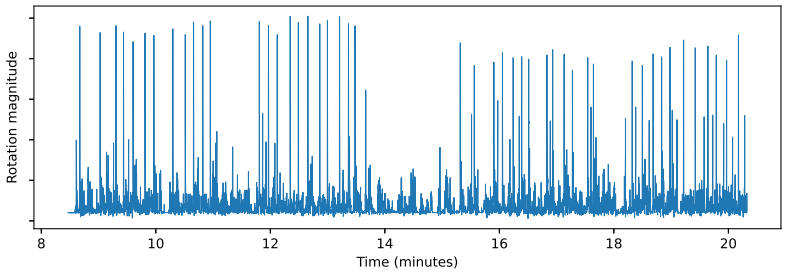
Figure 2.5.1 shows the steps we performed to annotate the data:



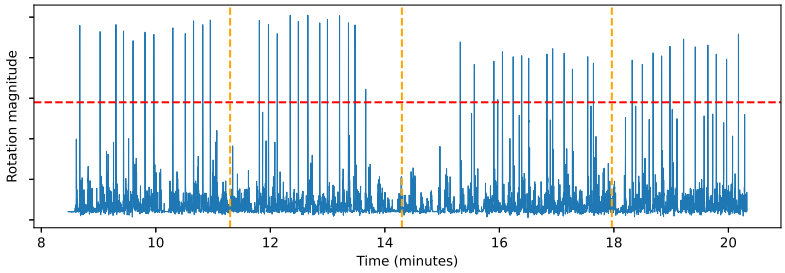
(a) Raw data



(b) Magnitude data



(c) Serves data



(d) Serves type data

Figure 2.5.1: *Example of data annotation recovery procedure for one user. The raw data from the gyroscope of the BlueSense positioned on the lower arm are plotted in the top Figure 2.5.1a. The magnitude of the signal is then computed in order to identify the section of the data regarding the serves 2.5.1b. This section is identified in Figure 2.5.1b by the two orange dashed lines. Figure 2.5.1c displays an enlargement of the section of data pertinents to the serves. The four sections for each one of the serves type is then visually individuated and represented in Figure 2.5.1d as separated by the vertical orange dashed lines. A threshold displayed with an horizontal red dashed line is then used to select the peaks in the data that would be selected as serves.*

1. We started by evaluating the raw data provided by the gyroscope of the BlueSense placed on the lower arm. We chose this sensors for its sensibility and because it is not affected by gravity (accelerometer) and external magnetic fields (magnetometer).

2. Considering the serve actions, we assumed that as they have higher energy compared to slower actions such as passing. Therefore we computed the magnitude of the three channel data as $\sqrt{x^2 + y^2 + z^2}$ as a measure of the quantity of rotation.

3. We identified the section of the data referring to the serves in the protocol. Starting from the synchronization claps that can be identified at the beginning of the time series, we found a section of recurring pattern divided in four sections, one per serves type in the protocol. The recurring patterns were identified by the highest peaks in the section. The peaks in the signal resulted by the impact of the hand with the ball at each serve.

4. We empirically defined a thresholds in order to filter only the peaks referring to serves. We then defined the boundaries of each section corresponding to long floats, long top spin, short float, short top spin serves. As expected according to the protocol and considering the threshold, around 12 peaks were contained in each section.

5. For every peak in each section, we selected a segment of the signal of 450 samples and we annotated such segment with the corresponding serves type. The length of the segments was defined as average length of the already annotated serves segments.

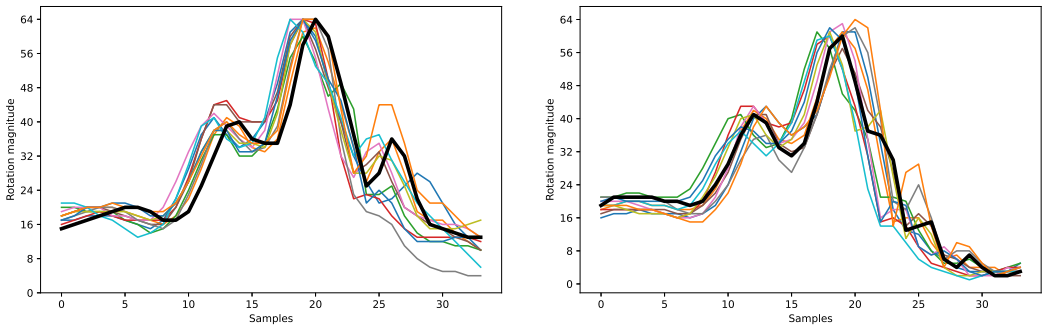
We repeated this process for all the four players in the session with missing videos.

In order to verify the correctness of our procedure, Figure 2.5.2 presents the segments annotated through this method for one player. This player participate in session 3 as well as in session 1 that was correctly annotated using the videos, allowing to compare the video-annotated segments and manually annotated segments for the same participant. In the figure, it is possible to notice how the manually annotated segments in colours have similar shape to the most representative segment of the video-annotated annotated ones.

This procedure allowed to annotate the serving data for an entire session, comprising of 4 players for a total of 16 sensors data that would otherwise be unusable for supervised movement recognition.

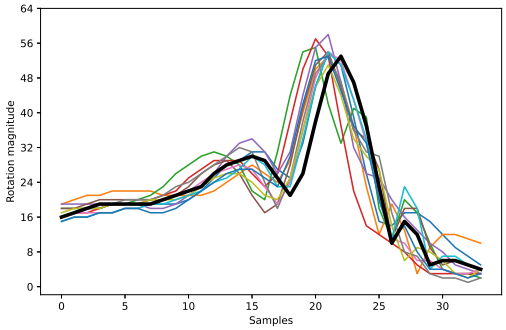
2.5.2 Sensors errors and reliability

During an initial session, we discovered the failures of two BlueSense on the same player. The failure were due to a bad configuration of the sensors themselves: in one case, the

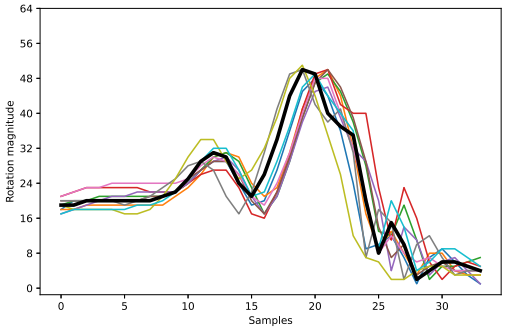


(a) Long float

(b) Long top spin



(c) Short float



(d) Short top spin

Figure 2.5.2: *Comparison of artificially extracted and video-annotated movements. A player participated in two sessions, one with videos and one with missing videos. Thanks to this, it is possible to compare the segments of the movements extracted from the sensors data (over-imposed in colours) and a video-annotated segment (in black), for each serves type. The video-annotated movement is selected as MRT among the segments for each serve type.*

microSD was not properly formatted, resulting in missing data from the torso sensors. The second issue was due to a bad configuration of the output format of the data from the hand sensor that made them non-readable. In the final release of the dataset, we still include the annotated data from the upper arm and the lower arm of the same user, as they were still available.

2.6 Dataset summary

Table 2.6.1 summarises the datasets, net of sensors failure. The dataset comprises a total of 585 annotated serves, divided in 147 long float, 140 long top spin, 155 short float and 143 short top spin serves. In term of time, the dataset includes a total of 2 hours and 12 minutes of annotated data and a total time of 11 hours and 45 minutes of sensors data. These times account for each users in the same session separately.

user_ID	gender	hand	session	lf	lts	sf	sts	errors	ann_time	total_time	sensors
01	F	R	1	16	12	11	9	6	00:12:08	00:41:16	-ul-
02	M	L	1	12	10	16	15	5	00:11:28	00:40:30	tulh
03	M	R	1	8	13	14	12	2	00:11:08	00:40:34	tulh
04	F	R	1	13	12	13	14	2	00:11:06	00:40:24	tulh
05	M	R	2	12	11	12	13	2	00:09:58	01:00:21	tulh
06	M	R	2	12	12	12	13	2	00:10:02	01:04:57	tulh
07	M	R	2	11	10	12	11	2	00:09:42	01:05:27	tulh
08	F	R	2	12	12	12	11	0	00:09:27	01:05:36	tulh
01	F	R	3	13	12	14	12	0	00:12:08	01:12:10	tulh
09	F	R	3	12	12	13	11	0	00:11:40	01:10:49	tulh
10	F	R	3	14	12	14	11	0	00:11:15	01:12:37	tulh
03	F	R	3	12	12	12	11	0	00:11:48	01:10:49	tulh
Total				147	140	155	143		02:11:55	11:45:34	

Table 2.6.1: *Beach volleyball dataset summary. The columns indicate respectively: the user_ID, the gender of the player, the handedness, the number of each serves with lf = long float, lts = long top spin, sf = short float, sts = short top spin. The number of errors is calculated as the number of serves of the wrong type performed by each player. ann_time is the amount of time of annotated data and total_time indicated the total amount of recorded data for each session. Finally, the sensors columns reports the sensor data that are available for each user: t = torso, u = upper arm, l = lower arm, h = hand.*

2.7 Discussion and Conclusion

Throughout the dataset recording and thanks also to the issues that we encountered, we were able to define a series of guidelines for future data collection that we want to discuss.

First, we think it is important to plan every single step of the data collection in advance. Defining an accurate protocol is fundamental for the success of the data collection as well as for handling possible errors of the participants and failure of the hardware.

About the hardware, a lesson we learnt is to invest in high quality hardware: the cameras failure was an issue that could be prevented by investing in more reliable and well known products. On the other hand, the wearable sensors were more reliable with a single issue caused by a mistake in the configuration and not from the sensor itself.

However, we want to stress how important is to extensively test the hardware beforehand. We conducted several tests of the cameras and of the sensors: this was important, for example, to set the sensitivity range for the IMU on-board of the BlueSense, adapting it to the specific application. Unfortunately, in all previous tests the cameras did behave correctly without displaying any problem: a confirmation that even extensive testing

sometimes cannot be enough.

For this reason, we learnt that possible missing data must be accounted for in such complex data collection. In some case, such as the missing videos, the data can be recovered through a manual inspection or other approaches (see Chapter 8). In other cases, such as for sensors failures, some data can be irrecoverably lost. This can be overcome by including a high number of participants.

We found that keeping the participant engaged is an important goal to keep in mind during a data collection. During the recording of the dataset, we found that players were quite keen to participate as they could play free of charge for some time, once they completed the first part of the protocol. This is an important aspect to consider, as the quality of a dataset can be greatly affected by the lost of commitment by the users [**V**].

Towards this goal, it is also important to extensively try the protocol in advance. Especially when it comes to human activities, if the protocol is too energy demanding for the participant, the quality of the movements can decrease after few repetitions. More importantly, a protocol that is too intense can even cause injuries in the players. For this reason, we included a mandatory warm-up section at the beginning of every session, as well as well defined rest interval between exercises.

Assessing the protocol in advance can also help to optimize the efforts of the researchers performing the data collection. For example, iterating the protocol through just few tests, we were already able to better spend our effort in setting up the courts, the cameras and the sensors, by reducing the discomfort of the player when the weather conditions were not optimal. This is visible in Table 2.6.1 as the total amount of data we collected increases in time with the progression of the sessions.

On the same note, the discomfort of the users could be minimised even further with a better placement of the sensors. We realised that in the effort of minimise the sensors noise by reducing accidental vibrations, we built the straps quite tight and potentially discomfortable. Additionally, the usage of strong industrial velcro resulted in some irritation on the skin of some players. This could be improved in future iterations for example, by building a full vest integrating the sensors instead of single bands, or even by fusing the sensors in the fabric.

We also want to discuss the environment condition for data collection. In our case, a specific setting such as the beach volleyball court was required. Unfortunately, being based in the United Kingdom affected our efforts in the data collection as the weather reduced the availability of the courts. We strongly advise to take the environment con-

dition into consideration when planning a data collection in the open. For example, we eventually realised that setting the data collection in more sunny countries would have greatly increased the availability of courts and people, with little impact on the costs.

However, despite the issues with sensors and cameras, and the numerous way this data collection could be improved, we consider this dataset a success. The dataset comprises a total of 585 annotated servers, for a total of more than 2 hours of annotated data plus an additional 9 hours of non-annotated data. The dataset is available to the community at https://ieee-dataport.org/open-access/wearlab-beach-volleyball-serves-and-games

Chapter 3

Privacy preserving annotation of movements using a 3D model

*We saw how dataset can present issues such as bad annotations. Especially when no video is available, we need tools to recover the movement data. This chapter presents an evaluation to which extent a 3D human model can be used for privacy preserving annotations. We introduce the model, the animation pipeline that has been first used in [**III**] and [**X**]. Finally, we describe three scenario in which the model has been deployed for movement annotation as published in [**I**].*

3.1 Introduction

In order to achieve a reliable recognition of the actions of the user, an annotated dataset is needed to train the machine learning classifier and TMMs. Annotation requires that someone manually specify the actions carried out during the data recording. To do this, usually several videos are recorded jointly with the recording of wearable sensor data. After the recording, the video from the cameras are synchronized with the data logged by the sensors, in order to annotate precisely the start and the end of each activity. This can be very time-consuming [48]. For this reason, it is usually done using cheap labour with recent research even looking at crowdsourcing. In such a case, it is yet important to preserve the privacy of the user whose data was recorded.

We investigate to which extent our 3D human model animated from inertial sensors placed on the user's limbs can be used to label the activities of that user while preserving his/her privacy. We describe:

- an investigation about the segmentation of the activities of the 3D model (i.e. identi-

fying the occurrence of an activity regardless of its class). We compare the results of segmentation done by the participants to the experiments with the ground truth segmentation of activities in the dataset.

- an analysis of the system in a open-ended annotation scenario: we let the user free to assign a custom label to each activity in a set. The goal of this experiment is to understand to which extent this model can be used without any knowledge of the action carried out by the user during the recording. The results of these test are presented using a set of tag clouds of the words used by testers.

- an evaluation of traditional annotation accuracy, where an activity annotated in the dataset is played by the model and the testers must pick which activity it could have been among a list of 11 activities. Results are summarised by a confusion matrix.

3.2 Related work

There are multiple approaches to annotation [2]. Usually the annotation or labelling of the collected data is done a posteriori using a video recording of the experiment synchronised with the sensor data [48]. During this process the video synchronized with the inertial data requires that each sequence of data must be accurately analysed to segment and recognize each activity correctly. Sometimes multiple cameras are used to record the scene from different perspectives [48]. The activity of labelling can be a tedious and time-consuming task: for 30 minutes of recording, the annotation could take 7-10 hours of analysis [48]. For this reason, usually the annotation phase is done using cheap labour.

Recently, crowd-sourcing has been suggested to help reduce the cost and time of annotating datasets. Crowd-sourcing is a process where a task can be completed by soliciting contributions from a large group of lay people. Thanks to this technique, the researchers can obtain an annotated dataset employing several people at the same time. This can be realised, for example, using platforms like Amazon Mechanical Turk (MTurk) [49]: this is a web service where users can ask for workforce. The workers can pick up a task and complete it earning a money reward. Crowdsourcing has been used to tag human activity from video [50]. It has also been used to label natural language [51], for speech recognition [52] and for multimedia tagging [53].

When the annotation task is done using the videos, one of main issues is to preserve the privacy of the subject in the dataset/video. One way to protect the anonymity of the subjects in the dataset can be the application of a mask (e.g. a blur or a pixelize video

filter [54]) to the elements in the video that could be considered meaningful from a privacy point of view. This step brings however additional time and work to the annotation task. Moreover, the preprocessing cannot be easily applied to all the elements in the scene in order to do not alter too much the video and to do not compromise the recognition of the activities. Another approach consists in using low resolution camera to preserve the privacy of the subject recorded [55].

A different way can be the real-time annotation of the data. That can be done using short audio labels recorded by the subject of the dataset together with the inertial data [56, 57]. This method preserves the privacy and can be accurate. It can be also used in a "open-ended" context thanks to the absence of a predefined set of labels. Real-time labelling requires the direct interaction of the user and in the everyday life this could be annoying. Instead, in case of a real time annotation made by an external experimenter that observe the scene, it requires that the user carries out the activities in a controlled environments and it is not always possible. Moreover, the external observer could condition the way the user will do the activities.

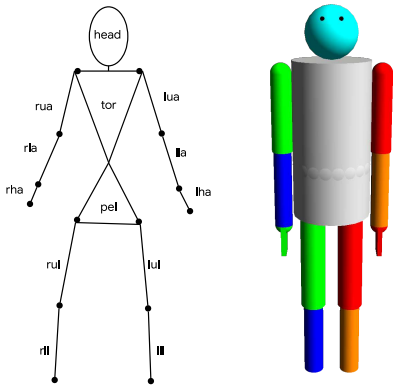
3.3 3D human model for movement animation

Several 3D human models are available online as asset ready to be imported in the major 3D engines [58] [59]. Tools are also available to create realistic 3D human models [60]. However, in both cases, most of the models are developed with a specific structure of the body parts, generally tuned for game characters, leaving little control to the structure of the model and therefore of the animation.

We created our own 3D human model for the annotation of activities and gestures. While our model is an unsophisticated set of shapes (see Figure 3.3.1), this allows a granular control of the animation of each body part. Moreover, we developed the model in order to have a wider applicability as motion tracking system, in both real-time and off-line applications.

The 3D human model is a composition of a skeleton and a mesh. The former provides the model with the internal structure for movement and positioning of the body parts, while the latter produces the external appearance of the model. The skeleton and the mesh of our 3D model are displayed in Figure 3.3.1. The description of each body part and its ID is presented in Table 3.3.1.

Our human model is built and rendered using the open-source 3D engine called jMonkeyEngine [61]. This multi-platform engine written in Java allows to develop a model from



(a) **(b)**

Figure 3.3.1: *3D human model skeleton and mesh.*

ID	Description
pel	Pelvis. It is a fixed point for the model.
tor	Torso
rul	Right upper leg
rll	Right lower leg
lul	Left upper leg
lll	Left lower leg
rua	Right upper arm
rla	Right lower arm
rha	Right hand
lua	Left upper arm
lla	Left lower arm
lha	Left hand

Table 3.3.1: *3D human model body part description*

the ground up. We built our custom model to keep easier the handling of the animation and to guarantee more flexibility during the application of inertial data to each body part. Moreover, the multi-platform nature of the engine would allow us also to deploy this model and its animation in a crowdsourcing scenario, as it can be integrated and used remotely, eg. in a web page.

3.3.1 Movement animation

The model is made of 13 parts. These parts are connected in a hierarchical structure starting from a fixed point set on the pelvis. The hierarchy is shown in Figure 3.3.2. The movement of a body parts affect all the other parts connected on lower level of the hierarchy. This allows the model to behave as a human body would do, where, for example, the movement of the torso affects the movement of the arms and head.

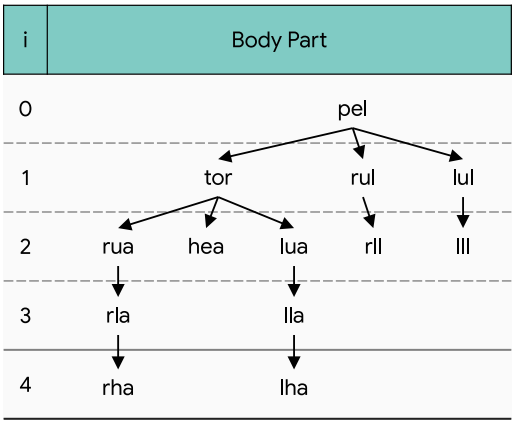


Figure 3.3.2: *Hierarchy of body parts as connected in the human body and our 3D model.*

During the animation, the position of each body part must be computed at every frame. The absolute orientation of each body part must be provided as a quaternion $q = \{w, x, y, z\}$ by, for example, one or multiple wearable AHRSs. Providing the orientation for each body part as quaternions, the 3D engine orients the body parts of the model skeleton accordingly and renders the mesh at the right position. However, because the body parts in a human body are connected to each other, the movement of a certain body section can be registered by multiple sensors, even on different parts. For example, lifting a straight arm would affect all the sensors on the limb, but in the 3D model is enough to animate the upper arm vector in order for the lower arm and hand vectors to follow.

For this reason, the absolute quaternions provided by sensors, as the BlueSense for example, must be pre-processed. Therefore, for each body part, we need to compute its relative orientation with respect to the previous part in the hierarchy. The relative orientation q_i for a body part i is computed as:

$$q_i = q_{i-1}^* \cdot q_{s_i}$$

where q_{i-1}^* is the conjugate quaternion of the previous body part in the hierarchy and q_{s_i} is the absolute quaternion provided by the sensor for the $i-th$ body part.

Once q_i is calculated, the 3D engine's native support of quaternions enable to apply the rotation to each segment of the skeleton.

The sensors quaternions q_{s_i} can be provided in real time forming a live motion tracking system (see Section 3.3.2) or off-line, as in the case study presented in the rest of this chapter.

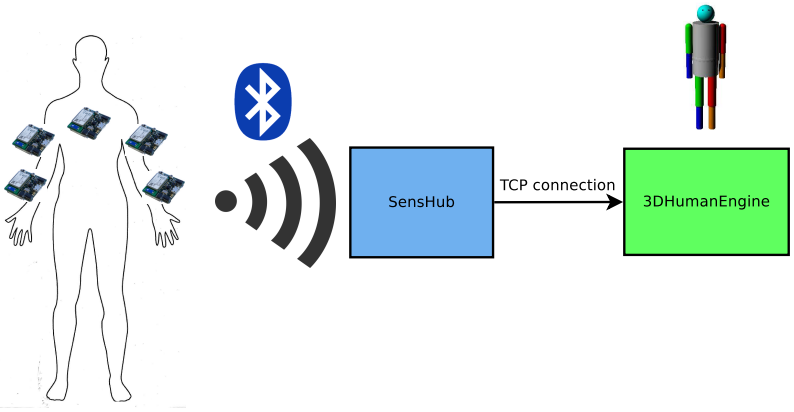


Figure 3.3.3: *Figure 4. Motion tracking system overview: the 3DHumanModel receives the data as a line of text through a TCP connection. Then it parses this text in order to extract the quaternions to animate the 3D model. At every rendering cycle, the 3D model is updated with the most recent orientation data.*

3.3.2 Motion tracking system

We created a motion tracking system that can animate our 3D model using a variable number of BlueSense as trackers. The motion tracking system is made of two applications and it is presented in Figure 3.3.3. First, the sensors data are collected using Bluetooth by the specifically developed application called SenseHub. This application: i) receives the sensors data, ii) synchronize them according with the timestamps of each sample, iii) forward them to the 3D model as a single line of text through a TCP connection. Then, the 3D model uses a TCP receiver in order to read the incoming line of text, unpack it and then animate the human model. The TCP connection allows the SenseHub and the 3D model to be on two different devices, potentially even in two different locations.

The 3D model can also be animated off-line: in this case, the data recorded by the sensors must be provided through a single file loaded at launch. This modality was demonstrated to be useful, for example, for privacy preserving annotation.

3.4 Privacy preserving annotation

3.4.1 Experimental setup

We deployed our 3D human model and motion tracking system in a off-line manner in order to study to which extent it can be used for privacy preserving annotations. For the experiments, we created three custom built interfaces around our 3D human model, as shown in Figure 3.4.1. The engine allows the users to rotate the camera around the model

and zoom in and out to better observe and evaluate each action.

To animate the model, we used the data provided by 5 Inertial Measurement Units (IMUs) placed on the upper arms, on the lower arms and on the torso provided by the Opportunity Dataset [48]. This dataset comprises a rich set of 17 naturalistic activities recorded in a kitchen environment. The activities are:

- Open and close two different doors;
- Open and close three drawers at three different heights;
- Open and close a dishwasher;
- Open and close a fridge;
- Clean a table;
- Drink from a cup;
- Toggle a switch.

This dataset consist of inertial data about the absolute orientation of each limb during the session. These data have been recorded using a set of XSens MTx inertial sensors [62]. For our experiments we used the "Drill" run subset. This subset consist of a fixed sequence of 17 actions repeated consecutively for about 20 minutes. Due to the absence of the environment in the 3D engine, we decided to join some of the similar labels (e.g. interacting with drawers at different heights is combined). "Open" and "Close" are considered different actions. From the initial 17 activities in the Opportunity dataset, we obtain the 11 activities shown in the Table 3.4.1, together with their average length in seconds.

We performed three experiments. The participants in the experiments were told that they would see a 3D model of a person performing typical activities in a kitchen. The participants were not given the list of activities at first. Essentially they have to "guess" from the animation of the model which activity may be undertaken. In the first experiment, a 15 minutes animation is played by the model. During this animation each participant must press the space-bar every time he/she notices something that he/she considers interesting and/or recognizable in the model's movements. It is up to the participant to decide what they consider "interesting". This experiment is used to evaluate the capability of the users to segment the activities using the model. The interface used during this test is shown in Figure 3.4.1.

Activity	sec
Open Fridge	2.450
Clean Table	3.967
Close Fridge	2.508
Drink from Cup	6.268
Open Dishwasher	2.958
Open Door	3.214
Close Dishwasher	2.862
Open Drawer	2.363
Close Door	3.378
Toggle Switch	1.295
Close Drawer	2.207
Average	3.043

Table 3.4.1: *List of activities and their respective average length in seconds. The last row shows the average length for all the activities.*

In the second and in the third experiment, a set of short animations of the body model are shown to the participant where the model performs exactly one of the 11 possible activities. For each of the 11 activities we showed the animations of 4 activity instances picked randomly from the dataset. In this way, we show to the user a random but balanced set of activities. The set of 44 short animations has been showed to the tester in a random order. This set was different for each participant.

The second experiment is carried out to investigate to which extent the application of our model fits an "open ended" scenario. In this experiment, the task of the users is to insert a short label for each animation. We developed the interface displayed in Figure 3.4.1b in order to allow the user to enter the label. This interface was showed after each animation of the set. The user had no time limit to enter the label. After he/she confirmed the inserted label, the next animation in the set is played.

The last experiment is useful to test the ability to annotate using a 3D model in a more traditional scenario. The system shows a push button for each of the 11 predefined activities. The participants must select which activity they think was performed by the model by pressing the corresponding push button with the mouse. The buttons are shown

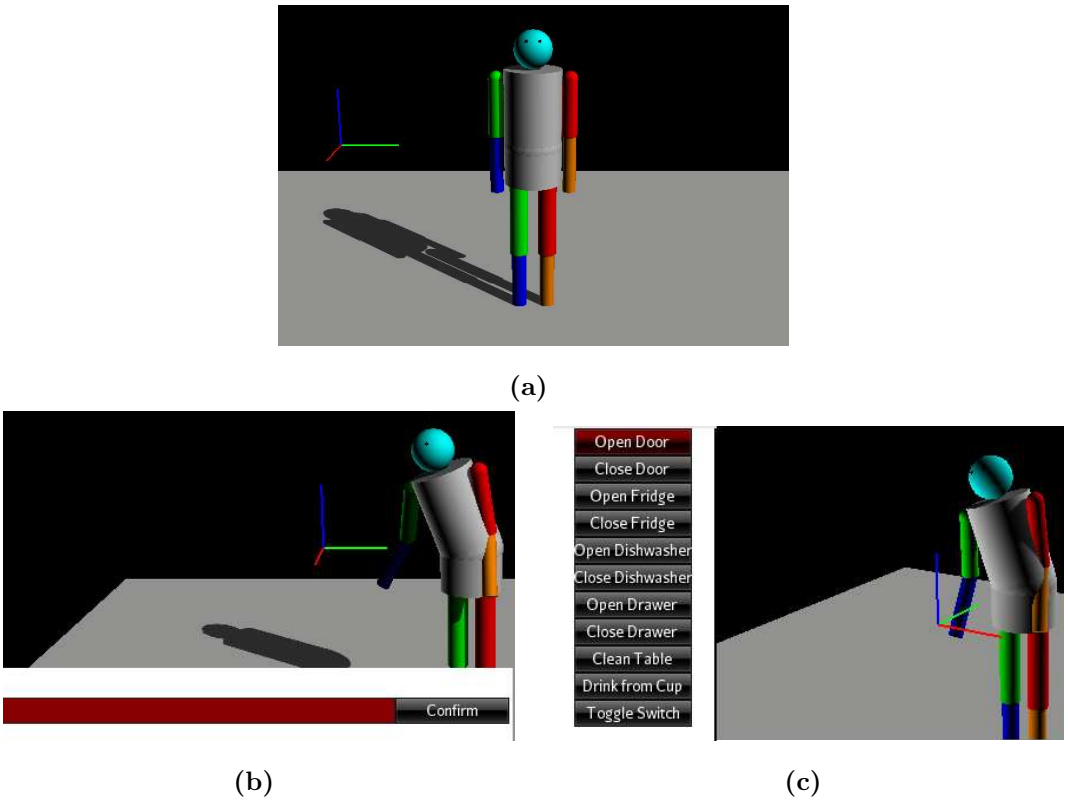


Figure 3.4.1: *Interfaces developed for the experiments. At the top, the interface developed for the Experiment 1 where the user must press a button to point out when something "interesting" happened to the model. No output is presented to the user when he/she press the button. On the bottom left, the interface proposed for Experiment 2 to let the user to insert the label manually. At the bottom right, the set of buttons that the user can click to select an annotation for the activity in Experiment 3.*

at the end of each short animation without any limit of time for the users. In Figure 3.4.1c are shown the buttons. After he/she selected a label, the next animation in the set is played. This corresponds to the traditional annotation approach where a pre-defined list of activities are annotated.

The experiment is carried out with 6 people, that are unaware about the dataset and the set of labels until the last experiment. The setup of the experiment is shown in Figure 3.4.2. All the participants performed the experiments in the same order and individually. They were instructed before each test as to what they would have to do next, in order to not influence the next phase of the experiment.

Figure 3.4.2: *Setup of the experiment.*

3.4.2 Results

Every experiment is designed to test a specific step or a different scenario of the annotation. With the first experiment, we aim to evaluate the capability of the users to segment the activities. In Figure 3.4.3, we show the results of the segmentation experiments for a subset of the dataset. The first row of the figure represents the distribution of the activities throughout the first 3 minutes of the experiment. The next 6 rows show the events pointed out by each participant. Every vertical line is an event. It allows to compare the distribution of the event recorded by each participant and the actual distribution of the activities.

As the experiment left the participants free to decide what they consider "interesting", we observe a large variations in the frequency of the events recorded by each participant. This may be explained because some participants tent to point out longer actions while other pointed out more shorter task. For example, the participant 3 recorded less events

User	TP	FN	TN	FP	Acc.
1	70	57	113	4	0.75
2	58	69	97	20	0.64
3	34	93	110	7	0.59
4	92	35	98	19	0.78
5	95	32	107	10	0.83
6	45	82	103	14	0.61

Table 3.4.2: *Test results for the first experiment. The true positive (TP) and the false positive (FP) do not keep in count if the user pointed out more events for the same activity/pause. Despite this affects also the accuracy, it can be used to better understand the results on a quality level. The total number of activities played by the model during the Test 1 is 127, with 117 pauses between some of the activities. A pause is intended as a moment between two actions, where the model is not doing anything "interesting".*

than the participant 5.

In Table 3.4.2, we show the number of activities made by the model and correctly pointed out (true positive), the number of pauses (intended as an interval in the animation where the model is not doing anything) correctly not pointed out (true negative), the number of pauses that the user identifies incorrectly as an activity in the model (false positive) and the number of activities made by the model and incorrectly ignored by the user (false negative). Multiple events pointed out by a user during the same activity or the same pause (true positive or false positive) are ignored. This has been made in order to not false the accuracy: in fact, counting multiple true positive or multiple false positive for the same instance of an activity would have introduced a bias.

In Figure 3.4.4, we present a cross-correlogram that shows the distribution of the delays between the pressing of the button by the user and the closest timestamp of the end of an activity. The distribution appears centered close to 0. This indicates that the participants actually recognized some movements in the model. Most of the delays are between -5 and +5 seconds from the closest activities. As the average length of activities is 3.043 sec (Table 3.4.1), a delay between -5 and +5 seconds is not enough to guarantee a correct segmentation.

The second experiment aims to evaluate the application of our model in the "open-ended" scenario. In Figure 3.4.5 we show the tag clouds of the words entered to describe the activities by all users for each label. We noticed that most of the participants mistake

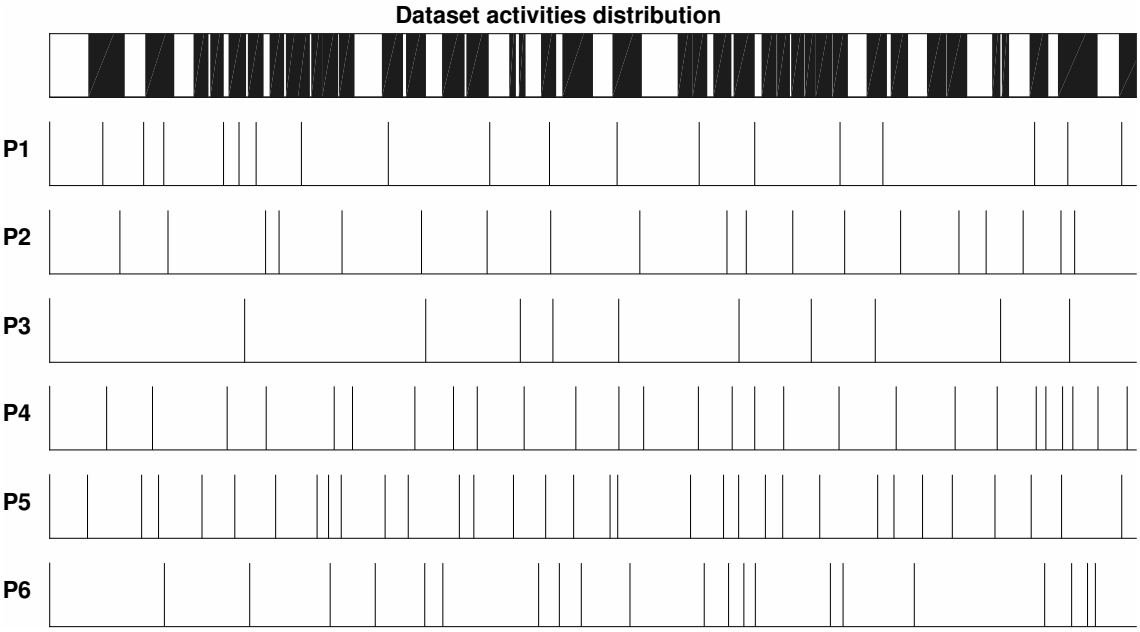


Figure 3.4.3: *Segmentation results: this figure represents a subset of the dataset of the first experiment (ca. 3 minutes). The first row represents the distribution of the activities during this interval of time. The 6 rows below indicate the events pointed out by each participant are displayed.*

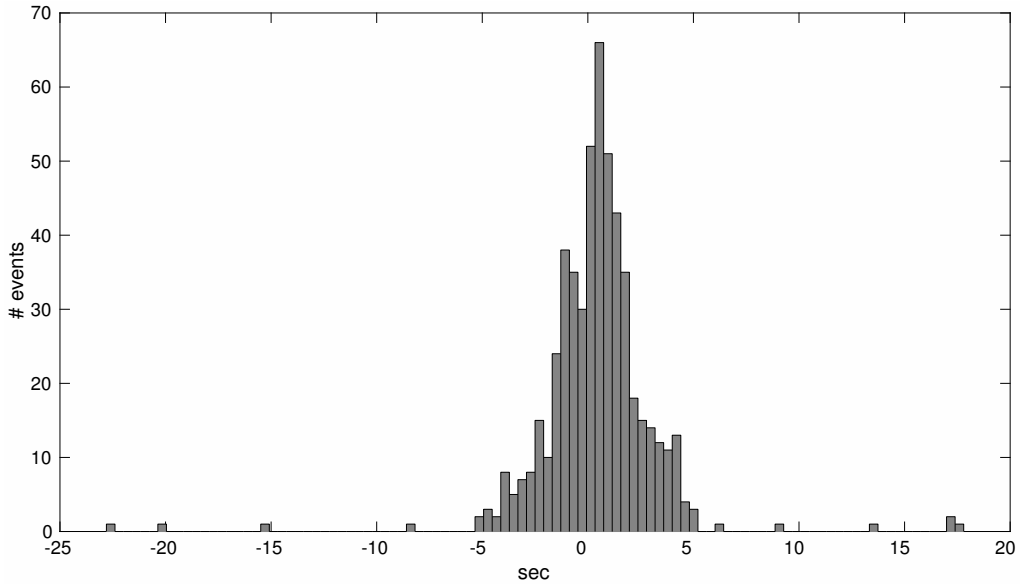


Figure 3.4.4: *Distribution of the delays between the events pointed out by the users and the closest end of an activity. A positive delay means that the user pointed out an event after the closest end of the activity. Instead, a negative delay means that the user pointed out an event too early in respect to the closest end of the activity (i.e. before the activity is actually completed).*

(a) Open door **(b)** Close door **(c)** Open Fridge

(d) Close Fridge **(e)** Open Dishwasher **(f)** Close Dishwasher

(g) Open Drawer **(h)** Close Drawer **(i)** Clean Table

(j) Drink from Cup **(k)** Toggle Switch

Figure 3.4.5: *Tag cloud of words used by testers in the open-ended experiment for each activity. The number of occurrences of a word is shown by its size. The colours are only used to make more clear the distinction between the words.*

the dishwasher with the oven: this is quite normal because both appliances can have the same kind of door. Moreover, the user interface is missing any rendering of the kitchen environment and no information is given to the user about the appliances and about the furniture at this stage. However the participants correctly identified the difference between open and close the dishwasher. Interesting, "drinking" is correctly recognized by all the participants.

In the last experiment, we investigate the traditional scenario where the user should annotate a dataset already segmented, choosing the correct label from the predefined "closed set" of 11 activities indicated in Table 3.4.1. The results are presented in the

Figure 3.4.6 using a confusion matrix between the choices of the users and the actual labels of the data.

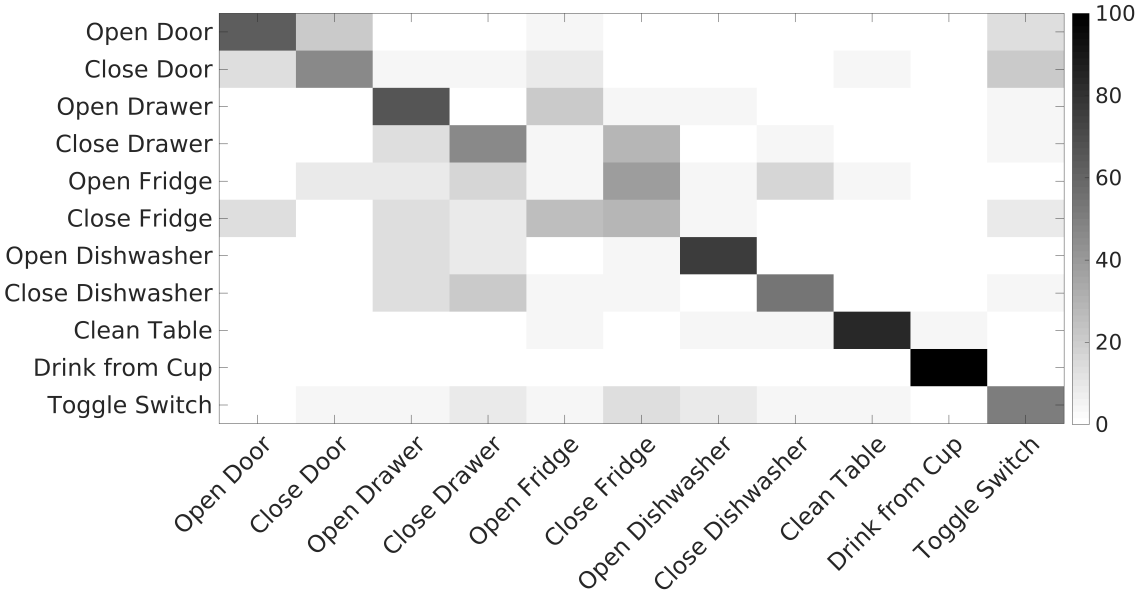


Figure 3.4.6: *Confusion matrix between labels chosen by users and actual labels.*

Some activities are correctly recognized by the user: "Drink from Cup" reaches an accuracy close to 100%. On the other hand, there are actions that are almost never identified: "Open the Fridge" is the activity identified with the lowest accuracy (4.2%). The main cause is likely the absence of any point of reference for the environment and the fact that the fridge was of small size. For this reason, the action of opening and closing can be easily confused with other actions applied to the same height, such as "Open a drawer". Moreover, the confusion between "Open/Close the door" is due to the lack of information about the direction of opening and closing of the door.

Finally, the participants reached an average accuracy of 56% in the controlled labelling experiment using our system.

3.4.3 Discussion

Our experiment revealed that a 3D human model can be used for activity annotation preserving the privacy of the user, but it would require some improvements.

About the segmentation of the activities, which is studied in the first experiments, the obtained results showed that further analyses are required such as the capability of point out the duration of the action. Allowing the users to identify the beginning and the

end of an activity could improve the accuracy in this step. It can be also important to specify to the user the granularity of the action. As we noticed during the experiment, the main trouble for the testers was: "What should I consider as an action to point out?". Answering this question can depend on the specific application scenario of each dataset: in some cases an action could be a simple movement as e.g. "move the right arm up". In others scenarios instead, it could be important to identify more complex activities as "make a sandwich", "prepare a coffee", etc.

Moreover, during our experiments, we pointed out that a main issue is the lack of the environment in the scene. It can be difficult for the users to recognize the wide set of possible activities without knowing the position of the objects and of the furniture in the environment. A typical example of this is the mistake between the opening of the fridge and the opening of the drawer: these two movements appear similar when reproduced with a simple model such as ours. This confusion between movements that appear similar is more observable in the "open-ended" annotation: in fact in this scenario it occurs that users identify correctly the movements (opening and closing), but they annotate the task with a different object whom those movements are applied to (the dishwasher mistaken with the oven).

Some improvements should be also applied to the model itself. In this first implementation, we used only basic solids to create the human figure: this brought some difficulties for the users to recognize actions made by short and limited movements. An example can be the toggling of the switch: in this case, the absence of the hands made it tricky to identify it. For this reason, we should explore whether a more realistic human model could improve the accuracy of the annotation.

In order to improve the accuracy of the movements performed by the model, a larger number of sensors can be a solution. In our experiments the data animate only two parts of each upper limb and the torso, but the model has been developed to be animated with a maximum of 12 sensors. The data from all these sensors can be also applied on the hands, on the legs and on the head. Furthermore, the software can be used with many different datasets passing specific parameters at start-up. The only requirement is that the dataset should contain IMU data for each body part the users want to animate. This can be a limitation: in fact, it can be difficult to use this system with those datasets already recorded and where the IMUs are placed only on few body parts, not allowing the model to reproduce all the movements correctly. Instead, this system can be a valid choice for new recorded dataset. If future work shows higher annotation accuracy, it might even

be feasible for some scenarios to remove cameras altogether. This could lead to cost and time savings because researchers will not need neither equipments to record the videos nor additional time to preprocessing them.

It is the first time that annotation using a 3D model has been proposed. Even though an average accuracy of 56% may be insufficient for ground truth, this system could exploit decision fusion among multiple annotators (e.g.majority voting) and filters, to improve accuracy as already done in [50] for video annotation. In this scenario, it would be also interesting to study the performance of a classifier trained using the data annotated using these algorithms.

3.4.4 Conclusion

We raise the need to create a privacy preserving annotation system, in order to speed up the process of labelling dataset using cheap labour and crowdsourcing. In these cases, a video can not be used due to lack of anonymity of the user recorded in the video itself.

We study to which extent a 3D human model animated in a virtual environment using the data from inertial sensors can be used to annotate the dataset. We developed the model in order to be used as motion tracking system in both real-time and off-line applications.

In order to evaluate its applicability to privacy preserving annotations, we animated a model using a prior labelled dataset. We then compared the annotation collected with our model and the actual labels of the dataset. We performed three analyses: a first one to study the capability of the user to recognize the activities done by the model and correctly segment the dataset. This is effectively the first step during the annotation process. The second experiment is designed to analyse to which extent the application of our annotation system fits in a "open-ended" scenario where the user can choose freely the label for each action. In the last test we investigate the accuracy of the annotation when a set of possible choices are given to the users, reaching a high level of accuracy for some specific actions and manifold results for others.

From the experiments, the results are threefold. The segmentation experiment requires further analysis in order to better evaluate to which extent a 3D human model can be actually used for this task. The obtained results are not sufficient to give a strong positive judgment. The "open-ended" annotation can be used but only when the dataset consist of actions that look very clear when reproduced by the model (e.g.drinking). For those actions where movements are limited and short, the accuracy with our system drops. Instead, using a "closed set" of annotations, our system allows users to reach an average

accuracy of 56% also using very similar actions and with only 6 people. In this scenario, adding more people and applying decision fusion algorithms and filters for bad taggers, our system could become an actual choice to annotate data preserving the privacy of the subject in the dataset.

Chapter 4

Action recognition using Template Matching

Once the data are collected and annotated, the recognition of movements and activities can be performed. This chapter provides an overview of action recognition using TMM. Starting from general concepts of template matching and activity recognition, we presents the action recognition pipeline used throughout this thesis. We also presents the datasets of actions and activities we gathered from the literature in order to evaluate our methods.

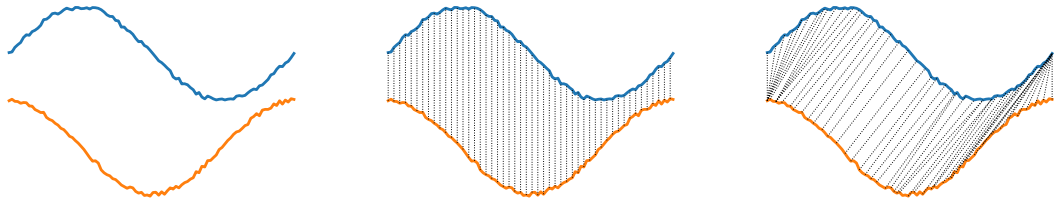
4.1 Template Matching Methods

Template Matching Methods (TMMs) are algorithms that provide a distance measure between two temporal sequences of data, called time series. The time series can represent a wide variety of data type. TMMs have been developed for and applied to speech recognition [63], pattern recognition in images [64], time series indexing [65] and activity recognition from wearable sensors [26].

In the following, we describe the mechanism of a TMM with an example of application to activity and movement recognition. Let us consider the problem of recognising beach volleyball serve movements performed by players wearing an accelerometer sensors on the wrist of their dominant hand. The player performs many serves during a training. In this situation, the accelerometer provides a time series of the acceleration of the hand when moving[1]. Therefore, a continuous stream of data originates from the wrist sensor. We refer to this time series data as $S = s_1, s_2, \ldots, s_M$.

[1] Accelerometers generally provide 3 time series, one for each component x, y, and z. In this example, we can consider a single channel accelerometer for conciseness and simplicity.

We wish to individually recognize 4 typical serves (e.g. long/short flat/top spin serves, see Chapter 2 for more details). Because we are only interested in the serve movements, the continuous series may be segmented in a multitude of shorter *segments* delimited by the interval between the player getting in position to serve and the landing of the ball after the serve. In order to recognise a serve, its time series is compared using TMM to the accelerometer readings of each single prototypical serve. We refer to these prototypes as the *templates*. We define a template as $T = t_0, t_1, \ldots, t_N$. As there are several prototypical serves - chiefly one for each of the 4 kind of serve - we define a set of templates $\mathbb{T} = \{T_0, T_1, \ldots, T_O\}$ each representing a prototypical serve movement.



(a) **(b)** **(c)**

Figure 4.1.1: *Two time series which are similar but misaligned 4.1.1a can be compared sample by sample ("exact match") in 4.1.1b, or they can be compared employing dynamic time warping methods 4.1.1c which looks for the best alignment between the samples of the two time series.*

Most TMMs tolerates some degree of accepted variability between T and S. This can be due to misalignment of two time series on the time axis or to different length of T and S (Figure 4.1.1). For example, two instances of the same serve movements can be performed at slightly different speed, resulting in producing two sensor signals of similar shape, but different lengths. In this case, the two signals must be recognize as similar movements despite the variations. This variation is called "time warping". Exploiting the concept of "time warping" opens up the possibility to employ shorter templates and therefore reduce the computational cost of the TMMs, as this is directly related to the template size ($\mathcal{O}(|T|)$).

A TMM is defined as $D_{T,S} = \mho(T, S, C)$[2] and it computes $D_{T,S}$, a similarity measure between T and S, parametrized by the set of parameters C. The use of the parameters C depends on the TMM and it is used in case of "time warping".

The distance $D_{T,S}$ can optionally be compared to a threshold V when a crisp detection

[2]This definition is applicable e.g. to Dynamic Time Warping, Euclidean distance, Edit distance, and others.

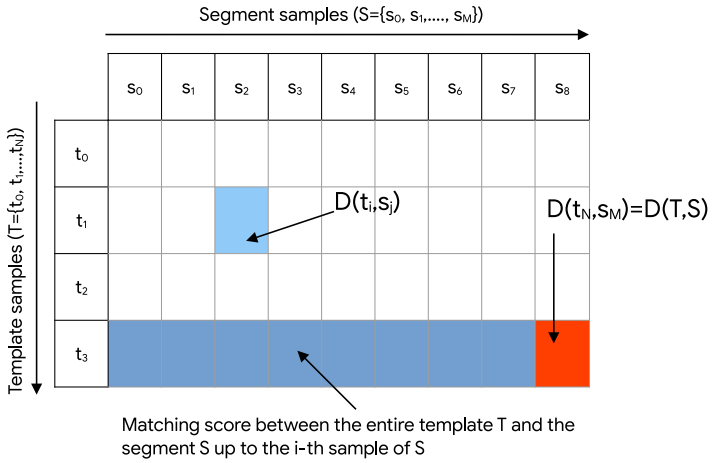


Figure 4.1.2: *A TMM computes the distance $D(t_i, s_j)$ recursively using the equation in Table 4.1.1. The distances calculated between every sample i of the template T and every sample j of the segment S can be visually displayed as a matrix. The total distance between T and S correspond to the distance computed between the last sample of the template T, t_N, and the last sample of the segment S, s_M, (red square).*

is required.

TMMs are generally used to match multiple templates $\mathbb{T} = \{T_1, \ldots, T_O\}$ with multiple times series $\mathbb{S} = \{S_1, \ldots, S_Q\}$. In this case, C and V can be extended to two sets, respectively $\mathbb{C} = \{C_1, \ldots, C_O\}$ and $\mathbb{V} = \{V_1, \ldots, V_O\}$, as parameters may be optimized for each template T_i individually.

The most common TMMs are summarised in Table 4.1.1. Most of them are based on dynamic programming and therefore defined through recursive equations. With the exception of the Euclidean distance, for all the other methods, the final $D_{T,S}$ is computed recursively as $D_{T,S} = \mho(t_N, s_M, C)$, where t_N and s_M are the last sample of T and S respectively (see Figure 4.1.2).

4.1.1 TMM for classification

TMMs provide of a similarity measure $D_{T,S}$ between the time series in $\mathbb{T}$ and S. This similarity can be a distance, the lower the better, or a matching score, the higher the better.

An important application of the similarity measure is time series classification. In this case, a ground truth label y_i is assigned to each T_i. The classifier h aims at assigning the correct label $\hat{y_i}$ to S as[4]:

[3] $w = 1$ and omitted for standard DTW. If $w \neq 1$ for any sample, the method is called Weighted-DTW.

[4] argmin is used for distance, argmax for matching score

TMM	D/S	D.P.	W	C.C.	Equation	Params.	Ref.
Exact match	Distance	No	No	$\mathcal{O}(N)$	$D_{T,S} = \left(\sum_{i=1}^{n} \lvert t_i - s_i \rvert^l\right)^{1/l}$ (4.1)	-	-
DTW		Yes	Yes	$\mathcal{O}(N \cdot M)$	$D_{t_i,s_j} = d_{t_i,s_j} + w \cdot \min \begin{cases} D(t_{i-1}, s_j) \\ D(t_i, s_{j-1}) \\ D(t_{i-1}, s_{j-1}) \end{cases}$ (4.2)	$w^{(3)}$	[63, 66, 67]
Edit distance					$D_{t_i,s_j} = \begin{cases} \max(i,j) & \text{if} \quad min(i,j) = 0 \\ \min \begin{cases} D_{t_{i-1},s_j} + D \\ D_{t_i,s_{j-1}} + I \\ D_{t_{i-1},s_{j-1}} + S_{(t_i \neq s_j)} \end{cases} & \text{otherwise} \end{cases}$ (4.3)	D, I, S	[68, 69]
WLCSS	Similarity				$D_{t_i,s_j} = \begin{cases} 0 & \text{if} \quad i \leq 0 \; or \; j \leq 0 \\ D_{t_{j-1},s_{i-1}} + R & \text{if} \quad f(t_i,s_j) \leq \epsilon \\ max \begin{cases} D_{t_{i-1},s_{j-1}} - P \cdot f(t_i,s_j) \\ D_{t_{i-1},s_j} - P \cdot f(t_i,s_j) \\ D_{t_i,s_{j-1}} - P \cdot f(t_i,s_j) \end{cases} & \text{if} \quad f(t_i,s_j) > \epsilon \end{cases}$ (4.4)	R, P, ϵ	[26]

Table 4.1.1: *TMM overview. For each method the following information are reported: i) whether the TMM calculate a distance (lower is better) or a matching score (higher is better) (**D/S**); ii) if it uses dynamic programming, meaning that the distance between T and S is actually the distance between their last samples $D(T,S) = D(t_N, s_M)$ (**D.P.**); iii) whether it allows warping between time series (**W**); iv) the computational complexity (**C.C.**) v) the equation (**Equation**); vi) the parameters specific of each TMM, when present (**Params.**); vii) the reference for the method, when possible (**Ref.**).*

$$h(S) = \underset{y_i}{\operatorname{argmax}}\, D_{T_i,S}$$

The method is called 1 Nearest Neighbour (1-NN) [70] and it assigns $\hat{y}_i$ to S as the label y_i of the template T_i most similar to S. In this case, the classification is named *segmented* (Figure 4.1.3).

In some classification tasks, it is possible for a S to be too dissimilar from any T_i. S is too dissimilar from T_i, if $D(T_i, S) < V_i$ for any T_i in $\mathbb{T}$. In this case, a $\hat{y} = 0$ often referred as *NULL class*, is assigned to S. Considering the aforementioned beach volleyball serve example, this can be the case of a movement of the player that is too distinct from any T_i to be recognized as a serve.

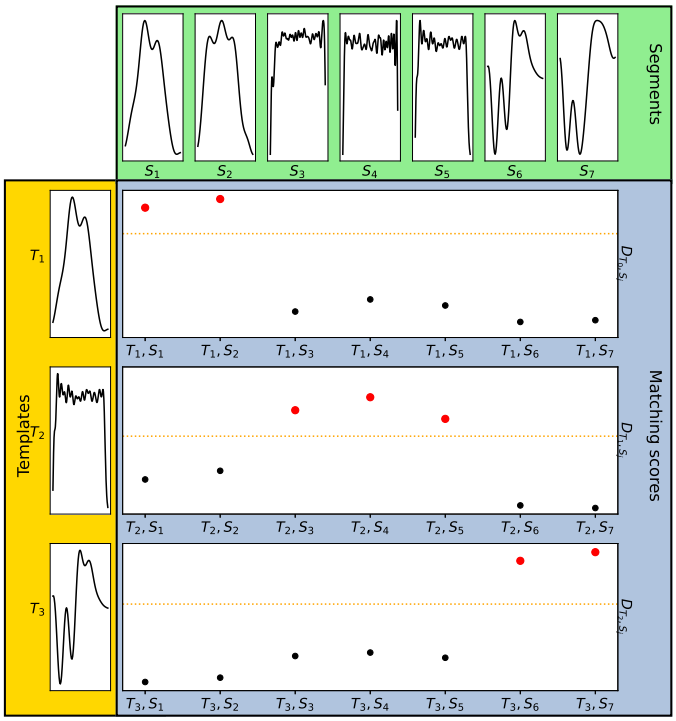


Figure 4.1.3: *Consider 7 time series segments $S_1, \ldots, S_7$ (which may correspond to time series generated by the accelerometer in the serve recognition problem). These 7 time series (top row) belong to 3 classes (e.g. 3 different kind of serve). For each of these classes, a prototype is chosen as template, T_1, T_2, T_3 (first left column). The matching scores D_{T_i,S_j} are then computed for every S_j in $S_1, \ldots, S_7$ for their classification (central 3 plots). In this case, WLCSS is used as TMM. The red dots correspond to the classification using 1-NN based on the matching scores. The black dots are a mismatch. A possible threshold V is reported as dashed line for each T_i, but not used in this case.*

TMM can also be used for *streaming* recognition when S is a continuous stream of

data. In the beach volleyball example, streaming recognition could be used to continuously recognize serves during a match in which the player wears the sensor that continuously samples acceleration data. However, in this work, we focus on the segmented recognition as first step towards improving continuous skill assessment. Also, given our contribution of a training method for TMMs, segmented recognition allows a direct comparison with other TMM training methods. More on this matter is discussed sections 5.6 and 6.4. Unless specified otherwise, any time series classification referred in this thesis is considered segmented.

4.1.2 Warping Longest Common Subsequence

The Warping Longest Common Subsequence (WLCSS) [26] is an algorithm developed for template matching in real-time applications that has been developed to be robust against noisy data and variation of the movements execution speed (warping). WLCSS was demonstrated to be suitable for real-time embedded implementation on low-power nodes which are fundamental in the field of wearable computing. This is possible thanks to its low memory footprint, as well as to the deployment of only compare, sum and product operations [71].

Using dynamic programming, the algorithm computes a matching score between a template T and a segment or stream S, updating it at every new sample of the stream. It can be used for movement recognition as it can handle actions performed with variation in their speed of execution. This is achieved by three parameters: reward (R), penalty (P) and acceptance distance (ϵ). Using these parameters, WLCSS computes the matching score D between the i-th sample of the stream S and the j-th sample of template T, according to equation 4.4. D is increased recursively by the reward R when the distance between the two samples ($f(t_i, s_j)$) is below the acceptance threshold ϵ. Otherwise, it finds the best warping (contraction, dilation, or alignment as-is) between T and S. In this case, D is decrease by the penalty P proportional to the distance between the samples ($P \cdot f(t_i, s_j)$). An example of WLCSS calculated between a template and a segment is displayed in Figure 4.1.4.

The values of R, P, ϵ and V, as well as the templates $\mathbb{T}$ needed for the matching must be found or generated during the training phase. We present an evolutionary training algorithm as main contribution of this thesis in chapter 5 and 6.

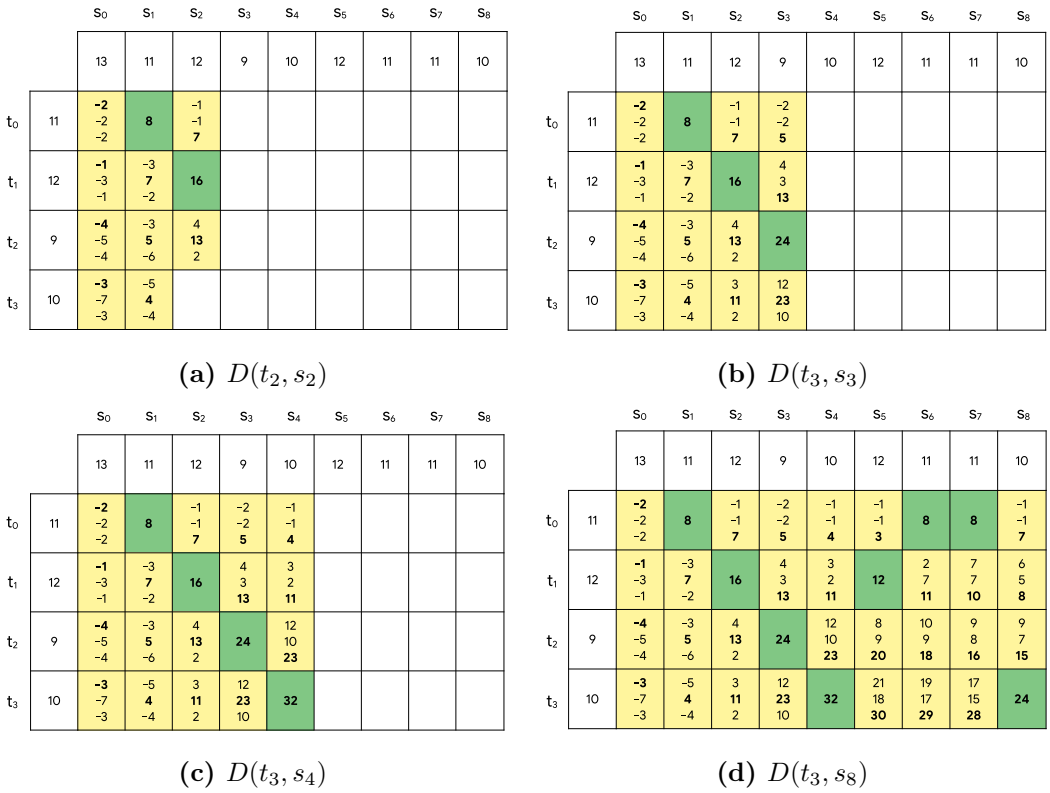


Figure 4.1.4: *Example of computation of matching score using WLCSS (see Eq.4.4), with parameters* $R = 8, P = 1, \epsilon = 0$ *and* $f(t_i, s_j) = |s_j - t_i|$. *The yellow cells indicate when a mismatch between* t_i *and* s_j *happens* ($f(t_i, s_j) > \epsilon$). *The green cells represent a match between* t_i *and* s_j ($f(t_i, s_j) \leq \epsilon$). *In this example, the template is defined as* $T = \{11, 12, 9, 10\}$, *while* $S = \{13, 11, 12, 9, 10, 12, 11, 11, 10\}$. *The figure presents 4 intermediate steps of the matching score calculation using WLCSS. Fig. 4.1.4a displays the matching score calculated up to* $D(t_2, s_2)$. *In this case,* $f(t_2, s_2) > 0$ *indicating a mismatch. Therefore, three possible values are calculated:* $D_{t_{i-1},s_{j-1}} - P \cdot f(t_i, s_j) = 4$, $D_{t_{i-1},s_j} - P \cdot f(t_i, s_j) = 13$, $D_{t_i,s_{j-1}} - P \cdot f(t_i, s_j) = 2$. *The maximum value is then selected as the matching score (value in bold), in this case* $D(t_2, s_2) = 13$. *Similarly,* $D(t_3, S_3)$ *is calculate recursively as 23. Fig. 4.1.4c shows an example of matching between* t_3 *and* s_4*: in this case* $D(t_3, s_4)$ *is calculated as* $D_{t_{j-1},s_{i-1}} + R$ *and therefore* $D(t_3, s_4) = 32$. *Finally, Fig. 4.1.4d shows the complete calculations for* $D(T, S) = D(t_N, s_M)$ *which in this example is* $D(t_3, s_8) = 24$.

4.2 Activity recognition using TMM

Multiple steps must be considered when a TMM is applied to the classification of movements from wearable sensors data. These steps compose a *recognition pipeline* that includes the sensors sampling, the pre-processing of the sampled data, the distance calculation using the TMM and the classification of the segments.

Figure 4.2.1 shows our recognition pipeline, from sensor sampling to the classification, using a WLCSS as TMM. A stream of data is sampled using one or multiple sensors such as, for example, accelerometers and gyroscopes. The sensors signal can be subject to several different pre-processing steps to remove noise from the data and therefore improve recognition [72]. In our pipeline, unless differently specified, a low-pass filter is employed.

In addition to filtering, the pre-processing step can be useful to reduce dimensionality of the data. Wearable sensors can provide multi-channel data that generally are not suitable for TMM that are developed for single channel time series. In our pipeline, we deployed two different procedures for dimensionality reduction each comprising multiple steps.

The first procedure is useful in case of raw data such acceleration and rate of rotation. The multiple channels, e.g. x, y and z components of an accelerometer, are merged for example by computing the magnitude of a signal as $\sqrt{x^2 + y^2 + z^2}$. The fused data is then downsampled in order to reduce the computational cost of the TMM. Finally, the downsampled signal is quantized to a fixed interval of values in order to reduce the dimensionality even further; in this case the dimensionality reduction is on the y-axis.

The second method for dimensionality reduction has been presented in [25] and it does not only allow the fusion of multiple channels from one sensor, but it also merges data from multiple sensors by encoding complex movements as strings. When available, this method uses the 3D orientation data from multiple inertial wearable sensors by computing the 3D trajectory of a hand, or any body part, when performing the movements. This 3D trajectory is then sampled temporally or spatially to produce a series of 3D vectors. These 3D vectors are then translated in symbols by using the codebook of 3D unit vectors displayed in Figure 4.2.2a. The translation of a 3D vector is performed by finding its closest 3D unit vector in the codebook using the angular distance. Since a symbol is associated to every 3D unit vector in the codebook, this symbols can be used to form the string representing the trajectory time series. In this case, the encoding of the trajectory correspond to the quantization of the first method.

A similar approach can be applied to 2D trajectories as well, by using a 2D codebook as displayed in Figure 4.2.2b. An example of 2D trajectories can be the trace produced

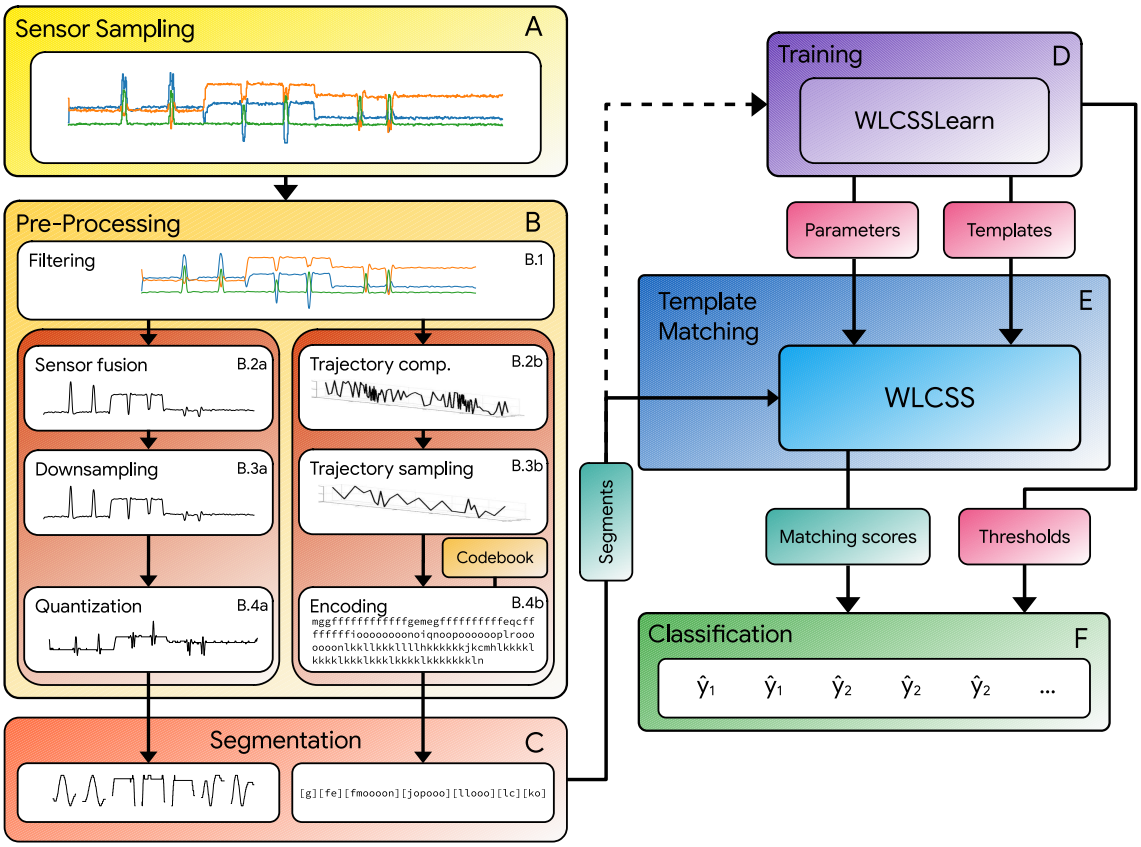
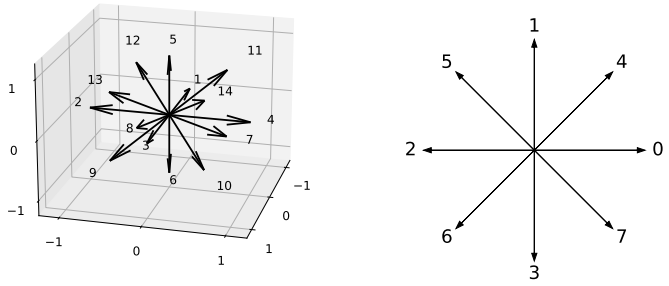


Figure 4.2.1: *Har pipeline example employing WLCSS for segmented classification of activities. A signal is sampled using, in this case, a 3-channel sensor, with x, y and z components (A). Then, the pipeline proceeds with a pre-processing step (B) which comprises: i) signal filtering (B.1), using a low pass filter for example, for every component; ii) Then according with the desired dimensionality reduction method, two ways of processing the data are possible. The first include sensor fusion of the 3 channels (B.2a), downsampling of the signal (B.3a), and quantization (B.4a). The second method includes the computation of 3D (or 2D) trajectories (B.2b), the trajectory downsampling, temporally or spatially, (B.3b), and the encoding of this samples using a codebook (B.4b). iii) Finally, the quantized/encoded signal is segmented with some heuristic (C), for example using a temporal sliding window for the classification, and the ground truth labels of the samples for the training. iv) Part of the segments can be used for the training (D) of parameters $\mathbb{C}$, the templates $\mathbb{T}$ and the thresholds $\mathbb{V}$ for the classification. v) successively, WLCSS is used to computed the matching scores (E). vi) The classification is then performed comparing the matching scores and the thresholds $\mathbb{V}$ (F).*

by a finger on a touch-screen. See Section 4.3 for more details.

Successively, independently from the quantization/encoding, the signal is clipped into



(a) 3D **(b)** 2D

Figure 4.2.2: *3D and 2D codebooks used for the encoding of trajectories. The unitary vector are equally spaced.*

segments. The segmentation can be performed using different heuristics such as time windows or the ground truth labels. The latter requires an annotated dataset and it is used for training in order to select specific segments of interest.

The choice of the templates $\mathbb{T}$ and the sets of the parameters $\mathbb{C}$ is fundamental to maximize the classification performance. An adequate selection of templates may also reduce the computation required for pattern recognition. For example, selecting a single optimal template for each class decrease the computation when a 1-NN approach is used for the classification, compared to using all the available templates. Moreover, when a TMM allows warping between time series, a shorter template may be used reducing its complexity $\mathcal{O}(N \cdot M)$, by decreasing templates' length N.

For this reasons, part of the segments, if annotated, can be used for training as shown in Figure 4.2.1. Unlike e.g. back-propagation as a standard algorithm for neural network training [73], TMMs do not offer a standard training procedure that can be applied across different algorithms. Therefore, in this thesis we present our training method named WLCSSLearn.

Finally, once the matching costs $D(T_i, S_j)$ are computed between a segments S_j and the templates in $\mathbb{T}$, the pipeline proceed with the classification. The classifier h assigns the label $\hat{y}_i$ to S_j as the y_i of T_i using the following:

$$h(S_j) = \max\left(0, \underset{y_i}{\operatorname{argmax}} \frac{D(T_i, S_j) - V_i}{V_i}\right) = \hat{y}_i$$

One thresholds V_i per template T_i is used for detection and classification. With this classifier, it is possible for a S_j to be too dissimilar from any T_i and therefore be classified as NULL-class.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	0	15	31	15	15	15	9	9	21	21	9	9	21	21
2	15	0	15	31	15	15	21	9	9	21	21	9	9	21
3	31	15	0	15	15	15	21	21	9	9	21	21	9	9
4	15	31	15	0	15	15	9	21	21	9	9	21	21	9
5	15	15	15	15	0	31	21	21	21	21	9	9	9	9
6	15	15	15	15	31	0	9	9	9	9	21	21	21	21
7	9	21	21	9	21	9	0	12	19	12	12	19	31	19
8	9	9	21	21	21	9	12	0	12	19	19	12	19	31
9	21	9	9	21	21	9	19	12	0	12	31	19	12	19
10	21	21	9	9	21	9	12	19	12	0	19	31	19	12
11	9	21	21	9	9	21	12	19	31	19	0	12	19	12
12	9	9	21	21	9	21	19	12	19	31	12	0	12	19
13	21	9	9	21	9	21	31	19	12	19	19	12	0	12
14	21	21	9	9	9	21	19	31	19	12	12	19	12	0

Table 4.2.1: *Matrix of distances between symbols of the 3D codebook in Figure 4.2.2a.*

4.2.1 Template matching of encoded movements

Since the different pre-processing steps can produce different values for the segments samples, a TMM must be robust to the difference in the time series quantization and encodings. For WLCSS, this is achieved thanks to $f(t_i, s_j)$ (see Equation 4.4): in case of quantized time series, the $f(t_i, s_j)$ is often the L1-norm between the two samples $|t_i - s_j|$. However, it can be also be other distance metrics defined for the possible values of t_i and s_j. This enables WLCSS to be used with numeric time series as well as with signals encoded using specific alphabets for which the L1-norm between t_i and s_j could be meaningless (e.g. time series encoded as strings [25]).

We created two distance matrices for the 3D and 2D encodings respectively. Table 4.2.1 and 4.2.2 show these matrices. The distance between two encoded samples t_i and s_j respectively of value a and b is the element at position (a, b) of the matrix. The matrices are created based on the angular distance between two vectors in each codebook. The angular is then multiplied by 10 and rounded to integer values in order to be more easily integrable in embedded wearable platform that might not have support for float numbers.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	15	31	15	7	23	23	7
2	15	0	15	31	7	7	23	23
3	31	15	0	15	23	7	7	23
4	15	31	15	0	23	23	7	7
5	7	7	23	23	0	15	31	15
6	23	7	7	23	15	0	15	31
7	23	23	7	7	31	15	0	15
8	7	23	23	7	15	31	15	0

Table 4.2.2: *Matrix of distances between symbols of the 2D codebook in Figure 4.2.2b.*

4.3 Datasets

Several datasets for activites, gestures and movements recognition have been presented in the community, including a wide variety of sensor modalities [74]. In order to evaluate our training methods, in addition to the previously collected beach volleyball dataset (see Chapter 2), we selected 5 multi-modal datasets of movements coming from the field of wearable and ubiquitous computing (see Table 4.3.1). We chose these datasets because they include a variety of movements, sensors modalities and environments in which they are collected, they are pre-processed differently and they are well established in the community. This ensures a robust evaluation of our methods. In the following, we describe the pre-processing applied to each of these datasets and to the beach_volleyball dataset to be used in the rest of this work.

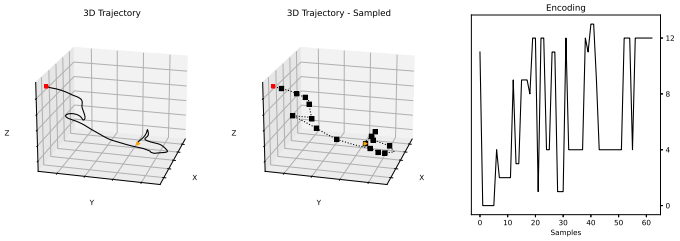
The **Skoda** dataset comprises actions performed in car manufacturing during a quality check routine [75]. The movements were performed by users wearing a set of inertial motion sensors on the back, on the upper arms, on the lower arms and on the hands. Each platform integrated a 9-axis inertial unit providing the device orientation. The hand coordinates are computed with respect to the torso position, at each timestep. The trajectory of the hand in then sampled at regular interval and encoded as in [25] to reduce dimensionality. The actions are thus represented as a sequence of symbols indicating the direction of movement of the hand. Each symbol is obtained using a codebook of predefined vectors in order to sample the displacement of the hand during the movement. We empirically chose a 15 vectors codebook (see Figure 4.2.2a). We selected actions that are commonly performed with the right hand (e.g. Adjusting the mirror for a left-hand driving car) or with both

Dataset	# of movements	|S|	Avg. Stream length	List of movements
skoda	8	308	57.19 ± 63.46	3001: open_hood, 3002: close_hood, 3003: open_trunk, 3005: close_trunk, 3013: mirror, 3014: check_trunk_gaps, 3018: open_swl, 3019: close_swl
opportunity_encoded	11	379	29.32 ± 13.98	406516: Open Door 1, 404516: Close Door 1, 406520: Open Fridge, 404520: Close Fridge, 406505: Open Dishwasher, 404505: Close Dishwasher, 406519: Open Drawer 1, 404519: Close Drawer 1, 408512: Clean Table, 407521: Drink from Cup, 405506: Toggle Switch
hci_guided	5	264	60.25 ± 6.12	49: Triangle, pointing up, 50: Square, 51: Circle, 52: Infinity, 53: Triangle, pointing down
skoda_mini	10	725	74.90 ± 20.81	48: write on notepad, 49: open hood, 50: close hood, 51: check gaps on the front door, 52: open left front door, 53: close left front door, 54: close both left door, 55: check trunk gaps, 56: open and close trunk, 57: check steering wheel
hci_table	26	583	54.51 ± 24.52	9: A, 10: B, 11: C, 12: D, 13: E, 14: F, 15: G, 16: H, 17: I, 18: J, 19: K, 20: L, 21: M, 22: N, 23: O, 24: P, 25: Q, 26: R, 27: S, 28: T, 29: U, 30: V, 31: W, 32: X, 33: Y, 34: Z

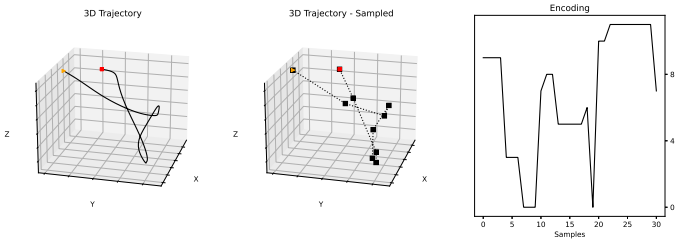
Table 4.3.1: *Movements datasets overview. The number of different movements, the number of segments, their average length and standard deviation is reported. Finally the lists of movements identified by a numeric identifier is presented.*

hands (e.g. Open/close hood). An example of this encoding for the action "Open hood (3001)" is shown in Figure 4.3.1a.

The **OPPORTUNITY** Activity Recognition Dataset [48] comprises activities performed in a kitchen environment. We use the "drill" part of the dataset which included 20 repetition of the same set of actions performed by participant. The motion capture system is similar to that in Skoda dataset as well as the processing of the trajectory data. We use the position of the right hand as it is the one used for majority of the actions. Some movements might be performed with the other hand, but this information is not available in the dataset. The set of actions used in this study is displayed in Table 4.3.1. An example of 3D encoding of the action "Open Dishwasher" is presented in Figure 4.3.1b.



(a) Skoda - "Open Hood"



(b) OPPORTUNITY - "Open Dishwasher"

Figure 4.3.1: *Pre-processing step for segments in Skoda 4.3.1a and OPPORTUNITY 4.3.1b datasets. The X-Y-Z coordinates of the hand (Skoda) and lower arm (OPPORTUNITY) are computed combining the 3D orientation data provided by the IMU sensors on the entire arm. Then, a trajectory is computed as sequence of 3D points in space (left). The orange and red points are respectively the start and the end point of the trajectory. Finally, this trajectory is sampled in time (center) and each vector between two sampling points is encoded using the 3D codebook 4.2.2a with symbol of the closest 3d vector (right).*

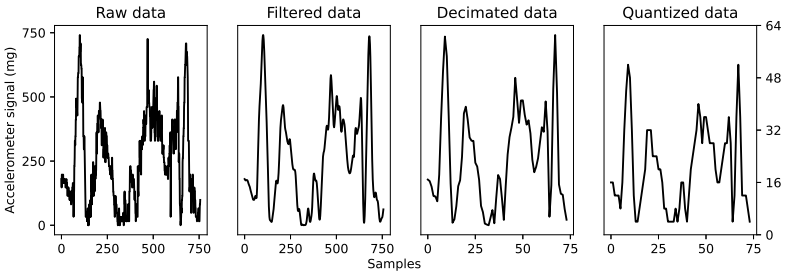
The **HCI** (guided) gestures dataset [76] comprises gestures performed by a user following a geometric shape place on a wall in front of the user. We can defined these as gestures since the movements included in the dataset are meant to communicate with an HCI system. Several 3D accelerometer were placed on the arm. We used a single channel of an accelerometer on the lower arm. The original signal was sampled at 96Hz.

We applied a low-pass filter with cut-off frequency of 5 Hz, downsampled the signal by a factor 10 and quantized it by mapping the sensing interval ±1G to integer values in the interval 0-64. Figure 4.3.2a presents an example of this pre-processing, using the gesture "Triangle, pointing up".

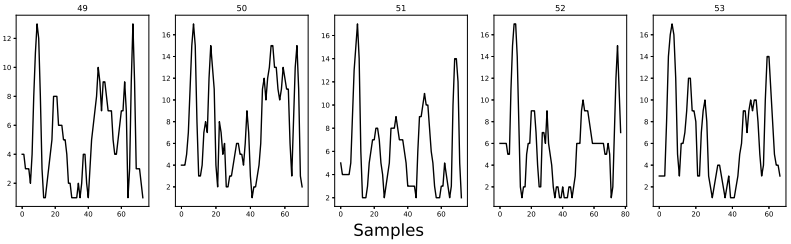
The **Skoda Mini Checkpoint** is a dataset of 10 manipulative actions performed in car maintenance scenario. These movements are a subset of the actions performed in the Skoda dataset [75] but recorded with accelerometers instead of IMUs, at 98 Hz. We chose to use a single channel of one of the sensor placed on the right lower arm. The signal was filtered with a low-pass filter at 5 Hz, decimated by a factor 10 and quantized to integer values in the interval 0-64. An example of this process is shown in Figure 4.3.2c for the action "Writing notes".

The **HCI table** is a dataset of Graffiti which was a single-stroke hand writing style used in PDAs based on PalmOS. In this dataset, the alphabet is performed by a user on a touch screen table while sitting (see Figure 4.3.3a). X,Y trace coordinates recorded at 200 Hz are filtered with a low-pass filter at 5 Hz. Finally, the 2D trajectories are downsampled at 50 Hz, and every sample is encoded with a symbol using the 2D codebook in Figure 4.2.2b) as shown in Figure 4.3.3b, using the letter F as example.

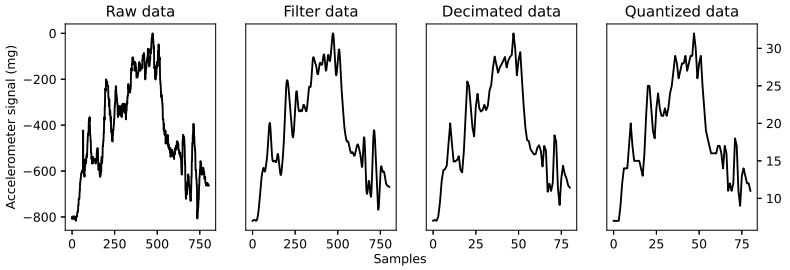
Finally, we report the pre-processing of our **Beach Volleyball** dataset (see Chapter 2) that will be used throughout the rest of this work. Figure 4.3.4 shows the following steps: we select the 3-channels gyroscope sensor signal of the BlueSense placed on the lower arm of the players. We reduce the noise in each of the three components with a low-pass filter with cut-off frequency of 5 Hz, then we fuse the three channels using the L2 norm. Then, the signal is downsampled by a factor 10 and quantized to integer values in the interval 0-64.



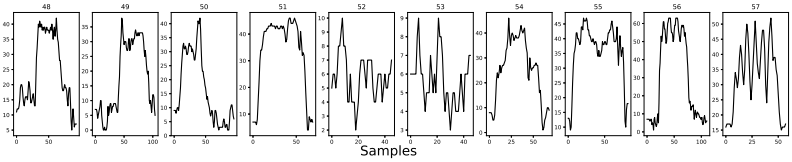
(a) Hci guided - "Triangle, pointing up"



(b) Hci guided - "Segments examples"

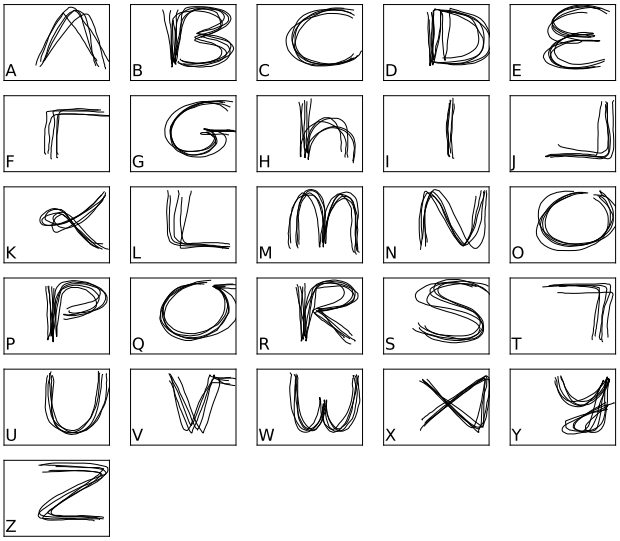


(c) Skoda mini - "Write on notepad"

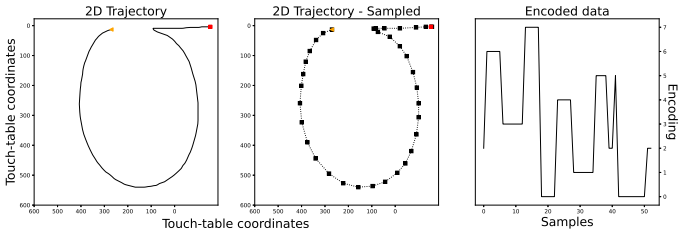


(d) Skoda mini - "Segments examples"

Figure 4.3.2: *Pre-processing steps for segments in hci_guided 4.3.2a and skoda_mini 4.3.2c datasets. Starting from the raw data, a low pass filter with cut-off frequency of 5Hz is applied. Then the signal is downsampled by a factor 10. Finally, it is quantized with integer value in the interval 0-64. Figures 4.3.2b and 4.3.2d show an example of templates for every action class respectively in hci_guided and skoda_mini datasets.*

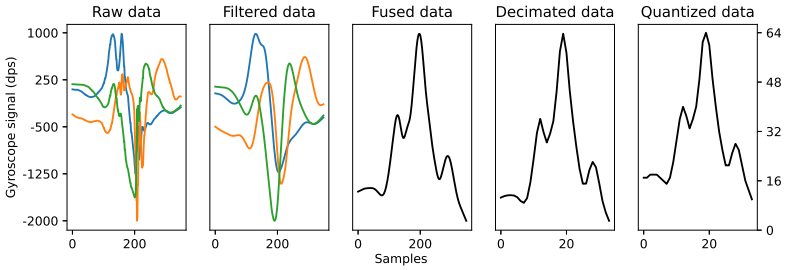


(a) Graffiti alphabet

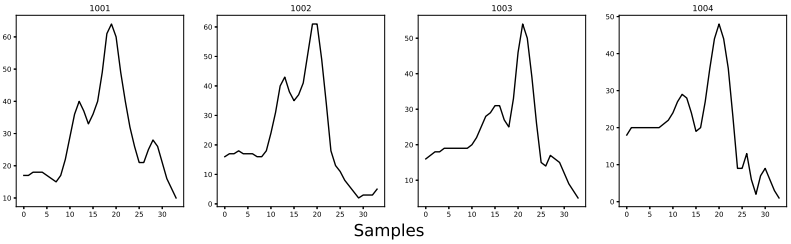


(b) Letter Q encoding example

Figure 4.3.3: *The Graffiti X,Y traces for the A-Z alphabet recorded by the touch-table are displayed in fig. 4.3.3a. Multiple traces are over-imposed for each letter. In fig. 4.3.3b an example of encoding of 2D trajectory for the letter Q is displayed. The orange and red points are respectively the start and the end point of the trajectory. The stroke of the finger recorded by the touch screen (left plot) is sampled temporally. The vectors created between sampling points (center plot) are encoded by finding the closest vector in a 2D codebook (4.2.2a) (right plot)*



(a) Beach volleyball - "Long float serve"



(b) Beach volleyball - "Segments examples"

Figure 4.3.4: *Pre-processing steps for segments in beach_volleyball dataset. Starting from the raw data, a low pass filter with cut-off frequency of 5Hz is applied to each of the 3 channels, x, y, and z. Then, the three channels are fused by using the L1-norm. The obtained signal is then downsampled by a factor 10. Finally, it is quantized with integer value in the interval 0-64. Figures 4.3.4b shows an example of templates for every action class.*

Chapter 5

Training of parameters for TMM based action recognition

*In the previous chapter, we saw that TMMs require a training phase for selecting the best parameters to maximize recognition. In this chapter, we present our method for training WLCSS parameters. Starting from the state of the art and its limitations, we introduce the algorithm and an extensive evaluation to individuate a framework for the training parameters. An initial version of this algorithm has been published in [**VII**]. An evolution of the algorithm and a more extensive analysis is in preparation for [**VIII**].*

5.1 Parameters training methods for TMM: state of the art

Parameters training for TMM has mostly been performed manually for each application (see Table 5.1.1). This is the case for WLCSS in [77, 35] and LCSS employed in [78]. None of the authors did specify whether they follow any systematic method for selecting $\mathbb{C}$.

A custom heuristic was suggested for LCSS in [25, 79]: as predefined 3D unit vectors were used to encode the signal (see Section 4.2), the cost D, I and S were computed as the angular distance between each pair of vectors.

Authors in [80] and [81] used multiple instances of DTW in a multi-joint gestures recognition application. A custom heuristic was designed in order to calculate a weight for each joint based on its contribution to a specific action. The heuristic was specifically created to optimize the weight of this DTW variant in this specific application.

TMM	Ref	Parameters C	Optimization method	Data
WDTW	[80] [81]	Weights of DTW	Custom heuristic	X,Y,Z coordinates of body joints from Kinect
Edit distance	[25] [79] [78]	Insertion, deletion, substitution costs, and V	Custom heuristic	Trajectories from X,Y,Z coordinates
			Manual choice	Fingers position recorded using Leap Motion sensor
WLCSS	[77] [35]	R, P, ϵ, and V	Manual choice	Accelerometer data
				1D Gyroscope data, 1D Accelerometer data, EMG data

Table 5.1.1: *Overview of parameter training for different TMM applied to human sensing data.*

5.1.1 Limitation of state of art

From our Section 5.1, it is clear how no systematic approach was identified for training TMMs parameters. Choosing them manually is a tedious task, considering the wide space of possible values for each parameter. Even a brute force search may not allow to explore the entire search space when a large number of parameters must be optimised. Moreover, as they are mostly defined using dynamic programming (see Table 4.1.1), TMM are not differentiable [82] (without define custom operators) and therefore standard optimization methods such as gradient descent cannot be applied to narrow the parameters search.

While other approaches that do not require a gradient could be used, i.e. back propagation, none of them would allow the optimization of variable length parameters. This would be required later in this work, as explained in Chapter 6.

Few custom heuristics have been proposed to solve this problem but while they reach good results, to the best of our knowledge, none of them could be easily transferred to other TMM or other applications.

5.2 Evolutionary algorithms

In order to overcome these limitations, we based our methods on Evolutionary Algorithms (EAs). EAs are a class of algorithms that "evolve" a randomly initialized population of individuals in order to find the individual that maximizes a fitness function, using principles loosely inspired by biological evolution.

EAs make it possible to search efficiently an optimum even when the space of possible solutions is large and when multiple highly correlated parameters need to be optimized at the same time [83].

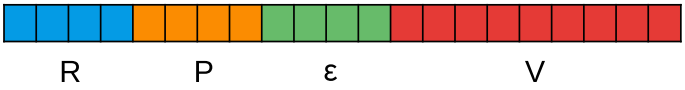


Figure 5.3.1: *Parameters R, P, ϵ and V binary encoded in a bitstring, using $bp = 4$ and $bt = 9$.*

The individuals in the populations are made by one or multiple genes. The genes can be real number or binary digits representing the encoding of a specific object of the optimization problem.

The EA evolves the population through an iterative process according to a fitness score computed for each individual. At each iteration, after the fitness computation, a new population is generated in 3 steps:

- Selection. A subset of the individuals in the previous iteration's population is selected, according to the fitness score. The size of this subset is defined by a *rank* parameter.
- Crossover. The selected individuals are crossed over with a probability *cp* in order to create new individuals.
- Mutation. With a certain probability *mp*, a random selection of genes in the new population is mutated. It means that they are swapped between 1 and 0, and vice-versa, in case of binary genes. Other heuristic can be used in order to mutate numeric genes (see 6.2 for an example).
- Occasionally, at each iteration, some of the best individuals from the previous population are copied to the new one in order to prevent a good individual to be destroyed through crossover/mutation. This is called *elitism*.

The evolution process can end after a predefined number of generations or when there are no significant changes in the fitness score.

5.3 WLCSSLearn_p

WLCSSLearn_p is our method for training WLCSS's parameters. Based on evolutionary algorithms, WLCSSLearn_p trains the set of parameter R, P, ϵ and V for every action class g individually. A template T must be provided for every g.

The training algorithm is presented in Algorithm 1.

WLCSSLearn_p encodes the set of parameters to train in bitstring as shown in Figure 5.3.1. The number of bits for R, P and ϵ is set through the parameter *bp*. The parameter *bt* defines the number of bits for the threshold V.

Symbol	Description
bp	Number of bits used to encode each WLCSS parameter R, P and ϵ.
bt	Number of bits used to encode the threshold V.

Table 5.3.1: *WLCSSLearn_p parameters to control the binary encoding of parameters and thresholds.*

In addition to *bp* and *bt*, WLCSSLearn_p uses a set of parameters to control the evolutionary training presented in Table 5.3.2. The usage of each parameter is explained in the following, for each step of the evolutionary training.

Algorithm 1 WLCSSLearn_p- WLCSS parameters per-action training

Input: dataset, GA_parameters
1: [templates], [segments] = WLCSSLearnInit(dataset)
2: **for each** g in [gestures] **do**
3: population = initPopulation()
4: **for** $a = 1, ...,$generations **do**
5: **for each** p_k **in** population **do**
6: [R, P, ϵ, V]$_k$ = decode(p_k)
7: p_k.fitness = computeFitness([R, P, ϵ, V]$_k$, [templates], [segments])
8: population = selection(population, rank)
9: population = crossover(population, cr)
10: population = mutation(population, mt)
11: $[R, P, \epsilon, V]_g$ = getBestFit(population)
Output: [R, P, ϵ, V]$_g$ **for each** g in [gestures]

WLCSSLearnInit. WLCSSLearn_p performs an initialization step before the training. This step is required to select the template T for each action class g. The MRT is selected as representative T for each g (see Section 6.1). Then, according to the parameters *bp* and *bt*, a population of *pop* individuals in generated randomly. Each individuals p_k encodes a single set of R, P, ϵ and V. WLCSSLearn_p uses the same value of *bp* for R, P, ϵ. The value of *bt* must be bigger than *bp* as the thresholds values are orders of magnitude larger than the value of R, P or ϵ. In our evaluation, the ratio between *bp* and *bt* is set empirically. The length of every individual p_k is therefore:

$$len(p_k) = 3 \cdot bp + bt$$

Symbol	Parameters
iter	Number of iterations to evolve the population for
pop	Number of individuals in the population
rank	Number of individuals selected at every iteration based on the top fitness scores
elit	Optional limited number of individuals carried over as they are (no evolution) at every iteration
cr	Probability of a crossover operation between two individuals
mt	Probability of the mutation of each bits in the bitstring

Table 5.3.2: *EA parameters controlling the number of individuals in the population, the number of selected individual at each iteration, the number of individuals in the elitism, the crossover and mutation probability.*

The algorithm then evolves the population through selection, crossover and mutation operators in order to maximise the fitness function at every iteration, for a single action class g at a time. This means, that for every g, the segments S_j in the training set can be divided in those belonging to g and those that do not. Therefore, during the training we consider the classification a 2 class problem. In this case, the threshold V is used to define a match/mismatch of a certain segment S for the trained action class g.

ComputeFitness A fitness function is needed in order to guide the evolution. WLC-SSLearn_p uses the F1 score as fitness function $F(p_k)$. For every individual p_k, the matching scores $D(T, S_j)$ are computed for every Sj. $D(T, S_j)$ is then compared to V encoded by each individual p_k and classified as g if $D(T, S_j) \geq V$ or $\overline{g}$ if $D(T, S_j) < V$. Then, according the ground truth label of every S_j, we can count:

- True positive (TP): S_j belonging to class g and correctly classified as g;
- True negative (TN): S_j not belonging to class g and correctly classified $\overline{g}$;
- False positive (FP): S_j not belonging to class g, but wrongly classified as g;
- False negative (FN): S_j belonging to class g, but erroneously classified as $\overline{g}$.

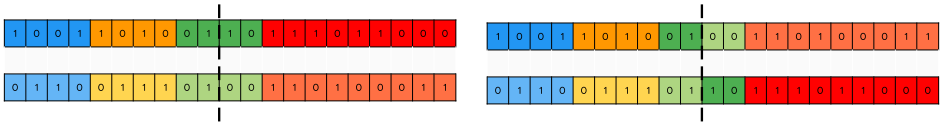
Using these definition, we can compute precision, recall and F1 score as our fitness function $F(p_k)$:

$$precision = \frac{TP}{TP + FP} \quad recall = \frac{TP}{TP + FN}$$

$$F(p_k) = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.1}$$

Selection The top *rank* best fitting individuals are selected in order to parent the future generation of individuals. The *rank* individuals are then replicated in order to maintain the fixed size *pop* of the population [1].

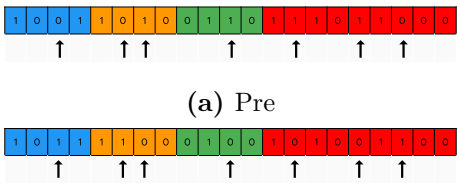
Crossover With a probability *cr*, every pair of selected parents bitstring generates a new offspring by swapping two parts of consecutive bits as displayed in Figure 5.3.2. The swapping point is chosen randomly from a discrete uniform distribution in the interval $[2, len(p_k) - 2]$, to ensure to exchange at least 2 bits. The crossover operator does not take into account the boundaries of R, P, ϵ and V within the bitstring (see Section 5.6 for more).



(a) Pre **(b)** Post

Figure 5.3.2: *Example of crossover, pre and post operation. The crossover operator takes two bistrings (left) and with probability cr generates two new bitstrings by swapping a random portion of the bitstrings.*

Mutation The bits of every bitstring in the population mutates with a per-bit probability *mt*. The mutation is a change of bits from 0 to 1 and viceversa. See Figure 5.3.3 for an example.



(a) Pre

(b) Post

Figure 5.3.3: *Example of mutation, pre and post operation. A random selection of bit is changed from 0 to 1, and viceversa, with a per-bit probability of change mt.*

Elitism A limited number of individuals indicated as *elit* from the previous iteration's population is carried over at every iteration. This prevents the evolution to diverge from an optimal solution once one is found.

[1] The actual number depends on the *elitism*. If $elit > 0$, then the number of individuals after the replication is $pop - elit$

Parameter	Start value	End value	Step
cr	0.1	0.5	0.05
mt	0.05	0.5	0.05

Table 5.4.1: *Interval of values of cr and mt considered in our evaluation.*

WLCSSLearn_p stops after a predefined number of iteration (This is further discussed in Section 5.6).

5.4 Evolutionary parameters analysis

We present an extensive evaluation of WLCSSLearn_p aimed at showing how evolutionary parameters and encodings affect training time and recognition performance. We characterize:

- the effect of *cr* and *mt* on the recognition performance, while keeping constant *pop*, *rank* and *elit*, using WLCSSLearn_p;
- we analyse how *pop*, *rank*, and *elit* affect the F1 score and the speed of training of WLCSSLearn_p, for fixed values of *cr* and *mt*;
- we evaluate the recognition performance and the speed of training for different values of *bp* and *bt*. Using a lower number of bits reduces the search space and could lead to faster convergence. On the other hand, a larger number of bits may slow the convergence but allowing to explore more the search space, finding better solutions.

Since the evolution of the populations in WLCSSLearn_p is a stochastic process, we repeat each test 10 times and we average the results. In order to speed-up our evaluation, we employ the parallelized implementation of WLCSS based on CUDA described in Chapter 7.

5.4.1 Crossover vs Mutation

First, we examine the impact of *cr* and *mt* on the training. We empirically choose two intervals of values for *cr* and *mt* as shown in Table 5.4.1. We set the other evolutionary parameters to fixed values displayed in Table 5.4.2.

We ideally want to find one set of *cr* and *mt* that would work for every dataset. Table 5.4.3 shows the best fitness score for each dataset and the values of *cr* and *mt*

Parameter	Value
pop	32
rank	12
elit	3
iter	500
bp	4
bt	9

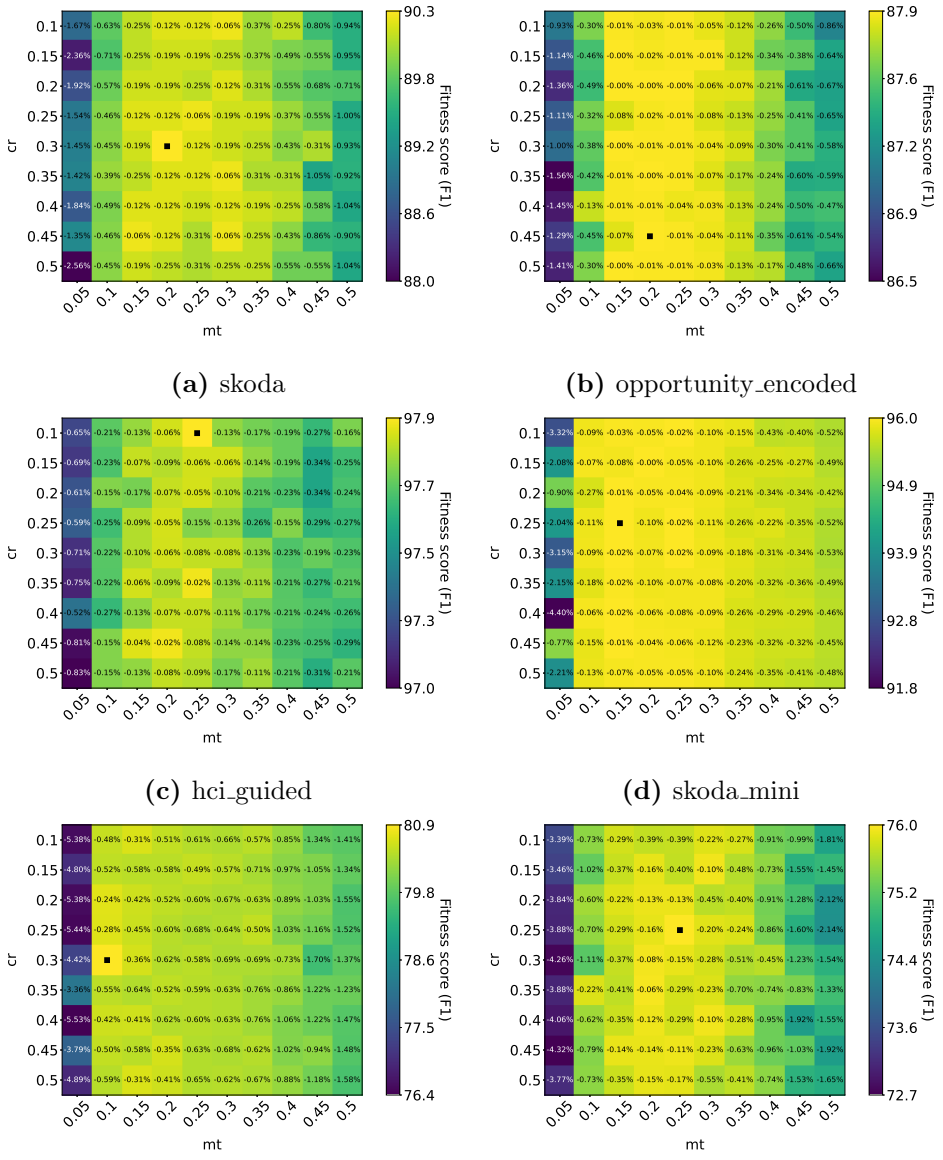
Table 5.4.2: *Fixed values of other evolutionary parameter and encodings during cr and mt evaluation*

used to obtain it. It is clear how there is no single value that works for every dataset. Finding a range of value suitable for a wider range of scenario is more valuable. Therefore, Figure 5.4.1 presents the fitness scores reached at the end of the training, for each dataset, and the performance loss compared to the best fitness scores presented in Table 5.4.3. Since the variation in performance for each value of cr and mt are between 0% and 5%, we find more useful and clear for the reader to analyse the performance loss compared to the best fitness scores. Figure 5.4.2 shows the average performance loss across all the dataset.

From the figures, we can observe that cr and mt greatly affect the recognition performance, although with different impact. mt is apparently more important than cr, as it is possible to notice that the bigger variations in the F1 score happen on the x-axis of the plots, rather than the y-axis. Secondly, given the large variability of the color distribution amongst datasets, it is possible to notice that the effect of cr and mt can change greatly across different datasets.

However, from Figure 5.4.2 it is possible to identify that the interval $[0.15 - 0.25]$ for mt minimize the performance loss, with values between -0.23% and -0.11%, depending on the value of cr. On the other hand, it is not possible to clearly identify an interval for cr that would strongly reduce the performance loss. However, from Table 5.4.3 it visible how, with the exception of the hci_guided dataset, the best performance are obtained when $cr \geq mt$, therefore we suggest that $cr > 0.25$ for any value of mt.

Finally, with a maximum performance loss of $< 1\%$, our findings show that cr and mt do not necessarily need to be optimized per application if they are in the suggested interval of values.



(a) skoda **(b)** opportunity_encoded

(c) hci_guided **(d)** skoda_mini

(e) hci_table **(f)** beach_volleyball

Figure 5.4.1: *Evaluation of maximum fitness score reached after 500 generations of WLCSSLearn_p with respect to different values of cr and mt averaged across action classes for all the datasets. The percentage loss of the score for each combination of cr and mt is showed with respect to the best score represented by the black square. The colormap represent the fitness score obtained at the end of the training, and it is computed with respect to minimum and maximum fitness score for each dataset singularly. The best and worst scores are displayed as the maximum and minimum values of the color-bar.*

Dataset	cr	mt	Max Fitness Score (F1)
skoda	0.30	0.20	90.35
opportunity_encoded	0.45	0.20	87.90
hci_guided	0.10	0.25	97.86
skoda_mini	0.25	0.15	95.98
hci_table	0.30	0.10	80.87
beachvolleyball	0.25	0.25	76.02

Table 5.4.3: *Best fitness scores for each dataset and relative values of cr and mt.*

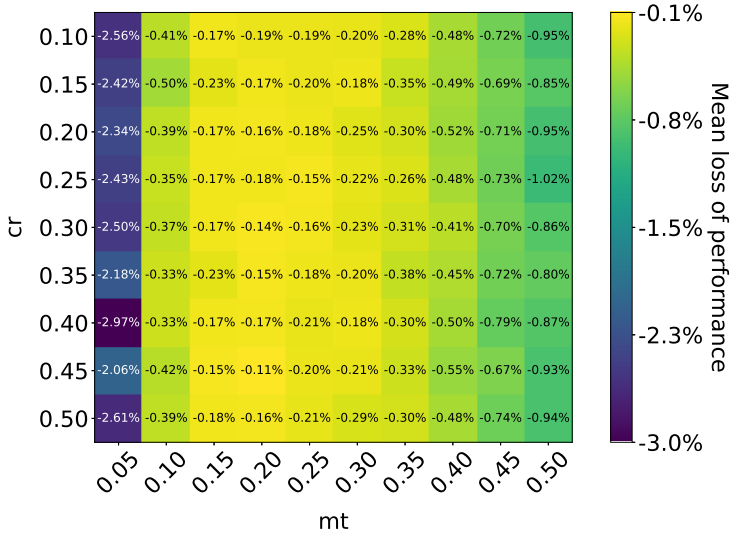


Figure 5.4.2: *Average loss of performance across all the datasets for each pair of values of cr and mt.*

5.4.2 Population size, rank and elitism

The parameters $pop, rank$ and $elit$ also affect the fitness score. We investigate this effect by executing WLCSSLearn_p with five different combination of pop, $rank$ and $elit$ presented in Table 5.4.4.

We also set the value of the other evolutionary parameters to constant values in order to simplify the evaluation. We manually set $cr = 0.35$, $mt = 0.25$, as from the previous analysis we show minimal change in performance within the specified intervals, and $iter = 1000$.

The results of this analysis are presented in Figure 5.4.3. An increase in population size appears to lead to convergence in a lesser number of iterations. However, the time for training grows linearly with the size of the population, meaning that using a smaller population would take less time to complete 1000 generations. While there is no necessarily a best choice between the two, we suggest an intermediate number of individuals (e.g. $pop = 32$) as a reasonable trade-off between training time and avoidance of local optimum, which can happen for low number of pop.

pop	rank	elit
8	3	1
16	6	2
32	12	3
64	24	6
128	48	12

Table 5.4.4: *Analysed combinations of pop, rank and elit.*

5.5 Genetic encoding evaluation

We evaluate the effect of the number of bits used to encode R, P, ϵ, i.e. the parameters bp and bt, after 1000 iterations. Smaller values of bp lead to smaller R, P and ϵ which would reduce the search space, speeding up the training. On the other hand, bt must variate accordingly to bp, as R and P affect the possible values of the matching scores that must be compared against V in order to define a match/mismatch.

We studied several values of bp and bt in pairs, presented in Table 5.5.1. In order to make the results only dependant from the number of bits, we keep the same values for all

bp	bt
3	7
4	9
6	13
8	17
10	21
12	25
14	29

Table 5.5.1: *Analysed pairs of values for bp and bt.*

the remaining parameters to $pop = 32, rank = 12, elitism = 3, iter = 500, cr = 0.35, mt = 0.25$.

Figure 5.5.1 presents the results of this analysis. Each plot represents the average of 10 evolutionary run for each pair of values of *bp* and *bt*. The results are also analysed at different steps of the training. It is possible to notice that for skoda_mini, hci_guided and beach_volleyball datasets even very small number of bits (3, 7) are sufficient to reach high fitness scores already after few iterations, i.e. 50. However, under the same conditions, the skoda, opportunity and hci_table datasets fails to reach their maximum performance as shown for other and higher number of bits.

By looking at the average fitness score reached after 1000 generations for all the dataset, the pairs of values of *bp* and *bt* (4, 9) and (6, 13) more consistently allow to get the maximum fitness score.

Higher values of *bp* and *bt* not only increase the time WLCSSLearn_p takes to converge, but also they may even fail to converge. For example, the pair (8, 17) takes longer to train than lower number of bits but still reach comparable fitness score for the skoda_mini, hci_guided and skoda datasets. On the other hand, for opportunity_encoded and hci_table, the same pair of bits values makes WLCSSLearn_p fails to reach its maximum potential fitness score, reached with the encoding (6, 13).

Figures 5.5.2 show the probability distribution of the numerical values obtained after training, for each pair of R, P and ϵ, for all the datasets, for each pair of *bp* and *bt*. While we would be interested in a clear concentration of probability on specific values, the scattered distribution of possible values emerging from the plots strongly suggests that all the three parameters must be part of the training.

5.6 Discussion

We demonstrated how WLCSSLearn_p is effective for the training of WLCSS's parameters. However, the training procedure as well as the genetic operators are not tied only to the WLCSS method but they can be used with other TMMs. With a different encoding of the individuals of the population, WLCSSLearn_p can be deployed for other parametrized TMMs such as LCSS and WDTW. In case of LCSS for example, the individuals would encode the substitution, deletion and insertion costs and the fitness function would use LCSS to compute $F(p_k)$.

Moreover, while we focused on the task of action recognition, our evaluation using different datasets including different encodings showed that our method is applicable to a large variety of input data. This makes WLCSSLearn_p exportable to other pattern recognition tasks even from different domains such as audio streams.

WLCSSLearn_p uses a genetic algorithm to explore the search space. As explained in Section 5.1.1, other methods, such as gradient descent cannot be applied to the optimization of non-differentiable functions as the equation of WLCSS. A brute force approach could be used instead: however considering the case of 4 bits for the WLCSS parameters and 9 bits for the threshold V, for a single action class, this would results in $2^{4 \cdot 3+9} = 2^{21} = 2097152$ possible sets to be evaluated in a brute force search. Considering the same number of bits, a populations of 32 individuals with $elit = 3$ and the maximum of 500 iterations, WLCSSLearn_p requires only $32+29 \cdot 499 = 14532$ computations to reach an optimal solution. In this scenario, WLCSSLearn_p requires 99.3% less computations. Additionally, from the analysis in Section 5.5, it is possible to notice that WLCSSLearn_p reaches the maximal recognition performance with even less than 500 iterations for most datasets, reducing the required computations even more.

There are several avenues to bring further to WLCSSLearn_p:

- enhancing the crossover operator. In the current implementation, the crossover operation is applied on the entire chromosome at once. It is possible to make this operator aware of the boundaries of the genes for R, P, ϵ and V, applying the crossover operation only between whole genes, instead of breaking them at bit level. This would enable a more precise evolution and possibly a faster convergence. Moreover, it could make the parameter cr more effective than the current implementation, where we demonstrated having lower impact, especially compared to mt.

- introducing a stop criteria for the evolutionary training. From our evaluation, it is

Dataset	**F1 score**	**Encoding** (bp, bt)
skoda	0.91 ± 0.01	6, 13
opportunity_encoded	0.88 ± 0.00	4, 9
hci_guided	0.98 ± 0.00	6, 13
skoda_mini	0.95 ± 0.00	3, 7
hci_table	0.81 ± 0.01	6, 13
beachvolleyball	0.76 ± 0.00	4, 9
Average	0.88 ± 0.08	

Table 5.7.1: *Summary of recognition performance (F1 Score) with WLCSSLearn_p. The best encoding for each dataset is indicated. The evolutionary parameters were all set to* $pop = 32, rank = 12, elitism = 3, iter = 1000, cr = 0.35, mt = 0.25$.

clear how the biggest and more significant changes of the training happens in the first tens of iterations. A stop criteria would accelerate the training by avoiding useless iterations once an optimal results has been achieved. A possible solution for this would be to compute the gradient of the fitness scores and stop the evolution when such gradient remains 0 for a predefined number of iterations.

5.7 Conclusion

We presented WLCSSLearn_p, a training algorithm for WLCSS with a focus on action recognition. Based on evolutionary algorithms and using custom encoding of chromosomes, WLCSSLearn_p trains the parameters of WLCSS in order to maximize the recognition performance.

We evaluated WLCSSLearn_p on 6 datasets of gestures including different kind of data encoding/quantization. We proved that our method is robust against the intrinsic variability of sensors data for action recognition, it is independent from the data encoding and it can work with minimal change from the user.

Through an extensive analysis we individuate a framework for the WLCSSLearn_p's training parameters:

- We found that the best intervals for the mutation probability is $[0.15 - 0.35]$. Within this interval, the value of the crossover probability does not affect performance, but considering our empirical results, we advices a value of $cr \geq mt$, and therefore

$cr > 0.25$. Within these range of values, the performance loss is $< 0.21\%$ meaning that cr and mt do not require any application-specific optimization.

- Our evaluation showed that a good balance between performance and training speed is achieved by a number of individuals $pop = 32$, with a proportional $rank = 12$ and $elitm = 3$, when the maximum number of iterations is 1000.

- Finally, across all datasets and given the previously suggested evolutionary parameters, we advise an encoding of 6 bits for the WLCSS parameters and 13 bits for the thresholds V.

Within this framework of parameters, WLCSSLearn_p showed recognition performance of $88\% \pm 8$ F1 score on average across all the datasets.

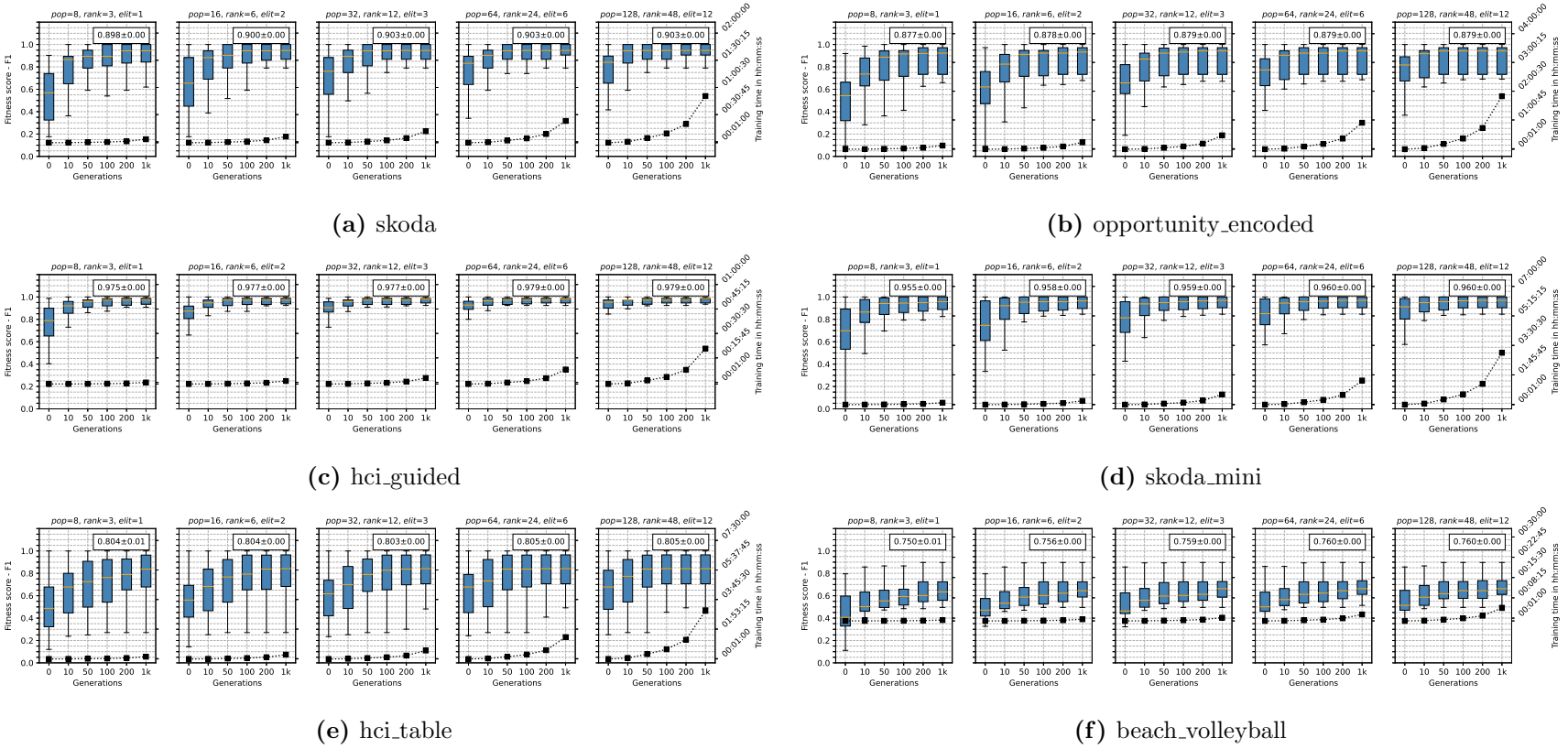
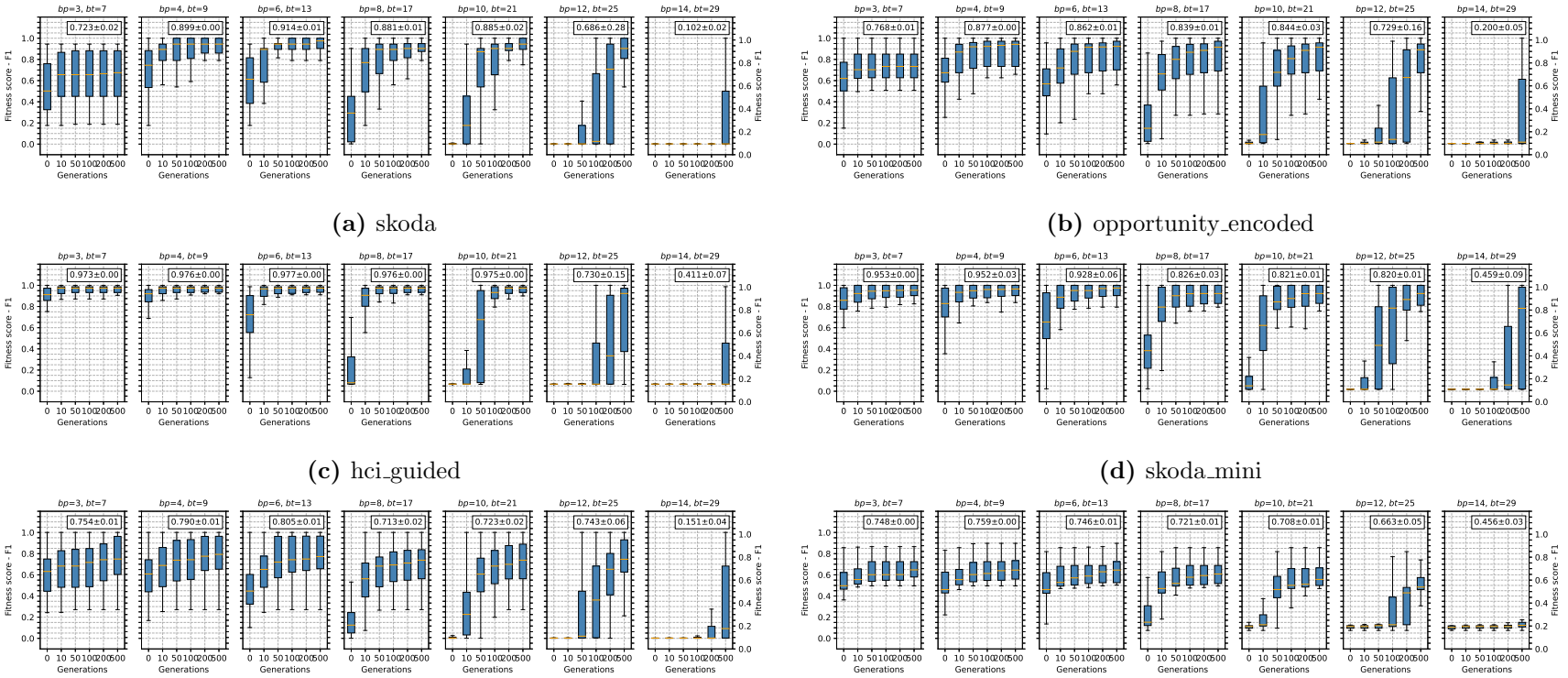


Figure 5.4.3: *Fitness score at the 0, 10th, 50th, 100th, 200th, and 1000th generation of the evolutionary training withWLCSSLearn_p, for different values of pop, rank and elitism (indicated as title of each subplot), for all the datasets. Each boxplot is the average of the training for every action repeated 10 times, for different randomly generated initial population. The training time of each configuration are reported using the right y-axis. These times cannot be compared across different datasets because the analysis were executed on different machines. However, since all the different combinations for the same dataset were executed on the same machine they can be used to evaluate the difference in time between different values of pop, rank and elit. These times are calculated as average of all the multiple executions. Finally, they are comparable within each dataset but not across different datasets as the tests were executed on different machines assigned on a per-dataset basis.*



(a) skoda

(b) opportunity_encoded

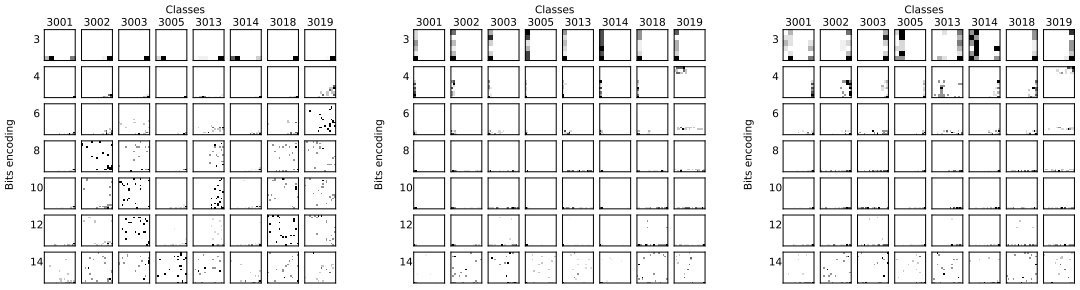
(c) hci_guided

(d) skoda_mini

(e) hci_table

(f) beach_volleyball

Figure 5.5.1: *Fitness score at the 0, 10th, 50th, 100th, 200th, 500th, and 1000th generations of the evolutionary training withWLCSSLearn_p, for different pairs of bp and bt (indicated as title of each subplot). Each boxplot is the average of the training for every action in each dataset repeated 10 times, for different randomly generated initial populations.*



(a) R, P **(b)** P, ϵ **(c)** R, ϵ

Figure 5.5.2: *Probability distribution of values R, P, and ϵ in pairs for Skoda dataset. It is possible to notice how the parameters do not follow any specific distribution, making necessary to train all of them. For conciseness sake, we report here only the results for the Skoda dataset. Additional plots for all the other datasets can be found in Figure B.1, in appendix.*

Chapter 6

Generation of variable templates for TMM based action recognition

*As shown in Chapter 4, a TMM requires also a set of accurately selected templates to match. Following the WLCSS parameter training described in Chapter 5, we present our method for power-aware template generation for WLCSS. Starting from the state of the art and its limitations, we introduce the algorithm and an evaluation of the template generation. The content of this chapter is being prepared to be submitted to [**VIII**].*

6.1 Template training methods for TMM: state of the art

Template training is needed in order to select time series segments that will be used as templates for the classification. From our literature review, we have identified two main categories of templates training: i) the selection of one or multiple templates among the ones available as training set, or ii) the creation of synthetic templates starting from those available a training time.

Falling in the first category, the simplest approach is to use the entire training set of templates for the classification. This has been proposed for classification in a 1-NN manner with WDTW [84], LCSS [78], and Manhattan distance [85].

A manual selection of a subset of templates from the training set is commonly used with Euclidean distance [86],DTW [87, 88, 89], and WLCSS [35]. None of the authors specified the criteria for the selection.

A more systematic solution for selecting templates from a training set named "most representative template" (MRT) has been proposed for LCSS [25, 79] and DTW [90]. For every recognition class, MRT computes a distance metric between each pair T_i and

T_k, with $i \neq k$, in the training set and chooses the template with the minimum sum of distances:

$$T = \underset{i}{\operatorname{argmin}} \sum_{i \neq k} D(T_i, T_k)$$

Effectively, MRT selects the template with lowest intra-class variation. This however does not take into account for inter-class variability, potentially resulting in lower classification performance. In addition to MRT, authors of [90] also explored random selection, best alignment and manual creation but without solving any of the aforementioned issues. As solution to this issue, cross-validation was used to select templates for DTW in [91].

The other family of templates training approaches aims at creating synthetic templates. The authors in [8] clustered the templates of each class and used the centroid obtained as average as templates. Improved methods for averaging the training templates have been proposed for DTW in [92, 93, 17] and MPLCS [94]. A more advanced combination of cross-validation and averaging is suggested for DTW [95]: authors used cross-validation to select 9 templates per class from the training set, while averaging is used on these 9 templates to generate a 10th one. These 10 templates per class are then employed in 1-NN classification. Synthetic templates are generated for DTW [96], by analysing the hand movements for the actions in the training set and manually defining a hand trajectories from scratch for each action.

Another approach for generating templates is to use specifically recorded templates for the classification of other segments. Compared to the rest of the segments, these specific templates are recorded under more favourable condition such as using expert users performing the movements or a more controlled environment to reduce noise in the sensors signal. These methods has been used for WDTW [80, 81] and WLCSS [77].

Finally, evolutionary strategies (ES) have been proposed for template training using DTW in [97] using a simple synthetic dataset and successfully applied to movement recognition in [98]. The entire training set is used as initial population of the ES.

6.1.1 Limitations of state of the art

For templates training, both categories of methods we identified have shortcomings. For the first category, using the entire training set for 1-NN classification accounts for larger variety of templates at the cost of increasing the complexity. Selecting a subset of training templates for the classification solves this issue. However, manual selection of the templates from the training set is intractable for large datasets. For this reason, more systematic methods such as MRT and cross-validation have been suggested. Nevertheless,

TMM	Ref	Training method	Data
DTW	[87]	Manual choice of sub-set from training set	Static and dynamic hand movements recorded using cameras and Kinect
	[91]	Cross-validation	Hand trajectories from Kinect tracking
	[95]	Cross-validation and custom heuristic	Actions from video recording
	[90]	Random choice, MRT, best alignment and manual creation	Walking patterns recorded using a gyroscope
	[96]	Hand movements using Time-of-flight (TOF) videos	Synthetically created through heuristic
	[88]	Manual choice	3D Accelerometer data
	[92]	Template averaging using custom heuristic	Fingerprint and signature images
	[93]		Fingerprint and signature images
WDTW	[97]	Generated with ES	Artificial dataset of time series
	[98]		
	[80]	Specifically recorded	X,Y,Z coordinates of body joints from Kinect
	[81]	Specifically recorded	
DTW, WDTW, DDTW (Derivative DTW)	[84]	Training instances used as templates	Various dataset (UCR Time Series Classification Archive [99])
DTW, Euclidean distance	[17]	Custom average	3D Accelerometer data
DTW, LCSS	[89]	Manual choice from dataset	Hand shapes from images
Euclidean distance	[8]	Computed as centroid of clustering	Fingers shape from hand pictures
	[86]	Manual selection	Features extracted from images of hand movements
Manhattan Distance	[85]	Training instances used as templates	Features extracted from images of hand movements
MPLCS (custom LCSS)	[94]	Custom average	Trajectory from 3D camera
Edit distance	[25]	MRT	Trajectories from X,Y,Z coordinates
	[79]		
	[78]	Training instances used as templates	Fingers position recorded using Leap Motion sensor
WLCSS	[77]	Specifically recorded	Accelerometer data
	[35]	Manual selection	1D Gyroscope data, 1D Accelerometer data, EMG data

Table 6.1.1: *Overview of training procedure to select $\{T\}$, for different TMMs applied to human sensing data.*

none of these methods can explore possible variation of the templates shapes to improve the recognition, or length reduction for computational saving.

Falling in the second category, methods for averaging the training templates have been proposed in order to create new synthetic templates. While some of these methods can

be beneficial to reduce the computation by using a single templates generated from the training segments, none of them explore systematically variations of shape and length of templates for further and potentially configurable reductions.

Evolutionary strategies have been proposed as solution in order to automatically generate shape variation of the training set. These works however do not attempt to optimise any variation of template length. Moreover, using the training set as the only initial population might lead to premature convergence missing better solutions that an unconstrained random sampling of the search space could offer.

Finally, specifically recording of special templates was proposed in order to have less noisy time series to match for the classification. However, setting up a data collection just for templates training can be expensive. Moreover it can be physically impossible if other researchers do not have access to the same setup used for the original data collection.

In summary, no standard method was identified for the training neither for the parameters nor the templates and none of the proposed approach can be easily be transferred across multiple TMM. More importantly, to the best of our knowledge, none of the custom method found in the related work aims at reducing the templates length for power saving.

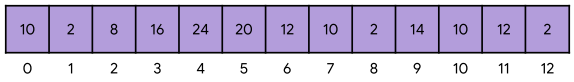
6.2 WLCSSLearn_t

WLCSSLearn_t generates a single template for each action class g. The generated templates maximise the inter-class differences, while maximising the intra-class similarities of the segments in the training set. By generating a single template per class, WLCSSLearn_t decrease the effort of the classification compared to using the entire training set or a large sub-set of it for the matching when using 1-NN classification (see Chapter 4).

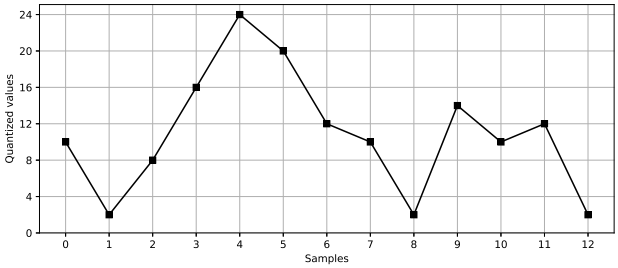
Moreover, the most unique feature of this algorithm is that it optimises the shape of the template as well as its length. We designed a fitness function which allows to emphasize either improving recognition performance or reducing template length, and therefore computational cost. This is achieved by using a variable-length genetic encoding.

WLCSSLearn_t encodes the templates to evolve as sequence of samples as in Figure 6.2.1. Each sample of the templates in WLCSSLearn_t is a natural number. The possible values for the samples are defined by the quantization interval or encoding used for the time series in the dataset (see Chapter 4). At first, all the templates have the same length: this length is computed as average length of training segments for the action class g.

A set of parameter $C_g = \{R, P, \epsilon\}$ per action class g must be provided at initialization



(a) WLCSSLearn_t individual p_k



(b) Representation of the same p_k as segment

Figure 6.2.1: *Example of WLCSSLearn_t individual. An individual p_k is generated as array of numeric sample 6.2.1a. 6.2.1b show its representation as time series. In this example, the samples of the segment are quantized on the interval 0-24.*

step. This set can be provided by an heuristic or, by a prior execution of WLCSSLearn_p. The thresholds V_i are computed automatically at the end of the evolution given that their values are highly dependant from the templates length.

WLCSSLearn_t generates the templates following the process presented in Algorithm 2. In addition to the evolutionary parameters described in Table 5.3.2, WLCSSLearn_t requires specific parameters for the template generation that are shown in Table 6.2.1.

WLCSSLearn_t initialization During the initialization, WLCSSLearn_t computes the starting length of the individuals in the population, for each action class g. Consequently, the algorithm calculates the maximum and minimum possible length of the generated templates according to the max_l and min_l parameters. Once this step is complete, the training begins for one action class g at a time generating a random population of individuals p_k.

ComputeFitness The fitness function comprises two terms: i) the length of the individual p_k indicated with $l(p_k)$ and ii) a measure of the recognition performance for each individual p_k. The latter is a measure of how well each individual p_k distinguish between segments S_j that belong to action class g from those that do not. This measure is indicated with $\Upsilon(p_k)$ and it is computed as:

$$\Upsilon(p_k) = D_g - D_{\overline{g}}$$

where D_g is the 5-th percentile of the matching scores $D(p_k, S_j)$ of all S_j for which $y(S_j) = g$ and $D_{\overline{g}}$ is the 95-th percentile of the matching scores $D(p_k, S_j)$ of all S_j for which

Algorithm 2 WLCSSLearn_t- per-action template generation

Input: dataset, [actions classes], $[R, P, \epsilon]_g$

WLCSSLearnInit(dataset)
for each g in [actions classes] **do**
    population = initPopulation()
    for $a = 1, ...,$generations **do**
        for each p_k **in** population **do**
            [templates] = decode(p_k)
            $F(p_k)$ = computeFitness([R, P, ϵ]$_g$, [templates], [segments])
        population = selection(population, rank)
        population = crossover(population, cr)
        population = mutation(population, mt)
        population = variate_length(population, λ)
    T_g = getBestFit(population)
    V_g = computeThreshold(T_g, [segments])

Output: [T, V]$_g$ **for each** g in [actions]

Symbol	Description
en	Enlarge probability. It defines how likely is for an individual to be enlarged at each iteration.
sh	Shrink probability. It defines how likely is for an individual to be shrunk at each iteration.
min_l	Minimum length. It is the minimum value that an individual can be shrunk to. It is expressed as percentage of the original length.
max_l	Maximum length. It is the maximum value that an individual can be expanded to. It is expressed as percentage of the original length.
λ	Parameters controlling the computational cost reduction vs. recognition performance trade-off.

Table 6.2.1: *WLCSSLearn_t parameters controlling the templates lengths and the computation reduction vs. recognition performance trade-off.*

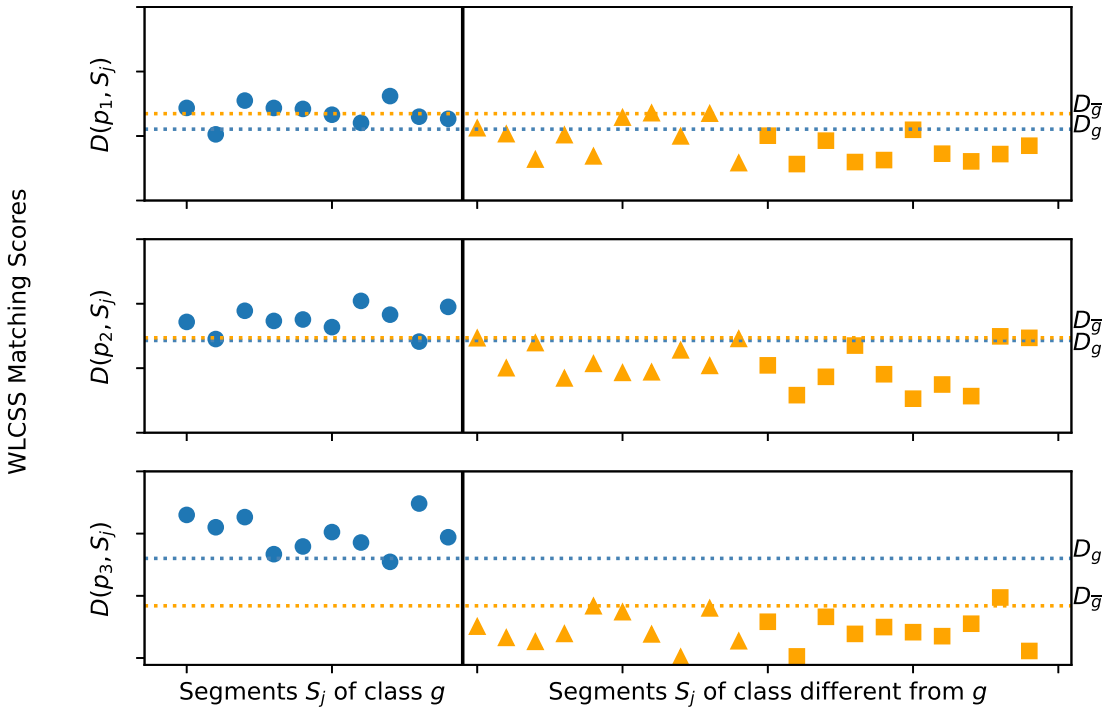


Figure 6.2.2: *In this example, a dataset comprising 30 segments of 3 classes is used. The first 10 segments are of class 1 (●), the segments 10-20 of class 2 (▲) and segments 20-30 of class 3 (■). WLCSSLearn is applied to find representative templates of class* $g = 1$. *Here the matching score between three individuals* p_1, p_2, p_3 *and all the segments* S_j *are represented by dots for a single iteration of the evolution. As the evolution has been ongoing, we notice that the matching scores between* p_1, p_2, p_3 *and the segments 1-10 tend to be higher than the matching scores between* p_1, p_2, p_3 *and segments 10-30. Visually,* p_3 *appears the best among the three individuals, as the matching scores achieved are substantially higher when comparing* p_3 *to segments 1-10, which are the ones the individuals are evolved to look like, and substantially lower when comparing p3 to segments 20-30. This is also represented by the dashed lines* D_g *and* $D_{\overline{g}}$.

$y(S_j) \neq g$. The usage of percentiles allows the performance measure Υp_k to account for possible outliers in the classification. More on this is discussed in Section 6.4. An example of the computation of Υp_k for 3 individuals p_1, p_2, p_3 for the same action class g is displayed in Figure 6.2.2.

Both the individual length $l(p_k)$ and $\Upsilon(p_k)$ are normalized on the interval 0-1 in order to combined in a single fitness value per p_k. The normalized length $\|l(p_k)\|$ is computed with min-max normalization using the minimum min_l and maximum length max_l of the templates for each action class g set at initialization step.

A sigmoid function is used to normalize $\Upsilon(p_k)$ in the same interval 0-1:

$$\|\Upsilon(p_k)\| = \frac{1}{1 + e^{-b \cdot \Upsilon(p_k)}}$$

We empirically set the value $b = \frac{5}{R \cdot \max l(S_j)}$, where $R \cdot \max l(S_j)$ is the upper-bound of the possible matching score considering the WLCSS functioning. More on this is discussed in Section 6.4.

Finally, the two terms are combined in the fitness function:

$$F(p_k) = \lambda \cdot \|l(p_k)\| + (1 - \lambda) \cdot \|\Upsilon(p_k)\| \tag{6.1}$$

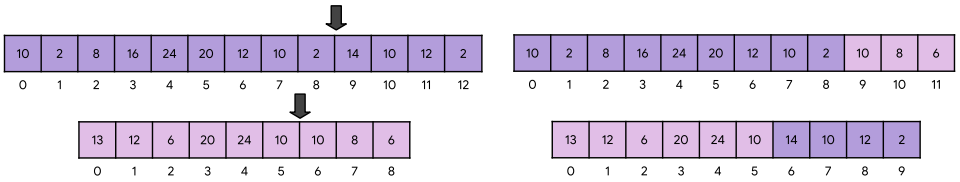
in which the parameter λ allows to configure the trade-off between reducing the length of the template for computation reduction or improving the recognition performance.

Selection The top *rank* best fitting individuals are selected based on their values of $F(p_k)$. The *rank* individuals are then replicated in order to maintain the fixed size *pop* of the population taking into account elitism as for WLCSSLearn_p.

Crossover The crossover operator is applied with probability *cr* to pair of individuals, coupled randomly after selection. Due to the variable-length genetic encoding, we designed a specific crossover operator as shown in Figure 6.2.3. A random number is selected from a standard uniform distribution $\mathcal{U}(0, 100)$. This value indicates the percentage of the length of each individual that must be exchanged. This values is then multiplied by the length of each individuals and round to integer in order to define the crossover sample position. This ensures that no individuals greater than the possible maximum length are generated at each step.

Mutation A random selection of genes are mutated at this step, in each individuals. We introduce a specific operator due to the encoding of the templates in WLCSSLearn_t as presented in Figure 6.2.4. When a sample s of the template is mutated into $\hat{s}$, the following operation is applied:

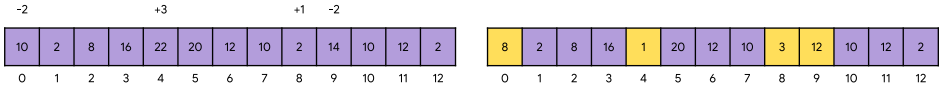
$$\hat{s} = (s + x) \mod \max(enc) \tag{6.2}$$



(a) Pre **(b)** Post

Figure 6.2.3: *Crossover operator for template training. The position is chosen randomly as percentage (0-100%) that the two individuals will swap. In this case, the crossover position indicated by the arrow is chosen as 75%. Rounding this percentage to the effective lengths of the two individuals, the swapping position is set to 8 and 5 for the two individuals respectively. Figure 6.2.3b shows the two new individuals resulting from the crossover operator.*

with x being a random value following the distribution $X \sim \mathcal{N}(0, 4)$, and $\max(enc)$ the maximum value of the quantization (or encoding) of the time series in the dataset. This constrains the shape of the generated template to variate at specific samples, while maintaining a similar shape overall from one generation to the next.

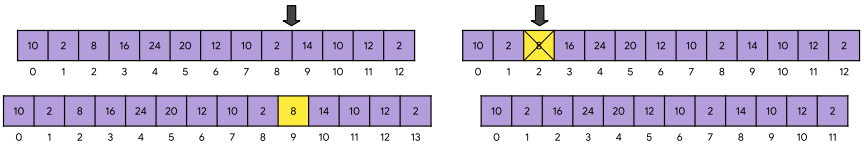


(a) Pre **(b)** Post

Figure 6.2.4: *Mutation operator for template training. The samples at position 0, 4, 8, 9 are mutated by adding random values following the distribution $X \sim \mathcal{N}(0, 4)$ using Equation 6.2. As example, considering the same quantization from Figure 6.2.1, the sample at position 4 is computed as* $24 + 3 \mod 24 = 1$.

VariateLength This operator can enlarge, shrink or leave the individuals as they are, respectively with probability en, sh, $1 - (en + sh)$, with $0 \leq en \leq sh \leq 1$. Figure 6.2.5 shows an example. When the operation of enlarge is selected, a random position within the individual to add 1 sample is picked. The value of this gene is the mean value of the genes immediately before and after the insertion point. On the other hand, the shrink operation removes a genes from the randomly selected position, for each individual. The operations of enlarging/shrinking are not permitted on templates that are already respectively of the maximum and minimum allowed length, no matter the values of en and sh. Figure 6.2.5 presents an example of this operator.

WLCSSLearn_t stops after a finite number of iterations. At the end of each per-action



(a) Enlarge **(b)** Shrink

Figure 6.2.5: *VariateLength operator. Enlarging 6.2.5a: a position j is selected randomly from a discrete uniform distribution $\mathcal{U}\{0, l(p_k)\}$, with $l(p_k)$ being the individual length. The value of the new sample introduced at that position, indicated by the gray arrow, is calculated as the mean value between the previous sample at j and the next sample at $j+1$. In this example, the new sample, shown in the yellow box, is introduced at position 8 and calculated as $(2 + 14)/2 = 8$. Shrinking 6.2.5b: a sample at a randomly chosen position j is removed. The removing position is randomly from a discrete uniform distribution $\mathcal{U}\{0, l(p_k)\}$.*

evolution, the threshold V_i is computed as the average point between D_g and $D_{\overline{g}}$ obtained from the individual with the best fitness $F(p_k)$ at the last iteration.

6.3 Template Generation Evaluation

In this section, we present the evaluation of WLCSSLearn_t and its configurable trade-off between recognition performance and computation saving. We use the knowledge acquired in the previous evaluations of WLCSSLearn_p in order to select a single set of evolutionary parameters and to focus only on the template generation analysis.

Since the chromosomes used by WLCSSLearn_t are wider than those employed in WLCSSLearn_p (the templates can be longer than the the sum of $b{\cdot}3$ and bt) and also more complex (the values of single genes in the chromosome are discrete number rather then binary bits), we aim at making this evaluation simpler without sacrificing its reliability. Thanks to the knowledge built in the previous evaluation and since WLCSSLearn_t is slower than WLCSSLearn_p, we select a single set of evolutionary parameters for all the tests in this analysis. In the following, we rather focus on the template generations itself showing:

- the generation of a single template, for a single dataset, over multiples steps of the evolutionary training, displaying the individuals in the population and their lengths's distribution;
- the similarity/difference of the best performing generated templates compared to

the MRT, for all the actions in a single datasets;

- an analysis of the recognition performance/computation saving trade-off. The tests for this analysis were conducted for all the classes, for all the datasets.

All the tests are performed using the following global parameters: $pop = 64$, $rank = 16$, $elitism = 3$, $iter = 500$, $cr = 0.3$ and $mt = 0.1$. Also, as we want to focus only on the trade-off, we keep constant the minimum and maximum possible length of generated templates; they are expressed as percentage of the starting length, which is computed as the length of the MRT for that action. They are set respectively to 5% and 100% of the length of MRT. Finally, we set the probability of enlargement or shrinking or leaving the template as it is to be equally likely, i.e. $en\_p = 0.33$ and $sh\_p = 0.33$.

6.3.1 Shape and length of generated templates

Figure 6.3.1 shows how the templates of a given action class evolve in shape and length over the course of the evolutionary training with WLCSSLearn_t. We sample the population of individuals at several steps of the generation. We then order them by similarity and selected a sub-sample of the population at every step. We compute the distribution of the lengths of the templates at every step and it is possible to notice how gradually the length of the individuals reduces as well as their visual complexity. This latter aspect becomes more clear in Figure 6.3.2.

The figure shows a comparison between the individuals with the highest fitness and the MRT of the corresponding action class. The figures indicates that the evolutionary process tends to reduce the length of the template and increase the saliency only of features that are most characteristic of the segments in each action class. This is a clear example of using the warping as an advantage, as shorter and simpler templates can still be used for longer time series classification.

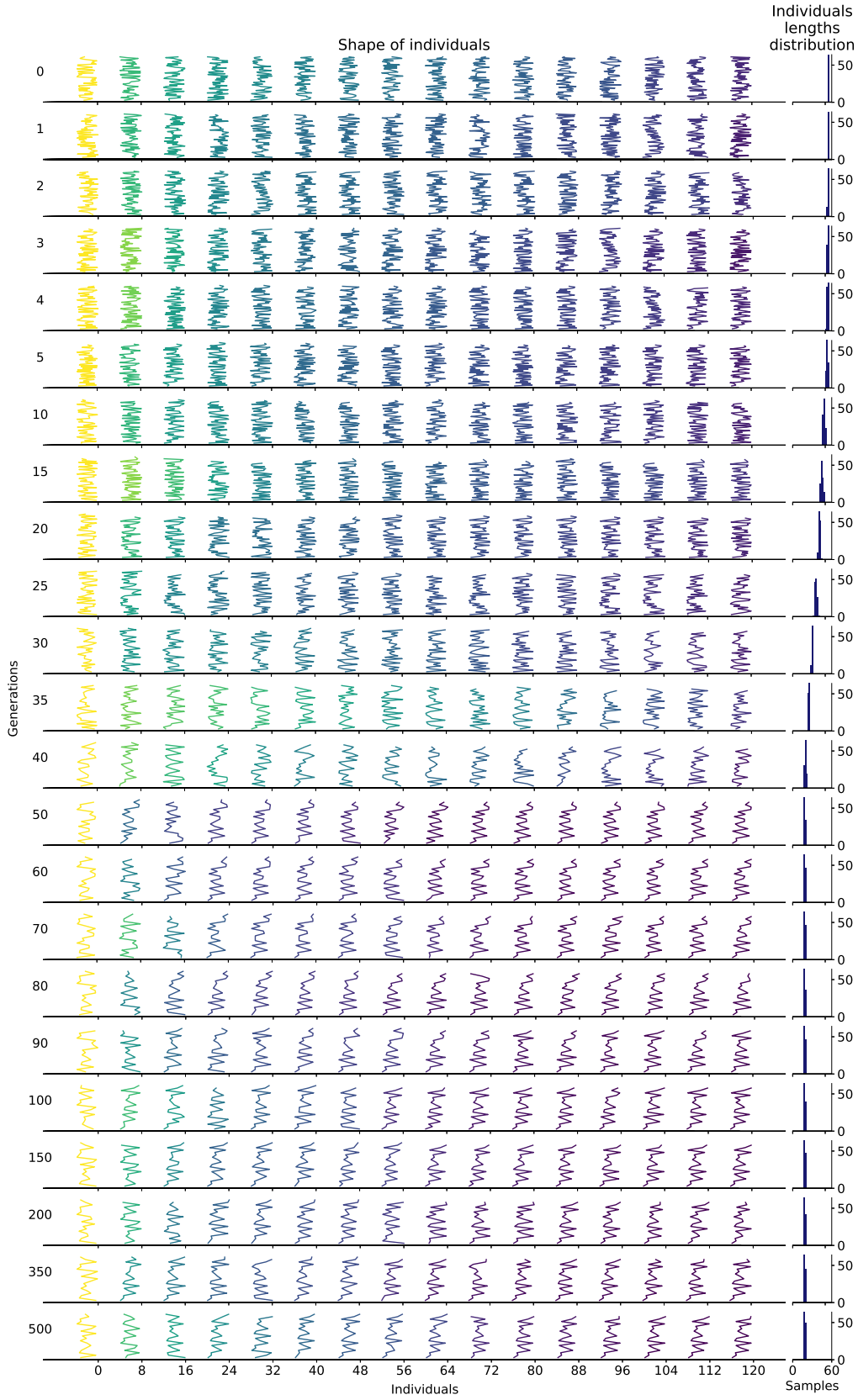


Figure 6.3.1: *Population of individuals during variable template generation for a single action class of the hci_guided dataset. The individuals are sampled at different step of the evolution. The color-map indicates the fitness score, normalized per iteration. The histograms on the right side show the distribution of the lengths of all the individuals (not only the ones shown).*

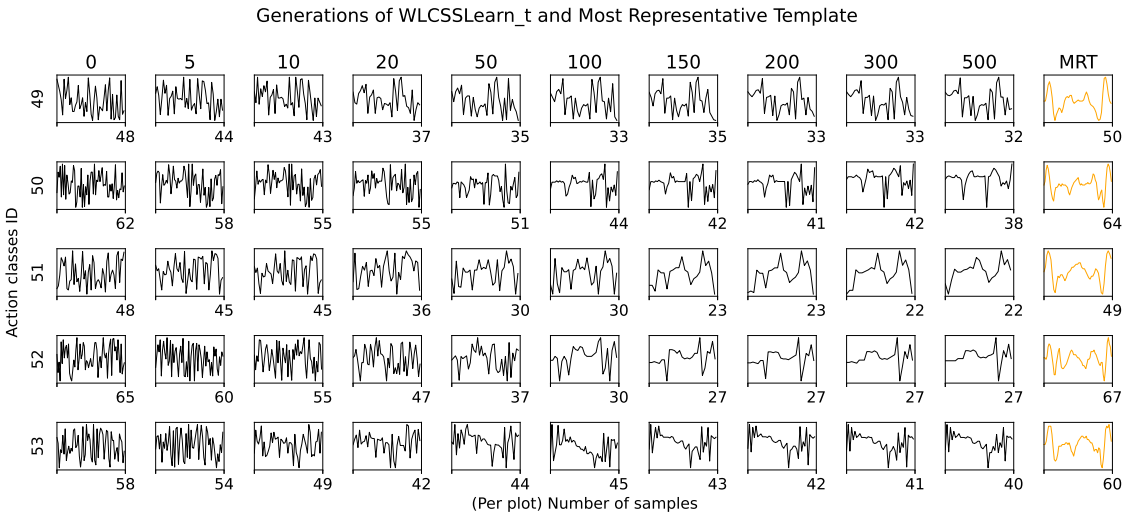


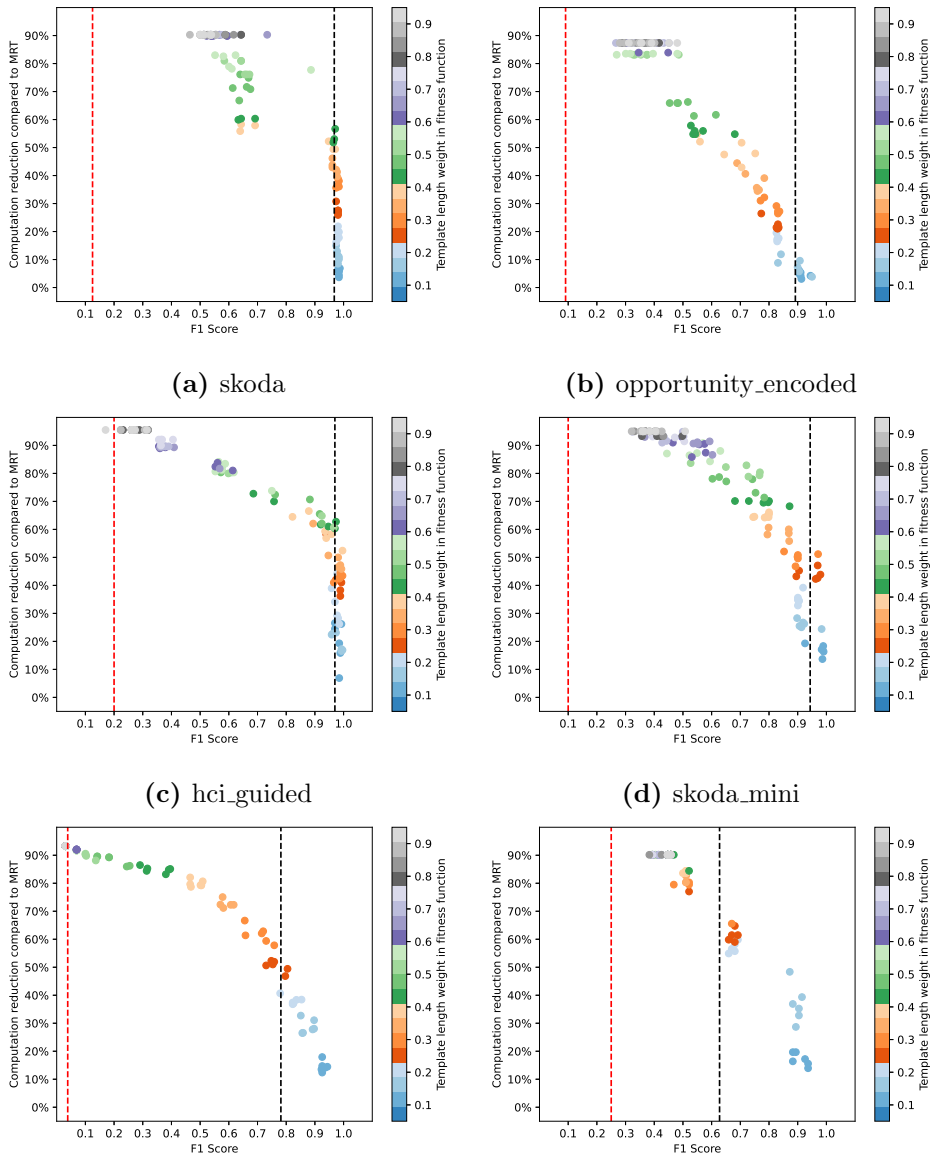
Figure 6.3.2: *Comparison of best fitting variable templates generated with WLCSSLearn_t (black), sampled at several generations, and MRT (orange) for hci_guided dataset. Over all the iteration, the length of the best individual reduces as well as its visual complexity.*

6.3.2 Computational reduction vs performance recognition

We evaluate to which extent the parameter λ in Equation 6.1 affect the trade-off between computational reduction vs recognition performance. Multiple results must be noted:

- Figure 6.3.2 displays that the parameter λ effectively allows to tune the trade-off between recognition performance and computation saving. An increase in the weight of the fitness function $F(p_k)$ from Equation 6.1 does affect the reduction of the generate template. This trade-off is more finely configurable with hci_table and skoda_mini dataset than for the opportunity_encoded, skoda and beach_volleyball datasets. This is a consequence of using a single template per actions and the ratio between number of action classes and number of segments per action. If the template is not good enough (e.g. too short or not optimized in shape), all the segments for that actions would affect the recognition performance reducing the F1 score at once. This it confirmed by the fact that the big jumps in the plots, when present, happen on the x-axis representing the F1 score.

- Table 6.3.1 presents a comparison of recognition performance calculated as F1 score between the MRT approach and WLCSSLearn_t, for reductions between 10% and 50%. The templates generated with WLCSSLearn_t increase the recognition by 7% on average across all the datasets, with a maximum of +15% for hci_table dataset, with a length reduction between 10% and 20%. When a larger reduction between



(a) skoda **(b)** opportunity_encoded

(c) hci_guided **(d)** skoda_mini

(e) hci_table **(f)** beach_volleyball

Figure 6.3.3: *Action recognition performance and computation reduction of WLCSS-Learn_t with variable length templates generation. The x-axis is the F1 score. The y-axis is the percentage of less computation required due to the generation of shorter templates compared to the MRT. The red dashed line represents the chance-level classification F1 score, while the black dashed line represents the F1 score for the MRT. The color-map represent the weight of the template length in the fitness function during the evolution (lw).*

Dataset	F1 score - MRT	F1 score with WLCSSLearn_t			
		10%-20% c.r.	20%-30% c.r.	30%-40% c.r.	40%-50% c.r.
skoda	0.97	0.98 (+0.01)	0.98 (+0.01)	0.98 (+0.01)	0.97 (-0.00)
opportunity_encoded	0.89	0.83 (-0.06)	0.82 (-0.07)	0.77 (-0.12)	0.70 (-0.19)
hci_guided	0.97	0.99 (+0.02)	0.97 (+0.00)	0.98 (+0.01)	0.99 (+0.02)
skoda_mini	0.94	0.98 (+0.03)	0.92 (-0.02)	0.90 (-0.04)	0.93 (-0.01)
hci_table	0.78	0.93 (+0.15)	0.88 (+0.09)	0.84 (+0.06)	0.79 (+0.01)
beachvolleyball	0.63	0.91 (+0.28)	0.89 (+0.27)	0.90 (+0.27)	0.87 (+0.24)
Mean difference	-	+0.07	+0.05	+0.03	+0.01

Table 6.3.1: *Comparison of F1 score computed using* MRT *and the average F1 score obtained with generated templates, for different range of computational reductions (c.r.), indicated as header of columns 3-6. In parenthesis, the difference between the F1 score for each range and the F1 score with MRT is reported. Finally, the average difference of F1 score obtained using WLCSSLearn_t across multiple datasets is also reported as last row.*

40% and 50% is enforced, WLCSSLearn_t has comparable performance to MRT within a margin of 1% on average across datasets.

- Table 6.3.2 shows a comparison of the F1 score resulting from chance-level classification for each dataset and the score obtained with WLCSSLearn_t, for large ranges of computational cost reduction (50%-90%). When the maximum reduction is applied (range 90%-80%), WLCSSLearn_t enables recognition performance 33% higher than chance-level classification on average across multiple datasets. For the range 80%-70%, the F1 score difference compared to chance-level with WLCSSLearn_t attain a 48% increase. As expected, the F1 score increases even further for lower reduction ranges. While for high computational reductions values the F1 score with WLCSSLearn_t may seems very low in absolute values, i.e. only 24% for hci_table, it has been demonstrated that any recognition accuracy greater than guessing can be greatly improved by fusing more sensors [100].

Dataset	F1 score - Chance	F1 score with WLCSSLearn_t			
		90%-80% c.r.	80%-70% c.r.	70%-60% c.r.	60%-50% c.r.
skoda	0.12	0.58 (+0.46)	0.67 (+0.54)	0.66 (+0.53)	0.81 (+0.68)
opportunity_encoded	0.09	0.36 (+0.27)	0.47 (+0.38)$^{(*)}$	0.51 (+0.42)	0.58 (+0.49)
hci_guided	0.20	0.48 (+0.28)	0.77 (+0.57)	0.92 (+0.72)	0.95 (+0.75)
skoda_mini	0.10	0.59 (+0.49)	0.71 (+0.61)	0.80 (+0.70)	0.88 (+0.78)
hci_table	0.04	0.28 (+0.24)	0.54 (+0.50)	0.69 (+0.65)	0.75 (+0.71)
beachvolleyball	0.25	0.51 (+0.26)	0.50 (+0.25)	0.68 (+0.43)	0.68 (+0.43)
Mean difference	-	+0.33	+0.48	+0.58	+0.64

Table 6.3.2: *Comparison of F1 score between chance-level classification, defined as* $\frac{1}{num\ of\ action\ classes}$*, and the average F1 score obtained with generated templates, for different ranges of computational reductions (c.r.), indicated as header of columns 3-6. In parenthesis, the difference between the F1 score for each range and the chance-level classification F1 score is reported. The average difference of F1 score obtained using WLCSSLearn_t across multiple datasets is also reported as last row.* $^{(*)}$ *The value for the range 80%-70% of computation reduction for the opportunity_encoded is not available from the experiment (see Figure 6.3.3b). Hence, the value in the table is an interpolation using 2nd degree polynomial regression fitted on the existing values.*

6.4 Discussion

While we demonstrated the effectiveness on WLCSSLearn_t for WLCSS, the algorithm can be applied to any template matching method. Similarly to WLCSSLearn_p, the genetic operators of WLCSSLearn_t are generic enough that, not only it can be deployed for other TMM in HAR, but also in different pattern recognition domains. This would bring the benefit of computational savings thanks to the variable length of the templates to other TMMs and applications.

The fitness function of WLCSSLearn_t provides a measure of how good an individual is at distinguish between segments of action class g from those of different action classes ($\overline{g}$). The standard F1 score (and therefore the fitness function used by WLCSSLearn_p) is not a good candidate as this measure must be continuously defined for matching/mismatching segments rather than being the result of just a discrete counting of match/mismatch. This measure, provided by $\Upsilon(p_k)$ in Equation 6.1, is computed by using D_g and $D_{\overline{g}}$ as the 5-th and the 95-th percentile of the matching scores respectively of the segments of action class g and $\overline{g}$. This accounted for 5% of outlier segments that could disrupt the evolution but these values can variate in order to increase/decrease sensitivity and specificity. Decreasing the percentile value for D_g increases the sensitivity allowing less

outliers in the segments of action class g. On the other hand, increasing the percentile value for $D_{\overline{g}}$ would increase the specificity by excluding a higher number of segments of action classes different from g. However, changes in these values could affect the evolution and the recognition performance: if the percentile values are closer, the evolution would converge faster at the cost of less recognition performance. Instead, farther values of the percentile would increase the recognition performance at the cost of potentially jeopardising the convergence of the evolution. We reckon that further evaluations on this matter can be beneficial.

The terms composing the fitness function in Equation 6.1 are normalized in order to be combined. While we used the min-max normalization for the length of the individuals, we had to choose a different and more peculiar solution for normalizing $\Upsilon(p_k)$. This originate from the difficulty of defining a minimum and maximum for $\Upsilon(p_k)$ that would be realistic in our domain. From the equation of WLCSS (eq. 4.4) it is possible to define a theoretical minimum and maximum for the matching costs (D_g and $D_{\overline{g}}$) respectively as:

$$\min \Upsilon(p_k) = P \cdot len(T) \cdot \max (t_i - s_j)$$

$$\max \Upsilon(p_k) = R \cdot len(T)$$

where P and R are penalty and reward, $len(T)$ is the length of the template and $\max t_i - s_j$ is the maximum possible difference between two samples of T and S. However, the minimum score $\min \Upsilon(p_k)$ can occur only in case the difference of every sample of T and S is equal to $\max t_i - s_j$, considering encoding/quantization (see 4). This is unlikely to happen in a real scenario. Therefore, we found out that, due to the wide interval between the theoretical $\min \Upsilon(p_k)$ and $\max \Upsilon(p_k)$, using a min-max function in this case failed. Indeed, the values of $\Upsilon(p_k)$ normalized using these minimum and maximum did not have enough variability in order to correctly drive the evolution. For this reason, we opted for the sigmoid function described in Section 6.2. We recognize however that this is a custom choice and that other options might be suitable. Moreover, our sigmoid function presents a parameter b defining the slope of the function that we decided empirically but that could benefit further exploration.

6.5 Conclusion

In this chapter, we presented WLCSSLearn_t: the first training algorithm for generating variable length templates with a configurable trade-off between recognition performance and computation saving. We evaluated this algorithm and the configurable trade-off on 6

datasets of actions including different kind of data all from the domain of wearable and ubiquitous computing.

WLCSSLearn_t was able to generate variable lengths templates that:

- can be configured through a single parameter λ between computational reduction and recognition performance
- have better recognition performance than the naive approach of selecting the MRT up to 15%, even with up to 10% of computation saved.
- have comparable performance, between +1% and +7%, to MRT with up to 50% of computational savings across all the datasets
- perform +33% better than chance-level classification when a 80%-90% computational reduction is enforced.

Chapter 7

WLCSSCuda: supporting TMM training with GPGPU

*This chapter describes a GPU accelerated implementation of WLCSS. Exploiting the General-Purpose computing capabilities of NVIDIA GPU, WLCSSCuda is capable of running large number of multiple instances of WLCSS simultaneously. WLCSSCuda finds great applications in the WLCSSLearn presented in chapters 5 and 6, where a highly parallelized implementation of WLCSS greatly increase the speed of the evolutionary training. The content of this chapter has been published in [**VI**].*

7.1 Introduction

WLCSSCuda is a GPU accelerated implementation of Warping Longest Common Subsequence (WLCSS) using CUDA. This is a framework developed by NVIDIA for general purpose processing on GPUs (GPGPU) [101]. In the past, GPUs have been used to accelerate other TMMs, such as Dynamic Time Warping [102] and Longest Common Subsequence [103]. However, to the best of our knowledge, this is the first GPU implementation of WLCSS.

WLCSSCuda allows to execute a high number of instances of the TMM simultaneously. This can be exploited during training which is not done on the embedded sensor node, but rather on much more capable and general purpose computers. For this reason, WLCSSCuda found great application in WLCSSLearn, where a high number of WLCSS computations are required at each generation of the evolutionary algorithm.

We compare WLCSSCuda with a multi-core CPU implementation of WLCSS and show an drastic speedup in the matching score computation.

7.2 WLCSSCuda

WLCSSCuda is our GPU implementation of WLCSS built using CUDA. Thanks to the high number of cores in modern GPUs, it is possible execute tasks, called kernels, with a high level of parallelization. CUDA abstracts the physical structure of the GPU in a grid of blocks. Each block can be addressed using a 1D, 2D, or 3D index. A block is a unit made by several threads that can be computed in parallel or serially on a GPU core. A 3D indexing is provided for the thread too. The scheduling of the threads's execution is transparent to the user. The number of maximum blocks and maximum threads per block depends on the GPU.

WLCSSCuda structures the computation using a 2D grid for the blocks and 1D structure for the threads, as shown in Figure 7.2.1: a single template/stream pair is assigned to each block. Then, within that block, a thread is used for every parameters set. During initialization, all the templates, streams and parameters are transferred to the GPU memory. Then, each kernel computes the pointer to the templates, segments and WLCSS parameters in memory using respectively the indexes q and k for the blocks, and r for the threads. Each triad template/segments/parameters is used by only one kernel which is executed by a single thread. Finally, when the matching scores are computed, they are transferred from the GPU memory back to the main memory. WLCSSCuda computes the entire score between the template and the segment.

WLCSSCuda supports 2D and 3D encoded templates, as presented in Section 4.2.1. By selecting the correct encoding at initialization time, WLCSSCuda uses the correct distance matrix to compute the distance $f(t_i, s_j)$ between samples.

WLCSSCuda is developed in C++ with a Python wrapper for loading the data and reading the results.

7.3 WLCSSCuda vs WLCSS

We evaluated to which extent WLCSSCuda could accelerate multiple template matching, taking into account the time required to transfer the data to/from the GPU, which has been demonstrated to be a bottleneck in CUDA applications [104]. We compared the execution of WLCSSCuda on 4 GPUs and a multi-threaded WLCSS on 4 CPUs (see Table 7.3.1). We simulates 4 test scenarios in which a different number of streams, templates and WLCSS parameters were employed (Table 7.3.2). We reported the average of 10 executions. The CPU implementation of WLCSS uses all the available threads in the CPU to run always

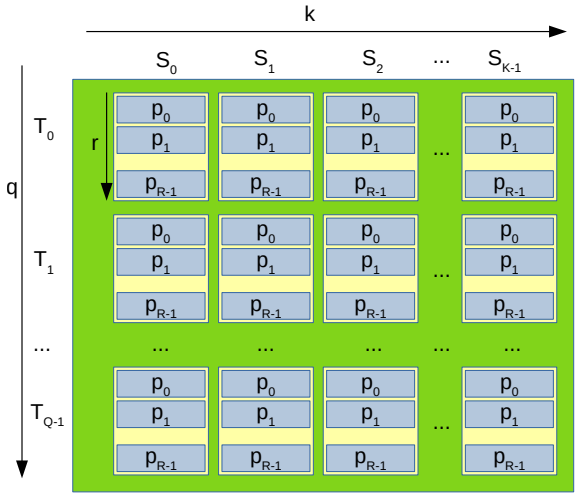


Figure 7.2.1: *WLCSSCuda structure: blocks are represented in yellow. A template* $\mathbf{T_q}$ *and a segments* $\mathbf{S_k}$ *are assigned to each block. A set of parameters* $\mathbf{p_r}$ *is assigned to each thread, displayed in blue.*

the maximum number of possible WLCSS simultaneously.

We used OPPORTUNITY Dataset [48] as source of templates and segments, selecting a random subset of movements segments for each test, to make them more realistic. The average length of the templates (and streams) is 98 samples, with a standard deviation of ± 46. The same subset of movements was used for WLCSSCuda and the CPU implementation in each test.

Table 7.3.3 shows the average time for each test for every CPU and GPU. For WLCSSCuda, all the values include the time to the transfer of data to/from the GPU memory. As we expected, the GPUs are faster in every scenario we evaluated. Moreover, it is possible to notice how WLCSSCuda scales better when the number of instances increase. Test *d* requires 100 times more evaluations than test *a*; the GPUs take on average only 85 times the time required by test *a* while the CPUs take 110 times more than *a*, on average, across all the different models.

7.4 Discussion and Conclusion

We presented WLCSSCuda, a GPU accelerate multiple TMM. We demonstrated that WLCSSCuda can drastically increase the computation of multiple template matching, with an increase of 67 times in the best case compared to a multi-threaded CPU approach.

[1] https://www.nvidia.com/en-gb/

[2] https://www.amd.com/en/products/ryzen-threadripper

[3] https://ark.intel.com

Table 7.3.1: *CPU and GPU tested. For more details about the GPUs and CPUs (AMD and Intel), visit respectively* [1], [2], [3]

GPU Model	CUDA cores	Cores Frequency	Memory
GTX 1080 Ti	3584	1645 MHz	11 GB GDDR5X
GTX 1050 Ti	768	1418 MHz	4 GB GDDR5
GTX 970	1664	1317 MHz	4 GB GDDR5
Titan XP	3840	1582 MHz	12 GB GDDR5X

CPU Model	Frequency (Max Turbo)	# of Cores	# of Threads
AMD Ryzen 1900X	3.8 GHz (4 GHz)	8	16
Intel i7-8750H	2.2 GHz (4.1 GHz)	6	12
Intel i7-4770K	3.5 GHz (3.9 GHz)	4	8
Intel i7-6700	3.4 GHz (4.0 GHz)	4	8

Table 7.3.2: *The number of templates, streams, parameters sets and the total number of WLCSS computation is shown, for each test scenario.*

Test	# Templates (Q)	# Streams (K)	# Params. sets (R)	Tot. WLCSS
a	10	1000	10	100000
b	20	2000	10	400000
c	50	5000	10	2500000
d	100	10000	10	10000000

Table 7.3.3: *Results of WLCSS running on the 4 GPUs and the 4 CPUs. The values are in seconds and they are averaged across multiple running. The improvements are computed respectively between the best CPU against the worst GPU, and viceversa.*

Platform / Test	*a*	*b*	*c*	*d*
Titan XP	0.49 ± 0.04	1.53 ± 0.15	9.66 ± 0.61	38.29 ± 1.87
GTX 1080 Ti	0.55 ± 0.07	1.98 ± 0.14	12.77 ± 1.51	51.38 ± 6.22
GTX 1050 Ti	0.79 ± 0.13	2.92 ± 0.26	18.15 ± 0.67	72.04 ± 3.59
GTX 970	0.91 ± 0.16	4.00 ± 0.67	18.97 ± 0.86	70.74 ± 3.69
AMD 1900X	8.29 ± 0.89	32.27 ± 1.74	231.75 ± 31.59	932.69 ± 86.19
i7-8750H	17.50 ± 2.57	81.23 ± 5.30	555.26 ± 15.39	2140.20 ± 141.37
i7-6700	20.35 ± 1.07	80.18 ± 8.28	523.05 ± 34.30	2008.19 ± 81.33
i7-4770K	22.78 ± 3.20	99.62 ± 10.00	647.21 ± 45.91	2452.15 ± 170.12
Improvement	10-46 times	8-65 times	12-67 times	13-64 times

However, there is still room for improvement: we plan to evaluate different organizations of data in order to better use the block/thread CUDA structure. Moreover, we aim to make WLCSSCuda automatically adapting such structure according with the number of templates/streams/parameters sets in order to increase the performance even further. Finally, WLCSSCuda is available as open source software at the address https://github.com/sussexwearlab/WLCSSCuda.

Chapter 8

Segment classification from poorly annotated data: a drinking movement recognition case study

*As we have seen in the previous chapters, datasets can be affected by bad annotations. Template Matching Methods and our training algorithm can be used to solve this issues in order to recover a highly valuable but poorly annotated dataset. This chapter presents a case study of recovering a dataset of drinking movements. The content of this chapter has been published first in [**IV**] and then extended and improved in [**V**].*

8.1 Introduction

Action recognition has applications in several fields such as healthcare and sports [105]. In order to create a reliable movement recognition system, it is important to have a well-annotated dataset [2]. However, creating high-quality datasets may require to rely on lab-like environments, with limited ecological validity [48]. Activity recognition research generally strives to employ datasets with unrealistically "perfect" ground truth annotations. In an ecologically valid data collection, however, it is likely that a highly valuable dataset is acquired, but that only poor quality annotations are available.

Experience sampling (ES) is a real-time annotation approach done by users themselves a mobile device [106]. This allows more ecologically valid data collection in everyday life (e.g. no need to video record the experiment). However, ES can lead to the following issues: i) the synchronisation between the activity performed and the label annotated by the user is generally of poor quality, with the user annotating the activity after the event,

or combining multiple activities in a single annotation; ii) the user may forget to label an event, iii) the user may annotate an activity with the wrong label.

In this work, we investigate how to make sense of a highly valuable dataset comprising a large number of drinking movements, which have been annotated through ES, leading to numerous deficiencies in the annotation quality. The dataset contains drinking movements annotated by the users with a mobile application. The dataset was collected in an office environment using a 3-axis accelerometer and it is made by 8825 "sensor events", with 1808 "drink events" annotated by users through ES. Using this dataset, we aim to address two main challenges: i) to understand why the quality of the annotation is low and consequently how would it be possible to improve in future data collection and ii) to understand whether it is still possible to use such big dataset without relying on the annotations for spotting drinking movements and how. This work is based on the research presented in [**IV**]. The main contributions are:

- A study of the annotations. We analyse the user annotations, their distribution in time during the data collection, and their relation to the sensor events, in order to understand the causes of the low quality and where the data collection process can be improved.

- A template matching approach, based on Warping Longest Common Subsequence (WLCSS) [26], to extract a subset of drinking movements, within a certain level of confidence. This subset will allow the dataset to be used for research purposes. As extension of the previous work, in this chapter, we evaluate multiple sensors channel in order to obtain a more precise selection of drinking movements.

- An unsupervised algorithm (K-Means) adapted to template matching. This algorithm is a new variation of K-Means [107] where the WLCSS is used as distance measure. It allows to cluster movements based on the raw signal of the sensors. At the same time, it clusters movements taking in account the variation in the way they can be performed, by using WLCSS which has been successfully used for robust movements detection [26].

8.2 Related work

The quality of annotations obtained through ES can be poor [106]. Annotations issues can include time shift of a label with respect to the activity, as well as wrong or missing labels [108].

Some approaches suggested to improve ES with manual re-annotation [106]. This is not feasible economically for a large datasets. Moreover, despite re-annotation the quality may still be insufficient for the training of machine learning algorithms [106]. The impact of ES on activity recognition has been studied in [109]. However, the authors simulated the ES in a controlled environment and they used only the data corresponding to the user annotations.

The problem of poorly labelled data can be tackled during the annotation itself or during the training of the machine learning algorithm. A method useful to reduce the effort of the users while annotating their activity has been proposed in [110]. The authors suggested a one-time point annotation method that requires the users to only label a single moment per activity rather than specifying the beginning and the end. The method then recognizes automatically the boundary of the activity in the annotated signal. Nevertheless, it requires that the labels are within the execution interval of the activity.

Unlabelled or poorly labelled data are available in big quantities nowadays due to the large diffusion of sensing devices, such as smartphones and wearables devices. For this reason, methods such as semi-supervised learning, active learning and unsupervised learning have been applied in order to extract useful information from sparsely annotated data. A combination of active learning and semi-supervised learning has been studied in [111]. The authors used a dataset of daily activities collected with two subjects wearing accelerometer sensors and motioned tracked with infrared sensors. This approach however uses a decision window of 30 seconds long and thus is not suitable for recognizing movements that occur in a short time. Unsupervised learning has been successfully applied to activity recognition in [112] and more recently in [113]. In the latter, an activity discovery method based on clustering is proposed to help with ES, although it is designed for periodic movements rather than sporadic actions. Unsupervised learning has been applied to actions clustering in [114], where a K-Means clustering has been evaluated specifically for hand movements.

Several studies have tried to address the challenge of activity recognition from poorly annotated data. While most of them used synthetic dataset and focused on periodic or long activity (such as walking, running, etc.), to the best of our knowledge none of them applies to drinking movements collected in a real-life office environment.

8.3 Dataset

The dataset was collected by providing a set of mugs to 60 users (one mug per user) in an office environment. Each user collected data for a period of 4 days. Each mug was instrumented with a logger comprising a 3-axis accelerometer [115]. The loggers were placed in a hollow at the bottom of each mug. As the mugs were custom made, the positioning of the loggers was not the same in all mugs. The loggers sample acceleration at 20 Hz, with a timestamp in ms. In order to save power, they start logging acceleration when a movement is detected. After 5 seconds of inactivity they automatically stop the recording, without record the inactivity period. We use the term *sensor events* to refer to every recording performed by the loggers that lasts at least 4 seconds (as configured on the loggers for this data collection). Therefore, sensor events can occur for a variety of reasons: moving the mug on the desk, washing it, drinking from the cup, etc.

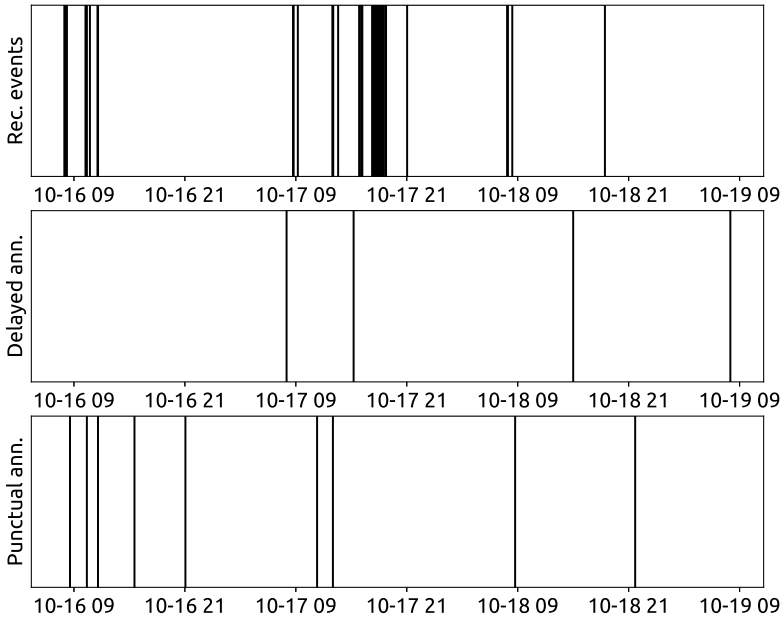
The data annotation was performed through experience sampling by the users themselves. They labelled each drinking event manually using an Android application installed on their smartphones. Each annotation could be *punctual* or *delayed*. An annotation is considered punctual when it was entered immediately after the drinking event. It is considered delayed, when it refers to an event in the past. The users could specify in the application whether their annotation was punctual or delayed. However they did not have to provide an indication as to how much the delay was. Furthermore, there were no guidance indicating after how much time an annotation should be considered "delayed" rather than "punctual".

The resulting dataset is made by 8825 sensor events, 1808 user annotations, of which 1477 marked as "punctual" and 331 as "delayed". The percentage of annotated movements with respect to the total amount of sensor events is of 20.5%.

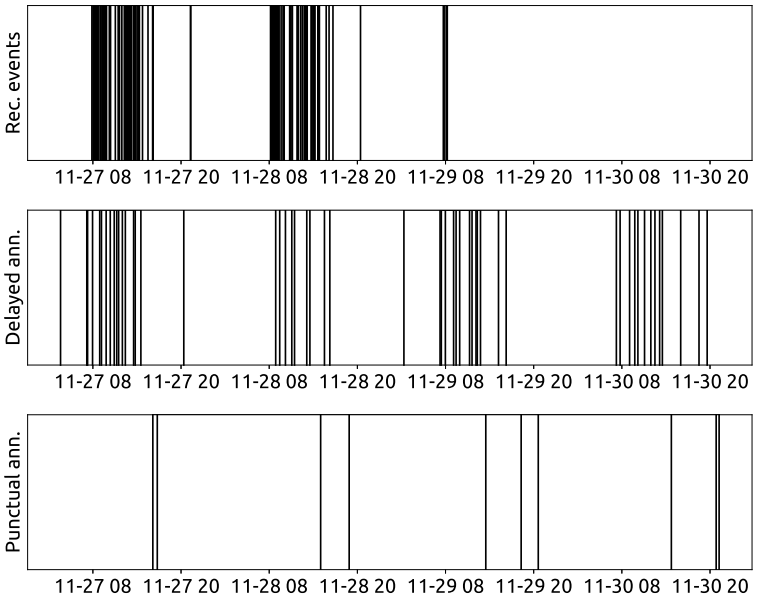
8.4 User annotation analysis

We aim to analyse the causes of the poor annotations in order to improve future data collections, as well as helping during the next steps of this study.

Figure 8.4.1 indicates the main challenge of the annotation protocol, which is how users understood differently how and what to annotate. The data collection protocol did not require participants to annotate drinking solely when using the instrumented mugs: they could annotate drinking as well when using regular mugs. It might happen that users annotated drinking events performed using other cups. The protocol did also not specify



(a) User 109



(b) User 461

Figure 8.4.1: *Example of annotations of two different users, over the 4 days period. The start time of the sensor events are displayed in the top plot of each figure, one thin line per event. The delayed and punctual annotations inserted by the users are displayed respectively in the second and the third plot of both figures. The X-axis reports date and time, in the format "MM-DD HH". It is also possible to notice the differences in the way two users annotated the drinking events.*

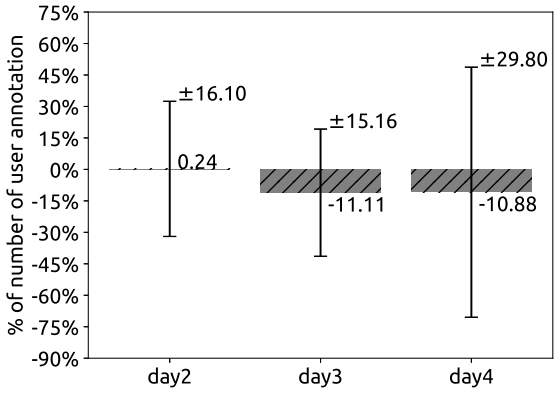


Figure 8.4.2: *Change in the user commitment in the annotation during day 2, 3, and 4. The grey bars represents the percentage of annotations for each day with respect to day 1. Day 2 displays an increment of 0.24%; Day 3 and 4 an average decrease of 11% in the number of annotations. The vertical bars represents the standard deviation for each day.*

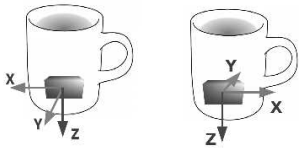
what to consider as a "drinking event". Users could interpret it as referring to a single sip, multiple sips, or the act of drinking the entire cup. It is also possible to notice how the annotations are not well aligned with the sensor events.

We also studied the distribution of the labels, per user, over the 4 days of data collection. It could help to understand the users' commitment in annotating their drinking movements, assuming they were keeping the same drinking habits among all the days. This may be useful in order to spot days for which the annotations can be more reliable. The results are presented in Figure 8.4.2. While there is no significant change between day 1 and day 2, with an average increase in the number of annotation of 0.24%, starting from day 3 the engagement decrease by 11% on average among all the users. The plot shows also a great variability in the data: there were users that increased their commitment over the 4 days, as well as users for which the commitment decreased over the 4 days.

From the analysis of the annotations, it can be concluded that they were not reliable enough to be used together with the data for the supervised classifier training.

8.5 Movement classification

In order to make the collected dataset useful for drinking movement recognition, each event recorded by the sensors had to be classified in drinking/non-drinking. As highlighted previously, the users annotations cannot be used as-is as they are not accurate enough. A manual relabelling of the entire dataset was unfeasible given the lack of any video recordings.



(a) **(b)**

Figure 8.5.1: *Examples of orientation of the loggers in the custom made mugs. Although X and Y axis can be different in each mug, the Z-axis is always facing downwards.*

We developed an approach based on a template matching method (TMM) to automatically spot a subset of events which are believed to be drinking movements with a certain confidence value. The approach then uses few events which are manually identified as drinking events with high confidence to train the TMM.

8.5.1 Data processing and training set selection

We used a heuristic method to select a few sensor events as the training set. We performed a few drinking movements using the same instrumented mug in order to chose the best sensor channel for template matching.

The chosen channel (or channels) must be orientation independent, as there was no information about the positioning of the logger in the mug (Figure 8.5.1). We discovered that the Z-axis of the accelerometer quite clearly indicates the movement of lifting the cup to drink. Also, despite different orientation of the loggers, this axis was always perpendicular to the bottom of the mug. A template of such movement is displayed in Figure 8.5.2. A subset of movements visually similar to this template was selected manually from the entire set of the available movements. We selected this subset trying to include some variability in the way the drinking movements were performed. Another subset of non-drinking movements was selected too, choosing the templates that were very visually different from the drinking movements. The training set is displayed in Figure 8.5.3. It is formed by 78 events: 37 drinking movements (8.5.3a) and 41 non-drinking movements (8.5.3b).

However, lifting the cup does not necessarily mean that a drinking was actually performed. For this reason, in order to better detecting the rotation of the mug due to the drinking, we also used the magnitude of the acceleration on X-Y plane as additional information. This was also due to the lack of the gyroscope on the loggers. The X-Y plane was chosen because it was always parallel to the bottom of the mug (Figure 8.5.1). The magnitude was computed as $m_{xy} = \sqrt{x^2 + y^2}$. A template for a drinking movement represented by the magnitude is shown in Figure 8.5.4. A different training set based on the

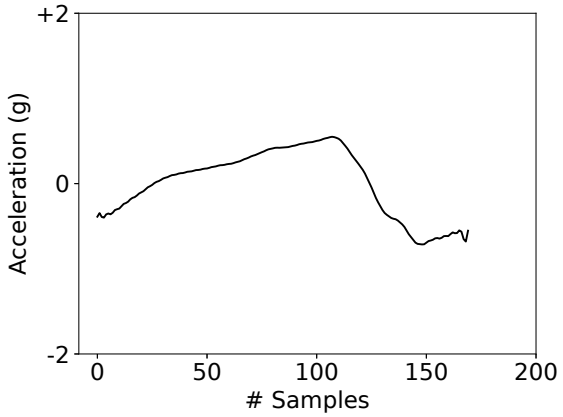


Figure 8.5.2: *Template of a drinking movement, performing a single sip, displaying the acceleration on the Z-axis.*

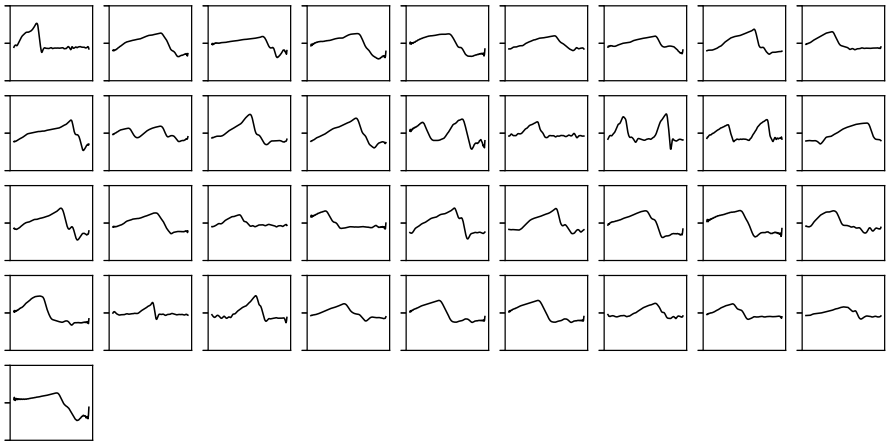
template of the magnitude was chosen. The choice was based on visual similarity to such template. A subset of non-drinking movement was also chosen for the XY magnitude The training set of drinking and non-drinking movement for the XY magnitude is displayed in 8.5.5. The drinking movements were selected only when they corresponded to a lifting of the mug (Z-axis).

All the instances in the dataset were filtered using a Butterworth low pass filter with cut off frequency set to 10 Hz. They were also resampled to a fixed number of samples. The number of samples was selected as the average length of a drinking movement, which is 170. This step was performed in order to reduce the impact of non-drinking events that can last longer time than drinking movements (e.g. washing the cup, moving the cup around the office, etc.).

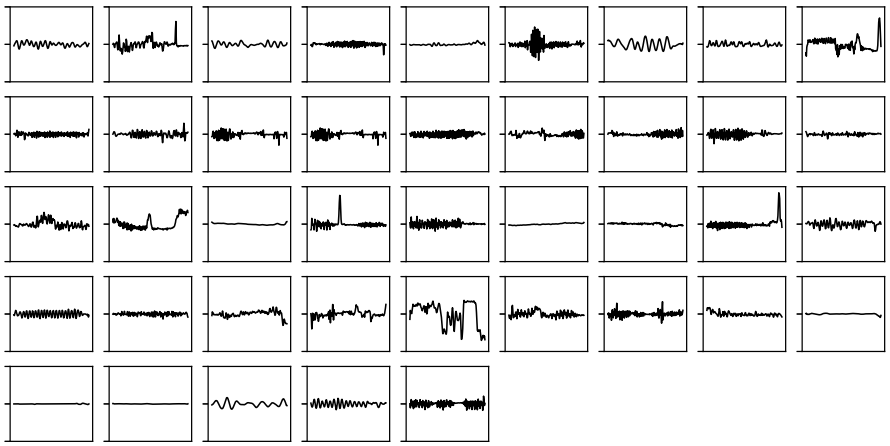
In order to evaluate which is the best channel for drinking movement recognition, we decided to match the templates for Z-axis signal and XY magnitude separately.

8.5.2 WLCSS training

The selected training segments are used with WLCSSLearn_p to optimise the values of R, P, ϵ and V to maximise the ability of WLCSS to distinguish drink from non-drink. We used our training in order to optimize the values of the parameters starting from a randomly generated population. Each individual of the population is an array containing the 4 parameters. The EA evolves this population through the usual selection, mutation and crossover operators [116]. Here, the F1 score is used for the selection. The optimization process stops after a predefined number of iterations, in this case 500.



(a) Drinking movements



(b) Non-drinking movements

Figure 8.5.3: *Training set of movements using the Z-axis of the accelerometer. 8.5.3a displays the templates chosen as drinking movements. 8.5.3b shows those selected as no-drinking movements. All the plots show the templates downsampled to the fixed length of 170 samples (X-axis). The Y-axis represents the acceleration within a range of* $\pm$ *2g.*

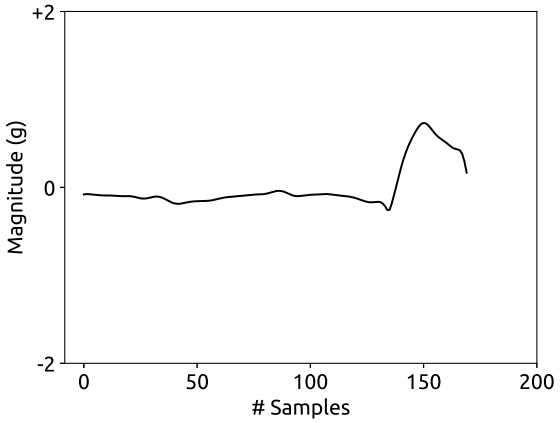


Figure 8.5.4: *Template of a drinking movement, performing a single sip, displaying the magnitude of the acceleration on the X-Y plane.*

8.5.3 Confidence computation

Given the unreliability of the labels in the dataset, it is not possible to evaluate precisely the correctness of a match for this particular dataset. For this reason, we opted to provide a confidence level for each movement as output of our method rather than a simple match/no-match. The confidence level was assigned using Four Parameter Logistic Regression:

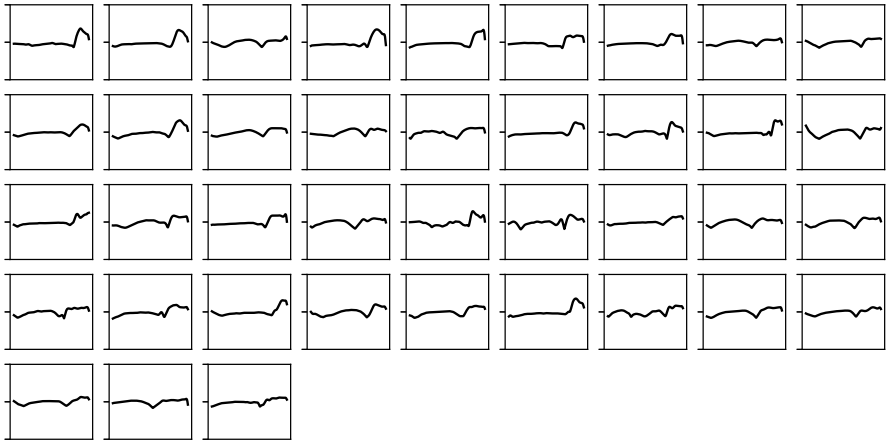
$$\upsilon = \delta + \frac{\alpha - \delta}{1 + (\frac{D}{V})^{\beta}}$$

where υ is the confidence level, D is the matching cost, and V is the threshold. The function, displayed in Figure 8.5.6, provides a confidence value in the range [0:1]. The range is defined by the parameters $\alpha = 0$ and $\delta = 1$. The parameter β, which define the slope of the function curve, was set manually to 5. Using a fixed interval for the confidence makes its value unrelated from the absolute value of V, which can vary according with the parameters R, P and ϵ.

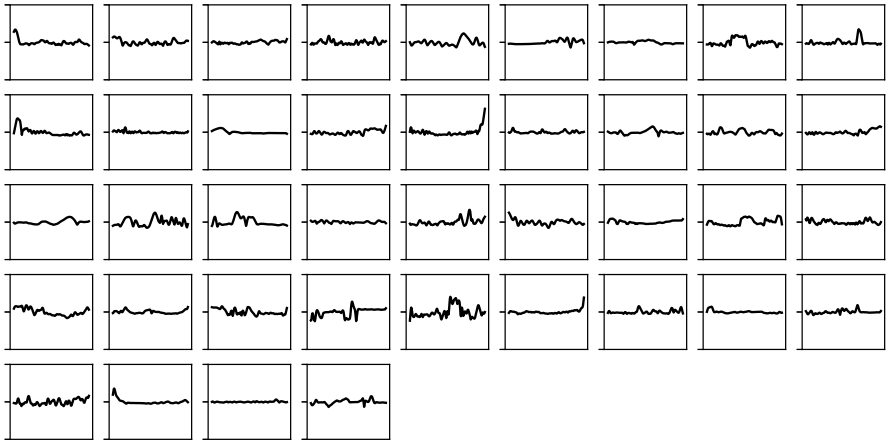
Finally, we computed three confidence values: i) a value for the Z-axis signal (c_z) used to detect the lifting of the mug, ii) a value for the matching of the XY magnitude template (c_{xy}) useful for the detection of the mug's rotation and iii) a combined value that takes into account the previous twos (c_{comb}). This latter value was empirically defined as:

$$c_{comb} = 0.7 \cdot c_z + 0.3 \cdot c_{xy}$$

The weights for c_z and c_{xy} were chosen experimentally based on the assumption that a drinking movement, intended as rotation of the mug, is performed only after the lifting the mug.



(a) Drinking movements



(b) Non-drinking movements

Figure 8.5.5: *Training set of movement using the magnitude of X and Y axis of the accelerometer. 8.5.5a displays the templates chosen as drinking movements. 8.5.5b shows those selected as no-drinking movements. All the plots show the templates downsampled to the fixed length of 170 samples (X-axis). The Y-axis represents the acceleration within a range of ± 2g.*

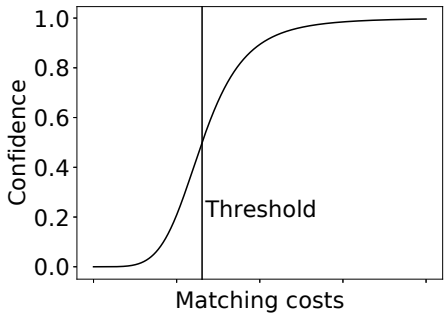


Figure 8.5.6: *4 Parameter Logistic Regression function used to assign a confidence with the respect to the threshold.*

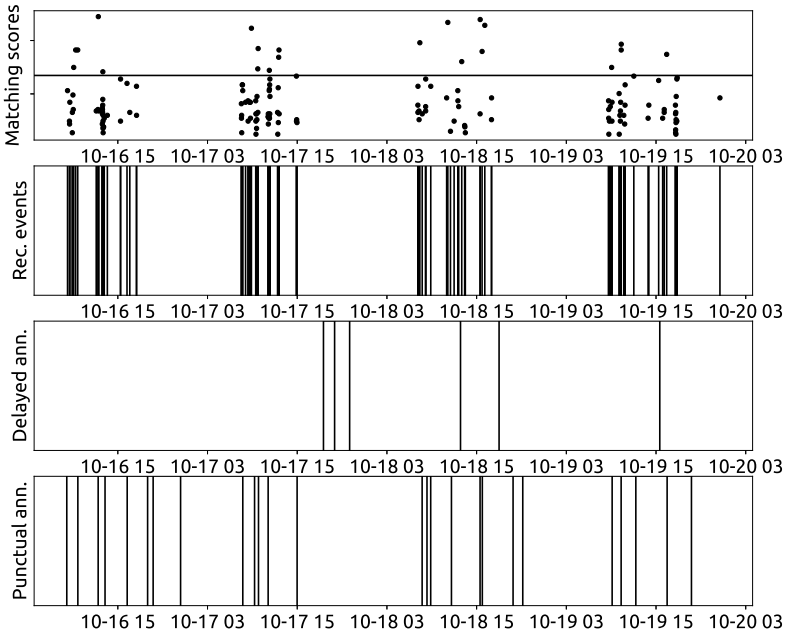
8.5.4 Evaluation

We used the two different subset of instances (Z-axis signal and XY magnitude) to train two separate instances of the system using the EA. As the EA is a stochastic process, we repeated the training 10 times, and we picked the best values for each of the two systems. The values are R=68, P=0, ϵ=28 and T=3364 for the Z-axis and R=23, P=11, ϵ=11 and T=726 for XY magnitude. With these parameters, we run the algorithm on the entire dataset, by using the templates displayed in Figure 8.5.2 and Figure 8.5.4, which were selected manually from the training sets as templates.

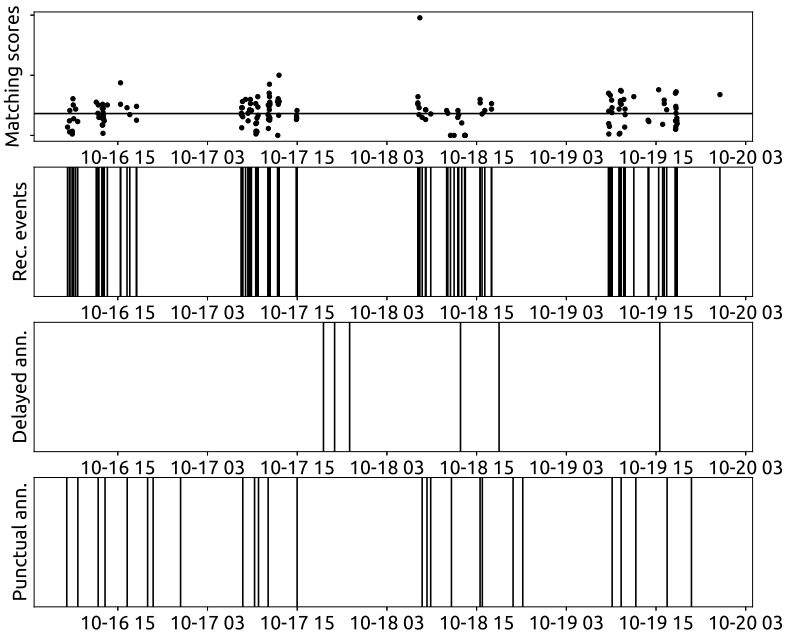
Figure 8.5.7 displays a comparison of the sensor events and the annotations for a single user, with the corresponding matching scores, for the Z-axis (8.5.7a) and XY Magnitude (8.5.7b). The matching of the template for XY magnitude is less restricting than the matching on the Z-axis. This is possibly due to the fact the rotation of the mug can happen even in case of movement different from just the drinking: e.g. while washing the mug, moving it around, etc.

The percentages of total detected movements for each scenario, compared to the total number of events are displayed in Table 8.5.1, for different confidence values. The low percentages are due to the nature of the sensors, which were collecting all sort of movements such as moving the mug on the desk, washing it or even accidental movements. It is important to note that the number of detected events is also lower than the number of user annotations (1808). This may be a result of the data collection protocol which did not specify to annotated only the drinking movements performed with the instrumented mug. Finally, for low confidence values ($\geq$25% and $\geq$50%) the matching of the Z-axis signal and the XY magnitude are quite different in term of percentage. For higher values of the confidence ($\geq$75%), the performance of the two systems tend to be the same, with similar percentages of detected events. Moreover, in order to better compare the performance of the two systems, we created the cross-correlogram presented in Figure 8.5.8. It displays the distribution of the differences between the confidence c_z and c_xy. While the two systems generally agree, as shown by the peak on 0, there is a negative skewness. This means that system for XY magnitude is less precise in detecting the drinking events than detecting the lifting of the mug using the Z-axis.

From the analysis, it is clear how the Z-axis signal is better for detecting a first subset of drinking movements rather then just the XY magnitude. The latter is useful to filter out events that refer to lifting of the cup but without an actual drinking performed by the user.



(a)



(b)

Figure 8.5.7: *Comparison of WLCSS matching scores with recorded data for a single user, for Z-axis (8.5.7a) and XY magnitude (8.5.7b). For each of the two template matching, from the top: the WLCSS matching scores and the threshold T, as horizontal line (first plot), the start time of the sensor events (second plot), and the user annotations delayed and punctual (respectively third and fourth plot). The data are for 4 days period, with the X-axis reporting date and time in the format "MM-DD HH"*

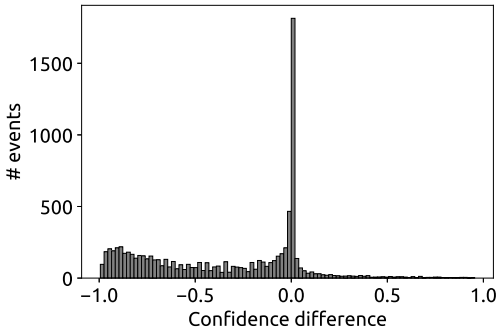


Figure 8.5.8: *Cross-correlogram representing the difference between the confidence obtained using the Z-axis signal and the XY magnitude.*

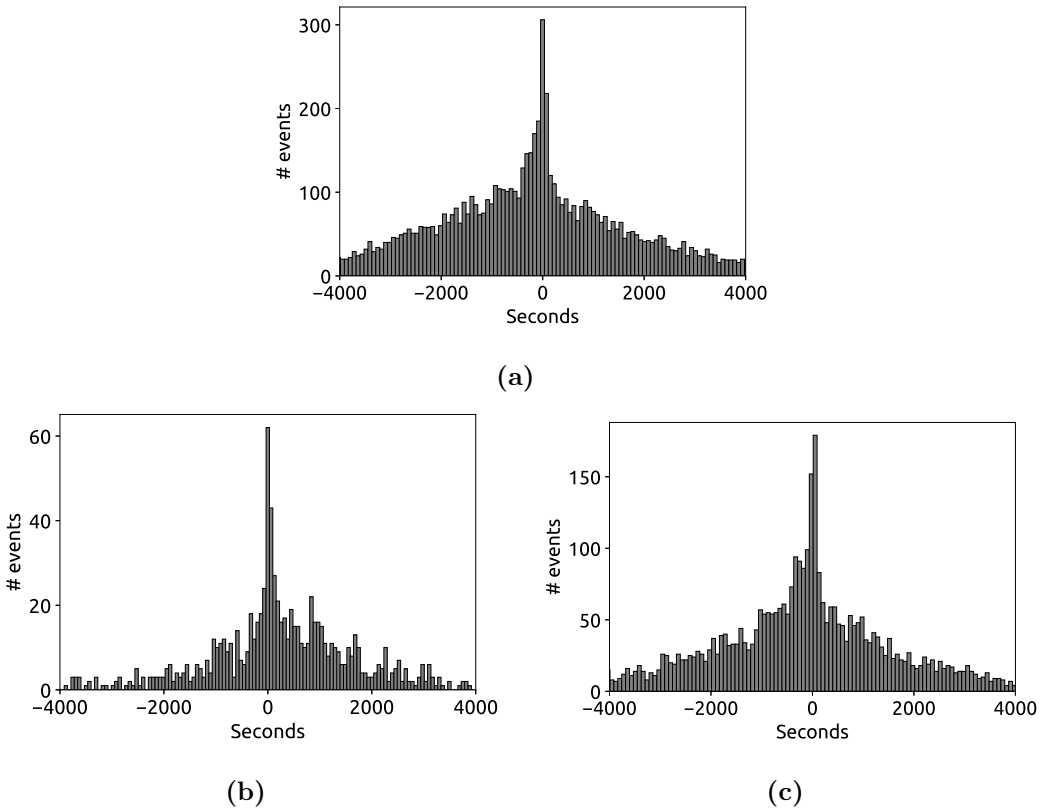


Figure 8.5.9: *Cross-correlogram representing the distribution of the delays (in seconds) between the user annotations and all the events recorded by the loggers (8.5.9a). Cross-correlogram representing the distribution of the delays (in seconds) between the user annotations and the events detected with a confidence ≥50%, using the signal from the Z-axis only (8.5.9b). Cross-correlogram representing the distribution of the delays (in seconds) between the user annotations and the events detected with a confidence ≥50%, using the XY magnitude (8.5.9c).*

Table 8.5.1: *Number of drinking movements detected for some confidence levels. The percentages are with respect to the total number of sensor events in the dataset (8825).*

Confidence	Z-Axis		XY-Magnitude		Combined	
	# movements	%	# movements	%	# movements	%
$\geq 25\%$	1481	17%	5253	60%	3930	44%
$\geq 50\%$	942	10%	4365	49%	1113	12%
$\geq 75\%$	543	6%	3453	39%	483	5%

We studied the relation between user annotations, sensor events and detected movements, for the two systems. To do this, we assigned to every recorded event the closest user annotation in time. Then, computing the time difference between the sensor events and the corresponding closest annotations, we created the cross-correlogram displayed in Figure 8.5.9a. Figure 8.5.9b and Figure 8.5.9b present the distribution of the same time differences, but considering only the movements detected by WLCSS with a confidence $\geq$50%, respectively for Z-axis and XY magnitude. The plot for Z-axis presents a more pointy distribution, confirming that, in this case, WLCSS detected events that were actually closer in time to the user annotations. The plot for XY magnitude confirms the lower precision of the detection with respect to the user annotation.

8.6 Unsupervised learning

We evaluated also an unsupervised approach in order to classify the movements in drinking/non-drinking as it does not require a training set. We developed a custom method based on K-Means. We modified K-Means in order to make it able to cluster movements performed with variation in their speed of execution.

8.6.1 K-Means with WLCSS

K-Means is a clustering technique that aims to partition n observation in k clusters. Each observation belongs to the cluster with the closest mean. It can be used for unsupervised learning by clustering the input data based on a distance measure. The algorithm is based on two steps, assignment step and update step, which are repeated until a stopping criteria is met [107]. This criteria can be reaching a maximum number of iterations, the change of the clusters in the update step in below a thresholds, etc. We implemented a modified version of the K-Means, where WLCSS was used a distance measure in place of

the Euclidean distance. The assignment in our implementation is modified as following:

$$\text{argmax}\, c_i WLCSS(x, c_i)$$

where x is a sensor events, c_i is the centroid for the i-th cluster. The function *argmin* is replaced by *argmax* as WLCSS compute a matching score rather than a distance. The update step is unmodified.

8.6.2 Evaluation

We compared our version of K-Means (named K-MeansWLCSS) against the standard version that uses the Euclidean distance in order to assign each instance to the closest cluster. We applied both the implementation on the training set from the previous step, with $k = 2$ as the goal was to distinguish between drinking and non-drinking movements. As all the instances were resampled to the same length, they could be used as feature vectors for both the implementations, without dealing with different lengths of the feature vectors (in this case the resampled raw signal). Applying the algorithms on the training set allowed us to compare the clustering results with the labels assigned manually to each movement, during the data selection. Figure 8.8.1 presents a visual comparison between the clusters obtained with the two versions of K-Means. For both the implementations, Cluster 1 seems to include mainly the drinking movements, while Cluster 2 the non-drinking movements. We used this consideration in order to evaluate the performance of the two algorithms computing precision, recall and F1 score, presented in Table 8.7.1. They are computed by comparing the clustering of K-Means with the manual labels of the instances in the extracted subset. K-Means_WLCSS increased the F1 score by 16%, being able to detect more variations in the drinking movements as it is also visible from Figure 8.8.1. It was able to cluster correctly drinking movements composed by two sips, such as the instances 8, 16, and 21 of Figure 8.8.1c.

8.7 Discussion

We discovered that the main issue for this dataset was the data collection protocol which was too relaxed. More precise instructions would increase considerably the quality of the data. Simultaneously, asking the users for more precision in following the protocol should be balanced with shorter sessions of data collection, as we noticed how the user commitment decreases over 4 days of continuous data collection. Lastly, as it has been demonstrated that experience sampling is not reliable, we recommend to increase the

Table 8.7.1: *Precision, Recall and F1 score for K-Means and K-MeansWLCSS computed on the training set. They are computed using the manual labels assigned to each instance in the training set as ground-truth. The majority of the movements in a cluster is used as classification label for the K-Means implementations.*

	K-M	K-M_WLCSS
Precision	100%	92.11%
Recall	62.16%	94.6%
F1	77%	93%

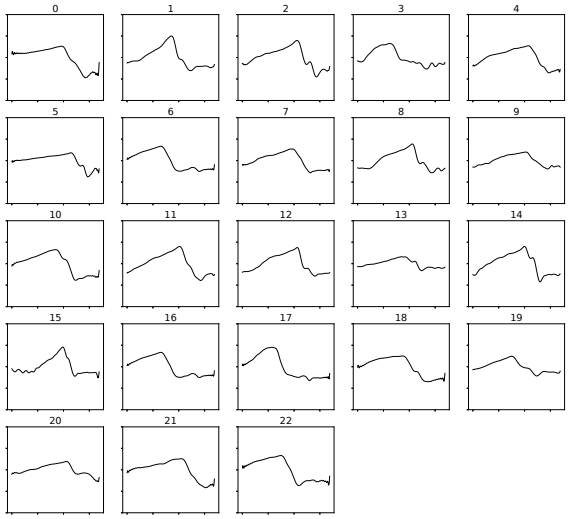
effort in the setup of the experiment by including a video recording. It would dramatically increase the quality and re-usability of the dataset, although it would require additional time for the labelling of the data. In order to reduce this effort, the video recordings can be used to precisely annotate just a small portion of the entire dataset. This well-annotated subset, which can be also collected a-posteriori, or can be used as training set instead of extracting one through heuristic. The well-annotated dataset would also allow to evaluate more precisely the performance of the TMM or any other classifier on the complete dataset, through statistical analysis.

In this study we extracted a subset of events which can be considered as drinking movements with a certain confidence. This extracted subset can be potentially used to re-train the TMM for a more reliable movement recognition system. The re-training phase can be performed using movements with different levels of confidence: an higher level of confidence would increase the specificity of the found movements. Decreasing this value would increase the sensitivity, potentially including more variations of the drinking movements.

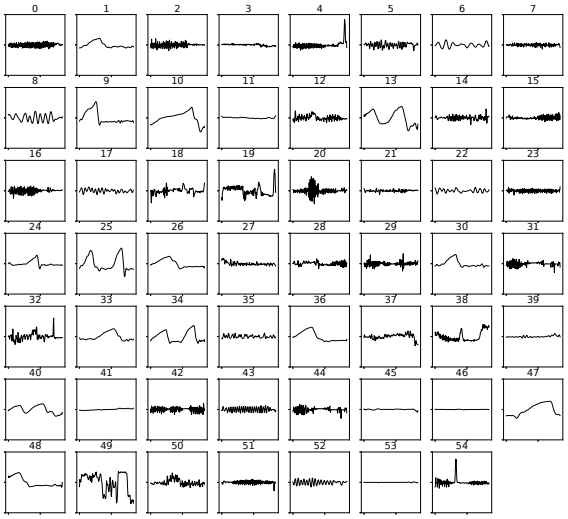
Finally, we aimed to evaluate how an unsupervised learning technique can be used in order to extract drinking movements from a poorly labelled dataset. We implemented a modified version of K-Means which uses WLCSS as distance measure for the assignment step. The results are promising: with 2 clusters it managed to differentiate between drinking and non-drinking movement with an 93% F1-score, although on a limited number of sensor events. A more extensive evaluation can be performed on the entire dataset, although without a reliable ground-truth, a validation of the results, in this case, could be difficult.

8.8 Conclusion

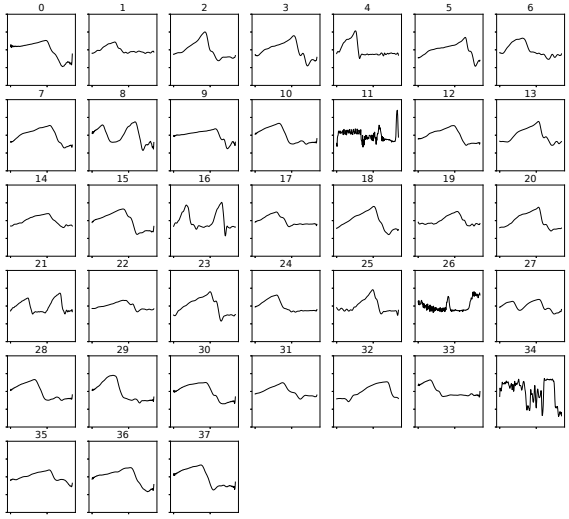
In this study, we investigate how to extract knowledge from a poorly labelled dataset of drinking movements. We analysed the user annotations in order to get qualitative information on how to improve the data collection. We exposed how the loose protocol created most of the problems and we highlighted the need of providing more precise instructions to the users. Then, by selecting instances manually and using a template matching algorithm, we demonstrated that it is possible to extract a subset of instances which are actually drinking movements within a certain level of confidence. We proved that an unsupervised approach based on K-Means and WLCSS can improve the clustering of movement over the standard K-Means implementation. Our method outperformed the baseline method by including a wider variety of drinking movements and increasing F1-score by 16%.



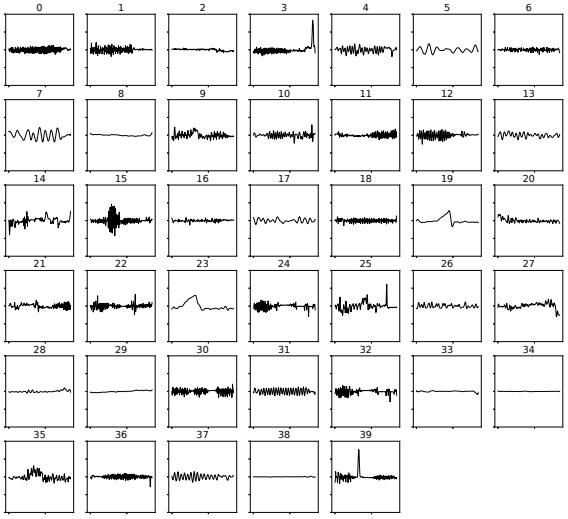
(a) Cluster 1 - K-Means



(b) Cluster 2 - K-Means



(c) Cluster 1 - K-MeansWLCSS



(d) Cluster 2 - K-MeansWLCSS

Figure 8.8.1: *Comparison between clusters obtained with K-Means using Euclidean distance (top left, top right), and using WLCSS (bottom left, bottom right).*

Chapter 9

Conclusion

This chapter concludes the thesis with a summary of the achievements of this thesis with regards of movement recognition using TMMs. We also discuss the limitations of our work and the future development to overcome these limitations. Finally, we conclude underlining the relevance of our contributions.

9.1 Achievements

In this thesis, we individuated some of the challenges of applying Template Matching Methods (TMMs) to movements recognition. In addressing these challenges solving the aims we presented in Section 1.2, we made the following contributions, here divided according with the thesis structure presented in Section 1.4.

With regards of the data availability:

- We collected and presented the first publicly available dataset of beach volleyball serves movements and games. The dataset comprises data from 10 users divided over 3 sessions, recorded using 4 wearable inertial platforms per user. In total, the dataset counts 585 annotated serves, divided in 4 categories: 147 long float, 140 long top spin, 155 short float and 143 short top spin serves. In addition to the annotated serves that span on over 2 hours, the dataset includes 9 hours of free unlabelled games rallies. It is freely available to the community at the link https://ieee-dataport.org/open-access/wearlab-beach-volleyball-serves-and-games.

- We developed a 3D human model that can be animated using the 3D orientation data provided by any wearable sensing platform for troubleshooting movement data issues and for movements analysis. The 3D model in made of 12 body segments that can be independently animated using each one a sensors worn by the user in

the corresponding position. We showed that our model can be animated both in real-time and offline. In this latter case, we demonstrated that it can be successfully deployed for privacy preserving annotation in three different scenarios:

- Event-detection: the animated model can be used to detect when an activity of interest is happening in the data stream;
- Open-ended annotation: the animated model can represent activities and movements with enough accuracy to let the annotator to open-ended label the data with the correct activity performed by the model;
- Closed-set annotation: when the 3D model is used for annotating activities from a closed set, it can reach up to 56% accuracy with only 6 annotators.

We presented a power-aware evolutionary based training method for WLCSS, named WLCSSLearn, in two versions. More specifically:

- WLCSSLearn_p is our method to train the parameters of WLCSS. We demonstrate that it can maximize the recognition performance on several dataset of activities including different sensor modalities and movements encoding, with minimal input by the user. Through an extensive evaluation, we set guidelines for the training parameters as well as for the genetic encoding of the parameters of WLCSS within WLCSSLearn_p. Our training method allowed recognition performance of 88%$\pm$0.08 F1 score, on average across 6 different datasets.

- WLCSSLearn_t is our method to generate variable length templates for WLCSS. The generated templates can be configured to maximise recognition performance or to reduce computational costs of the template matching. We showed that our method can improve by up to 15% the recognition performance compared to previous methods of selecting the most representative template (MRT) in the training set, while also saving between 10% and 20% of computations for some dataset. Overall, we demonstrated that WLCSSLearn_t shows comparable recognition performance to MRT while saving up to 50% of computational cost. Finally, WLCSSLearn_t can reduce the computations up to 90% while still increasing by 33% the recognition performance over chance-level recognition.

We demonstrated that WLCSSLearn is robust to difference in movements, sensor modalities and application scenarios.

Finally, we evaluate a case study of recovering a highly valuable but poorly annotated dataset of drinking actions. In this case, our 3D model could not be deployed to the lack of orientation data. However, we demonstrated that:

- WLCSS and our training method can be used to annotate drinking/non-drinking actions with different levels of confidence;

- using WLCSS as distance measure for unsupervised K-Mean clustering of actions segments can improve by 26% the F1 score compared to using the standard Euclidean distance.

9.2 Limitations and future work

Throughout this thesis, we dealt with isolated recognition of activities and movements. However, continuous recognition of actions is a more realistic scenario where the sensors produce a continuous stream of data that must be classified in order to provided feedback on the user movement. In this case, a continuous matching scores is produced by the TMM which creates further challenges to adapt our work to continuous recognition. We identified these challenges as:

- taking into account the NULL-class. When sensors produce a continuous stream of data, a portion of these data refers to movements that are not of interest for the application. For example, in the beach volleyball serving application, the NULL class can include all those movements between serves such as, for example, picking up the ball from the ground, removing the sand from it, etc. In this case, because it is impossible to forecast these movements, NULL class cannot be identified with one or more templates in order to be classified as such. Therefore it must be just discarded by increasing the specificity of the matching through better templates and especially the threshold V. However, it is not possible to predict the length of the NULL-class data and consequently it is not possible to predict how low the value of the continuous matching score computed by the TMM due to a long mismatch. This makes difficult to adapt the value of V at training time in order to exclude the NULL-class by using only the isolated segments as done in this work. As a possible solution, we suggest to use a windows based approach to continuous recognition where the continuous stream of data is broken into fixed time windows that would then be classified singularly and that can be used for training as independent segments. In

this case, part of the segments included for the training with WLCSSLearn will include data of the NULL-class.

Another possibility is to set a maximum lower bound to the matching score that would cap the score in case of long mismatch. This would prevent a long mismatch due to the NULL-class and the consequent low value in matching score to affect the detection of a match when needed.

Both these methods would requires further analysis in order to set application specific window length or lower bound for the matching score.

- distinguishing between long and short variation of similar but different actions. Considering the beach volleyball application, the long version of a specific serve can be mistaken for a warped variation of a shorter one. Since in continuous recognition there is not specified start and end of a movement, this can become an issue where shorter movements cause several false positives when a longer actions is being detected. In this case, a possible solution is to deploy templates for the shorter actions longer than the minimal length in order to include additional information about the end of the short movements. This would increase the differences with respect to the templates of the longer actions.

As described in this work, WLCSSLearn is a two step process: WLCSSLearn_p trains the parameters of WLCSS and WLCSSLearn_t generates the templates for the matching. However, both steps would be required at each time. Therefore, we identify this as one of the limitation of this work and the main future improvement. A single step training that would both train the parameters and generate the templates simultaneously would be beneficial, reducing some of the overhead of the two step process. For example, in the current implementation the threshold V is computed twice: i) first it is calculated with the trained parameters as its value depends on R and P, ii) then it is computed again for every generated template as the value of V also depends on the length of the generated T and the value of its sample. A unique comprehensive WLCSSLearn for simultaneous parameters training and template generation would require a new encoding of individuals chromosomes, new genetic operators for this new encoding and a new fitness function that could drive the evolution.

9.3 Relevance and conclusion

In this thesis, we looked at the problem of movement recognition with Template Matching Methods. We selected TMMs as recognition algorithm to automatically classify actions from wearable sensors data as they offer a granular distance between two sensor signals and therefore two movements. The granular distance can be used in order to finely assess the quality of the movement. We identified the main challenge in the lack of a training algorithm for TMMs that would maximise the recognition performance while also allowing a power-aware training. In the process of addressing this challenge:

- we presented a new dataset of beach volleyball serve actions and games in order to evaluate our methods in an application scenario that would benefit from skill assessment. This is the first publicly available dataset of beach volleyball movements collected with these kind of sensors.
- We also produced a 3D human model for troubleshooting potential problem with sensors data in a dataset which has been demonstrated to be useful for privacy preserving annotations and a case study of recovering poorly annotated data that can be used for template matching for skill assessment.
- Finally, we addressed the lack of training method for TMM by presenting the first training method for Warping Longest Common Subsequence, named WLCSSLearn, that:
  - maximize per-application recognition performance with minimal input from the user
  - unprecedentedly, allows a configurable trade-off between recognition performance and computational cost reduction.
  - can be exported to other TMMs with little adjustments: with the right encoding, WLCSSLearn_p can be used to train the parameters of Longest Common Subsequence (LCSS), i.e. insertion, deletion and substitution costs, and the weight of Weighted Dynamic Time Warping (W-DTW). By changing the TMM used to compute the match in WLCSSLearn_t, our training algorithm can generate templates for any template matching method, i.e. DTW, LCSS, etc.

We argue that, thanks to our methods, movement recognition using TMMs can be effectively improved.

Bibliography

[1] Jeffrey W. Lockhart, Tony Pulickal and Gary M. Weiss. 'Applications of mobile activity recognition'. In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*. ACM Press, 2012. DOI: 10.1145/2370216.2370441. URL: https://doi.org/10.1145/2370216.2370441.

[2] Andreas Bulling, Ulf Blanke and Bernt Schiele. 'A tutorial on human activity recognition using body-worn inertial sensors'. In: *ACM Computing Surveys* 46.3 (2014), pp. 1–33. DOI: 10.1145/2499621. URL: https://doi.org/10.1145/2499621.

[3] Lorenzo Porzi et al. 'A smart watch-based gesture recognition system for assisting people with visual impairments'. In: *Proceedings of the 3rd ACM international workshop on Interactive multimedia on mobile & portable devices - IMMPD '13*. ACM Press, 2013. DOI: 10.1145/2505483.2505487. URL: https://doi.org/10.1145/2505483.2505487.

[4] G. Saggio et al. 'Gesture recognition and classification for surgical skill assessment'. In: *2011 IEEE International Symposium on Medical Measurements and Applications*. IEEE, 2011. DOI: 10.1109/memea.2011.5966681. URL: https://doi.org/10.1109/memea.2011.5966681.

[5] Fabien Despinoy et al. 'Unsupervised Trajectory Segmentation for Surgical Gesture Recognition in Robotic Training'. In: *IEEE Transactions on Biomedical Engineering* 63.6 (2016), pp. 1280–1291. DOI: 10.1109/tbme.2015.2493100. URL: https://doi.org/10.1109/tbme.2015.2493100.

[6] Chongyang Wang et al. 'Leveraging Activity Recognition to Enable Protective Behavior Detection in Continuous Data'. In: *arXiv preprint arXiv:2011.01776* (2020).

[7] Jinxian Qi et al. 'Intelligent Human-Computer Interaction Based on Surface EMG Gesture Recognition'. In: *IEEE Access* 7 (2019), pp. 61378–61387. DOI: 10.1109/

access . 2019 . 2914728. URL: https : / / doi . org / 10 . 1109 / access . 2019 . 2914728.

[8] Meenakshi Panwar and Pawan Singh Mehra. 'Hand gesture recognition for human computer interaction'. In: *2011 International Conference on Image Information Processing*. IEEE, 2011. DOI: 10 . 1109 / iciip . 2011 . 6108940. URL: https : //doi.org/10.1109/iciip.2011.6108940.

[9] Cassim Ladha et al. 'ClimbAX'. In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013. DOI: 10 . 1145/2493432.2493492. URL: https://doi.org/10.1145/2493432.2493492.

[10] Alberto Cannavò et al. 'A Movement Analysis System based on Immersive Virtual Reality and Wearable Technology for Sport Training'. In: *Proceedings of the 4th International Conference on Virtual Reality - ICVR 2018*. ACM Press, 2018. DOI: 10.1145/3198910.3198917. URL: https://doi.org/10.1145/3198910.3198917.

[11] Jenny Cifuentes et al. 'Medical gesture recognition using dynamic arc length warping'. In: *Biomedical Signal Processing and Control* 52 (2019), pp. 162–170. DOI: 10.1016/j.bspc.2019.04.022. URL: https://doi.org/10.1016/j.bspc.2019.04.022.

[12] Yixin Gao et al. 'Query-by-example surgical activity detection'. In: *International Journal of Computer Assisted Radiology and Surgery* 11.6 (2016), pp. 987–996. DOI: 10.1007/s11548-016-1386-3. URL: https://doi.org/10.1007/s11548-016-1386-3.

[13] Aftab Khan et al. 'Beyond activity recognition: skill assessment from accelerometer data'. In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15*. ACM Press, 2015. DOI: 10 . 1145/2750858.2807534. URL: https://doi.org/10.1145/2750858.2807534.

[14] Anand Asokan, Allan Joseph Pothen and Raj Krishnan Vijayaraj. 'ARMatron — A wearable gesture recognition glove: For control of robotic devices in disaster management and human Rehabilitation'. In: *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*. IEEE, 2016. DOI: 10 . 1109/raha . 2016 . 7931882. URL: https : //doi . org/10 . 1109/raha . 2016.7931882.

[15] Chee Seng Chan, Honghai Liu and David J. Brown. 'Recognition of Human Motion From Qualitative Normalised Templates'. In: *Journal of Intelligent and Robotic Systems* 48.1 (2006), pp. 79–95. DOI: 10.1007/s10846-006-9100-2. URL: https://doi.org/10.1007/s10846-006-9100-2.

[16] Cristina Soaz and Klaus Diepold. 'Step Detection and Parameterization for Gait Assessment Using a Single Waist-Worn Accelerometer'. In: *IEEE Transactions on Biomedical Engineering* 63.5 (2016), pp. 933–942. DOI: 10.1109/tbme.2015.2480296. URL: https://doi.org/10.1109/tbme.2015.2480296.

[17] Jenny Margarito et al. 'User-Independent Recognition of Sports Activities from a Single Wrist-worn Accelerometer: A Template Matching Based Approach'. In: *IEEE Transactions on Biomedical Engineering* (2015), pp. 1–1. DOI: 10.1109/tbme.2015.2471094. URL: https://doi.org/10.1109/tbme.2015.2471094.

[18] Donald J Berndt and James Clifford. 'Using dynamic time warping to find patterns in time series.' In: *KDD workshop*. Vol. 10. 16. Seattle, WA, USA: 1994, pp. 359–370.

[19] Abdel Salam Malek et al. 'Automated detection of premature ventricular contraction in ECG signals using enhanced template matching algorithm'. In: *Biomedical Physics & Engineering Express* 6.1 (2020), p. 015024. DOI: 10.1088/2057-1976/ab6995. URL: https://doi.org/10.1088/2057-1976/ab6995.

[20] Haemwaan Sivaraks and Chotirat Ann Ratanamahatana. 'Robust and Accurate Anomaly Detection in ECG Artifacts Using Time Series Motif Discovery'. In: *Computational and Mathematical Methods in Medicine* 2015 (2015), pp. 1–20. DOI: 10.1155/2015/453214. URL: https://doi.org/10.1155/2015/453214.

[21] A. Beatrice Dorothy, S. Britto Ramesh Kumar and J. Jerlin Sharmila. 'IoT Based Home Security through Digital Image Processing Algorithms'. In: *2017 World Congress on Computing and Communication Technologies (WCCCT)*. IEEE, 2017. DOI: 10.1109/wccct.2016.15. URL: https://doi.org/10.1109/wccct.2016.15.

[22] Miguel Saez et al. 'Anomaly detection and productivity analysis for cyber-physical systems in manufacturing'. In: *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*. IEEE, 2017. DOI: 10.1109/coase.2017.8256070. URL: https://doi.org/10.1109/coase.2017.8256070.

[23] Haipeng Wang et al. 'Using parallel tokenizers with DTW matrix combination for low-resource spoken term detection'. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013. DOI: 10.1109/icassp.2013.6639333. URL: https://doi.org/10.1109/icassp.2013.6639333.

[24] Charles Ramey et al. 'Wear-a-CUDA: a GPU based dolphin whistle recognizer for underwater wearable computers'. In: *Proceedings of the 2018 ACM International Symposium on Wearable Computers - ISWC '18*. ACM Press, 2018. DOI: 10.1145/3267242.3267275. URL: https://doi.org/10.1145/3267242.3267275.

[25] Thomas Stiefmeier, Daniel Roggen and Gerhard Tröster. 'Gestures are strings: efficient online gesture spotting and classification using string matching'. In: *Proceedings of the Second International Conference on Body Area Networks BodyNets*. ICST, 2007. DOI: 10.4108/bodynets.2007.143. URL: https://doi.org/10.4108/bodynets.2007.143.

[26] Long-Van Nguyen-Dinh et al. 'Improving online gesture recognition with template matching methods in accelerometer data'. In: *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*. IEEE, 2012. DOI: 10.1109/isda.2012.6416645. URL: https://doi.org/10.1109/isda.2012.6416645.

[27] Francisco Ordóñez and Daniel Roggen. 'Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition'. In: *Sensors* 16.1 (2016), p. 115. DOI: 10.3390/s16010115. URL: https://doi.org/10.3390/s16010115.

[28] Daniele Ravi et al. 'A Deep Learning Approach to on-Node Sensor Data Analytics for Mobile or Wearable Devices'. In: *IEEE Journal of Biomedical and Health Informatics* 21.1 (2017), pp. 56–64. DOI: 10.1109/jbhi.2016.2633287. URL: https://doi.org/10.1109/jbhi.2016.2633287.

[29] Henry Friday Nweke et al. 'Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges'. In: *Expert Systems with Applications* 105 (2018), pp. 233–261. DOI: 10.1016/j.eswa.2018.03.056. URL: https://doi.org/10.1016/j.eswa.2018.03.056.

[30] Elisa Spano, Stefano Di Pascoli and Giuseppe Iannaccone. 'Low-Power Wearable ECG Monitoring System for Multiple-Patient Remote Monitoring'. In: *IEEE*

Sensors Journal 16.13 (2016), pp. 5452–5462. DOI: 10.1109/jsen.2016.2564995. URL: https://doi.org/10.1109/jsen.2016.2564995.

[31] James Dieffenderfer et al. 'Low-Power Wearable Systems for Continuous Monitoring of Environment and Health for Chronic Respiratory Disease'. In: *IEEE Journal of Biomedical and Health Informatics* 20.5 (2016), pp. 1251–1264. DOI: 10.1109/jbhi.2016.2573286. URL: https://doi.org/10.1109/jbhi.2016.2573286.

[32] Simone Benatti et al. 'A Versatile Embedded Platform for EMG Acquisition and Gesture Recognition'. In: *IEEE Transactions on Biomedical Circuits and Systems* 9.5 (2015), pp. 620–630. DOI: 10.1109/tbcas.2015.2476555. URL: https://doi.org/10.1109/tbcas.2015.2476555.

[33] Jongpal Kim and Hyoungho Ko. 'Reconfigurable Multiparameter Biosignal Acquisition SoC for Low Power Wearable Platform'. In: *Sensors* 16.12 (2016), p. 2002. DOI: 10.3390/s16122002. URL: https://doi.org/10.3390/s16122002.

[34] Mahtab J. Fard, Sattar Ameri and R. Darin Ellis. 'Toward personalized training and skill assessment in robotic minimally invasive surgery'. In: *Lecture Notes in Engineering and Computer Science* 2226.October (2016), pp. 719–724. ISSN: 20780958. arXiv: 1610.07245.

[35] Daniel Roggen et al. 'Limited-memory warping LCSS for real-time low-power pattern recognition in wireless nodes'. In: *European conference on wireless sensor networks*. Springer. 2015, pp. 151–167.

[36] M. Michalopoulou et al. 'Computer analysis of the technical and tactical effectiveness in Greek Beach Volleyball'. In: *International Journal of Performance Analysis in Sport* 5.1 (June 2005), pp. 41–50. DOI: 10.1080/24748668.2005.11868314. URL: https://doi.org/10.1080/24748668.2005.11868314.

[37] A.B. Lopez-Martinez and J.M. Palao. 'Effect of serve execution on serve efficacy in men's and women's beach volleyball'. In: *International Journal of Applied Sports Sciences* 21.1 (2009), pp. 1–16. ISSN: 15982939.

[38] Markus Tilp et al. 'Digital game analysis in beach volleyball.' In: *International Journal of Performance Analysis in Sport* 6.1 (June 2006), pp. 140–148. DOI: 10.1080/24748668.2006.11868362. URL: https://doi.org/10.1080/24748668.2006.11868362.

[39] Mikko Häyrinen and Kostas Tampouratzis. *Technical and tactical game analysis of elite female beach volleyball*. 37. 2012, pp. 1–45. ISBN: 9789525676594.

[40] José Manuel Jiménez-Olmedo et al. 'Serve analysis of professional players in beach volleyball'. In: *Journal of Human Sport and Exercise* 7.3 (2012), pp. 706–713. DOI: 10.4100/jhse.2012.73.10. URL: https://doi.org/10.4100/jhse.2012.73.10.

[41] Roberto Lobietti. 'A review of blocking in volleyball: from the notational analysis to biomechanics'. In: *Journal of Human Sport and Exercise* 4.2 (2009), pp. 93–99. DOI: 10.4100/jhse.2009.42.03. URL: https://doi.org/10.4100/jhse.2009.42.03.

[42] Jonathan C. Reeser et al. 'Upper Limb Biomechanics During the Volleyball Serve and Spike'. In: *Sports Health: A Multidisciplinary Approach* 2.5 (Sept. 2010), pp. 368–374. DOI: 10.1177/1941738110374624. URL: https://doi.org/10.1177/1941738110374624.

[43] Gabriel Gomez et al. 'Tracking of Ball and Players in Beach Volleyball Videos'. In: *PLoS ONE* 9.11 (Nov. 2014). Ed. by Ke Lu, e111730. DOI: 10.1371/journal.pone.0111730. URL: https://doi.org/10.1371/journal.pone.0111730.

[44] Thomas Mauthner et al. 'Visual Tracking of Athletes in Beach Volleyball Using a Single Camera'. In: *International Journal of Computer Science in Sport* 6.2 (2008), pp. 21–34.

[45] L. Ponce Cuspinera et al. 'Beach volleyball serve type recognition'. In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. ACM, Sept. 2016. DOI: 10.1145/2971763.2971781. URL: https://doi.org/10.1145/2971763.2971781.

[46] Thomas Kautz et al. 'Activity recognition in beach volleyball using a Deep Convolutional Neural Network'. In: *Data Mining and Knowledge Discovery* 31.6 (Feb. 2017), pp. 1678–1705. DOI: 10.1007/s10618-017-0495-0. URL: https://doi.org/10.1007/s10618-017-0495-0.

[47] S. O. H. Madgwick, A. J. L. Harrison and R. Vaidyanathan. 'Estimation of IMU and MARG orientation using a gradient descent algorithm'. In: *2011 IEEE International Conference on Rehabilitation Robotics*. IEEE, 2011. DOI: 10.1109/icorr.2011.5975346. URL: https://doi.org/10.1109/icorr.2011.5975346.

[48] Daniel Roggen et al. 'Collecting complex activity datasets in highly rich networked sensor environments'. In: *2010 Seventh International Conference on Networked Sensing Systems (INSS)*. IEEE, 2010. DOI: 10.1109/inss.2010.5573462. URL: https://doi.org/10.1109/inss.2010.5573462.

[49] Amazon. *Amazon Mechanical Turk*. https://www.mturk.com/mturk/welcome, accessed 16/06/2016. 2005.

[50] Long-Van Nguyen-Dinh et al. 'Tagging human activities in video by crowdsourcing'. In: *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval - ICMR '13*. ACM Press, 2013. DOI: 10.1145/2461466.2461508. URL: https://doi.org/10.1145/2461466.2461508.

[51] Aobo Wang, Cong Duy Vu Hoang and Min-Yen Kan. 'Perspectives on crowdsourcing annotations for natural language processing'. In: *Language Resources and Evaluation* 47.1 (Mar. 2012), pp. 9–31. DOI: 10.1007/s10579-012-9176-1. URL: https://doi.org/10.1007/s10579-012-9176-1.

[52] Gabriel Parent and Maxine Eskenazi. 'Speaking to the Crowd: Looking at past achievements in using crowdsourcing for speech and predicting future challenges'. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2011, pp. 3037–3040.

[53] Carl Vondrick, Donald Patterson and Deva Ramanan. 'Efficiently Scaling up Crowdsourced Video Annotation'. In: *International Journal of Computer Vision* 101.1 (Sept. 2012), pp. 184–204. DOI: 10.1007/s11263-012-0564-1. URL: https://doi.org/10.1007/s11263-012-0564-1.

[54] Michael Boyle, Christopher Edwards and Saul Greenberg. 'The effects of filtered video on awareness and privacy'. In: *Proceedings of the 2000 ACM conference on Computer supported cooperative work - CSCW '00*. ACM Press, 2000. DOI: 10.1145/358916.358935. URL: https://doi.org/10.1145/358916.358935.

[55] Ji Dai et al. 'Towards privacy-preserving activity recognition using extremely low temporal and spatial resolution cameras'. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2015. DOI: 10.1109/cvprw.2015.7301356. URL: https://doi.org/10.1109/cvprw.2015.7301356.

[56] Susumu Harada et al. 'VoiceLabel'. In: *Proceedings of the 10th international conference on Multimodal interfaces - IMCI '08*. ACM Press, 2008. DOI: 10.1145/1452392.1452407. URL: https://doi.org/10.1145/1452392.1452407.

[57] Tim van Kasteren et al. 'Accurate activity recognition in a home setting'. In: *Proceedings of the 10th international conference on Ubiquitous computing - UbiComp '08*. ACM Press, 2008. DOI: 10.1145/1409635.1409637. URL: https://doi.org/10.1145/1409635.1409637.

[58] Unity. *Unity Asset Store*. Retrieved on November 2nd, 2021 from https://assetstore.unity.com/. 2021. URL: {https://assetstore.unity.com/}.

[59] Unreal. *Unreal Engine Marketplace*. Retrieved on November 2nd, 2021 from https://www.unrealengine.com/marketplace/en-US/. 2021. URL: {https://www.unrealengine.com/marketplace/en-US/}.

[60] Makehuman. *Makehuman*. Retrieved on November 2nd, 2021 from http://www.makehumancommunity.org/. 2021. URL: http://www.makehumancommunity.org/.

[61] The jME core team. *jMonkeyEngine*. http://jmonkeyengine.org/. 2016.

[62] XSens. *MTx 3D Tracker*. 2000. URL: https://www.xsens.com/products/mtx/ (visited on 16/06/2016).

[63] Taras K Vintsyuk. 'Speech discrimination by dynamic programming'. In: *Cybernetics* 4.1 (1968), pp. 52–57.

[64] T.M. Rath and R. Manmatha. 'Word image matching using dynamic time warping'. In: IEEE Comput. Soc. DOI: 10.1109/cvpr.2003.1211511. URL: https://doi.org/10.1109/cvpr.2003.1211511.

[65] Eamonn Keogh and Chotirat Ann Ratanamahatana. 'Exact indexing of dynamic time warping'. In: *Knowledge and Information Systems* 7.3 (2005), pp. 358–386. DOI: 10.1007/s10115-004-0154-9. URL: https://doi.org/10.1007/s10115-004-0154-9.

[66] Young-Seon Jeong, Myong K. Jeong and Olufemi A. Omitaomu. 'Weighted dynamic time warping for time series classification'. In: *Pattern Recognition* 44.9 (2011), pp. 2231–2240. DOI: 10.1016/j.patcog.2010.09.022. URL: https://doi.org/10.1016/j.patcog.2010.09.022.

[67] Eamonn J. Keogh and Michael J. Pazzani. 'Derivative Dynamic Time Warping'. In: *Proceedings of the 2001 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2001. DOI: 10.1137/1.9781611972719.1. URL: https://doi.org/10.1137/1.9781611972719.1.

[68] Dan Gusfield. 'Algorithms on Stings, Trees, and Sequences'. In: *ACM SIGACT News* 28.4 (1997), pp. 41–60. DOI: 10.1145/270563.571472. URL: https://doi.org/10.1145/270563.571472.

[69] David Maier. 'The Complexity of Some Problems on Subsequences and Supersequences'. In: *Journal of the ACM* 25.2 (1978), pp. 322–336. DOI: 10.1145/322063.322075. URL: https://doi.org/10.1145/322063.322075.

[70] Anthony Bagnall et al. 'The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances'. In: *Data Mining and Knowledge Discovery* 31.3 (2016), pp. 606–660. DOI: 10.1007/s10618-016-0483-9. URL: https://doi.org/10.1007/s10618-016-0483-9.

[71] Daniel Roggen et al. 'Limited-Memory Warping LCSS for Real-Time Low-Power Pattern Recognition in Wireless Nodes'. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2015, pp. 151–167. DOI: 10.1007/978-3-319-15582-1_10. URL: https://doi.org/10.1007/978-3-319-15582-1_10.

[72] Daniel Roggen et al. 'The adARC pattern analysis architecture for adaptive human activity recognition systems'. In: *Journal of Ambient Intelligence and Humanized Computing* 4.2 (2011), pp. 169–186. DOI: 10.1007/s12652-011-0064-0. URL: https://doi.org/10.1007/s12652-011-0064-0.

[73] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams. 'Learning representations by back-propagating errors'. In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[74] Simon Ruffieux et al. 'A Survey of Datasets for Human Gesture Recognition'. In: *Human-Computer Interaction. Advanced Interaction Modalities and Techniques*. Springer International Publishing, 2014, pp. 337–348. DOI: 10.1007/978-3-319-07230-2_33. URL: https://doi.org/10.1007/978-3-319-07230-2_33.

[75] Thomas Stiefmeier, Daniel Roggen and Gerhard Troster. 'Fusion of String-Matched Templates for Continuous Activity Recognition'. In: *2007 11th IEEE International Symposium on Wearable Computers*. IEEE, 2007. DOI: 10.1109/iswc.2007.4373775. URL: https://doi.org/10.1109/iswc.2007.4373775.

[76] Kilian Forster, Daniel Roggen and Gerhard Troster. 'Unsupervised Classifier Self-Calibration through Repeated Context Occurences: Is there Robustness against Sensor Displacement to Gain?' In: *2009 International Symposium on Wearable Computers*. IEEE, 2009. DOI: 10.1109/iswc.2009.12. URL: https://doi.org/10.1109/iswc.2009.12.

[77] Michael Hardegger et al. 'Enhancing action recognition through simultaneous semantic mapping from body-worn motion sensors'. In: *Proceedings of the 2014 ACM International Symposium on Wearable Computers - ISWC '14*. ACM Press, 2014. DOI: 10.1145/2634317.2634323. URL: https://doi.org/10.1145/2634317.2634323.

[78] Alex Ming Hui Wong and Dae-Ki Kang. 'Stationary Hand Gesture Authentication Using Edit Distance on Finger Pointing Direction Interval'. In: *Scientific Programming* 2016 (2016), pp. 1–15. DOI: 10.1155/2016/7427980. URL: https://doi.org/10.1155/2016/7427980.

[79] Thomas Stiefmeier et al. 'Wearable Activity Tracking in Car Manufacturing'. In: *IEEE Pervasive Computing* 7.2 (2008), pp. 42–50. DOI: 10.1109/mprv.2008.40. URL: https://doi.org/10.1109/mprv.2008.40.

[80] Tarik Arici et al. 'Robust gesture recognition using feature pre-processing and weighted dynamic time warping'. In: *Multimedia Tools and Applications* 72.3 (2013), pp. 3045–3062. DOI: 10.1007/s11042-013-1591-9. URL: https://doi.org/10.1007/s11042-013-1591-9.

[81] Sait Celebi et al. 'Gesture recognition using skeleton data with weighted dynamic time warping.' In: *VISAPP (1)*. 2013, pp. 620–625.

[82] Arthur Mensch and Mathieu Blondel. 'Differentiable dynamic programming for structured prediction and attention'. In: *arXiv preprint arXiv:1802.03676* (2018).

[83] Carlos M Fonseca, Peter J Fleming et al. 'Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization.' In: *Icga*. Vol. 93. July. 1993, pp. 416–423.

[84] Tomasz Górecki and Maciej Łuczak. 'Using derivatives in time series classification'. In: *Data Mining and Knowledge Discovery* 26.2 (2012), pp. 310–331. DOI: 10.1007/s10618-012-0251-4. URL: https://doi.org/10.1007/s10618-012-0251-4.

[85] Massimo Panella and Rosa Altilio. 'A Smartphone-Based Application Using Machine Learning for Gesture Recognition: Using Feature Extraction and Template Matching via Hu Image Moments to Recognize Gestures'. In: *IEEE Consumer Electronics Magazine* 8.1 (2019), pp. 25–29. DOI: 10.1109/mce.2018.2868109. URL: https://doi.org/10.1109/mce.2018.2868109.

[86] Liu Yun, Zhang Lifeng and Zhang Shujun. 'A Hand Gesture Recognition Method Based on Multi-Feature Fusion and Template Matching'. In: *Procedia Engineering* 29 (2012), pp. 1678–1684. DOI: 10.1016/j.proeng.2012.01.194. URL: https://doi.org/10.1016/j.proeng.2012.01.194.

[87] Hong Cheng et al. 'An image-to-class dynamic time warping approach for both 3D static and trajectory hand gesture recognition'. In: *Pattern Recognition* 55 (2016), pp. 137–147. DOI: 10.1016/j.patcog.2016.01.011. URL: https://doi.org/10.1016/j.patcog.2016.01.011.

[88] Gerrit Niezen and Gerhard P Hancke. 'Gesture recognition as ubiquitous input for mobile phones'. In: *Devices that Alter Perception* (2008).

[89] Ana Kuzmanic and Vlasta Zanchi. 'Hand shape classification using DTW and LCSS as similarity measures for vision-based gesture recognition system'. In: *EUROCON 2007 - The International Conference on "Computer as a Tool"*. IEEE, 2007. DOI: 10.1109/eurcon.2007.4400350. URL: https://doi.org/10.1109/eurcon.2007.4400350.

[90] Tristan Dot et al. 'Non-Linear Template-Based Approach for the Study of Locomotion'. In: *Sensors* 20.7 (2020), p. 1939. DOI: 10.3390/s20071939. URL: https://doi.org/10.3390/s20071939.

[91] Jingren Tang et al. 'Structured dynamic time warping for continuous hand trajectory gesture recognition'. In: *Pattern Recognition* 80 (2018), pp. 21–31. DOI: 10.1016/j.patcog.2018.02.011. URL: https://doi.org/10.1016/j.patcog.2018.02.011.

[92] Manabu Okawa. 'Template Matching Using Time-Series Averaging and DTW With Dependent Warping for Online Signature Verification'. In: *IEEE Access* 7 (2019), pp. 81010–81019. DOI: 10.1109/access.2019.2923093. URL: https://doi.org/10.1109/access.2019.2923093.

[93] Manabu Okawa. 'Online signature verification using single-template matching with time-series averaging and gradient boosting'. In: *Pattern Recognition* 102 (2020), p. 107227. DOI: 10.1016/j.patcog.2020.107227. URL: https://doi.org/10.1016/j.patcog.2020.107227.

[94] Darya Frolova, Helman Stern and Sigal Berman. 'Most Probable Longest Common Subsequence for Recognition of Gesture Character Input'. In: *IEEE Transactions on Cybernetics* 43.3 (2013), pp. 871–880. DOI: 10.1109/tsmcb.2012.2217324. URL: https://doi.org/10.1109/tsmcb.2012.2217324.

[95] A. Corradini. 'Dynamic time warping for off-line recognition of a small gesture vocabulary'. In: *Proceedings IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*. IEEE Comput. Soc. DOI: 10.1109/ratfg.2001.938914. URL: https://doi.org/10.1109/ratfg.2001.938914.

[96] Javier Molina, José Antonio Pajuelo and José M. Martínez. 'Real-time Motion-based Hand Gestures Recognition from Time-of-Flight Video'. In: *Journal of Signal Processing Systems* 86.1 (2015), pp. 17–25. DOI: 10.1007/s11265-015-1090-5. URL: https://doi.org/10.1007/s11265-015-1090-5.

[97] Bastian Hartmann, Ingo Schwab and Norbert Link. 'Prototype optimization for temporarily and spatially distorted time series'. In: *2010 AAAI Spring Symposium Series*. 2010.

[98] Bastian Hartmann and Norbert Link. 'Gesture recognition with inertial sensors and optimized DTW prototypes'. In: *2010 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2010. DOI: 10.1109/icsmc.2010.5641703. URL: https://doi.org/10.1109/icsmc.2010.5641703.

[99] Yanping Chen et al. *The UCR Time Series Classification Archive*. www.cs.ucr.edu/~eamonn/time_series_data/. 2015.

[100] Piero Zappi et al. 'Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness'. In: *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. IEEE, 2007. DOI: 10.1109/issnip.2007.4496857. URL: https://doi.org/10.1109/issnip.2007.4496857.

[101] Nvidia. *CUDA Toolkit*. Retrieved on April 26, 2019 from https://developer.nvidia.com/cuda-toolkit. 2019. URL: https://developer.nvidia.com/cuda-toolkit.

[102] Doruk Sart et al. 'Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs'. In: *2010 IEEE International Conference on Data Mining*. IEEE, Dec. 2010. DOI: 10.1109/icdm.2010.21. URL: https://doi.org/10.1109/icdm.2010.21.

[103] Yang Jiaoyun et al. *An Efficient Parallel Algorithm for Longest Common Subsequence Problem on GPUs*. 2010. ISBN: 9789881701299. URL: https://www.researchgate.net/publication/45534436.

[104] Chris Gregg and Kim Hazelwood. 'Where is the data? Why you cannot debate CPU vs. GPU performance without the answer'. In: *(IEEE ISPASS) IEEE INTERNATIONAL SYMPOSIUM ON PERFORMANCE ANALYSIS OF SYSTEMS AND SOFTWARE*. IEEE, Apr. 2011. DOI: 10.1109/ispass.2011.5762730. URL: https://doi.org/10.1109/ispass.2011.5762730.

[105] Sushmita Mitra and Tinku Acharya. 'Gesture Recognition: A Survey'. In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 37.3 (May 2007), pp. 311–324. DOI: 10.1109/tsmcc.2007.893280. URL: https://doi.org/10.1109/tsmcc.2007.893280.

[106] Maja Stikic and Bernt Schiele. 'Activity Recognition from Sparsely Labeled Data Using Multi-Instance Learning'. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 156–173. DOI: 10.1007/978-3-642-01721-6_10. URL: https://doi.org/10.1007/978-3-642-01721-6_10.

[107] Manchek A Wong and JA Hartigan. 'Algorithm as 136: A k-means clustering algorithm'. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108.

[108] Long-Van Nguyen-Dinh, Alberto Calatroni and Gerhard Tröster. 'Robust Online Gesture Recognition with Crowdsourced Annotations'. In: *Gesture Recognition*. Springer International Publishing, 2017, pp. 503–537. DOI: 10.1007/978-3-319-57021-1_18. URL: https://doi.org/10.1007/978-3-319-57021-1_18.

[109] William Duffy et al. 'Addressing the Problem of Activity Recognition with Experience Sampling and Weak Learning'. In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, Nov. 2018, pp. 1238–1250. DOI:

10.1007/978-3-030-01054-6_86. URL: https://doi.org/10.1007/978-3-030-01054-6_86.

[110] Long-Van Nguyen-Dinh, Alberto Calatroni and Gerhard Troster. 'Supporting One-Time Point Annotations for Gesture Recognition'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.11 (Nov. 2017), pp. 2270–2283. DOI: 10.1109/tpami.2016.2637350. URL: https://doi.org/10.1109/tpami.2016.2637350.

[111] Maja Stikic, Kristof Van Laerhoven and Bernt Schiele. 'Exploring semi-supervised and active learning for activity recognition'. In: *2008 12th IEEE International Symposium on Wearable Computers*. IEEE, 2008. DOI: 10.1109/iswc.2008.4911590. URL: https://doi.org/10.1109/iswc.2008.4911590.

[112] Yongjin Kwon, Kyuchang Kang and Changseok Bae. 'Unsupervised learning for human activity recognition using smartphone sensors'. In: *Expert Systems with Applications* 41.14 (Oct. 2014), pp. 6067–6074. DOI: 10.1016/j.eswa.2014.04.037. URL: https://doi.org/10.1016/j.eswa.2014.04.037.

[113] Hristijan Gjoreski and Daniel Roggen. 'Unsupervised online activity discovery using temporal behaviour assumption'. In: *Proceedings of the 2017 ACM International Symposium on Wearable Computers*. ACM, Sept. 2017. DOI: 10.1145/3123021.3123044. URL: https://doi.org/10.1145/3123021.3123044.

[114] Xu Zhang et al. 'A Framework for Hand Gesture Recognition Based on Accelerometer and EMG Sensors'. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41.6 (Nov. 2011), pp. 1064–1076. DOI: 10.1109/tsmca.2011.2116004. URL: https://doi.org/10.1109/tsmca.2011.2116004.

[115] Rüdiger Zillmer et al. 'A robust device for large-scale monitoring of bar soap usage in free-living conditions'. In: *Personal and Ubiquitous Computing* 18.8 (Feb. 2014), pp. 2057–2064. DOI: 10.1007/s00779-014-0760-9. URL: https://doi.org/10.1007/s00779-014-0760-9.

[116] Zbigniew Michalewicz. 'Evolution Strategies and Other Methods'. In: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Berlin Heidelberg, 1996, pp. 159–177. DOI: 10.1007/978-3-662-03315-9_9. URL: https://doi.org/10.1007/978-3-662-03315-9_9.

[117] Mathias Ciliberto et al. 'High reliability Android application for multidevice multimodal mobile data acquisition and annotation'. In: *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems - SenSys '17*. ACM Press, 2017. DOI: 10.1145/3131672.3136977.

[118] Hristijan Gjoreski et al. 'A Versatile Annotated Dataset for Multimodal Locomotion Analytics with Mobile Devices'. In: *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems - SenSys '17*. ACM Press, 2017. DOI: 10.1145/3131672.3136976.

[119] Hristijan Gjoreski et al. 'The University of Sussex-Huawei Locomotion and Transportation Dataset for Multimodal Analytics With Mobile Devices'. In: *IEEE Access* 6 (2018). DOI: 10.1109/ACCESS.2018.2858933. URL: https://ieeexplore.ieee.org/document/8418369/.

[120] Lin Wang et al. 'Benchmarking the SHL recognition challenge with classical and deep-learning pipelines'. In: *UbiComp/ISWC 2018 - Adjunct Proceedings of the 2018 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2018 ACM International Symposium on Wearable Computers*. Association for Computing Machinery, Inc, 2018, pp. 1626–1635. ISBN: 9781450359665. DOI: 10.1145/3267305.3267531.

[121] Lin Wang et al. 'Enabling reproducible research in sensor-based transportation mode recognition with the sussex-huawei dataset'. In: *IEEE Access* 7 (2019). ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2890793.

[122] Lin Wang et al. 'Summary of the Sussex-Huawei locomotion-transportation recognition challenge 2019'. In: *UbiComp/ISWC 2019- - Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*. Association for Computing Machinery, Inc, 2019. DOI: 10.1145/3341162.3344872.

[123] Sebastien Richoz et al. 'Human and machine recognition of transportation modes from body-worn camera images'. In: *2019 Joint 8th International Conference on Informatics, Electronics and Vision, ICIEV 2019 and 3rd International Conference on Imaging, Vision and Pattern Recognition, icIVPR 2019 with International Conference on Activity and Behavior Computing, ABC 2019*. 2019. DOI: 10.1109/ICIEV.2019.8858537.

[124] Lin Wang et al. 'Summary of the sussex-huawei locomotion-transportation recognition challenge 2020'. In: *UbiComp/ISWC 2020 Adjunct - Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*. Association for Computing Machinery, 2020. DOI: 10.1145/3410530.3414341.

Appendices

Appendix A

Recruiting participant for collecting data of beach volleyball serves and games

This appendix integrates the recruitment of participants for the data collection described in Chapter 2. The procedure of recruitment was approved by an ethical committee by the University of Sussex. In order to get the approval, the committee required to fill a risk assessment, a privacy information sheet for the participant to sign and to display any recruiting materials such as flyers. The ethical approval was granted under the internal code ER/MC606/1. An extension in time was also approved with the code ER/MC606/2, with no additional material required. In the following, the recruiting flyer and the privacy information sheet are reported.

Are you a beach volleyball player? Do you want to participate to a research project involving wearable technologies to analyse and improve the quality of your movements during the training?

If you are interested or for any further information, please contact:

Mathias Ciliberto
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

Mathias Ciliberto
Beach volleyball research project
m.ciliberto@sussex.ac.uk

University of Sussex

PARTICIPANT INFORMATION SHEET

PROJECT TITLE: Improvement of the quality of the execution of beach volleyball techniques using wearable sensors

What is the purpose of the project?
The goal of this project is to develop and to evaluate an automated system for performance analysis and improvement of beach volleyball player during the training. Specifically, this project is aimed to evaluate and improve the execution of a set of beach volleyball techniques by analysing the movements of the body parts instrumented with wearable sensors. These sensors are inertial sensors capable to measure their movement in a 3D space.

The movements of a player will be analysed using 12 sensors with pattern matching algorithms. The project requires first collecting a dataset of executions of every beach volleyball techniques (serve, receive, pass, hit, block, dig), for each possible variation (e.g. float serve, spin serve, jump float serve, jump spin serve, etc.). Several videos of each execution will be recorded during the data collection. By using these videos, each instance in this dataset will be evaluated and annotated by an expert with a score of the quality of the execution.

The system will be then able to evaluate future executions by finding the most similar instance in the dataset using algorithm for pattern matching. In this case, the patterns will be the sequence of the orientation of the body parts measured with the inertial sensors.

Participants in this project will be instrumented with the 12 sensors and will be asked to perform a sequence of techniques.

Do I have to take part?
It is up to you to decide whether or not take part. If you decide to accept to participate, you would be given this information sheet to keep and be asked to sign a consent form. If you decide to take part, you are still free to withdraw at any time and without giving a reason.

What will happen to me if I accept?
If you accept you would be involved in the project data collection and system evaluation.

As first, during the data collection phase, you would be asked to perform a set of beach volleyball techniques, following a specific protocol, simulating a training session. After having worn the sensors, an initial briefing and a warm up, you will be asked to perform each technique for a 2 minutes length interval, following the indication that will be given and at the best of you possibilities. The entire session won't last for more than 1 hour. You will have 2-5 minutes of rest between every exercise interval. Anyway, you can stop and pause whenever you want.

During the evaluation of the system, you will be asked to perform a subset of the techniques wearing the system. You should perform each technique 5-10 times, then the system will analyse your performance. After that, you will be asked to perform the same technique 5-10 times. Between each set of executions you can rest for 1-2 minutes. Anyway, you can stop and pause whenever you want.

Along all the experiments and the evaluations, you could be asked to answer several qualitative surveys anonymously.

What are the possible disadvantages and risks of taking part?
You would be requested to perform a set of beach volleyball techniques. The data collection and system's evaluation sessions will be similar to training sessions, where the risks are in line with those you could encounter during an actual training (without the sensors and the system).
These sessions may be tiring, however you wouldn't be asked to over exert your body or engage in any excessive physical activity. Even though there will be several breaks, you could pause and/or stop if you need it.

Will my information in this study be kept confidential?
The dataset collected during this project will consist of several different data: the inertial data from the sensors, the videos recorded during the data collection, the evaluation of the expert annotators of the quality of the executions and the data derived from the qualitative surveys.

The inertial data as well as the qualitative surveys' data would be unlinked from your personal information and they will be anonymised. These data would be stored in a deidentified way (e.g. using ID numbers not names), and kept separate from other details about you.

The videos will be treated according with your preference on the consent form. They will never be linked directly to your personal details, but you could choose if you want to publish your videos in the dataset entirely (leaving your face visible), if you want to make your face blurred/pixelated, or if you prefer to exclude your videos from the datasets.

After the data collection is completed, no sensor information leading to the identification of any individual would be kept.

What will happen to the result of the research project?
The results of this research may be written into scientific reports for publication in academic venues. Your anonymity will be ensured in the way described before. If you are interested in copies of the published research, you can contact any researcher involved in the project.

We plan to release publicly the collected dataset (partly or completely), after proper anonymization and unlinking process, to make it accessible to the wider scientific community.

Who is organising the research?
This research is conducted by staff members of the University of Sussex, School of Engineering and Informatics.

Who has approved this study?
This study has been approved by the Sciences & Technology Cross-Schools Research Ethics Committee (crecscitec@sussex.ac.uk). The project reference number is ER/MC606/2.
University of Sussex has insurance in place to cover its legal liabilities in respect of this study.

Contact for Further Information
In case you would require further information, you can contact:

Mathias Ciliberto (PhD student – Project coordinator) at m.ciliberto@sussex.ac.uk
Dr. Daniel Roggen (Reader in Sensor Technology – Project Supervisor) at daniel.roggen@ieee.org
Dr. Luis Ponce Cuspinera (Lecturer in Mechatronics – Project Supervisor) at l.ponce-cuspinera@sussex.ac.uk

If you have any concerns about the way in which the study is conducted, you may contact the Science and Technology Cross-Schools Research Ethics Committee at crescitec@sussex.ac.uk

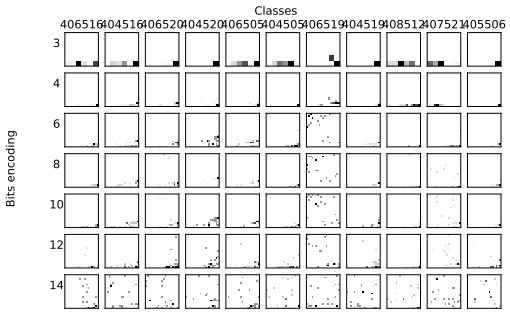
Thank you

Date
03/03/2019

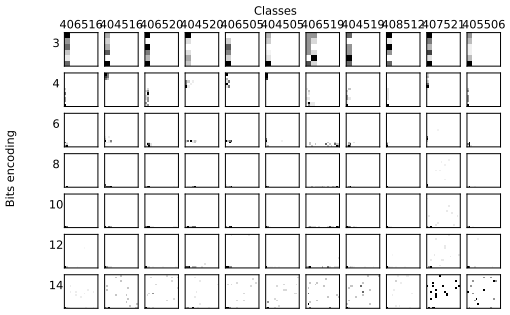
Appendix B

WLCSSLearn_p: Distribution of generated R,P, ϵ

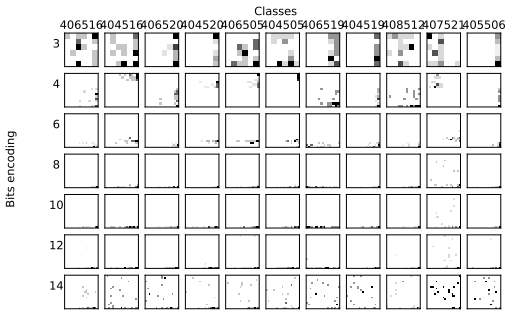
This appendix complements the analysis from Section 5.5 with the additional plots presented in Figure B.1 showing the distribution of the generated R, P, and ϵ for the opportunity_encoded, hci_guided, skoda_mini, hci_table, beach_volleyball datasets not included in the main chapter. The figures confirm the findings from Section 5.5: there is no clear probability distribution of the values of R, P, and ϵ, which means that they must be optimized in every application.



(a) R, P

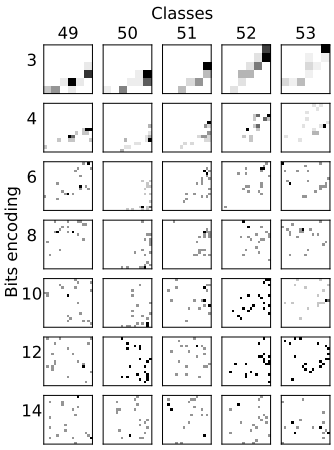


(b) P, ϵ

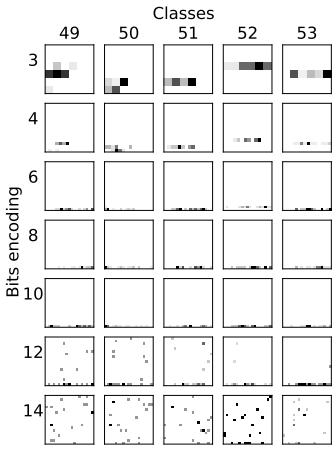


(c) R, ϵ

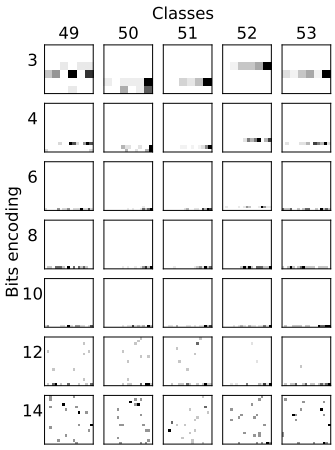
opportunity_encoded



(d) R, P

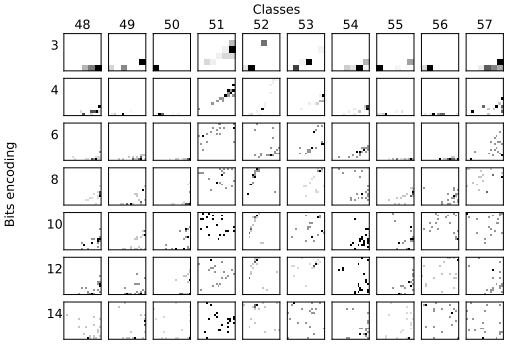


(e) P, ϵ

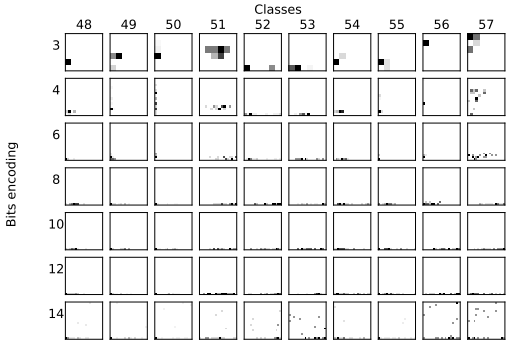


(f) R, ϵ

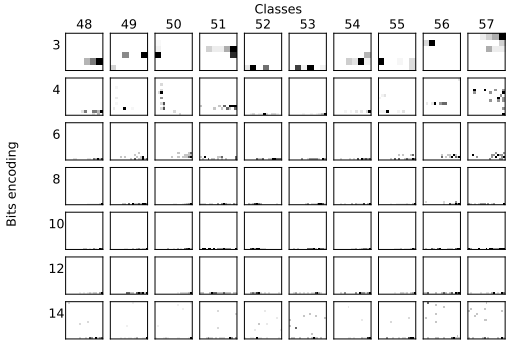
hci_guided



(g) R, P

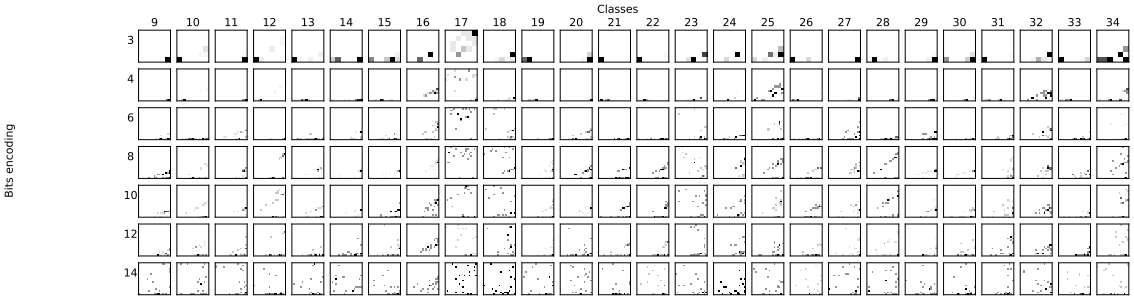


(h) P, ϵ

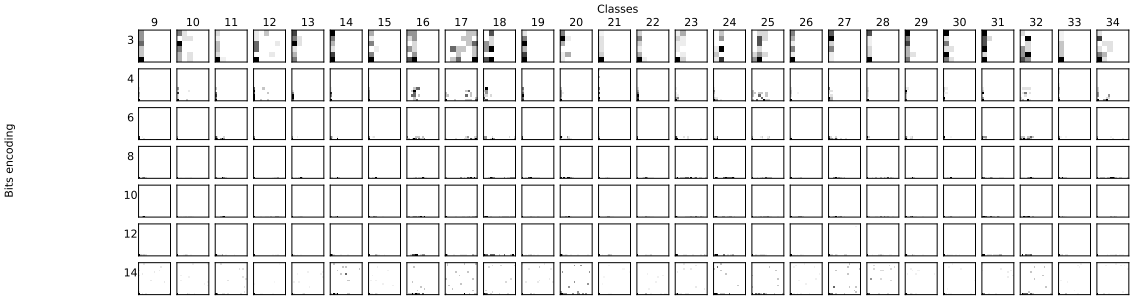


(i) R, ϵ

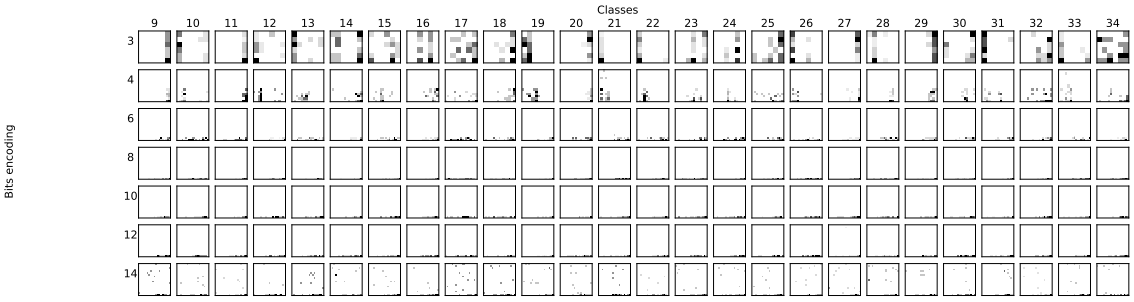
skoda_mini



(j) R, P

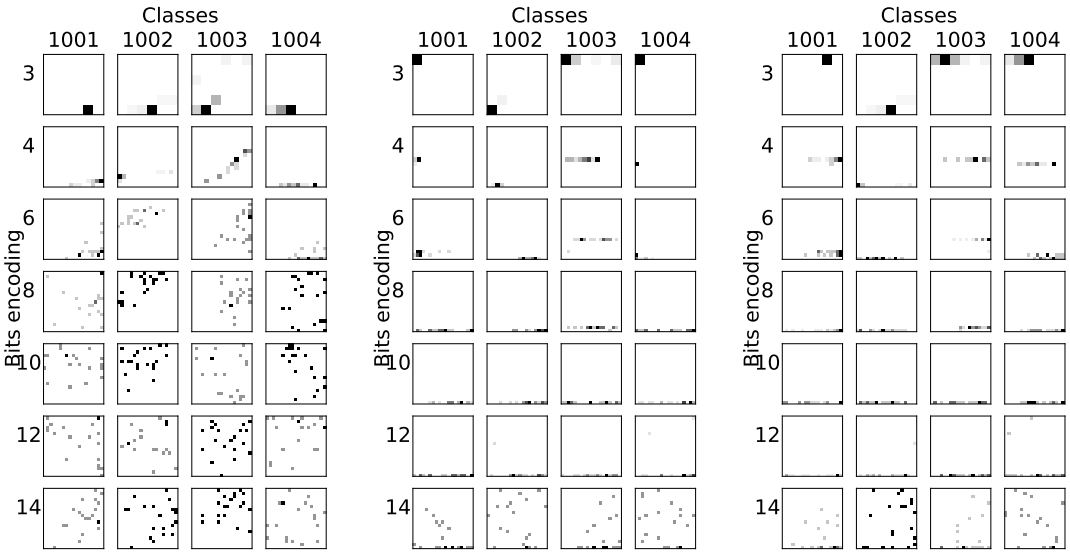


(k) P, ϵ



(l) R, ϵ

hci_table



(m) R, P **(n)** P, ϵ **(o)** R, ϵ

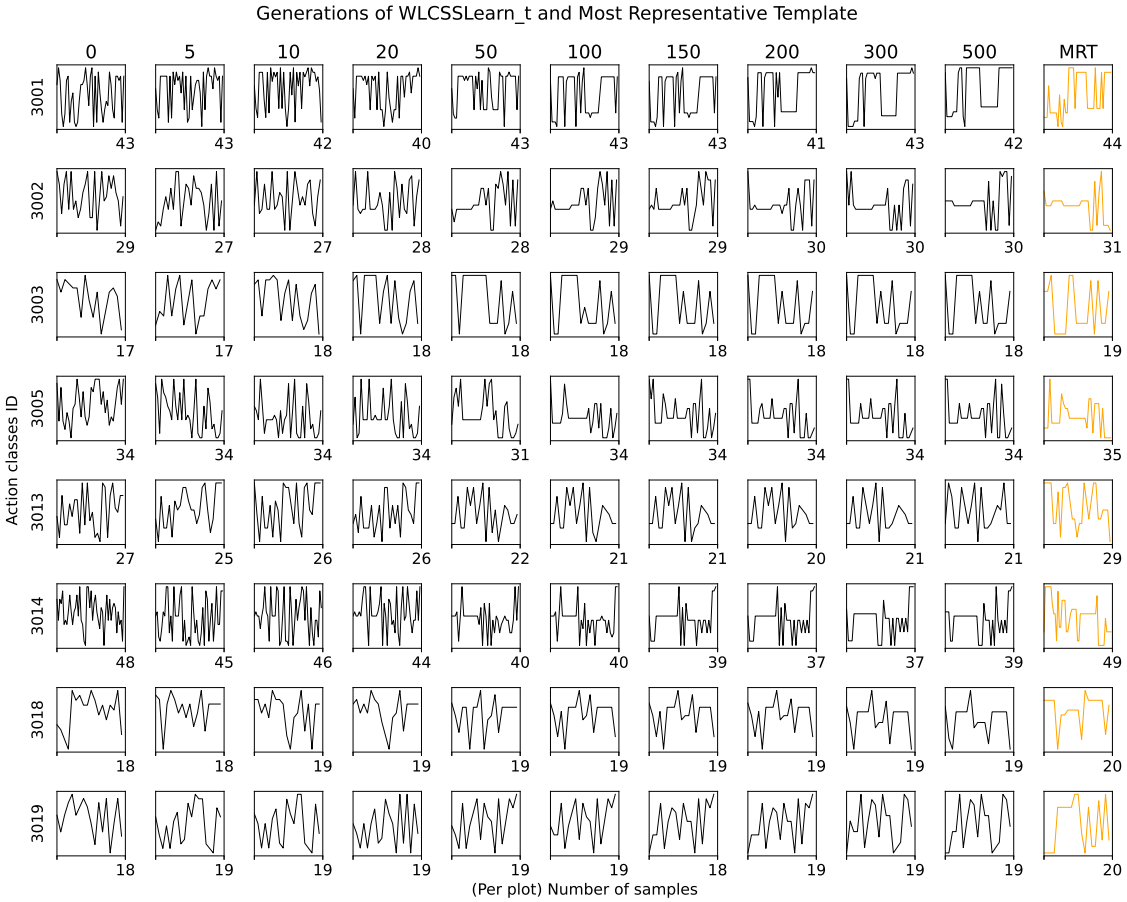
beach_volleyball

Figure B.1: *Probability distribution of values R, P, and ϵ in pairs for opportunity_encoded, hci_guided, skoda_mini, hci_table and beach_volleyball dataset. It is possible to notice how the parameters do not follow any specific distribution, making necessary to train all of them.*

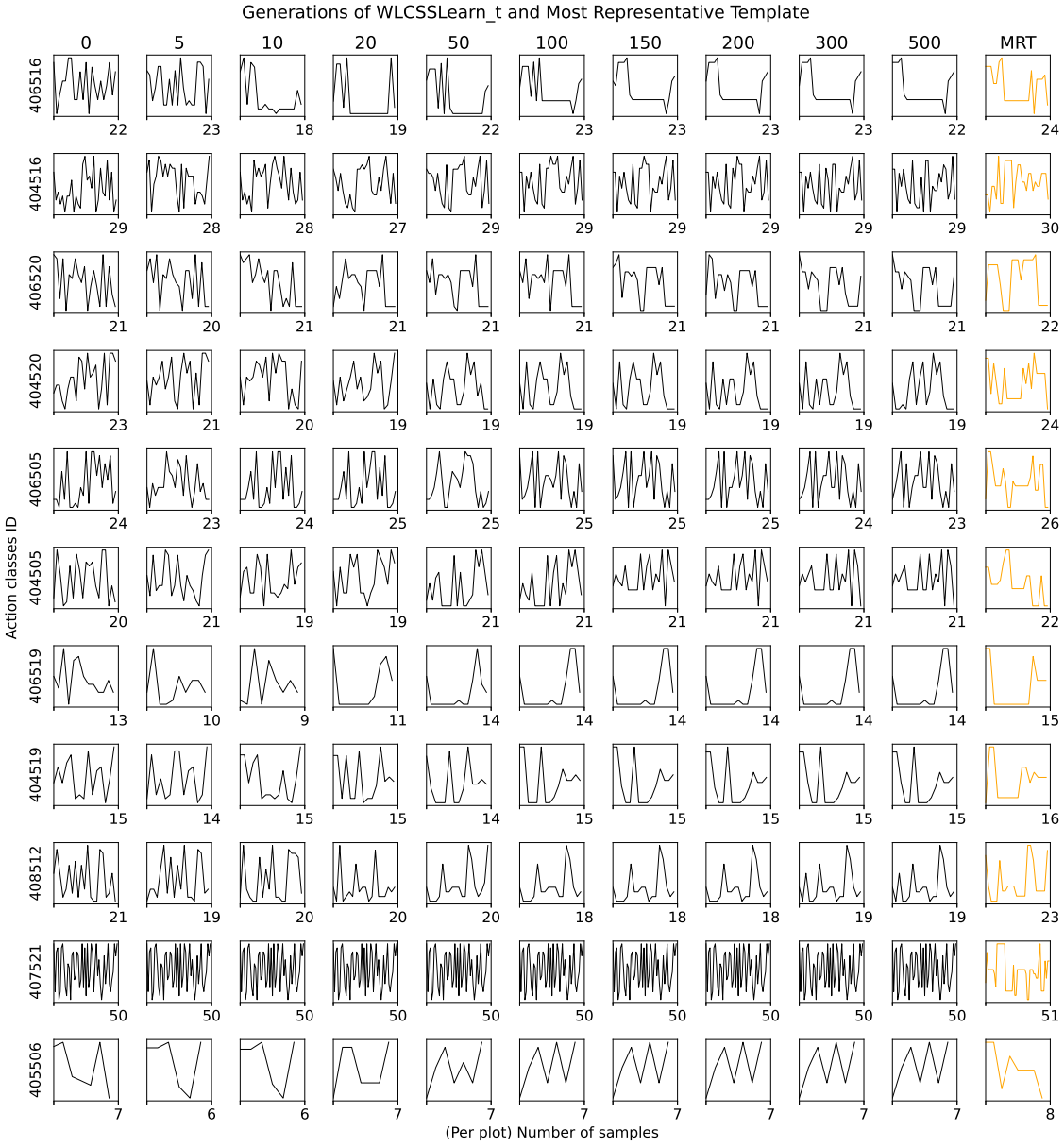
Appendix C

WLCSSLearn_t: Shape and length of generated templates compared to MRT

This appendix integrates the comparison of the shape and length of the templates with the highest fitness generated by WLCSSLearn_t with respect to the Most Representative Templates (MRT), for each action class in every dataset (see Section 6.3.1). The figures show how the evolutionary process tends to reduce the length of the templates while increasing the most relevant features of the templates for each action class.


Generations of WLCSSLearn_t and Most Representative Template
0
5
10
20
50
100
150
200
300
500
MRT
Action classes ID
3001
3002
3003
3005
3013
3014
3018
3019
(Per plot) Number of samples

(a) *skoda*



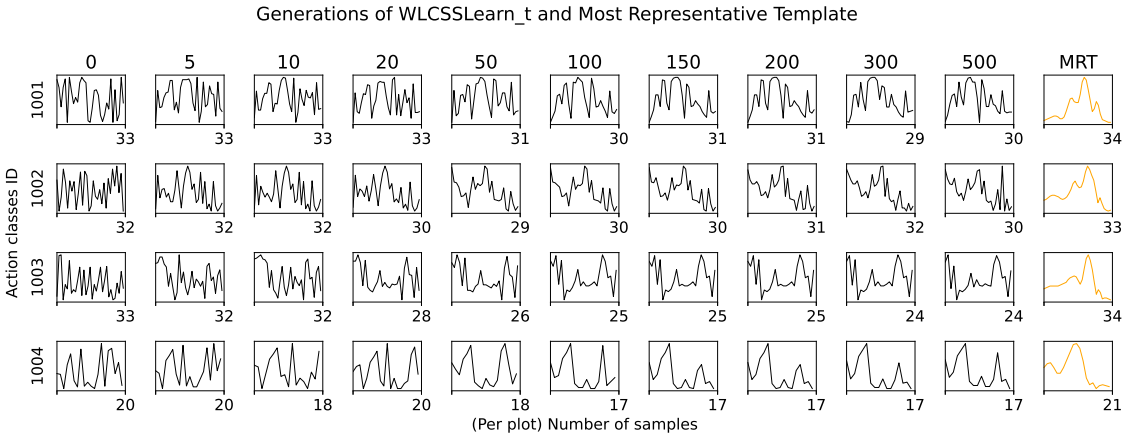
(b) *opportunity_encoded*


Generations of WLCSSLearn_t and Most Representative Template
0
5
10
20
50
100
150
200
300
500
MRT
Action classes ID
48
49
50
51
52
53
54
55
56
57
(Per plot) Number of samples

(c) *skoda_mini*


Generations of WLCSSL on t and Most Representative Template
0
5
10
20
50
100
150
200
300
500
MRT
Action classes ID
(Per plot) Number of samples

(d) *hci_table*



(e) *beach_volleyball*

Figure C.1: *Comparison of best fitting variable templates generated with WLCSSLearn_t (black), sampled at several generations, and MRT (orange) for all the dataset. Over all the iteration, the length of the best individual reduces as well as its visual complexity.*

Appendix D

Publications

This appendix includes a re-print of all the original publications this thesis is built upon and that are listed in the List of Publications. The work described in Chapter 5 and 6 is being prepared to be published in [**VIII**] and therefore has not been included in this appendix.

Exploring Human Activity Annotation Using a Privacy Preserving 3D Model

Mathias Ciliberto
Wearable Technologies,
Sensor Technology Research Centre,
University of Sussex
m.ciliberto@sussex.ac.uk

Francisco Javier Ordóñez Morales
Wearable Technologies,
Sensor Technology Research Centre,
University of Sussex
f.ordonez-morales@sussex.ac.uk

Daniel Roggen
Wearable Technologies,
Sensor Technology Research Centre,
University of Sussex
daniel.roggen@ieee.org

Ubicomp/ISWC'16 Adjunct, September 12 - 16, 2016, Heidelberg, Germany

ACM 978-1-4503-4462-3/16/09...$15.00
DOI: http://dx.doi.org/10.1145/2968219.2968290

Abstract
Annotating activity recognition datasets is a very time consuming process. Using lay annotators (e.g. using crowdsourcing) has been suggested to speed this up. However, this requires to preserve privacy of users and may preclude relying on video for annotation. We investigate to which extent using a 3D human model animated from the data of inertial sensors placed on the limbs allows for annotation of human activities. We animate the upper body of the 3D model with the data from 5 inertial measurement sensors obtained from the OPPORTUNITY dataset. The animated model is shown to 6 people in a suite of experiments in order to understand to which extent it can be used for labelling. We present 3 experiments where we investigate the use of a 3D model for i) activity segmentation, ii) for "open-ended" annotation where users freely describe the activity they see on screen, and iii) traditional annotation, where users pick one activity among a pre-defined list of activities. In the latter case, results show that users recognise the model's activities with 56% accuracy when picking from 11 possible activities.

Author Keywords
Activity recognition; annotation; wearable technologies; 3D human model.

ACM Classification Keywords
H.5.m [Information interfaces and presentation]: Miscellaneous

Introduction
Activity recognition is fundamental for context-aware computing [3]. For example, it can be used to understand the routines of a user to support rehabilitation or personalised healthcare [2].

In order to achieve a reliable recognition of the activity of the user, an annotated dataset is needed to train the machine learning classifier. Annotation requires that someone manually specify the actions carried out during the data recording. To do this, usually several videos are recorded jointly with the recording of wearable sensor data. After the recording, the video from the cameras are synchronized with the data logged by the sensors, in order to annotate precisely the start and the end of each activity. This can be very time-consuming [11]. For this reason, it is usually done using cheap labour with recent research even looking at crowdsourcing. In such a case, it is yet important to preserve the privacy of the user whose data was recorded.

In this work we investigate to which extent a 3D human model animated from inertial sensors placed on the user's limbs can be used to label the activities of that user while preserving his/her privacy.

The contribution of this paper are:

- a 3D human model created to reproduce the user movements. The model is developed in Java and it can be and deployed on different platforms, allowing a wide application in a crowdsourcing scenario. We animate the model's upper and lower arms and back from the inertial sensor data available in the OPPORTUNITY dataset [11].
- an investigation about the segmentation of the activities of the 3D model (i.e. identifying the occurence of an activity regardless of its class). We compare the results of segmentation done by the participants to the experiments with the ground truth segmentation of activities in the dataset.
- an analisys of the system in a open-ended annotation scenario: we let the user free to assign a custom label to each activity in a set. The goal of this experiment is to understand to which extent this model can be used without any knownledge of the action carried out by the user during the recording. The results of these test are presented using a set of tag clouds of the words used by testers.
- an evaluation of traditional annotation accuracy, where an activity annotated in the dataset is played by the model and the testers must pick which activity it could have been among a list of 11 activities. Results are summarised by a confusion matrix.

State of the art
There are multiple approaches to annotation [5]. Usually the annotation or labelling of the collected data is done a posteriori using a video recording of the experiment synchronised with the sensor data [11]. During this process the video synchronized with the inertial data requires that each sequence of data must be accurately analyzed to segment and recognize each activity correctly. Sometimes multiple cameras are used to record the scene from different perspectives [11]. The activity of labelling can be a tedious and time-consuming task: for 30 minutes of recording, the annotation could take 7-10 hours of analysis [11]. For this reason, usually the annotation phase is done using cheap labour.

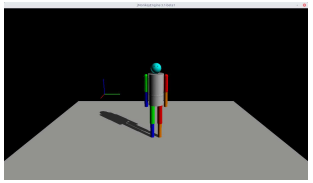


Figure 1: The human model. This interface is also used during the first experiment, where the user must press a button to point out when something "interesting" happened to the model. No output is presented to the user when he/she press the button.

Recently, crowd-sourcing has been suggested to help reduce the cost and time of annotating datasets. Crowd-sourcing is a process where a task can be completed by soliciting contributions from a large group of lay people. Thanks to this technique, the researchers can obtain an annotated dataset employing several people at the same time. This can be realised, for example, using platforms like Amazon Mechanical Turk (MTurk) [1]: this is a web service where users can ask for workforce. The workers can pick up a task and complete it earning a money reward. Crowd-sourcing has been used to tag human activity from video [9]. It has also been used to label natural language [14], for speech recognition [10] and for multimedia tagging [13].

When the annotation task is done using the videos, one of main issues is to preserve the privacy of the subject in the dataset/video. One way to protect the anonymity of the subjects in the dataset can be the application of a mask (e.g. a blur or a pixelize video filter [4]) to the elements in the video that could be considered meaningful from a privacy point of view. This step brings however additional time and work to the annotation task. Moreover, the preprocessing cannot be easily applied to all the elements in the scene in order to do not alter too much the video and to do not compromise the recognition of the activities. Another approach consists in using low resolution camera to preserve the privacy of the subject recorded [6].

A different way can be the real-time annotation of the data. That can be done using short audio labels recorded by the subject of the dataset together with the inertial data [7] [12]. This method preserves the privacy and can be accurate. It can be also used in a "open-ended" context thanks to the absence of a predefined set of labels. Real-time labelling requires the direct interaction of the user and in the everyday life this could be annoying. Instead, in case of a real time annotation made by an external experimenter that observe the scene, it requires that the user carries out the activities in a controlled enviroments and it is not always possible. Moreover, the external observer could condition the way the user will do the activities.

Experimental setup

The human model is rendered using the open-source 3D engine called jMonkeyEngine [8]. This multiplatform engine written in Java allows to develop a model from the ground up: we built our custom model to keep easier the handling of the animation and to guarantee more flexibility during the application of inertial data to each body part (Figure 1).

To animate the model, we used the data provided by 5 Inertial Measurement Units (IMUs) placed on the upper arms, on the lower arms and on the torso. IMUs provide a quaternion representing the orientation of the sensor and limb in a world coordinate system. After a precomputation step, we applied the respective quaternion to each body part thanks to the engine, which handle directly this formalism. The multiplatform nature of the engine would allow us also to deploy this model and its animation in a crowdsourcing scenario, because it can be integrated and used remotely, eg. in a web page.

The engine allows the users to rotate the camera around the model and zoom in and out to better observe and evaluate each action.

In this paper we use the the Opportunity Dataset [11]. This dataset comprises a rich set of 17 naturalistic activities recorded in a kitchen environment. The activities are:

- Open and close two different doors;
- Open and close three drawers at three different heights;
- Open and close a dishwasher;

Activity	sec
Open Fridge	2.450
Clean Table	3.967
Close Fridge	2.508
Drink from Cup	6.268
Open Dishwasher	2.958
Open Door	3.214
Close Dishwasher	2.862
Open Drawer	2.363
Close Door	3.378
Toggle Switch	1.295
Close Drawer	2.207
Average	3.043

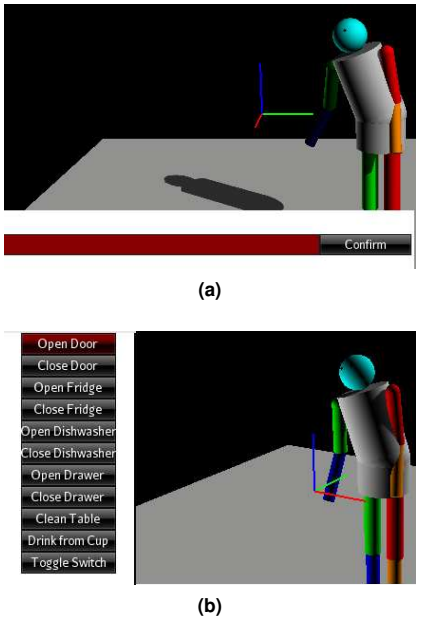
Table 1: List of activities and their respective average length in seconds. The last row shows the average length for all the activities.

- Open and close a fridge;
- Clean a table;
- Drink from a cup;
- Toggle a switch.

This dataset consist of inertial data about the absolute orientation of each limb during the session. These data have been recorded using a set of XSens MTx inertial sensors [15]. For our experiments we used the "Drill" run subset. This subset consist of a fixed sequence of 17 actions repeated consecutively for about 20 minutes. Due to the absence of the environment in the 3D engine, we decided to join some of the similar labels (e.g. interacting with drawers at different heights is combined). "Open" and "Close" are considered diffent actions. From the initial 17 activities in the Opportunity dataset, we obtain the 11 activities shown in the Table 1, together with their average length in seconds.

We perfomed three experiments. The participants in the experiments were told that they would see a 3D model of a person performing typical activities in a kitchen. The participants were not given the list of activities at first. Essentially they have to "guess" from the animation of the model which activity may be undertaken. In the first experiment, a 15 minutes animation is played by the model. During this animation each participant must press the space-bar every time he/she notices something that he/she considers interesting and/or recognizable in the model's movements. It is up to the participant to decide what they consider "interesting". This experiment is used to evaluate the capability of the users to segment the activities using the model. The interface used during this test is the same shown in Figure 1.

In the second and in the third experiment, a set of short animations of the body model are shown to the participant



(a)

(b)

Figure 2: Interfaces developed for the experiments. At the top, the interface developed to let the user to insert the label manually (Experiment 2). At the bottom, the set of buttons that the user can click to select an annotation for the activity (Experiment 3).

where the model performs exactly one of the 11 possible activities. For each of the 11 activities we showed the animations of 4 activity instances picked randomly from the dataset. In this way, we show to the user a random but balanced set of activities. The set of 44 short animations has been showed to the tester in a random order. This set was different for each participant.

The second experiment is carried out to investigate to which extent the application of our model fits an "open ended" scenario. In this experiment, the task of the users is to in-

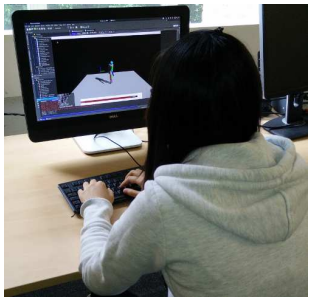


Figure 3: Setup of the experiment.

TP	FN	TN	FP	Acc.
70	57	113	4	0.75
58	69	97	20	0.64
34	93	110	7	0.59
92	35	98	19	0.78
95	32	107	10	0.83
45	82	103	14	0.61

Table 2: Test results for the first experiment. The true positive (TP) and the false positive (FP) do not keep in count if the user pointed out more events for the same activity/pause. Despite this affects also the accuracy, it can be used to better understand the results on a quality level. The total number of actitities played by the model during the Test 1 is 127, with 117 pauses between some of the activities. A pause is intended as a moment between two actions, where the model is not doing anything "interesting".

sert a short label for each animation. We developed the interface displayed in Figure 2(a) in order to allow the user to enter the label. This interface was showed after each animation of the set. The user had no time limit to enter the label. After he/she confirmed the inserted label, the next animation in the set is played.

The last experiment is useful to test the ability to annotate using a 3D model in a more traditional scenario. The system shows a push button for each of the 11 predefined activities. The participants must select which activity they think was performed by the model by pressing the corresponding push button with the mouse. The buttons are shown at the end of each short animation without any limit of time for the users. In Figure 2(b) are shown the buttons. After he/she selected a label, the next animation in the set is played. This corresponds to the traditional annotation approach where a pre-defined list of activities are annotated.

The experiment is carried out with 6 people, that are unaware about the dataset and the set of labels until the last experiment. The setup of the experiment is shown in Figure 3. All the participants performed the experiments in the same order and individually. They were instructed before each test as to what they would have to do next, in order to not influence the next phase of the experiment.

Results

Every experiment is designed to test a specific step or a different scenario of the annotation. With the first experiment, we aim to evaluate the capability of the users to segment the activities. In Figure 4, we show the results of the segmentation experiments for a subset of the dataset. The first row of the figure represents the distribution of the activities throughout the first 3 minutes of the experiment. The next 6 rows show the events pointed out by each participant.

Every vertical line is an event. It allows to compare the distribution of the event recorded by each participant and the actual distribution of the activities.

As the experiment left the participants free to decide what they consider "interesting", we observe a large variations in the frequency of the events recorded by each participant. This may be explained because some participants tent to point out longer actions while other pointed out more shorter task. For example, the participant 3 recorded less events than the participant 5.

In Table 2, we show the number of activities made by the model and correctly pointed out (true positive), the number of pauses (intended as an interval in the animation where the model is not doing anything) correctly not pointed out (true negative), the number of pauses that the user identifies incorrectly as an activity in the model (false positive) and the number of activities made by the model and incorrectly ignored by the user (false negative). Multiple events pointed out by a user during the same activity or the same pause (true positive or false positive) are ignored. This has been made in order to not false the accuracy: in fact, counting multiple true positive or multiple false positive for the same instance of an activity would have introduced a bias.

In Figure 5, we present a cross-correlogram that shows the distribution of the delays between the pressing of the button by the user and the closest timestamp of the end of an activity. The distribution appears centered close to 0. This indicates that the participants actually recognized some movements in the model. Most of the delays are between -5 and +5 seconds from the closest activities. As the average length of activities is 3.043 sec (Table 1), a delay between -5 and +5 seconds is not enough to guarantee a correct segmentation.

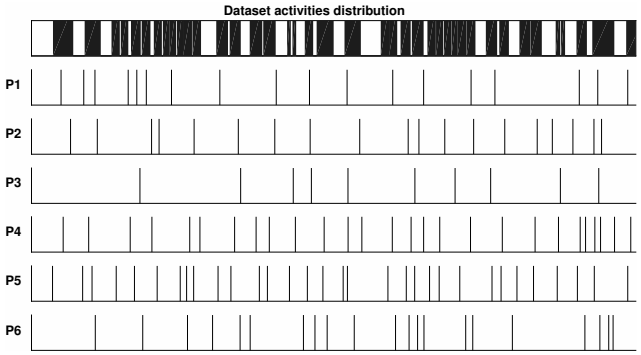


Figure 4: Segmentation results: this figure represents a subset of the dataset of the first experiment (ca. 3 minutes). The first row represents the distribution of the activities during this interval of time. The 6 rows below indicate the events pointed out by each participant are displayed.

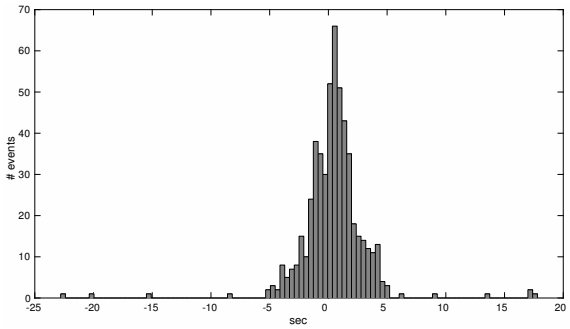


Figure 5: Cross-correlogram representing the distribution of the delays between the events pointed out by the users and the closest end of an activity. A positive delay means that the user pointed out an event after the closest end of the activity. Instead, a negative delay means that the user pointed out an event too early in respect to the closest end of the activity (i.e. before the activity is actually completed).

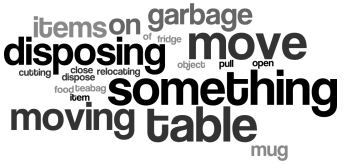
The second experiment aims to evaluate the application of our model in the "open-ended" scenario. In Figure 6 we show the tag clouds of the words entered to describe the activities by all users for each label. We noticed that most of the participants mistake the dishwasher with the oven: this is quite normal because both appliances can have the same kind of door. Moreover, the user interface is missing any rendering of the kitchen environment and no information is given to the user about the appliances and about the furnitures at this stage. However the participants correctly identified the difference between open and close the dishwasher. Interesting, "drinking" is correctly recognized by all the participants.

In the last experiment, we investigate the traditional scenario where the user should annotate a dataset already segmented, choosing the correct label from the predefined "closed set" of 11 activities indicated in Table 1. The results are presented in the Figure 7 using a confusion matrix between the choices of the users and the actual labels of the data.

Some activities are correctly recognized by the user: "Drink from Cup" reaches an accuracy close to 100%. On the other hand, there are actions that are almost never identified: "Open the Fridge" is the activity identified with the lowest accuracy (4.2%). The main cause is likely the absence of any point of reference for the environment and the fact that the fridge was of small size. For this reason, the action of opening and closing can be easily confused with other actions applied to the same height, such as "Open a drawer". Moreover, the confusion between "Open/Close the door" is due to the lack of information about the direction of opening and closing of the door.

(a) *Open door*

(b) *Close door*

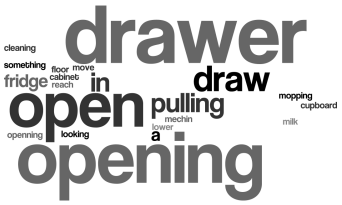


(c) *Open Fridge*

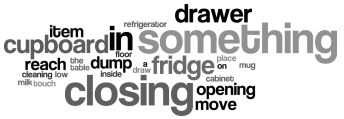
(d) *Close Fridge*

(e) *Open Dishwasher*

(f) *Close Dishwasher*



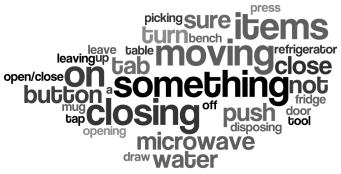
(g) *Open Drawer*



(h) *Close Drawer*

(i) *Clean Table*

(j) *Drink from Cup*



(k) *Toggle Switch*

Figure 6: Tag cloud of words used by testers in the open-ended experiment for each activity. The number of occurrencies of a word is shown by its size. The colors are only used to make more clear the distinction between the words.

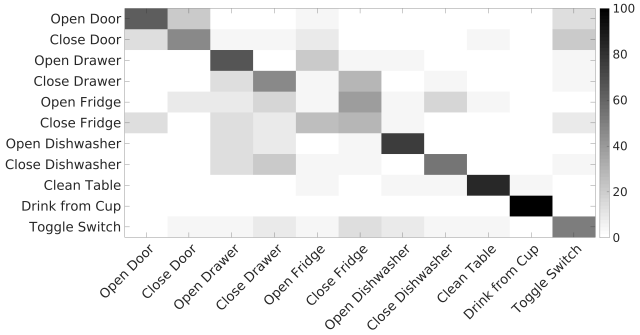


Figure 7: Confusion matrix between labels chosen by users and actual labels.

Finally, the participants reached an average accuracy of 56% in the controlled labelling experiment using our system.

Discussion

Our experiment revealed that a 3D human model can be used for activity annotation preserving the privacy of the user, but it would require some improvements.

About the segmentation of the activities, which is studied in the first experiments, the obtained results showed that further analyses are required such as the capability of point out the duration of the action. Allowing the users to identify the beginning and the end of an activity could improve the accuracy in this step. It can be also important to specify to the user the granularity of the action. As we noticed during the experiment, the main trouble for the testers was: "What should I consider as an action to point out?". Answering this question can depend on the specific application scenario of each dataset: in some cases an action could be a simple gesture as e.g. "move the right arm up". In others scenarios instead, it could be important to identify more complex actions as "make a sandwich", "prepare a coffee", etc.

Moreover, during our experiments, we pointed out that a main issue in the lack of the environment in the scene. It can be difficult for the users to recognize the wide set of possible activities without knowing the position of the objects and of the furniture in the environment. A typical example of this is the mistake between the opening of the fridge and the opening of the drawer: these two movements appear similar when reproduced with a simple model such as ours. This confusion between movements that appear similar is more observable in the "open-ended" annotation: in fact in this scenario it occurs that users identify correctly the movements (opening and closing), but they annotate the task with a different object whom those movements are applied to (the dishwasher mistaken with the oven).

Some improvements should be also applied to the model itself. In this first implementation, we used only basic solids to create the human figure: this brought some difficulties for the users to recognize actions made by short and limited movements. An example can be the toggling of the switch: in this case, the absence of the hands made it tricky to identify it. For this reason, we should explore whether a more realistic human model could improve the accuracy of the annotation.

In order to improve the accuracy of the movements performed by the model, a larger number of sensors can be a solution. In our experiments the data animate only two parts of each upper limb and the torso, but the model has been developed to be animated with a maximum of 12 sensors. The data from all these sensors can be also applied on the hands, on the legs and on the head. Furthermore, the software can be used with many different datasets passing specific parameters at start-up. The only require-

ment is that the dataset should contain IMU data for each body part the users want to animate. This can be a limitation: in fact, it can be difficult to use this system with those datasets already recorded and where the IMUs are placed only on few body parts, not allowing the model to reproduce all the movements correctly. Instead, this system can be a valid choice for new recorded dataset. If future work shows higher annotation accuracy, it might even be feasible for some scenarios to remove cameras altogether. This could lead to cost and time savings because researchers will not need neither equipments to record the videos nor additional time to preprocessing them.

It is the first time that annotation using a 3D model has been proposed. Even though an average accuracy of 56% may be insufficient for ground truth, this system could exploit decision fusion among multiple annotators (e.g.majority voting) and filters, to improve accuracy as already done in [9] for video annotation. In this scenario, it would be also interesting to study the performance of a classifier trained using the data annotated using these algorithms.

Conclusion

In this work we raise the need to create a privacy preserving annotation system, in order to speed up the process of labelling dataset using cheap labour and crowdsourcing. In these cases, a video can not be used due to lack of anonymity of the user recorded in the video itself.

We study to which extent a 3D human model animated in a virtual environment using the data from inertial sensors can be used to annotate the dataset. To test this we created and animated a model using a prior labelled dataset. We compared the annotation collected with our model and the actual labels of the dataset.

We performed three analyses: a first one to study the capability of the user to recognize the activities done by the model and correctly segment the dataset. This is effectively the first step during the annotation process. The second experiment is designed to analyze to which extent the application of our annotation system fits in a "open-ended" scenario where the user can choose freely the label for each action. In the last test we investigate the accuracy of the annotation when a set of possible choices are given to the users, reaching a high level of accuracy for some specific actions and manifold results for others.

From the experiments, the results are threefold. The segmentation experiment requires furter analysis in order to better evaluate to which extent a 3D human model can be actually used for this task. The obtained results are not sufficient to give a strong positive judgment. The "open-ended" annotation can be used but only when the dataset consist of actions that look very clear when reproduced by the model (e.g.drinking). For those actions where movements are limited and short, the accuracy with our system drops. Instead, using a "closed set" of annotations, our system allows users to reach an average accuracy of 56% also using very similar actions and with only 6 people. In this scenario, adding more people and applying decision fusion algorithms and filters for bad taggers, our system could become an actual choice to annotate data preserving the privacy of the subject in the dataset.

Acknowledgments

UK EPSRC Grant EP/N007816/1 "Lifelearn: Unbounded activity and context awareness"

REFERENCES

1. Amazon. 2005. Amazon Mechanical Turk. (2005). `https://www.mturk.com/mturk/welcome`, accessed 16/06/2016.

2. Akin Avci, Stephan Bosch, Mihai Marin-Perianu, Raluca Marin-Perianu, and Paul Havinga. 2010. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *Architecture of computing systems (ARCS), 2010 23rd international conference on*. VDE, 1–10.
3. Ling Bao and Stephen S Intille. 2004. Activity recognition from user-annotated acceleration data. In *Pervasive computing*. Springer, 1–17.
4. Michael Boyle, Christopher Edwards, and Saul Greenberg. 2000. The effects of filtered video on awareness and privacy. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 1–10.
5. Andreas Bulling, Ulf Blanke, and Bernt Schiele. 2014. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)* 46, 3 (2014), 33.
6. J. Dai, J. Wu, B. Saghafi, J. Konrad, and P. Ishwar. 2015. Towards privacy-preserving activity recognition using extremely low temporal and spatial resolution cameras. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 68–76. DOI: http://dx.doi.org/10.1109/CVPRW.2015.7301356
7. Susumu Harada, Jonathan Lester, Kayur Patel, T Scott Saponas, James Fogarty, James A Landay, and Jacob O Wobbrock. 2008. VoiceLabel: using speech to label mobile sensor data. In *Proceedings of the 10th international conference on Multimodal interfaces*. ACM, 69–76.
8. The jME core team. 2016. jMonkeyEngine. (2016). http://jmonkeyengine.org/.
9. Long-Van Nguyen-Dinh, Cédric Waldburger, Daniel Roggen, and Gerhard Tröster. 2013. Tagging human activities in video by crowdsourcing. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*. ACM, 263–270.
10. Gabriel Parent and Maxine Eskenazi. 2011. Speaking to the Crowd: Looking at Past Achievements in Using Crowdsourcing for Speech and Predicting Future Challenges.. In *INTERSPEECH*. Citeseer, 3037–3040.
11. Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, and others. 2010. Collecting complex activity datasets in highly rich networked sensor environments. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*. IEEE, 233–240.
12. Tim Van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. 2008. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 1–9.
13. Carl Vondrick, Donald Patterson, and Deva Ramanan. 2013. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision* 101, 1 (2013), 184–204.
14. Aobo Wang, Cong Duy Vu Hoang, and Min-Yen Kan. 2013. Perspectives on crowdsourcing annotations for natural language processing. *Language resources and evaluation* 47, 1 (2013), 9–31.
15. XSens. 2000. MTx 3D Tracker. (2000). https://www.xsens.com/products/mtx/

Poster: BlueSense - Designing an extensible platform for wearable motion sensing, sensor research and IoT applications

Daniel Roggen
University of Sussex
daniel.roggen@ieee.org

Arash Pouryazdan
University of Sussex
a.pouryazdan@sussex.ac.uk

Mathias Ciliberto
University of Sussex
m.ciliberto@sussex.ac.uk

Abstract

We present an extensible sensor research platform for wearable and IoT applications. The result is a 30x30mm platform capable of 500Hz motion and orientation sensing using 98mW when logging the data. The platform can wake up at programmed intervals using only 70uW in hardware off mode. A maximum 0.6ppm time deviation between nodes allows usage in a network for whole body movement sensing.

1 Introduction

Our work is motivated by sensor-based human activity recognition which is key to smart-assistive systems. It addresses issues we experienced in prior work collecting large scale datasets for human activity recognition [7] and takes into account experiences reported by other researchers. The key observations are: i) some applications require real-time recognition and thus data streaming, whereas others perform offline analysis which requires data logging; ii) multiple sensors are generally improving recognition performance, thus their recordings must be synchronised [4]; iii) using limb coordinates instead of raw motion data is well suited for fine gesture recognition, which thus requires orientation sensing [8]; iv) some applications require high sample rate, especially in sports [2]; v) activity recognition can benefit from novel sensors [5], and thus a platform should be extensible.

A secondary motivation is measurements over extended periods of time at low sample rate (e.g. once per day), which is common in Internet of Things (IoT) applications. Instead of hardware event detectors [9] we combine true hardware off with programmed wake-up through a real-time clock.

2 Hardware

The platform (fig. 1) comprises an ATmega1284p microcontroller at 11MHz, 3V regulator and LiPo battery charger, Micro SD slot, a single-chip 3D accelerometer, gyroscope and magnetometer (MPU9250), a coulomb counter, and USB and Bluetooth 2 interfaces. Classic Bluetooth allows enough bandwidth for real-time analytics. It has a one of the highest accuracy real-time clock (RTC) on the market (DS3232M), with ± 5 ppm accuracy over the entire temperature range. It is used to timestamp the recordings of independent nodes. We measured the drift of the RTC to be less than 0.6ppm when nodes are at room temperature. Overall, the platform is 30x30mm. It accepts extension boards on top or bottom (fig. 2). One connector comprises the programming interface, SPI interface, regulated power, USB power, two GPIO which can also be used as ADC inputs, and an open-drain line which can be used to wake up the system from hardware off. This may be used to implement event detectors using low-power analog circuitry [9]. The other connector comprises the I2C interface, analog and battery power and 5 GPIO pins, 3 of which can be used as ADC inputs.

True hardware off is achieved by turning off the power regulator (LTC3553 in fig. 1). The system can wake up from this mode when the power button is pressed, when a real-time clock alarm occurs, or when a pin on the expansion connector is pulled low. The logic to wake up the system (power logic in fig. 1) is powered by the battery directly.

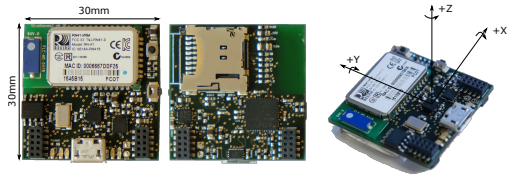


Figure 2. PCBs and node fitted with a 160mAh battery. Larger batteries can be employed if needed.

3 Firmware

The firmware offers a terminal interface over USB and Blueooth to setup the node and start/stop data acquisition. We designed the firmware to achieve high sample rate with low jitter. No operating system is used to minimise overheads. However a comprehensive library abstracts the user application from the hardware details. Most I/O library functions rely on interrupt routines to receive or send data from or to a peripheral. The interrupt routines stores or reads the data from memory buffers to which the library functions called from user code can also access. The SD card interface however is not interrupt driven and SD card writes are blocking.

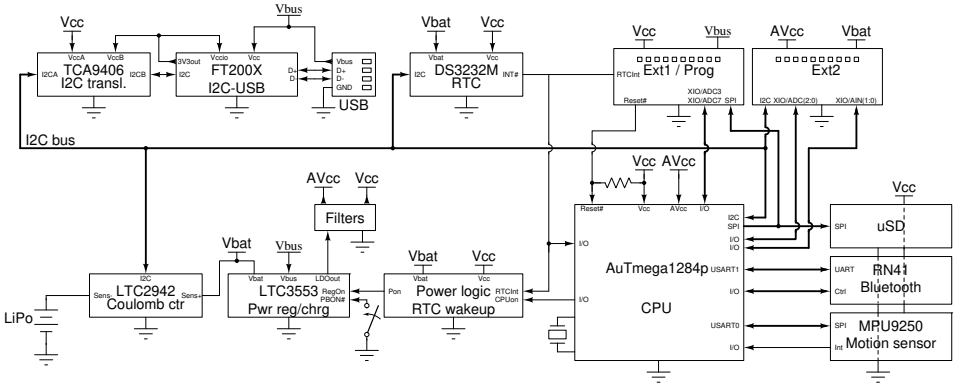


Figure 1. Extensible sensing platform for wearable and IoT applications.

Table 1. Power use in various modes.

Logging		Streaming		Idle		
500Hz	100Hz	500Hz	100Hz	No conn.	BT conn.	Off
98mW	94mW	200mW	184mW	18mW	92mW	70μW

Data logging uses an optimised FAT32. Log files are pre-allocated contiguously on the SD card. This allows to employ SD card "pre-erase" and "multi block writes" commands which allow streaming writes of data, without having to regularly update the FAT entries and cluster link. Only when a file is closed is the FAT updated to reflect the size of the file. With this we achieved 1KHz ADC sample rate with jitter less than $\pm 30\mu S$ [5].

The motion sensor data is converted into orientation quaternions using a variation of Madgwick's algorithm [3], where the corrective step using the accelerometer and magnetometer is carried out at a fixed 12.5Hz. This allows to keep the computation time below $1100\mu S$ and is instrumental to achieve 500Hz motion sensing. We did not observe adverse effects thanks to the low noise of the gyroscope.

Timekeeping is obtained from a combination of an internal AVR timer and the RTC. The AVR timer provides a time resolution of 1ms. A 1Hz RTC signal is used to regularly reset the AVR timer. This ensures that the timekeeping error is bound by the RTC timekeeping accuracy.

4 Characterisation

We minimised CPU power consumption by sleeping the processor when busy-looping. In idle mode (i.e. waiting for commands), the dominant power contribution is the Bluetooth radio, which we minimised by modifying the inquiry and page scan window and duty cycling. This decreased idle power by 19mW at the expense of slightly longer discovery and connection time. In hardware off mode, the only components directly powered by the battery are the coulomb counter, the RTC and the power-up logic. The type of SD card used has a significant influence on power use during logging. Table 1 shows power use with a 32GB Samsung EVO+ SD card; using a 32GB SanDisk Extreme instead increased power use by 40mW.

5 Conclusion

BlueSense offers a better tradeoff and versatility for wearable sensing applications compared to many other platforms. It is smaller at 30x30mm than commercial solutions by Xsens (47x30mm for the wireless MTw), Shimmer (51x34mm for the Shimmer3 IMU) and x-io technologies (42x33mm for the x-IMU) and it offers higher sample rate (500Hz) than the XSens MTw (120Hz) and many other platforms [6], inluding highly miniaturised ones [1]. It is extensible, as is the x-IMU. True hardware off also allows usage in IoT applications. The bill of material is below £80 per unit (excluding assembly) in batches of 30.

An AVR processor was used to reduce development time by exploiting our large existing code base. The firmware development time was nevertheless significantly underestimated due to the highly specific needs of this platform requiring a large number of new software modules. A lesson learned for embedded systems development is that a more modern microcontroller could have been used (e.g. an ARM Cortex-M) while incurring only a very limited increase in development time. The platform will be open-hardware[1].

6 References

[1] H. Harms et al. ETHOS: Miniature orientation sensor for wearable human motion analysis. In *IEEE Sensors*, pages 1037–1042, 2010.

[2] M. Lapinski et al. A distributed wearable, wireless sensor system for evaluating professional baseball pitchers and batters. In *Proc Int Symp on Wearable Computers*, pages 131–138, 2009.

[3] S. Madgwick et al. Estimation of IMU and MARG orientation using a gradient descent algorithm. In *IEEE Int Conf on Rehabilitation Robotics*, 2011.

[4] S. Münzner et al. CNN-based sensor fusion techniques for multimodal human activity recognition. In *Proc Int Symp Wearable Computers*, pages 158–165, 2017.

[5] A. Pouryazdan et al. Wearable electric potential sensing: a new modality sensing hair touch and restless leg movement. In *Proc. ACM Int Joint Conf on Pervasive and Ubiquitous Computing: Adjunct*, pages 846–850, 2016.

[6] D. Rodríguez-Martín et al. A wearable inertial measurement unit for long-term monitoring in the dependency care area. *Sensors*, 13(10):14079–14104, 2013.

[7] D. Roggen et al. Collecting complex activity data sets in highly rich networked sensor environments. In *7th Int. Conf. on Networked Sensing Systems*, pages 233–240. IEEE Press, 2010.

[8] T. Stiefmeier et al. Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing Magazine*, 7(2):42–50, 2008.

[9] F. Sutton et al. The design of a responsive and energy-efficient event-triggered wireless sensing system. In *Proc Int Conf on Embedded Wireless Systems and Networks*, pages 144–155, 2017.

[1] http://github.com/droggen/BlueSense2

Demo: Complex Human Gestures Encoding from Wearable Inertial Sensors for Activity Recognition

Mathias Ciliberto
University of Sussex
m.ciliberto@sussex.ac.uk

Luis Ponce Cuspinera
University of Sussex
l.ponce-cuspinera@sussex.ac.uk

Daniel Roggen
University of Sussex
daniel.roggen@ieee.org

Abstract

We demonstrate a method to encode complex human gestures acquired from inertial sensors for activity recognition. Gestures are encoded as a stream of symbols which represent the change in orientation and displacement of the body limbs over time. The first novelty of this encoding is to enable the reuse previously developed single-channel template matching algorithms also when multiple sensors are used simultaneously. The second novelty is to encode changes in orientation of limbs which is important in some activities, such as sport analytics. We demonstrate the method using our custom inertial platform, *BlueSense*. Using a set of five BlueSense nodes, we implemented a motion tracking system that displays a 3D human model and shows in real-time the corresponding movement encoding.

1 Introduction

Inertial sensors and template matching algorithms have been used successfully for activity recognition in healthcare, well-being and sports applications [2]. Template matching algorithms can be embedded on low-power sensor nodes [6]. Nevertheless, they are generally designed to work with a single channel of data. In certain situation, such as in beach volleyball movements analytics, this can be a limitation as multiple sensors are required to be employed on different body parts in order to get and analyse the complexity of the movements. For this reason, using them can become challenging for complex gestures recognition.

Modern inertial platforms, such as XSens [5], Ethos [4] and our BlueSense, can provide orientation data, generally as quaternions. We present an encoding approach for complex gestures that elaborate the orientation data provided by several inertial sensors worn by the user. This method encodes the position and the orientation of the user's hand during a movement as a single stream of symbols, simplifying the application of single channel pattern matching algorithms. It is an extension of [7] with the novelty of including the rotation of the hand during the movements in the encoding. This will be important in future applications, such as sport analytics and specially in beach volleyball gesture recognition.

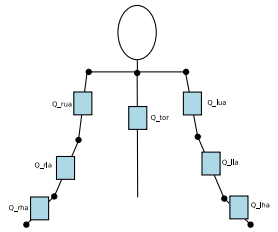
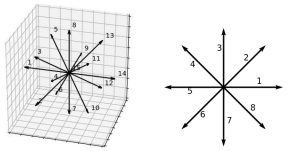


Figure 1. Setup of BlueSense on the user's upper body.



(a) *3D, 15 symbols* **(b)** *2D, 8 symbols*

Figure 2. 3D and 2D codebooks examples. Different numbers of vectors can be used in each codebook, in order to reduce or increase the granularity of the displacement sampling.

2 Gesture Encoding

The system described in [7] computes the position of the upper body joints using the 3D orientation of sensors placed on each limbs and the torso of the user, as displayed in Figure 1. Combining these positions, the algorithm finds the position of the hand in the 3D space. A gesture is then expressed as successive positions of the hand forming a trajectory. Then, the trajectory can be sampled at regular time intervals or after that a certain distance has been covered. For each sample, the vector difference between two contiguous positions is calculated. This vector is finally encoded to a symbol using a codebook: this is a set of 3D unit vectors equally distributed with respect to their direction (Figure 2). The symbol corresponding to the displacement vector is given by the closest codebook vector. The coded symbols are indexes in the codebook.

The method as described in [7] lacks of information about the rotation of the hand during the movement. Two different gestures can lead to the same displacement encoding, for ex-

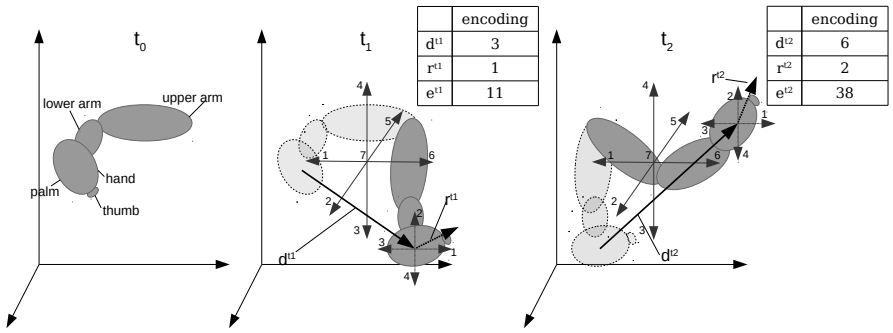


Figure 3. Example of encoding of two movements of a right arm, from t_0 to t_1 and from t_1 to t_2. At every instant t_i, the displacement of the hand is encoded with the d^{ti}, the rotation of the hand is coded with r^{ti} and the final encoding, computed using the Cantor pairing function, is represented by the symbol e^{ti}.

ample if they have been performed once with the palm facing upwards and another time with the palm facing downwards. In order to overcome this issue, we introduced a second encoding for the rotation of the hand.

This extra encoding uses a 2D codebook (Figure 2) in order to represent the rotation of the hand. Defining a starting position encoding (for example the palm facing downwards parallel to the ground is encoded as *1*), it is possible to represent the rotation of the hand with the closest symbol in term of angular distance. The two symbols for position and rotation of the hand are eventually combined in order to obtain a single symbol. As natural numbers are used to index the codebooks vectors, this step is performed using a pairing function. An example, the Cantor pairing function, is presented in 1:

$$e(d,r) := \frac{1}{2}(d+r)(d+r+1)+r \qquad (1)$$

where d is the symbol for the displacement, r is the symbol for the rotation of the hand and e is the final encoding. The addition and product operators are the arithmetical operations of sum and multiplication.

An example of the whole system can be observed in Figure 3. A 7-symbols codebook for the displacement and a 4-symbols codebook for the rotation are used.

3 Demonstration

In order to visualize the data provided by BlueSense nodes, we created a motion tracking system which is presented in Figure 4. The system includes a set of 4 BlueSense sensor nodes and two programs: SensHub and the 3D human model. The former collects the orientation data through Bluetooth from the sensors, synchronizes them and forwards them to the 3D model as single line of text through a TCP connection. The latter is a 3D human model developed using the JMonkeyEngine 3D engine [1]. The TCP connection allows to place SensHub and the 3DHumanEngine on two different devices, potentially in two different locations.

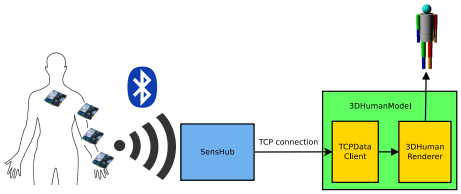


Figure 4. Motion tracking system overview. The 3DHumanModel receives the data as a line of text through a TCP connection. Then it parses this text in order to extract the quaternions to animate the 3D model. At every rendering cycle, the 3D model is updated with the most recent orientation data.

We were able to evaluate of the 3D model and the motion tracking system during the British Science Festival 2017 [3]. During the event, we deployed the system in a more complex simulation where people where asked to play a virtual beach volleyball game. We analysed the latency and the battery life of the sensors. The sensors are able to stream quaternions up to 500 Hz, but considering the framerate of the 3D rendering set to 60fps, we set the sample rate to 100 Hz. The latency is highly related to the hardware of the PC that runs the simulation. During the event it was acceptable for real time gaming. We also tested the battery life of the sensors that streamed the data for about 4 hours continuously.

In the future, we plan to employ the motion tracking system to support the training of beach volleyball players.

4 Acknowledgements

U.K. EPSRC First Grant EP/N007816/1

5 References

[1] Jmonkeyengine, 3d game engine. `http://jmonkeyengine.org/`, 2017. [Online; accessed 28-November-2017].

[2] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. Activity Recognition Using Inertial Sensing for Healthcare, Wellbeing and Sports Applications: A Survey. *Architecture of computing systems (ARCS), 2010 23rd international conference on*, pages 1–10, 2010.

[3] M. Ciliberto. "How good are you in beach volleyball?". `https://www.britishsciencefestival.org/event/how-good-are-you-at-beach-volleyball/`, 2017. [accessed 28-November-2017].

[4] H. Harms, O. Amft, R. Winkler, J. Schumm, M. Kusserow, and G. Tröster. Ethos: Miniature orientation sensor for wearable human motion analysis. In *Sensors, 2010 IEEE*, pages 1037–1042. IEEE, 2010.

[5] D. Roetenberg, H. Luinge, and P. Slycke. Xsens mvn: full 6dof human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep*, 2009.

[6] D. Roggen, L. P. Cuspinera, G. Pombo, F. Ali, and L.-V. Nguyen-Dinh. Limited-Memory Warping LCSS for Real-Time Low-Power Pattern Recognition in Wireless Nodes. *European Conference on Wireless Sensor Networks*, 8965:151–167, 2015.

[7] T. Stiefmeier, D. Roggen, and G. Tröster. Gestures are strings: efficient online gesture spotting and classification using string matching. *Proceedings of 2nd International Conference on Body Area Networks*, pages 16:1–8, 2007.

A Case Study for Human Gesture Recognition from Poorly Annotated Data

Mathias Ciliberto
Wearable Technologies Lab
Sensor Technology Research Centre
University of Sussex, UK
m.ciliberto@sussex.ac.uk

Daniel Roggen
Wearable Technologies Lab
Sensor Technology Research Centre
University of Sussex, UK
daniel.roggen@ieee.org

Lin Wang
Wearable Technologies Lab
Sensor Technology Research Centre
University of Sussex, UK
w23@sussex.ac.uk

Ruediger Zillmer
Unilever R&D Port Sunlight
ruediger.zillmer@gmail.com

UbiComp/ISWC'18 Adjunct, October 8–12, 2018, Singapore, Singapore

ACM ISBN 978-1-4503-5966-5/18/10...$15.00
DOI: https://doi.org/10.1145/3267305.3267508

Abstract
In this paper we present a case study on drinking gesture recognition from a dataset annotated by Experience Sampling (ES). The dataset contains 8825 "sensor events" and users reported 1808 "drink events" through experience sampling. We first show that the annotations obtained through ES do not reflect accurately true drinking events. We present then how we maximise the value of this dataset through two approaches aiming at improving the quality of the annotations post-hoc. First, we use template-matching (Warping Longest Common Subsequence) to spot a subset of events which are highly likely to be drinking gestures. We then propose an unsupervised approach which can perform drinking gesture recognition by combining K-Means clustering with WLCSS. Experimental results verify the effectiveness of the proposed method.

Author Keywords
Gesture recognition; dataset annotation; activity discovery; dataset curation

ACM Classification Keywords
H.3.m [Information storage and retrival]: Miscellaneous

Introduction

Gesture recognition has applications in several fields such as healthcare and sports [7]. In order to create a reliable gesture recognition system, it is important to have a well-annotated dataset [1]. However, creating high-quality datasets may require to rely on lab-like naturalistic environments, with limited ecological validity [11]. Activity recognition research generally strives to employ datasets with unrealistically "perfect" ground truth annotations. In an ecologically valid data collection, however, it is likely that a highly valuable dataset is acquired, but that only poor quality annotations are available.

Experience sampling (ES) is a real-time annotation approach done by users themselves a mobile device [13]. This allows more ecologically valid data collection in everyday life (e.g. no need to video record the experiment). However, ES can lead to the following issues: i) the synchronisation between the activity performed and the label annotated by the user is generally of poor quality, with the user annotating the activity after the event, or combining multiple activities in a single annotation; ii) the user may forget to label an event, iii) the user may annotate an activity with the wrong label.

In this work, we investigate how to make sense of a dataset with high business value, which has been annotated through ES, which led to numerous deficiencies in the annotation quality. The dataset contains drinking gestures annotated by the users with a mobile application. The dataset was collected in an office environment using a 3-axis accelerometer and it is made by 8825 "sensor events", with 1808 "drink events" annotated by users through ES. Using this dataset, we aim to address two main challenges: i) to understand why the quality of the annotation is low and consequently how would it be possible to improve in future data collection and ii) to understand whether it is still possible to use such big dataset without relying on the annotations for spotting drinking gestures and how. The contributions of this work are:

- A study of the annotations. We analyse the user annotations, their distribution in time during the data collection, and their relation to the sensor events, in order to understand the causes of the low quality and where the data collection process can be improved.
- A template matching approach, based on Warping Longest Common Subsequence (WLCSS) [8], to extract a subset of drinking gestures, within a certain level of confidence. This subset will allow the dataset to be used for research purposes.
- An unsupervised algorithm (K-Means) adapted to template matching. This algorithm is a new variation of K-Means [4] where the WLCSS is used as distance measure. It allows to cluster gestures based on the raw signal of the sensors. At the same time, it clusters gestures taking in account the variation in the way they can be performed, by using WLCSS which has been successfully used for robust gesture detection [8].

Related work

The quality of annotations obtained through ES can be poor [13]. Annotations issues can include time shift of a label with respect to the activity, as well as wrong or missing labels [9].

Some approaches suggested to improve ES with manual re-annotation [13]. This is not feasible economically for a large number of users and however, in [13] the quality of

the annotations was still not sufficient for the training of machine learning algorithms. The impact of ES on activity recognition has been studied in [2]. However, the authors simulated the ES in a controlled environment and they used only the data corresponding to the user annotations.

The problem of poorly labelled data can be tackled during the annotation itself or during the training of the machine learning algorithm. A method useful to reduce the effort of the users while annotating their activity has been proposed in [10]. The authors proposed a one-time point annotation method that requires the users to only label a single moment per activity rather than specifing the beginning and the end. The method then recognizes automatically the boundary of the activity in the annotated signal. Nevertheless, it requires that the labels are within the execution interval of the activity.

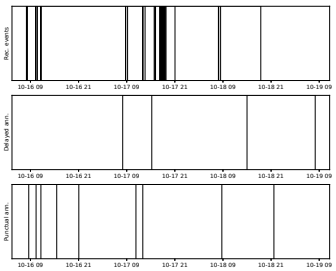
Unlabelled or poorly labelled data are available in big quantities nowadays due to the large diffusion of sensing devices, such as smartphones and wearables devices. For this reason, methods such as semi-supervised learning, active learning and unsupervised learning have been applied in order to extract useful information from sparsely annotated data. A combination of active learning and semi-supervised learning has been studied in [12]. The authors used a dataset of daily activities collected with two subjects wearing accelerometer sensors and motioned tracked with infrared sensors. This approach however uses a decision window of 30 seconds long and thus is not suitable for recognizing gestures that occur in a short time. Unsupervised learning has been successfully applied to activity recognition in [5] and more recently in [3]. In the latter, an activity discovery method based on clustering is proposed to help with ES, although it is designed for periodic movements rather than sporadic gesture. Unsupervised learning has been applied to gestures clustering in [14], where a K-Means clustering has been evaluated specifically for hand gestures.

Several studies have tried to address the challenge of activity recognition from poorly annotated data. While most of them used synthetic dataset and focused on periodic or long activity (such as walking, running, etc.), to the best of our knowledge none of them applies to drinking gestures collected in a real-life office environment.

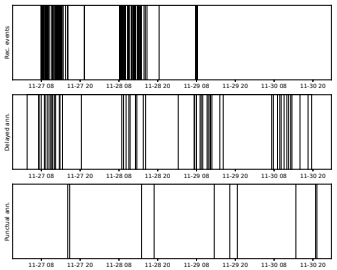
Dataset

The dataset was collected by providing a set of mugs to 60 users (one mug per user) in an office environment. Each user collected data for a period of 4 days. Each mug was instrumented with a logger comprising a 3-axis accelerometer [15]. The loggers were placed in a hollow at the bottom of each mug. As the mugs were customly made, the positioning of the loggers was not the same in all mugs. The loggers sample acceleration at 20 Hz, with a timestamp in ms. In order to save power, they start logging acceleration when a movement is detected. After 5 seconds of inactivity they automatically stop the recording, without record the inactivity period. We use the term *sensor events* to refer to every recording performed by the loggers that lasts at least 4 seconds (as configured on the loggers for this data collection). Therefore, sensor events can occur for a variety of reasons: moving the mug on the desk, washing it, drinking from the cup, etc.

The data annotation was performed through experience sampling by the users themselves. They labelled each drinking event manually using an Android application installed on their smartphones. Each annotation could be *punctual* or *delayed*. An annotation is considered punctual when it was entered immediately after the drinking event.



(a) *User 109*



(b) *User 461*

Figure 1: Example of annotations of two different users, over the 4 days period. The start time of the sensor events are displayed in the top plot of each figure, one thin line per event. The delayed and punctual annotations inserted by the users are displayed respectively in the second and the third plot of both figures. The X-axis reports date and time, in the format "MM-DD HH". It is also possible to notice the differences in the way two users annotated the drinking events.

It is considered delayed, when it refers to an event in the past. The users could specify in the application whether their annotation was punctual or delayed. However they did not have to provide an indication as to how much the delay was. Furthermore, there were no guidance indicating after how much time an annotation should be considered "delayed" rather than "punctual".

The resulting dataset is made by 8825 sensor events, 1808 user annotations, of which 1477 marked as "punctual" and 331 as "delayed". The percentage of annotated gestures with respect to the total amount of sensor events is of 20.5%.

User annotation analysis

We aim to analyse the causes of the poor annotations in order to improve future data collections, as well as helping during the next steps of this study.

Figure 1 indicates the main challenge of the annotation protocol, which is how users understood differently how and what to annotate. The data collection protocol did not require participants to annotate drinking solely when using the instrumented mugs: they could annotate drinking as well when using regular mugs. It might happen that users annotated drinking events performed using other cups. The protocol did also not specify what to consider as a "drinking event". Users could interpret it as referring to a single sip, multiple sips, or drinking the entire cup. It is also possible to notice how the annotations are not perfectly aligned with the sensor events.

We also studied the distribution of the labels, per user, over the 4 days of data collection. It could help to understand the users' commitment in annotating their drinking gestures, assuming they were keeping the same drinking habits among all the days. This may be useful in order to spot days for which the annotations can be more reliable. The results are presented in Figure 2. While there is no significant change between day 1 and day 2, with an average increase in the number of annotation of 0.24%, starting from day 3 the engagement decrease by 11% on average among all the users. The plot shows also a great variability in the data: there were users that increased their commitment over the 4 days, as well as users for which the commitment decreased over the 4 days.

From the analysis of the annotations, it can be concluded that they were not reliable enough to be used together with the data for the supervised classifier training.

Gesture classification

In order to make the collected dataset useful for drinking gesture recognition, each event recorded by the sensors had to be classified in drinking/non-drinking. As highlighted previously, the users annotations cannot be used as-is as they are not accurate enough. A manual relabelling of the entire dataset was unfeasible given the lack of any video recordings.

We developed an approach based on a template matching method (TMM) to automatically spot a subset of events which are believed to be drinking gestures with a certain confidence value. The approach then uses few events which are manually identified as drinking events with high confidence to train the TMM.

Data processing and training set selection

We used a heuristic method to select a few sensor events as the training set. We performed a few drinking gestures using the same instrumented mug and discovered that the Z-axis of the accelerometer quite clearly indicates the gesture of lifting the cup to drink. A template of such gesture is displayed in Figure 3. A subset of gestures visually sim-

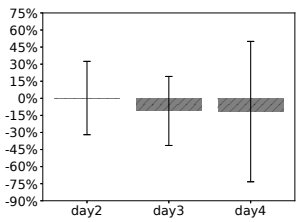


Figure 2: Change in the user commitment in the annotation during day 2, 3, and 4. The gray bars represents the percentage of annotations for each day with respect to day 1. Day 2 displays an increment of 0.24%; Day 3 and 4 an average decrease of 11% in the number of annotations. The vertical bars represents the standard deviation for each day.

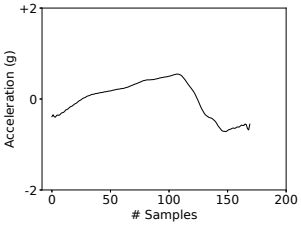
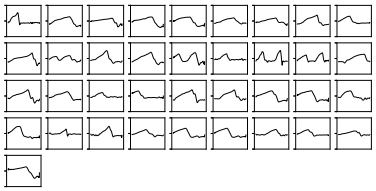
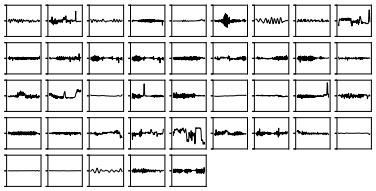


Figure 3: Template of a drinking gesture, performing a single sip.



(a) *Drinking gestures*



(b) *Non-drinking gestures*

Figure 4: Training set of gesture. 4a displays the templates chosen as drinking gestures. 4b shows those selected as no-drinking gestures. All the plots show the templates downsampled to the fixed length of 170 samples (X-axis). The Y-axis represents the acceleration within a range of $\pm$ 2g.

ilar to this template was selected manually from the entire set of the available gestures. We selected this subset trying to include some variability in the way the drinking gestures were performed. Another subset of non-drinking gestures was selected too, chosing the templates that were very visually different from the drinking gestures. The training set is displayed in Figure 4. It is formed by 78 events: 37 drinking gestures (4a) and 41 non-drinking gestures (4b).

All the instances in the dataset were filtered using a Butterworth low pass filter with cut off frequency set to 10 Hz. They were also resampled to a fixed number of samples. The number of samples was selected as the average length of a drinking gesture, which is 170. This step was performed in order to reduce the impact of non-drinking events that can last longer time than drinking gestures (e.g. washing the cup, moving the cup around the office, etc.).

Template matching using WLCSS

The Warping Longest Common Subsequence (WLCSS) [8] is an algorithm developed for template matching in real-time applications. Using dynamic programming, the algorithm can compute a matching score between a template and a stream, updating it at every new sample of the stream. It can be used for gesture recognition as it can handle gestures performed with variation in their speed of execution. This is achieved by three parameters: reward (R), penalty (P) and acceptance distance (ϵ). The algorithm is shown in (1).

$$M(i,j) = \begin{cases} 0 & \text{if} \quad i \leq 0 \quad or \quad j \leq 0 \\ M(j-1,i-1) + R & \text{if} \quad |S(i) - T(j)| \leq \epsilon \\ max \begin{cases} M(j-1,i-1) - P \cdot |S(i) - T(j)| \\ M(j-1,i) - P \cdot |S(i) - T(j)| \\ M(j,i-1) - P \cdot |S(i) - T(j)| \end{cases} & \text{if} \quad |S(i) - T(j)| > \epsilon \end{cases} \tag{1}$$

The matching score M(i,j) is computed as function of the previous scores, by adding a reward (R) when the distance between the i-th stream sample ($S(i)$) and the j-th template sample ($T(j)$) is below an acceptance distance (ϵ), or by subtracting a penalty (P) proportional to the distance, when this is above ϵ. In addition to R,P, and ϵ, WLCSS needs a threshold T. As WLCSS computes a matching score (M) between an instance in the dataset and a template, T is used to define whether an instance matches with the template (M$\geq$T) or not (M$<$T). The value of the threshold is related to the specificity and the sensitivity of the matching algorithm: a high threshold means high specificity, while a low

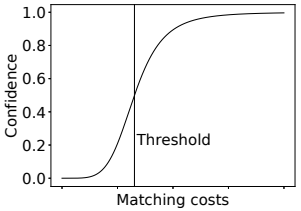


Figure 5: 4 Parameter Logistic Regression function used to assign a confidence with the respect to the threshold.

threshold means high sensitivity. The values of R,P,ϵ and T must be found during the training phase. We optimized this based on an evolutionary optimization technique.

WLCSS optimization using evolutionary algorithm
We optimise the values of R, P, e, T to maximise the ability of WLCSS to distinguish drink from non-drink using an evolutionary algorithm (EA). We used the EA in order to optimize the values of the parameters starting from a randomly generated population. Each individual of the population is an array containing the 4 parameters. The EA evolves this population through the usual selection, mutation and crossover operators [6]. Here, the F1 score is used for the selection. The optimization process stops after a predefined number of iterations, in this case .

Confidence computation
Given the unreliability of the labels in the dataset, it is not possible to evaluate precisely the correctness of a match for this particular dataset. For this reason, we opted to provide a confidence level for each gesture as output of our method rather than a simple match/no-match. The confidence level was assigned using Four Parameter Logistic Regression:

$$y = d + \frac{a-d}{1+(\frac{M}{T})^b}$$

where y is the confidence level, M is the matching cost, and T is the threshold. The function, displayed in Figure 5, provides a confidence value in the range [0:1]. The range is defined by the parameters $a = 0$ and $d = 1$. The parameter b, which define the slope of the function curve, was set manually to 5. Using a fixed interval for the confidence makes its value unrelated from the absolute value of T, which can vary according with the parameters R, P and ϵ.

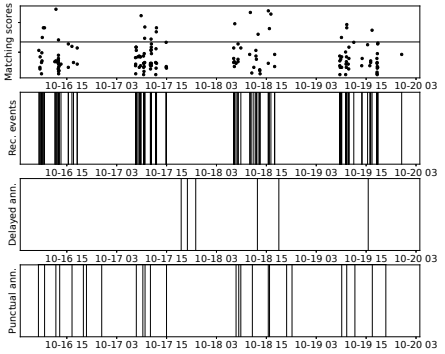


Figure 6: Comparison of WLCSS matching scores with recorded data for a single user. From the top: the WLCSS matching scores and the threshold T, as horizontal line (first plot), the start time of the sensor events (second plot), and the user annotations delayed and punctual (respectively third and fourth plot). The data are for 4 days period, with the X-axis reporting date and time in the format "MM-DD HH"

Evaluation
We trained the system using the EA and the subset of instances selected as training set. As the EA is a stochastic process, we repeated the training 10 times, and we picked the best values of R=68, P=0, ϵ=28 and T=3364 for the WLCSS. With these values, we run the algorithm on the entire dataset, by using the template displayed in Figure 3, which was selected manually from the training set as template.

Figure 6 displays a comparison of the sensor events and the annotations for a single user, with the corresponding matching scores. In the figure, it is also reported the threshold: the matching costs $>= T$ are those which are detected as drinking gesture with a confidence $\geq$50%. The percentages of total detected gestures compared to the total number of events are displayed in Table 1, for different confer-

Table 1: Number of drinking gestures detected for some confidence levels. The percentages are with the respect to the total number of sensor events in the dataset (8825).

Confidence	# gestures	%
$\geq$ 25%	1481	17%
$\geq$ 50%	942	10%
$\geq$ 75%	543	6%

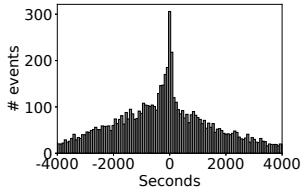


Figure 7: Cross-correlogram representing the distribution of the delays (in seconds) between the user annotations and all the events recorded by the loggers.

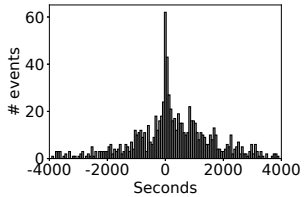


Figure 8: Cross-correlogram representing the distribution of the delays (in seconds) between the user annotations and the events detected with a confidence $\geq$50%.

ence values. The low percentages are due to the nature of the sensors, which were collecting all sort of movements such as moving the mug on the desk, washing it or even accidental movements. It is important to note that the number of detected events is also lower than the number of user annotations (1808). This may be a result of the data collection protocol which did not specify to annotated only the drinking movements performed with the instrumented mug.

We studied the relation between user annotations, sensor events and detected gestures. To do this, we assigned to every recorded event the closest user annotation in time. Then, computing the time difference between the sensor events and the corresponding closest annotations, we created the cross-correlogram displayed in Figure 7. Figure 8 presents the distribution of the same time differences, but considering only the gestures detected by WLCSS with a confidence $\geq$50%. The latter plot presents a more pointy distribution, confirming that WLCSS detected events that were actually closer in time to the user annotations.

Unsupervised learning

We evaluated also an unsupervised approach in order to classify the gestures in drinking/non-drinking as it does not require a training set. We developed a custom method based on K-Means. We modified K-Means in order to make it able to cluster gestures performed with variation in their speed of execution.

K-Means with WLCSS

K-Means is a clustering technique that aims to partition n observation in k clusters. Each observation belongs to the cluster with the closest mean. It can be used for unsupervised learning by clustering the input data based on a distance measure. The algorithm is based on two steps, assignment step and update step, which are repeated until a stopping criteria is met [4]. This criteria can be reaching a maximum number of iterations, the change of the clusters in the update step in below a thresholds, etc. We implemented a modified version of the K-Means, where WLCSS was used a distance measure in place of the Euclidean distance. The assignment in our implementation is modified as following:

$$\arg\max_{c_i} WLCSS(x, c_i)$$

where x is a sensor events, c_i is the centroid for the i-th cluster. The function *argmin* is replaced by *argmax* as WLCSS compute a matching score rather than a distance. The update step is unmodified.

Evaluation

We compared our version of K-Means (named K-Means-WLCSS) against the standard version that uses the Euclidean distance in order to assign each instance to the closest cluster. We applied both the implementation on the training set from the previous step, with $k = 2$ as the goal was to distinguish between drinking and non-drinking gestures. As all the instances were resampled to the same length, they could be used as feature vectors for both the implementations, without dealing with different lengths of the feature vectors (in this case the resampled raw signal). Applying the algorithms on the training set allowed us to compare the clustering results with the labels assigned manually to each gesture, during the data selection. Figure 9 presents a visual comparison between the clusters obtained with the two versions of K-Means. For both the implementations, Cluster 1 seems to include mainly the drinking gestures, while Cluster 2 the non-drinking gestures. We used this consideration in order to evaluate the performance of the two algorithms computing precision, recall and F1 score, presented in Table 2. They are computed by comparing the clustering of K-Means with the manual labels

Table 2: Precision, Recall and F1 score for K-Means and K-MeansWLCSS computed on the training set. They are computed using the manual labels assigned to each instance in the training set as ground-truth. The majority of the gestures in a cluster is used as classification label for the K-Means implementations.

	K-M	K-M_WLCSS
Precision	100%	92.11%
Recall	62.16%	94.6%
F1	77%	93%

of the instances in the extracted subset. K-Means_WLCSS increased the F1 score by 16%, being able to detect more variations in the drinking gestures as it is also visible from Figure 9. It was able to cluster correctly drinking gestures composed by two sips, such as the instances 8, 16, and 21 of Figure 9c.

Discussion

We discovered that the main issue for this dataset was the data collection protocol which was too relaxed. More precise instructions would increase considerably the quality of the data. Simultaneously, asking the users for more precision in following the protocol should be balanced with shorter sessions of data collection, as we noticed how the user commitment decreases over 4 days of continuous data collection. Lastly, as it has been demonstrated that experience sampling is not reliable, we recommend to increase the effort in the setup of the experiment by including a video recording. It would dramatically increase the quality and re-usability of the dataset, although it would require additional time for the labelling of the data. In order to reduce this effort, the video recordings can be used to precisely annotate just a small portion of the entire dataset. This well-annotated subset, which can be also collected a-posteriori, or can be used as training set instead of extracting one through heuristic. The well-annotated dataset would also allow to evaluate more precisely the performance of the TMM or any other classifier on the complete dataset, through statistical analysis.

In this study we extracted a subset of events which can be considered as drinking gestures with a certain confidence. This extracted subset can be potentially used to re-train the TMM for a more reliable gesture recognition system. The re-training phase can be performed using gestures with different levels of confidence: an higher level of confidence would increase the specificity of the found gestures. Decreasing this value would increase the sensitivity, potentially including more variations of the drinking gestures.

In our approaches, we used only the accelerometer signal recorded on the Z-axis. Using this axis, the lifting gesture was clear (see Figure 3), but it does not necessarily mean that the user lifts the cup and drinks. A more extensive and potentially robust approach would include also the X and the Y axis, computing the combined acceleration (on X and Y), in order to spot the actual sipping gesture.

Finally, we aimed to evaluate how an unsupervised learning technique can be used in order to extract drinking gestures from a poorly labelled dataset. We implemented a modified version of K-Means which uses WLCSS as distance measure for the assignment step. The results are promising: with 2 clusters it managed to differentiate between drinking and non-drinking gesture with an 93% F1-score, although on a limited number of sensor events. A more extensive evaluation can be performed on the entire dataset, although without a reliable ground-truth, a validation of the results, in this case, could be difficult.

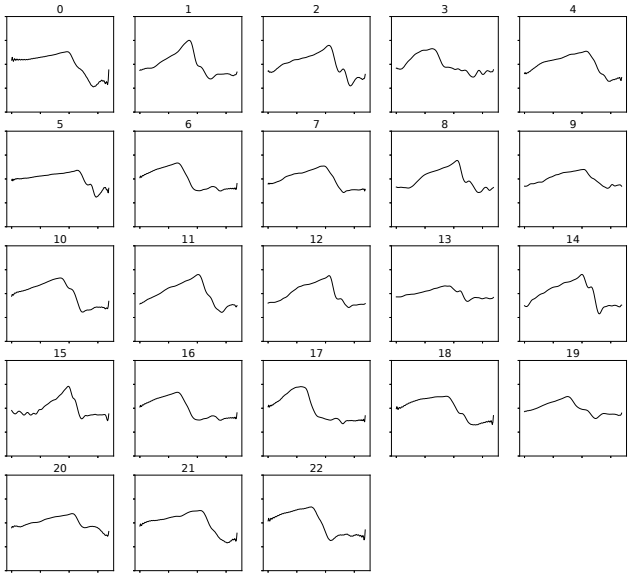
Conclusion

In this study, we investigate how to extract knowledge from a poorly labelled dataset of drinking gestures. We analysed the user annotations in order to get qualitative information on how to improve the data collection. We exposed how the loose protocol created most of the problems and we highlighted the need of providing more precise instructions to the users. Then, by selecting instances manually and using a template matching algorithm, we demonstrated that it is possible to extract a subset of instances which are actually drinking gestures within a certain level of confidence. We proved that an unsupervised approach based on K-Means

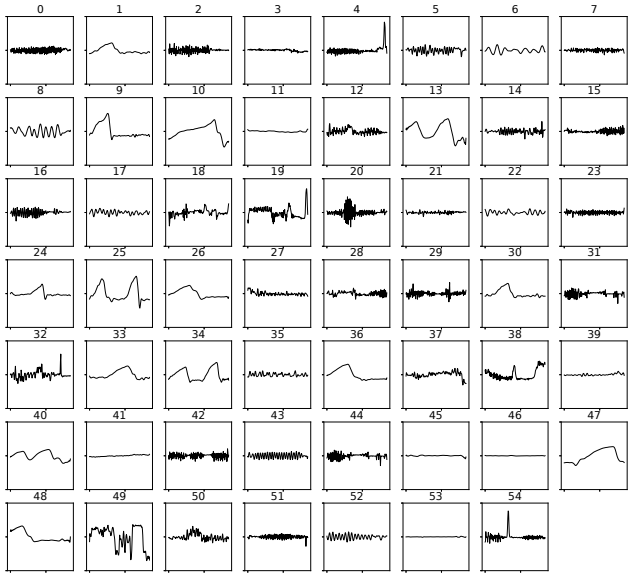
and WLCSS can improve the clustering of gesture over the standard K-Means implementation. Our method outperformed the baseline method by including a wider variety of drinking gestures and increasing F1-score by 16%.

REFERENCES

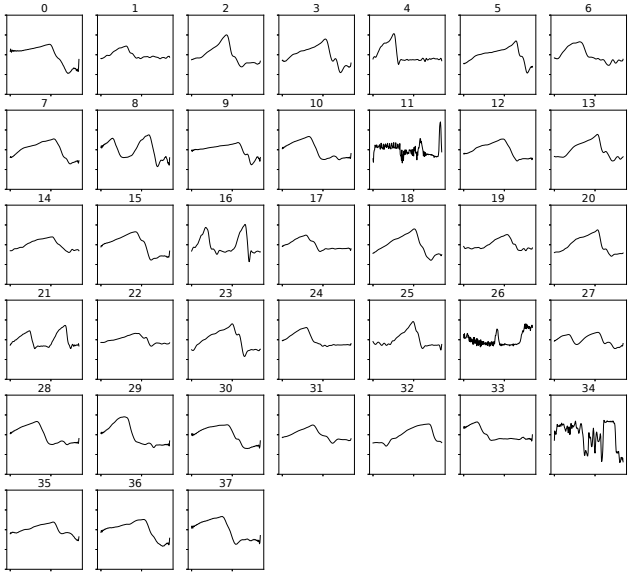
1. A. Bulling et al. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):1–33, 2014.
2. W. Duffy et al. Addressing the problem of Activity Recognition with Experience Sampling and Weak Learning. *Proceedings of SAI Intelligent Systems Conference*, pages 1–6, 2018.
3. H. Gjoreski and D. Roggen. Unsupervised online activity discovery using temporal behaviour assumption. *Proceedings of the ACM International Symposium on Wearable Computers*, pages 42–49, 2017.
4. J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.
5. Y. Kwon et al. Unsupervised learning for human activity recognition using smartphone sensors. *Expert Systems with Applications*, 41(14):6067–6074, 2014.
6. Z. Michalewicz. Evolution strategies and other methods. In *Genetic algorithms+ data structures= evolution programs*, pages 159–177. Springer Berlin Heidelberg, 1996.
7. S. Mitra et al. Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 37(3):311–324, 2007.
8. L.-V. Nguyen-Dinh et al. Improving online gesture recognition with template matching methods in accelerometer data. *International Conference on Intelligent Systems Design and Applications*, pages 831–836, 2012.
9. L.-V. Nguyen-Dinh et al. Robust Online Gesture Recognition with Crowdsourced Annotations. *Journal of Machine Learning Research*, 15:3187–3220, 2014.
10. L.-V. Nguyen-Dinh et al. Supporting One-Time Point Annotations for Gesture Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2270–2283, 2017.
11. D. Roggen et al. Collecting complex activity datasets in highly rich networked sensor environments. *Proceedings of International Conference on Networked Sensing Systems*, pages 233–240, 2010.
12. M. Stikic et al. Exploring semi-supervised and active learning for activity recognition. *IEEE International Symposium on Wearable Computers*, pages 81–88, 2008.
13. M. Stikic et al. Activity Recognition from Sparsely Labeled Data Using Multi-Instance Learning. In *International Symposium on Location- and Context-Awareness*, pages 156–173. IEEE, 2009.
14. X. Zhang et al. A framework for hand gesture recognition based on accelerometer and EMG sensors. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 41(6):1064–1076, 2011.
15. R. Zillmer et al. A robust device for large-scale monitoring of bar soap usage in free-living conditions. *Personal and Ubiquitous Computing*, 18(8):2057–2064, 2014.



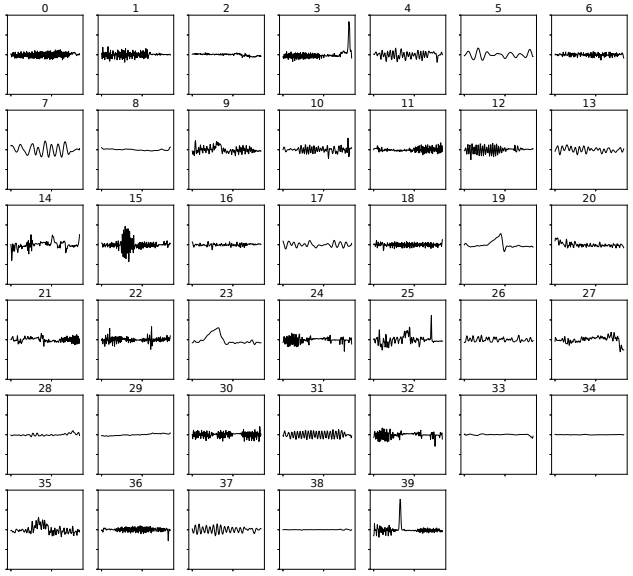
(a) *Cluster 1 - K-Means*



(b) *Cluster 2 - K-Means*



(c) *Cluster 1 - K-MeansWLCSS*



(d) *Cluster 2 - K-MeansWLCSS*

Figure 9: Comparison between clusters obtained with K-Means using Euclidean distance (top left, top right), and using WLCSS (bottom left, bottom right).

Drinking gesture recognition from poorly annotated data: a case study

Mathias Ciliberto, Lin Wang, Daniel Roggen and Ruediger Zillmer

Abstract In this chapter we present a case study on drinking gesture recognition from a dataset annotated by Experience Sampling (ES). The dataset contains 8825 "sensor events" and users reported 1808 "drink events" through experience sampling. We first show that the annotations obtained through ES do not reflect accurately true drinking events. We present then how we maximise the value of this dataset through two approaches aiming at improving the quality of the annotations post-hoc. Based on the work presented in [1], we extend the application of template-matching (Warping Longest Common Subsequence) to multiple sensor channels in order to spot a subset of events which are highly likely to be drinking gestures. We then propose an unsupervised approach which can perform drinking gesture recognition by combining K-Means clustering with WLCSS. Experimental results verify the effectiveness of the proposed method, with an improvement of the F1 score by 16% compared to standard K-Means using Euclidean distance.

Mathias Ciliberto
University of Sussex, Brighton, UK, e-mail: m.ciliberto@sussex.ac.uk

Lin Wang
Queen Mary University of London, UK, e-mail: lin.wang@qmul.ac.uk

Daniel Roggen
University of Sussex, Brighton, UK e-mail: daniel.roggen@ieee.org

Ruediger Zillmer
Unilever R&D Port Sunlight, UK e-mail: ruediger.zillmer@gmail.com

1 Introduction

Gesture recognition has applications in several fields such as healthcare and sports [2]. In order to create a reliable gesture recognition system, it is important to have a well-annotated dataset [3]. However, creating high-quality datasets may require to rely on lab-like environments, with limited ecological validity [4]. Activity recognition research generally strives to employ datasets with unrealistically "perfect" ground truth annotations. In an ecologically valid data collection, however, it is likely that a highly valuable dataset is acquired, but that only poor quality annotations are available.

Experience sampling (ES) is a real-time annotation approach done by users themselves a mobile device [13]. This allows more ecologically valid data collection in everyday life (e.g. no need to video record the experiment). However, ES can lead to the following issues: i) the synchronisation between the activity performed and the label annotated by the user is generally of poor quality, with the user annotating the activity after the event, or combining multiple activities in a single annotation; ii) the user may forget to label an event, iii) the user may annotate an activity with the wrong label.

In this work, we investigate how to make sense of a dataset with high business value comprising drinking gestures, which has been annotated through ES, leading to numerous deficiencies in the annotation quality. The dataset contains drinking gestures annotated by the users with a mobile application. The dataset was collected in an office environment using a 3-axis accelerometer and it is made by 8825 "sensor events", with 1808 "drink events" annotated by users through ES. Using this dataset, we aim to address two main challenges: i) to understand why the quality of the annotation is low and consequently how would it be possible to improve in future data collection and ii) to understand whether it is still possible to use such big dataset without relying on the annotations for spotting drinking gestures and how. This work is based on the research presented in [1]. The main contributions are:

- A study of the annotations. We analyse the user annotations, their distribution in time during the data collection, and their relation to the sensor events, in order to understand the causes of the low quality and where the data collection process can be improved.
- A template matching approach, based on Warping Longest Common Subsequence (WLCSS) [5], to extract a subset of drinking gestures, within a certain level of confidence. This subset will allow the dataset to be used for research purposes. As extension of the previous work, in this chapter, we evaluate multiple sensors channel in order to obtain a more precise selection of drinking gestures.
- An unsupervised algorithm (K-Means) adapted to template matching. This algorithm is a new variation of K-Means [6] where the WLCSS is used as distance measure. It allows to cluster gestures based on the raw signal of the sensors. At the same time, it clusters gestures taking in account the variation in the way they can be performed, by using WLCSS which has been successfully used for robust gesture detection [5].

2 Related work

The quality of annotations obtained through ES can be poor [7]. Annotations issues can include time shift of a label with respect to the activity, as well as wrong or missing labels [8].

Some approaches suggested to improve ES with manual re-annotation [7]. This is not feasible economically for a large datasets. Moreover, despite re-annotation the quality may still be insufficient for the training of machine learning algorithms [7]. The impact of ES on activity recognition has been studied in [9]. However, the authors simulated the ES in a controlled environment and they used only the data corresponding to the user annotations.

The problem of poorly labelled data can be tackled during the annotation itself or during the training of the machine learning algorithm. A method useful to reduce the effort of the users while annotating their activity has been proposed in [10]. The authors suggested a one-time point annotation method that requires the users to only label a single moment per activity rather than specifying the beginning and the end. The method then recognizes automatically the boundary of the activity in the annotated signal. Nevertheless, it requires that the labels are within the execution interval of the activity.

Unlabelled or poorly labelled data are available in big quantities nowadays due to the large diffusion of sensing devices, such as smartphones and wearables devices. For this reason, methods such as semi-supervised learning, active learning and unsupervised learning have been applied in order to extract useful information from sparsely annotated data. A combination of active learning and semi-supervised learning has been studied in [11]. The authors used a dataset of daily activities collected with two subjects wearing accelerometer sensors and motioned tracked with infrared sensors. This approach however uses a decision window of 30 seconds long and thus is not suitable for recognizing gestures that occur in a short time. Unsupervised learning has been successfully applied to activity recognition in [12] and more recently in [13]. In the latter, an activity discovery method based on clustering is proposed to help with ES, although it is designed for periodic movements rather than sporadic gesture. Unsupervised learning has been applied to gestures clustering in [14], where a K-Means clustering has been evaluated specifically for hand gestures.

Several studies have tried to address the challenge of activity recognition from poorly annotated data. While most of them used synthetic dataset and focused on periodic or long activity (such as walking, running, etc.), to the best of our knowledge none of them applies to drinking gestures collected in a real-life office environment.

3 Dataset

The dataset was collected by providing a set of mugs to 60 users (one mug per user) in an office environment. Each user collected data for a period of 4 days. Each mug was instrumented with a logger comprising a 3-axis accelerometer [15]. The loggers

were placed in a hollow at the bottom of each mug. As the mugs were customly made, the positioning of the loggers was not the same in all mugs. The loggers sample acceleration at 20 Hz, with a timestamp in ms. In order to save power, they start logging acceleration when a movement is detected. After 5 seconds of inactivity they automatically stop the recording, without record the inactivity period. We use the term *sensor events* to refer to every recording performed by the loggers that lasts at least 4 seconds (as configured on the loggers for this data collection). Therefore, sensor events can occur for a variety of reasons: moving the mug on the desk, washing it, drinking from the cup, etc.

The data annotation was performed through experience sampling by the users themselves. They labelled each drinking event manually using an Android application installed on their smartphones. Each annotation could be *punctual* or *delayed*. An annotation is considered punctual when it was entered immediately after the drinking event. It is considered delayed, when it refers to an event in the past. The users could specify in the application whether their annotation was punctual or delayed. However they did not have to provide an indication as to how much the delay was. Furthermore, there were no guidance indicating after how much time an annotation should be considered "delayed" rather than "punctual".

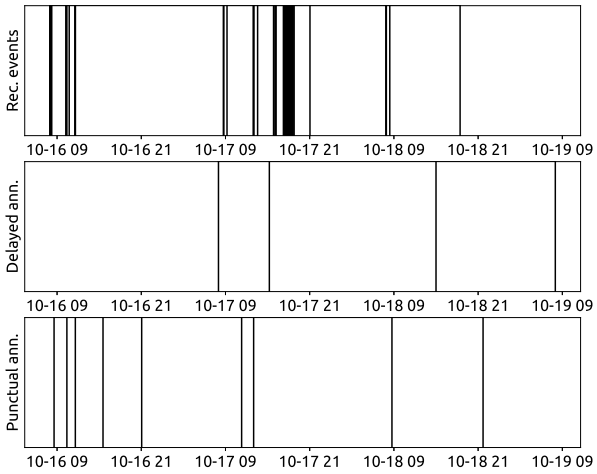
The resulting dataset is made by 8825 sensor events, 1808 user annotations, of which 1477 marked as "punctual" and 331 as "delayed". The percentage of annotated gestures with respect to the total amount of sensor events is of 20.5%.

4 User annotation analysis

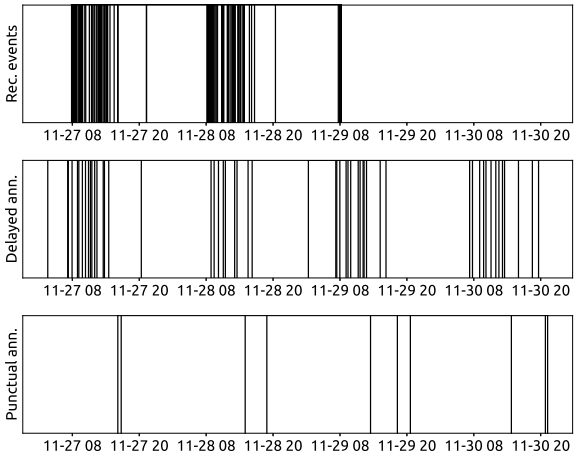
We aim to analyse the causes of the poor annotations in order to improve future data collections, as well as helping during the next steps of this study.

Figure 1 indicates the main challenge of the annotation protocol, which is how users understood differently how and what to annotate. The data collection protocol did not require participants to annotate drinking solely when using the instrumented mugs: they could annotate drinking as well when using regular mugs. It might happen that users annotated drinking events performed using other cups. The protocol did also not specify what to consider as a "drinking event". Users could interpret it as referring to a single sip, multiple sips, or the act of drinking the entire cup. It is also possible to notice how the annotations are not well aligned with the sensor events.

We also studied the distribution of the labels, per user, over the 4 days of data collection. It could help to understand the users' commitment in annotating their drinking gestures, assuming they were keeping the same drinking habits among all the days. This may be useful in order to spot days for which the annotations can be more reliable. The results are presented in Figure 2. While there is no significant change between day 1 and day 2, with an average increase in the number of annotation of 0.24%, starting from day 3 the engagement decrease by 11% on average among all the users. The plot shows also a great variability in the data: there were users



(a) *User 109*



(b) *User 461*

Fig. 1: Example of annotations of two different users, over the 4 days period. The start time of the sensor events are displayed in the top plot of each figure, one thin line per event. The delayed and punctual annotations inserted by the users are displayed respectively in the second and the third plot of both figures. The X-axis reports date and time, in the format "MM-DD HH". It is also possible to notice the differences in the way two users annotated the drinking events.

that increased their commitment over the 4 days, as well as users for which the commitment decreased over the 4 days.

From the analysis of the annotations, it can be concluded that they were not reliable enough to be used together with the data for the supervised classifier training.

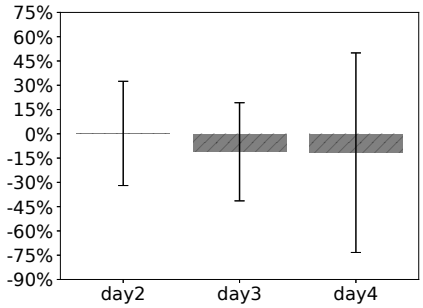


Fig. 2: Change in the user commitment in the annotation during day 2, 3, and 4. The grey bars represents the percentage of annotations for each day with respect to day 1. Day 2 displays an increment of 0.24%; Day 3 and 4 an average decrease of 11% in the number of annotations. The vertical bars represents the standard deviation for each day.

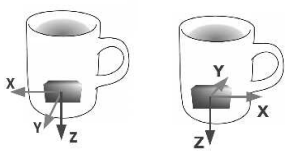
5 Gesture classification

In order to make the collected dataset useful for drinking gesture recognition, each event recorded by the sensors had to be classified in drinking/non-drinking. As highlighted previously, the users annotations cannot be used as-is as they are not accurate enough. A manual relabelling of the entire dataset was unfeasible given the lack of any video recordings.

We developed an approach based on a template matching method (TMM) to automatically spot a subset of events which are believed to be drinking gestures with a certain confidence value. The approach then uses few events which are manually identified as drinking events with high confidence to train the TMM.

5.1 Data processing and training set selection

We used a heuristic method to select a few sensor events as the training set. We performed a few drinking gestures using the same instrumented mug in order to chose the best sensor channel for template matching.



(a) (b)

Fig. 3: Examples of orientation of the loggers in the custom made mugs. Although X and Y axis can be different in each mug, the Z-axis is always facing downwards.

The chosen channel (or channels) must be orientation independent, as there was no information about the positioning of the logger in the mug (Figure 3). We discovered that the Z-axis of the accelerometer quite clearly indicates the gesture of lifting the cup to drink. Also, despite different orientation of the loggers, this axis was always perpendicular to the bottom of the mug. A template of such gesture is displayed in Figure 4. A subset of gestures visually similar to this template was selected manually from the entire set of the available gestures. We selected this subset trying to include some variability in the way the drinking gestures were performed. Another subset of non-drinking gestures was selected too, choosing the templates that were very visually different from the drinking gestures. The training set is displayed in Figure 5. It is formed by 78 events: 37 drinking gestures (5a) and 41 non-drinking gestures (5b).

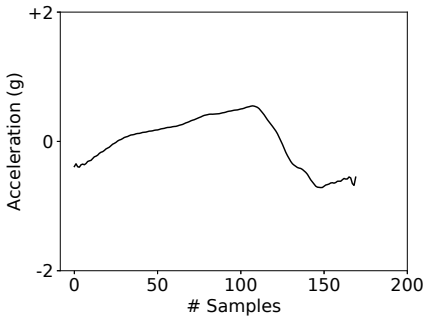
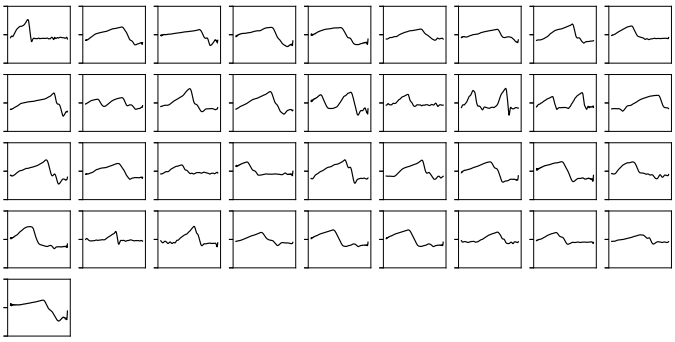
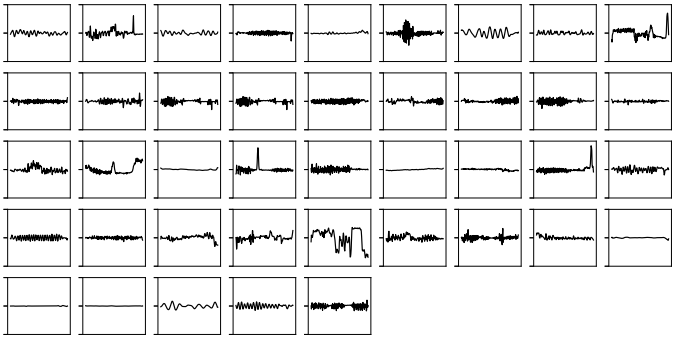


Fig. 4: Template of a drinking gesture, performing a single sip, displaying the acceleration on the Z-axis.

However, lifting the cup does not necessarily mean that a drinking was actually performed. For this reason, in order to better detecting the rotation of the mug due to the drinking, we also used the magnitude of the acceleration on X-Y plane as additional information. This was also due to the lack of the gyroscope on the loggers. The X-Y plane was chosen because it was always parallel to the bottom of the mug (Figure 3). The magnitude was computed as $m_{xy} = \sqrt{x^2 + y^2}$. A template



(a) *Drinking gestures*



(b) *Non-drinking gestures*

Fig. 5: Training set of gestures using the Z-axis of the accelerometer. 5a displays the templates chosen as drinking gestures. 5b shows those selected as no-drinking gestures. All the plots show the templates downsampled to the fixed length of 170 samples (X-axis). The Y-axis represents the acceleration within a range of ± 2g.

for a drinking gesture represented by the magnitude is shown in Figure 6. A different training set based on the template of the magnitude was chosen. The choice was based on visual similarity to such template. A subset of non-drinking gesture was also chosen for the XY magnitude The training set of drinking and non-drinking gesture for the XY magnitude is displayed in 7. The drinking gestures were selected only when they corresponded to a lifting of the mug (Z-axis).

All the instances in the dataset were filtered using a Butterworth low pass filter with cut off frequency set to 10 Hz. They were also resampled to a fixed number of samples. The number of samples was selected as the average length of a drinking gesture, which is 170. This step was performed in order to reduce the impact of non-drinking events that can last longer time than drinking gestures (e.g. washing the cup, moving the cup around the office, etc.).

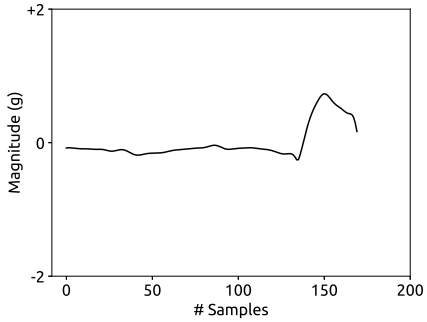


Fig. 6: Template of a drinking gesture, performing a single sip, displaying the magnitude of the acceleration on the X-Y plane.

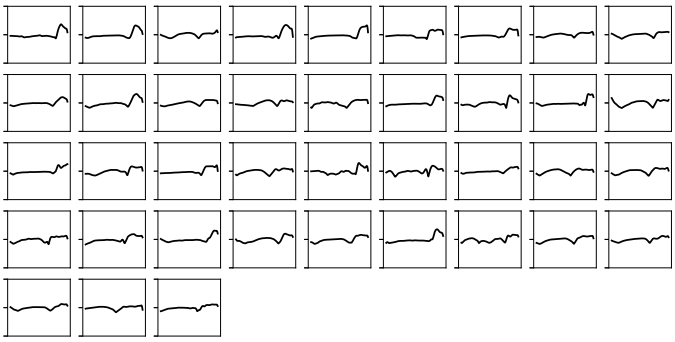
In order to evaluate which is the best channel for drinking gesture recognition, we decided to match the templates for Z-axis signal and XY magnitude separately.

5.2 Template matching using WLCSS

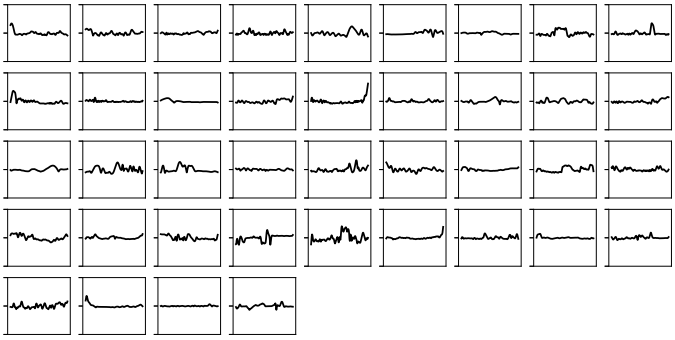
The Warping Longest Common Subsequence (WLCSS) [5] is an algorithm developed for template matching in real-time applications. Using dynamic programming, the algorithm can compute a matching score between a template and a stream, updating it at every new sample of the stream. It can be used for gesture recognition as it can handle gestures performed with variation in their speed of execution. This is achieved by three parameters: reward (R), penalty (P) and acceptance distance (ϵ). The algorithm is shown in (1).

$$M(i,j) = \begin{cases} 0 & \text{if} \quad i \leq 0 \quad or \quad j \leq 0 \\ M(j-1,i-1) + R & \text{if} \quad |S(i) - T(j)| \leq \epsilon \\ max \begin{cases} M(j-1,i-1) - P \cdot |S(i) - T(j)| \\ M(j-1,i) - P \cdot |S(i) - T(j)| \\ M(j,i-1) - P \cdot |S(i) - T(j)| \end{cases} & \text{if} \quad |S(i) - T(j)| > \epsilon \end{cases} \tag{1}$$

The matching score M(i,j) is computed as function of the previous scores, by adding a reward (R) when the distance between the i-th stream sample ($S(i)$) and the j-th template sample ($T(j)$) is below an acceptance distance (ϵ), or by subtracting a penalty (P) proportional to the distance, when this is above ϵ. In addition to R,P, and ϵ, WLCSS needs a threshold T. As WLCSS computes a matching score (M) between an instance in the dataset and a template, T is used to define whether an instance matches with the template (M≥T) or not (M<T). The value of the threshold



(a) *Drinking gestures*



(b) *Non-drinking gestures*

Fig. 7: Training set of gesture using the magnitude of X and Y axis of the accelerometer. 7a displays the templates chosen as drinking gestures. 7b shows those selected as no-drinking gestures. All the plots show the templates downsampled to the fixed length of 170 samples (X-axis). The Y-axis represents the acceleration within a range of ± 2g.

is related to the specificity and the sensitivity of the matching algorithm: a high threshold means high specificity, while a low threshold means high sensitivity. The values of R,P,ϵ and T must be found during the training phase. We optimized this based on an evolutionary optimization technique.

5.3 WLCSS optimization using evolutionary algorithm

We optimise the values of R, P, e, T to maximise the ability of WLCSS to distinguish drink from non-drink using an evolutionary algorithm (EA). We used the EA in

order to optimize the values of the parameters starting from a randomly generated population. Each individual of the population is an array containing the 4 parameters. The EA evolves this population through the usual selection, mutation and crossover operators [16]. Here, the F1 score is used for the selection. The optimization process stops after a predefined number of iterations, in this case .

5.4 Confidence computation

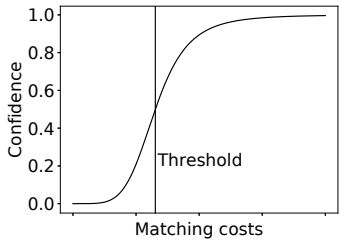


Fig. 8: 4 Parameter Logistic Regression function used to assign a confidence with the respect to the threshold.

Given the unreliability of the labels in the dataset, it is not possible to evaluate precisely the correctness of a match for this particular dataset. For this reason, we opted to provide a confidence level for each gesture as output of our method rather than a simple match/no-match. The confidence level was assigned using Four Parameter Logistic Regression:

$$y = d + \frac{a - d}{1 + (\frac{M}{T})^b}$$

where y is the confidence level, M is the matching cost, and T is the threshold. The function, displayed in Figure 8, provides a confidence value in the range [0:1]. The range is defined by the parameters $a = 0$ and $d = 1$. The parameter b, which define the slope of the function curve, was set manually to 5. Using a fixed interval for the confidence makes its value unrelated from the absolute value of T, which can vary according with the parameters R, P and ϵ.

Finally, we computed three confidence values: i) a value for the Z-axis signal (c_z) used to detect the lifting of the mug, ii) a value for the matching of the XY magnitude template (c_{xy}) useful for the detection of the mug's rotation and iii) a combined value that takes into account the previous twos (c_{comb}). This latter value was empirically defined as:

$$c_{comb} = 0.7 * c_z + 0.3 * c_{xy}$$

The weights for c_z and c_{xy} were chosen experimentally based on the assumption that a drinking gesture, intended as rotation of the mug, is performed only after the lifting the mug.

5.5 Evaluation

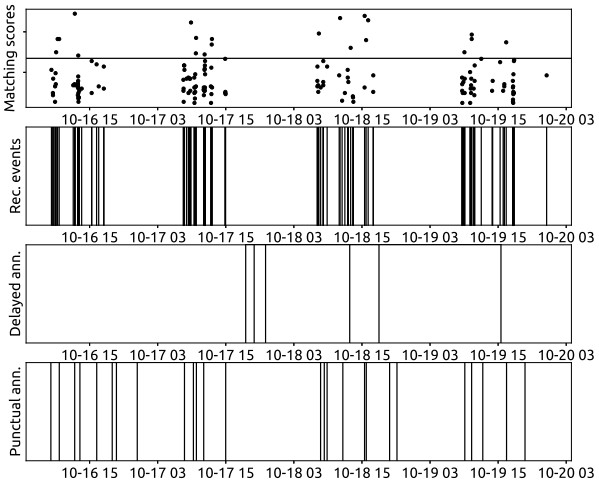
We used the two different subset of instances (Z-axis signal and XY magnitude) to train two separate instances of the system using the EA. As the EA is a stochastic process, we repeated the training 10 times, and we picked the best values for each of the two systems. The values are R=68, P=0, ϵ=28 and T=3364 for the Z-axis and R=23, P=11, ϵ=11 and T=726 for XY magnitude. With these parameters, we run the algorithm on the entire dataset, by using the templates displayed in Figure 4 and Figure 6, which were selected manually from the training sets as templates.

Figure 9 displays a comparison of the sensor events and the annotations for a single user, with the corresponding matching scores, for the Z-axis (9a) and XY Magnitude (9b). The matching of the template for XY magnitude is less restricting than the matching on the Z-axis. This is possibly due to the fact the rotation of the mug can happen even in case of gesture different from just the drinking: e.g. while washing the mug, moving it around, etc.

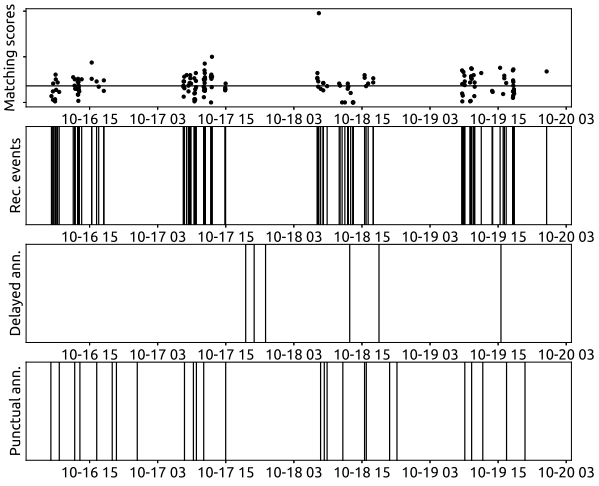
The percentages of total detected gestures for each scenario, compared to the total number of events are displayed in Table 1, for different confidence values. The low percentages are due to the nature of the sensors, which were collecting all sort of movements such as moving the mug on the desk, washing it or even accidental movements. It is important to note that the number of detected events is also lower than the number of user annotations (1808). This may be a result of the data collection protocol which did not specify to annotated only the drinking movements performed with the instrumented mug. Finally, for low confidence values ($\geq$25% and $\geq$50%) the matching of the Z-axis signal and the XY magnitude are quite different in term of percentage. For higher values of the confidence ($\geq$75%), the performance of the two systems tend to be the same, with similar percentages of detected events. Moreover, in order to better compare the performance of the two systems, we created the cross-correlogram presented in Figure 10. It displays the distribution of the differences between the confidence c_z and c_xy. While the two systems generally agree, as shown by the peak on 0, there is a negative skewness. This means that system for XY magnitude is less precise in detecting the drinking events than detecting the lifting of the mug using the Z-axis.

From the analysis, it is clear how the Z-axis signal is better for detecting a first subset of drinking gestures rather then just the XY magnitude. The latter is useful to filter out events that refer to lifting of the cup but without an actual drinking performed by the user.

We studied the relation between user annotations, sensor events and detected gestures, for the two systems. To do this, we assigned to every recorded event the closest user annotation in time. Then, computing the time difference between the sensor



(a)



(b)

Fig. 9: Comparison of WLCSS matching scores with recorded data for a single user, for Z-axis (9a) and XY magnitude (9b). For each of the two template matching, from the top: the WLCSS matching scores and the threshold T, as horizontal line (first plot), the start time of the sensor events (second plot), and the user annotations delayed and punctual (respectively third and fourth plot). The data are for 4 days period, with the X-axis reporting date and time in the format "MM-DD HH"

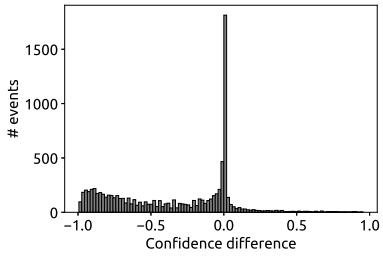
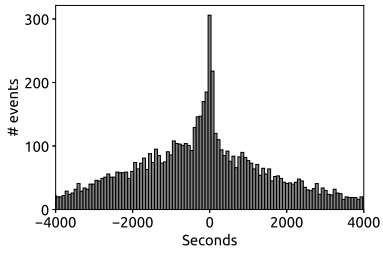
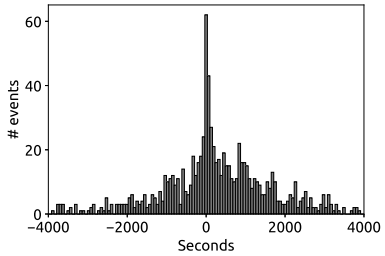


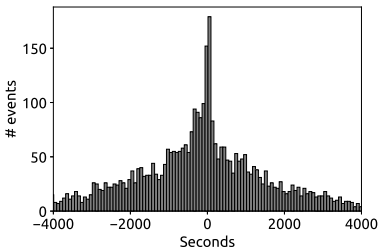
Fig. 10: Cross-correlogram representing the difference between the confidence obtained using the Z-axis signal and the XY magnitude.



(a)



(b)



(c)

Fig. 11: Cross-correlogram representing the distribution of the delays (in seconds) between the user annotations and all the events recorded by the loggers (11a). Cross-correlogram representing the distribution of the delays (in seconds) between the user annotations and the events detected with a confidence ≥50%, using the signal from the Z-axis only (11b). Cross-correlogram representing the distribution of the delays (in seconds) between the user annotations and the events detected with a confidence ≥50%, using the XY magnitude (11c).

Table 1: Number of drinking gestures detected for some confidence levels. The percentages are with respect to the total number of sensor events in the dataset (8825).

Confidence	Z-Axis		XY-Magnitude		Combined	
	# gestures	%	# gestures	%	# gestures	%
≥ 25%	1481	17%	5253	60%	3930	44%
≥ 50%	942	10%	4365	49%	1113	12%
≥ 75%	543	6%	3453	39%	483	5%

events and the corresponding closest annotations, we created the cross-correlogram displayed in Figure 11a. Figure 11b and Figure 11b present the distribution of the same time differences, but considering only the gestures detected by WLCSS with a confidence ≥50%, respectively for Z-axis and XY magnitude. The plot for Z-axis presents a more pointy distribution, confirming that, in this case, WLCSS detected events that were actually closer in time to the user annotations. The plot for XY magnitude confirms the lower precision of the detection with respect to the user annotation.

6 Unsupervised learning

We evaluated also an unsupervised approach in order to classify the gestures in drinking/non-drinking as it does not require a training set. We developed a custom method based on K-Means. We modified K-Means in order to make it able to cluster gestures performed with variation in their speed of execution.

6.1 K-Means with WLCSS

K-Means is a clustering technique that aims to partition n observation in k clusters. Each observation belongs to the cluster with the closest mean. It can be used for unsupervised learning by clustering the input data based on a distance measure. The algorithm is based on two steps, assignment step and update step, which are repeated until a stopping criteria is met [6]. This criteria can be reaching a maximum number of iterations, the change of the clusters in the update step in below a thresholds, etc. We implemented a modified version of the K-Means, where WLCSS was used a distance measure in place of the Euclidean distance. The assignment in our implementation is modified as following:

$$\arg\max_{c_i} WLCSS(x, c_i)$$

where x is a sensor events, c_i is the centroid for the i-th cluster. The function *argmin* is replaced by *argmax* as WLCSS compute a matching score rather than a distance. The update step is unmodified.

6.2 Evaluation

We compared our version of K-Means (named K-MeansWLCSS) against the standard version that uses the Euclidean distance in order to assign each instance to the closest cluster. We applied both the implementation on the training set from the previous step, with $k = 2$ as the goal was to distinguish between drinking and non-drinking gestures. As all the instances were resampled to the same length, they could be used as feature vectors for both the implementations, without dealing with different lengths of the feature vectors (in this case the resampled raw signal). Applying the algorithms on the training set allowed us to compare the clustering results with the labels assigned manually to each gesture, during the data selection. Figure 12 presents a visual comparison between the clusters obtained with the two versions of K-Means. For both the implementations, Cluster 1 seems to include mainly the drinking gestures, while Cluster 2 the non-drinking gestures. We used this consideration in order to evaluate the performance of the two algorithms computing precision, recall and F1 score, presented in Table 2. They are computed by comparing the clustering of K-Means with the manual labels of the instances in the extracted subset. K-Means_WLCSS increased the F1 score by 16%, being able to detect more variations in the drinking gestures as it is also visible from Figure 12. It was able to cluster correctly drinking gestures composed by two sips, such as the instances 8, 16, and 21 of Figure 12c.

7 Discussion

Table 2: Precision, Recall and F1 score for K-Means and K-MeansWLCSS computed on the training set. They are computed using the manual labels assigned to each instance in the training set as ground-truth. The majority of the gestures in a cluster is used as classification label for the K-Means implementations.

	K-M	K-M_WLCSS
Precision	100%	92.11%
Recall	62.16%	94.6%
F1	77%	93%

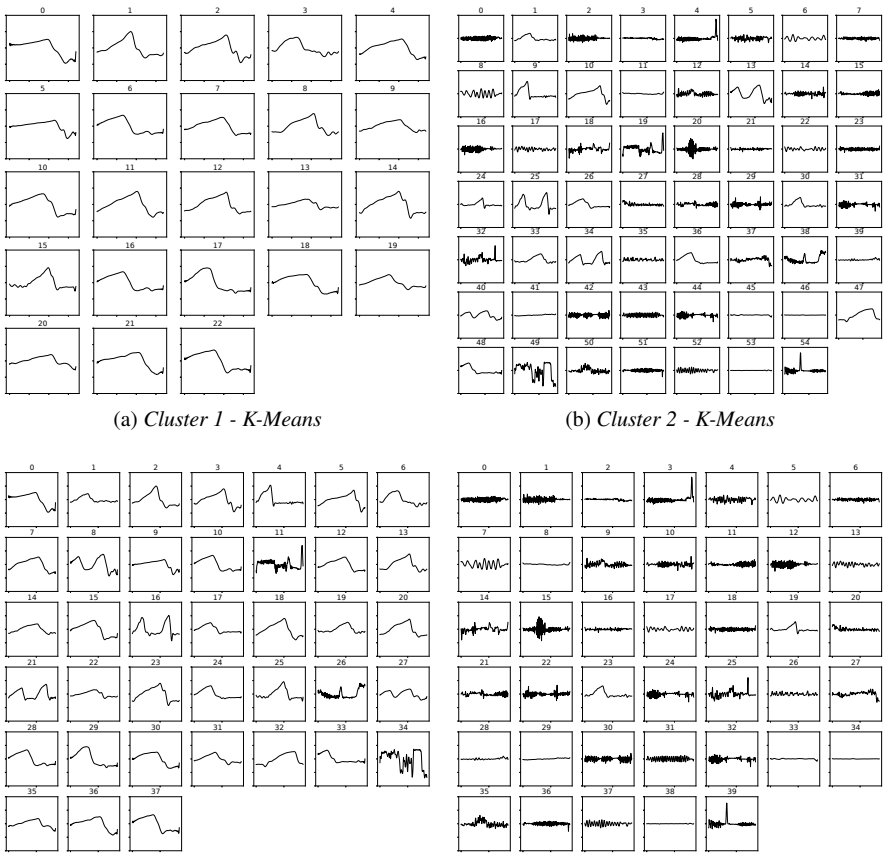
We discovered that the main issue for this dataset was the data collection protocol which was too relaxed. More precise instructions would increase considerably the quality of the data. Simultaneously, asking the users for more precision in following the protocol should be balanced with shorter sessions of data collection, as we noticed how the user commitment decreases over 4 days of continuous data collection. Lastly, as it has been demonstrated that experience sampling is not reliable, we recommend to increase the effort in the setup of the experiment by including a video recording. It would dramatically increase the quality and re-usability of the dataset, although it would require additional time for the labelling of the data. In order to reduce this effort, the video recordings can be used to precisely annotate just a small portion of the entire dataset. This well-annotated subset, which can be also collected a-posteriori, or can be used as training set instead of extracting one through heuristic. The well-annotated dataset would also allow to evaluate more precisely the performance of the TMM or any other classifier on the complete dataset, through statistical analysis.

In this study we extracted a subset of events which can be considered as drinking gestures with a certain confidence. This extracted subset can be potentially used to re-train the TMM for a more reliable gesture recognition system. The re-training phase can be performed using gestures with different levels of confidence: an higher level of confidence would increase the specificity of the found gestures. Decreasing this value would increase the sensitivity, potentially including more variations of the drinking gestures.

Finally, we aimed to evaluate how an unsupervised learning technique can be used in order to extract drinking gestures from a poorly labelled dataset. We implemented a modified version of K-Means which uses WLCSS as distance measure for the assignment step. The results are promising: with 2 clusters it managed to differentiate between drinking and non-drinking gesture with an 93% F1-score, although on a limited number of sensor events. A more extensive evaluation can be performed on the entire dataset, although without a reliable ground-truth, a validation of the results, in this case, could be difficult.

8 Conclusion

In this study, we investigate how to extract knowledge from a poorly labelled dataset of drinking gestures. We analysed the user annotations in order to get qualitative information on how to improve the data collection. We exposed how the loose protocol created most of the problems and we highlighted the need of providing more precise instructions to the users. Then, by selecting instances manually and using a template matching algorithm, we demonstrated that it is possible to extract a subset of instances which are actually drinking gestures within a certain level of confidence. We proved that an unsupervised approach based on K-Means and WLCSS can improve the clustering of gesture over the standard K-Means implementation. Our method outperformed the baseline method by including a wider variety of drinking gestures and increasing F1-score by 16%.



(a) *Cluster 1 - K-Means* (b) *Cluster 2 - K-Means*

(c) *Cluster 1 - K-MeansWLCSS* (d) *Cluster 2 - K-MeansWLCSS*

Fig. 12: Comparison between clusters obtained with K-Means using Euclidean distance (top left, top right), and using WLCSS (bottom left, bottom right).

References

1. M. Ciliberto, L. Wang, D. Roggen, R. Zillmer, in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers* (ACM, 2018), pp. 1434–1443
2. S. Mitra, et al., IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews (3), 311 (2007)
3. A. Bulling, et al., ACM Computing Surveys (3), 1 (2014)
4. D. Roggen, et al., Proceedings of International Conference on Networked Sensing Systems pp. 233–240 (2010)
5. L.V. Nguyen-Dinh, et al., International Conference on Intelligent Systems Design and Applications pp. 831–836 (2012)
6. J.A. Hartigan, M.A. Wong, Journal of the Royal Statistical Society. Series C (Applied Statistics) (1), 100 (1979)

7. M. Stikic, et al., in *International Symposium on Location- and Context-Awareness* (IEEE, 2009), pp. 156–173
8. L.V. Nguyen-Dinh, et al., Journal of Machine Learning Research pp. 3187–3220 (2014)
9. W. Duffy, et al., Proceedings of SAI Intelligent Systems Conference pp. 1–6 (2018)
10. L.V. Nguyen-Dinh, et al., IEEE Transactions on Pattern Analysis and Machine Intelligence (11), 2270 (2017). URL http://ieeexplore.ieee.org/document/7778186/
11. M. Stikic, et al., IEEE International Symposium on Wearable Computers pp. 81–88 (2008)
12. Y. Kwon, et al., Expert Systems with Applications (14), 6067 (2014)
13. H. Gjoreski, D. Roggen, Proceedings of the ACM International Symposium on Wearable Computers pp. 42–49 (2017)
14. X. Zhang, et al., IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans (6), 1064 (2011)
15. R. Zillmer, et al., Personal and Ubiquitous Computing (8), 2057 (2014)
16. Z. Michalewicz, in *Genetic algorithms+ data structures= evolution programs* (Springer Berlin Heidelberg, 1996), pp. 159–177

WLCSSCuda: A CUDA Accelerated Template Matching Method for Gesture Recognition

Mathias Ciliberto
m.ciliberto@sussex.ac.uk
Wearable Technologies Lab,
University of Sussex
Brighton, UK

Daniel Roggen
d.roggen@ieee.org
Wearable Technologies Lab,
University of Sussex
Brighton, UK

ABSTRACT

Template matching methods can benefit from multi-cores architecture in order to parallelise and accelerate the matching of multiple templates. We present WLCSSCuda: a GPU accelerated implementation of the Warping Longest Common Subsequence (WLCSS) pattern recognition algorithm. We evaluate our method on 4 NVIDIA GPUs and 4 multi-cores CPUs. We observe a 67-times speedup for the GPU implementation in the best case against the multithreaded CPU implementation.

CCS CONCEPTS

• **Computing methodologies** → *Parallel algorithms.*

KEYWORDS

WLCSS; CUDA; GPU acceleration; Template Matching

ACM Reference Format:
Mathias Ciliberto and Daniel Roggen. 2019. WLCSSCuda: A CUDA Accelerated Template Matching Method for Gesture Recognition. In *Proceedings of the 2019 International Symposium on Wearable Computers (ISWC '19), September 9–13, 2019, London, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3341163.3347718

1 INTRODUCTION

Template Matching Methods (TMM) have been successfully applied in activity recognition [2]. They compute a matching score between one or more templates and a stream of sensor data. Therefore, they can benefit from multi-core architectures in order to improve their speed of execution. They can take advantage of such architectures when multiple templates must be simultaneously compared to an incoming sensor signal as well as during training when multiple instances of the TMM must be run in order to evaluate different parameter sets. WLCSSCuda is a GPU accelerated implementation of Warping Longest Common Subsequence (WLCSS) using CUDA. This is a framework developed by NVIDIA for general purpose processing on GPUs (GPGPU) [3]. In the past, GPUs have been used to accelerate other TMMs, such as Dynamic Time Warping [5] and Longest Common Subsequence [6]. However, to the best of our knowledge, this is the first GPU implementation of WLCSS. WLCSSCuda allows to execute multiple instances of the TMM simultaneously. We compare this approach with a multi-core CPU implementation of WLCSS and show an drastic speedup in the matching score computation.

ISWC '19, September 9–13, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6870-4/19/09.
https://doi.org/10.1145/3341163.3347718

2 WARPING LONGEST COMMON SUBSEQUENCE

Warping Longest Common Subsequence (WLCSS) is a TMM suitable for continuous pattern recognition - such as spotting complex gestures - which is more robust than Dynamic Time Warping to noisy sensor data [2]. It computes a matching score ($M_{i,j}$) between a template (T) and a stream (S) by adding a reward (R) every time a sample i from S matches with a sample j from T within an acceptance threshold (ϵ). Otherwise, it decrements $M_{i,j}$ by a penalty (P) proportional to the mismatch.

Optimising WLCSS parameters requires an exhaustive grid search for R, P, ϵ, which benefits from computational speedups.

3 WLCSSCUDA

WLCSSCuda is our GPU implementation of WLCSS built using CUDA. Thanks to the high number of cores in modern GPUs, it is possible execute tasks, called kernels, with a high level of parallelization. CUDA abstracts the physical structure of the GPU in a grid of blocks. Each block can be addressed using a 1D, 2D, or 3D index. A block is a unit made by several threads that can be computed in parallel or serially on a GPU core. A 3D indexing is provided for the thread too. The scheduling of the threads' execution is transparent to the user. The number of maximum blocks and maximum threads per block depends on the GPU.

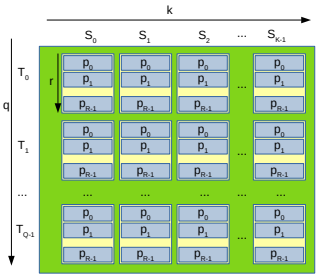


Figure 1: WLCSSCuda structure: blocks are represented in yellow. A template T_q and a stream S_k are assigned to each block. A set of parameters p_r is assigned to each thread, displayed in blue.

WLCSSCuda structures the computation using a 2D grid for the blocks and 1D structure for the threads, as shown in Figure 1: a single template/stream pair is assigned to each block. Then, within that block, a thread is used for every parameters set. During initialization, all the templates, streams and parameters are transferred to the GPU memory. Then, each kernel computes the pointer to the templates, stream and WLCSS parameters in memory using respectively the indexes q and k for the blocks, and r for the threads. Each triad template/stream/parameters is used by only one kernel which is executed by a single thread. Finally, when the matching scores are computed, they are transferred from the GPU memory back to the main memory. WLCSSCuda computes the entire score between the template and the stream.

WLCSSCuda is developed in C++ with a Python wrapper for loading the data and reading the results.

4 EVALUATION

We evaluated to which extent WLCSSCuda could accelerate multiple template matching, taking into account the time required to transfer the data to/from the GPU, which has been demonstrated to be a bottleneck in CUDA applications [1]. We compared the execution of WLCSSCuda on 4 GPUs and a multithreaded WLCSS on 4 CPUs (see Table 1). We simulates 4 test scenarios in which a different number of streams, templates and WLCSS parameters were employed (Table 2). We reported the average of 10 executions. The CPU implementation of WLCSS uses all the available threads in the CPU to run always the maximum number of possible WLCSS simultaneously.

We used OPPORTUNITY Dataset [4] as source of templates and streams, selecting a random subset of gestures for each test, to make them more realistic. The average length of the templates (and streams) is 98 samples, with a standard deviation of ± 46. The same subset of gestures was used for WLCSSCuda and the CPU implementation in each test.

GPU Model	CUDA cores	Cores Frequency	Memory
GTX 1080 Ti	3584	1645 MHz	11 GB GDDR5X
GTX 1050 Ti	768	1418 MHz	4 GB GDDR5
GTX 970	1664	1317 MHz	4 GB GDDR5
Titan XP	3840	1582 MHz	12 GB GDDR5X

CPU Model	Frequency (Max Turbo)	# of Cores	# of Threads
AMD Ryzen 1900X	3.8 GHz (4 GHz)	8	16
Intel i7-8750H	2.2 GHz (4.1 GHz)	6	12
Intel i7-4770K	3.5 GHz (3.9 GHz)	4	8
Intel i7-6700	3.4 GHz (4.0 GHz)	4	8

Table 1: CPU and GPU tested. For more details about the GPUs and CPUs (AMD and Intel), visit respectively [1], [2], [3]

Test	# Templates (Q)	# Streams (K)	# Params. sets (R)	Tot. WLCSS
a	10	1000	10	100000
b	20	2000	10	400000
c	50	5000	10	2500000
d	100	10000	10	10000000

Table 2: The number of templates, streams, parameters sets and the total number of WLCSS computation is shown, for each test scenario.

Platform / Test	*a*	*b*	*c*	*d*
Titan XP	0.49 ± 0.04	1.53 ± 0.15	9.66 ± 0.61	38.29 ± 1.87
GTX 1080 Ti	0.55 ± 0.07	1.98 ± 0.14	12.77 ± 1.51	51.38 ± 6.22
GTX 1050 Ti	0.79 ± 0.13	2.92 ± 0.26	18.15 ± 0.67	72.04 ± 3.59
GTX 970	0.91 ± 0.16	4.00 ± 0.67	18.97 ± 0.86	70.74 ± 3.69
AMD 1900X	8.29 ± 0.89	32.27 ± 1.74	231.75 ± 31.59	932.69 ± 86.19
i7-8750H	17.50 ± 2.57	81.23 ± 5.30	555.26 ± 15.39	2140.20 ± 141.37
i7-6700	20.35 ± 1.07	80.18 ± 8.28	523.05 ± 34.30	2008.19 ± 81.33
i7-4770K	22.78 ± 3.20	99.62 ± 10.00	647.21 ± 45.91	2452.15 ± 170.12
Improvement	10-46 times	8-65 times	12-67 times	13-64 times

Table 3: Results of WLCSS running on the 4 GPUs and the 4 CPUs. The values are in seconds and they are averaged across multiple running. The improvements are computed respectively between the best CPU against the worst GPU, and viceversa.

Table 3 shows the average time for each test for every CPU and GPU. For WLCSSCuda, all the values include the time to the transfer of data to/from the GPU memory. As we expected, the GPUs are faster in every scenario we evaluated. Moreover, it is possible to notice how WLCSSCuda scales better when the number of instances increase. Test *d* requires 100 times more evaluations than test *a*; the GPUs take on average only 85 times the time required by test *a* while the CPUs take 110 times more than *a*, on average, across all the different models.

[1] https://www.nvidia.com/en-gb/
[2] https://www.amd.com/en/products/ryzen-threadripper
[3] https://ark.intel.com

5 DISCUSSION AND CONCLUSION

We presented WLCSSCuda, a GPU accelerate multiple TMM. We demonstrated that WLCSSCuda can drastically increase the computation of multiple template matching, with an increase of 67 times in the best case compared to a multi-threaded CPU approach. However, there is still room for improvement: we plan to evaluate different organizations of data in order to better use the block/thread CUDA structure. Moreover, we aim to make WLCSSCuda automatically adapting such structure according with the number of templates/streams/parameters sets in order to increase the performance even further. Finally, WLCSSCuda is available as open source software at the address https://github.com/sussexwearlab/WLCSSCuda.

ACKNOWLEDGMENTS

This study was partly funded through the FFG project #5766494 "MinIAttention: Attention Management in Minimal Invasive Surgery". We also thank NVIDIA for their Titan XP donation.

REFERENCES

[1] Chris Gregg and Kim Hazelwood. 2011. Where is the data? Why you cannot debate CPU vs. GPU performance without the answer. In *International Symposium on Performance Analysis of Systems and Software*. IEEE.

[2] Long Van Nguyen-Dinh et al. 2012. Improving online gesture recognition with template matching methods in accelerometer data. *International Conference on Intelligent Systems Design and Applications* (2012).

[3] Nvidia. 2019. *CUDA Toolkit.* https://developer.nvidia.com/cuda-toolkit Retrieved on April 26, 2019 from https://developer.nvidia.com/cuda-toolkit.

[4] Daniel Roggen et al. 2010. Collecting complex activity datasets in highly rich networked sensor environments. In *International Conference on Networked Sensing Systems*. IEEE.

[5] Doruk Sart et al. 2010. Accelerating dynamic time warping subsequence search with GPUs and FPGAs. In *International Conference on Data Mining*. IEEE.

[6] Jiaoyun Yang et al. 2010. An efficient parallel algorithm for longest common subsequence problem on GPUs. In *Proceedings of the World Congress on Engineering*, Vol. 1.

WLCSSLearn: Learning Algorithm for Template Matching-based Gesture Recognition Systems

Mathias Ciliberto
Wearable Technologies Lab
University of Sussex
Brighton, UK
m.ciliberto@sussex.ac.uk

Luis Ponce Cuspinera
University of Sussex
Brighton, UK
l.ponce-cuspinera@sussex.ac.uk

Daniel Roggen
Wearable Technologies Lab
University of Sussex
Brighton, UK
daniel.roggen@ieee.org

***Abstract*—Template matching algorithms are well suited for gesture recognition, but unlike other machine learning approaches there are no established methods to optimize their parameters. We present *WLCSSLearn*: an optimization approach for the WarpingLCSS algorithm based on a genetic algorithms. We demonstrate that WLCSSLearn makes the optimization procedure automatic, fast and suitable for new recognition problems even when there is no a-priori knowledge about suitable range of parameter values. We evaluate WLCSSLearn on three different datasets of gestures. We demonstrated that our method increased the accuracy and F1 score up to 20% compared to previous literature.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Human gesture recognition from sensor data has important application in healthcare, sports and human-computer interfaces (e.g. for gaming). Templates matching methods (TMMs) have been applied successfully to recognize complex human gestures. They can have comparable performance to other machine learning approaches with the advantage of requiring less training data [1]. Moreover, they can be used to extract richer information during the classification: for example, it is possible to use them to measure the distance in centimetre between two gestures by using the right encoding [2]. This is important in sports performance assessment or skills assessment [3].

Recently, the "Warping Longest Common Subsequence" (WLCSS) TMM method was introduced as a more robust alternative to e.g. Dynamic Time Warping when the sensor data is noisy [4]. It showed a 12% improvement in F1 score over DTW when recognizing activities of daily living from noisy data [4].

While other machine learning techniques present a well defined training algorithm, WLCSS, and more in general TMMs, do not have a standard training procedure. Training of WLCSS requires to: i) choose the templates to use for matching and ii) optimise its parameters. For the first, a possible approach is to choose the most representative template (MRT) among those available at training time, such in [4], [5]. The training instance with the highest similarity to all the training instance for a gesture is considered the most representative.

On the other hand, the optimization of the parameters is generally done manually for every specific application. For this reason, we introduce WLCSSLearn, a training algorithm for WLCSS. WLCSSLearn is based on an genetic algorithm and it has been developed to autonomously find the best set of parameters for WLCSS for a specific application with the minimum input from the user. We evaluate WLCSSLearn on three dataset of gestures collected using a variety of sensors.

II. Related work

A. Template Matching Methods

Template matching methods (TMMs) are algorithms that use a similarity measure to spot and/or recognize a template within a stream of data. Several TMMs have been proposed and applied for gesture recognition over the years: Dynamic Time Warping (DTW) has been employed for gesture recognition using accelerometers and inertial sensors in [6] and [13]. It computes the optimal match between two time series warped in time, trying to align them sample by sample. In [5], the Edit distance (ED) has been used in order to recognize human gestures in a car manufacturing environment. It measure the distance between two sequences as the minimum number of single-symbol operation (insertion, deletion and substitution) required to change one sequence into the other, assigning to each of them a cost. Longest Common Subsequence (LCSS) has been explored as similarity measure for visual gestures in [8]. A variation of LCSS, called WarpingLCSS (WLCSS) has been proposed [4] and evaluated in sport application [12]. WLCSS was shown to improve tolerance to noisy data [14] in comparison to DTW. Moreover, WLCSS can be implemented on limited memory devices, such as wearable devices [12]. It computes the LCSS between two sequence that can be warped in time, using three parameters: a reward (R), a penalty (P) and a noise tolerance threshold (ε) (see subsection III-A).

B. TMM Parameter optimization

As other machine learning approaches, the parameters of TMMs must be optimised during the training stage. However, unlike e.g. back-propagation as a standard algorithm for neural network training, there is no standard procedure to train TMMs. TMMs need one or more templates to match, and, in addition, each of them have different parameters to be optimized, generally for each specific application. Table I presents an overview of different optimization approaches for several TMMs. The overview is limited to gesture recognition

TABLE I: Overview of TMMs for gesture recognition and their optimization

Ref	TMM	Parameters	Data	Optimization method
[6]	DTW	Templates	3D Accelerometer data	Manual choice from dataset.
[7]	Custom DTW	Weights of custom DTW, templates	X,Y,Z coordinates from Kinect	Maximization of custom heuristic for the weights. Specifically recorded templates.
[1]	Euclidean distance, DTW, cross-correlation and Rce	3D Accelerometer data	Templates	Generation of templates by custom average.
[8]	DTW, LCSS	Templates	Images	Manual choice from dataset.
[9]	MPLCS (custom LCSS)	Matching thresholds. Templates	Trajectory from 3D camera	Manual setting of threshold. Specifically recorded templates.
[10] [5]	Edit distance	Insertion, deletion, substitution costs. Matching thresholds. Templates.	Trajectories from X,Y,Z coordinates	Computation of costs relatively to the encoding through heuristic. Computation of matching thresholds through heuristic. Templates chosen as the most representative within all the templates of each gesture class.
[11]	WLCSS	R, P, ε, and matching thresholds. Templates	Accelerometer data	Manual setting of parameters. Specifically recorded templates.
[12]	LM-WLCSS	R, P, ε, and thresholds. Templates	1D Gyroscope data 1D Accelerometer data EMG data	Manual setting of parameters R, P, ε, and matching thresholds. Manual choice of the templates from dataset.

applications, although they may be performed using different data such as videos, images or inertial data.

The related work shows that there is a wide variety of methods which have been suggested to optimise TMMs (Table I). DTW is used in [6], [1] and [8] but the three studies have all different methods for choosing the gesture templates: in [6] and [8] the templates are chosen manually by the authors from the dataset, while in [1] the templates are generated from the training set for each gestures by averaging all the instances for the same gestures using a custom heuristic. In [7] and [9] instead, a set of gestures is specifically collected to be used as templates. The former uses a custom DTW that requires also a set of weights as parameter; they are optimized by maximizing an heuristic specifically designed. The latter uses a modified version of LCSS: in this case, the authors set manually the matching thresholds.

In [10] and [5], the Edit distance is used as the TMM. The costs for the operations of insertion, deletion and substitution are computed according to the specific encoding used by the authors. This encoding transforms the hand displacement performed during a gestures in a stream of symbols. The matching threshold for each gesture class is computed with an heuristic using the average matching costs for that class, the standard deviation and a manually chosen parameter. Finally, the MRT within each gesture class is chosen for the matching. While this method may increase the recognition of the gestures of the same class of the template, it does not assure that all the other gestures are correctly discarded.

WLCSS is used as TMM in [11] and [12]. In both studies, the values of the parameters R, P, ε are set manually as well as the matching thresholds. Finally, the templates are selected by hand in [12], while in [11] a set of templates was recorded just for that end. However, this is not always possible especially when the recognition is done a posteriori without information about the setup of the data collection.

TMMs do not offer a standard training procedure that can be applied across different algorithms and different data. More specifically, when manually setting the parameters, it is often not possible to explore efficiently the entire space of possible solutions.

III. Methods

We chose to develop WLCSSLearn for WLCSS as it is a robust and powerful TMM against noisy data. Its applicability to different TMMs is discussed in V. In the following, the template matching algorithm and its parameters are described in detail. Then, WLCSSLearn is presented.

A. WLCSS

$$M(i,j) = \begin{cases} 0 & \text{if} \quad i \leq 0 or j \leq 0 \\ M(j-1,i-1)+R & \text{if} \quad f(S(i),T(j)) \leq \varepsilon \\ max \begin{cases} M(j-1,i-1) - P \cdot f(S(i),T(j)) \\ M(j-1,i) - P \cdot f(S(i),T(j)) \\ M(j,i-1) - P \cdot f(S(i),T(j)) \end{cases} & \text{if} \quad f(S(i),T(j)) > \varepsilon \end{cases} \tag{1}$$

WLCSS is a TMM capable of spotting and recognizing gestures which may be dilated or contracted in time and it is suitable for real-time low-power implementation [14] [12]. It uses dynamic programming in order to compute the similarity between a template and a stream of data. WLCSS is defined by the recurrence relation (1), where $M(i,j)$ is the matching score up to date to the i-th sample of the stream ($S(i)$) and the j-th sample of the template ($T(j)$). $f(S(i),T(j))$ is a function used to compute the distance between the stream and the template samples. WLCSS will increase the matching score by a reward (R) whenever samples of the template and sensor stream matches within a noise tolerance threshold (ε). Otherwise, it will find the best warping (contraction, dilation, or alignment as-is) between template and sensor stream. In this case, it penalizes the matching score by a penalty (P) proportional to the difference between the warped template and sensor stream.

In addition to R, P, and ε, WLCSS needs a set of thresholds $\mathbb{T}$, one for each of the k classes of gestures to recognise.

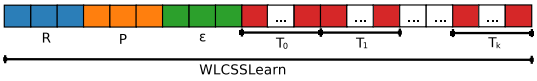


Fig. 1: Parameters encoded as genes. The thresholds values $(\mathbb{T}_0, \mathbb{T}_1, ..., \mathbb{T}_k)$ are encoded sequentially (red genes). The number of genes for every threshold is fixed, but the number of thresholds depends on the number of gestures to be recognized.

These thresholds are used to define whether a matching score M_k is high enough ($M_k > \mathbb{T}_k$) to be considered as a detected event for the corresponding k-th class. When the matching score is computed between an isolated gesture instance and the templates, it is called isolated recognition. In this case, when gestures matches with multiple templates, it can be classified unambiguously by defining an heuristic, such as:

$$class = \arg\max_k \frac{M_k - \mathbb{T}_k}{\mathbb{T}_k}$$

When the matching scores are computed between all the samples of a stream and the templates, it is called continuous recognition (see III-C).

The performance of WLCSS on each specific application are highly dependant on the set of parameters, R, P, ε and $\mathbb{T}$.

B. WLCSSLearn

WLCSSLearn is a training algorithm for WLCSS that aims to find the optimal parameters R, P, ε, and threshold values $\mathbb{T}$. The algorithm operates the training in isolated recognition. It uses a genetic algorithm (GA) to optimise that set of parameters. GAs are a class of algorithms that "evolve" a randomly initialized population of individuals in order to find the best fitting individual, maximizing a fitness function, using principle loosely inspired by biological evolution. GAs make it possible to search efficiently an optimum even when the space of possible solutions is large and when multiple parameters need to be optimized at the same time [15].

Each individual of the population encodes the parameters to evolve in a string of bits (genes). WLCSSLearn encodes these parameters using a variable number of genes as shown in Figure 1.

The GAs evolves the population through an iterative process according to a fitness score computed for each individual. At each iteration, after the fitness computation, a new population is generated in 3 steps:

- Selection. A subset of the individuals in the previous iteration's population is selected, according to the fitness score. The size of this subset is defined by the *rank* parameter.
- Crossover. The selected individuals are crossed over with a probability cp in order to create new individuals.
- Mutation. With a probability mp, a random selection of genes in the new population is mutated. It means that they are swapped between 1 and 0, and vice-versa.
- At each iteration, some of the best individuals from the previous population are copied to the new one in order to prevent a good individual to be destroyed through crossover/mutation. The number of these individuals is controlled by the parameter *elitism*.

Algorithm 1 WLCSSLearn - R, P, ε and thresholds optimization

Input: dataset, [gestures_classes], iterations
1: [templates], [gestures] = WLCSSLearnInit(dataset, [gestures_classes])
2: [pop] = initPopulation()
3: **for** $k = 1, ...,$iterations **do**
4: **for** p_i **in** [pop] **do**
5: (R, P, ε, [thresholds]) = decode(p_i)
6: p_i.fitness = computeFitness(R, P, ε, [templates], [gestures], [thresholds])
7: [pop] = evolve([pop])
8: (R, P, ε, [thresholds]) = getBestFit([pop])
Output: (R, P, ε, [thresholds])

The evolution process can end after a predefined number of iterations, as WLCSSLearn, or when there are no significant changes in the fitness score.

WLCSSLearn is detailed in Algorithm 1. The initialisation function WLCSSLearnInit extract all the gestures from the dataset as isolated instances to be used as training set. During this step, data from the null-class is added to the training set in order to provide for the lack of null class that derives from the extraction of the gestures. A subset of null class data is added proportionally to the percentage of the null class in the dataset. This null-class data is added in the form of fragments extracted randomly from the whole null-class signal. In order to prevent over-fitting to the null-class during the training, the length of each null-class fragment is the average length of the gestures in the dataset.

After the initialization, WLCSSLearn starts to evolve the population, decoding, at each iteration, each individual in R, P, ε and the thresholds in order to compute the fitness score.

It is clear how the choice of the function to compute the fitness score is fundamental in order to find the optima. WLCSSLearn uses a combination of F1 score and accuracy. It targets to maximize the product $F1 * accuracy$. The F1 score assures the best general performance over the entire training set. The accuracy guarantees that the best performance must be reached in all the recognition classes at the same time. This avoids the possibility that WLCSSLearn get stuck in a local optimum where the best F1 score is achieved only by a subset of the recognition classes.

C. Performance metrics for continuous template matching

Template matching algorithms, and more in general activity recognition, can work in isolated or continuous case. While WLCSSLearn performs the training in the isolated case, we evaluated it also in the continuous recognition.

In the isolated case, using the heuristic presented in III-A, it is possible to classify each instance unambiguously. It is possible to count true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) to evaluate the performance of the TMM. Finally, precision, recall and F1 score can be calculated.

In the continuous case, the matching score is computed between a template and the entire stream of data. For each template, a new matching score is computed at every new

TABLE II: List of activities for each dataset: (a) OPPORTUNITY, (b) Skoda, (c) HCI guided

(a)

Open Door	Close Door
Open Drawer	Close Drawer
Open Fridge	Close Fridge
Open Dishwasher	Close Dishwasher
Clean Table	Drink from Cup
Toggle Switch	Null

(b)

Open hood	Adjusting mirror
Close hood	Check trunk gaps
Open trunk	Open spare wheel box
Close trunk	Close spare wheel box

(c)

Triangle, pointing up	Infinity
Square	Triangle, pointing down
Circle	

sample of the stream. It means that for every activity in the ground truth, multiple matching scores are available as every activity lasts for more samples. This entails that the focus is more about the detection of an event. The continuous case presents several challenges, compared to the isolated case, when a TMM is used:

- multiple matching scores can be above a single acceptance threshold within the same single instance of an activity, potentially at different time
- the matching score for a specific template can go above and below the threshold multiple times within the same ground truth activity instance
- within the same ground truth activity, multiple templates can match multiple times
- a template can match with a reasonable delay after a ground truth event

An approach for performance evaluation for continuous activity recognition has been presented in [16]. However, this method is designed for activity recognition based on time fragments. It means that the activity recognition is based on temporal fragments that last for a specific amount of time. When the task is to detect an event, the fragment based evaluation is not suitable anymore as a "match" is unlikely to last for the entire ground truth activity. For these reasons, counting true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) is not trivial. We introduce a performance measure approach for TMM as a variation of [16]. We redefine TP, TN, FP and FN for TMM as described in the following:

TP is a single match occurred within a ground truth event or within a predefined window, in seconds, after the end of the ground truth event.

TN is a segment of null class, within of which no match has occurred. It is weighted by its duration in seconds.

FP Within the FP category, we distinguish between:

- FP-merge is a FP that occurs within a ground truth event which is not null-class. It happens when multiple events for the actual class are detected. It means that those event should be considered a single merged one.
- FP-insertion is a FP that occurs within a ground truth event when an event of the wrong class is detected.
- FP-substitution is a Fp within a ground truth event which is not a null class. It happens when an event of the wrong class is detected and no events of the correct class are detected.

FN occurs when a ground truth event which is not null-class is not detected at all.

With this approach, it is possible to compute accuracy, precision, recall and F1 score using the standard definition.

IV. Evaluation

A. Data

Three different dataset were used for the evaluation of WLCSSLearn. These datasets are made by sensor data collected using inertial platforms worn by users while performing human complex gestures or activities. Although they may not be specifically made for gesture recognition, they have been chosen because they are publicly available and they have been used in previous studies, making possible a comparison of the results. The details of each dataset and how it is pre-processed are explained in the following. We used the same pre-processing used in the reference study for each dataset in order to allow a comparison with the previous studies.

The *OPPORTUNITY Activity Recognition Dataset* is a multi-modal dataset recorded in a domestic environment [17]. The dataset includes the set of gestures displayed in table IIb. In this study, we used the data of a single acceleration channel from the lower arm sensor. The signal, originally sampled at 30 Hz, was filtered using a low pass filter with cut-off frequency of 10 Hz and then quantized in the range -64 to 63.

The *Skoda dataset* comprises gestures performed in car manufacturing, during a quality check routine [5]. The gestures were performed by users wearing a set of inertial platforms on the back, on the upper arms, on the lower arms and on the hands. Each platform integrated a 9-axis inertial unit providing its orientation in quaternions. Each users executed the gestures with the dominant hand, unless differently specified, or with both, when required. Due to the lack of information about which hand is dominant for each user, we focus on a subset of gestures that are commonly performed with the right hand (e.g. Adjusting the mirror for a left-hand driving car) or with both hands (e.g. Open/close hood). The set of gesture is presented in Table IIa. For this dataset, we applied the encoding of gestures presented in [2] in order to reduce the dimensionality of the data. With this encoding, human gestures are represented as a sequence of encoded symbols. Each symbol is obtained using a codebook of predefined vectors in order to sample the displacement of the hand during the gesture. We empirically chose spatial sampling and a 27 vectors codebook.

The *HCI dataset* is composed of a set of 5 hand gestures performed by a user wearing an inertial unit on the lower arm [4]. The gestures are performed moving the hand to form the shapes described in Table IIc. The 3 signals provided by the accelerometer (x, y, and z), originally sampled at 96Hz, were filtered with a low pass filter with cutoff frequency of 5 Hz. Finally the magnitude computed as $\sqrt{x^2+y^2+z^2}$ is used.

B. WLCSSLearn setup

We evaluated WLCSSLearn on the three datasets using the same set of parameters [1]. This allowed a comparison of the performance of WLCSSLearn although the three dataset presented different data.

As the choice of the template is fundamental for a TMM, we evaluated WLCSSLearn using two different heuristics for choosing the template:

- *Random choice* A template was chosen randomly at each test. This case represents the average performance of WLCSS picking a template by chance.
- *Most representative* The MRT for each gesture among the training instances is selected. Similarity between all the templates for a single gesture is computed using the LCS algorithm. This TMM was chosen as is quite robust with respect to variation in the performing speed of a gesture and it does not require any parameter (which would be subject to optimization, otherwise).

As the GA algorithm used by WLCSSLearn is a stochastic process, we repeated the optimization 5 times for each dataset, for every template choice method, for a total of 10 tests. WLCSSLearn optimizes the parameters in isolated recognition, with a percentage of null class added by WLCSSLearnInit during the training. After the optimization, we evaluate the obtained parameters in isolated recognition without null class (Iso_NoNull), isolated with the addition of null class (Iso_Null) fragments and in continuous recognition (Cont).

C. Results

We studied the evolution of the parameters in term of the algorithm complexity and recognition accuracy.

Figure 2 displays the values of the max fitness score at each iteration for every dataset. The values are averaged across all the tests we performed. It is visible how WLCSSLearn converges to an optimal value even before the limit of 500 iteration we set manually. However, in order to better understand the performances of the algorithms, looking only at the number of iterations is not enough. The number of individuals per iteration must be taken into account too. WLCSSLearn found the optimal values after 1390, 2290 and 2470 evaluations, on average for the three datasets respectively. In comparison, a brute force search for the same space of parameters would take $(2^5)^3$=32768 evaluations in the worst case, considering the R, P and ε made of 5 bits. Moreover, this worst case does not take into account the thresholds, as they are variable for each application, while WLCSSLearn optimizes them simultaneously.

Table III presents an overview of the results in term of accuracy and F1 score, for every combination of template choice method and recognition type (Iso_NoNull, Iso_Null, or Cont). These values are obtained as the average of accuracy and F1 score, on all the classes, for every set of 5 tests. The last column presents the best results for the same dataset and, where possible, the same algorithm, for every dataset. These values

[1]Population=10, Iteration=500, Genes=5 for R, P, ε, 10 for each threshold, *rank*=3, *elitism*=1, *cr*=0.3, *mp*=0.1

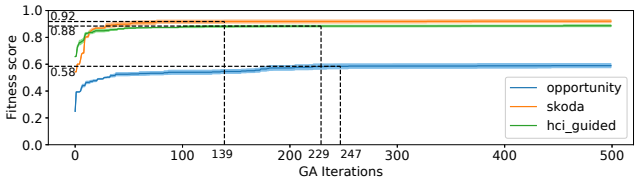


Fig. 2: Max fitness score ($F1score * accuracy$) evolution for 3 datasets during training using WLCSSLearn. The values are averaged across all the training test.

are placed as general reference, but a specific comparison is often not possible due to the lack of information about the process for the reference studies.

It is possible to notice how Iso_NoNull and Iso_Null are the scenarios with the highest scores. This is due to the nature of the TMM and the great variability in the null class for two of the three datasets (OPPORTUNITY and Skoda). The HCI dataset has a lower variability in the way the gestures are performed as well as in the null class between each gesture.

Although in the continuous recognition the performance is generically lower, WLCSSLearn led to increased recognition performance compared to previous literature in almost all the tests. The OPPORTUNITY dataset, which is the most challenging one, is the one of the cases where the performance are not matched to the previous literature. In this case, it is important to notice that the accuracy is increased thanks to WLCSSLearn. Moreover, we suspect that the lower F1 score is due to two main differences between our study and [4], taken as reference. The first is that we used the data from a single channel of acceleration instead of the magnitude of all the three axis. Also, we used our own method for evaluate the performance of the TMM, as [4] do not provide any insight on the approached used. Similarly, for the Skoda dataset, the accuracy increased by 18%, while the lower F1 score is probably due to difference in the input data. In [18], the results are obtained as the fusion of multiple templates matching. Also, DTW is used as TMM instead of WLCSS, making the comparison less effective.

From the table, it is also visible how the choice of the templates influences the performance of the recognition. Especially when the variability of the gestures is wide, as in OPPORTUNITY, a random choice of the template can decrease the performance by 11% on F1 score, compared to choosing the most representative template.

V. Discussion & Future Work

The main drawback of the algorithm is the time consuming optimization process. For the three dataset, OPPORTUNITY, Skoda and HCI, WLCSSLearn took respectively 7h45min, 25min and 1h30min on average to complete 500 iterations, on a desktop PC with a 3.40 GHz CPU. The main bottleneck is the calculation of WLCSS for computing the fitness scores. This issue escalate dramatically with the increase of the number and of the length of the templates. A possible solution is to use faster way to compute WLCSS: a highly parallelized computation (e.g. using GPUs) would allow to compute multiple matching

TABLE III: Comparison of F1 scores of WLCSS on the three datasets. The parameters obtained with WLCSSLearn are evaluated in isolated without null class (Iso_NoNull), isolated with null class (Iso_Null), and continuous (Cont). The results are compared to the best results for each dataset in the literature.

Dataset	Template selection	WLCSSLearn						Previous Studies		
		Iso_NoNull		Iso_Null		Cont				
		Accuracy	F1 score	Accuracy	F1 score	Accuracy	F1 score	Accuracy	F1 score	Ref
OPPORTUNITY	Random choice	0.92 ± 0.01	0.53 ± 0.04	0.94 ± 0.01	0.43 ± 0.02	0.72 ± 0.07	0.37 ± 0.04	0.50 ± 0.06	0.48 ± 0.05	[4]
	MRT (LCS)	0.94 ± 0.00	0.61 ± 0.05	0.97 ± 0.00	0.58 ± 0.03	0.83 ± 0.04	0.48 ± 0.03			
Skoda	Random choice	0.98 ± 0.02	0.90 ± 0.09	0.99 ± 0.01	0.88 ± 0.10	0.94 ± 0.02	0.42 ± 0.06	0.76	0.58	[5] [18]
	MRT (LCS)	0.99 ± 0.01	0.97 ± 0.04	1.00 ± 0.00	0.95 ± 0.05	0.94 ± 0.01	0.44 ± 0.02			
HCI guided	Random choice	0.93 ± 0.07	0.83 ± 0.14	0.96 ± 0.03	0.83 ± 0.14	0.79 ± 0.08	0.88 ± 0.05	0.74	0.72	[4]
	MRT (LCS)	0.98 ± 0.01	0.95 ± 0.03	0.99 ± 0.00	0.95 ± 0.03	0.94 ± 0.02	0.97 ± 0.01			

scores at the same time, increasing drastically the performance of WLCSSLearn.

In our test, we set manually the values of the WLCSSLearn parameters Population, Iteration, Genes, *rank*, *elitism*, *cr*, and *mp*. We aim to characterize more in details such parameters in future studies.

We also did not evaluated different way of pre-process data as we aimed to a comparison with previous study, but we intend to exploit this in future extensions of this research as it may really improve the performance of the TMM.

Finally, we found that the choice of the template may influence the performance of WLCSSLearn. For this reason, we realize that WLCSSLearn can potentially be used for generating an optimized template instead of picking one from the training set. In this case, an evolution strategy may be used instead of a genetic algorithm, as the values of the genes for each template may have to vary within a set of discrete values. For example, this would be the case for generating a template for the encoded Skoda dataset which represent the samples with values in the range 0-27.

The generation of the template can be useful to adapt WLCSSLearn for other TMM as well. We chose WLCSS as it is a robust TMM with several parameters. With the right fitness function, our algorithm can be adapted for generating the template for DTW or to compute the cost of insertion, deletion and substitution for the Edit distance.

VI. Conclusion

We presented WLCSSLearn, an automated learning algorithm for WLCSS. Using a genetic algorithm, WLCSSLearn is able to learn and optimize the parameter R, P and ε required by WLCSS. Moreover, it can find the best matching thresholds for each recognition class. We demonstrated its applicability using three dataset of human gestures. The results show that our method increase the accuracy and F1 score by up to 20% compared to a manual optimization, but more specifically it is automatic and it does not require any previous knowledge on the data. Although there is still space for improvement, WLCSSLearn is a robust foundation for a well performing learning method for TMM.

References

[1] J. Margarito, R. Helaoui, A. M. Bianchi, F. Sartor, and A. G. Bonomi, "User-independent recognition of sports activities from a single wrist-worn accelerometer: A template-matching-based approach," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 4, 2016.

[2] M. Ciliberto, L. P. Cuspinera, and D. Roggen, "Complex human gestures encoding from wearable inertial sensors for activity recognition," 2017.

[3] A. Khan, S. Mellor, E. Berlin, R. Thompson, R. McNaney, P. Olivier, and T. Plötz, "Beyond activity recognition: Skill assessment from accelerometer data," *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2015.

[4] L. V. Nguyen-Dinh, D. Roggen, A. Calatroni, and G. Tröster, "Improving online gesture recognition with template matching methods in accelerometer data," *International Conference on Intelligent Systems Design and Applications, ISDA*, 2012.

[5] T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and G. Tröster, "Wearable activity tracking in car manufacturing," *IEEE Pervasive Computing*, vol. 7, no. 2, 2008.

[6] G. Niezen and G. P. Hancke, "Gesture recognition as ubiquitous input for mobile phones," *Devices that Alter Perception*, 2008.

[7] T. Arici, S. Celebi, A. S. Aydin, and T. T. Temiz, "Robust gesture recognition using feature pre-processing and weighted dynamic time warping," *Multimedia Tools and Applications*, vol. 72, no. 3, 2014.

[8] A. Kuzmanic and V. Zanchi, "Hand shape classification using dtw and lcss as similarity measures for vision-based gesture recognition system," in *EUROCON, 2007. The International Conference on" Computer as a Tool"*, IEEE, 2007.

[9] D. Frolova, H. Stern, and S. Berman, "Most probable longest common subsequence for recognition of gesture character input," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, 2013.

[10] T. Stiefmeier, D. Roggen, and G. Tröster, "Gestures are strings: efficient online gesture spotting and classification using string matching," *Proceedings of 2nd International Conference on Body Area Networks*, 2007.

[11] M. Hardegger, L.-V. Nguyen-Dinh, A. Calatroni, G. Tröster, and D. Roggen, "Enhancing action recognition through simultaneous semantic mapping from body-worn motion sensors," in *Proceedings of the 2014 ACM International Symposium on Wearable Computers (ISWC '14)*, ACM Press, 2014.

[12] D. Roggen, L. P. Cuspinera, G. Pombo, F. Ali, and L.-V. Nguyen-Dinh, "Limited-Memory Warping LCSS for Real-Time Low-Power Pattern Recognition in Wireless Nodes," *European Conference on Wireless Sensor Networks*, vol. 8965, 2015.

[13] B. Hartmann and N. Link, "Gesture recognition with inertial sensors and optimized DTW prototypes," in *2010 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, 2010.

[14] L.-V. Nguyen-Dinh, A. Calatroni, and G. Troster, "Robust Online Gesture Recognition with Crowdsourced Annotations," *Journal of Machine Learning Research*, vol. 15, 2014.

[15] C. M. Fonseca, P. J. Fleming, *et al.*, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization.," in *Icga*, vol. 93, pp. 416–423, 1993.

[16] J. A. Ward, P. Lukowicz, and H. Gellersen, "Performance metrics for activity recognition," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 1, 2011.

[17] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, J. Del R. Millàn, J. Del, and R. Milì, "Collecting complex activity datasets in highly rich networked sensor environments," *INSS 2010 - 7th International Conference on Networked Sensing Systems*, no. 00, 2010.

[18] T. Stiefmeier, *Real-Time Spotting of Human Activities in Industrial Environments*. No. 17907, 2008.

Collecting a dataset of gestures for skill assessment in the field: a beach volleyball serves case study

Mathias Ciliberto
m.ciliberto@sussex.ac.uk
Wearable Technologies Lab,
University of Sussex
Brighton, UK

Luis Alejandro Ponce Cuspinera
luisalejandroponce.cuspinera@dyson.com
Dyson Institute of Engineering and Technology
Malmesbury, UK

Daniel Roggen
d.roggen@ieee.org
Wearable Technologies Lab,
University of Sussex
Brighton, UK

ABSTRACT

Activity and gesture recognition from wearable sensors data can be used for skill assessment in order to gauge the capability of a user at performing a task. As many other problem of automatic classification, gesture recognition relies on annotated data for the training of the classification system and to gather a set of gestures for the assessment. The collection of a multi-sensors dataset for this goal can be challenging, especially when it is performed in the field rather than in a more controlled environment such as a laboratory. In this paper, we present the collection of a beach volleyball gestures dataset in the field. The resulting dataset is made publicly available to the community and it includes 585 annotated gestures, collected by 10 users, with 4 wearable inertial sensors per user. In addition, we also provide a list of lessons learnt, suggestions and guidelines to improve future data collections in the field.

KEYWORDS

Dataset; Sport; Beach Vollyball; Gesture recognition; Skill assessment

ACM Reference Format:
Mathias Ciliberto, Luis Alejandro Ponce Cuspinera, and Daniel Roggen. 2021. Collecting a dataset of gestures for skill assessment in the field: a beach volleyball serves case study. In *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers (UbiComp-ISWC '21 Adjunct), September 21–26, 2021, Virtual, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3460418.3479355

1 INTRODUCTION

Skill assessment is the procedure to gauge the capability of the user at performing a certain task. In the field of wearable computing, gesture recognition can be deployed for skill assessment through the application of pattern recognition methods on inertial sensors data in order to measure the similarity between streams of data associated to different gestures.

UbiComp-ISWC '21 Adjunct, September 21–26, 2021, Virtual, USA

ACM ISBN 978-1-4503-8461-2/21/09…$15.00
https://doi.org/10.1145/3460418.3479355

Gesture recognition relies on datasets of well annotated gestures to detect, recognize and then assess the quality of the movements. A dataset is required for each specific assessment application. The datasets for this goal can also include a wide variety of sensors in order to capture the complexity of human movements [8]. For these reasons, the need of data is a continuous challenge [16].

Collecting a dataset for skill assessment can be a complicated task due to factors as preparing the environment, choosing and setting up the sensors, as well as accurately define the data collection protocol. These tasks can be even more challenging when the data collection is required by the application domain to be performed in the field rather than in an artificial environment. For example, this is the case of data recorded for industrial applications, surgical skill assessment, and in outdoor sports.

In this paper, we illustrate our process of collecting a multi-sensor multi-user dataset of gestures through a case study of beach volleyball gestures. We selected this application scenario because, as sport, it can greatly benefit from skill assessment in the future, while also presenting the main challenge of being an outdoor sport played in an environment that could be cumbersome for a data collection. Creating a new dataset allowed us to:

- have full control on the set and positioning of the sensors, enabling, for example, the encoding of the gestures for recognition as presented in [17]. By including a wide set of sensors, we increase the applicability of this dataset for the evaluation of skill assessment systems;
- customize the data collection protocol in order to focus on specific gestures. In this case, we focus on the serving gesture because i) it is the first movement of every rally which means it can greatly affect every consecutive action; ii) it is not affected by any previous action of the game, allowing players to have full control and freedom in performing the movement.

The contributions of this paper are:

- a publicly available dataset of beach volleyball serves gestures recorded by 10 users in 3 session, with multiple wearable inertial sensors. The serves gestures are annotated with 4 types of serves. The dataset also include data recorded during several 2-vs-2 games recorded with the same setup.
- a case study of annotation recovery for one of the session, due to a failure in the camera system set up for a-posteriori data labelling.
- a list of lesson learnt, suggestions and guidelines aimed at simplifying and improving future data collection performed in the field with wearable sensors.

2 BACKGROUND

Beach volleyball is a sport that comprises a limited set of very different basic movements (serving, passing, hitting) that can be combined during a game between two teams of two players. Several studies explored the technical aspect of beach volleyball gestures and their effectiveness on the game [10, 13] through the visual and manual analysis of videos in what is called *notational analysis*. This method has been used to investigate the variations of techniques among different kind of players [18], the dependency of the different techniques to the success of the game [5], to evaluate a single technique [6], to understand the biomechanics of a specific technique [9, 14].

From the literature, it has also emerged the high correlation between a good quality in the basic techniques and the success in the game [10]. For this reason, it is important for the athletes to achieve the best quality in the performing of the game technique.

The analysis of the beach volleyball games has been performed used mainly the video recording and the tracking of the players [4, 12]. However, setting up a controlled environment with cameras and tracker for a sport that is mainly played outdoor is often not possible. A first attempt to use wearable sensors to autonomously recognize the beach volleyball serves and other gestures has been studied in [3, 7].

Unfortunately, despite a wide set of videos publicly available for beach volleyball training and games, we were unable to retrieve a dataset of playing gestures recorded using wearable sensors. The few studies investigating wearable sensors for beach volleyball application did not make their datasets available to the community.

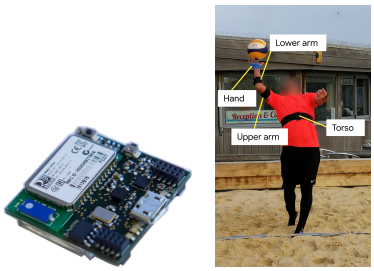
3 EQUIPMENT

3.1 BlueSense

The data from the players were recorded using a set of wearable in-house developed inertial sensing platform, called BlueSense. BlueSense is a sensor research platform developed for wearable and IoT application [15]. Its small size of just 30x30mm allows to be worn with minimal to none hindering of the user while performing gestures. The platform is instrumented with a 3-axis inertial unit (IMU) including an accelerometer, a gyroscope and a magnetometer. BlueSense is capable of sampling raw data at 1 KHz, or computing 3D orientation on-board up to 500 Hz, using a variation of Madgwick's algorithm [11]. The data can be logged on a locally stored microSD, or streamed over the integrated Bluetooth module. The Bluetooth module is also used for streaming of commands (e.g. start/stop recording, set timestamps, etc.). The platform also includes an on-board real time clock (RTC) with minimal time deviation of 0.6ppm, allowing multiple nodes to stay synchronised for the duration of the data collection, enabling an easier merge of data and annotation once the collection is complete.

For this data collection, we set the sampling rate to 500Hz as we recorded both raw and orientation data. We also used the integrated microSD as streaming data over Bluetooth would results in possible loss of samples. The commands were sent to the sensors using a specifically developed Android application [1].

[1] https://play.google.com/store/apps/details?id=net.danielroggen.bs2mgr



(a) ***BlueSense*** **(b)** ***Sensors placement***

Figure 1: BlueSense platform and sensors placement on the players.

3.2 Sensor placement

A set of 4 BlueSense per player was placed on the torso, the dominant upper arm, lower arm and hand of the players, thanks to a set of strap bands as shown in Figure 1b. The straps were made by using an an elastic band and industrial velcro. This ensured a tight placement of the sensor on the body of players minimising accidental vibrations of the platform.

By including the torso and the dominant arm, the orientation data can be used to encoded gestures as explained in [17] or to animate a 3D model like the one described in [1].

3.3 Cameras

The entire data collection was recorded using 3 wide-angle video cameras for a-posteriori annotation. The cameras were placed as shown in Figure 2a. These cameras were chosen because of their wide field of view, allowing a comprehensive recording of the entire court (see Figure 2b-2d). They were set to record at 1280x720 pixels and 120fps. While they were able to record at higher resolution, we opted to reduce the frame size in exchange for a higher framerate that could be more useful when looking at fast movements as beach volleyball serves.

4 DATA COLLECTION

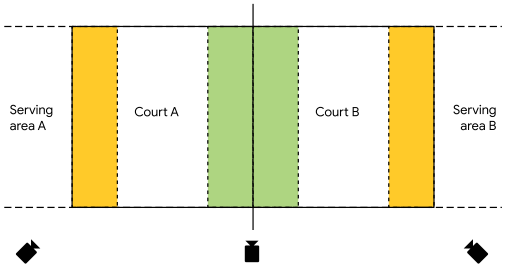
We accurately defined the data collection protocol for the participants in order to reduce the time and effort required.

4.1 Participants recruitment

The first step of the collection was the choice of the participants. In order to minimise errors while performing the protocol, we needed players that would be able to serve and perform specific game gestures on command. The participants were selected according with their level of expertise from the community of players at our closest beach volleyball venue. The players' level varied between semi-professional to former professional players. We did not constrain gender or physicality. We selected 10 players, 5 females and 5 males, divided in 3 sessions of 4 players. 2 female players were present in two sessions. 9 players were right-handed and 1 was left-handed.

4.2 Court Setup

The court was setup accounting for one serving areas and two landing areas on each side of the net, as shown in Figure 2. The



(a) *Setup of the court*

(b) *Left view* (c) *Center view* (d) *Right view*

Figure 2: Setup of the court during data collection and camera view examples. In Figure 2a, the green areas are the aimed landing area for short serves, while the orange areas are for long. Players can serve from any point in the serving area of each court, aiming at the landing areas of the opposite side. The three wide angle cameras, placed as shown by the black symbol, offer a comprehensive coverage of the entire court for annotation. An example of the camera view is displayed at the bottom, respectively for left 2b, center 2c and right cameras 2d.

two landing areas were delimited by little cones and differentiated between short and long serves. We empirically set the landing areas to 1.5m deep. They were delimited by little cones that were removed for the gaming phase of the data collection.

The net height was set to 2.33m as average between men (2.43m) and women (2.24m) official heights.

4.3 Data collection protocol

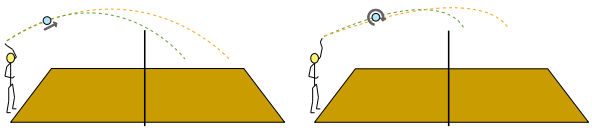
The players were instructed to follow a precise but flexible protocol during each session of the data collection. The protocol was developed in order to record more controlled serves gestures as well as free in-game movements. 4 players would perform such protocol in a 2 hours session, excluding some time for setup and synchronisation.

The protocol was divided in two main parts: in the first part, the participants were asked to perform specific serves individually, as they were in a training session. In the second part, the participants were allowed to freely play few sets following the standard 2-vs-2 rules of beach volleyball.

We identified 2 main type of serves to consider, graphically presented in Figure 3 and described in the following. According with the force applied by the player and therefore the length of the trajectory, each type can be short or long:

Float serve In this serve, the player hits the ball at the centre with a flat and stiff hand. This cause the ball to float in the air with a small unpredictable side-to-side movements.

Top spin In this serve, the player perform a wider movement with the arm, compared to the float serve, and he hits the ball at the top-centre with a spinning forward movement of the hand. The ball to rotate forward during the flight and the trajectory result more curved compared to a float serve.



(a) *Float serves* (b) *Top spin serves*

Figure 3: Example of float and top spin serves with trajectories, for long (orange) and short (green) serves.

Table 1: Data collection protocol. The columns indicates in order: the step number, the duration of the step in number of repetitions or minutes, the activity required in each step and a brief description.

Step No.	R./D.	Activity	Description
1	8 mins	Warm up	The participants could warm up freely with the ball. This was necessary to reduce the risk of injuries and it also allowed the players to get familiar with the sensors while playing.
2	12x	Long float serves	The players placed themselves at the end of the court in the serving area. They were asked to serve 12 long float serves. Each player must count its own serves.
3	1 min	Rest	
4	12x	Long top spin serves	They were asked to serve 12 long float serves. Each player must count its own serves.
5	1 min	Rest	
6	12x	Short float serves	They were asked to serve 12 long float serves. Each player must count its own serves.
7	1 min	Rest	
8	12x	Short top spin serves	They were asked to serve 12 long float serves. Each player must count its own serves.
9	1 min	Rest	
10	Rest of time	Free game	The players were asked to freely play for the remaining time of the 2 hours session. The free game respected the standard 2-vs-2 rules, with sets up to 21 points and changing side every 7-points. The players teams stayed the same for this entire part.

Once the players were instrumented and the court was prepared, the recording on the sensors and the cameras started. Then the protocol begin with a synchronisation step comprising 3 hand claps in front of the cameras. This allowed the synchronization of all the involved data sources for an accurate annotation (see Section 5). The protocol proceeded as presented in Table 1.

Finally, the players were asked to repeat the synchronisation step before to be de-instrumented from all the sensors.

Players were asked to follow the protocol at their best. However, we left room to correct possible mistakes during the data collection such as the rest breaks and additional time in between the two parts of the protocol.

5 ANNOTATION

The annotation was performed off-line using a previously in-house developed software. The software supports multiple video and data sources, as well as multiple annotation tracks. All the data sources can be synchronised within the application. A screenshot of the annotation software is shown in Figure 4.

For this first iteration of the dataset, we labelled only the part of the data collection including the controlled serves of the players. The labels for each of the four serve types were introduced manually for each player. The manual annotation allowed to correct possible mistakes between serve requested by the protocol and the one actually performed by the players.

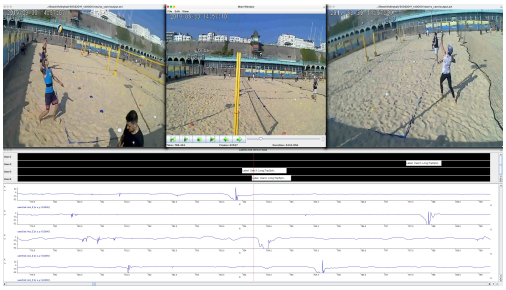


Figure 4: Annotation tool used to label the data. The videos on the top are used to synchronise all the source and to analyse the movements. The annotation are then inserted in the black tracks (one track per player), in the centre of the screen. On the bottom, a downsampled signal from one channel of the BlueSense's IMU is shown to check the consistency of the sensor data with the movements. One signal is displayed for each player.

The annotations were then exported from the software and merged with the data of each player singularly. The data from the multiple sensors, torso, upper arm, lower arm and hand, for each player were synchronized thanks to the timestamps associated with the sampled data from the BlueSense.

6 DATASET ISSUES

Unfortunately, during the data collection we encountered two main issues that could affect the quality of the data: i) the missing videos for one of the session due to a failure of the cameras, ii) the malfunctioning of two sensors for one participant resulting in difficulties in annotating such data.

6.1 Missing videos

During one of the sessions, the cameras failed after few seconds of recording. This resulted in the impossibility of annotating the data as described in Section 5. While we could not figure out the reason for the failure, we managed to recover the annotation thanks to the precise protocol.

We could indeed analyse the data knowing the order of the steps each player followed and therefore annotate the serving segments net of possible mistakes made by the players in the serving.

Figure 5 shows the steps we performed to annotate the data:

(1) We started by evaluating the raw data provided by the gyroscope of the BlueSense placed on the lower arm. We chose this sensor for its sensibility and for not being affected by gravity (accelerometer) and external magnetic fields (magnetometer).

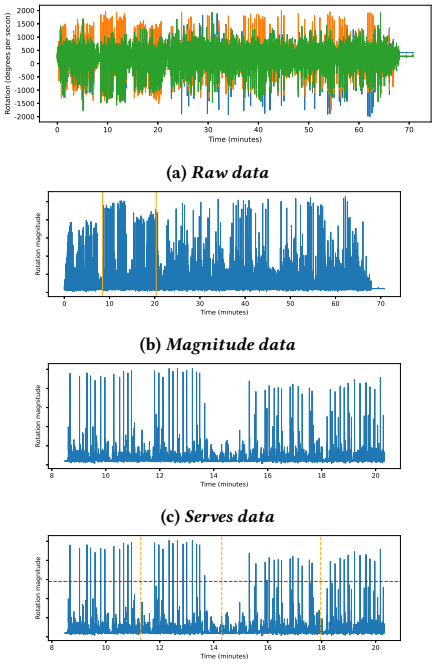


Figure 5: Example of data annotation recovery procedure for one user. The raw data from the gyroscope of the BlueSense positioned on the lower arm are plotted in the top Figure 5a. The magnitude of the signal is then computed in order to identify the section of the data regarding the serves. This section is identified in Figure 5b by the two orange dashed lines. Figure 5c displays an enlargement of the section of data pertinent to the serves. The four sections for each one of the serves type are then visually individuated and represented in Figure 5d as separated by the vertical orange dashed lines. A threshold displayed with an horizontal red dashed line is then used to select the peaks in the data that would be selected as serves.

(2) Considering the serve gestures, we assumed them having higher energy compared to slower gestures such as passing. Therefore we computed the magnitude of the three channel data as $\sqrt{x^2 + y^2 + z^2}$ as a measure of the quantity of rotation.

(3) We identified the part of the data referring to the serves in the protocol. Starting from the synchronization claps that can be identified at the beginning of the time series, we found a section of recurring pattern divided in four subsections, one per serves type in the protocol. The recurring patterns were identified by the highest peaks in the section. The peaks in the signal resulted by the impact of the hand with the ball at each serve.

(4) We empirically defined a thresholds in order to filter only the peaks referring to serves. We then defined the boundaries of each section corresponding to long floats, long top spin, short float, short top spin serves. As expected according to

the protocol and considering the threshold, around 12 peaks were contained in each section.

(5) For every peak in each section, we selected a segment of 450 samples that we then annotated with the corresponding serves type. The length of the segments was defined as average length of the already annotated serves.

We repeated this process for all the four players in the session with missing videos. In order to verify the correctness of our procedure, Figure 6 presents the segments annotated through this method for one player. This player participate in session 3 as well as in session 1 that was correctly annotated using the videos, allowing to compare the video-annotated segments and manually annotated segments for the same participant. In the figure, it is possible to notice how the manually annotated segments in colours have similar shape to the most representative segment of the video-annotated annotated ones.

This procedure allowed to annotate the serving data for an entire session, comprising of 4 players for a total of 16 sensors data that would otherwise be unusable for supervised gesture recognition.

6.2 Sensors errors and reliability

During an initial session, we discovered the failures of two BlueSense on the same player. The failure were due to a bad configuration of the sensors themselves: in one case, the microSD was not properly formatted, resulting in missing data from the torso sensors. The second issue was due to a bad configuration of the output format of the data from the hand sensor that made them non-readable. In the final release of the dataset, we still include the annotated data from the upper arm and the lower arm of the same user, as they were still available.

7 DISCUSSION

Throughout the dataset recording and thanks also to the issues that we encountered, we were able to define a series of guidelines for future data collection that we want to discuss.

Defining an accurate protocol is fundamental for the success of the data collection as well as for handling possible errors of the participants and failure of the hardware. It is important to plan every single step of the data collection in advance.

Invest in high quality hardware equipment. The cameras failure was an issue that could be prevented by investing in more reliable and well known products. On the other hand, the wearable sensors were more reliable with a single issue due to a mistake in the configuration.

Extensively test any hardware. We conducted several tests of the cameras and the sensors: this was important, for example, to set the sensitivity range for the IMU of the BlueSense for this specific application. Unfortunately, in all previous tests the cameras did behave correctly without displaying any problem: a confirmation that even extensive testing sometimes cannot be enough.

Account for missing data. In some case, such as the missing videos, the data can be recovered through a manual inspection or other approaches [2]. In other cases, such as for sensors failures, some data can be irrecoverably lost. This can be overcome by including a high number of participants.

Keep the participants engaged. During the recording of the dataset, we found that players were quite keen to participate as they could play free of charge for some time, once they completed the first part of the protocol. This is an important aspect to consider, as the quality of a dataset can be greatly affected by the lost of commitment by the users [2].

Testing the data collection protocol would help to assess the engagement of the participants as well as the intensity of the protocol. If the protocol is too energy demanding for the participant, the quality of the gestures can decrease after few repetitions. More importantly, if the protocol is too intense can even cause injuries in the players. For this reason, we included a mandatory warm-up section at the beginning of every session, as well as well defined rest interval between exercises.

Testing the protocol in advance can also help to optimize the efforts of the researchers performing the data collection. For example, just after few iterations of the protocol, we were able to better spend our effort in setting up the courts, the cameras and the sensors, by reducing the discomfort of the player when the weather conditions were not optimal. This is visible in Table 2 as the total amount of data we collected increases in time with the progression of the sessions.

Reduce the discomfort of the participants. In addition to improving the protocol, the discomfort of the participants could be minimised even further with a better placement of the sensors. In the effort of minimising the sensors noise by reducing accidental vibrations, we built the straps quite tight and potentially discomfortable. Additionally, the usage of strong industrial velcro resulted in some irritation on the skin of some players. This could be improved in future iterations for example, by building a full vest integrating the sensors or even by fusing the sensors in the fabric.

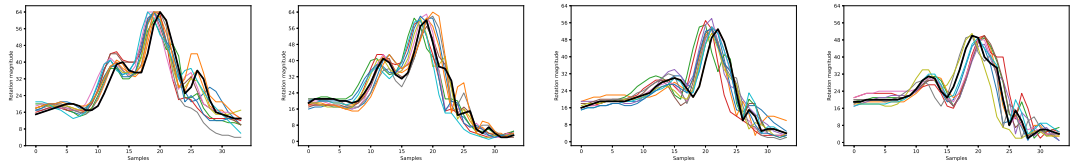
Assess the environment for the data collection. In our case, a specific setting such as the beach volleyball court was required. Unfortunately, being based in the United Kingdom affected our efforts as the weather reduced the availability of the courts. We strongly advise to take the environment conditions into consideration when planning a data collection in the open. For example, we eventually realised that setting the data collection in more sunny countries would have greatly increased the availability of courts and people, with little impact on the costs.

8 CONCLUSION

We illustrated the creation of a dataset of beach volleyball serves gestures for skill assessment. We described the process of data recording, as well as data labelling and recovering of data with missing annotation. Finally, we discussed a set of guidelines to improve future collection of gestures data in the field.

The resulting dataset is summarised in Table 2: net of sensors failures, it comprises a total of 585 annotated servers, divided in 147 long float, 140 long top spin, 155 short float and 143 short top spin serves. In term of time, the dataset includes a total of 2 hours and 12 minutes of annotated data and a total time of 11 hours and 45 minutes of sensors data. These times account for each users in the same session separately.

The dataset is freely available to the community at
https://ieee-dataport.org/open-access/wearlab-beach-volleyball-serves-and-games



(a) *Long float* (b) *Long top spin* (c) *Short float* (d) *Short top spin*

Figure 6: Comparison of artificially extracted and video-annotated gestures. A player participated in two sessions, one with videos and one with missing videos. Thanks to this, it is possible to compare the segments of the gestures extracted from the sensors data (over-imposed in colours) and a video-annotated gesture segment (in black), for each serves type. The video-annotated gesture is selected as MRT among the segments for each serve type.

Table 2: Beach volleyball dataset summary. The columns indicate respectively: the user_ID, the gender of the player, the handedness, the number of each serves with lf = long float, lts = long top spin, sf = short float, sts = short top spin. The number of errors is calculated as the number of serves of the wrong type performed by each player. ann_time is the amount of time of annotated data and total_time indicated the total amount of recorded data for each session. Finally, the sensors columns reports the sensor data that are available for each user: t = torso, u = upper arm, l = lower arm, h = hand.

user_ID	gender	hand	session	lf	lts	sf	sts	errors	ann_time	total_time	sensors
01	F	R	1	16	12	11	9	6	00:12:08	00:41:16	-ul-
02	M	L	1	12	10	16	15	5	00:11:28	00:40:30	tulh
03	M	R	1	8	13	14	12	2	00:11:08	00:40:34	tulh
04	F	R	1	13	12	13	14	2	00:11:06	00:40:24	tulh
05	M	R	2	12	11	12	13	2	00:09:58	01:00:21	tulh
06	M	R	2	12	12	12	13	2	00:10:02	01:04:57	tulh
07	M	R	2	11	10	12	11	2	00:09:42	01:05:27	tulh
08	F	R	2	12	12	12	11	0	00:09:27	01:05:36	tulh
01	F	R	3	13	12	14	12	0	00:12:08	01:12:10	tulh
09	F	R	3	12	12	13	11	0	00:11:40	01:10:49	tulh
10	F	R	3	14	12	14	11	0	00:11:15	01:12:37	tulh
03	F	R	3	12	12	12	11	0	00:11:48	01:10:49	tulh
Total				147	140	155	143		02:11:55	11:45:34	

ACKNOWLEDGEMENTS

This work received support from the EU H2020-ICT-2019-3 project "HumanE AI Net" (project number 952026).

REFERENCES

[1] Mathias Ciliberto, Luis Ponce Cuspinera, and Daniel Roggen. 2018. Complex human gestures encoding from wearable inertial sensors for activity recognition. In *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*. ACM.

[2] Mathias Ciliberto, Lin Wang, Daniel Roggen, and Ruediger Zillmer. 2019. Drinking Gesture Recognition from Poorly Annotated Data: A Case Study. In *Human Activity Sensing*. Springer International Publishing, 71–89.

[3] L. Ponce Cuspinera, Sakura Uetsuji, F. J. Ordonez Morales, and Daniel Roggen. 2016. Beach volleyball serve type recognition. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. ACM.

[4] Gabriel Gomez, Patricia Herrera López, Daniel Link, and Bjoern Eskofier. 2014. Tracking of Ball and Players in Beach Volleyball Videos. *PLoS ONE* 9, 11 (Nov. 2014), e111730.

[5] Mikko Häyrinen and Kostas Tampouratzis. 2012. *Technical and tactical game analysis of elite female beach volleyball.* Number 37. 1–45 pages.

[6] José Manuel Jiménez-Olmedo, Alfonso Penichet-Tomás, Sheila Saiz-Colomina, José A. Martínez-Carbonell, and Marcelo A. Jove-Tossi. 2012. Serve analysis of professional players in beach volleyball. *Journal of Human Sport and Exercise* 7, 3 (2012), 706–713.

[7] Thomas Kautz, Benjamin H. Groh, Julius Hannink, Ulf Jensen, Holger Strubberg, and Bjoern M. Eskofier. 2017. Activity recognition in beach volleyball using a Deep Convolutional Neural Network. *Data Mining and Knowledge Discovery* 31, 6 (Feb. 2017), 1678–1705.

[8] Joseph J. LaViola. 2013. 3D Gestural Interaction: The State of the Field. *ISRN Artificial Intelligence* 2013 (Dec. 2013), 1–18.

[9] Roberto Lobietti. 2009. A review of blocking in volleyball: from the notational analysis to biomechanics. *Journal of Human Sport and Exercise* 4, 2 (2009), 93–99.

[10] A.B. Lopez-Martinez and J.M. Palao. 2009. Effect of serve execution on serve efficacy in men's and women's beach volleyball. *International Journal of Applied Sports Sciences* 21, 1 (2009), 1–16.

[11] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. 2011. Estimation of IMU and MARG orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*. IEEE.

[12] Thomas Mauthner, Christina Koch, Markus Tilp, and Horst Bischof. 2008. Visual Tracking of Athletes in Beach Volleyball Using a Single Camera. *International Journal of Computer Science in Sport* 6, 2 (2008), 21–34.

[13] M. Michalopoulou, K. Papadimitriou, N. Lignos, K. Taxildaris, and P. Antoniou. 2005. Computer analysis of the technical and tactical effectiveness in Greek Beach Volleyball. *International Journal of Performance Analysis in Sport* 5, 1 (June 2005), 41–50.

[14] Jonathan C. Reeser, Glenn S. Fleisig, Becky Bolt, and Mianfang Ruan. 2010. Upper Limb Biomechanics During the Volleyball Serve and Spike. *Sports Health: A Multidisciplinary Approach* 2, 5 (Sept. 2010), 368–374.

[15] Daniel Roggen, Arash Pouryazdan, and Mathias Ciliberto. 2017. BlueSense: designing an extensible platform for wearable motion sensing, sensor research and IoT applications. In *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*. ACM.

[16] Simon Ruffieux, Denis Lalanne, Elena Mugellini, and Omar Abou Khaled. 2014. A Survey of Datasets for Human Gesture Recognition. In *Human-Computer Interaction. Advanced Interaction Modalities and Techniques*. Springer International Publishing, 337–348.

[17] Thomas Stiefmeier, Daniel Roggen, and Gerhard Tröster. 2007. Gestures are strings: efficient online gesture spotting and classification using string matching. In *Proceedings of the Second International Conference on Body Area Networks BodyNets*. ICST. https://doi.org/10.4108/bodynets.2007.143

[18] Markus Tilp, Christina Koch, Sibylle Stifter, and S. Georg Ruppert. 2006. Digital game analysis in beach volleyball. *International Journal of Performance Analysis in Sport* 6, 1 (June 2006), 140–148.

Appendix E

The Sussex-Huawei Locomotion (SHL) dataset

In addition to the work presented in this thesis, we contributed to the wearable and ubiquitous computing research community with other works, presented in this Appendix.

We organized the collection and curation of a dataset for Locomotion and Transportation mode recognition in collaboration with Mathematical and Algorithmic Sciences Lab, PRC, Huawei Technologies France. Over a period of 7 person months, we collected data from 3 participants instrumented with 4 smartphones and a front facing cameras travelling over different means of transports. The dataset resulted to be the largest dataset of its kind, with more than 750 hours of annotated data, 16 sensors modalities.

The dataset collection process was described is a series of publications listed in the following. The recognition of transportation through several sensors modalities was also explored with both machine and deep learning methods.

In addition, one of the most important contribution for the community is the creation of a very successful series of machine learning challenges, that reached the fourth edition at the time of writing, in 2021. The results of the challenges as well as our baseline for each edition are published in a yearly publication, presented in the Human Activity Sensing Corpus and Applications (HASCA) workshop, as part of the Ubiquitous Computing conference and International Symposium on Wearable Computer (Ubicomp/ISWC).

List of additional publications

- Mathias Ciliberto et al. 'High reliability Android application for multidevice multimodal mobile data acquisition and annotation'. In: *Proceedings of the 15th ACM*

Conference on Embedded Network Sensor Systems - SenSys '17. ACM Press, 2017. DOI: 10.1145/3131672.3136977

- Hristijan Gjoreski et al. 'A Versatile Annotated Dataset for Multimodal Locomotion Analytics with Mobile Devices'. In: *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems - SenSys '17*. ACM Press, 2017. DOI: 10.1145/3131672.3136976

- Hristijan Gjoreski et al. 'The University of Sussex-Huawei Locomotion and Transportation Dataset for Multimodal Analytics With Mobile Devices'. In: *IEEE Access* 6 (2018). DOI: 10.1109/ACCESS.2018.2858933. URL: https://ieeexplore.ieee.org/document/8418369/

- Lin Wang et al. 'Benchmarking the SHL recognition challenge with classical and deep-learning pipelines'. In: *UbiComp/ISWC 2018 - Adjunct Proceedings of the 2018 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2018 ACM International Symposium on Wearable Computers*. Association for Computing Machinery, Inc, 2018, pp. 1626–1635. ISBN: 9781450359665. DOI: 10.1145/3267305.3267531

- Lin Wang et al. 'Enabling reproducible research in sensor-based transportation mode recognition with the sussex-huawei dataset'. In: *IEEE Access* 7 (2019). ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2890793

- Lin Wang et al. 'Summary of the Sussex-Huawei locomotion-transportation recognition challenge 2019'. In: *UbiComp/ISWC 2019- - Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*. Association for Computing Machinery, Inc, 2019. DOI: 10.1145/3341162.3344872

- Sebastien Richoz et al. 'Human and machine recognition of transportation modes from body-worn camera images'. In: *2019 Joint 8th International Conference on Informatics, Electronics and Vision, ICIEV 2019 and 3rd International Conference on Imaging, Vision and Pattern Recognition, icIVPR 2019 with International Conference on Activity and Behavior Computing, ABC 2019*. 2019. DOI: 10.1109/ICIEV.2019.8858537

- Lin Wang et al. 'Summary of the sussex-huawei locomotion-transportation recognition challenge 2020'. In: *UbiComp/ISWC 2020 Adjunct - Proceedings of the 2020*

ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers. Association for Computing Machinery, 2020. DOI: 10.1145/3410530.3414341