Segmentation et reconaissance des gestes pour l'interaction homme-robot cognitive

Miguel Simao

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

ARTS ET MÉTIERS ParisTech

2016-ENAM-0048

École doctorale n° 432 : Science des Métiers de l'ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité " Mécanique-Matériaux "

présentée et soutenue publiquement par

Miguel SIMÃO

le 17 décembre 2018

Segmentation et reconnaissance des gestes pour l'interaction homme-robot cognitive

Directeur de thèse : **Olivier GIBARU**
Co-encadrement de la thèse : **Pedro NETO**

Jury

M. Richard BEAREE, Professeur, LISPEN, École Nationale Supérieure d'Arts et Métiers	Président
M. Marcello PELLICCIARI, Professeur, Università di Modena e Reggio Emilia	Rapporteur
M. A. Paulo MOREIRA, Professeur, INESC TEC, Universidade do Porto	Rapporteur
M. Michael WOLF, Chercheur, NASA's JPL, California Institute of Technology	Examinateur
M. Olivier GIBARU, Professeur, LISPEN, École Nationale Supérieure d'Arts et Métiers	Examinateur
M. Pedro NETO, Professeur, CORLUC, Universidade de Coimbra	Examinateur

T H È S E

Arts et Métiers ParisTech - Campus de Lille
Laboratoire d'Ingenierie des Systèmes Physiques et Numériques

Gesture Segmentation and Recognition for Cognitive Human-Robot Interaction

by

Miguel Ângelo Fernandes Castanheiro e Simão

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Joint Doctor of Philosophy
with Universidade de Coimbra

in

Mechanical Engineering

in the

Graduate Division

of the

École Nationale Supérieure d'Arts et Métiers, Campus Lille

Committee in charge:

Professor Olivier Gibaru, Chair
Professor Pedro Neto, Co-chair

2018

Gesture Segmentation and Recognition for Cognitive Human-Robot Interaction

Abstract

Gesture Segmentation and Recognition for Cognitive Human-Robot Interaction

by

Miguel Ângelo Fernandes Castanheiro e Simão

Doctor of Philosophy in Mechanical Engineering

École Nationale Supérieure d'Arts et Métiers and Universidade de Coimbra

Humans are still the most versatile and reliable resource to perform complex tasks in everyday life and manufacturing, despite the progress of the state of the art sensors and actuators allied to new machine learning methods. However, sensors and robots are very often more precise and repeatable. Therefore, it would be beneficial to develop better means of collaboration between robots and humans, improving their performance by combining the coordination and adaptation capabilities of humans with the robots' precision and repeatability. Human-Robot Collaboration (HRC) is currently limited by the existing interaction modalities. Programming an industrial robot is a tedious and time-consuming task that requires technical expertise in robot programming. The most intuitive way for humans to interact with a robot without special training is using the same skills humans use to interact with each other. One challenge is having, on one hand, an interaction process that is intuitive for humans and, on the other hand, easy to be decoded by a robot.

Reliable online recognition of gestures is important to accurately program a robot or to operate it in real time. The problem is that it is difficult to achieve accurate gesture recognition in real unstructured environments, many times from noisy and incomplete multi-sensory data streams. In a real-world application, any type of gesture, communicative or not, can occur at any time. This thesis presents a Human-Robot Interaction (HRI) framework to solve this problem, allowing the use of large vocabularies of static and dynamic hand gestures, captured with wearable sensors. A segmentation method for online data streams is proposed which enables motion detection for any kind of gesture, in an unsupervised fashion without prior training. A quantitative analysis on real-time streams showed a small over-segmentation error with most of the segments having small delays in the detection of boundaries of a gesture.

The segmentation process greatly simplifies the structure of the data stream by splitting it into static and dynamic segments. Different classifiers then predict what gestures are included in these segments, whether they are in the predefined vocabulary or not. Static Gestures (SGs) and Dynamic Gestures (DGs) are classified separately thanks to the segmentation process. Experimental tests on the UC2017 hand gesture dataset showed high accuracy and good generalization capabilities on the classification of SGs. Feature vectors for DGs are obtained by applying Data Dimensionality Reduction (DDR) – cubic interpolation and Principal Component

Analysis (PCA) – to raw data, achieving high accuracy. In online frame-by-frame classification using raw incomplete data, Long Short-Term Memory (LSTM) deep nets and Convolutional Neural Networks (CNNs) performed better than static models with specially crafted features at the cost of training and inference time. Online classification of DGs allows predictive classification, i.e., before the gesture is completed, which showed higher classification accuracy during mid-gesture than at its end.

The rejection of out-of-vocabulary gestures is proposed to be done through semi-supervised learning of a classification Artificial Neural Network (ANN) trained in an adapted Auxiliary Conditional Generative Adversarial Network (AC-GAN) framework. The proposed GAN achieved a high accuracy on the rejection of untrained patterns of the UC2018 DualMyo dataset.

Keywords: Human-robot interaction, collaborative robotics, unsupervised segmentation, online pattern recognition, supervised learning, deep learning

Résumé

Segmentation et reconnaissance des gestes pour l'interaction homme-robot cognitive

pour

Miguel Ângelo Fernandes Castanheiro e Simão

Doctorat Mécanique-Matériaux

École Nationale Supérieure d'Arts et Métiers et Universidade de Coimbra

Malgré les progrès effectués par les capteurs, actionneurs et les nouvelles méthodes d'apprentissage automatique, l'Homme reste la ressource la plus polyvalente et la plus fiable pour effectuer des tâches complexes de la vie quotidienne ou dans un contexte industriel. Par conséquent, il est très intéressant en termes de performance de développer des moyens facilitant la collaboration entre humains et robots, et permettant de combiner leurs atouts respectifs. Les humains apportent leur coordination et leur capacité d'adaptation et les robots leur précision et leur répétabilité. Aujourd'hui, la principale limitation à la collaboration Homme-robot (HRC) est le faible développement des modalités d'intéraction. La programmation d'un robot industriel est une tâche fastidieuse qui nécessite une expertise technique en programmation de robots. La manière la plus intuitive pour les humains d'interagir avec un robot - sans un entraînement spécifique - est d'utiliser des compétences similaires à celles qu'ils utilisent pour interagir entre eux. L'un des défis est d'avoir un processus d'intéraction qui soit intuitif pour les humains et qui soit facile à décoder par un robot.

Une reconnaissance des gestes, réalisée en ligne, est importante pour pouvoir programmer avec précision un robot ou pour le commander en temps réel. Le problème, dans les environnements réels et non structurés, est qu'il est difficile d'obtenir une reconnaissance précise des gestes, d'autant que les mesures, souvent partielles, proviennent généralement de flux de données multi-sensoriels contenant du bruit. Dans une application réel, tout type de geste, communicatif ou non, peut être réalisée à n'importe quel moment. Cette thèse présente un cadre formel pour l'interaction Homme-robot (HRI), qui classifie un important lexique de gestes statiques et dynamiques mesurés par des capteurs portatifs. Dans un premier temps, une méthode de segmentation des flux en ligne est proposée, qui permet la détection de mouvements pour tout type de geste, sans supervision et sans entraînement préalable. Une analyse quantitative sur les flux en temps réel a montré une petite erreur de sur-segmentation, la plupart des segments ayant de légers retards dans la détection des limites d'un geste.

Le processus de segmentation simplifie grandement la structure du flux de données en le divisant en segments statiques et dynamiques. Différents classifieurs prédisent alors quels gestes sont inclus dans ces segments, qu'ils soient ou non dans

le vocabulaire prédéfini. Gestes statiques (SGs) et gestes dynamiques (DGs) sont classés séparément grâce à un processus de segmentation. Les tests expérimentaux sur la base de données de gestes UC2017 ont montré une grande précision et de bonnes capacités de généralisation sur la classification de SGs. Les vecteurs de caractéristiques pour DGs sont obtenus en appliquant une réduction de dimensionnalité de données (DDR) – correspondant à une interpolation cubique et une analyse des composantes principales – aux données brutes, avec une haute précision. Dans la classification pas à pas en ligne utilisant des données incomplètes brutes, les réseaux de neurones profonds *Long-Short Term Memory* (LSTM) et à convolution (CNN) sont plus performants que les modèles statiques entraînés avec des caractéristiques spécialement conçues, au détriment du temps d'entraînement et d'inférence. La classification en ligne de DGs permet une classification prédictive, avant que le geste ne soit terminé, ce qui montre une plus grande précision de classification à mi-geste qu'à sa fin.

Le rejet des gestes hors du groupe préétabli est proposé par apprentissage semi-supervisé par un réseau de neurones artificiel (ANN) de classification, entraîné dans un cadre adapté à partir d'une *Auxiliary Conditional Generative Adversarial Network* (AC-GAN). La GAN proposée a atteint une haute précision de rejet des gestes non entraînés de la base de données UC2018 DualMyo.

Mots clés: Interaction homme-robot, robotique collaborative, segmentation non supervisée, reconnaissance de formes en ligne, apprentissage supervisée, apprentissage en profondeur

Resumo

Segmentação e Reconhecimento de Gestos para Interação Homem-Robô Cognitiva

por

Miguel Ângelo Fernandes Castanheiro e Simão

Doutoramento em Engenharia Mecânica

École Nationale Supérieure d'Arts et Métiers e Universidade de Coimbra

Os seres humanos continuam a ser o recurso mais versátil e fiável para executar as tarefas mais complexas do dia-a-dia e da manufatura, apesar do progresso do estado da arte de sensores e atuadores, aliado ao desenvolvimento de novos métodos de aprendizagem máquina. No entanto, os robôs e sensores têm muito frequentemente maior precisão e repetibilidade. Por isso, seria benéfico desenvolver meios que facilitem a colaboração entre robôs e seres humanos, desta forma combinando as suas respetivas vantagens. O ser humano contribui com a sua capacidade de coordenação e de adaptação, e os robôs com a sua precisão e repetibilidade. Atualmente, a colaboração homem-robô (HRC) está limitada pelos modos de interação existentes. A programação de um robô industrial é uma tarefa tediosa e demorada que exige conhecimentos técnicos em programação de robôs. A forma mais intuitiva de interação entre humanos e robôs sem treino específico é usar as mesmas habilidades que os seres humanos usam para comunicar entre si. Um desafio é ter, por um lado, um processo intuitivo para as pessoas e, ao mesmo tempo, ser facilmente descodificado por um robô.

O reconhecimento com fiabilidade de gestos em tempo real é importante para programar um robô com precisão ou para o operar em tempo real. O problema reside na dificuldade em atingir uma precisão de reconhecimento elevada em ambientes reais pouco estruturados, muitas vezes com dados distorcidos e incompletos, provenientes de fontes de dados multi-sensoriais. Numa aplicação real, qualquer tipo de gesto, comunicativos ou não, pode ocorrer a qualquer momento. Esta tese apresenta um sistema modular de interação homem-robô (HRI) para resolver este problema, permitindo o uso de grandes vocabulários de gestos manuais estáticos e dinâmicos, adquiridos por sensores corporais. Em primeiro lugar, é proposto um método de segmentação de fontes de dados em tempo real que permite a deteção de movimento para qualquer tipo de gesto, de uma forma não supervisionada e sem treino prévio. Uma análise quantitativa em tempo real revelou uma quantidade reduzida de erros de sobre-segmentação, sendo que a maior parte dos segmentos têm pequenos atrasos na deteção dos seus limites.

O módulo de segmentação permite simplificar bastante a estrutura da fonte de dados ao dividi-la em segmentos estáticos e dinâmicos. Diferentes classificadores indicam então que gestos ocorreram nesses segmentos, quer estes façam parte do vocabulário pré-definido ou não. Gestos estáticos (SGs) e dinâmicos (DGs) são

classificados separadamente graças ao processo de segmentação. Os testes experimentais na base de dados de gestos manuais UC2017 mostraram elevadas taxas de reconhecimento de SGs. Os vetores característicos para a classificação de DGs são obtidos por redução dimensional de dados (DDR) – interpolação cúbica e análise de componentes principais (PCA) – aos dados não processados, atingindo uma elevada precisão. Na classificação em tempo real de dados não processados passo-a-passo (na dimensão tempo), as redes neuronais profundas *Long Short-Term Memory* (LSTM) e de convolução (CNN) tiveram melhor performance que modelos estáticos com vetores característicos especiais, à custa de tempos de treino e inferência mais elevados. Este tipo de classificação de DGs também permite classificação antecipada, i.e., antes do gesto ser completado, o que mostrou uma taxa de reconhecimento mais elevada durante o gesto do que depois de terminar.

A rejeição de gestos fora do vocabulário pré-definido é feita através de aprendizagem semi-supervisionada de uma rede neuronal artificial (ANN) de classificação treinada com a adaptação de uma *Auxiliary Conditional Generative Adversarial Network* (AC-GAN). A adaptação proposta resulta numa elevada taxa de precisão na rejeição de gestos não treinados da base de dados UC2018 DualMyo.

Palavras-chave: Interação homem-robô, robótica colaborativa, segmentação não-supervisionada, reconhecimento de padrões em tempo real, aprendizagem supervisionada, aprendizagem profunda

To my family.

Contents

List of Figures

List of Tables

Acronyms

AC-GAN	Auxiliary Conditional Generative Adversarial Network
ANN	Artificial Neural Network
BP	Backpropagation
CNN	Convolutional Neural Network
DDR	Data Dimensionality Reduction
DG	Dynamic Gesture
DOF	Degree of Freedom
EMG	Electromyography
FFNN	Feed-Forward Neural Network
GAN	Generative Adversarial Network
HCI	Human-Computer Interaction
HMI	Human-Machine Interaction
HRC	Human-Robot Collaboration
HRI	Human-Robot Interaction
IMU	Inertial Measurement Unit
LSTM	Long Short-Term Memory
ME	Movement Epenthesis
ND	Novelty Detection
NLP	Natural Language Processing
PC	Principal Component
PCA	Principal Component Analysis
RNN	Recurrent Neural Network

SCG Scaled Conjugate Gradient

sEMG Surface Electromyography

SG Static Gesture

SGD Stochastic Gradient Descent

Acknowledgments

The redaction of this thesis would not have been possible without the contribution of many people. First and foremost, I would like to express gratitude to my doctoral advisor Professor Pedro Neto from the University of Coimbra (UC) for introducing me to the field of collaborative robotics, while also providing important advice and encouragement to push myself further than I would have otherwise. I would also like to express my utmost appreciation to my co-advisor Professor Olivier Gibaru, from the École Nationale Supérieure d'Arts et Métiers (ENSAM), for receiving me at Lille and allowing me to collaborate in the research done at the Laboratoire d'Ingénierie des Systèmes Physiques et Numériques (LISPEN). His advice was also indispensable for the success of this thesis.

I would also like to give thanks Doctors Michael Wolf and Christopher Assad for receiving me at NASA's Jet Propulsion Laboratory (JPL) despite the complex bureaucracy and enabling my contribution to their research team; it was a very enriching experience that allowed me to meet amazing people. A special thanks to the Foundation for Luso-American Development (FLAD) for proving the necessary funding for this stay at the United States.

My gratitude is extended to my home institution, the University of Coimbra, the Department of Mechanical Engineering, where I have spent the most time during my academic career, with special thanks to my colleagues at the Collaborative Robotics Laboratory (CORLUC) and the staff, all of whom contributed to make the University of Coimbra feel like a second home. A word of appreciation goes also to my colleagues at the LISPEN for all the out-of-the-box ideas and discussions that they provided. Last but not least, I am extremely thankful to my close friends and family for their unwavering support and encouragement, and for always being there when I needed the most.

This thesis was possible thanks to the support of the Foundation for Science and Technology (FCT), through grant SFRH/BD/105252/2014.

Miguel Simão
Coimbra, September 2018

Chapter 1

Introduction

1.1 Human-Robot Interaction and Collaboration

The paradigm for robot usage has changed in the last few years, from an idea in which robots work with complete autonomy to a scenario where robots cognitively collaborate with human beings. Despite the progress of the state of the art sensors and actuators, allied to the new applications of machine learning, humans are still the most versatile and reliable resource to perform complex tasks in manufacturing, such as assembly operations. However, sensors and robots are often more precise and repeatable. Therefore, it would be beneficial to have a better integration between robots and humans, improving their performance by combining the coordination and adaptation capabilities of humans with the robots' accuracy and repeatability.

The extension of robot applications to new and unconventional tasks will be in part due to incremental improvements in sensing and actuation, as well as breakthroughs in data processing enabled by machine learning. For example, the recent advances in image recognition with the use of deep learning may allow a robot to more accurately identify humans or objects in a manufacturing cell or in a social setting [1]. The correct identification of a scene is paramount for a reliable Human-Robot Collaboration (HRC). A robot can only plan its trajectory after knowing the operational scene, including human behaviour, with some degree of precision. Industrial robot manufacturers have identified this new market and are starting to offer collaborative robots with integrated sensing capabilities and more user-friendly software, enabling smaller safety zones and the use of robots without fences. Safe and intuitive human-robot interfaces are essential to increase their market penetration rates.

The most intuitive way for humans to interact with a robot without special training is using the same skills humans use to interact with each other. The interpretation of these interactions is studied by the field of Natural Language Processing (NLP). These interaction modalities include reading, listening, watching and touching. The challenge is having, on one hand, interactions that are intuitive for humans and, on the other hand, easy to be detected and recognized by a robot. The problem is that the existing interaction modalities for programming industrial robots are not intuitive, e.g., a teach pendant. They are tedious, time-consuming and require technical expertise in robot programming. Besides, while programs can

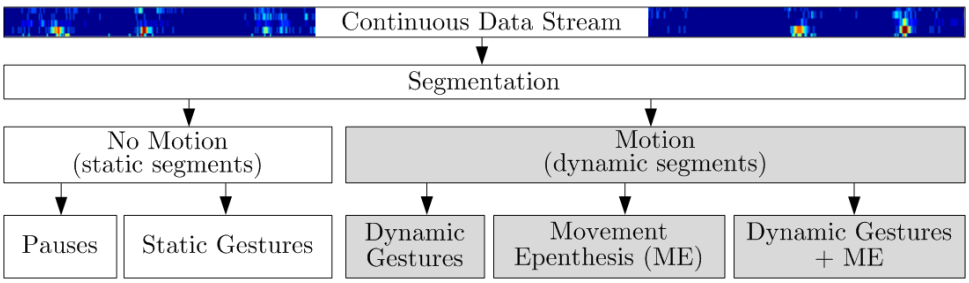


Figure 1.1: Types of gesture that may appear in a continuous data stream. Firstly, the stream can have static segments, which include pauses and static gestures. Dynamic segments correspond to dynamic gestures, ME, or a combination of both. Gestures can be communicative or otherwise.

be changed quickly, their static nature leads to limited flexibility.

Text, speech and image recognition have advanced greatly in the last few years with the advent of deep learning. Speech is an example of an interaction modality that has seen great developments recently due to its usability in Human-Computer Interaction (HCI), e.g., in smartphones. Despite its advantages, it is not reliable in noisy environments, such as in an industrial robotic cell. Text-based interfaces are time-consuming and require either a keyboard or a digital pen, thus requiring knowledge of a programming language. An alternative is to somehow watch the human to learn their intent. One way is teaching by demonstration [2], where the user performs an action and the robot trajectory is optimized to mimic the user. Another way is to start an action depending on what gesture the user is doing [3]. For instance, a human co-worker can use a pointing gesture to indicate an object to grasp or do a thumbs-up to confirm an action [4]. In this scenario, the interaction process utilizes social cues, so the user does not need specific knowledge and can focus on the task instead. The robot assists the human when required, reducing the exposition to poor ergonomic conditions and possible injuries.

In this thesis, the focus is on Human-Robot Interaction (HRI) based on hand gestures. A major challenge is related to the continuous and online recognition of gestures from real-time data streams obtained from multiple sources. The problem is that these data streams are normally unstructured, in the sense that any type of gesture, communicative or not, can occur in any sequence. This is represented in figure 1.1. In a data stream, there are periods with or without activity, i.e., dynamic or static segments, respectively. There are communicative gestures, which are pre-defined Static Gestures (SGs) or Dynamic Gestures (DGs) that belong to a vocabulary of gestures that both participants know. SGs and DGs appear intermittently with non-communicative gestures, such as pauses and Movement Epenthesis (ME) [5], which is the name given to the movement between the ending pose of a communicative gesture and the starting pose of the following. In a vocabulary of n gestures, there are n^2 classes of ME, so it is unfeasible to train all of them. Besides the communicative gestures, there are also other patterns outside of the defined vocabulary. These may be of accidental nature, such as the movement a person does

when sneezing, grabbing a falling object, or the hand gesturing a person does while speaking. It is difficult to predict what these out-of-domain gestures may be classified as, so it is a problem that should be studied. There is a very limited number of studies presenting approaches to gesture classification that work in a continuous fashion and take into account the negative effect of ME and out-of-domain gestures.

1.2 Research Objectives and Contributions

The main objective of the thesis is to propose a framework for the online classification of hand gestures for human-robot interfaces. This objective can be separated into the following work vectors:

1. Provide hand gesture data sets obtained from distinct sensors;
2. Study the performance of unsupervised data stream segmentation and comparing it to supervised methods;
3. Improving the classification of in-vocabulary gestures with various classifier models;
4. Allow predictive classification of dynamic gestures even before they are completed;
5. Improving the recognition of out-of-vocabulary gestures;
6. Demonstration of these functionalities in a collaborative robot interface.

Achieving these objectives should enable the implementation of gesture-based interaction systems that are reliable even in the presence of out-of-vocabulary gestures. Another important contribution of this thesis are the two original data sets:

1. UC2017 Static and Dynamic Gestures data set, with a CyberGlove II and position tracker [6][1];
2. UC2018 DualMyo data set, which uses two Thalmic Labs' Myo electromyography sensors [7][2].

These data sets filled a gap in the literature of data sets with gestures that can be used in the context of human-robot interaction, captured with either datagloves or electromyography. Additionally, we have done a thorough literature review on the most recent work on action recognition using Electromyography (EMG) data.

For the segmentation process, we propose a novel method to segment a continuous data stream into dynamic and static blocks in an unsupervised fashion, i.e. without previous training or knowledge of the type of gestures that might occur. The original (raw) sequence is unsegmented and unbounded, i.e., there is no beginning

[1] https://zenodo.org/record/1319659
[2] https://zenodo.org/record/1320922

and no end (online), and we do not know when a gesture starts and ends. For this segmentation method, dynamic features are required, such as the first and second derivatives of the signals. A single optimal threshold is optimized for each feature using a genetic algorithm – the performance function is not linear – fed by calibration data taken from generic static and dynamic data. Gesture patterns with sudden inversions of signal direction may originate over-segmentation using this method. This effect was seen on the velocities and accelerations numerically calculated from positional data, where single motions were split into two or more segments at the moment of signal inversion [8]. The proposed method deals with gesture motion patterns varying in spatio-temporal scale, rate of occurrence and different motion constraints. A sliding window addresses the problem of spatio-temporal variability.

The data sets and segmentation method allowed the development of several classification models based on state of the art machine learning methods. Combinations of features and classifier models were tested and originated the following contributions, in terms of hand gesture recognition:

1. Feature extraction for DG classification on wearable sensor data resulted in high classification accuracy that compares favourably with standard classifiers, including deep learning Long Short-Term Memory (LSTM) nets and Convolutional Neural Networks (CNNs). Accuracies of 95.6% for SGs and 99.3% for DGs were achieved on the UC2017 SG/DG data set;

2. Good online performance in continuous and close to real-time applications, with different subjects (user independent), and in an unstructured environment;

3. Sequential classification of DGs with features based on Data Dimensionality Reduction (DDR) showed an accuracy that is higher with incomplete data (50% or 75% of the initial data frames of a DG) than with 100% of DG data, across several classification models. In this context, DGs can be classified in anticipation, before the user finishes the gesture.

According to the objectives of the thesis, the final contribution is related to the recognition of out-of-vocabulary gestures. We have achieved very good accuracy with the extension of Generative Adversarial Networks (GANs) to gesture data, through data set augmentation on the UC2018 DualMyo data set [7].

The scientific contributions found in this thesis resulted in two published papers in international journals, and two others currently in the late stages of peer-review. There were also three participations in conferences. The papers are listed below.

Journals (published):

- M. A. Simão, P. Neto, and O. Gibaru. "Unsupervised Gesture Segmentation by Motion Detection of a Real-Time Data Stream". In: *IEEE Transactions on Industrial Informatics* in press.99 (2016), pp. 1–1. ISSN: 1551-3203. DOI: 10.1109/TII.2016.2613683

- Miguel Simão, Pedro Neto, and Olivier Gibaru. "Using data dimensionality reduction for recognition of incomplete dynamic gestures". In: *Pattern Recognition Letters* 99 (2017). User Profiling and Behavior Adaptation for Human-Robot Interaction, pp. 32–38. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2017.01.003. URL: http://www.sciencedirect.com/science/article/pii/S016786551730003X

Journals (in second review):

- M. Simão, P. Neto, O. Gibaru. "A Review on Electromyography Decoding and Pattern Recognition for Human-Machine Interaction". In review: *Biomedical Signal Processing and Control.*

- M. Simão, P. Neto, O. Gibaru. "Online Recognition of Incomplete Gesture Data to Interface Collaborative Robots". In review: *IEEE Transactions on Industrial Electronics.*

Conferences:

- M. A. Simão, P. Neto, and O. Gibaru. "Unsupervised gesture segmentation of a real-time data stream in MATLAB". in: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society.* Oct. 2016, pp. 809–814. DOI: 10.1109/IECON.2016.7793517

- M. Simão, P. Neto, and O. Gibaru. "Natural control of an industrial robot using hand gesture recognition with neural networks". In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society.* Oct. 2016, pp. 5322–5327. DOI: 10.1109/IECON.2016.7793333

- M. Simão, P. Neto, and O. Gibaru. "Taking Advantage of Data Dimensionality Reduction for Dynamic Gesture Recognition from Incomplete Data". In: *Workshop on Behavior Adaptation, Interaction and Learning for Assistive Robotics, IEEE RO-MAN 2016, NYC, USA.* 2016

1.3 Thesis Structure

This thesis is composed of several individual chapters which present the methodology developed for each of the modules of the proposed gesture recognition framework. Additionally, chapter 3 introduces the data sets that were created in the course of this thesis. A diagram is presented on figure 1.2, describing the link between the different modules of the framework, and the individual chapters in this thesis. The framework is divided in the following modules:

1. Data acquisition;

2. Motion segmentation;

3. Gesture classification;
4. Partial gesture classification;
5. Outlier detection;
6. Sequential classification;
7. Applications.

Each module was developed and tested individually, with nearly all of them resulting in papers, some of them already published and others currently in review. As a result, this thesis is mainly a collection of those papers, which were slightly adapted into chapters, while maintaining the original structure.

Data Acquisition:

For most studies in the recent literature, the input data for gesture recognition systems are captured with vision-based sensors (cameras). However, gesture classification from video stream requires large amounts of training data, especially for state of the art deep learning classifiers. Moreover, it is difficult to construct reliable features from exclusively vision-based systems due to occlusions, varying light conditions and free movement of the user in the scene [12, 13]. Hand gestures in particular are difficult to detect if the camera has limited resolution and if it is not focused on the hands, a condition difficult to respect in real-world applications. For these reasons, we focus on the study of data obtained from wearable sensors, such as Inertial Measurement Units (IMUs), position trackers, data gloves and EMG sensors. In fact, these interaction technologies have been proven to provide reliable features in unstructured environments, despite the added burden on the user.

Different sensor configurations were studied: the UC2017 data set was acquired with the fusion of a data glove and a magnetic position tracker; the UC2018 data set used a set of 16 EMG sensors. The data acquisition module is described in chapter 3, which describes the UC2017 Static and Dynamic Gestures data set, and UC2018 DualMyo data set. Additionally, visualizations of the data sets are provided. We also present a comprehensive literature review on pattern recognition for Human-Machine Interaction (HMI) based on EMG signals, shown on chapter 2.

Motion segmentation:

Once the data from the sensors is available, the segmentation module performs the segmentation of the data into static and dynamic segments. It is represented in the blue area of figure 1.2.

Gesture segmentation from an online data stream is the problem of identifying data segments that are more likely to contain meaningful interactions. This may simplify and improve the subsequent classification methodology by adding boundaries to the data that should be classified. Most of the supervised classification algorithms present reliable results only if the input data is similar to the samples

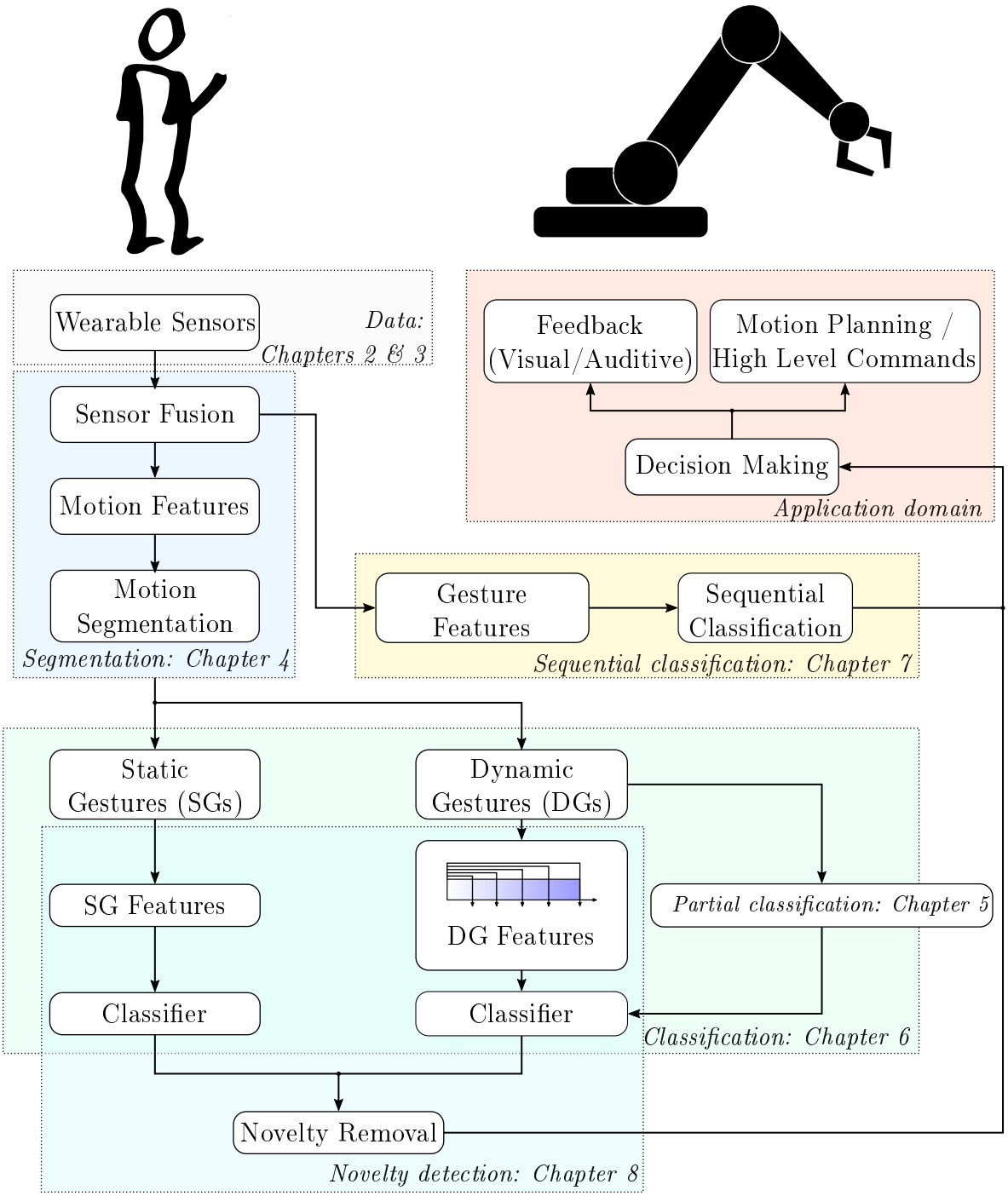


Figure 1.2: General overview of the modules of the proposed framework and thesis structure.

in the training database. A useful and effective segmentation algorithm should not depend upon previous knowledge of the possible gesture classes, nor can it be dependent of posterior data. Moreover, the algorithm must provide an output in real-time in the context of HRI (about 200ms).

The solution proposed in this thesis (chapter 4) divides a continuous data stream into dynamic and static blocks in an unsupervised fashion, i.e., without knowledge of the type of gestures that may occur. Motion features must be engineered for each specific set of sensors, with a fixed threshold differentiating static from dynamic segments. The definition of these thresholds is a non-linear and non-smooth convex optimization problem that is solved through the use of a genetic algorithm. A sliding window addresses the problem of spatio-temporal variability of gesture motion patterns. The proposed method can be implemented online without significant delays, since a decision can be made without waiting for future information and its computational load is relatively low.

Gesture classification:

The classification module is represented by the green area of figure 1.2. The static and dynamic gestures are classified individually, according to the methodology shown on chapter 6. This module has the task of extracting classification features from the raw sensor data and classify them, with the accuracy and inference time being the most important performance indicators. The objective is the evaluation of the performance of several classifier models on the UC2017 SG/DG data set, with both raw data and engineered features. Additionally, the application of DDR methods increased classification accuracy, reduced the training time and the number of samples required to train the classifiers. Principal Component Analysis (PCA) demonstrated a capability of online, frame-by-frame classification, while maintaining high recognition rates. The last performance index of interest was the generalization capability in respect to untrained samples and new users. The capability of partial (or predictive) classification of DGs with a limited amount of data was further evaluated on chapter 5.

Novelty detection:

The performance of gesture classifiers which are trained offline on a data set is typically not indicative of their online performance. The reason is that normally, a data set does not include a large fraction of real-world scenarios, such as when the user does a mistake. In those cases, there is no way of confidently predicting the performance of a classifier. The determination of which gestures classes are intended or not (or relevant to the HRI) is called Novelty Detection (ND) and is represented by the teal region in figure 1.2. On chapter 8, we propose the use of a semi-supervised methodology based on GANs, which uses the labelled samples of a data set and generated unlabelled samples corresponding either to gestures or non-gestures. This method is typically used for data generation, and we present some changes that are shown to improve the detection of out-of-vocabulary gesture

classes. The methodology presents very satisfactory results on the UC2018 DualMyo data set.

Sequential classification:

We also propose a simplification of the framework with Recurrent Neural Networks (RNNs), by joining the segmentation and classification modules as described on chapter 7, a module named sequential classification – yellow region of figure 1.2. This is an alternative that bypasses the segmentation and classification modules, therefore simplifying the data processing chain from sensors until classification. In this case, the data stream is classified sequentially (step by step), where each new frame of data originates a new prediction without having to wait for the gesture to finish. In this way, given n new time steps, the classification model outputs n predictions. The RNN implemented is an LSTM network and showed very good performance on the synthetic sequences of the UC2018 DualMyo data set. The performance compares favourably to the tradition method of extracting features from windows of data and posterior filtering of the classification output. Additionally, we propose a different gesture detection accuracy criterion which is better suited for the evaluation of online systems compared to the Jaccard distance.

Application domain:

Finally, the application domain contains a decision making module that sends instructions to the robot depending on the detected gestures and the overall context of the system, such as positions of people and objects, and the interaction's current state. Even though it is possible to obtain high accuracy classification systems and good ND, they can fail, so the decision making module is the last guard against damages to the robot, people or their environment. The robot should never move on a command that would endanger itself or its environment. This module (represented as the red region of figure 1.2) is not described on this thesis and does not have a specific chapter, but it was developed for the demonstrators presented in chapters 4 and 6.

Chapter 2

A Review on Electromyography Decoding and Pattern Recognition for Human-Machine Interaction

Miguel Simão[1], Nuno Mendes[1], Olivier Gibaru[2], and Pedro Neto[1]

Based on the paper submitted to:
Biomedical Signal Processing and Control

Abstract

This study presents a literature review on applied pattern recognition of electromyography (EMG) signals. EMG technology is introduced and the main aspects that are relevant to design an EMG sensor-based system are highlighted. EMG-based systems are used nowadays with relative success to control upper- and lower-limb prostheses, different electronic devices and machines, and for monitoring human behaviors. Nevertheless, the existing systems are still inadequate and are often abandoned by their users, prompting for further research. Many of the emerging systems are based on pattern analysis. Besides controlling prostheses, this type of research is also beneficial for the development of machine learning-based devices that capture the intention of the human user by detecting his/her gestures, opening the way for new human-machine interaction (HMI) modalities. EMG signal acquisition and filtering are discussed. This study also reviews the current feature extraction techniques, including signal processing and dimensionality reduction. We also present novel classification methods and approaches for detecting non-trained gestures. Finally, the study focuses on the current applications and different EMG systems are compared and their advantages/drawbacks are discussed.

Keywords: EMG, Pattern Classification, Regression, Human-Machine Interaction

[1]Collaborative Robotics Laboratory, University of Coimbra, Portugal.
[2]Laboratoire d'Ingénierie des Systèmes Physiques et Numériques, ENSAM ParisTech, Lille, France.

2.1 Introduction

Intuitive interfaces for prosthetic devices, robots and other smart machines, are still inaccessible to the common individual. Although there is a plethora of devices that facilitate HMI, there are obstacles in the use of those interfaces. For example, gesture-based interfaces often rely in vision sensors to capture human behavior. However, vision-based systems are still very challenging to develop and do not provide enough reliability for demanding applications. Besides, vision sensors are typically fixed in space and present limitations, such as occlusions, that reduce their area of action [14, 15]. Wearable sensors are alternative solutions that do not have such limitations, but introduce new challenges and may be cumbersome to wear when not well designed. An example of such a system is shown in [14], where a data glove is used to continuously decode hand gestures. However, hand gestures are insufficient for a reliable and intuitive interaction process. Multimodal HMI interfaces combining gestures, speech, tactile and visual cues are essential for a complete, accurate and reliable interaction process. For instance, an individual can point to indicate a target to a robot, use a dynamic gesture to instruct the robot to move and a static gesture to stop the robot. In this scenario, the user has little or nothing to learn about the interface, focusing on the desired task and not on the interaction modality. Nevertheless, all these modalities require a number of sensors that can be cumbersome to use, namely data gloves, magnetic trackers, inertial sensors, microphones, cameras, etc.

Researchers are looking into the feasibility of using electroencephalography (EEG) and EMG to decode the user's intentions [16]. For example, a data glove that is used to capture hand gestures may be replaced by a forearm band with EMG sensors, making the interaction process more natural since the user does not have to wear a glove. This is one of the current research foci in the EMG field.

In general, there are two types of EMG sensors, surface EMG (sEMG) and intramuscular EMG sensors (imEMG). imEMG sensors use needles that puncture the skin and contact directly the target muscle. For this reason, imEMG provide better signal to noise ratio (SNR). However, they are intrusive, uncomfortable, painful and difficult to setup. Some researchers have found that there is no significant difference between the two types of sensors when measuring gesture classification accuracy [17]. Despite these results, other authors have found that imEMG are more accurate when the gestures are complex motions, i.e., movements that change multiple degrees of freedom (DOF) of the hand [18]. For these reasons, this review focuses especially on surface EMG sensors.

In this review, the basics of EMG signal acquisition and filtering are presented in section 2.2. Afterwards, section 2.3 presents a detailed exposition of the state of the art EMG-specific signal pre-processing techniques, feature extraction and pattern classification methods. 2.4 is reserved for the review of the current EMG-based applications and finally, in section 2.5, the current research paths and future challenges are described.

2.2 Electromyography

This section aims to provide the reader with some introductory knowledge about the nature of electromyography signals. Moreover, we describe the techniques used for EMG signal acquisition, how they can be affected by the environment and their limitations. Understanding the signals is a fundamental step to extract features relevant to pattern recognition and to design an overall robust recognition system. A thorough description of an acquisition circuit, signal conditioning and analysis is presented by Botter *et al.* [19].

2.2.1 Signal Overview

Rather than reading the electric potential on the motor nerves, an EMG electrode reads the electric potential generated in the muscle fibers when they contract. An EMG electrode usually consists of a pair of poles aligned along the muscle fiber direction. There are also sensors with monopoles which measure the potential in respect to other reference electrodes. Monopoles have the advantage of allowing more flexible setups, since any two poles can be connected to obtain a reading. Bipolar electrodes are limited to specific electrode widths. The distance between each electrode pole and their diameter also have a significant influence on the EMG signal [20].

The provenance of signals measured with sEMG electrodes is the potentials generated by muscle cells when excited by motor nerves, rather than the electric potentials within the nerves themselves. However, there is a strong correlation between these two potentials [21]. The EMG potential reading is also correlated with the activation level of muscles and the force they generate. However, this relationship is nonlinear and difficult to model. sEMG signals have inherently low SNR, which means that they are very susceptible to environmental noise. Two important works in the field that study EMG signals and their noise are in [22, 23]. The first study describes methods to decrease the captured noise, signal artifacts and interferences in EMG recordings, as well as signal processing techniques for noise suppression (e.g. band-pass filtering, adaptive noise cancellation filters and filters based on the wavelet transform). In [23], the authors found that the environmental noise can be significantly reduced by rubbing the skin surface with an abrasive conductive paste. Other authors propose empirical mode decomposition (EMD) in order to suppress signal noise [24, 25].

sEMG electric potentials are acquired with electrodes placed on the surface of the skin just above the target muscle, which is a non-invasive technique. A graphical representation of the forearm muscles is presented in Fig. 2.1, [17]. The signals obtained from these muscles are particularly interesting for the actuation of hand prostheses and gesture recognition. Farrell [17] found that with adequate processing techniques, pattern recognition on signals measured with non-targeted electrodes can be as successful as when measured with targeted electrodes. While a targeted electrode is defined as a surface electrode that is carefully positioned above the target muscle, a non-targeted electrode is placed above the muscle of interest but

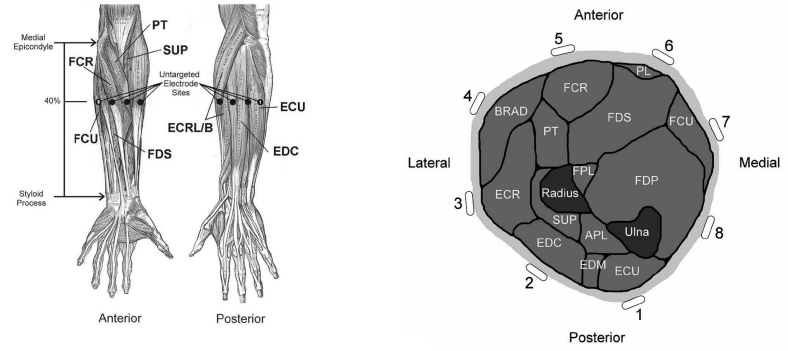


Figure 2.1: Longitudinal and transversal representations of the forearm muscles [17].

without concern for its positioning accuracy. An example of non-targeted electrodes are the elements of a linearly spaced array of electrodes. Such a setup is likely to capture signals from several muscles simultaneously. On the other hand, Farrell did not study the effects on pattern recognition (PR) accuracy caused by issues such as electrode lift-off and shift.

The surface of the electrodes is typically silver and the contact with the skin can be either wet, using a silver-chloride gel, or dry (without any medium) [26]. Wet contact electrodes offer lower skin contact impedance, which reduces the effect of external interference sources on sEMG electrodes and improves SNR. However, the use of a gel is an inconvenience for a user who might place and remove the electrodes several times a day. On the other hand, since dry electrodes do not require the use of a gel, the setup procedure is simpler. Furthermore, they are typically kept in place by an elastic band rather than an adhesive, meaning that they are fully reusable. This would enable a user to wear and remove the sensors whenever necessary, which is especially important for a user-friendly HMI system.

When the focus is the discrete pattern recognition of sEMG signals, it is important that each pattern/class remains consistent between experiments and that the features discriminate the classes correctly. The consistency is very dependent on the chosen set of features and can be negatively influenced by electrode shifts, changes in arm posture [27], fatigue and electrode-skin contact impedance, which changes in the presence of sweat [21]. EMG signals of the forearm muscles can be successfully decoded independently of the subject's gender and dominant hand [28]. Nevertheless, the study presented in [28] was limited to the control of upper limb prostheses and used only 5 classes, obtaining around 90% of recognition rate (RR).

2.2.2 Signal Filtering

Although there are many different approaches to sEMG signal pre-processing, most sensors use high-pass filters in the range of 10 to 50 Hz (sometimes higher) and low-pass filters of around 500 Hz. Most of the times, a notch filter is also used to remove power line interference at 50/60 Hz. Signal amplification is also necessary to improve the quality of its digitization. The amplification is usually between 500

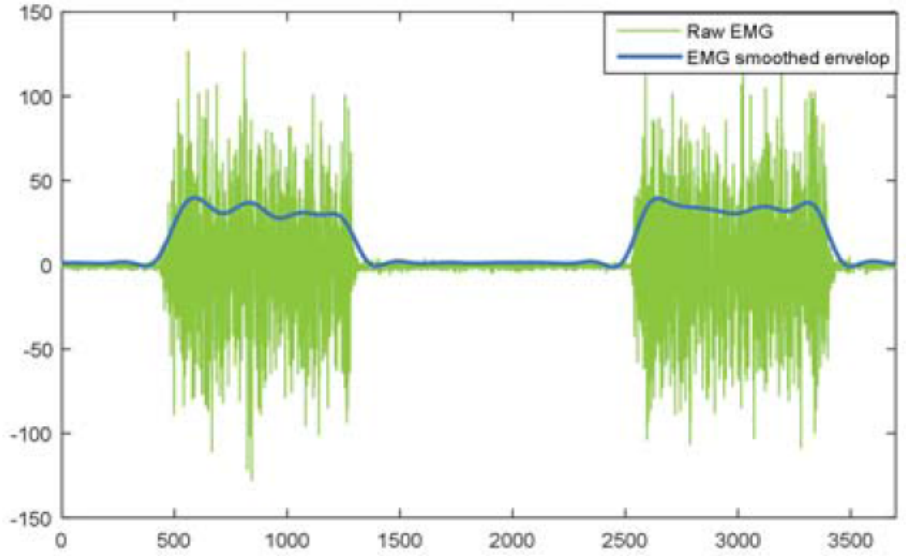


Figure 2.2: sEMG signal enveloping [29].

and 2000 fold. Bipolar electrodes are typically arranged with a differential amplifier in which the potential difference between the two poles is amplified. This type of amplification is less likely to capture external noise or signals from distant muscles.

In [30], the authors present a novel circuit for a double differential active electrode with comparatively low complexity and cost. It is composed by a single quadruple operational amplifier and few passive components, allowing accurate recordings of sEMG signals with dry contacts while also maintaining the advantages of differential amplification. This circuit configuration is also shown to reduce crosstalk between electrodes.

One of the first studies on EMG-based pattern recognition uses a band-pass filter (BPF) of 10 to 500 Hz in which the signal is amplified 1000 times and digitally sampled at a rate of 1 kHz [31]. Wet disposable electrodes were used on the *biceps brachii* and on the lateral head of triceps. The system was tested with six types of motion: elbow flexion and extension, wrist pronation and supination, and humeral (shoulder) rotation in/out. Other authors propose the use of a notch filter to reduce power line-induced interference [32]. A 30-400 Hz BPF was used and the signal is sampled by an AD converter at a rate of 2.5 kHz. The authors claim that the proposed filtering scheme avoids unstable signals for EMG pattern classification. Another study proposed a similar setup where the sEMG signals are filtered with a 20-500 Hz BPF and digitally sampled at 2 kHz [33]. In this case, the authors built a large acquisition prototype device with 57 wet contact sEMG electrodes (5 mm diameter). A reference study in the field uses a 20-450 Hz BPF on their sEMG signals with sampling rates of 1 and 2 kHz [15]. The authors built a self-contained signal acquisition armband (BioSleeve) with 16 dry-contact sEMG electrodes. Fig. 2.2 shows a smoothed signal from sEMG data after being pre-processed using a BPF and a low-pass filter [29]. A 10-500 Hz BPF (digital 4-order Butterworth filter), a notch filter at 50 Hz and a sampling rate of 1024 Hz was proposed in [34]. The

authors reported the successful control of an upper limb power-assisting exoskeleton with four electrodes on the upper limb: biceps-short head, triceps-long head, *flexor carpi radialis* and *extensor carpi radialis*. An EMG acquisition system dedicated to medical studies with a BPF of 90-450 Hz band and 1 kHz sampling is described in [28]. This system has a common-mode rejection ratio (CMRR) above 100 dB. It uses 6 channels arranged equidistantly around the forearm.

One of the advantages of systems that use dry contact electrodes, such as the BioSleeve system [15], is that they do not require precise placement of their electrodes. We consider this feature an advantage because the electrodes are placed on the forearm without gluing them to the skin, decreasing the discomfort the user feels. However, dry contact electrodes may experience slippage during a recording session, which must be prevented with mechanical forces (e.g., springs) or corrected in post-processing. Furthermore, the position repeatability of the electrodes is typically low, especially if the user is not a specialist, so post-processing is a better solution for the average user. A software-based solution benefits from high density electrode arrays since it greatly increases the likelihood that the relevant data points for gesture recognition are picked up by the electrodes.

In [35], Spanias *et al.* propose a method for the detection and compensation of disturbances in EMG recordings caused by issues such as electrode shift, liftoff and short-circuit. The detection is achieved by setting a threshold to a log-likelihood metric. When a disturbance is detected, the EMG data are disregarded and the classification only uses the undisturbed mechanical sensors, effectively minimizing the error caused by the EMG disturbances on the prothesis' control loop. In another study, the use of high-density (HD) EMG to overcome problems such as electrode shift and channel malfunction is proposed [36]. When a longitudinal shift (in the direction of the muscle fibers) is simulated, the recognition accuracy decreased from 96% to about 90%. A transverse shift (perpendicular to the direction of the muscle fibers) caused a reduction to about 80%. This technique is also accurate even when a large proportion of channels is omitted.

A common challenge related to electrode-based sensor systems such as EMG or electrocardiography (ECG) is how to avoid or reduce external noise captured by the sensors. These systems must be able to deal with different interference sources such as the electric power source which feeds the acquisition hardware. Owing to the low amplitude characteristic of EMG signals, they are easily overwhelmed by electrical disturbances. In order to reduce their negative effects in the output signal, some guidelines should be followed during the design of data acquisition setups, namely:

1. Arranging cables so that inductive and capacitive coupling between different channels is minimized, e.g., separating wires connecting distinct electrodes [20];

2. Using differential amplifiers and/or a reference electrode for common-mode rejection [28];

3. Skin preparation (shaving and/or rubbing with an abrasive conductive paste) in order to reduce the difference between the electrode poles impedances [23].

In order to remove power line noise from raw recorded EMG signals, the use of different post-processing methods have been proposed. Examples are second-order recursive digital notch filters at the power line frequency, regression-subtraction [37] and spectrum interpolation [38]. In the cases where a battery is used as a power source, its wires may also introduce noise into the output signal because of electrode pole impedance unbalances. These impedances change over time in sEMG electrodes due to disturbances in the electrode-skin interface. Table 2.1 summarizes a list of studies with an emphasis on signal filtering and suppression of power line noise.

2.2.3 Electrode Arrays

Researchers have been using high-density (HD) arrays of electrodes for the acquisition of EMG signals in both the longitudinal and transverse directions of muscle fibers, which may eliminate the need for precise placement of the electrodes, as mentioned in the previous section. The increase in the number of channels also increases the likelihood of capturing key EMG patterns. When electrode shift or liftoff occurs, the topographical map of relative muscle activity should stay roughly the same but shifted in a certain direction. However, increasing the number of channels, also adding redundancy, requires proportionally more computational power to post-process the signals and to extract regression/classification features. The detection and correction of electrode shift also add complexity to the recognition system.

Liu and Zhou use 57 EMG channels of which 48 are arranged in a 6x8 grid (6 straps with 8 electrodes each) and the remaining 9 channels are distributed across 3 other locations [33]. These locations are the first dorsal interosseous (FDI), the thenar group and the hypothenar group. A lower density array with 16 electrodes (4 straps with 4 bipolar electrodes each) is presented and evaluated in [15]. The measurements used dry contact electrodes worn in the forearm which were kept in place by a stretching sleeve. This system showed a high accuracy on gesture classification without precise positioning of the electrodes. However, the gesture classes must be retrained on each recording session.

Tkach *et al.* tested various electrode grid sizes, between 3x2 and 4x4, on transhumeral and shoulder dis-articulated amputees who went under a targeted muscle re-innervation (TMR) surgery [45]. A major difference from other approaches using electrode grids is that their arrangement includes electrode poles connected across different muscle groups. The authors indicate that it is possible that electrode pairs in the transverse direction may record regularities in activation patterns across different muscle groups, improving PR accuracy. On an earlier study, the authors demonstrated that wider inter-electrode spacing is beneficial for PR.

Researchers have been studying the use of monopolar high-density arrays (192 channels) [46]. This number of channels leads to a very high dimensionality of the EMG feature set, which in turn increases the post-processing computational load. Online reduction of the number of channels is proposed by the analysis of the topological distribution of the activity areas and limiting feature calculation to areas with significant activity. Nevertheless, the authors found that the classification accuracy

Table 2.1: Summary of the acquisition and filtering techniques of EMG signals.

Author	Classes/ Channels	BPF Range (Hz)	PLI Filter	Freq. (kHz)	Additional comments
Mewett *et al.*, 2001 [37]	1/1	10 - 500	Recursive digital notch filter; Regression: Subtraction and Spectrum Interpolation	2	Results: Although the notch filter works well for simulated interference, it does not work well for real interference.
Mewett *et al.*, 2004 [38]	1	10 - 500	Second order recursive digital notch filter, Spectrum Interpolation	2	Spectrum interpolation performed similarly to, or significantly better than, the notch filter.
Levkov *et al.*, 2005 [39]	1	-	Subtraction procedure	500	Reducing power line interference from ECG signals
Malboubi *et al.*, 2010 [40]	1	-	Adaptive Laguerre filter with fuzzy step size	10	This filter eliminates the power line interference of EMG signals successfully and it was more effective than other adaptive algorithms.
Phinyomark *et al.*, 2014 [41]	4	-	Band-pass filter with a cutoff frequency of 20 and 500 Hz and a notch filter with a cutoff frequency of 50 Hz	1024	
Botter and Vieira, 2005 [42]	16 / 96	10-500	Filtered virtual reference (FVR)	2048	FVR method attenuates the power line interference minimizing the distortions of the actual, monopolar EMGs.
Niegowski *et al.*, 2015[43]	64	1-500	Non-negative matrix factorization (NMF)	2048	NMF method outperformed two state of the art reference methods(a Butterworth filter and a polynomial quasi-harmonic modeling).
Afsharipour *et al.*, 2016[44]	128	10-750	Spectrum interpolation	2048	Signal quality and information content are not affected for 30 min.

always decreases when EMG channels are dropped, using a linear discriminant analysis (LDA) classifier.

High-density electrode grids have also been used for EMG signal decomposition [47, 48]. EMG signals are composed of superimposed motor unit action potentials (MUAP) trains which cause motor units (MU) to contract. Therefore, it should be possible to correlate complex EMG signals with the firing of individual MUs. Solving this problem would allow the detailed identification of the muscle contractions and the force they generate, effectively opening the way to improved prosthetics control, HMI and medical diagnosis. In [47], the authors propose the use of the K-means clustering (KMC) method combined with modified convolution kernel compensation (CKC) in order to reconstruct innervation pulse trains (IPT) from EMG signals. A novel CKC technique was proposed for real-time implementation, requiring processing of about 3 s of EMG signal in the initialization stage [49]. Rasheed *et al.* present in [50] a Matlab toolbox with their approach to EMG signal decomposition into MUAP trains. In [16], the authors present a technique for parametric model estimation of MUAP from EMG signals using homomorphic deconvolution. Fast independent component analysis (FastICA) can also be used for the identification of MUs from HD-EMG grids [51].

2.3 Pattern Recognition

This section presents the state of the art methods used in the recognition of recurring patterns in EMG data streams. Pattern recognition usually has three stages:

1. Signal pre-processing: reduction of the influence of external noise sources and SNR improvement;
2. Feature extraction: determination of the gesture pattern predictors;
3. Classification.

This section is structured according to the aforementioned stages. Feature extraction may originate feature vectors of relatively high dimensionality, whether because a large number of distinct features was chosen or because the number of signal channels is large. High dimensionality of the input space of classification predictors may decrease the performance of classification models in many cases. Therefore, some authors propose data dimensionality reduction techniques to face this challenge. The issue of novelty detection (detection of untrained actions/gestures) has also been studied during the past few years. This is an important topic in the area of pattern recognition that is seldom studied. Often-times, the classifiers perform well on the classification of trained patterns, yielding good results on benchmarks. Despite this, non-trained patterns may still be wrongly classified as one of the trained classes.

Most research studies in EMG-based PR follow a discrete recognition approach. This means that classification models are trained, tested and run with finite segments of data that represent examples of each one of the pattern classes. On the other hand, there are researchers that are modeling the tension or extension of a muscle

using continuous EMG data, i.e., regression of a physical quantity. In this case, the objective is not to label recurring patterns in the data, but to use the measured force or extension of a muscle to control the actuated joint's angle or torque. The methods currently used for regression of physical quantities from EMG data are presented in the sub-section 2.3.5.

2.3.1 Signal Processing and Feature Selection

The selection and extraction of features from real signals is one of the most important steps in the development of a PR system. This step may reduce undesirable parts of the signal data and emphasize more relevant data. A thorough study of several time domain and frequency domain features is presented in [52]. Strategies for feature selection and redundancy avoidance are also presented. From the 37 time domain and frequency domain features studied, the EMG features which provided the best performance were: mean absolute value (MAV), waveform length (WL), Willison amplitude (WA), auto-regressive coefficients (AR) and two different modifications to the mean absolute value (MAV1 and MAV2). This study states that EMG features based on frequency domain are not well suited for EMG signal classification.

Park and Lee implemented a PR system combining several types of EMG features simultaneously [31]. Although the inclusion of additional features with poor separability may degrade the performance of a PR algorithm, the authors tested the integral absolute value, difference absolute mean value, variance, auto-regressive (AR) model coefficients and linear cepstrum coefficients. The Dempster-Shafer theory of evidence was used as an evidence accumulation method applied through a fuzzy mapping function. A 4th order AR model was applied to extract features from an EMG signal [32]. This approach rejects all input EMG pattern classes not included in the training data of the classifier, improving classification performance. Earlier, Hu and Nenov tested a similar model which performs relatively well [53]. Their implementation relies on the extraction of features for each signal acquisition channel on 400 ms windows at 2.5 kHz. Besides the AR coefficients, which provide 4 features, they also build an EMG histogram (HEMG) with 9 bins, totaling 13 features per channel.

A novel technique that aims to remove user variability from sEMG samples is presented in [54]. In this scenario, the same system can correctly classify samples from new users without retraining. This is achieved with a bilinear model to extract user-independent features. The model can be rebuilt for a new user after only one single motion example is performed, which can be seen as a calibration step. The user-independent signal features are split into windows of 128 timesteps and incremented by 25 steps. The classifier used is a support vector machine (SVM), which takes as input features the channel-wise root mean square (RMS) value of the window's data. Their tests were performed with a 4 EMG channel setup for 5 distinct hand gesture classes. To evaluate the performance of the bilinear model, the authors use the leave-one-subject-out approach, in which one subject's data are used as test data, and the remaining users' data are used to train the model. The

classification accuracy on the test (new) subjects is 73±13%, while the baseline performance, without the bilinear model, for new users, is 54±11%.

In [33], three sets of features, (1) a set of time domain (TD) features, (2) 6th order AR (6-AR) coefficients + RMS, and (3) TD + 6-AR + RMS, resulted in 4, 7 and 11 features per channel, respectively (228, 399 and 627 features, in total). This dimensional space is relatively high for pattern recognition, so two dimensionality reduction methods were compared, principal component analysis (PCA) and uncorrelated linear discriminant analysis (ULDA). The best results were achieved using ULDA reducing the number of features to 6, which is the maximum number of linearly independent feature dimensions for a 7-class problem. These results were achievedusing of windows of 256 ms with a step of 32 ms. Wolf *et al.* analyze the signal on 500 ms windows, at a rate of 10 Hz, resulting in an overlap of 400 ms of data between consecutive windows [15]. The features are the standard deviation (SD) of the signal during each window. The authors claim that there may occur significant amplitude offsets over time on the signal's RMS, making this feature unsuitable for PR. On the other hand, the SDT is correlated to the signal's strength and invariant in respect to amplitude offsets. An Inertial Measurement Unit (IMU) is used to detect motion of the arm in respect to the body and the surroundings. The window length used for EMG signal analysis can vary significantly. For example, in a recent study, the authors analysed sEMG signals in windows of 300 ms and 125 ms[55]. While on the 300 ms experiments the authorsused disjointed windows (no overlap), the experiments with 125 ms windows had 90 ms overlaps. The proposed classification model was based on boosted classification with majority voting. The classifiers that were trained with windows of 125 ms showed significantly lower error rates. Another study combines a CyberGlove and a tactile force measurement system (FingerTPS) with 16 EMG channels, where the signal is analysed in windows of 300 ms with 50 ms steps [56]. This study had 10 classes of motions and the attained accuracy was 92.75% using just the EMG signal RMS.

Tkach *et al.* [45] propose electrode grids of 14 or 15 EMG channels, depending on the subject's type of amputation. Electrode pairs are connected in the transverse and diagonal directions. The signal was amplified 4800 times, sampled digitally (16 bits) using a custom-built data acquisition (DAQ) system at 1 kHz, high-pass filtered at 20 Hz and low-pass filtered at 450 Hz using 2nd order Butterworth filters. The signals are processed in 250 ms windows and with a step of 50 ms. In a recent study, Ortiz-catalan *et al.* [57] extract features from 200 ms windows, with 50 ms increments. Two types of electrode arrangements were considered, one with 4 electrodes (for 11 motion classes) and another with 8 electrodes (27 classes). The features selected for both configurations were: MAV, number of zero crossings (ZC), slope sign changes (SSC) and WL. Raw sEMG signals measured with a frequency of 1 kHz were passed through a BPF with bandwidth of 10-450 Hz to calculate wavelength features and control a prosthetic hand for forearm amputees [58].

In [28], Riillo *et al.* propose the use of a supervised algorithm based in Common Spatial Patterns (CSP) to improve class separability. This technique is applied before feature extraction and improves the SNR of the sEMG recordings. Spatial filters are implemented to maximize the variance of one class while minimizing the

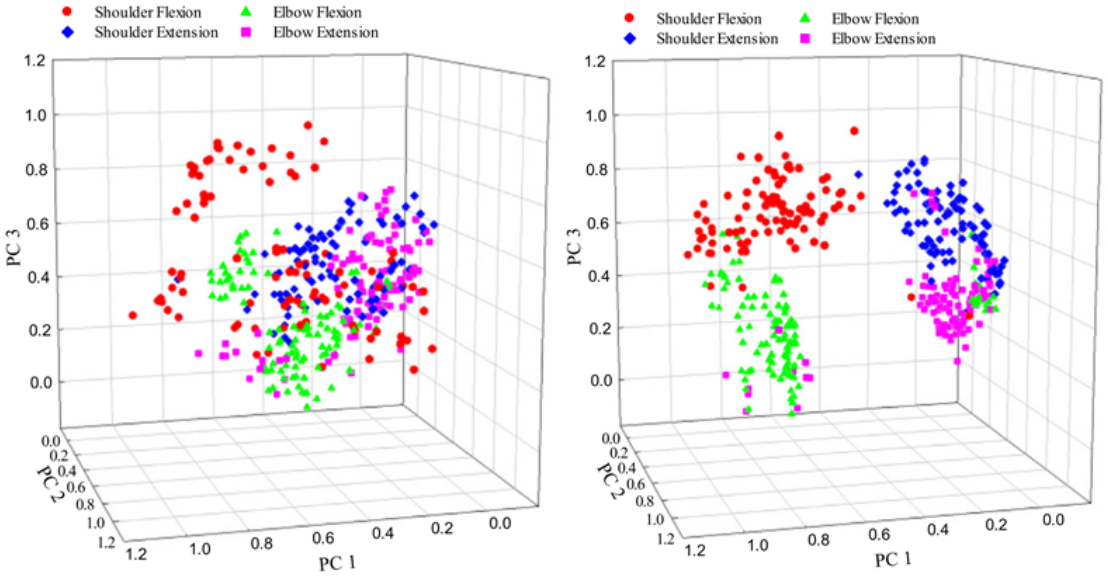


Figure 2.3: The first three principal components of feature maps: for TD features on the left, and features processed with the STFT-ranking technique, on the right [59]. The features show smaller intra-class dispersion and better separability.

variance of the remaining classes by projecting data into a subspace.

A novel short-term Fourier transform ranking feature (STFT-ranking) was recently introduced in [59]. Comparing to the usual arrangement of TD and fractal domain (FD) features, STFT-ranking reflects better the relationship between EMG signals of different muscles. It also normalizes the features to a fixed discrete range, with unit increments between 1 and the number of channels. The Fourier transform is used to retrieve the frequency coefficients from windows of data. The number of coefficients is typically high, so they are combined into a lower number of bins and then ranked, being attributed the score 1 to the highest combined coefficient. Thus, the number of features obtained is the number of bins per signal channel. Even then, the dimensionality may be too large for a machine learning problem, in which case further dimensionality reduction may be necessary, by means of PCA or other techniques. Since STFT-ranking ranks the contribution of individual EMG channels, it must be used with more than one channel. An example of the improvement of class separability with the STFT-ranking technique is shown in Fig. 2.3.

Coelho and Lima evaluated the use of fractal dimension methods to extract EMG features [60]. Their experiments involved the classification of 7 distinct limb motions using 8 EMG channels.

The study of the impact of multiple dynamic factors (forearm rotation angles and contraction force levels) on pattern recognition is detailed in [61]. The authors showed that time-domain power spectral descriptors (TD-PSD) and discrete Fourier transform (DFT) features allowed superior accuracy. Liu *et al.* proposed an invariant feature extraction (IFE) framework based on Fisher kernel discriminant analysis (FDA) [62]. This method minimizes intra-class deviation of features extracted in

different days and maximizes the inter-class dispersion (class separability). These improvements may allow the training of classifiers with better generalization capabilities.

There is no clear solution to sEMG channel and feature selection for PR-based prostheses controllers. Deterministic methods have been applied to select the feature-channel pairs that present the best results in the classification of hand postures at different arm positions [63]. Two methods are proposed, namely distance-based feature selection (DFSS) to determine a separability index and a correlation-based feature selection (CFSS) method to measure the amount of mutual information between features and classes. The two methods demonstrated better classification accuracy than experiments that used all of the available features. When compared to DFSS, the CFSS method requires less feature-channel pairs for the same performance.

The influence of sEMG data provided by anterior and posterior muscles of a forearm in the control of finger flexion movements is demonstrated in the context of prosthesis development and control [64]. This study claims that the posterior muscles are the main contributor for simple finger flexion whereas the anterior muscles are the most important for complex finger flexion. Simple flexions include the movement of a single finger at a time, while complex finger motions occur when two or more fingers move concurrently. Nevertheless, it is suggested that some simple and complex finger flexion motions can be identified from either posterior muscles or anterior muscles, so there is data source redundancy in the classification of both types of motion.

EMG signals may vary significantly depending on the user, namely due to differences in body composition, muscle size and electrode positioning. To solve this problem and obtain a robust and simple multi-user interface, an automated user calibration method is highly desirable. Cannan and Hu try to partially solve this problem by establishing a linear relationship between the Maximum Voluntary Contraction (MVC) and the upper forearm circumference [65]. MVC is defined as the maximum ability to contract muscles, i.e., the maximum attainable EMG signal. The MVC allows the definition of an activation threshold that has better performance than the RMS or MAV. This threshold is then used to normalize EMG signals across different subjects and improve motion classification accuracy. Their experiments used 4 Biometrics differential EMG electrodes and DAQ system, a Dynamometer hand grip strength sensor, and a custom-built extensometer to measure the forearm's perimeter. The hand grip strength measures the force the subjects exert when closing the hand, which is proportional to the force being exerted by the forearm's muscles. The extensometer is composed of a load cell whose ends are connected by an elastic band worn tightly around the forearm. Hooke's law allows the determination of the perimeter of the cross-section of the forearm as a function of the force measured by the load cell. The authors showed that EMG signal thresholds determined using the proposed method (as a function of forearm circumference) perform better than fixed thresholds calculated from a small group of subjects. In other words, the circumference is a better predictor of the maximum EMG signal that new subjects can generate than the average maximum signal estimated from a

subset of subjects. The study also showed that the body mass index (BMI) of an individual changes the MVC value, since the BMI is related to the thickness of the subcutaneous tissue, which in turn has an effect on the muscle-electrode interface impedance.

2.3.2 Dimensionality Reduction

An efficient data dimensionality reduction technique reduces intra-class variability while increasing inter-class distance, thus simplifying the classification task. Class separability measures the distance between data points of different classes [60]. Good class separability indicates that data points of a class have a distance from points of other classes greater than the distance between points of the original class. Thornton's separability index (SI) has been proposed for the design of machine learning systems [66].

PCA is a commonly used linear dimensionality reduction technique. It is an unsupervised method which is invariant in respect to the data points' classes [28]. PCA performs an orthogonal linear transformation that projects the data onto a new lower-dimensional space with decreased dimension correlation. The resulting eigenvectors (or principal vectors) are the basis of the new space. The eigenvalues determine the variance of the data in the direction of the respective eigenvectors. Low eigenvalues mean that the data have reduced variance on that dimension, so the dimensions with the lowest eigenvalues can be truncated without losing a significant amount of information. The importance of having relevant embedded muscle activity features in a low-dimensional space is highlighted in [67]. This study shows how PCA can be used as an unsupervised feature extraction method and illustrates the efficacy of this method in capturing features from sEMG signals.

Naik and Nguyen studied the performance of the Non-negative Matrix Factorization (NMF), another dimensionality reduction method [68]. NMF reduces dimensionality by approximately factorizing the source data into two matrices [69]. The factorization is based on the minimization of a cost function that is inversely proportional to the quality of the approximation. If there is latent structure in the source data, the approximation will be effective in lower dimensional spaces. Some studies have found that NMF can be used for the determination of muscle synergies on EMG signals [21]. NMF establishes a constraint of non-negative data only, which is not valid for raw EMG signals. Nevertheless, the inversion of the negative values of EMG signals is sufficient to obtain good results in EMG signal decomposition. Another approach is the application of NMF to the signal's RMS. Other methods for dimensionality reduction are ULDA [33], the locality preserving projections (LPP) and neighborhood preserving embedding (NPE). A method based on discriminant analysis (OFNDA) is proposed in [70].

From a clinical point of view, sEMG is considered to be a good data source for the control of multifunctional upper limb prostheses, due to its noninvasive nature and comfort of use. However, sEMG generates redundant data which may be detrimental to the performance of prostheses control systems. In order to extract significant information from sEMG raw data and successfully recognize the motions an amputee

intends to do, a promising approach based on independent component analysis (ICA) is proposed in [71]. Achieving a classification accuracy above 95% for a library of 11 gestures, this approach encourages further developments in real-time prosthetic applications.

Finally, it is important to reinforce that a wrong application of a data dimensionality reduction method may lead to the loss of information that may introduce new failure modes on the classification task.

2.3.3 Classification

Human action recognition is achievable by applying PR on EMG signals. Static and dynamic hand and arm gesture recognition is an active research topic, existing several different methods for its implementation. Some researchers use PR to recognize motion intention, such as the contraction of a muscle [34, 45]. Another important classification problem is novelty detection. In this case, the recognition system must also recognize when a pattern does not fit any of the trained classes, as proposed in [32]. Novelty detection is implemented prior to the pattern classification with a one-class classifier called support vector data description (SVDD). The authors also mention that there are several other promising methods for this type of classification: one-class SVM, the single-class minimax probability machine (MPM) and the kernel PCA (KPCA) method. Despite being supervised classifiers, they only have to be trained with the target patterns. This is true for most of the one-class classifiers.

In [72], the authors compare the classification accuracy of discriminant analysis methods against a SVM. The only feature retrieved from the EMG data was the MAV and the number of classes considered was 4 (upper-arm movements). The achieved accuracy was better for the SVM classifier (99%) using 10-fold cross validation, while discriminant analysis models achieved accuracies in the range between 96% and 98%.

A self-recovering PR system that relies on LDA for classification and allows online retraining using an optimized LDA-based retraining algorithm is presented in [73]. This system is designed to detect and overcome disturbances on the EMG signal. It is able to retrain the classifier for 3 classes and 4 channels in 0.55 ms, which does not affect real-time usage. The system reaches a recovery rate of 93.5%, which means that it substantially reduced the number of errors after retraining. It is unknown if the algorithm is as efficient for a larger number of classes, or if the retraining time increases exponentially with the number of classes.

A classification system of finger movements for dexterous hand prosthesis control using sEMG is presented in [74]. The system was tested on both intact-limbed subjects and transradial amputees. The effect of the number of electrode channels (NCh) on the classification accuracy was also studied. It was concluded that the accuracy increases with a higher NCh. Nevertheless, 6 channels allowed an accuracy of 98% over 10 intact-limbed subjects on 15 classes (class-channel ratio of 2.5). They used 6th order AR coefficients and TD features (RMS, waveform length, number of zero crossings, integral absolute value and slope sign changes), a total of 11 features per channel. Both PCA and orthogonal fuzzy neighborhood discriminant

analysis (OFNDA) were tested for dimensionality reduction. OFNDA presented the best results, while also being suitable for applications with a large number of pattern classes. The reported classification accuracy was achieved using LDA. A recent study proposes the classification of EMG signals using multi-scale principal component analysis (MSPCA) for denoising, discrete wavelet transform (DWT) for feature extraction and decision tree algorithms for classification [75]. It is reported that the best performance (96.67% classification accuracy) was achieved with a combination of DWT and a random forest classifier.

A PR system was applied to discriminate 6 hand grasp patterns and one rest pattern (7 classes) in patients with spinal injuries (degraded motor control of the upper limbs) [33]. The authors propose a data segmentation scheme based on the amplitude of the EMG signals to determine the onset and end of active segments. Two different classifiers were studied, LDA and k-nearest neighbors (KNN). The majority vote was used as a post-processing method to make the final classification decision. There are a total of 17 deciders (centred window – 8 time steps before, 8 after, 1 in between), each using data with time offsets of 32 ms, which in turn means that there is a decision delay of 272 ms in respect to the newest time step. The achieved accuracy was 97.20±4.0% for a single model, and the majority vote increased it to 97.93±3.3%. The results are influenced the most by the feature set and less significantly by the classification model itself.

A new classifier for sEMG signals, boosted random forests (MCLPBoost), allows the detection of novel patterns [55]. In this work, the data were obtained from 6 subjects, from 6 EMG channels, which repeated each class 42 times. The chosen features were all TD features: MAV, ZC, SSC, WL. The proposed classifier achieved a RR of 92% but the novelty detection accuracy was just 20%. However, the authors propose the use of a threshold on the sample score provided by the classifier, where samples that do not reach the threshold are considered to be novel. A low threshold prevents the detection of novel classes and higher thresholds cause more trained patterns to be classified as novelty. The authors tuned the threshold so that there is a novel detection accuracy of 80%, but concurrently the RR in samples of trained classes decreased to 80%. Wavelengths from sEMG signals of forearm amputees were used as input to a regression artificial neural network (ANN) that estimated the hand shape (joint angles of each finger, wrist pronation/supination angle and palmar flexion/dorsiflexion) in order to control a virtual prosthesis hand, Fig. 2.4, [58]. Experimental results for a motion set with 4 pattern classes showed an average normalized joint angle RMS error of 0.164.

An interesting approach relies on the overlapped windowing technique, with a length of 300 ms, and 75 ms of delay between windows [28]. EMG features were extracted for each channel and window. Four time-domain features were tested: mean (M), RMS, WA and SSC. PCA was used to reduce the dimensionality of the feature vector and the classifier employed was an ANN with 10 hidden nodes (5 classes). The highest accuracy was achieved using a feature combination of M, RMS and WA, achieving an accuracy of 88.85% ± 7.19% on 20 different subjects.

In a recent study, EMG signals were used to classify 10 classes of motion intention using three different approaches: Fuzzy Gaussian Mixture Models (FGMM),

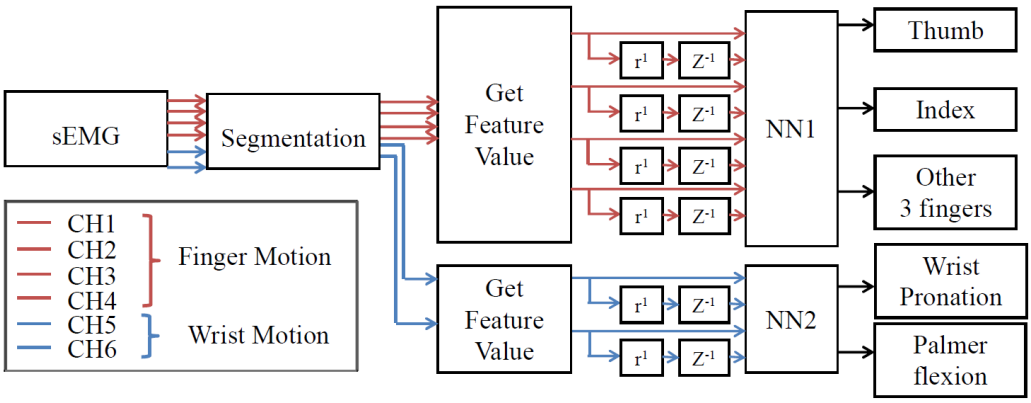


Figure 2.4: Estimator of the finger and wrist joint angles [58].

Gaussian Mixture Models (GMM) and SVM [56]. The study describes thoroughly the algorithm used during the training process. The acquisition setup is composed by 16 EMG electrodes carefully placed over a predefined set of muscles, while a single feature is extracted from each channel (RMS). This setup achieved a maximum overall RR of 92.75% with a distance-based FGMM. The GMM and the SVM reached average recognition rates of 87.25% and 88.13%, respectively.

In a different approach, TD and AR features calculated on windows of 250 ms were extracted from 14 to 15 EMG channels [45]. The TD features were MAV, ZC, SSC, WL and 6th order AR coefficients, a total of 10 features per channel. The chosen classifier was LDA and no dimensionality reduction technique is mentioned. The number of motion classes can be as high as 29, including 8 single joint movements (elbow flexion/extension, wrist flexion/extension, wrist pronation/supination, hand open/close), 20 combinations of two of these movements and a class without any motion. Their classification error was relatively low when using only 9 classes (1.3%). However, increasing the number of classes to 13 caused the error percentage to increase to 17.1%. The authors also implemented real-time sets of tests called Target Achievement Control (TAC). These tests consisted on moving a virtual limb from a nominal position to a target posture and maintaining it for 1 second. To evaluate the controllability in the TAC test, 3 metrics were used for each classifier: completion rate (percentage of total trials completed within 5 s), completion time and length of movement error (length of movement beyond the minimum required distance). The average length error was 27%, the average completion rate was 96% and the average completion time was 2 s.

The performance of 4 classification models, namely LDA, Multi-Layer Perceptron (MLP), Self-Organized Map (SOM, unsupervised ANN) and Regulatory Feedback Networks (RFN, classification based on negative feedback) were compared [57]. Furthermore, the authors explored the possibility of multi-class classification, which means that a given pattern may belong to multiple classes. Six different classification setups were tested: Single, All Movements as Individual (AMI), Ago/Antagonist-Mixed (AAM), One-vs-All (OVA), One-vs-One (OVO) and All-And-One (AAO).

Generally, they differ in the number of classifiers used, the strategy for classification and class definition (requiring mixed classes to be defined, or multi-class classification). For the 11 single-class patterns, which only have a single target class each, the offline accuracy was the highest for the LDA classifier using the OVO structure. The OVO structure relies on a multitude of classifiers that classify the pattern between two classes. The final output is set by majority voting on the combination of all classifiers. The second best result was obtained with the SOM-OVO structure, with 94.5% accuracy. Considering the problem with 27 pattern classes, which include multi-class samples, the highest accuracy achieved was 94.2% using the MLP-AMI structure. It was closely followed by a single SOM and LDA-AMI with a RR of 93.8% and 93.7%, respectively. The AMI structure uses a single classifier and requires the mixed classes to be considered separate classes, therefore increasing the number of outputs of the classifier.

The performance of a dimensionality reduction technique, the Non-negative Matrix Factorization (NMF), is presented in [68]. It was used to data mine the EMG signals and perform unsupervised learning using the RMS and the 4th order AR coefficients as features. The classifier used was an ANN. This method was studied on an existing dataset[1] that contains data from 2 EMG channels (extensor and flexor muscles) for 10 different finger movements (5 simple and 5 complex, involving one or more fingers moving). For the 5 single finger flexion classes, the attained classification accuracy was 93.92±0.63% (inter-subject). The 5 complex finger flexions were decoded accurately up to 87.58±0.36% of the samples.

Riillo *et al.* presented a comparative study between supervised and unsupervised data pre-processing on healthy subjects and transradial amputees [28]. The unsupervised PCA approach was compared to the common spatial pattern (CSP) supervised methodology. The PCs are calculated for the feature vectors, while the CSP methodology is used before feature extraction. The average accuracy for the PCA approach was 88.81%±6.58% using an ANN classifier and RMS-WA features, whereas the accuracy for the CSP strategy was slightly better, with 89.35%±6.16% using an ANN classifier, and M-RMS-WA as features. These results are valid for a set of 5 classes using 6 EMG channels. The amputee's classification accuracies were also relatively high, 92.04% for PCA and 93.4% for CSP. Seethanjali and Ray found that for their setup, simple logistic regression (SLR) fared better than other classifiers, such as decision trees, logistic model trees (LMT), ANNs, SVMs or LDA [76]. TD features allowed to achieve an accuracy of 93% for 6 classes. A spiking neural network (SNN) achieved an accuracy of 95.3% for 6 hand motions using 8 electrodes on the forearm [77]. For the development of the SNN, the study proposes the use of the NeuCube environment. The authors demonstrated that frequency-domain features do not represent the classes properly.

Stango *et al.* [36] studied the effect on recognition accuracy of problems such as electrode shift and channel loss. The classifier used was a SVM with a linear kernel with features retrieved from a variogram function. This resulted in 96% accuracy for 9 classes of gestures. When a longitudinal shift was simulated, it decreased to

[1] http://www.rami-khushaba.com/electromyogram-emg-repository.html

about 90%. A transversal shift caused a reduction to 80% without retraining. This technique is also accurate even when a large proportion of the EMG channels is omitted. Another study in which missing data are considered is presented in [78]. The authors propose the use of an extended full-dimensional GMM.

Table 2.2: Pattern recognition results using machine learning techniques in EMG data.

Authors	Classes/ EMG Channels	Features	Classifier(s)	Accuracy
Xiaorong Zhang *et al.*, 2013 [73]	3/4	MAV, ZC, WL, SSC	LDA	95.4% (99.7% after online retraining)
Matsubara and Morimoto, 2013 [54]	5/4	RMS	SVM	73±13%
Al-Timemy *et al.*, 2013 [74]	15/6	6AR, RMS, WL, ZC, IAV, SSC	OFNDA (reduction) LDA (classifier)	98.0%
J. Liu and Zhou, 2013 [33]	7/57	TD, 6AR, RMS	LDA, KNN	97-98%
Z. Li *et al.*, 2013 [55]	7/6	MAV, ZC, SSC, WL	Boosted Random Forests (rejection untrained classes)	92%/20% (trained/ untrained) or 80%/80%
Phinyomark *et al.*, 2014 [41]	8/4	dAR	QDA	97.96±1.1%
Pan *et al.*, 2014 [79]	7/7	MAV, ZC, WL, SSC	LDA	95.64%
Riillo *et al.*, 2014 [28]	5/6	M, RMS, WA, SSC	PCA (reduction) ANN (classifier)	88.85±7.19%
Ju and Liu, 2014 [56]	10/5	RMS	FGMM	92.75%
Tkach *et al.*, 2014 [45]*	(9-13-17-29) /(14-15)	6AR, MAV, ZC, SSC, WL	LDA	97.5% (9 classes) 92.0% (13 classes) 88.8% (17 classes) 81.0% (29 classes)
Ortiz-catalan *et al.*, 2014 [57]	11/4 or 27/8	MAV, ZC, SSC, WL	LDA, MLP, SOM, RFN	11 classes: 95.7% (LDA-OVO) 27 classes: 94.2% (MLP-AMI) 93.8% (SOM)
Naik and Nguyen, 2014 [68]	5/2	4AR, RMS	NMF (reduction) ANN (classifier)	93.92±0.63% (single finger) 87.58±0.36% (two fingers)

continued ...

...continued

Authors	Classes/ EMG Channels	Features	Classifier(s)	Accuracy
Li *et al.*, 2014 [80]	4/4	4AR	ANN	93.0%
Purushothaman and Ray, 2014 [81]	6/4	MAV, ZC, SSC, WL	SLR	91.1%
Amatanon, 2014 [82]	10/8	MAV or MV	ANN	94.7±4.6% (MV) 97.9±1.5% (MAV)
Kawasaki *et al.*, 2014 [58]	4/6	WL	ANN	For 4 patterns an average RMS error of 0.164 (joint angles)
Tsai *et al.*, 2015 [59]	4/6	STFT-ranking	PCA (reduction) SVM (classifier)	93.9±4.3%
Peng *et al.*, 2015 [77]	6/8	MAV, WL	NeuCube SNN	95.3%
Liu *et al.*, 2015 [62]	11/4	6AR	FDA	91.2%
Paleari *et al.*, 2015 [83]	11/192	RMS	ANN	96.5%
Stango *et al.*, 2015 [36]	9/48	Variogram	SVM	95.8±2.3%

* The subjects were previously subjected to a TMR procedure.

Rosati *et al.* presented a hierarchical clustering-based method to group strides and by this way to characterize human gait from EMG data [84]. Results show that the variability of the pattern onset/end timing is significantly reduced after clustering. Yamanoi *et al.* proposed a myoelectric hand that estimates hand posture (8 grip postures) and grip force simultaneously [85]. The ability to simultaneously estimate parameters is important for a number of EMG-based applications. EMG-based classification accuracy is critical for controlling forearm prosthetic devices. The effect of the temporal and spatial information was studied in the classification accuracy of 7 hand gestures recorded from partial-hand and trans-radial amputee volunteers, as well as able-bodied volunteers [86]. The authors concluded that the classification accuracy is significantly impacted by the number of electrodes and the signal processing window length. The optimal window size was also found to be independent of the number of electrodes used.

Table 2.2 summarizes the most recent studies in the field of EMG-based pattern recognition, showing for each study the number of classes considered, the number of EMG channels, the features extracted from the signals, the chosen classifier and

the attained recognition accuracy.

2.3.4 Novelty Detection

The problem of novelty detection is important in order to improve the robustness of pattern recognition systems. Even if a system presents a relatively high accuracy in the classification of predetermined (trained) classes, it is still likely to miss-classify novel classes as one of the trained classes. This is a seldom addressed failure mode that could lead to unexpected results and potentially endanger users and their environment.

Liu and Huang propose the use of an ensemble of one-class (OVA) SVDD classifiers that demonstrates a high level of generalization [32]. If a new sample does not fall into any of the SVDD hyper-spheres, it is considered an outlier, a novelty. Else, it is considered a targeted pattern and can be further processed. However, this ensemble does not replace a multi-class classifier because the SVDD hyper-spheres may intersect. In this case, the same pattern is classified as multiple classes, so an extra step must be taken to determine the final classification output.

A different solution using modified boosted Random Forests (MCLPBoost) was studied to solve the problem of novelty detection [55]. The effect of arm movements on sEMG pattern recognition for hand and wrist motions was studied in [87]. Results showed that arm movements significantly impact classification performance when the classifier is trained in one arm condition and tested in another.

2.3.5 Regression

Regression of EMG data is the process of predicting the motion of a biological joint using the sEMG signals from the muscles actuating that joint. sEMG signals may be affected by physiological and non-physiological factors. The accuracy of force estimation using sEMG is affected by changes in joint angle and contraction type, among others [88, 89]. An actual example of this effect is shown in [90]. Nine electrodes are placed over specific arm muscles (the most relevant for the actuation of the arm) and the movements are mapped to the EMG states using a linear model trained by an iterative prediction-error minimization algorithm.

Fleischer and Hommel developed a torque-assistive knee exoskeleton in which the operator's intended torque is decoded from EMG signals [91]. The control system requires models of the operator's body and exoskeleton, which use signals obtained from various sensors, thus requiring considerable calibration. A comparison between the ground truth torque and the produced torque on a stair climbing exercise can be seen on Fig. 2.5.

Haddad and Mirka studied the effects of muscle fatigue on the EMG-force relationships [92]. The authors present an algorithm to adjust the variable gain factor of the load estimator, as a function of the median frequency's negative shift measured on short term fatigue. Their work reduced the error caused by an invariant gain factor from 21.4% to 12.9%.

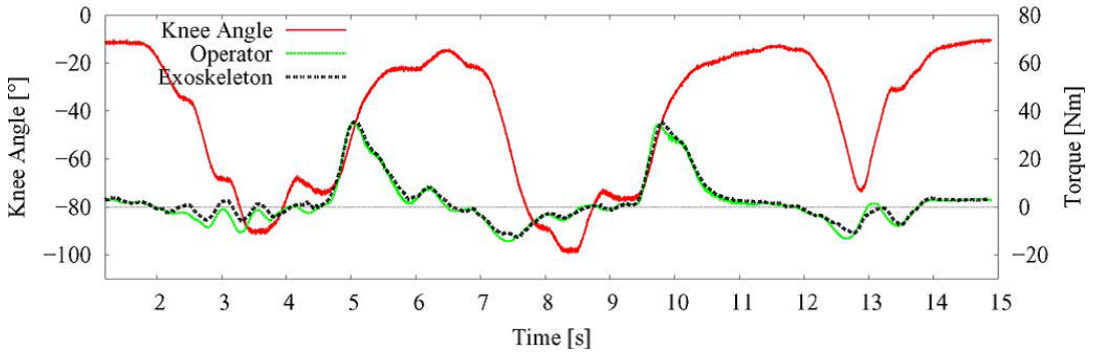


Figure 2.5: Torque produced (black) by a lower-limb exoskeleton system presented in [91]. The operator's intended torque (green) and the knee angle (red), considering full torque support.

Hashemi *et al.* use angle-based EMG amplitude calibration and parallel cascade identification (PCI) modeling to predict the nonlinear and dynamic relationship between EMG signals and force in dynamic contractions of the *biceps* and *triceps brachii* muscles [93]. The authors achieved a minimum %RMSE of 8.3% for concentric contractions, 10.3% for eccentric contractions and 33.3% for fully dynamic contractions.

In order to estimate the force generated by muscles, Li *et al.* use a high-pass filter to remove most of the signal components in the low-frequency range, rectify the signal and calculate the signal envelope [34]. Considering further processing steps, the methodology achieved a good estimation of the resulting force on the joint. An alternative method based on pattern classification uses sliding windows of 100 time steps (10 Hz) with 2 features per channel (MAV and WL). The minimum %RMSE obtained was 6.9% on the *biceps brachii*. The authors also found that for a natural control of their exoskeleton application, the delay between the onset of the muscle contraction and the corresponding motion of the device must not exceed 300 ms. Another study showed that a non-parametric machine learning model obtained by Gaussian Process Regression (GPR) modeled the elbow torque better than a pneumatic artificial muscle (PAM) model [94]. EMG signals were obtained from the *biceps* and *triceps*. Recently, Valentini *et al.* used a GMM to estimate the angle of a human joint with wavelet transform features obtained from sEMG signals [95]. Their system achieved real-time performance with a processing time below 3 ms. The same problem has been tackled with recurrent neural networks (RNN), namely a nonlinear auto-regressive exogenous (NARX) model [96]. In [97], a time-delay neural network (TDNN) was used to predict the elbow's torque in dynamic conditions using only sEMG data. The test data showed a root mean square error of 1.0 Nm in a range of around ±14.0 Nm.

A signal normalization approach proposed by Han *et al.* demonstrated a significant reduction of the dependence on varying external loads [98]. Their system estimated joint movement continuously from EMG signals using a state-space Hill-based muscle model closed-loop approach. Other authors studied the effect of vary-

ing loads on joint angle estimation and proposed different alternatives to overcome the classification problems those loads may bring [99].

In [100], the authors propose a method to obtain finger movement for SPC using linear and kernel ridge regression (KRR) from sEMG and accelerometer data. The results showed that the correlation was $R^2 = 0.79$ for KRR. The authors also found that non-linear regression may outperform linear regression during within-movement generalization. Their performance is comparable when generalizing to novel movements.

2.3.6 Multi-modal Sensing

Ju and Liu use 3 types of sensors to capture simultaneously the finger angle trajectories, the hand contact forces, and the forearm EMG signals [56]. They establish correlations between the sensors' signals by using Empirical Copula. Chang *et al.* propose the combination of EMG sensors with an IMU to improve the accuracy of hand gesture recognition [101]. The authors report a RR of 97.5% for 6 classes of gestures which were performed by 10 subjects. A SVM with a linear kernel provided the best results.

A device with a wrist-worn motion sensor (IMU) and 4 sEMG sensors achieved a RR of 95.9% for 40 distinct signs of the American Sign Language with an SVM classification model [102]. Nevertheless, most of the features were obtained from the inertial sensors, which explains the relatively high RR considering the large amount of distinct gestures.

2.4 Applications

EMG signals have been used in different applications, namely for rehabilitation, prostheses control, medical diagnosis and for the most diverse human-machine interaction approaches. However, most of these applications are confined to controlled laboratory environments and do not have yet the required level of reliability for deployment in the real world. Research in EMG-based devices will proceed in the coming years towards the improvement of EMG pattern classification in multi-user systems and reduction of the effect of real-world issues, such as electrode shift and presence of novel patterns. Additionally, prosthesis control must be extended to more complex tasks in order to reduce the prosthesis rejection rates and allow the intuitive control of machines. Furthermore, the the combination of EMG data with other sensors appears to improve substantially the robustness of classification systems.

Farina *et al.* have previously presented a thorough review on upper-limb prostheses, including previously studied control strategies and the challenges for future studies [21]. They also present a list of ideal characteristics of a EMG-based control system for upper limb prostheses. Although sEMG control has been studied for over 60 years, so far there has not been a significant evolution on clinical and commercial upper limb prostheses. Current systems are robust, but they are not very natural

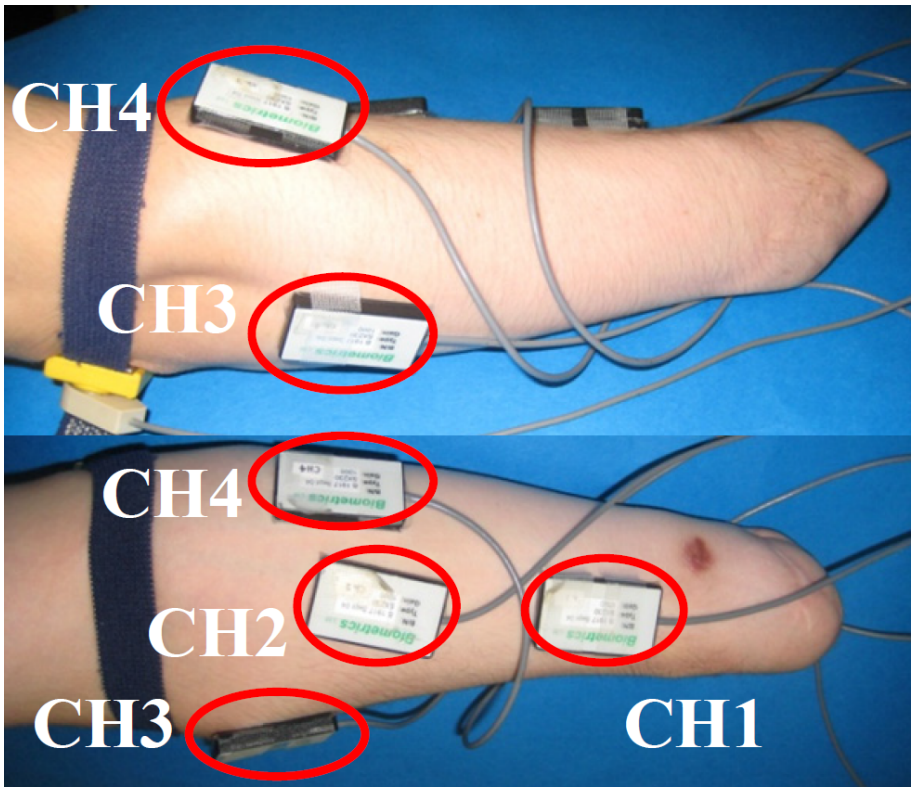


Figure 2.6: Electrodes positioning in the forearm of an amputee [58].

nor practical, so they are often abandoned by their users. Today's control techniques rely on direct control (DC), in which the user has to elevate the EMG signal above a predetermined threshold to generate a command (controlling a single DOF in one direction). If a patient intends to control multiple DOFs, the DOF combination must have been predefined, or the patient has to select and control each DOF individually. Some researchers are currently exploring the use of pattern recognition for the simultaneous control of several DOFs, as described in section 2.3.3.

An architecture for the control of a UB Hand IV device is presented in [103]. Tele-operation is performed using 8 sEMG channels. Performance was measured on grasping tasks and a high success rate was achieved. A prosthetic hand is controlled by forearm amputees using regression models for the hand joints' angles (fingers, wrist and palmar angles) from sEMG data [58]. The system was successfully tested by a right forearm amputee using a set of 4 hand patterns, Fig. 2.6. Recently, EMG has been used to predict the intended movements of an upper limb amputee for prosthesis control [104]. The proposed methodology demonstrated to be robust to the negative effect of untrained actions such as limb position changes.

Lower-limb prosthesis control is not studied as often as upper-limb protheses. Nevertheless, a reference study presents a lower-limb DAQ system in which the EMG signal is used to predict gait mode change of a leg [106].

In the conventional DC method for prostheses, each EMG signal is analyzed independently and concurrently to match the number of DOFs to be controlled.

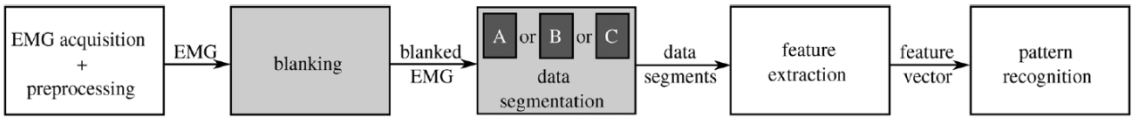


Figure 2.7: Signal processing chain including blanking [105].

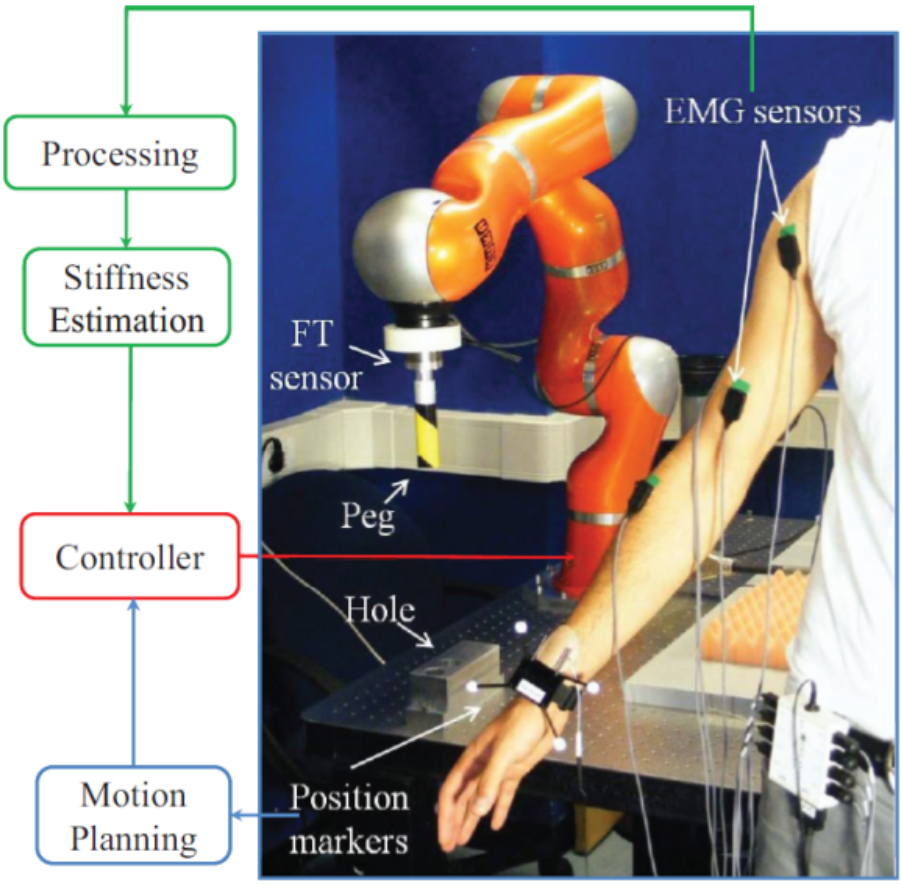


Figure 2.8: Diagram of a data acquisition system used to estimate the joint stiffness to be replicated by a robot during a tele-operation session [29].

In this scenario, each EMG channel is used as the unique control signal for each actuator and the range of actuation is controlled proportionally using the amplitude or RMS of the EMG signal [21]. This means that only one function can be used at a time. However, it is possible to create a control system in which multiple functions are directly controlled, simultaneously and proportionally (SPC), without the need for a switching function. This type of control may be improved by the use of a Bayesian filter on sEMG data [107].

Novel control methods based on PR assume that when a given subject tries to perform a certain task, there is a consistent EMG pattern in a group of electrodes. Using these methods, the subject is not required to switch between DOFs in order to perform a task, as necessary in DC systems. Nevertheless, this type of approach does not allow the simultaneous control of more than one motion, a characteristic of natural movements. Research in PR shows high classification accuracy for large sets of motions, but there are still no commercial/clinical systems exploiting PR-based control. Even in the presence of high accuracy, a small amount of repeatable errors may be unacceptable in some tasks. A classification error may lead to an unwanted motion that could compromise the entire functionality of the system. Furthermore, the control is still sequential, controlling only one type of motion at a time.

Based on PR, researchers have been trying to interpret combined motions as new classes [57]. This may become unfeasible as the number of elementary motions increases, since the number of classes would grow exponentially. Alternatively, there have been approaches based on regression control of joint kinematics with EMG. This requires modeling the relationships between forces and kinematics, which are difficult to determine. The coefficient of determination has been ascertained to be in the range of 70% to 90%, when comparing ANNs, linear regression and kernel methods. Unsupervised methods can also provide a solution to this problem, such as signal factorization and NMF [68]. The motion accuracy in control tests between different approaches have been shown to be similar, since the users are often capable of learning how to adapt their muscle contractions to improve motion RR.

Scheme *et al.* introduced two new proportional control algorithms based on PR control [108]. In their work, the classifier output is used to determine movement direction, while the proportional control (PC) gain is computed with other metrics. The traditional approach is based on mapping linearly the EMG amplitude (using the MAV with a class-specific gain) to the force or speed to be generated by the actuators. If multiple sensors are used, a summation and normalization of the MAV of all channels is performed, with equal weights. Their first approach takes into account the intra-class and intra-channel averages. Also, the PC-EMG gain is square-mapped, so that the PC has better resolution at lower speeds. The other approach is expressed in terms of a percentage of the maximum contraction and it is mapped cubically. In either case, the data needed to create the PC model are limited to a small training data set. The authors concluded that this last method presented better results in terms of usability for both able bodied subjects and amputees when combined with a PR system.

Ison *et al.* developed a 4-DOF EMG SPC control interface for a 7-DOF manipulator using a high density array of sEMG electrodes [48]. They also circumvented

the necessity of narrow electrode placement constraints with signal decomposition inspired by natural muscle synergies. Succinctly, muscle synergies are visible when 2 activation signals control a single joint DOF, each one in a different direction. A formulation proposing the use of task-specific muscle synergy activation coefficients is shown in [109]. The coefficients are modeled as the latent system state and estimated using a constrained Kalman filter. The accuracy in task discrimination is around 90% and it is computationally efficient (decision reached in under 3 ms).

In order to restore the performance of pattern recognition in prosthesis control when EMG signals are corrupted by electrical stimulation artifacts (due to electrocutaneous stimulation feedback provided to the user by the prosthetic system), researchers investigated artifact blanking with three data segmentation approaches, following the signal processing chain displayed in Fig. 2.7. Basically, the blanking block in Fig. 2.7 removes stimulation segments from the raw signal, so that the samples used for feature extraction are free of signal interference. The results demonstrated that the proposed artifact blanking method can be used as a practical solution to eliminate the negative influence of the stimulation artifact on EMG pattern classification [105].

Neuromuscular diseases can be diagnosed using EMG signals [110]. This is achieved by the decomposition of the signals into frequency sub-bands using the discrete wavelet transform (DWT), and classified using a SVM model whose parameters were determined by particle swarm optimization (PSO). The system yielded an overall accuracy of 97.41% on 1200 signal sets. Research has been carried out to improve the motor control of patients who have had central nervous system (CNS) injuries that caused degraded CNS-Motor signals [111]. Several disorders of the muscular system may be diagnosed using EMG signals, such as dystonic muscles [112]. PR on EMG signals has also been recently used to improve electro-larynx performance on patients who lost their voice-box, usually due to larynx cancer. Using 2 EMG channels positioned over neck muscles, researchers tried to decode the intended voice sound patterns, using either SVM PR or SVM regression on 4 distinct classes. The regression achieved 33% lower RMSE than the classification, but subjectively, the classification provides a more accurate representation of the patient's intention. The accuracy rate achieved was 78.05±6.3% [113].

In an interesting study, the stiffness command to a robot was derived in real-time from the measurement of 8 EMG channels from an operator's arm, Fig. 2.8, [29].

2.5 Conclusion

Although current research on EMG-based control is targeted mostly for upper-limb prostheses through the application of EMG sensors on the forearm, there is a need for the development of sophisticated EMG-based human-machine interfaces. Robust PR on the forearm's EMG signals may create a strong alternative to hand gesture recognition from vision systems or data gloves. The current solutions are either too restrictive and unreliable (vision) or cumbrous to use (gloves). A forearm band with

dry sEMG and inertial sensors would be a good solution for discrete hand gesture classification.

The current efforts on EMG PR development are targeted at increasing the reliability of the classification by anticipating and correcting sources of EMG disturbance, such as muscle fatigue, varying skin impedance, electrode shift and liftoff. The solutions studied are based on (1) the use of HD-EMG monopole electrode arrays to obtain topographical maps of muscle activity, (2) further signal conditioning, (3) and increased robustness of the classification models. The use of electrode arrays greatly increases the number of signal channels. In turn, this requires the use of dimensionality reduction strategies and smart feature selection (or deep learning), since high-dimensional input features require much higher computation power to train and use classifiers, thus undesirably increasing the time between signal acquisition and classification. On the other hand, these high-dimensional spaces have the potential to be more robust to imprecise electrode placement and electrode shift. Some studies have showed their potential to detect muscle activation synergies.

Classification accuracy is still very dependent on the feature-classifier combination. The features selected are very often time-domain features and AR model coefficients. Novel features include variogram-based and STFT features. The classifier models most commonly used are LDA and its variations, SVM and ANN in several forms. Joint angle/torque regression based on machine learning is being developed with RNN, PCI, GPR, GMM and KRR. All these methods are being used on single muscle-actuation with relative success and will possibly be used in the upcoming HMI interfaces.

An important element that is lacking discussion in most studies in this domain is the reproducibility of the presented experiments. The data sets are seldom published, making it difficult to compare different methodologies. Furthermore, it is challenging for different researchers to reproduce exactly the same data acquisition setup, due to the limited availability of the instrumentation (many researchers use custom-build sensors) and high sensibility of the EMG sensors in respect to the electrode placement conditions. We believe that in the future, setups with dry electrodes will prevail because of their ease of use. Furthermore, larger data sets should be built around reproducibility.

Chapter 3

Data Sets

The tests that were performed to evaluate the various methodologies presented in this thesis required the use of hand gesture data sets. While some tests were done in data sets that predate this work, these were hard to replicate for real-world applications, either because of unknown acquisition factors or limited equipment availability. For that reason, we created new data sets with the available acquisition hardware. We were also able to introduce new classes of gestures specifically to be used in human-robot interaction demonstrators.

In this chapter, we present the two original data sets we have created: (1) the UC2017 SG/DG data set, with a CyberGlove II and position tracker; (2) the UC2018 DualMyo data set, which uses two Thalmic Labs' Myo electromyography sensors. The data sets are split into two sections which include the data acquisition protocols, data set composition and data visualizations. Each data set also includes predefined train, validation and test splits to aid the comparison with other approaches in the same conditions.

3.1 UC2017 SG/DG Data Set

In the study presented in chapter 6, we introduced the UC2017 Static and Dynamic Gestures data set[1] [6]. The reason for the creation of the data set is the lack of hand gesture data sets captured with data gloves. An example of such a data set is presented in [114], the Australian Sign Language (Auslan) signs (High Quality) data set. However, this data set is very limited, since it contains 95 classes of gestures and only 27 examples for each. Furthermore, only one subject participated in this effort.

Most researchers use vision-based systems such as the Microsoft Kinect to acquire and classify hand gesture data. Despite that, we believe that we can achieve more reliable results and allow the use of more complex gestures with wearable systems. There are not many data sets with data captured from wearable systems, due to the plethora of data gloves in the market and their relative high cost. For these reasons, we have chosen to create a new data set to present and evaluate our gesture recognition framework.

[1] UC2017 Static and Dynamic Gesture Dataset: `https://zenodo.org/record/1319659`

We built a new hand gesture data set with data obtained from a CyberGlove II data glove [115] and a Polhemus Liberty position tracker with 6 Degrees of Freedom (DOFs) [116]. The objectives of the data set are:

1. Provide a large library of hand gestures
2. Represent user variability through the acquisition of samples from multiple users
3. Being actual gestures used in a real-world interaction

There are two types of gestures: SGs and DGs. The difference between both is that SGs are snapshots of sensor data values at an instant the hand describes a static gesture. DGs are variable-length timeseries of poses and orientations that correspond to a specific pattern in time.

Data Acquisition

The library is composed of 24 SGs classes and 10 DGs, which are represented in figure 3.1, respectively. The data set includes SG data from eight subjects with a total of 100 repetitions for each of the 24 classes (2400 samples in total). The DG samples were obtained from six subjects and there are cumulatively 131 repetitions of each class (1310 samples in total). All of the subjects are right-handed and performed the gestures with their left hand.

The CyberGlove provides digital signals that are proportional to the bending angle of each one of the 22 sensors which are elastically attached to a subset of the hand's joints, giving an approximated measurement of the hand's shape. The tracker's sensor is rigidly attached to the glove on the wrist and measures its position and orientation in respect to a ground-fixed frame. The orientation is the rotation between the fixed frame and the frame of the sensor, given as the intrinsic Euler angles yaw, pitch and roll (ZYX). We fuse the sensor data together online since the sensors have slightly different acquisition rates (100Hz for the glove and 120Hz for the tracker). The tracker data are under-sampled by gathering only the closest tracker frame in time.

The magnetic tracker reference was fixed in a location free of magnetic interference. The users are then asked to put on the data glove on their own on their left hand, leaving the right hand available to perform other tasks – all of the test subjects were right-handed. As a result, the sensors are not carefully placed, which should yield a data set with larger variance. There is no calibration done in this setup. The subjects then follow a graphical interface that shows the representation of the gesture to be performed and press a button to save a sample. The order of the gestures was randomized to prevent order dependencies. Furthermore, the subjects were requested to repeat the sampling for two to three different sessions. We have also implemented an online movement detection algorithm to facilitate the labelling of DGs, namely the starting and ending frames, as shown in Chapter 4.

A final point should be made about the random sampling of the DGs. We have included in the samples the transition between the ending pose of the previous

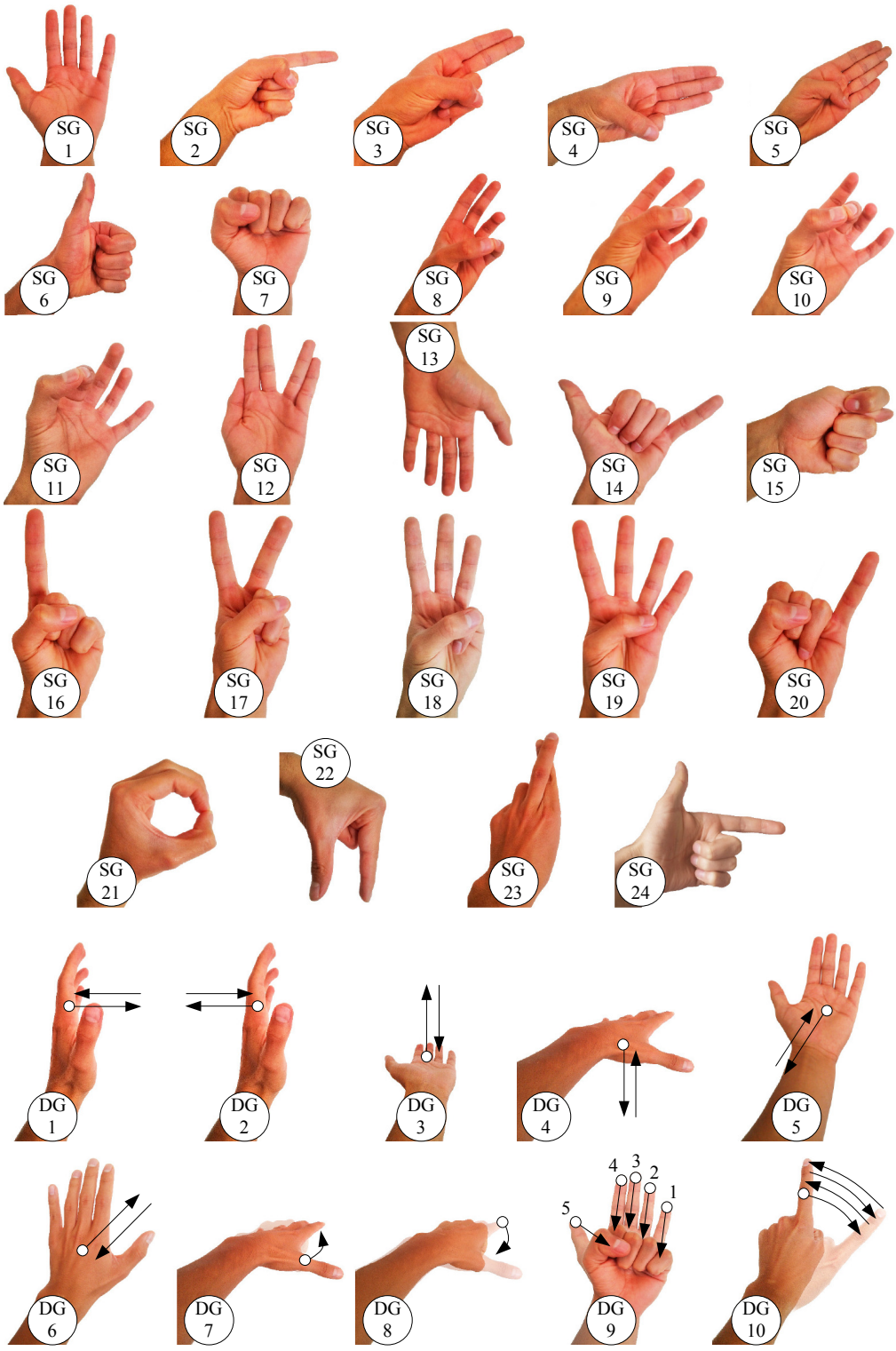


Figure 3.1: Representations of the library of 24 static and 10 dynamic gesture classes of the UC2017 SG/DG data set.

sample and the starting pose of a sample. This transition stage is also known as movement epenthesis. Due to random sampling, there is a high likelihood that all of the possible transitions were recorded.

3.1.1 UC2017SG

It is important to visualize and analyse the acquired data to make sure that there are no discernible problems. Because of the relatively high dimensionality of the data (28 features), we have to resort to dimensionality reduction techniques to visualize them in a 2D or 3D space. For that, we have applied PCA, which linearly transforms the data points into a new orthogonal coordinate system whose axes follow the directions of most variance in the data. This means that the first axes of the new system will describe most of the variance in the data. By extension, we can generally truncate the last dimensions of this new system without losing meaningful information.

As previously mentioned, the data set is divided in SGs and DGs. The SGs have just two dimensions, the first being the observation dimension and the second the variable dimension, related to the sensors. The data variables are standardized by their mean and variance, and then PCA is applied. We show the 2D PCA projection in figure 3.2, for each class and user individually. There is a scatter plot for each of the 24 classes. We show every data point in each plot and highlight the points corresponding to each class. Distinct markers are used for different subjects.

The visualization of the data has the following objectives:

1. Finding clusters of data points related to the gesture classes;

2. Checking for the existence of unexpected inter-user and intra-class dispersion.

Firstly, if we look at the whole distribution of data, it is hard to discern clusters that would match the classes. One of the reasons for this is the high dimensionality of the data points, which causes classes to overlap. Additionally, the sensors are not calibrated between different users. It is desirable to avoid calibration when changing users, in real applications.

Secondly, there is considerable intra-class dispersion, especially between different users. For example, almost all classes show two clusters: one for subjects 1 and 2, and another for the remaining subjects. The first two subjects were more involved in the construction of the data set and thus had more experience doing the gestures. The remaining subjects had no training whatsoever and performed the gestures with pictures as reference.

Examples of two separate clusters of data points corresponding to different users can be seen in figure 3.2, in gestures SG1 through SG5. While unexpected, this is not a predictor of failure of the classifier. In section 6.3, we show the classification accuracy for several models, but preliminary tests with Artificial Neural Networks (ANNs) show that the accuracy on the test set is 95.25% across classes and users. However, as can be seen in table 3.1, the accuracy of the samples of subject 1 is on average 6.42 percent points lower than the other users. This may be explained

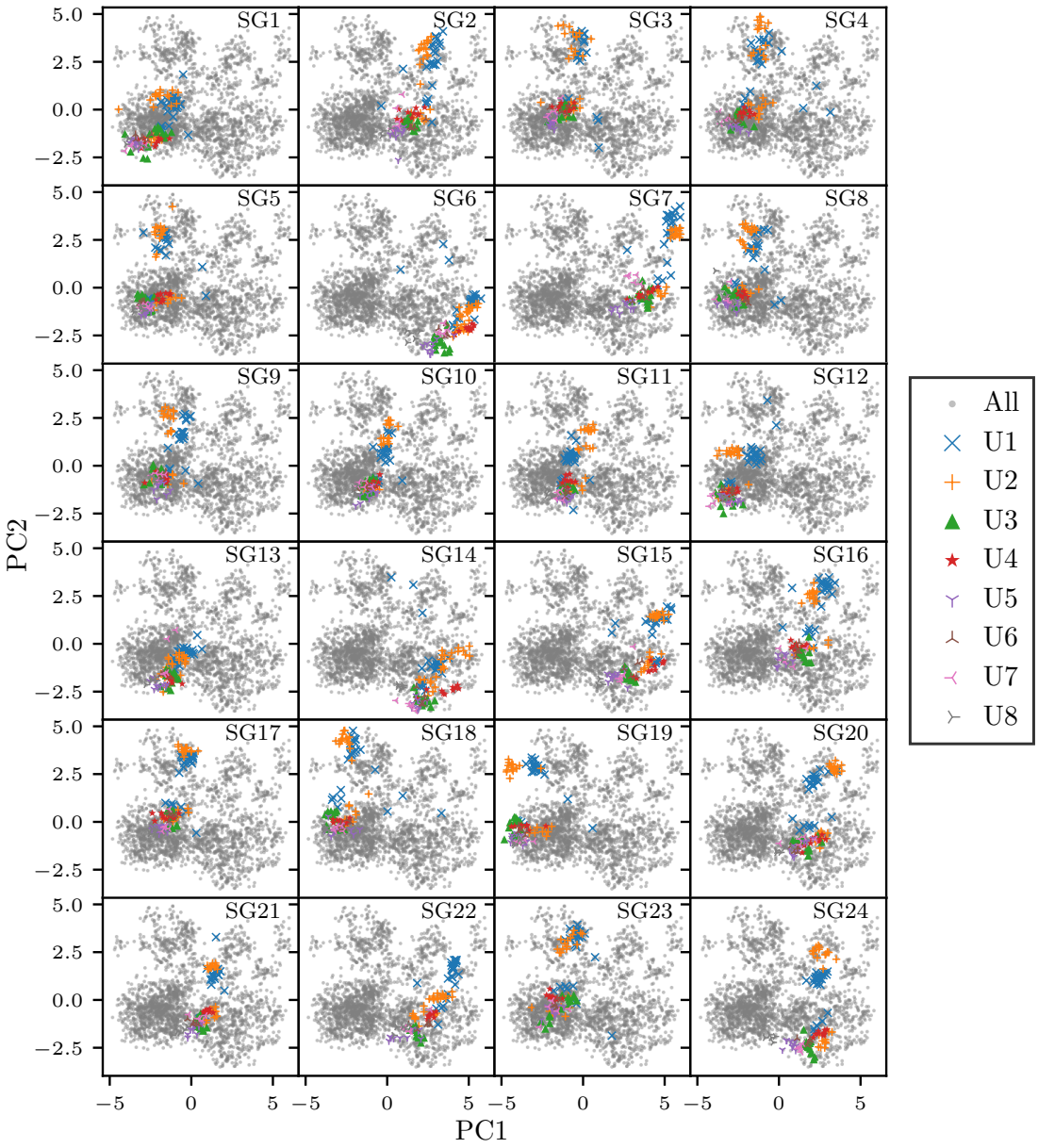


Figure 3.2: Representation of the UC2017SG data projected on the plane defined by the first two principal components of the data. The data set points of all classes are represented in grey in every plot. In each plot, the points relative to each class are highlighted in colour. The colours and markers map the sample to each of the 8 participating subjects, as shown in the legend.

Table 3.1: Accuracy per user of a neural network when trained on the UC2017SG data set.

	U1	U2	U3	U4	U5	U6	U7	U8
Accuracy (%)	91.30	100.00	96.77	96.30	95.31	100.00	95.65	100.00

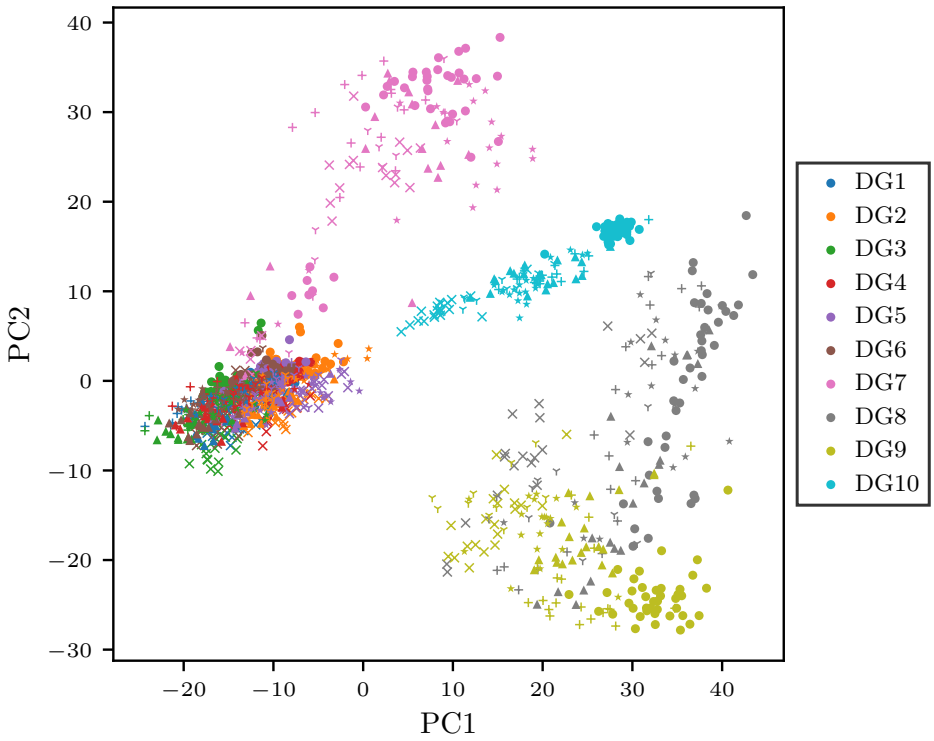


Figure 3.3: Representation of the UC2017 DG data projected on the plane defined by the first two principal components of the data. The ten DG classes are represented with different colors. Distinct markers also correspond to different subjects.

by distance between the clusters. U2 also has many samples outside of the main clusters, but proportionally, U1 generally has more.

3.1.2 UC2017DG

DGs have 3 dimensions: the observations (samples), time and variables. Since gesture length varies between different observations, the samples are resampled to a fixed length of 30 frames with cubic interpolation. This length was chosen arbitrarly, since it does not affect the visualization significantly. The time dimension is then squeezed by concatenating every timestep into a single vector. The vectors are normalized and their dimensionaly is reduced with PCA. The projection in a 2D space is shown in figure 3.3. The gestures are represented in figure 6.2.

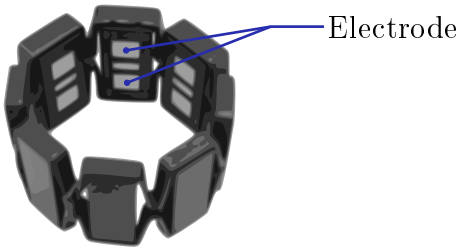


Figure 3.4: Representation of the Myo EMG wireless sensor developed by Thalmic Labs [117]. It has a set of 8 bipolar electrodes divided equidistantly on a band. Two poles of an electrode are highlighted.

The inspection of figure 3.3 shows 5 clusters of observations. The classes DG7 through DG10 have well separated clusters, while the others form a single cluster in this view. This is justified by their similarity, since they do not vary in hand-shape, only in hand position and orientation. Despite the apparent difficulty in separating the classes DG1 through DG6, several classifier models reached close to perfect classification on these features, as shown in table 6.3.

The intra-class variance of the DG samples is greater than that of the SGs, which is a result of the very large number of variables that the time dimension adds to DGs. Typically, different users also produce small individual clusters, as seen in figure 3.3, classes DG7 through DG10. But unlike the SGs, each class has a single large cluster, which in principle should result in better separation boundaries and classification models.

3.2 UC2018 DualMyo

The second data set included in this study is based on Surface Electromyography (sEMG), i.e., the sensing of electric potentials generated by the muscles when they are exercised. It is possible to detect some hand gestures and finger movement by measuring the patterns of electric potentials on the forearm, as showed in Chapter 2. The objective is to provide an introductory database [2] for the development of a robust and highly accurate human-robot interface [7].

The acquisition of EMG signals is done with two Thalmic Labs Myo [117] devices, which is composed by bipolar electrodes, figure 3.4. The Myo is a wireless armband with 8 equidistantly spaced EMG electrodes that communicates the signals via a Bluetooth connection at a maximum rate of 200Hz. For reference, there is another recently published data set with a similar setup in [118].

The data set includes eight distinct gestures, shown in figure 3.5: (0) rest, (1) closed fist, (2) open hand, (3) wave in, (4) wave out, (5) double-tap, (6) hand down, (7) hand up. While the hand shape is represented in the figures, it is not strictly defined. However, it is important that the correct muscles groups are actuated.

[2] UC2018 DualMyo Hand Gesture Dataset: `https://zenodo.org/record/1320922`

Figure 3.5: Representation of the library of gestures of the UC2018 DualMyo data set. By order, the gestures are: (0) rest, (1) closed fist, (2) open hand, (3) wave in, (4) wave out, (5) double-tap, (6) hand down, (7) hand up.

Their actuation is guaranteed by forcing the palm moves in the correct direction. For example, for the wave-in gesture, the palm must be forced towards the body.

Each one of the gesture classes was repeated 110 times across 5 recording sessions, so there is a total of 880 samples. Each sample is 2 seconds long and contains data from 16 synchronized EMG channels. The data set has the following definition:

$$\mathcal{S}_{myo} = \left\{(\mathbf{X}^i, \iota^i),\ i = 1, 2, ..., n_{samples}\right\} \tag{3.1}$$

where $\mathbf{X}^i \in \mathbb{M}^{T \times D}$ is a 2D matrix of length T time steps and D channels with the raw recorded data. $iota^i \in$ is the index of the class the sample, an integer between 0 and 7.

Acquisition Setup and Protocol

The Myos are worn over the forearm muscles, which actuate the hand's fingers and wrist, therefore providing information about the activity of the hand. They are worn interlaced, providing twice the number of data points, with an axial resolution of 22.5 deg around the forearm, as displayed in figure 3.6.

The sensors are placed carefully according to the following instructions:

1. The Myos are placed on the forearm with the usb port pointing outwards, palm and sensor 5 (Myo's own numbering system) facing upwards.

2. They are placed next to one another with their middle position close to the thickest section of the forearm.

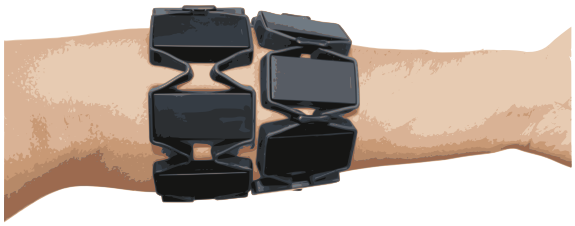


Figure 3.6: Placement of the two Myos on the forearm. They are placed on the thickest part of the forearm. The Myos have the same orientation and are shifted by 22.5 deg between each other.

3. The outer Myo is rotated slightly so that sensor 5 is aligned with the middle of the *palmaris longus* tendons.

4. The inner Myo is rotated so that it has an angle of 22.5 degrees with the first Myo, in clockwise direction (subject perspective).

The acquisition software was developed for Linux in the python programming language. It is responsible by establishing the communication between the Myos and the Bluetooth dongles, recording data and interfacing with the user. A graphical interface displays the data to the user in real time as feedback. Additionally, the recording of data is controlled by keyboard inputs.

Before recording any data, the subjects place the armbands according to the protocol previously presented and the sensors run for a few minutes to warm-up. The subjects are then asked to hold the position of a gesture for a few seconds while 2 seconds of data are recorded. The gestures are repeated in random order in several sessions. To account for offsets during the placement of the sensors, the subject took the armbands off between consecutive sessions.

3.2.1 Data Visualizations

Some data samples are shown in figure 3.7. The figure shows the normalized standard deviation across time of 16 samples for each class of gesture. The rows correspond to the interlaced Myo EMG channels, i.e., consecutive rows correspond to the physically closest electrodes on the forearm. Columns represent different observations.

Figure 3.7 shows that the signal patterns remain the same between observations for all classes. The 16 samples represented were picked randomly, so they are representative of the whole data set, showing that the acquisition procedure was done correctly. Furthermore, there is a clear difference in signal strength between G0 (rest) and the remaining classes. However, the fifth channel shows the existence of signal even in the rest state. Also, this channel does not show a constant signal along time and does not appear to be influenced by gesture class. It is probable that this channel is not accurate due to either hardware failure or poor contact.

Most of the gesture classes show distinct activation patterns. However, G5 (double tap) shows a very weak signal overall, since there is little activation of the forearm

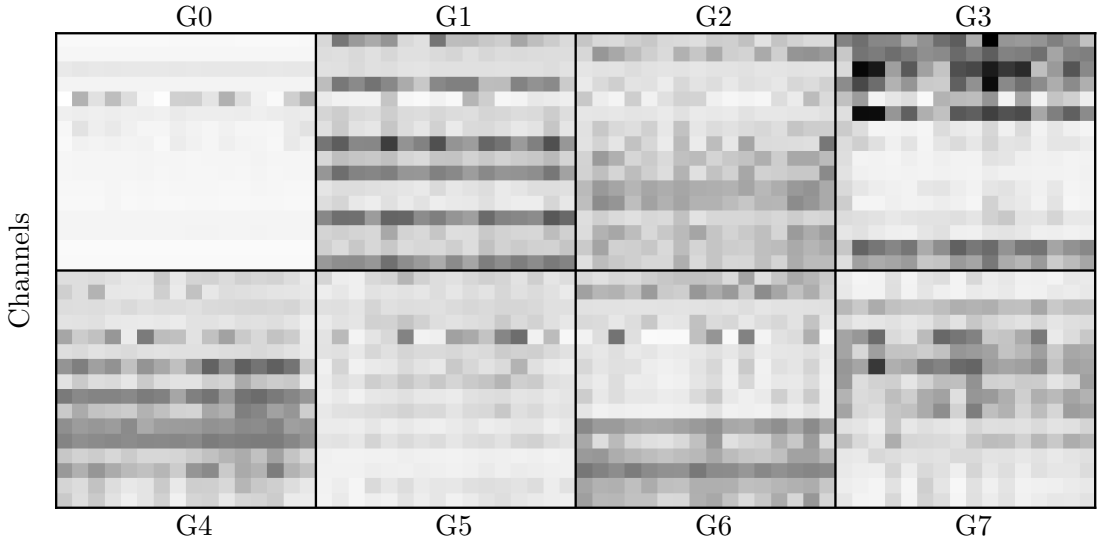


Figure 3.7: Standard deviation across time of samples from each class of gestures G0 through G7. The rows correspond to the interlaced Myo EMG channels and the columns to the observations. The signals are color-mapped so that white corresponds no muscle activation and black to maximum activation.

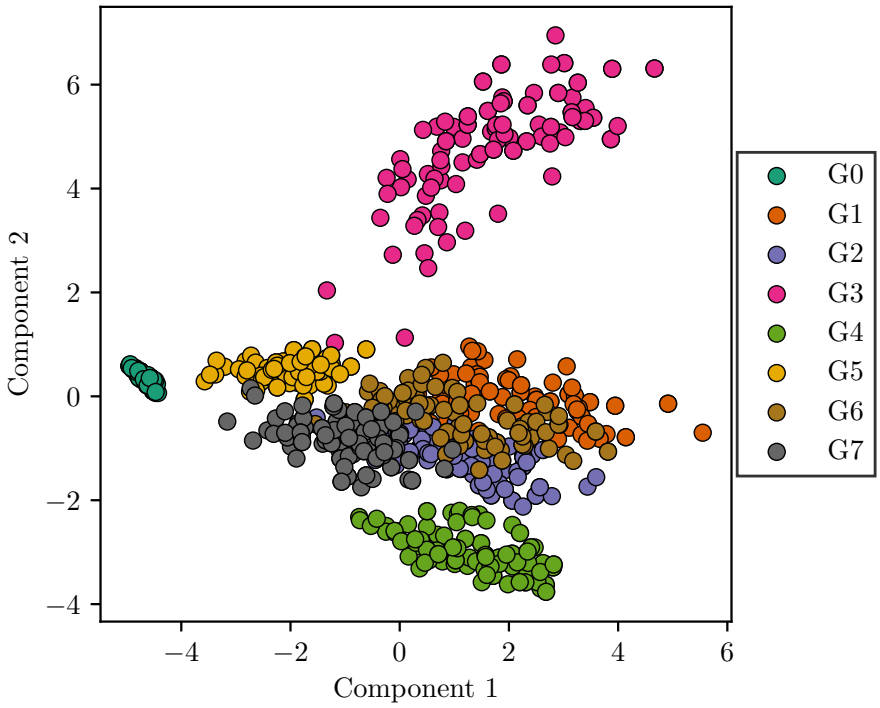


Figure 3.8: Projection of the observations of the DualMyo data set in their first two principal components. The gesture classes are represented by different colors.

muscles when just the middle finger and thumb move. It might be possible to classify this gesture, but it is probably difficult to distinguish from out-of-vocabulary patterns that may occur in the real world. All of the others show different clusters of muscle activation, since they clearly require the activation of different groups of muscle.

These features can be projected into a lower dimensionality space using PCA, for easier visualization. All of the samples of the data set are shown in figure 3.8, with the 16 channels projected into their two first PCA components. The class clusters are all very well defined. The rest class (G0) is a very small, separated cluster. Classes G3, G4 and G5 form well separated clusters with more dispersion than G0, but the class boundaries are easily drawn. The remaining classes G1, G2, G6 and G7 have well defined clusters but they overlap in this view. The classification methodologies will show that these classes are also separable.

3.2.2 Synthetic Gesture Sequences

Classifiers may also be tested on gesture sequences, rather than on gesture samples. This is particularly useful to evaluate the performance of the classifier on the boundary between gestures. Often-times, this is evaluated on test sequences performed by the subjects which are then carefully annotated to indicate where a gesture begins and ends. However, this approach is time consuming, requires human expertise, is prone to errors and does not scale well for new sequences or classes. For these reasons, we present an alternative in the form of a synthetic sequence of gestures interleaved with rest periods.

A synthetic sequence is formed by extracting a number of non-rest samples, from the data set. Two examples of sequences are shown in figure 3.9, where gestures G1 through G7 can be seen, as well as the rests between them. The rest states interleaving the samples are sampled, channel by channel, from a normal distribution parametrized by the mean and standard deviation of the signal for all rest samples (G0), derived from their definitions:

$$\mu_{j,rest} = \frac{1}{n_{rest} + l} \sum_{m=1}^{n_{rest}} \sum_{i=1}^{l} \boldsymbol{X}_{ij}^{(m)} \tag{3.2a}$$

$$\sigma_{j,rest}^2 = \frac{1}{n_{rest} + l} \sum_{m=1}^{n_{rest}} \sum_{i=1}^{l} \left(\boldsymbol{X}_{ij}^{(m)} - \mu_{j,rest} \right)^2 \tag{3.2b}$$

where $\mu_{j,rest}$ and $\sigma_{j,rest}$ are the mean and standard deviation, respectively, of the j^{th} channel calculated on all rest samples. $\boldsymbol{X}_{ij}^{(m)}$ is the i^{th} value of the j^{th} channel, for the m^{th} rest sample. n_{rest} and l are the number of rest samples and sample length.

Rest intervals between consecutive gestures are variable and follow a normal distribution of mean 5 and standard deviation 0.5 seconds. Finally, to avoid discontinuities between the rest state and the gestures, we had a short transition phase. This phase is also variable in length, following also a normal distribution

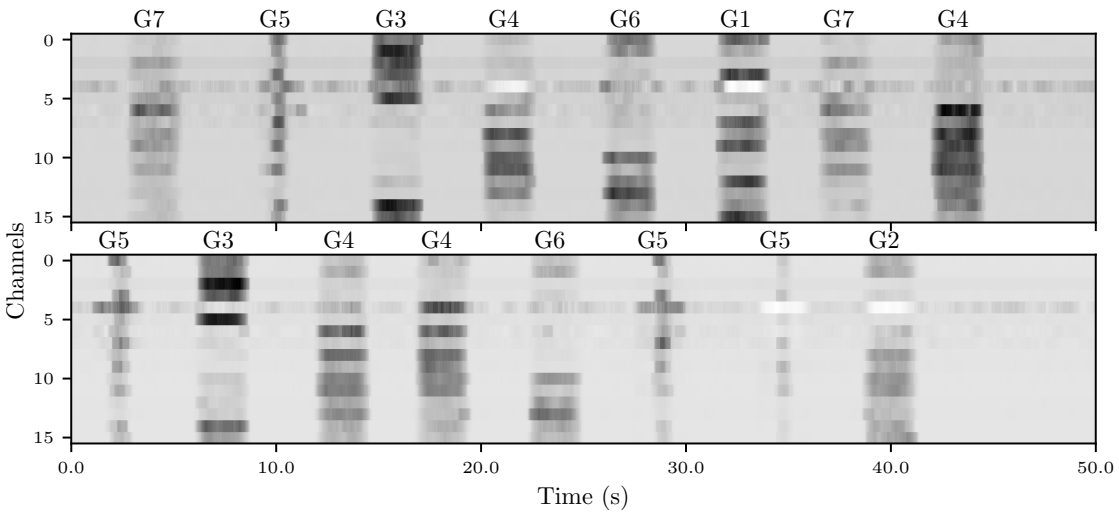


Figure 3.9: Two synthetic sequences of gesture samples from the DualMyo data set with rests in-between. The darker regions correspond to the gestures and their class is shown above them.

$\mathcal{N}(\mu = 0.2, \sigma = 0.05)$ seconds. The transition is sampled from a scaled and translated logistic function which connects the last value of a gesture or rest segment to the first value of the following segment. The logistic function is defined in the following equation:

$$f(t) = a + \frac{b - a}{1 + e^{-4.6\tau t}} \tag{3.3}$$

where t is the time variable, a and b correspond to the start and end limit values of the logistic transfer function. t is scaled by 4.6τ, where τ is the time constant, which defines the slope of the transition. The 4.6τ multiplier was tuned so that when $\tau = \pm 1$, 98% of the value change between a and b has occurred.

Finally, since the transition and rest state lengths are variable, the full sequence of n non-rest gestures also has variable length. All of the resulting sequences are padded with the sampled rest state until their length equals the sequence with the maximum length.

The data set is accompanied by a script to generate the synthetic sequences by following this process. By default, sequences of 8 gesture samples are used. This gives origin to 96 sequences with 50 seconds of length.

Chapter 4

Unsupervised Gesture Segmentation by Motion Detection of a Real-Time Data Stream

Miguel Simão[1], Pedro Neto[1], and Olivier Gibaru[2]

Based on the paper published on:
IEEE Transactions on Industrial Informatics

Abstract

Continuous and real-time gesture spotting is a key factor in the development of novel human-machine interaction (HMI) modalities. Gesture recognition can be greatly improved with previous reliable segmentation. This chapter introduces a new unsupervised threshold-based gesture segmentation method to accurately divide continuous data streams into dynamic and static blocks, without previous knowledge of gesture data (unsegmented and unbounded input data). This type of segmentation may reduce the number of wrongly classified gestures in real world conditions, improving current gesture-based HMI systems. The proposed approach identifies sudden inversions of movement direction which are a cause of over segmentation. This is achieved by the analysis of velocities and accelerations numerically derived from positional data. A genetic algorithm is used to compute optimal thresholds from calibration data. Experimental tests were based on the application of the proposed method using a data glove and a magnetic tracking device as interaction technology. An over-segmentation error of 2% was achieved in a benchmark for motion segmentation with a non-optimal sliding window size.

Keywords: segmentation, unsupervised, motion, gestures, human-machine interaction, robotics.

[1]Collaborative Robotics Laboratory, University of Coimbra, Portugal.
[2]Laboratoire d'Ingénierie des Systèmes Physiques et Numériques, ENSAM ParisTech, Lille, France.

4.1 Introduction

Flexible industrial machines in general and robots in particular are traditionally instructed either by text-based programming or by direct control, using a teach pendant or a joystick. The ability to interact with a machine in a natural and intuitive way, using speech and gestures, has brought important advances to modern industry. The paradigm for robot usage has changed in the last few years, from an idea in which robots work with complete autonomy to a scenario in which robots cognitively collaborate with human beings. This brings together the best of each partner, robot and human, by combining the coordination and cognitive capabilities of humans with the robots' accuracy and ability to perform monotonous tasks. This will allow a greater presence of robots and intelligent systems in companies, with consequent positive impact on society's life standards. The problem is that the existing interaction modalities are neither intuitive nor reliable. Instructing and programming an industrial robot by the traditional teaching method is a tedious and time-consuming task that requires technical expertise. The increasing demand by industry for robot-based solutions makes the need for intuitive human-machine interaction (HMI) more visible, especially in small and medium sized enterprises (SMEs).

Multimodal HMI interfaces combining gestures, speech and tactile based-actions are expected to be in a near future the standard for a reliable and intuitive interaction process. Nonverbal communication cues in the form of gestures are considered to be an effective way to approach intuitive HMI. For instance, a person can point to indicate a position to a robot, use a dynamic gesture to instruct a robot to move and a static gesture to stop the robot [14, 119]. In this scenario the user has little or nothing to learn about the interface, focusing on the task and not on the interaction modality [120]. For all the reasons mentioned above, continuous and real-time gesture spotting is a key factor in bridging the gap between research and real world application of novel HMI modalities.

Temporal gesture segmentation from a real-time data stream is the problem of identifying data segments that are more likely to contain meaningful interactions. This may simplify the subsequent pattern recognition, increasing its reliability. Most of the classification algorithms only present reliable results if the input data roughly represents a specific gesture on its database (previously trained – supervised method). A major challenge in continuous gesture recognition has to do with the fact that there are movement segments between gestures that have no meaning. Such inter-gesture transition periods are transition frames and are known as movement epenthesis (ME). Most studies treat ME as a classification problem [5, 121].

An effective and valuable segmentation is achieved with no previous knowledge of gesture data, no training and no information about the next gesture in the sequence. In general, there are two major difficulties in the segmentation process of a real-time stream of data:

1. Stream unboundedness: no information about when a gesture starts and ends

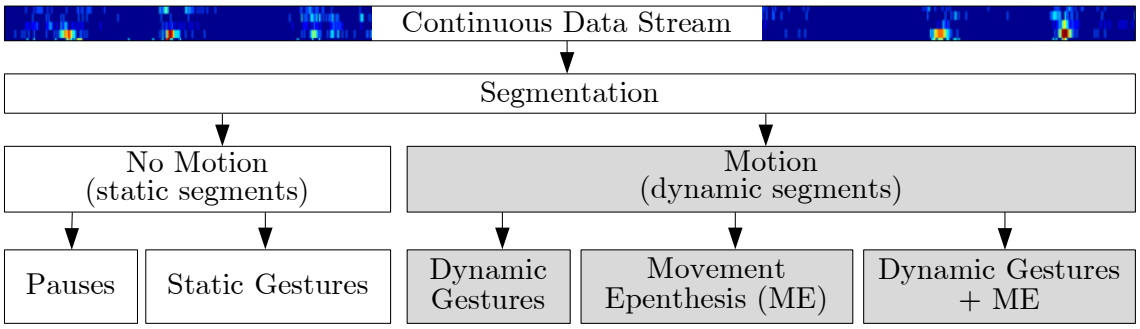


Figure 4.1: The role of segmentation by motion in gesture recognition from a continuous data stream.

in a continuous sequence;

2. Spatio-temporal variability: a gesture may vary in shape, duration and trajectory, even when it is performed by the same person.

Fig. 4.1 highlights the role of segmentation by motion as a previous procedure before the classification/recognition of static gestures, dynamic gestures and ME. If the aim is to classify static gestures, segmentation by motion solves the entire problem of segmentation due to the fact that in the presence of static segments, the data recorded from the interaction technologies will serve as input only to the static gesture pattern classifier. On the contrary, when there is motion, that set of data is not used to feed the static gesture classifier. In the presence of dynamic blocks, the recorded data can have different meanings, i.e., the segment can be classified either as a dynamic gesture or as ME. This is a classification problem in which the dynamic gestures and ME can be correctly recognized if they were previously trained. ME classification based on its training is totally non-natural so that some authors exclude ME from effective trained gesture patterns or apply a weakly supervised training approach.

4.1.1 Problem Specification and Challenges

A major problem in gesture-based HMI is related with the reliable recognition of gestures continuously from real-time streams. Continuous gesture recognition is the natural way used by humans to communicate, in which communicative gestures (with an explicit meaning) appear intermittently with pauses and ME, without a specific order.

It is difficult to accurately segment continuous data streams to feed the classifiers. There is no standard solution for gesture segmentation, it depends on several factors: (1) interaction technologies, (2) classification method (supervised or unsupervised), (3) if gestures are static, dynamic or both, (4) if ME was previously trained or not, among other factors. Another problem in segmentation is related with the difficulty of creating a stable system that eliminates the appearance of false gestures (false positives) and false negatives (leading to over segmentation). Sometimes, the system

is too sensitive in the process of detecting motion. This may be related with (1) the interaction technologies with their inherent noise and resolution, and (2) the natural human shaking.

In the context of gesture segmentation, it can be stated that false negatives are more costly than false positives since they divide the data representing a dynamic gesture into two sections, completely corrupting the meaning of that gesture. False positives are more easily accommodated by the classifier, which just reports that the pattern is not a trained gesture. According to all of the above, several challenges related with gesture segmentation can be pointed out:

1. Creating an unsupervised segmentation technique that is robust enough to accurately divide a data stream into dynamic and static blocks without false negatives and false positives, thus avoiding over-segmentation;

2. Avoiding false negatives in the segmentation of dynamic gestures by anticipating inversions of movement;

3. Achieving real-time performance and being able to segment gestures in continuous streams. In the presence of unconstrained spatiotemporal variations there is no information about the next gesture in the sequence;

4. Being user independent.

4.1.2 Related Work

Off-line analysis of gesture segmentation by motion detection has been applied with relative success, for example by computing the variance of motion data and applying a threshold that defines if motion exists or not [122], or in the detection of motion in video sequences [123]. Nevertheless, the desired solution for intuitive HMI requires real-time and continuous gesture recognition. The evolution of HMI lead to the necessity of understanding natural human motion [124]. Many existing segmentation techniques use the backward spotting scheme to first detect the end point of a gesture and then traces back to the starting point. The problem is that in this methodology, as in the cases shown above, the real-time character of the system is lost, making it unsuitable for continuous gesture recognition [125, 126]. This problem can be solved by implementing a forward spotting scheme for simultaneous gesture segmentation and recognition in which the start and end points of a gesture are determined by zero crossing from negative to positive (and vice versa) of a competitive differential observation probability [125, 126]. In simple words, this is an automatic threshold method based on the comparison of the probability of a given frame being a gesture or non-gesture. The problem is that this method is supervised, requires the previous training of gestures and sequences.

A reference study in the field reports the application of an adaptive threshold [127]. Such adaptive threshold is based on the addition of an additional label to the Conditional Random Field (CRF) model to overcome the weakness of the fixed threshold method. However, initial training is necessary for the CRF.

Interesting studies in the field propose a method for spotting gestures in continuous data by using Hidden Markov Models (HMMs) to detect the endpoint of a given gesture [128]. Lee and Kim developed a method deploying HMMs to spot gestures in a continuous stream of sensor data [129]. They apply a threshold model that calculates the likelihood threshold of an input pattern. The start and end points of a gesture are defined by comparing the threshold model with predefined gesture models.

The automatic video temporal segmentation recurring to an algorithm that computes the frame difference/similarity by analyzing pixel and histogram difference was studied as in [130]. Other authors propose a fuzzy machine vision-based framework for humans' behavior recognition that relies on the analysis of feature cues reporting the human silhouette [131]. Meaningful movements can be recognized while concurrently separating unintentional movements from a given image sequence [132]. The importance of the selection and adaptation of the window data length and its dependency on the complexity, duration and granularity of the human activities to recognize is demonstrated in [133]. An analysis of motion segments using Principal Component Analysis (PCAs) to represent hand motion is in [134]. This method allows to reduce the data dimensionality but according to our experience and real-time requirements this appears to be a computationally expensive solution in which important information is lost. An automated process of segmenting gesture trajectories based on a simple set of threshold values is proposed in [135]. A likelihood-based state segmentation was implemented in addition to the threshold-based method to ensure that the gesture sets are segmented consistently (the trajectories are previously trained). The gesture segmentation process is also highlighted in a study on vision-based action recognition for the human entire body considering a large vocabulary of gestures [136]. A different approach is to study the perception of human postures and gestures recurring to a marker-less vision-based solution for upper body tracking with multiple cameras [137]. An unsegmented/unbounded vision-based continuous gesture recognition system is presented in [120]. For a library of 10 body-and-hand gestures, the achieved recognition accuracy was 94% for isolated gestures and 88% for continuous gestures. This clearly shows the importance of segmentation for continuous gesture recognition.

An approach to gesture spotting in a continuous data stream from body-worn inertial sensors is presented in [138]. Other authors use temporal segmentation based on the discontinuity of the movements [139]. Such an approach allows the recognition of gestures that were defined in vocabularies only, i.e., non-gesture patterns are not considered. A recent study reports the separation of acceleration and surface electromyography signals as a mean to segment gesture data [140].

Regarding the role of ME in gesture segmentation, some authors try to solve this problem by developing temporal gesture models which address the problem of ME detection without the need for explicit epenthesis training [141, 121], while other authors train ME [14, 127, 129]. In both cases ME is analyzed in the context of a classification problem requiring previous training. Previous studies demonstrated that approaches that explicitly model ME yield better results than those ignoring ME [14]. However, modeling ME is difficult and its complexity increases exponen-

tially with the number of distinct gestures.

Complex human activity recognition in sensor rich environments has been studied in the last few years [142, 143, 144]. A multimodal segmentation method that merges information from inertial sensors (IMUs), muscle activity and location (RFID) is presented in [143]. A string-matching-based segmentation and classification method is also presented to recognize activity using wearable devices with limited computational power [143]. Online gesture recognition for wearable computing purposes based on crowdsourced annotations is in [144]. In this study segmentation and warping longest common subsequence algorithm (LCSS) are applied as template matching methods. Wearable accelerometers have been successfully applied for human activity recognition. A framework for the recognition of motion primitives relying on Gaussian mixture modeling (GMM) and Gaussian mixture regression (GMR) is presented in [145, 146]. A recognition procedure based on dynamic time warping (DTW) and Mahalanobis distance is proposed to ensure run-time classification. A reliable method for real-time continuous human action recognition with embedded segmentation is presented in [147]. However, this system is trained with previous segmented data.

Analyzing existing representative studies allows us to conclude that most studies approach the segmentation problem together with classification, requiring previous training – supervised methods. This work proposes to treat segmentation by motion separately in an unsupervised and computationally inexpensive fashion. This results in cleaner data to feed pattern classification methods.

4.1.3 Proposed Approach and Overview

Real-time segmentation relies on the comparison of the current state (frame) $\mathbf{f}_i$ with the previous states, $\{\mathbf{f}_{i-1}, \ldots, \mathbf{f}_{i-n}\}$. This work proposes a new unsupervised threshold-based gesture segmentation method to accurately divide continuous data streams into dynamic and static blocks with minimal error and avoiding under and over-segmentation (false negatives and false positives). The proposed approach identifies inversions of movement direction resorting to the analysis of velocities and accelerations derived from positional data.

A fixed threshold is a very limitative solution for motion segmentation. As mentioned in the previous section, existing studies present excellent results using automatic threshold methods or adaptive thresholds for gesture segmentation [125, 126, 127]. However, such approaches are supervised, requiring a significant amount of training data from specific users.

This chapter proposes a novel method to segment a continuous data stream into dynamic and static blocks in an unsupervised fashion, i.e. without previous training or knowledge of gestures and the sequence, unsegmented and unbounded. It is proposed to establish an optimal single threshold for each feature using a genetic algorithm – because the performance function is not linear – fed by nonlinear calibration data. Gesture patterns with sudden inversions of movement direction are analyzed recurring to the analysis of velocities and accelerations numerically derived from positional data. The proposed method deals with gesture motion patterns

varying in scale, rate of occurrence and different motion constraints. A sliding window addresses the problem of spatio-temporal variability.

The proposed system was evaluated by conducting a set of experiments using a set of continuous dynamic and static gestures. Experimental tests were carried out using wearable/body-worn sensing as interaction technologies: a data glove and a magnetic tracking device. The tests demonstrated that:

1. Motion is always detected for any kind of gesture in a continuous sequence of unsegmented and unbounded data;

2. It is an unsupervised method, no prior training is required;

3. A quantitative analysis indicates an over-segmentation error of 2% using a suboptimal sliding window size;

4. Segmented data present in most cases a shift-right and extend behavior in relation to the ground truth. The extend behavior has a very limited magnitude and the shift-right is due to several factors that are not directly related with the proposed method, namely feature capture and processing delay;

5. The automatic optimization of the thresholds vector demonstrated reliable behavior;

6. Features related with velocity are important but have to be combined with acceleration features to deal with sudden inversions of movement direction;

7. The proposed fine segmentation method reduces the noise in data for subsequent classification;

8. The system accepts any kind of sequential positional data as input (or change in position) for a number of different sensors reporting positional data: magnetic, inertial, vision, etc.

Section 4.2 gives an overview of the proposed segmentation method. The sliding window threshold decision method is presented, as well as the method to optimize the threshold vector. It is also discussed what variables/features are more relevant. Section 4.3 describes experiments and results. Section 4.4 concludes with a summary of contributions and suggesting directions for future work.

4.2 Gesture Segmentation

4.2.1 Sliding Window Threshold Decision Method

The absence of movement can be defined by the lack of change in every degree of freedom (DOF) of a given system. At rest, natural human body shaking generates noisy DOF difference functions. These functions are part of what we call motion features. To separate their noise from an actual static stance, there was a need

to set a threshold for each of the features. We consider that there is motion if there are motion features above the defined thresholds. Section 4.2.2 shows how the optimal thresholds are achieved. These thresholds should be easily recalibrated for any individual. The threshold is a vector, $\mathbf{t}_0$, with a length equal to the number of motion features chosen, n_t. The features obtained from a certain frame are represented by the vector $\mathbf{t}_j$, in which $1 \leq j \leq n_t$. The sliding window $\mathbf{T}$ is composed of w consecutive frames of $\mathbf{t}$. At an instant i, the real-time sliding window $\mathbf{T}(i)$ is given by:

$$\mathbf{T}(i) = \left[\begin{array}{cccc} \mathbf{t}(i-w+1) & \cdots & \mathbf{t}(i-1) & \mathbf{t}(i) \end{array} \right] \tag{4.1}$$

At each instant i, the w sized window slides forward one frame and $\mathbf{T}(i)$ is updated and evaluated. To further increase the stability of this method, a static frame is only acknowledged as such if none of the motion features exceed the threshold within the sliding window. This means that a frame will not be classified as static if there is within the window at least one feature above its threshold. This way, we guarantee that a motion start is acknowledged with minimal delay (real-time). On the other hand, this also causes a fixed delay on the detection of a gesture end, equal to the size of the window w. This is not necessarily a problem because it is possible to accurately classify a dynamic gesture even before it ends. Moreover, it is possible to remove those delay frames a posteriori from the dynamic gesture feature matrix.

The proposed method to achieve the motion function $m(i)$ relies in the computation of the infinite norm of a vector $\mathbf{a}$ that contains feature-wise binary motion functions:

$$m(i) = \begin{cases} 1, & \text{if } \|\mathbf{a}\|_\infty \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

where vector $\mathbf{a}$, for each instant of time i, is calculated by comparing the sliding window with the threshold vector:

$$\begin{aligned} \mathbf{a}_m = (\max_n |\mathbf{T}_{mn}| \geq k_s \cdot \mathbf{t}_{0_m}), \quad & m = 1, \ldots, n_t, \\ & n = 1, \ldots, w \end{aligned} \tag{4.3}$$

in which k_s represents a user-defined threshold sensitivity factor. A possible result for a single feature $\mathbf{t}_m$ is shown in Fig. 4.2. A motion segment starts as soon the feature rises above the threshold (green area). On the other hand, even after the feature drops below the threshold, the segment is not finished until a full window of frames is below the threshold.

4.2.2 Threshold Optimization

Calibrating the threshold $\mathbf{t}_0$ in an unsupervised fashion is not a straightforward task because it depends on several factors, namely the input data/devices, selected features, sliding window size and the device wearer himself. If the parameters are set too low, it can become oversensitive and generate too many false positives, declaring

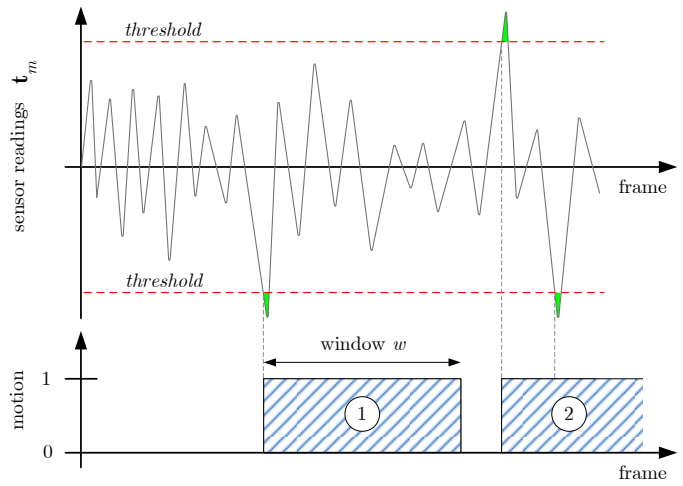


Figure 4.2: Sliding window threshold decision method for motion detection considering a single feature.

motion when there is noise. If they are set too high, it will not be capable of detecting slow movements, generating false negatives and over-splitting the stream. Thus, a balance must be found to avoid these issues. Since data streams are generally high-dimensional, the manual determination by trial and error of the thresholds is difficult. There are two main challenges to take into account when solving this problem:

1. Being aware that the choice of motion features has a great influence on the result;

2. Selecting t_0 and w parameters that perform well for most individuals.

To perform the parameter optimization, the authors recommend obtaining two distinct sets of data (which we call calibration data) with equal length/time t_s:

1. A static sample $\mathbf{C}_S$, with dimension $n_t \times t_s$, recorded with the user at rest. For this set, the target motion function is constant and equal to 0, with dimension $1 \times (t_s - w + 1)$;

2. A motion sample $\mathbf{C}_M$, with dimension $n_t \times t_s$, recorded with the user performing gesture-type hand movements. These movements are done randomly and should trigger every motion feature. The target motion function is constant and equal to 1, with dimension $1 \times (t_s - w + 1)$.

The matrices $\mathbf{C}_S$ and $\mathbf{C}_M$ establish what we call the ground truth for the process of calibrating the features' thresholds $\mathbf{t}_0$. This is not considered training as it is performed by the user in a short span of time and the samples do not need to have meaningful gestures.

The proposed calibration process uses an optimization method to minimize the error of the sliding window method applied to the calibration data. The window

size is kept fixed and the variables are the motion features' thresholds. The error E is the objective function to minimize, (4.4). It has two terms, one corresponding to error for the static sample, e_S, and the other corresponding to motion sample, e_M.

$$E = \sum_{i=1}^{t_s} e_S(i) + e_M(i) \tag{4.4}$$

where e_s and e_m are binary functions computed from the samples $\mathbf{C}_s$ and $\mathbf{C}_m$:

$$e_S(i) = \begin{cases} 0, & \text{if } m(C_S(i)) = 0 \\ 1, & \text{if } m(C_S(i)) = 1 \end{cases} \tag{4.5a}$$

$$e_M(i) = \begin{cases} 0, & \text{if } m(C_M(i)) = 1 \\ 1, & \text{if } m(C_M(i)) = 0 \end{cases} \tag{4.5b}$$

It is recommended to calibrate each threshold individually in order to simplify the optimization problem and to improve the robustness of the final solution. This is achieved by implementing the motion function $m(i)$ with a single motion feature. This way, each threshold is optimized to approximate the segmentation output to the ground truth. If the optimization problem is set to optimize all the thresholds at once, some thresholds might be better optimized than others. The result might be as good as with the calibration data, but worse with new data.

Since the objective function is nonlinear and non-smooth, the chosen algorithm was a genetic algorithm. The nonlinear Generalized Reduced Gradient [148], was also tried. However, this method is more suitable for objective functions that are differentiable, which is not the case. This lead to the use of a genetic algorithm to find an optimal solution. The optimization algorithm benefits from having its variables constrained. The lower limit should be zero, and the upper limit can be the maximum values of the ground truth motion features.

The size of the sliding window w can also be adjusted at will. Setting a larger window will surely improve the stability and help eliminate false negatives, reducing over-splitting. Over-splitting occurs frequently in gestures that have inversion of the direction of the motion because velocity-dependent features are at their lowest on that point. Increasing the window size helps to reduce this problem. On the other hand, increasing the window size also increases the delay of the detection of a gesture end event. This delay should be reduced as much as possible to keep real-time usage as possible. Figure 4.3 shows that for this use case there are different values for the thresholds and sliding window size that when combined conduct to zero error. The proposed algorithm for the motion function in pseudocode is shown in Algorithm 1.

4.2.3 Motion Features

As mentioned before, in an ideal system, the absence of movement would be defined by null differences of the system variables between frames. Therefore, the simplest set of features that can be used for this method is the frame differences, $\triangle\mathbf{f}$, that at an instant i is given by:

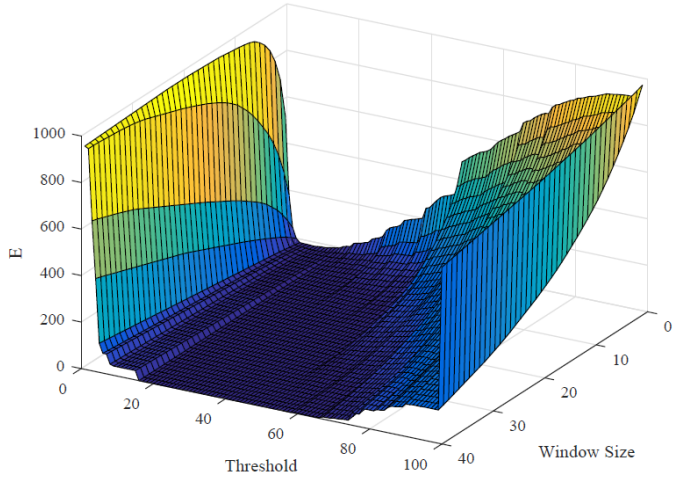


Figure 4.3: Error dependence on threshold value and sliding window size.

Algorithm 1 SlidingWindow($\boldsymbol{O}$,n,$\boldsymbol{t}$,k_s,w)

inputs: $\boldsymbol{O}$ observations matrix
n number of observations to evaluate
$\boldsymbol{t}$ threshold vector
k_s threshold sensitivity factor
w window size
output: m motion function

for $i \in [1,\text{LENGTH}(\boldsymbol{t})]$ **do**
  $\boldsymbol{t}^{(i)} \leftarrow k_s \cdot \boldsymbol{t}^{(i)}$
end for
$l \leftarrow n + w - 1$
$\boldsymbol{F} \leftarrow \text{GETFEATURES}(\boldsymbol{O})$ ▷ Calculate the features from the obtained observations.
for $i \in [1, l]$ **do**
  $\boldsymbol{m}(i) \leftarrow 0$
  for $j \in [1, \text{LENGTH(t)}]$ **do**
    $m(i) \leftarrow (\boldsymbol{F}^{(i,j)} \geq \boldsymbol{t}^{(i)}) \vee m(i)$
  end for
end for
for $i \in [1, n]$ **do**
  for $j \in [1, w-1]$ **do**
    $m(i) \leftarrow m(i) \vee m(i+j)$
  end for
end for

$$\Delta \mathbf{f}\,(i) = \mathbf{f}\,(i) - \mathbf{f}\,(i-1) \tag{4.6}$$

However, these features do not yield consistently reliable results. For example, if we consider a system with a tracker that measures the position in Cartesian coordinates, this approach performs poorly, since the differences would be relative to the coordinated axis. A motion pattern with a direction oblique to an axis would have lower coordinate differences compared to a pattern parallel to an axis with similar speed, thus producing different results. This issue can be solved by replacing the three coordinate differences with the respective Euclidian length, $d(i)$, which is a value proportional to the average speed between frames, here denominated by $v(i)$.

$$v\,(i) = \sqrt{\Delta x(i)^2 + \Delta y(i)^2 + \Delta z(i)^2}, \quad i \in \mathbb{R}^+ \tag{4.7}$$

where:

$$\Delta x(i) = |x\,(i-1) - x\,(i)| \quad \subset \quad \Delta \mathbf{f} \tag{4.8a}$$

$$\Delta y(i) = |y\,(i-1) - y\,(i)| \quad \subset \quad \Delta \mathbf{f} \tag{4.8b}$$

$$\Delta z(i) = |z\,(i-1) - z\,(i)| \quad \subset \quad \Delta \mathbf{f} \tag{4.8c}$$

Another common problem is when the user performs gesture patterns with sudden inversions of direction. The performance of a dynamic pattern recognition system is highly dependent on the precedent segmentation, so these types of false negatives are very detrimental to the classifier accuracy. The proposed solution is adding an extra motion feature, the acceleration, $\dot{v}(i)$. The acceleration is at its highest when an inversion of direction occurs, which solves the low velocity problem. This feature does not cause false positives in a static gesture and deals successfully with the inversions of movement on dynamic gestures. The average acceleration between frames is proportional to the difference of velocities between frames:

$$\dot{v}\,(i) = v\,(i) - v\,(i-1) \tag{4.9}$$

In Fig. 4.4 the shaded areas imply that a variable is below the threshold, thus meaning an absence of motion in the respective variable. There would be a problem if the shades overlapped, because if all the variables are below the threshold, the conditions for triggering a pause are met. If the shades do not overlap, a pause is not declared, resulting in the elimination of the false negative.

4.3 Testing Methodology

The performance of the proposed segmentation by motion methodology was evaluated in different tests. Experimental tests were conducted with a participant wearing

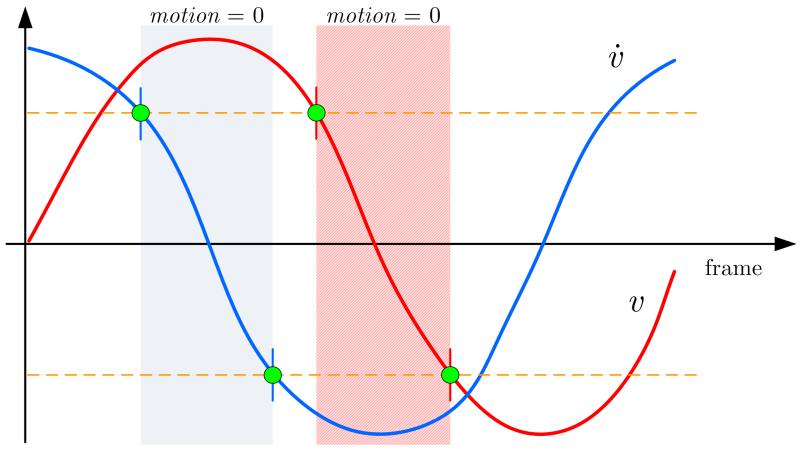


Figure 4.4: Example of kinematic quantities in a dynamic gesture with sudden direction change. The flat shade corresponds to acceleration below the threshold, while the dotted shade corresponds to a velocity below the threshold. If they overlap during a large enough number of frames, a false negative will be triggered.

a data glove and a magnetic tracker device (interaction technologies). The participant performed a number of gestures (static, dynamic and ME) while the segmentation system was detecting motion in real-time from a gesture dataset. The devices and the segmentation system are synced by a video recording device to capture the ground truth data. The results are then compared and discussed.

4.3.1 Interaction Technologies and Data Acquisition

Two different sensors are used to acquire human behavior during the interaction process: a magnetic tracker device (hand and arm motion) and a data glove (finger motion), Fig. 4.5. The devices are both connected to a computer running Matlab.

The electromagnetic tracker (Polhemus Liberty) has low latency and outputs at a rate of 120 Hz. The tracker provides the position and orientation of the sensor (along and around x, y and z) in relation to a magnetic source in a total of 6 DOF, $\mathbf{l}(i) = (l_1, l_2, \ldots, l_6)$, in which i represents a frame at a certain instant of time. The first three indexes describe the position along the three coordinated axes, $(x, y, z)_i = (l_1, l_2, l_3)_i$. The last three indicate the angles yaw, pitch and roll, respectively, $(\Psi, \theta, \phi)_i = (l_4, l_5, l_6)_i$.

The data glove (CyberGlove II) has 22 resistive bend sensors integrated: three flexion sensors per finger, four abduction sensors, a palm-arch sensor, and sensors to measure wrist flexion and abduction. The glove provides the hand shape by giving the angles of a number of the hand's joints at a certain instant of time, $\mathbf{g}(i) = (g_1, g_2, \ldots, g_{22})_i$. The sensors output real-time digital joint-angle filtered data at an average rate of 100 Hz. Timestamps were not available for this device, but the system gives a low-accuracy timestamp for each frame based on software running time, t_g.

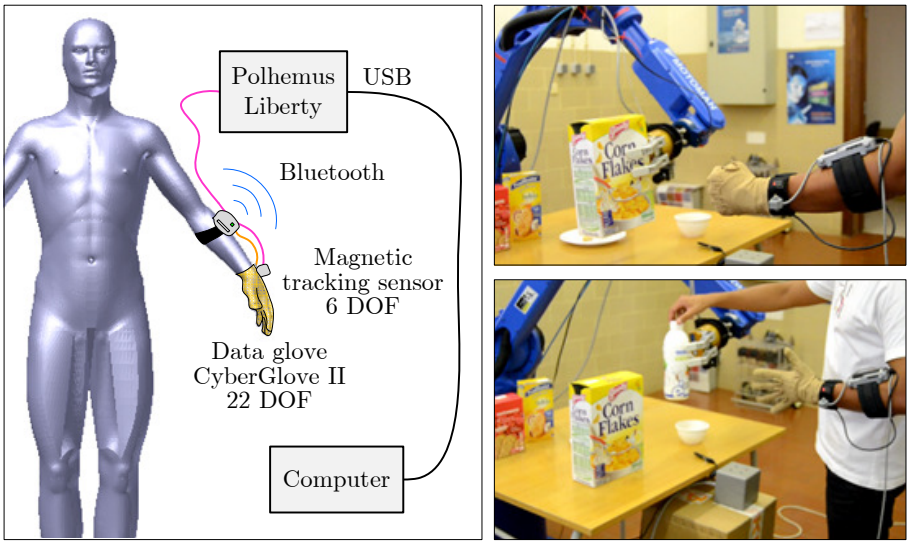


Figure 4.5: Interaction technologies and system setup.

Data from the glove and the tracking sensor are acquired and saved as vectors on device-specific buffers. These buffers are then sampled for the newest sampels and processed at a rate of 20 Hz. Since the sensors have different rates, the remaining (older) frames of the faster device are dropped. The frames from both devices are then concatenated into one single vector $\mathbf{f}$ (4.10) and put in a circular buffer.

$$\mathbf{f}(i) = (t_l, l_1, l_2, \ldots, l_6, t_g, g_1, g_2, \ldots, g_{22})_i \tag{4.10}$$

During sampling, in order to synchronize the system's output with the actual gesture (ground truth), the sampling sessions were recorded using a consumer grade camera at 30 Hz. The actual gestures were recorded as well as the system's graphical user interface (GUI), which graphically shows the segmented data stream.

4.3.2 Gesture Dataset

According to the functionalities to be achieved and based in [14], a dataset of gestures was created to test the developed method. It is our aim to have gestures containing fingers and arm motion from wearable sensing data, with different lengths (0.5 to 2 seconds), including the transition data between gestures. Existing gesture datasets rarely consider finger motion, but on the other hand, they have full body motion. The proposed dataset contains various static and dynamic gestures performed continuously.

We selected 8 gestures to be performed in a certain sequence containing static gestures, dynamic gestures and ME, Fig. 4.6. This sequence, known as Sequence 1, was performed 11 times by a participant, summing up to 88 gesture samples. The sequence samples have an average of 16 seconds of data recorded at a rate of 100 Hz. Further validation was performed on a second sequence (Sequence 2) with 10 other gestures to be performed in sequentially, Fig. 4.6. This sequence was performed

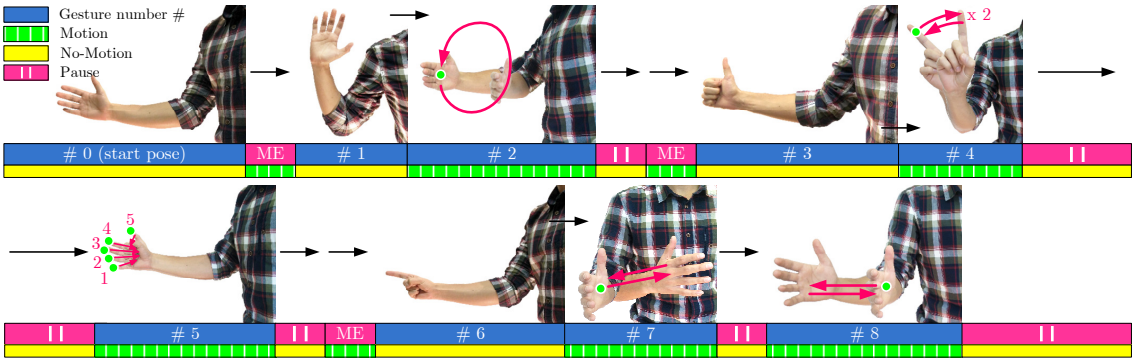


Figure 4.6: Gesture sequence performed during the sampling process to create Sequence 1 samples.

10 times by each of the participants (two subjects, A and B), summing up to 200 gesture samples.

There is an infinite number of possible combinations of individual gestures to create a gesture sequence. A representative sequence that triggers most of the variable's thresholds was selected:

0. Starting pose
1. Gesture 1 (static): stop
2. Gesture 2 (dynamic): move on
3. Gesture 3 (static): thumb up
4. Gesture 4 (dynamic): no
5. Gesture 5 (dynamic): close fingers in order
6. Gesture 6 (static): point
7. Gesture 7 (dynamic): move left
8. Gesture 8 (dynamic): move right

4.3.3 Motion Features and Sliding Window

The proposed motion features are:

1. Wrist velocity (1 DOF);
2. Wrist acceleration (1 DOF);
3. Wrist angular velocity (1 DOF);

4-25) Finger joints angular velocities (21 DOF).

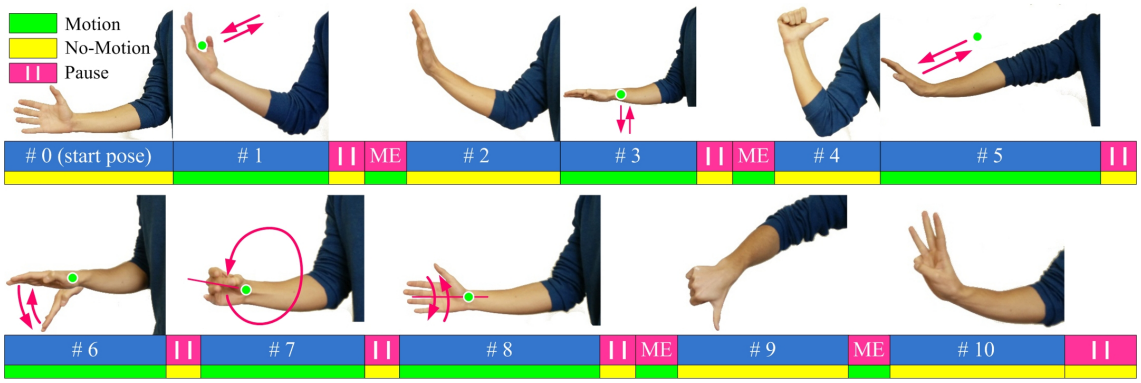


Figure 4.7: Combination of the 10 gestures that compose the validation sequence (Sequence 2).

These features are organized in a feature vector $\mathbf{t}$:

$$\mathbf{t}(i) = \left[\begin{array}{cccccc} v(i) & \dot{v(i)} & \omega(i) & \dot{g}_1(i) & \cdots & \dot{g}_{22}(i) \end{array} \right]^T \tag{4.11}$$

The wrist velocity $v(i)$ is defined by:

$$v\left(i\right) = \frac{\sqrt{\triangle l_1^2 + \triangle l_2^2 + \triangle l_3^2}}{t_l\left(i\right) - t_l\left(i-1\right)} \tag{4.12}$$

where $\Delta l_h = l_h(i) - l_h(i-1)$, $1 \leq h \leq 3$. The wrist acceleration:

$$\dot{v}\left(i\right) = \frac{\left|v\left(i\right) - v\left(i-1\right)\right|}{t_l\left(i\right) - t_l\left(i-1\right)} \tag{4.13}$$

The wrist angular velocity is:

$$\omega\left(i\right) = \frac{\sqrt{\triangle l_4^2 + \triangle l_5^2 + \triangle l_6^2}}{t_l\left(i\right) - t_l\left(i-1\right)} \tag{4.14}$$

where $\Delta l_q = l_q(i) - l_q(i-1)$,$1 \leq q \leq 3$. The finger joints angular velocities are:

$$\dot{g}_n\left(i\right) \quad = \frac{\left|g_n(i) - g_n(i-1)\right|}{\Delta t_g},\ n = 1, 2, \ldots, 22 \tag{4.15}$$

In (4.15), Δt_g is the average period of the glove's output (10 ms), since we do not have accurate frame timestamps for this device.

4.3.4 Threshold Calibration

Threshold calibration is a previous step for the testing phase. For this end, the authors obtained 2 samples of 10 seconds each of motion features during static and moving poses.

The window size was kept fixed at 20 frames (200 ms) in order to minimize segmentation delay. The calibration was implemented using Solver's Evolutionary

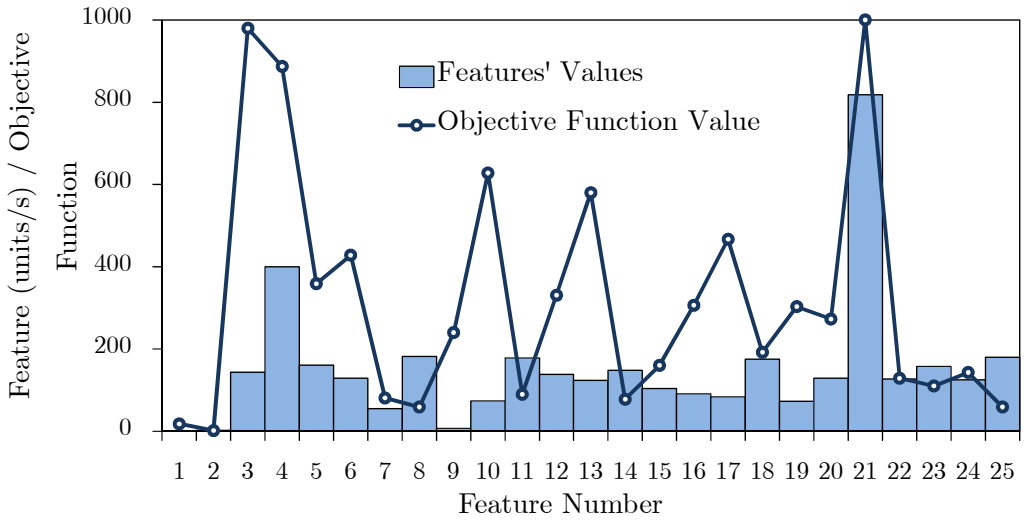


Figure 4.8: Optimized thresholds and the objective function values for each of the features.

Algorithm. The algorithm was run for each feature individually with a population size of 100 and a mutation rate of 0.075. The local minima are shown in Fig. 4.8. Some of the features have very high objective function values (error next to 1000), which is explained by poor correlation between the respective feature and the samples' segmentation ground truth (features 3, 4 and 21). This is most likely caused by low activation of some sensors by the sampling movements. Nevertheless, when the method is used with all the thresholds applied concurrently, the objective function drops to the size of the window size. This amount of error is anticipated by the method and is the minimum attainable, so the authors considered the optimization to be complete. A threshold sensitivity factor was manually determined during real-time usage with random movements. The value used for the tests was 3.0. No further calibration was done.

4.3.5 Tests

Literature demonstrates that different authors propose different quantitative parameters to measure segmentation accuracy. This study proposes the following performance parameters:

1. Average start delay, $\overline{\Delta s}$ (ms);
2. Average end delay, $\overline{\Delta e}$ (ms);
3. Segmentation error percentage, S_{error} ;
4. Extend level, EL.

The start delay represents the delay between the start of the motion (ground truth) and the timing in which the proposed system detects such motion, including all processing and communications delay. It is calculated by:

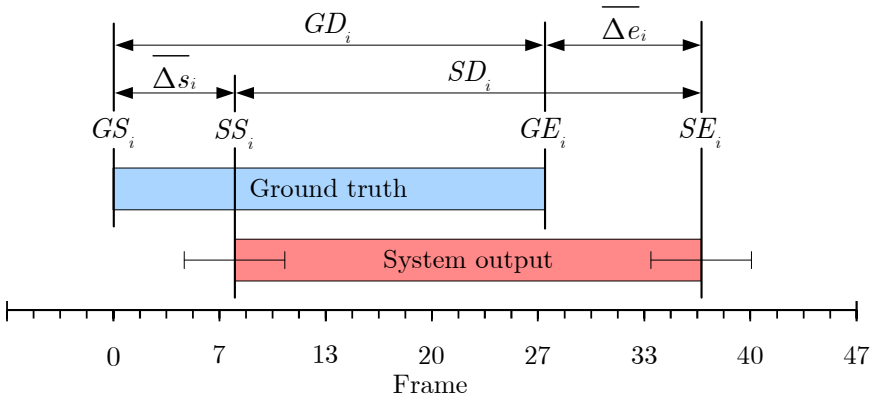


Figure 4.9: Example of an output segment and the corresponding ground truth.

$$\overline{\Delta s_i} = \frac{\sum_i SS_i - GS_i}{N} \cdot \frac{1000}{FR} \tag{4.16}$$

As shown in Fig. 4.9, SS_i is the segmentation start frame number for gesture sample i, GS_i is the ground truth gesture start frame, FR is the video capture frame rate, and N is the total number of samples.

The end delay represents the delay between the end of the motion (ground truth) and the time of which the proposed system detects the end, including all processing and communications delay. It is calculated by:

$$\overline{\Delta e_i} = \frac{\sum_i SE_i - GE_i}{N} \cdot \frac{1000}{FR} \tag{4.17}$$

where SE_i is the segmentation end frame number for gesture sample i and GE_i is the ground truth gesture end frame for gesture sample i.

The segmentation error S_{error} is the fraction between the number of segmentation errors (the sum of the number of times a gesture is over split and false segments of motion) and the number of samples. For example, if the ground truth segment is split into three, the number of segmentation errors is 2. If one segment appears during a pause, one segmentation error is added.

$$S_{error_i} = \frac{\#\text{ of segmentation errors}}{N} \cdot 100 \tag{4.18}$$

The extend level, EL, measures the discrepancy in length between the ground truth and the system output. This indicates that relevant data for the gesture recognition is captured but some noise is also captured.

$$EL_i = \frac{SD_i - GD_i}{GD_i} = \frac{\overline{\Delta e_i} - \overline{\Delta s_i}}{GD_i} \tag{4.19}$$

Determining the start and the end of the gesture segments (both the system's output and the ground truth), was done by manual inspection of the recorded video samples. During the inspection, there are 4 values per gesture acquired: GS_i, SS_i,

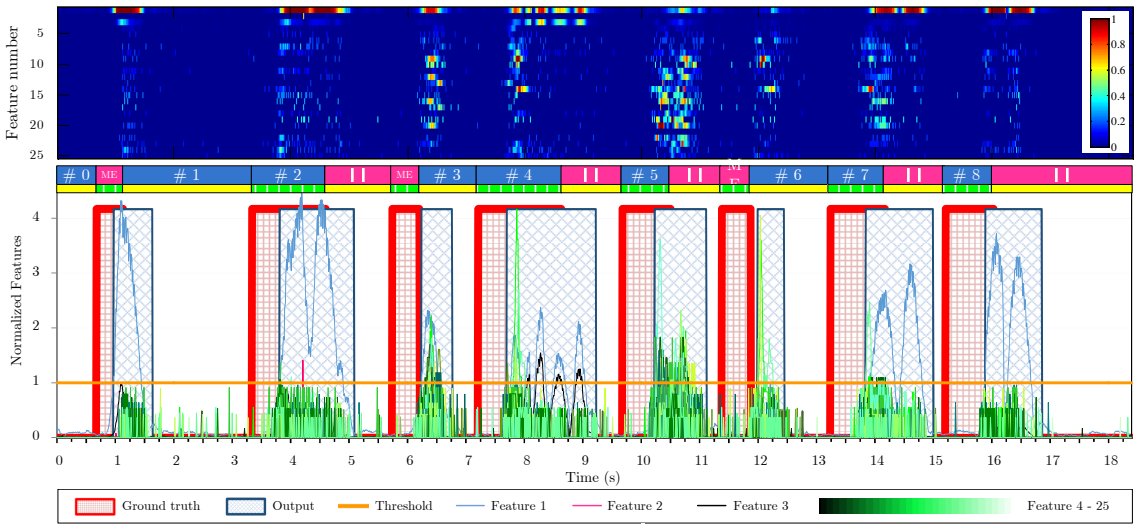


Figure 4.10: Data from one of the samples of Sequence 1. On the top, the features over time are normalized and colormapped. Below that, the features are normalized by the threshold and plotted over time. The bottom plot shows the number of features above the threshold at any point in time.

GE_i and SE_i. There is a somewhat high degree of inaccuracy with this analysis because determining which video frame represents the beginning of a gesture is not trivial. It is complicated for an individual to visually discern the beginning of a gesture.

It is important to point out that the delays measured using this test method are not only caused by the segmentation method itself, but also by inaccuracy in the estimation of the ground truth, delays in data acquisition and processing delays.

An analysis of the influence of the window size on the segmentation accuracy is also proposed. For this particular purpose, the segmentation accuracy for a certain sample i is calculated from:

$$SA_i = \frac{\max(\text{RSE}) - \text{RSE}}{\max(\text{RSE}) - \text{ESE}} \cdot 100 \tag{4.20}$$

where RSE stands for the number of reported start events and ESE is the number of expected start events.

Another performance indicator chosen was the segmentation level, SL. The segmentation level is the fraction between the number of reported start events (RSE) and the number of expected start events (ESE):

$$SL = \frac{\text{RSE}}{\text{ESE}} \cdot 100 \tag{4.21}$$

4.3.6 Results and Discussion

The motion segments of this sequence are dynamic gestures and ME. More complex gestures are expected to be correctly segmented as well since they should hit a wider

Table 4.1: Performance parameters for the samples of Sequence 1.

Gesture	1	2	3	4	5	6	7	8	Average
$\overline{\Delta s}$ (ms)	181.8 ± 40.5	269.7 ± 50.5	221.2 ± 37.3	266.7 ± 49.4	260.6 ± 41.7	254.5 ± 94.6	209.1 ± 53.9	172.7 ± 71.2	229.5 ± 66.4
$\overline{\Delta e}$ (ms)	300.0 ± 55.8	248.5 ± 98.2	266.7 ± 93.1	306.1 ± 59.3	369.7 ± 60.5	209.1 ± 77.6	339.4 ± 51.2	324.2 ± 90.8	295.5 ± 87.3
S_{error}(%)	0.0	0.0	0.0	18.0	0.0	0.0	0.0	0.0	2.3
EL(%)	25.3	−1.8	10.0	2.7	14.8	−10.5	14.5	17.8	9.1

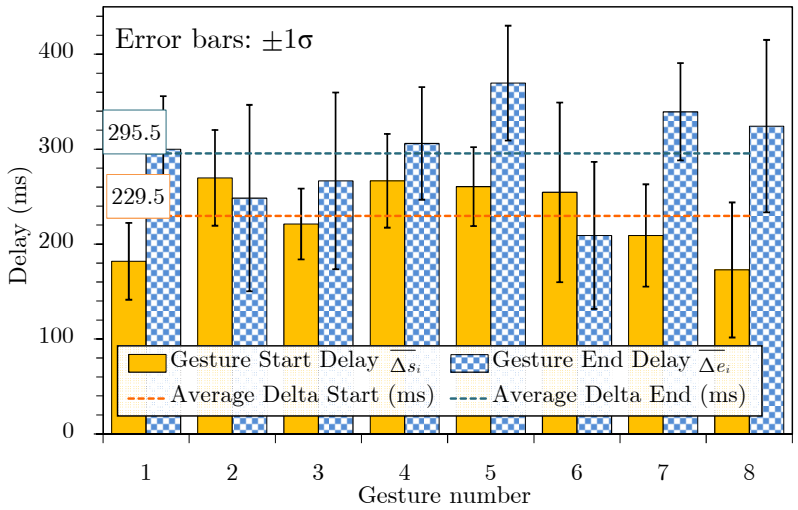


Figure 4.11: Gesture-wise average segmentation delay and the overall averages on samples of Sequence 1.

range of motion features.

The performance parameters on Sequence 1 are shown in Table 4.1. The average segmentation start delay was 230 ms while the end delay was 296 ms. The gesture end delay was higher, as expected, because of the sliding window. Nevertheless, the difference is 66 ms, considerably less than the size of the window (200 ms). The most likely reason for this is asymmetry between the features at the start and end of a gesture, for example, a slow start and a fast end. The total delay should not be problematic for the subsequent classification process, since most of the gesture data is within the output segment, as can be seen in Fig. 4.11.

On the top of Fig. 4.10, we show the evolution of the vector $\mathbf{t}$ along the time for a sample of Sequence 1. It shows the regions in which gestures may have occurred by colour mapping the features' values over time, normalized by the maximum values of the features. However, it is visually difficult or impossible to accurately define when motion starts and ends. In the middle, the figure shows the plotted motion features over time during a sampling session. The segmentation output results are superimposed onto the ground truth as seen by the user. The bottom plot shows the amount of features above the threshold at each instant. The ruling feature for most gestures is feature 1, which is the wrist's speed. This is due to the fact that most gestures in this sequence have wrist movement.

The figure also shows the importance of taking into account the inversions of movement direction when performing gesture segmentation. Gestures 2, 4, 7 and

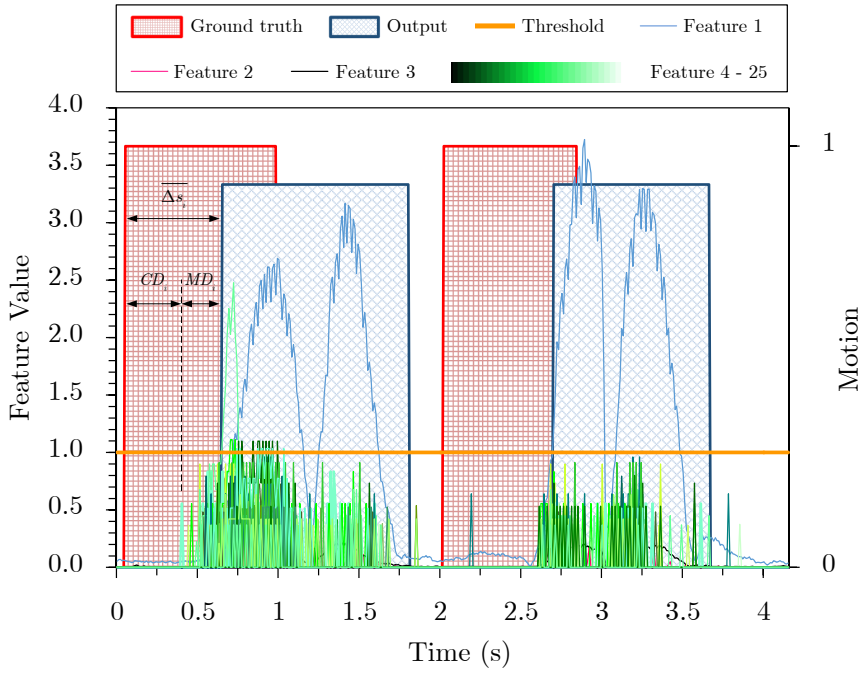


Figure 4.12: Zooming in on the transition from gesture 7 to gesture 8: normalized features and unit threshold (orange straight line).

8 – gestures with clear inversions – show clear drops in hand speed in the course of the gesture. However, in some scenarios this segmentation problem still persists, most likely because the speed and acceleration are not calculated accurately enough. Nevertheless, false negatives were detected only in gesture 4 due to a long stop during the inversion of movement.

Fig. 4.12 zooms in on a transition scenario from gesture 7 to gesture 8. The segmentation start delay $\overline{\Delta s}$ has two main components: (1) the feature capture/processing delay CD, and (2) the method's delay MD. For this study, MD is the most relevant, since CD changes according to the code efficiency and sensor latency. The total delay can be seen in Table 4.1 and Fig. 4.12. The fraction of the delay corresponding to the method itself in our experiments is from 15 to 35. This originates from the time the features need to reach the threshold while the gesture is speeding up.

Different classification methods tolerate different segmentation delays for gesture start, gesture end and extend level [144]. The results obtained compare favorably with the best results in literature.

We performed the same tests on the samples of Sequence 2. One of its samples is shown in Fig. 4.13. On the top, the dark colours represent the frames in which each feature is above the threshold. The most important feature is also the wrist's speed. Most gestures in this sequence have significant wrist movement. The figure also shows the importance of inversions of movement direction when performing gesture segmentation of gestures 1, 3, 5 and 8.

Segmentation performance depends largely on the size of the sliding window.

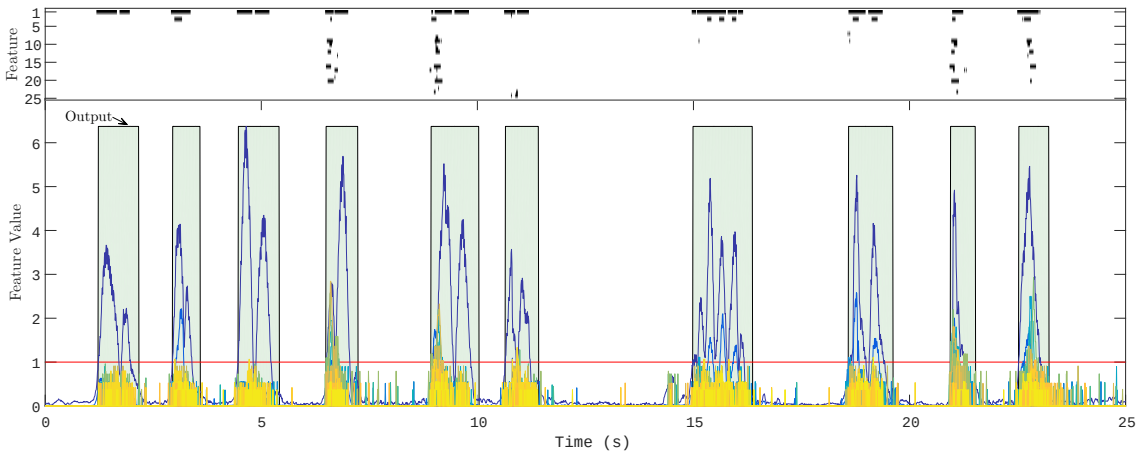


Figure 4.13: Data from one of the samples of Sequnce 2. On the top, the dark segments represent frames in which the feature is above the threshold. Below that, the features are normalized by the respective threshold and plotted over time. The shade represents the segmentation output

The result of the window size analysis on Sequence 1 is shown in Fig. 4.14. Initially, with small sliding windows, there is excessive segmentation, i.e., every peak of the features is considered a segment. This leads to low accuracy. Nevertheless, the accuracy quickly rises to 100% at a window size of 24 frames. The window size used in the previous tests was below this value, hence the over segmentation errors on gesture 4. The accuracy then plateaus at 100% until window sizes of 65 frames. It then drops, but this time because of under segmentation. Under segmentation occurs when a gesture is detected but its end is not. This causes consecutive gestures to overlap. The accuracy plateaus again at higher window sizes, at which point all sample gestures are overlapping into one single segment.

Fig. 4.14 demonstrates that the proposed method behaves as in [125]. The segmentation accuracy is improved initially as the size of the sliding window increases and degrades as the window size increases further.

In respect to Sequence 2, for a window size of 20, S_{error} was 0% for subject A and 11% for subject B. The result of the window size analysis is shown in Fig. 4.15. Small sliding windows cause excessive segmentation as well. Nevertheless, the segmentation level drops to 100% at a window size of 17 and 52 frames for subjects A and B, respectively. Its performance then plateaus at 100% until window sizes of 69 and 62 frames, respectively. It then drops because of under segmentation. The discrepancy between the results for each subject are justified by their familiarity with the gestures. Subject B didn't practice of the gesture sequence, which led to hesitation and pauses during the gestures, causing over-segmentation. If a larger window size were used, the number of errors would have been significantly less for subject B.

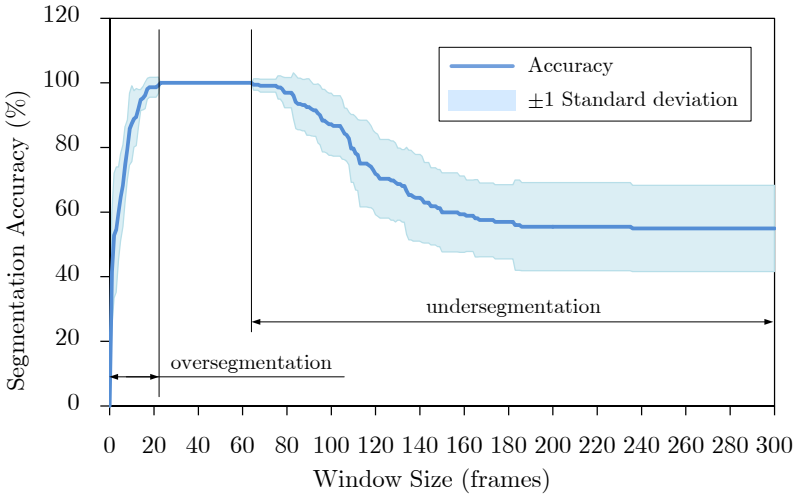


Figure 4.14: Segmentation accuracy variation in samples of Sequence 1 according to sliding window size.

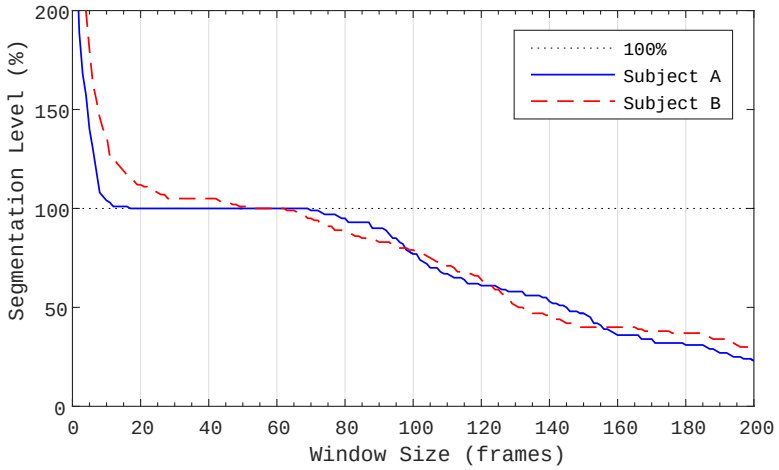


Figure 4.15: Segmentation level variation in samples of Sequence 2 with sliding window size.

4.4 Conclusion

This study presented a novel method for unsupervised continuous gesture segmentation by motion detection. It can be concluded that the proposed solution accurately divides a continuous stream of motion data in static and dynamic blocks. A quantitative analysis indicates an over-segmentation error of 2% for a non-optimal sliding window size of 20 frames. On a second sequence with mixed users, the over-segmentation error was 5.5%, in average. Segmented data present in most cases a shift-right and extended behavior when compared to the ground truth. The automatic optimization of the thresholds vector using a genetic algorithm demonstrated reliable behavior. Acceleration features were demonstrated to be necessary to achieve better segmentation for gestures with sudden inversions of movement direction. The proposed fine segmentation method reduces the noise in input data (any kind of sequential positional data) for subsequent classification.

Future work will be dedicated to testing the proposed solution with other interactions technologies: vision, IMUs and Electromyography (EMG). Performances will be compared. Additional efforts will be dedicated to improve the accuracy in computing velocity and acceleration features by filtering them. The resulting systems should also be tested by larger groups of people.

Chapter 5

Using Data Dimensionality Reduction for Recognition of Incomplete Dynamic Gestures

Miguel Simão[1], Pedro Neto[1] and Olivier Gibaru[2]

Based on the paper published on:
Pattern Recognition Letters

Abstract

Continuous gesture spotting is a major topic in human-robot interaction (HRI) research. Human gestures are captured by sensors that provide large amounts of data that can be redundant or incomplete, correlated or uncorrelated. Data dimensionality reduction (DDR) techniques allow to represent such data in a low-dimensional space, making the classification process more efficient. This study demonstrates that DDR can improve the classification accuracy and allows the classification of gesture patterns with incomplete data, i.e., with the initial 25%, 50% or 75% of data representing a given dynamic gesture (DG) - time series of positional and hand shape data. Re-sampling raw data with bicubic interpolation and principal component analysis (PCA) were used as DDR methods. The performance of different classifiers is compared in the classification 95 different signs of the UCI Australian Sign Language (High Quality) Dataset. Experimental tests on the Auslan dataset indicate that the use of PCA-based features result in a classification accuracy that is higher with 25% of gesture data (93% accuracy) than with 100% of gesture data (82% accuracy). The behaviour is similar on the UC2016 DG10 dataset where the DDR PCA features achieved a classification accuracy is higher with 50% of gesture data (100% accuracy) than with 100% of gesture data (96% accuracy). These results were obtained from a non-trained data set and the recognized gestures are used to control a robot in an collaborative process.

Keywords: Gesture Recognition, Data Dimensionality Reduction, Human-Robot Interaction

[1]Collaborative Robotics Laboratory, University of Coimbra, Portugal.
[2]Laboratoire d'Ingénierie des Systèmes Physiques et Numériques, ENSAM ParisTech, Lille, France.

5.1 Introduction

The ability to interact with a robot in a natural and intuitive way, for example using speech and gestures, has brought important advances to the way our societies look to robots. The paradigm for robot usage has changed in the last few years, from a concept in which robots work with complete autonomy to a scenario in which robots cognitively collaborate with human beings. This brings together the best of each partner, robot and human, by combining the coordination and cognitive capabilities of humans with the robots' accuracy and ability to perform monotonous tasks. For this end, robots and humans have to understand each other and interact in a natural way, creating a co-working partnership. This will allow a greater presence of robots in our companies, schools, hospitals, etc., with consequent positive impact on society's life standards. The current problem is that the existing interaction modalities are neither intuitive nor reliable. Instructing and programming an industrial robot by the traditional teaching method is a tedious and time-consuming task that requires technical expertise.

The robot market is growing and the human-robot interaction (HRI) interfaces will have a main role in the acceptance of robots as co-workers. Gestures and other natural interaction modalities may decrease the need for technical expertise in robot programming, therefore decreasing the cost of owning a robot.

Multimodal HRI interfaces combining gestures, speech and tactile based-actions are expected to be in a near future the standard for a reliable and intuitive interaction process. Nonverbal communication cues in the form of gestures are considered to be an effective way to approach natural HRI. For instance, a person can point to indicate a position to a robot, use a dynamic gesture to instruct a robot to move and a static gesture to stop the robot [14, 119]. In this scenario, the user has little or nothing to learn about the interface, focusing on the task and not on the interaction [120]. For all the reasons mentioned above, continuous and real-time gesture spotting (segmentation and recognition) are key factors to bridge the gap between laboratory research and real world application of novel HRI modalities [8]. A major challenge in continuous gesture recognition has to do with the fact that there are movement segments between gestures that have no meaning. Such inter-gesture transition periods are transition frames and are known as Movement Epenthesis (ME). Most studies treat ME as a classification problem [121].

Some gestures, although not all, can be defined by their spatial trajectory. This is particularly true for pantomimic gestures, which are often used to demonstrate a certain motion to be done, e.g., a circle. Burke and Lasenby focused with success on using PCA and Bayesian filtering to classify these time series [149]. The importance of DDR PCA to reduce the dimensionality of a data set representing human gestures for HRI has been studied in [150] [151]. In a different study, PCA is applied to a continuous stream of time series data capturing body motions with an accuracy of 91% [152]. In [153], it is proposed a unified sparse learning framework by introducing the sparsity or L1-norm learning, which further extends the locally linear embedding (LLE)-based methods to sparse cases. A DDR approach using sparsified singular value decomposition (SSVD) technique to identify and remove

trivial features before applying feature selection is proposed in [154]. A reference study presents a sequence kernel dimension reduction approach (S-KDR) in which spatial, temporal and periodic information is combined in a principled manner and an optimal manifold is learned [155]. A novel support vector number reduction method is in [156]. The support vector number is reduced by more than 99.5% without accuracy degradation.

A recent study indicates that after DDR PCA the classification accuracy is higher with incomplete gesture data than with complete gesture data [11]. These results were obtained in a relatively small library of 10 hand/arm dynamic gestures. Reasoning with incomplete data can be associated in some way to the concept of anticipation. For example we may identify a gesture before it is finished. An important study in the field represents each possible future using an anticipatory temporal conditional random field (ATCRF) that models the rich spatial-temporal relations through object affordances [157].

Gesture classification has been studied over the years. However, there remains the problem with reliability and intuitiveness, which are key factors for a system's acceptance by end-users. Other challenges are related with the ability to perform the recognition in real-time and continuously, recognize gesture patterns with incomplete data, and performing DDR keeping or increasing the recognition accuracy. DDR allows to reduce training data in supervised classification methods and consequently reduce the training time.

5.1.1 Overview and Proposed Approach

In a real-world system we usually have a device setup that captures features of what we are analyzing. In the case of dynamic gestures, time dependencies may also be important. As opposed to static gestures, of which we only need snapshots of the data at certain instants, dynamic gestures require that data are recorded over time, generating a multivariate time series. Oftentimes, the data are sampled at high rates, quickly generating high dimension data sets. Performing time series recognition is a currently active research topic.

Furthermore, there are other challenges in time series classification. Not only is there spatial variation between sequences, but there is also temporal variation, which is not necessarily linear.

In this work we introduce an approach to perform the recognition of time series segments, targeted at natural language processing (NLP) of hand gestures. This approach has the advantage of allowing accurate classification to be performed with partial (incomplete) gesture data. This makes it especially well suited for real-time applications, Fig. 5.1.

To overcome the problem of classification of time sequences, i.e., DGs, we present two approaches to their feature extraction and DDR: one based on up- or down-sampling of gesture frames using interpolation, and another based on PCA. The PCA approach has the advantage of being capable of yielding a good classification even before the gesture is finished – with incomplete/partial gesture data. Complex DGs are defined by a large set of features, with a variable number of frames, and

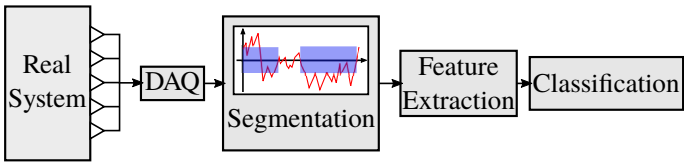


Figure 5.1: Overview of the proposed gesture recognition system.

with both trajectory and finger movement. The performance of different classifiers is compared in the process to classify the 95 different signs of the UCI Australian Sign Language (High Quality) data set.

Independently of the type of gesture and feature extraction approach, the predictor for a certain sample is represented by the vector $\mathbf{z} \in \mathbb{R}^d$:

$$\mathbf{z}^{(i)} = [f_1 \ f_2 \ ... \ f_d] \tag{5.1}$$

where f_i is the ith element of $\mathbf{z}$.

5.2 Data Dimensionality Reduction and Classification

5.2.1 Overview

Lets assume we have a data set $\mathcal{S}$ which contains a number of samples and corresponding labels. A specific sample $\mathcal{S}(i)$ of the data set is represented by the matrix $\mathbf{X}^{(i)}$ and its label is $\mathbf{t}^{(i)}$. A function f is then used to extract the feature-vector $\mathbf{z}$ from each sample:

$$\begin{array}{rll} f: & \mathbb{R}^{d \times n} & \rightarrow \mathbb{R}^b \\ & \mathbf{X} & \rightarrow \mathbf{z} \end{array} \tag{5.2}$$

Above, b is the dimensionality of the feature vector. This transformation can have several steps and DDR may be one of them. The vector $\mathbf{z}$ is the input for the classifiers and $\mathbf{t} \in \{0,1\}^{n_{classes}}$ is the target value of the classifier for that sample (supervised learning). If the target class has the number o, the target vector $\mathbf{t}^{(o)}$ is defined by:

$$\mathbf{t}_j^{(o)} = \delta_{oj}, \ j = 1, ..., n_{classes} \tag{5.3}$$

where δ is the Kronecker delta and $\mathbf{t}_j$ is the jth element of $\mathbf{t}$.

The transformation f may still yield a vector with very high dimensionality b, which may be detrimental for the classification. Therefore, intermediate processing techniques, such as PCA and interpolation, are proposed for DDR and are defined as in (5.4). Nevertheless, DDR can be done either before or after feature extraction.

$$r: \begin{array}{ll} \mathbb{R}^b & \rightarrow \mathbb{R}^{b'} \\ \mathbf{z} & \rightarrow \mathbf{z}' \end{array}, \qquad b' < b \tag{5.4}$$

5.2.2 Resampling with Bicubic Interpolation

Resampling is done with bicubic interpolation to transform a DG sample $\mathbf{X}^{(i)},\ i \in i^D,\ \mathbf{X} \in \mathbb{M}^{d \times n}$, which has a variable number of frames n, into a fixed-dimension sample $\mathbf{X}',\ \mathbf{X}' \in \mathbb{M}^{d \times k}$. In this work, we propose down-sampling the gesture data so that $k \leq n$, being k arbitrarily defined as the minimum n in all of the samples i so that $i \in i^D$. In this way, the proposed transformation is effectively down-sampling the gesture data, thus reducing the dimensionality of the feature vector.

$$interp: \begin{array}{ll} \mathbb{R}^{d \times n} & \rightarrow \mathbb{R}^{d \times k} \\ \mathbf{X} & \rightarrow \mathbf{X}' \end{array} \tag{5.5}$$

The bicubic interpolation method [158] yields a surface p described by 3rd order polynomials in both dimensions of space. Given a patch of dimension 2×2, there are 4 data points in which we know the values f and derivatives f_x, f_y and f_{xy}. The derivatives are not known at the boundaries, but they can be estimated using finite differences. The interpolated values inside the uniformized 2×2 sector are given by:

$$p(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \tag{5.6}$$

A representation of the sector is in Fig. 5.2.

The problem is determining the 16 coefficients a_{ij}. The function values and 3 derivatives at the 4 points provide $4 \times 4 = 16$ linear equations, which can be written as an equation system $\mathbf{A}\alpha = \mathbf{x}$ with:

$$\alpha = [a_{00}\ a_{10}\ a_{20}\ a_{30}\ a_{01}\ ...\ a_{33}]^T \tag{5.7}$$

$$\mathbf{x} = [f(0,0)\ f(1,0)\ ...\ f_x(0,0)\ ...\ f_y(0,0)\ ... \\ f_{xy}(0,0)\ ...\ f_{xy}(1,1)]^T \tag{5.8}$$

The matrix $\mathbf{A}$ is nonsingular, so the equation system can be rewritten as $\alpha = \mathbf{A}^{-1}\mathbf{x}$. This process is used for all patches in the bi-dimensional grid. The derivatives at the boundaries of a patch are maintained across neighbouring patches. In order to apply the method to the whole data grid efficiently, techniques such as Lagrange polynomials, cubic splines or cubic convolution algorithms are used. The resulting interpolated data points are smother and have less artifacts than those using other interpolation methods, such as bilinear interpolation.

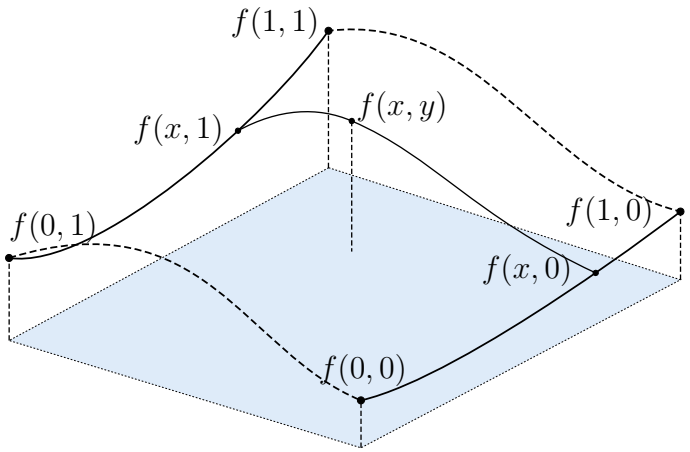


Figure 5.2: Representation of the result of bicubic interpolation on a 2×2 grid of points $f(0,0)$, $f(1,0)$, $f(0,1)$, $f(1,1)$.

5.2.3 Principal Component Analysis

PCA is a mathematical tool that performs an orthogonal linear transformation of a set of n p-dimensional observations, $\mathbf{X} \in \mathbb{R}^{n \times p}$, into a space defined by the principal components (PC). The PC have necessarily a size less than or equal to the number of original dimensions, p. The first component has the largest possible variance observed in the observations. Each of the following PC is orthogonal to the preceding component and has the highest variance possible under this orthogonality constraint. The PC are the eigenvectors of the covariance matrix and its eigenvalues are a measure of the variance in each of the PC. Therefore, PCA can be used for reducing the dimensionality of data by projecting that data into the PC space and truncating the lowest-ranked dimensions. These dimensions have the lowest eigenvalues, so truncating them retains most of the variance present in the data.

The first step in PCA is centering the data, because PCA is sensitive relative to the scaling of the original dimensional space. This is done by subtracting each of a dimension's values by its overall average.

The PC transformation is very often determined by another matrix factorization method, the SVD of $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{5.9}$$

where $\mathbf{X} \in \mathbb{M}^{n \times p}$ is the original data matrix. $\mathbf{\Sigma} \in \mathbb{M}^{n \times p}$ is a diagonal matrix with the singular values of $\mathbf{X}$, $\mathbf{U}$ is a $n \times n$ matrix whose columns are orthogonal unit vectors that are the left singular vectors of $\mathbf{X}$, and $\mathbf{V} \in \mathbb{M}^{p \times p}$ is a matrix whose columns are unit vectors, the right singular vectors. Both $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, so that $\mathbf{U}^T\mathbf{U} = \mathbf{I}_p$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}_p$. The singular values $\sigma_1, \sigma_2, ...$ in the diagonal of $\mathbf{\Sigma}$ are the positive square roots, $\sigma_i = \sqrt{\lambda_i} > 0$, of the nonzero eigenvalues of the Gram matrix $\mathbf{K} = \mathbf{X}^T\mathbf{X}$, thus being always positive.

The implementation used for this purpose was Matlab's *pca* function. Since the input data matrix $\mathbf{X}$ is most often rectangular, the function uses the aforementioned

SVD method (5.9) for the matrix decomposition. The singular values, i.e., the variance in each of the PC, are the eigenvalues of the covariance matrix of $\mathbf{X}$. The covariance matrix of p sets variates $\{\mathbf{x}_1\}$, ..., $\{\mathbf{x}_p\}$, $\mathbf{x}_i = \mathbf{X}_{\bullet i}$ is defined by $\mathbf{W} \in \mathbb{M}^{p \times p}$:

$$\mathbf{W}_{ij} = \text{cov}(\mathbf{x}_i, \mathbf{x}_j) \equiv \langle (\mathbf{x}_i - \mu_i)(\mathbf{x}_j - \mu_j) \rangle, i, j = 1, ..., p \tag{5.10}$$

where μ and $\langle\ \rangle$ denote mean value, being $\mu_i = \langle \mathbf{x}_i \rangle$. $\mathbf{W}$ can also be written as $\mathbf{W}_{ij} \equiv \frac{1}{n-1}\mathbf{X}\mathbf{X}^T$. The product $\mathbf{X}\mathbf{X}^T$ has as eigenvectors the columns of $\mathbf{U}$.

Although PCA is most often performed to reduce the dimensionality of the observations, in this work we preferred to use the PCs as features. The first PC or singular vector $\mathbf{U}_{\bullet 1}$ determines the direction in the PC-space in which there is the most variance during a DG. The variance is measured by the respective singular value, $\mathbf{\Sigma}_{11}$. Therefore, we expect these values to produce good features for the DG classification, even if the gesture is incomplete. We also used PCA to represent gesture features in lower two- and three-dimensional spaces, for easier visualization.

5.2.4 Classifiers

In this study we experimented with three distinct classification methods: DTW, SVM and ANN. DTW is a method often used for classification in NLP-based time series. It is an algorithm that measures similarity between two temporal sequences that may vary in speed. This similarity can be interpreted as the minimum cumulative distance between the two non-linearly warped sequences. The DTW distance can then be used as a metric for a k-NN classifier. In this case, the DTW distance is computed between a test sample and all the training samples. The final classification is the mode of the k closest training sample classes. An SVM is a supervised classification method that finds the hyperplane(s), in the training data set, that optimally separates the data set classes. Their performance is typically good in multi-class problems, but it depends greatly on the structure of the data. To solve this issue, the user must select an appropriate data transformation kernel, while an ANN is an universal approximator that can adapt to any type of data.

5.3 Experimental Results

5.3.1 UCI Auslan Data Set

The proposed approach was tested with the Australian Sign Language (Auslan) signs (High Quality) data set, from the UCI machine learning repository [114]. Each sample of this set is a multidimensional time series. The whole data set was used, with 95 distinct classes and 27 examples for each one of the classes. The samples were obtained from one native Auslan signer over a period of 9 weeks.

The data acquisition setup consisted of two Fifth Dimension Technologies (5DT) gloves and two Ascension Flock-of-Birds magnetic position trackers. Each of the subjects two hands was equipped with a glove and a tracker. The data gloves

measured finger flexion for each of the 5 fingers and the trackers recorded positional and orientation of the hands – 6 degrees of freedom (DOF). These 22 degrees of freedom were measured over time at a rate of close to 100 frames per second. The average length of each sign is about 60 frames. Each frame is represented as a 15 dimensional feature vector consisting of hand position (X, Y,Z), roll, yaw, pitch, bend measurements of different fingers.

In this work, a sample in the data set is represented by $\mathcal{S}$:

$$\mathcal{S} = \left\{\mathbf{X}^{(i)},\ t^{(i)}\right\},\ \mathbf{X}^{(i)} \in \mathbb{R}^{d \times n},\ t^{(i)} \in \{1, ..., L\} \tag{5.11}$$

where d is the number of DOF of the system, n is the number of data frames in the sample, t is the target class number and L is the number of classes of the set.

5.3.2 UC2016 DG10 Data Set

A data set of dynamic gestrues was built with two sensors were used to capture the hand shape, position and orientation: (1) a data glove; (2) a magnetic tracker.

The data glove was a CyberGlove II, developed by CyberGlove Systems. The version used has 22 resistive bend sensors: three flexion sensors per finger, four abduction sensors, a palm-arch sensor, and two sensors to measure wrist flexion and abduction. The sensors output real-time digital joint-angle filtered data at an average rate of 100 Hz.

The magnetic tracker used was a Polhemus Liberty. It has a very low latency and outputs at a rate of 120 Hz.

The glove transmits the data to through a Bluetooth connection and the tracker is connected by a physical serial connection. Serial objects read the available data in windows of 10 samples and store the available data with timestamps in buffers. A script then reads the k newest samples from both the buffers and processes them. A full frame of data with the 22 DOF from the glove and 6 DOF from the tracker is represented by $\mathbf{f}$ (5.13), where g_k represents the kth DOF of the glove and l_k represents the kth DOF of the tracker. Before further processing, the stream of data is segmented by a motion-threshold method [8]. The method appends to the frame a binary segmentation variable m which represents whether the frame belongs to a dynamic segment or not.

$$\mathbf{f} = [g_1\ g_2\ ...\ g_{22}\ l_1\ l_2\ ...\ l_6\ m] \tag{5.12}$$

A sample in this data set is represented by ($\mathcal{S}^D$):

$$\mathcal{S}^D = \left\{\mathbf{X}^{(\mathbf{i})}, t^{(i)}\right\},\ \mathbf{X}^{(i)} \in \mathbb{R}^{d \times n},\ t^{(i)} \in \left\{1, ..., L^D\right\} \tag{5.13}$$

where d is the number of DOF of the system, n is the number of data frames in the sample, t is the target class and L^D is the number of classes for DG.

A total of 10 DGs combining hand/arm and fingers motion were selected, Fig. 5.3. For training and validation purposes, a total of 400 samples were obtained from two subjects.

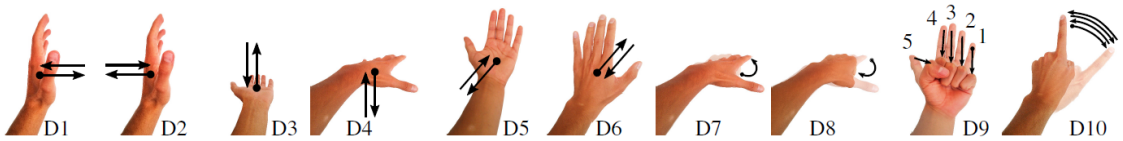


Figure 5.3: Representations of the 10 DGs of the UC2016 data set.

5.3.3 Feature Extraction

Two distinct feature extraction methods are proposed:

1. Re-sampling the samples with bicubic interpolation (FE1);
2. Extracting principal vectors and values using PCA (FE2).

In the first case, FE1, given a sample $\mathbf{X}^{(i)} : i \in i^D$ with n frames ($\mathbf{X}^{(i)} \in \mathbb{M}^{22\times n}$), the goal is to resample it to a fixed size p. The number p can be chosen arbitrarily, but in order to reduce the number of features, p should be below a lower bound such that $p \leq n,\ \forall n|\mathbf{X^{(i)}} \in \mathbb{M}^{22\times n}$. For this data set, this lower bound is 41 and we chose a $p = 20$, which is about half of the minimum original gesture length. Applying the bicubic interpolation algorithm results in a matrix $\mathbf{X'} \in \mathbb{R}^{22\times p}$. The following step is to transform $\mathbf{X'}$ into a vector $\mathbf{z} \in \mathbb{R}^{22p\times 1}$, which is done by concatenating every frame vertically:

$$\mathbf{z}^{(i)} = \begin{pmatrix} \mathbf{Z}^{(i)}_{\bullet 1} \\ \vdots \\ \mathbf{Z}^{(i)}_{\bullet (28p)} \end{pmatrix} \tag{5.14}$$

In a second case, FE2, we use PCA to extract features. The advantage is that it allows us to obtain features from incomplete gestures and still obtain coherent features. From each sample $\mathbf{X}^{(i)} : i \in i^D$, $\mathbf{X} \in \mathbb{M}^{22\times n}$ we can extract b feature vectors $\mathbf{z}^{(i)}_k : \ k \in]0, 1]$, where k defines the fraction of the number of frames that were used:

$$\mathbf{z}^{(i)}_k = \mathbf{U}_{\bullet 1} \tag{5.15}$$

where $\mathbf{U}_{\bullet 1}$ is the first singular vector. The singular vector has the same dimensionality as the data source. It is calculated using the the partial sample $\mathbf{X}^{(i)}_{\bullet \mathbf{m}}$, so that:

$$\text{pca}(\mathbf{X}^{(i)}_{\bullet \mathbf{m}}),\ \mathbf{m} = \{1, ..., \lceil nk \rceil\} \tag{5.16}$$

where n is the number of frames of the sample and $\lceil nk \rceil$ represents the ceiling function, since $\lceil nk \rceil \in \mathbb{N}$. Therefore, $\lceil nk \rceil$ represents the cutoff frame, that is, the last frame within the sample that is used for feature extraction.

The last feature processing step is feature scaling. Feature scaling is essential for achieving smaller training times and often better classification performance with less

Table 5.1: Classification accuracy of the ANN and DTW models of the features extracted from full gestures, on the Auslan data set.

	DTW	FE1-ANN	FE2-ANN
Accuracy (%)	77.02	86.67	85.72

training. It harmonizes the values of different features so that all of them fall within the same range. This is especially important when some features have distinct orders of magnitude. The scaling function chosen was linear rescaling, l:

$$l(\mathbf{x}) = \frac{2\mathbf{x} - \widehat{\mathbf{X}}^T}{\widehat{\mathbf{X}}^T} \tag{5.17}$$

where $\widehat{\ }$ is the max+min operator defined in (5.18). $\mathbf{X}^T = \left(\cup \mathbf{z}^{(i)} : \ i \in i^T\right)$ is the set of unscaled features of the training set. This operator is valid both for static and dynamic gestures but the sample subsets used should be exclusive.

$$\widehat{\mathbf{X}}_i = \max \mathbf{X}_{i\bullet} + \min \mathbf{X}_{i\bullet}, \quad i = 1, ..., d \tag{5.18}$$

5.3.4 Results and Discussion

Auslan Data Set

The available samples $\mathcal{S}(i) : \ i \in \mathbf{i}^D$ were divided in two sets of approximately the same size: a training set ($i \in \mathbf{i}^{DT}$) and a validation set ($i \in \mathbf{i}^{DV}$). Each set has about the same number of samples per class. In this case, we use all the available samples in the data set, 27 samples per each of the 95 classes (2565 samples total). All the accuracy results presented in this study correspond to those obtained from the validation set.

We present the results for the DTW approach, FE1 and FE2 (full gesture) in Table 5.1. The DTW classifier achieved a 77.02% accuracy rate, while the proposed feature sets FE1 and FE2 achieved significantly better results (86.67% and 85.72%, respectively) when discriminated with an ANN. A representation of the features for the first 10 classes for FE2 is shown in Fig. 5.4. In this $\mathbb{R}^2$ space, the classes show good separability, with low intra-class dispersion.

The ANN architecture is composed by one hidden layer with 100 nodes and 95 output neurons (classes) in both approaches, Fig. 5.5. The sole difference between FE1 and FE2 is the size of the input feature vector, 440 and 23 for FE1 (5.14) and FE2 (5.15), respectively. In both cases the transfer function is the hyperbolic tangent in the first layer and the *softmax* function in the second layer.

While for the FE1 case the gesture frames are interpolated and the features are extracted after the gesture is finished, for FE2 the features can be extracted at any time during a gesture. This allows for recognition even before the gesture is even finished. If we plot the evolution of the features of a gesture sample over time, we obtain what is shown in Fig. 5.6. Over time, the features still form

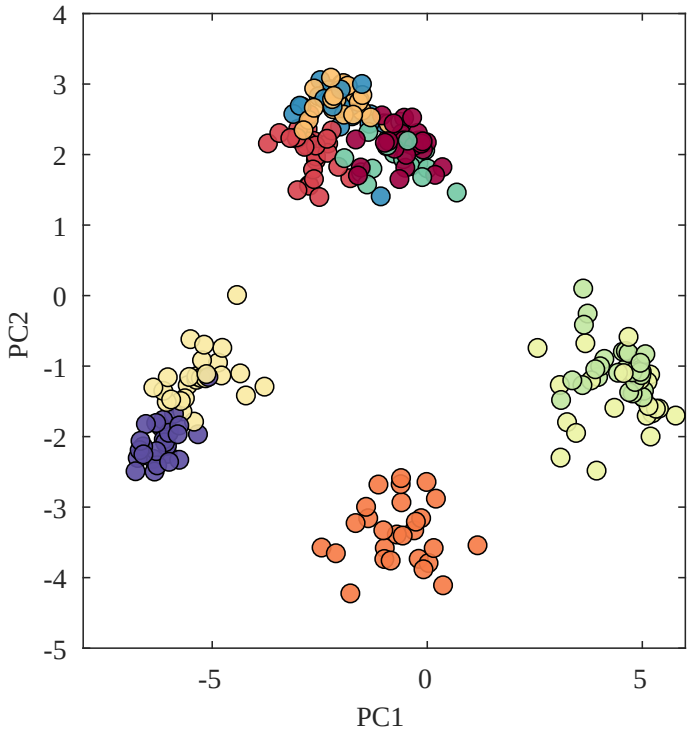


Figure 5.4: Distribution of features (FE2 with 100% of data) in a reduced principal component space. Only the first 10 classes of the Auslan data set are represented, with colours discriminating different classes.

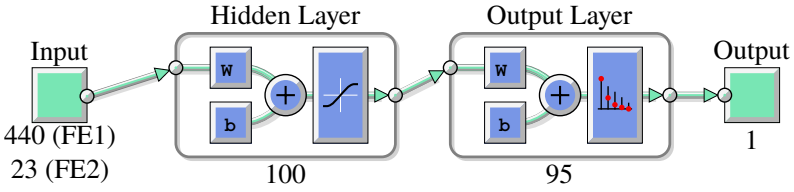


Figure 5.5: ANN architecture used for classification for test cases FE1 and FE2, on the Auslan data set.

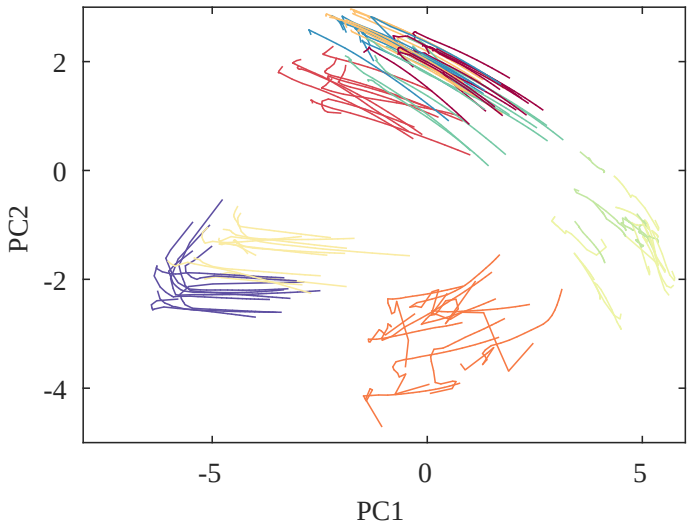


Figure 5.6: Representation of the evolution of FE2 features over gesture completion rate, on the Auslan data set. Each line represents one sample and its colour refers to the target class.

Table 5.2: Classification accuracy on the validation set of the Auslan data set, considering test case FE2.

Classifier Model	Accuracy over time (%)				
	25%	50%	75%	100%	Mean
ANN	93.06	90.88	88.31	81.92	88.08
SVM	83.79	81.29	78.18	73.89	80.01

defined clusters, which are precursors to good classifiers. Given that, the ANN for FE2 was trained and validated with 25 sets of features originated from $\mathbf{i}^{DT}$, $\mathbf{z}_k^{(i)} : k \in \{1/25,2/25,3/25,...,25/25\}$, see (5.15).

For FE2, the accuracy results are displayed in Table 5.2. In this table we are displaying the accuracy using 25, 50, 75 and 100% of the gesture data. The best accuracy, reaching 93.06%, was obtained when only about 25% of the initial gesture data was used. When more data was used, the accuracy decreased to 81.92%, Fig. 5.8. It is also possible to see the evolution of the FE2 features obtained from 25, 50, 75 and 100% of the data in Fig. 5.7. Even at 25% there is already good separation of the classes and the clusters are maintained over time.

The UCI Auslan data set has been applied in a number of studies related with pattern classification. In a recent study the authors randomly selected four subsets of the whole data set with each subset containing 20 categories [159]. It is reported that the best classification results were obtained with the proposed order preserving sparse coding method (MTO-SC), with an accuracy of about 94% in the classification of subsets with 20 categories. In [160] it is reported an accuracy of about 94% using SVM and logistic regression models. A Dual Square-Root Function (DSRF)

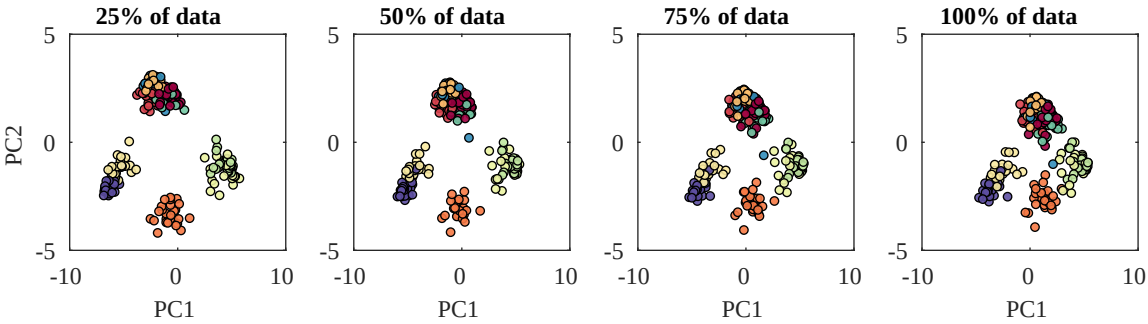


Figure 5.7: Plots of the features obtained from Auslan's validation set (including the sets with incomplete data – 25%, 50%, 75% and 100% of the data) in a reduced principal component space. Colours discriminate different classes.

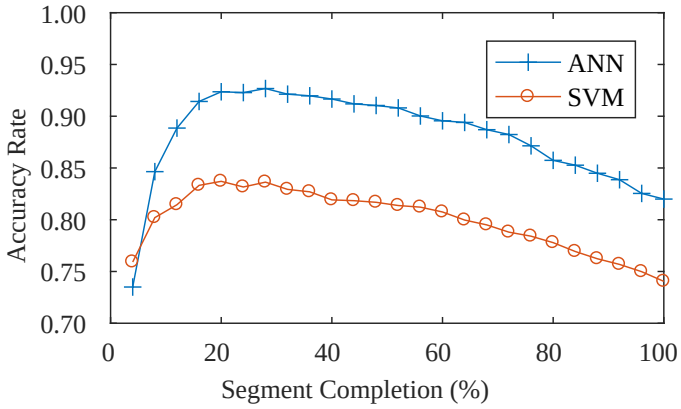


Figure 5.8: Evolution of gesture prediction accuracy with percentage of data used, on the Auslan data set.

descriptor obtained by calculating gradient-based shape features of normalized rigid body motion trajectories was applied with an accuracy of 88%, [161].

UC2016 DG10 Data Set

In this data set, the available samples $\mathcal{S}(i) : \ i \in \mathbf{i}^{D}$ were too randomly divided in two sets – a training set ($i \in \mathbf{i}^{DT}$), and a validation set ($i \in \mathbf{i}^{DV}$). In this case, there are 40 samples per each of the 10 classes, across two subjects. Both the training and validation sets have 20 samples/class.

The ANN architecture is composed by one hidden layer with 25 nodes and 10 output neurons (classes) in both approaches, as shown in Fig. 5.9. The difference between FE1 and FE2 is the size of the input feature vector, 4508 and 29 for FE1 (5.14) and FE2 (5.15), respectively. In both cases the transfer function is the hyperbolic tangent in the first layer and the *softmax* function in the second layer.

The classification results are displayed in table 5.3. For FE1, the accuracy in the validation data set was 99.0% (99/100) for subject A, where the only error was gesture 7 being classified as 8. This can be seen in the confusion table in Fig. 5.11(a).

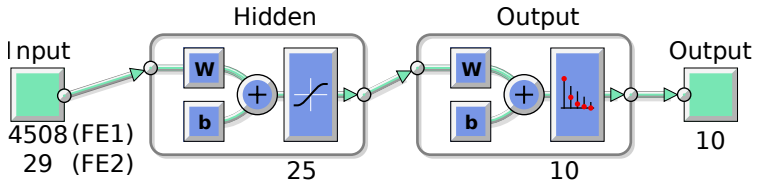


Figure 5.9: ANN architecture for the classification of DG10 gestures using interpolated frames (FE1) and the first principal component (FE2) as features.

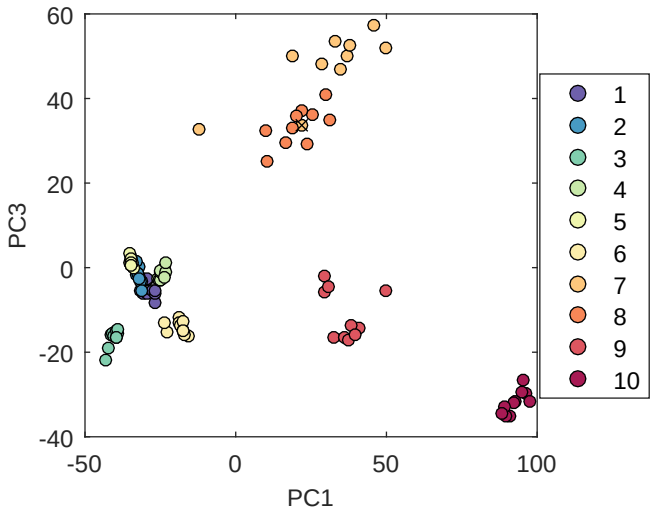
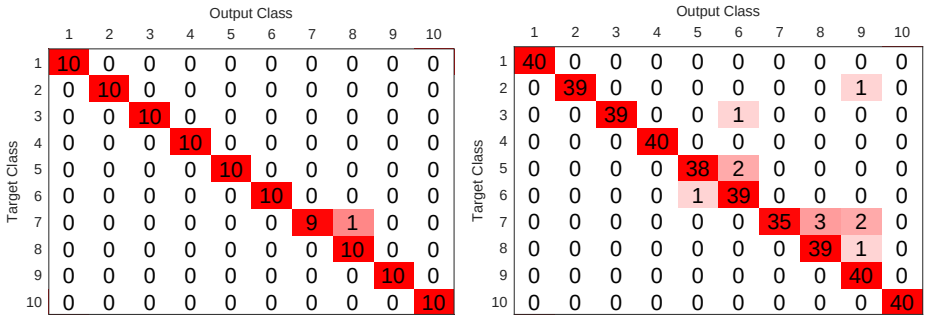


Figure 5.10: Distribution of DG10's validation samples $\mathbf{i}^{DV}$ for test case FE1. Each colour discriminates a class and the wrongly classified samples are marked with an $\times$.

The accuracy for subject B was lower, at just 90%. In Fig. 5.10 it is possible to see the distribution of the features in a reduced principal component space (first and third PCs) for FE1. In this $\mathbb{R}^2$ space, the classes show good separability, being gesture 7 the exception, where higher than normal dispersion can be seen. One of the samples representatives of class 7 fell into the middle of the class 8 cluster, which led to it being miss-classified. There are other very close clusters, such as the gestures D1, D2, D4 and D5, but they have very low dispersion and show better separability when seen in higher spaces.

While for the FE1 case the gesture frames are interpolated after the gesture is finished, for FE2 the features can be extracted at any time during a gesture. This allows for recognition even before the gesture is even finished. Given that, the network FE2 was trained and validated with four sets of features originated from $\mathbf{i}^{DT}$, $\mathbf{z}_k^{(i)} : k \in \{0.25, 0.50, 0.75, 1\}$, see (5.15). Therefore, the training and validation sets had each 40 feature vectors per gesture. The network has the same configuration as in FE1 except for the number of input nodes which is 29, Fig. 5.9.

The accuracy results in the validation data set given this configuration are shown in Fig. 5.3 for each of the 4 sets of features. With only the initial 25% of the gesture,



(a) (b)

Figure 5.11: Confusion table for the classification of the DG10 (subject A) samples from the validation set $\mathbf{i}^{DV}$ using both approaches: (a) FE1, (b) FE2.

Table 5.3: Final results for the classification accuracy on DG10's validation split for FE1 and the various time steps of FE2.

		Accuracy over time (%)				
Subject	FE1	25%	50%	75%	100%	Mean
A	99.00	85.00	96.00	98.00	99.00	94.50
B	90.00	72.00	90.00	97.00	88.00	86.75
Mean	94.50	78.50	93.00	97.50	93.50	90.63

the accuracy was 85% for subject A and increased to 96% when half of the gesture data was used. The accuracy further increased to 98% and 99% when 75% and 100% of the data were used, respectively. Subject B had lower performance overall, with a maximum accuracy of 97% at 75% of gesture completion. However, the accuracy decreased to 88% when all of the data were used, following a behaviour similar to that seen in the Auslan data set. The confusion table in Fig. 5.11(b) shows that the gesture with the most errors was gesture 7, being classified three times as 8 and two times as 9. All these gestures are short in duration (less data) and are similar, e.g., the starting and ending poses of 7 and 9 are the same (closed hand), the difference being in the order the fingers are closed.

It is possible to see the evolution of the features obtained from 25% to 100% of the data in Fig. 5.12. Even at 25% there is already good separation of the classes, although the dispersion is still high, compared to the later stages. As more data is available, the dispersion decreases and the feature samples form defined intra-class clusters. In the case of gestures D6, D7 and D8 this also happens, but the clusters partially intercept, causing the decrease of the accuracy rate. This also means that the selected features are not ideal for the classification of these three gestures.

In both cases FE1 and FE2 the features for the classes D1 through D6 have well defined albeit close clusters, sometimes generating errors. This happens because there is very little inter-gesture variation of most of the variables during these gestures. They are distinguished by path and hand orientation, which are defined by 6

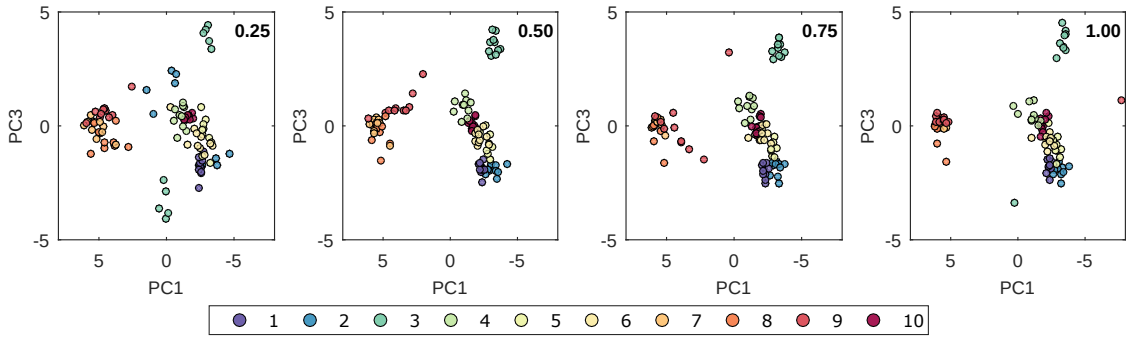


Figure 5.12: Plots of the features obtained from DG10's validation split (including the sets with incomplete data – 25%, 50%, 75% and 100% of the data) in a reduced principal component space. Colours discriminate different classes.

out of the 28 DOFs of the system. The classification accuracy remained high for these samples, but in the future these gestures should be represented by better features, e.g, trajectory direction, in order to improve the robustness of the classification.

5.3.5 Human-Robot Interaction

The proposed classification system was tested in a real robot, controlling it using gestures in a collaborative task: preparing a breakfast meal. In such task the robot grabs a cereal box (pouring the contents into a bowl) and grabs a yogurt bottle (also pouring its contents into the same bowl), Fig. 5.13. Our setup is composed by a robot with 6 DOF, a data glove and a magnetic tracker. We are only recognizing 10 gestures from one hand/arm. The classified gestures are used as input to teleoperate the robot. This means that the recognized gestures were directly associated to robot commands like stop, move along X, Y or Z, rotate the robot end-effector in turn of X, Y or Z, open/close the gripper, and select modes of operation.

5.4 Conclusions

This study demonstrated that dynamic gesture data with variable length can be classified accurately with features obtained by DDR methods, while also making the classification process more efficient: increased accuracy, reduced training time and smaller models. The first principal component of the gesture data was shown to be a predictor nearly as effective as the full gesture data resampled with bicubic interpolation, in all data sets. The PCA-based features were also shown to perform better in predictive classification with lower amounts of data. We concluded that, on the Auslan data set (95 gesture classes), the classification accuracy is higher with 25% of initial gesture data (93% accuracy) than with 50% (91% accuracy), 75% (88% accuracy) or 100% (83% accuracy) of gesture data. The same behaviour was shown on the UC2016 DG10 data set with an accuracy higher with 75% of gesture data (97.5% accuracy) than with 100% of gesture data (93.5% accuracy). These

Figure 5.13: Visualization of different stages of robot teleoperation process: (a) starting point, (b) virtual joystick guidance to a goal, (c) forceful stop command, (d) rotation of the end-effector, (e) gesture-command to open the gripper, (f) grabbing a bottle and puling it up, (g) rotation of the end-effector, (h) safe collaboration with the robot. NOTE: The virtual joystick mode moves the end-effector in a direction defined by the vector that joins a center position in which the hand is closed and the position of the hand when it is moved.

results show that the initial part of the gesture discriminates it better than the final part. The recognized gestures proved to be a natural and intuitive human-robot interface.

Future work will be dedicated to explore the ability to classify gesture patterns with initial 25% in the context of anticipation activities in the process of human-robot interaction. In addition, this approach will be tested with other type of input data (obtained from other sensing technology).

Chapter 6

Online Recognition of Incomplete Gesture Data to Interface Collaborative Robots

Miguel Simão[1], Pedro Neto[1] and Olivier Gibaru[2]

Based on the paper submitted to:
IEEE Transactions on Industrial Electronics

Abstract

Online recognition of gestures is critical for intuitive human-robot interaction (HRI) and further push collaborative robotics, making robots accessible to everyone. The problem is that it is difficult to achieve accurate gesture recognition in real unstructured environments, many times from distorted and incomplete multi-sensory data. This chapter introduces a HRI framework to classify large vocabularies of interwoven static gestures (SGs) and dynamic gestures (DGs) from wearable sensors. Features representing DGs are obtained by applying data dimensionality reduction (DDR) to raw data from sensors (resampling with cubic interpolation and principal component analysis). Experimental tests were conducted using the UC2017 hand gesture dataset with samples from eight different subjects. The classification models show an accuracy of 95.6% for a library of 24 SGs using a random forest (RF) and 99.3% for 10 DGs using artificial neural networks (ANNs). These results compare equally or favourably with different commonly used classifiers. Long Short-Term Memory (LSTM) deep networks achieved similar performance in online frame-by-frame classification using raw incomplete data, performing better than static models with specially crafted features in accuracy and worse in training and inference time. The recognized gestures are used to teleoperate a robot in a collaborative process that consists in preparing a breakfast meal.

Keywords: Human-Robot Interaction, Collaborative Robotics, Online Gesture Recognition, Neural Networks

[1]Collaborative Robotics Laboratory, University of Coimbra, Portugal.
[2]Laboratoire d'Ingénierie des Systèmes Physiques et Numériques, ENSAM ParisTech, Lille, France.

6.1 Introduction

The paradigm for robot usage has changed in the last few years, from an idea in which robots work with complete autonomy to a scenario in where robots cognitively collaborate with human beings. This brings together the best of each partner, robot and human, by combining the coordination and cognitive capabilities of humans with the robots' accuracy and ability to perform monotonous tasks. Robots and humans have to understand each other and interact in a natural way (using gestures, speech and physical interaction), creating a co-working partnership. This will allow a greater presence of robots in all domains of our society. The problem is that the existing interaction modalities are neither intuitive nor reliable. Instructing and programming an industrial robot by the traditional teaching method is a tedious and time-consuming task that requires technical expertise in robot programming.

The collaborative robotics market is growing fast and the HRI interfaces have a main role in the acceptance of robots as co-workers. Gestures are an intuitive way to teleoperate a robot (technical skills in robot programming are not required) [3]. For instance, a human co-worker can use a DG to indicate a grasping position and use a SG to stop the robot [4]. In this scenario, the human has little or nothing to learn about the interface, focusing on the task being performed. The robot assists the human when required reducing the exposition to poor ergonomic conditions and possible injuries.

This work proposes an integrated modular gesture-based HRI framework, Fig. 6.1. Static and dynamic gesture segments, composed of data from a data glove and a magnetic tracker, are obtained automatically applying a sliding-window motion detection method. Static segments are used as inputs for SG classifiers. Dynamic segments, which are the inputs for DG classifiers, are subject to data dimensionality reduction (DDR) by resampling gesture segments using cubic interpolation (CI) or principal component analysis (PCA). Traditional probabilistic latent variable models such as PCA are static linear approaches in which the dynamics and nonlinearities are not properly considered. In [162] it is proposed a weighted linear dynamic system (WLDS) for nonlinear dynamic feature extraction that demonstrated superior performance for prediction accuracy. Nevertheless, the real-time performance degrades compared to static approaches (consumes more computational time), a limitation for the proposed gesture-based HRI system that requires online performance. The proposed CI and PCA approaches demonstrated to be computationally inexpensive and presented a relatively high classification accuracy, even from incomplete data. We use large vocabularies of gestures (total of 34) and a low number of training samples to simplify and expedite the training process. Experiments demonstrated that standard classifiers such as artificial neural networks (ANNs) are reliable to classify both SGs and DGs, comparing equally/favorably with deep learning classifiers (LSTM and Convolutional Neural Networks (CNNs)) not only in terms of inference time but also accuracy. Finally, a robot task manager correlates the classified gestures with robot commands.

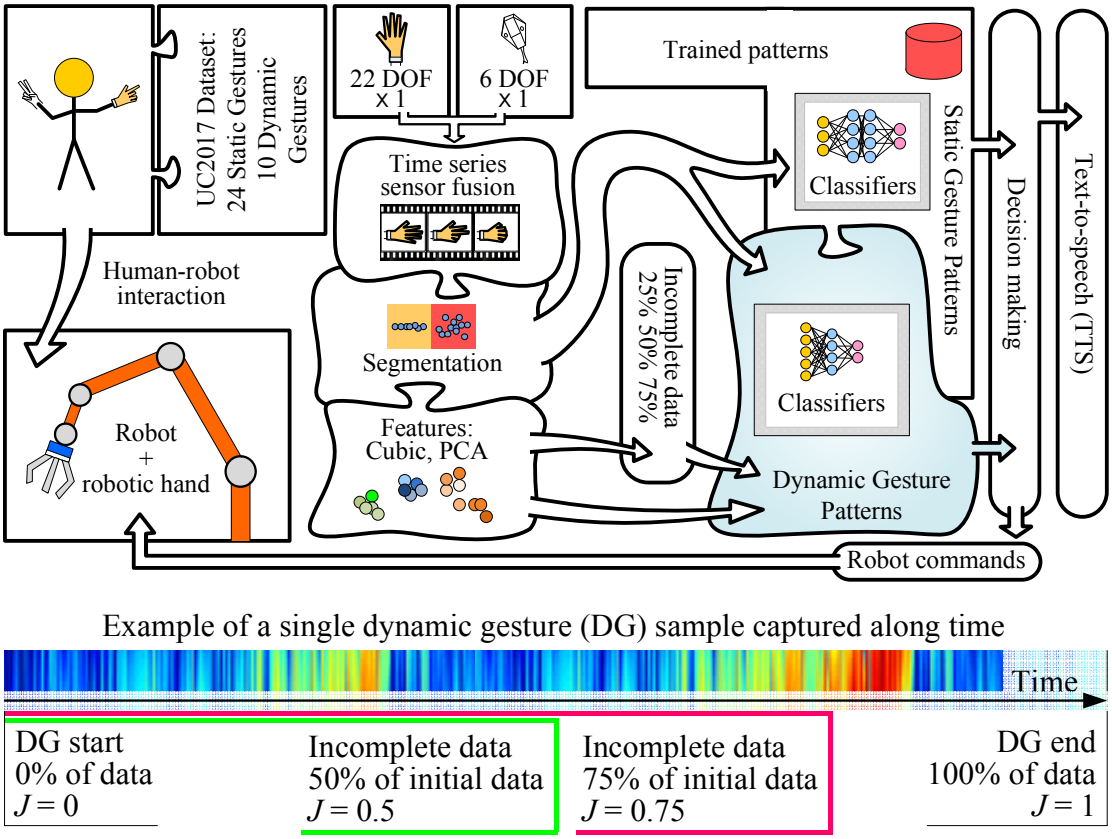


Figure 6.1: Overview of the proposed gesture-based HRI framework: data acquisition, segmentation, features, classification and the robot interface. At the bottom it is explained the meaning of incomplete data for DG classification. For example a DG can be classified with initial 50% ($J = 0.5$) of data representing such gesture, i.e., DGs can be classified in anticipation, before the user finishes the gesture in real world.

6.1.1 Motivation, Challenges and Contributions

A major challenge is related to the continuous, online and reliable recognition of gestures from real-time data streams. Continuous gesture recognition is the natural way used by humans to communicate with each other, in which communicative gestures (the effective SGs and DGs with an explicit meaning) appear intermittently with non-communicative gestures (pauses and movement epenthesis (ME) – inter-gesture transition periods), without a pre-defined sequence [5]. Many studies do not approach gesture classification in continuous and the negative effect of ME is not considered. It is also a challenge to recognize gesture patterns from incomplete data, as well as the effective process of intuitively mapping the recognized gestures into robot commands for natural and safe HRI. In this context, the motivations behind this study are:

1. Combine and fuse multi-wearable-sensor data to capture the human gestures (hand and arms) accurately, without occlusions;

2. Application of proper DDR methods to increase classification accuracy, reduce the training time, reduce the number of samples required to train the classifiers, and to allow online implementation;

3. Achieve high recognition rates (close to 100%) and assure the generalization capability in respect to untrained samples and new users;

4. Classification of DGs from incomplete data from sensors, allowing to classify a gesture while it is being performed by the human;

5. Intuitive and online interfacing with a robot using gestures.

The proposed system was evaluated by conducting several experiments using wearable/body-worn sensors (a data glove and a magnetic tracker), resulting in the following contributions:

1. The combination of DDR (CI and PCA) and ANNs for DG classification from wearable sensor data resulted in high classification accuracy that compares favorably with standard classifiers, including deep learning LSTM and CNNs. This method is computationally inexpensive, allowing the gesture-based online interaction with the robot. Gestures are recognized with an accuracy of 95.6% for a library of 24 SGs and 99.3% for 10 DGs.

2. The above results were obtained in continuous, real-time, with different subjects (user independent), and in an unstructured environment;

3. Sequential classification of DGs showed an accuracy that is higher with incomplete data (50% or 75% of initial partial data representing a gesture) than with 100% of DG data across different classifiers. In this context, DGs can be classified in anticipation, before the user finishes the gesture in real world;

4. Framework tested in real unstructured environment in which the recognized gestures serve as intuitive interface to manage an online collaborative process in which the robot helps the human preparing a breakfast meal.

6.1.2 Related Work

Gesture-based HRI for collaborative robotics is an emerging and multidisciplinary research field. Communicative gestures provide information that is hard to convey in speech, i.e., command gestures, pointing, gestures to represent meaningful, objects or actions, and mimicking gestures [149, 13]. Gestures have been proven to be one of the most effective and natural mechanisms for reliable HRI [163]. They have been used for robot teleoperation and to coordinate the interactive process and cooperation activities between human and robot. As stated in [164], a robotic task generally consists of individual actions, operations, and motions that are arranged in a hierarchical order so that this process can be managed by human gestures when properly classified [165].

An inefficient segmentation process (determining when a gesture starts and ends) results in a classification process that is more likely to fail [8]. The analysis of continuous data streams to solve spatial and temporal segmentation is challenging [166].The problem is that it is difficult to automate the segmentation process, making gesture recognition in real world scenarios a difficult task [4].

The input features for gesture recognition are normally the hand/arm/body position, orientation and motion [5], often captured from vision sensors. Owing to its naturalness, in opposition to wearable sensors that need to be attached to the human body, vision sensing is the most common interaction technology. Gesture classification from video stream requires large amount of training data, especially for state-of-the-art deep learning classifiers. Moreover, it is difficult to construct reliable features from only vision sensing due to occlusions, varying light conditions and free movement of the user in the scene [12, 13]. To improve classification reliability, a significant number of studies combine and fuse data from vision and wearable sensors. With this in mind, several approaches to gesture recognition rely on wearable sensors such as data gloves, magnetic tracking sensors, inertial measurement units (IMUs), Electromyography (EMGs), etc. In fact, these interaction technologies have been proven to provide reliable features in unstructured environments. Nevertheless, they also place an added burden on the user since they are wearable.

Some gestures, although not all, can be defined by their spatial trajectory, e.g., a circle. Burke and Lasenby succeed on using PCA and Bayesian filtering to classify these time series gestures [149]. Hidden Markov Models (HMMs) can be used to find time dependencies in skeletal features extracted from image and depth data (RGB-D) with a combination of Deep Belief Networks (DBNs) and 3D CNNs [167]. Deep learning combined with recurrent LSTM networks demonstrated state-of-the-art performance in the classification of human activities from wearable sensing [168]. Features are automatically extracted from raw sensor data, i.e., does not require expert knowledge in designing features. Reported results show that this framework outperforms competing deep non-recurrent networks. ANNs in series demonstrated superior performance in the classification of high number of gesture patterns [14]. Field et al. used a Gaussian Mixture Model (GMM) to classify human's body postures (gestures) with previous unsupervised temporal clustering [169]. Switching Gaussian Process Dynamic Models (SGPDM) are proposed to capture motion dynamics and to identify motion classes such as walk or run, and smile or angry [170]. Recognition performance on real videos (low quality, pose change and low frame rate) demonstrated that the SGPDM model can efficiently track composite motions with diverse dynamics. A framework for dynamic hand gesture recognition using Generalized Time Warping (GTW) for alignment of time series is proposed in [171]. Features are extracted from the aligned sequences of hand gestures based on texture descriptors and CNNs are used for hand motion recognition. Autoencoders and stacked autoencoders (SAE) have been successfully used for feature representation in diverse applications [172].

Recent studies report state-of-the-art results in hand detection and gesture classification from RGB-D video using deep learning [173]. Generally speaking, deep learning requires large amount of training data, it is computationally expensive to

train, and it is challenging to determine hyperparameters since deep networks are a black box (it is difficult to know how and why we have a certain output). Boosting methods, based on ensembles of weak classifiers, allow multi-class hand detection [174]. Nevertheless, problems related to false positives/negatives still difficult the use of gestures as a reliable mechanism to online interact with a machine like a robot.

The evolution in pattern recognition has been enormous in the last few years. However, many of existing solutions address object or SG classification which is less challenging than DG classification. Results obtained in well-established datasets have good accuracy in offline classification but normally are not tested online and the processing time is not mentioned. The ability to classify a gesture online is critical to interface with a machine/robot. Finally, no studies approach gesture classification from incomplete data being normally assumed that more data results in better accuracy.

6.2 Gesture Classification

6.2.1 Problem Formulation

Within a continuous data stream, there may be a sequence of SGs and DGs with no specific order. As new frames are acquired, they are segmented into static or dynamic frames with a motion detection algorithm [8]. This algorithm identifies motion, or a static behaviour, including the sudden inversions of movement direction which are common in DGs. This is achieved by the analysis of velocities and accelerations numerically derived from positional data. A genetic algorithm is used to compute feasible thresholds from calibration data. As a result, we have static and dynamic blocks of frames contiguous in time. Static blocks are SG candidates and dynamic blocks are DG candidates. Therefore, we propose two independent classifiers, one for the classification of SGs and the other for the classification of DGs.

The segmentation function Γ based on a motion-threshold algorithm is applied to a window of a stream of data $\boldsymbol{S}$, of dimensionality d and length n: $\{(\boldsymbol{S}, \Gamma(\boldsymbol{S})) : \boldsymbol{S} \in \mathbb{R}^{d\times n}$ and $\Gamma(\boldsymbol{S}) \in \{0,1\}^n\}$. The static frames indicating no motion (input data for the SG classifier) are defined by $m_i = 0$ and the dynamic frames indicating motion (input data for the DG classifier) by $m_i = 1$.

The dynamic segments are extracted by a search function that finds transitions in m (from 0 to 1 and 1 to 0). Given two consecutive transitions in the frames i and $i + k$ so that $m_{i-1} = 0$, $m_i = 1$, $m_{i+k-1} = 1$ and $m_{i+k} = 0$, a DG sample is defined by:

$$\mathbf{X}^D = [S_{\bullet i}\ S_{\bullet i+1}\ ...\ S_{\bullet i+k-1}], \quad \mathbf{X}^D \in \mathbb{R}^{d\times k} \tag{6.1}$$

where the $S_{\bullet i}$ vector is the ith column (frame) of the data stream. In terms of matrix notation, being $\mathbf{A} \in \mathbb{M}^{p\times q}$, $\mathbf{A}_{ij}$ represents the element of the array $\mathbf{A}$ with row i and column j, $\mathbf{A}_{i\bullet} \equiv [\mathbf{A}_{i1} \cdots \mathbf{A}_{in}]$ and $\mathbf{A}_{\bullet j} \equiv [\mathbf{A}_{1j} \cdots \mathbf{A}_{nj}]^T$, and $\mathbb{M}$ is the notation for a real-valued matrix.

The static gesture samples are considered the first frame after a transition from $m_{i+k-1} = 1$ to $m_{i+k} = 0$:

$$\mathbf{X}^S = S_{\bullet i+k}, \quad \mathbf{X}^S \in \mathbb{R}^d \tag{6.2}$$

Hereinafter, the notation for a sample independently of its nature (static or dynamic) is $\mathbf{X}$. Static and dynamic samples are differentiated by their dimensionality. The i-th sample of a dataset is represented by $\mathbf{X}^{(i)}$. We represent the feature extraction pipeline by Π, which is used to transform the samples into the predictors $\mathbf{z}$ that feed the classifiers: $\left\{(\boldsymbol{X}, \mathbf{z} = \Pi(\boldsymbol{X})) : X \in \mathbb{R}^{d \times n} \text{ and } \mathbf{z} \in \mathbb{R}^b\right\}$, where d is the number of channels of the sample and n its length. The target vectors are one-hot encoded class indexes. For any given sample, the target class has the index o and the target vector $\mathbf{t}^{(o)}$ is defined by $\mathbf{t}_j^{(o)} = \delta_{oj},\ j = 1, ..., n_{classes}$. Therefore $\mathbf{t} \in \{0, 1\}^{n_{classes}}$, δ is the Kronecker delta and $\mathbf{t}_j$ is the j-th element of $\mathbf{t}$.

For DGs, the transformation Π could yield a long vector, which often makes training the classifier harder. Therefore, we introduce DDR at the end of the pipeline Π, such as PCA and CI. The feature vectors are fed into the respective classifiers.

In this study, our aim is to map the classified gestures into robot actions, such as moving to a target or halting movement. However, before issuing an action command, we must exclude poorly classified patterns from the stream. For example, we can exclude classifications by context and by applying a threshold to the classification score:

$$\mathbf{o}_i = \begin{cases} \text{SG}_t, & \text{if } p(\mathbf{y} = \mathbf{t}|\mathbf{z}) \geq \tau^S \wedge m_i = 0 \\ \text{DG}_t, & \text{if } p(\mathbf{y} = \mathbf{t}|\mathbf{z}) \geq \tau^D \wedge m_i = 1 \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$

where $\mathbf{o}_i$ is the output gesture class number, $p(\mathbf{y} = \mathbf{t}|\mathbf{z})$ is the likelihood of the classifier's output $\mathbf{y}$ being in the class $\mathbf{t}$ given the $\mathbf{z}$ input, τ is the likelihood threshold and m_i is the motion variable associated to $\mathbf{z}$.

6.2.2 Feature Dimensionality Reduction

For SGs no DDR is proposed since the feature space is relatively small. On the contrary, DDR is beneficial for DGs feature extraction due to the relatively large feature space and to standardize the variability of DG length (reducing the gesture length to a small fixed size – resampling). We propose two forms of dimensionality reduction, using CI and PCA. CI is used to transform any variable-length DG sample $\mathbf{X}^{(i)} \in \mathbb{M}^{d \times n}$ into a fixed-dimension sample $\mathbf{X}' \in \mathbb{M}^{d \times k}$, effectively reducing the sample dimensionality if $k < n$. PCA performs an orthogonal linear transformation of a set of n d-dimensional observations, $\mathbf{X} \in \mathbb{M}^{d \times n}$, into a subspace defined by the Principal Components (PCs). The PCs have necessarily a size less than or equal to the number of original dimensions, d. The first PC has the largest variance observed. Each of the following PCs is orthogonal to the preceding component and has the highest variance possible under this orthogonality constraint. The PCs are the eigenvectors of the covariance matrix and its eigenvalues are a measure of the

variance in each of the PCs. Therefore, PCA can be used for reducing the dimensionality of gesture data by projecting such data into the PC space and truncating the lowest-ranked dimensions. These dimensions have the lowest eigenvalues, so that truncating them retains most of the variance present in the data, i.e., most of the information of the original data is kept in the reduced space. In this study, the singular vectors of the samples across time are calculated and used as features. The first singular vector determines the direction in the PC-space in which there is the most significative variance along a DG. This means that the singular vector is a measure of the relative variance of each variable along time and good features for DG classification are expected. Another advantage is that the PCs can be calculated even before a DG is finished (incomplete data), remaining a good predictor. As a result, these features are actually time series since we can calculate them on any window of data starting with the first frame and ending with any arbitrary frame of the sample.

6.2.3 UC2017 Hand Gesture Dataset

We introduce the UC2017 static and dynamic gesture dataset. Most researchers use vision-based systems to acquire hand gesture data. Despite that, we believe that more reliable results from more complex gestures can be obtained with wearable sensor systems. There are not many datasets using wearable systems due to the plethora of data gloves in the market and their relative high cost. For these reasons, we opted by creating a new dataset to present and evaluate our gesture recognition framework. The objectives of the dataset are: (1) provide a superset of hand gestures for HRI, (2) have user variability, (3) to be representative of the actual gestures performed in a real-world interaction.

We divide the dataset in two types of gestures: SGs and DGs. SGs are described by a single timestep of data representing a single hand pose and orientation. DGs are variable-length time series of poses and orientations with particular meanings. Some of the gestures of the dataset are correlated with a certain meaning in the context of HRI, while others were arbitrary selected to enrich the dataset and add complexity to the classification problem.

The library is composed of 24 SGs and 10 DGs, Fig. 6.2. The dataset includes SG data from eight subjects with a total of 100 repetitions for each of the 24 classes (2400 samples in total). The DG samples were obtained from six subjects and has cumulatively 131 repetitions of each class (1310 samples in total). All of the subjects are right-handed and performed the gestures with their left hand.

A data glove (CyberGlove II) and a magnetic tracker (Polhemus Liberty) are used to capture the hand shape, position and orientation over time. The glove provides digital signals g_i proportional to the bending angle of each one of the 22 sensors elastically attached to a subset of the hand's joints: 3 flexion sensors per finger, 4 abduction sensors, a palm-arch sensor, and 2 sensors to measure wrist flexion and abduction. These 22 sensors provide a good approximation of the hand's shape. The tracker's sensor is rigidly attached to the glove on the wrist and measures its position in Cartesian space and orientation in respect to a ground-fixed frame (a magnetic

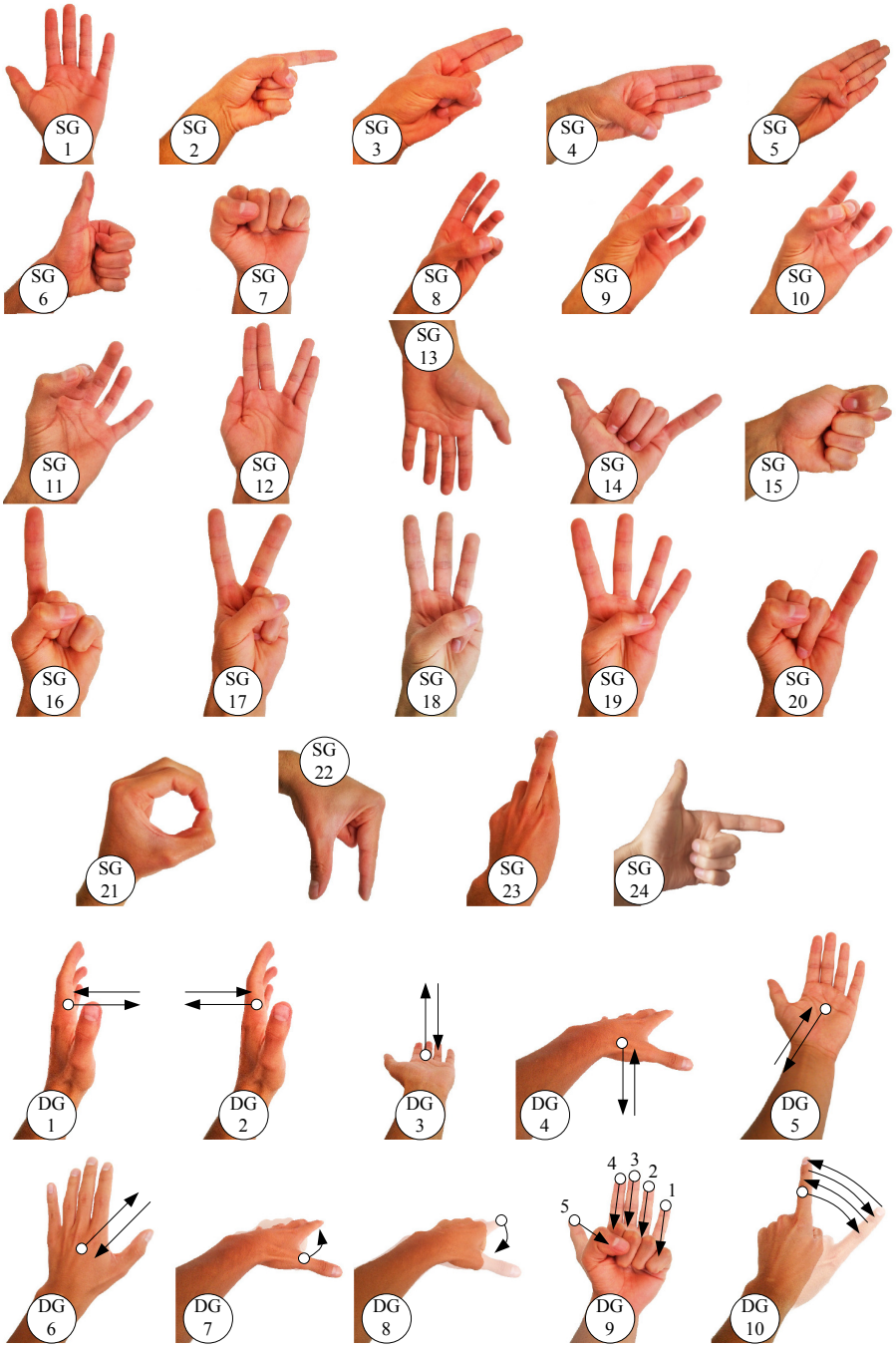


Figure 6.2: Representations of the library of 24 static gestures and 10 dynamic gestures of the UC2017 library.

source cube defines the reference coordinate system frame). The orientation is the rotation between the fixed frame and the frame of the tracker sensor, given as the intrinsic Euler angles yaw, pitch and roll (ZYX). The Cartesian position (x, y, z) is denominated by (l_1, l_2, l_3) and in terms of orientation the roll angle is denominated by l_4, the pitch by l_5 and the yaw by l_6. Sensor data are fused together online since the sensors have slightly different acquisition rates – 100Hz for the glove and 120Hz for the tracker. Tracker data are under-sampled by gathering only the closest tracker frame in time.

A goal was to obtain multiple repetitions of each gesture in the library to build the dataset. We also want the dataset to be representative of real-world conditions, so we must guarantee that the samples are independent.

The magnetic tracker reference was fixed in a location free of magnetic interference. The users are then asked to put on the data glove on their own on their left hand, even though all of our test subjects were right-handed. As a result, the sensors are not carefully placed, which should yield a dataset with larger variance. There is no calibration done in this setup. The subjects follow a graphical interface that shows the representation of the gesture to be performed and press a button to save a sample. The order of the gestures was randomized to prevent order dependencies. Furthermore, the subjects were requested to repeat the sampling for two to three different sessions. We have also implemented an online movement detection algorithm to facilitate the labeling of DGs, namely the starting and ending frames.

A final point should be made about the random sampling of the DGs. We have included in the samples the transition between the ending pose of the previous sample and the starting pose of a sample (movement epenthesis). Owing to random sampling, there is a high likelihood that all of the possible transitions were recorded.

The dataset and accompanying code are publicly available on Github (`https://github.com/MiguelSimao/UC2017_Classification`).

Coordinate transformation: Gesture classification has to be independently of the subject's position and orientation in space. Since the user is free to move around in the world reference frame $\{W\}$, we need to make sure that every gesture sample has its feature data reported to their local reference frame $\{L\}$. Origin and orientation of $\{L\}$ are defined in relation to $\{W\}$ at the instant a gesture begins.The proposed transformation is composed of a 3D translation and a rotation around the vertical axis z.

We denote l_i and g_i as the i-th DOFs of the tracker and glove, respectively. At the beginning of a DG sample $\mathbf{X}^{(i)}$, the yaw ψ_0 (l_6) and position $\mathbf{p}_0^W$ (l_1 through l_3) of the first frame are stored. The yaw angle is used to calculate the rotation matrix for each sample. It allows to consistently distinguish important directions, such as right, left, front and back, in respect to the user. The rotation is applied to every frame of a sample so that we can translate the coordinate system to frame $\{L\}$, Fig. 6.3:

$$\mathbf{p}_i^L = \mathbf{R}_L^{W\,T}(\psi_0) \cdot \left(\mathbf{p}_i^W - \mathbf{p}_0^W\right) \tag{6.4}$$

where $\mathbf{p}_i^L$ is the position of the ith frame in respect to the local reference frame

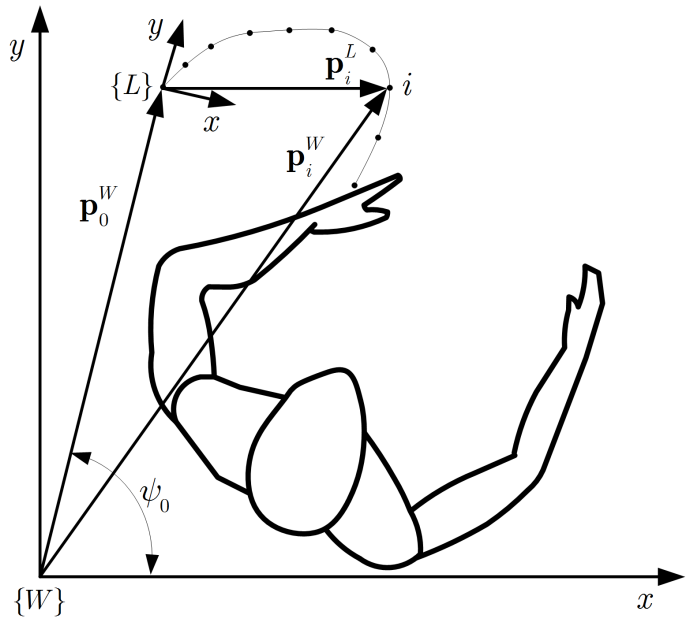


Figure 6.3: Representation of the transformation of the world coordinates $\{W\}$ of a gesture to a local coordinate frame $\{L\}$.

$\{L\}$, and $\mathbf{R}_L^W(\psi_0)$ is the rotation matrix that represents the rotation around z of the world reference frame $\{W\}$ to $\{L\}$ by ψ_0 degrees. The rotation matrix is given by:

$$\mathbf{R}_z(\psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6.5}$$

After the transformation, the yaw angle is $\psi_i^L = \psi_i^W - \psi_0^W$. In summary, we have a transformation function Ψ applied to a DG sample $\mathbf{X}^{(k)} \in \mathbb{M}^{d \times n}$:

$$\begin{aligned} \Psi: \quad & \mathbb{R}^{6\times\bullet} \rightarrow \mathbb{R}^{6\times\bullet} & & \\ & \mathbf{X}_{ij}^{(k)} \rightarrow \left(\mathbf{p}_1^L\ \mathbf{p}_2^L\ \mathbf{p}_3^L\ \mathbf{l}_4\ \mathbf{l}_5\ \psi^L\right)_{\bullet j}^{(k)\,T}, & \begin{matrix} i = 23, ..., 28 \\ j = 1, 2, ..., n \end{matrix} \end{aligned} \tag{6.6}$$

where $(...)_{\bullet j}^{(k)}$ corresponds to the j-th timestep of of sample k.

Dataset split: The dataset is split before feature extraction. It was shuffled and split in three subsets: training (70%), validation (15%) and test (15%). The training set was used to train the classifiers and to obtain feature scaling parameters, such as mean and standard deviation. The classifiers' hyperparameters were optimized for accuracy on the validation set. The generalization of the model is measured by the accuracy on the test set. The samples of one subject were held-out from the training set to ascertain the performance on new users. Users that trained the system are designated by "trained users" and users that that did not train the system are designated by "untrained users".

6.2.4 Feature Extraction

For **Static Gestures** the features are all the angles provided by the glove, g_1 g_2 ... g_{22}, and the pitch angle, l_5 (to differentiate gestures with similar hand-shapes and distinct orientation). Thus, the features chosen are simply a subset of the available data. Finally, the features are standardized by $x'_i = (x_i - \bar{x}_i)/s_i$, where x'_i is the standardized value of feature i, x_i is the value of the feature, $\bar{x}_i$ and s_i are the mean and standard deviation of the feature in the training set. The validation and test sets are standardized by these same means and standard deviations.

For **Dynamic Gestures** we propose three different sets of features. For all sets, data samples are preprocessed according to Ψ, defined in (6.6):

$$\mathbf{X}'^{(i)} = \Psi(\mathbf{X}^{(i)}), \quad \mathbf{X}^{(i)} \in \mathbb{R}^{28\times n} \tag{6.7}$$

where n is the length of the DG sample. Starting from $\mathbf{X}'$, the first proposed set DG-CI uses the full DG data resized to a fixed length by applying CI. The second set, DG-PV, is based on PCA and represents the extraction of the first principal vector (PV) from DG data. The third set is the preprocessed data, which we call RAW.

For DG-CI, given a DG sample $\mathbf{X}^{(i)}$ with n frames ($\mathbf{X}^{(i)} \in \mathbb{R}^{28\times n}$), the goal is to resample it to a fixed size n'. The value for n' can be chosen arbitrarily but higher values have a detrimental effect on the classification accuracy. Training the classifier is faster and often better with less features. For all the experiments we have used $n' = 20$ because the gestures length in the dataset varies between 20 and 224 frames and therefore, it was selected the lowest length. Applying CI, the result is a matrix $\mathbf{Z} \in \mathbb{R}^{28\times 20}$. By concatenating every frame vertically, $\mathbf{Z}$ is transformed into a vector $\mathbf{z} \in \mathbb{R}^{560}$:

$$\mathbf{z}^{(i)} = \left({\mathbf{Z}^{(i)}_{\bullet 1}}^T, {\mathbf{Z}^{(i)}_{\bullet 2}}^T, \ldots, {\mathbf{Z}^{(i)}_{\bullet 20}}^T \right)^T \tag{6.8}$$

In DG-CI the feature extraction involves the whole DG data, so there is only a classification after the gesture is complete. However, it is beneficial to have an early classification from incomplete data, i.e., before the whole gesture data is available. For DG-PV, PCA allows to obtain features from incomplete gesture data and still obtain time-coherent features. We apply this methodology for each timestep of the gesture. The feature vector for sample i at timestep j is calculated by:

$$\boldsymbol{z}^{(i)}_j = pv\left(\left[\mathbf{X}'^{(i)}_{\bullet 1} \; \mathbf{X}'^{(i)}_{\bullet 2} \; \ldots \; \mathbf{X}'^{(i)}_{\bullet j} \right] \right), \quad j > 1 \tag{6.9}$$

where pv is a function that extracts the principal vector from its argument. $\mathbf{X}'^{(i)}$ is the standardized sample $\mathbf{X}^{(i)}$, i.e., with zero mean and unit variance.

A single sample originates multiple feature vectors depending on the timestep j. We used the set $J^{(i)} = \left\{ x : \; 2 \leq x \leq n^{(i)} \wedge x \in \mathbb{N} \right\}$ for training and validation of the classifiers, where $n^{(i)}$ is the sample length. To simplify the display of results, we tested with a subset $J^{(i)} = \{\lceil 0.25n \rceil, \lceil 0.5n \rceil, \lceil 0.75n \rceil, \lceil 1.0n \rceil\}$, where $\lceil\rceil$ represents the ceiling function. Simply put, this means that we are testing the features sets

extracted from the samples starting at the first timestep and ending at 25%, 50%, 75% and 100% of gesture length.

The final step of feature extraction for all features sets is feature scaling, the standardization of the features as described for SGs.

6.2.5 Multi-Layer Neural Networks

A multi-layer feed-forward neural network is a fully connected ANN that is very commonly used for classification tasks. It has a node input layer, a number of hidden layers and an output layer, in which every node is connected to every node of the following layer. Formally, they define a transformation $f: \mathbb{R}^D \rightarrow \mathbb{R}^N$, where D is the size of the input vector (features) and N is the size of the output vector (number of classes). The state $y^{(i+1)}$ of each layer $(i+1)$ is defined by the state $y^{(i)}$ of the previous layer (i), so that:

$$\mathbf{y}^{(i+1)} = f^{(i+1)}(\mathbf{y}^{(i)}) = s^{(i+1)}\left(\mathbf{b}^{(i+1)} + \mathbf{W}^{(i+1)}\mathbf{y}^{(i)}\right) \tag{6.10}$$

where s is the transfer function, $\mathbf{b}$ is the biases vector and $\mathbf{W}$ is the weight matrix. Typically, the function s is the hyperbolic tangent, $\tanh \mathbf{x} = \frac{e^{\mathbf{x}}-e^{-\mathbf{x}}}{e^{\mathbf{x}}+e^{-\mathbf{x}}}$. For classification ANNs, the codomain of the last layer is normally $[0,1]$ –, which is not the codomain of *tanh*, so either the sigmoid function ($\text{sig}(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$) or a *softmax* function is used:

$$\sigma(\mathbf{x})_j = \frac{e^{x_j}}{\Sigma_{k=1}^{K} e^{x_k}},\ j = 1, ..., K \tag{6.11}$$

where σ_j represents the j-th element of the *softmax* function output, e is the exponential function and K is the length of $\mathbf{x}$, i.e., the number of classes. For the last layer, $\mathbf{y} \in \mathbb{R}^K$ with $\sigma(\mathbf{y}) \in [0,1]$ and $\Sigma\sigma(\mathbf{y}) = 1$. The input on the network is state for $i = 0$ and is denoted by $\mathbf{x}$. Generalizing for a network with n layers ($n-1$ hidden layers), f is obtained from the composition function, considering that $f^{(n)} \circ f^{(n-1)}\left(\mathbf{y}^{(n-1)}\right) \equiv f^{(n)}\left(f^{(n-1)}\left(\mathbf{y}^{(n-1)}\right)\right)$:

$$\mathbf{y} = f^{(n)} \circ f^{(n-1)} \circ ... \circ f^{(1)}(\mathbf{x}) \tag{6.12}$$

The problem is that $\mathbf{W}$ and $\mathbf{b}$ are undetermined. Their estimation is called the training of the network. Given a set of samples $\mathbf{X}$ and being known the target classes $\mathbf{t}$ – the training samples (supervised learning), the objective is obtaining weights and biases that optimize a performance parameter E, e.g., the squared error $E = (t-y)^2$. The optimization is usually done with the Scaled Conjugate Gradient (SCG) Backpropagation (BP) method, typically with a classification cross-entropy loss function, $E_{ce} = -\mathbf{t} \cdot \log \mathbf{y}$.

BP is an iterative optimization method that relies on the initialization (often done randomly) of the weight and bias vector, $\tilde{w}_1$ $(k = 1)$. The next step is determining the search direction $\tilde{p}_k$ and step size α_k so that $E(\tilde{w}_k + \alpha_k) < E(\tilde{w}_k)$. This leads to the update $\tilde{w}_{k+1} = \tilde{w}_k + \alpha_k \tilde{p}_k$. If the first derivative $E'\left(\tilde{w}_k \neq \tilde{0}\right)$, meaning that we are not yet at a minimum/maximum, then a new iteration is made

Table 6.1: Training and inference times of several classifiers, accuracy on the train, validation and test data subsets for SGs. The test scores are divided into the scores of the trained and untrained (other) users.

	Time (s)		Accuracy (%)		
Model	Train	Test	Train	Validation	Test (other users)
ANN	127.0	0.1	97.9	94.2	94.6 (87.9)
QDA	0.0	0.0	99.6	91.7	94.9 (66.7)
RBF SVM	0.1	0.2	98.0	94.2	95.2 (83.3)
Gaussian Process	23.8	13.1	99.8	991	94.9 (69.7)
KNN	0.0	4.2	96.0	89.2	93.9 (53.0)
Naive Bayes	0.0	0.0	93.4	88.3	91.2 (69.7)
Random Forest	0.1	0.0	99.8	94.7	95.6 (92.4)

$(k = k + 1)$ and a new search direction is found. Else, the process is over and $\tilde{w}_k$ should be returned as the desired minimum. BP variations typically rely on different methods to find $\tilde{p}_k$, determination of α_k or new terms to the weight update equation. This leads to the introduction of user-defined hyperparameters that will have to be determined by trial and error.

6.3 Results and Discussion

The accuracy of the classifier models on both the SG and DG data was obtained considering a segmentation accuracy estimated to be about 98%. The error is mostly oversegmentation, i.e., pauses in the middle of a DG where the subject slows down or hesitates. In this scenario, the classification of the DGs is more likely to fail due to lack of gesture data.

6.3.1 Static Gestures

The accuracy of several classifiers was evaluated on the UC2017 dataset. The objective is to compare the performance of ANNs with other machine learning models: K-Nearest Neighbors (KNN), Support Vector Machines with a Radial Basis Function kernel (RBF SVM), Gaussian Processes (GP), Random Forests (RF), Gaussian Naive Bayes (NB) and Quadratic Discriminant Analysis (QDA). We measured the training and inference times, and the accuracy of the models.

The ANN used, implemented with Keras, is a feed-forward neural network (FFNN). Its architecture and hyperparameters were optimized on the validation dataset using random grid search. Grid search and manual search are the most common techniques for hyperparameter optimization [175]. Grid search generates candidates from a grid of parameter values in which every possible combination of values is tested to optimize hyperparameters (detailed parameters and Python code available in supplementary material). The optimal network has two dense hidden

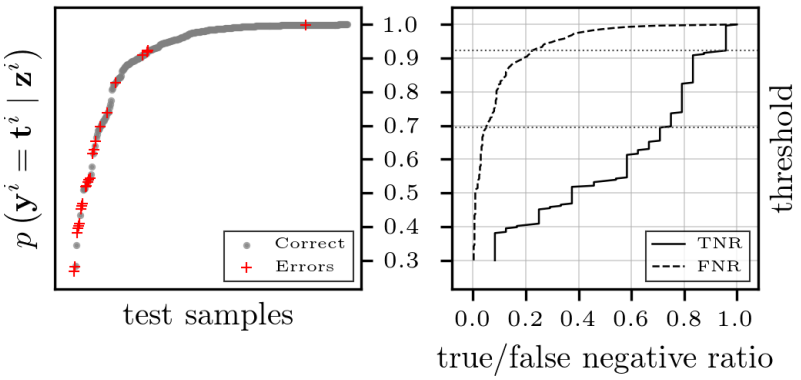


Figure 6.4: On the left, sorted activation values of the winning class for each sample of the SG test set. The red crosses correspond to classification errors. On the right, the true and false negative ratios (TNR/FNR) when we apply a threshold to discard errors. The horizontal dotted lines correspond to the 0.696 and 0.923 thresholds.

layers of 200 neurons each. Between these layers, there is a Gaussian noise layer with $\sigma = 0.6$. The transfer functions of the dense layers are linear and rectified. A final layer implements the *softmax* function to obtain the probabilities of each class. For weight regularization, we used the L2 distance with a factor of 0.005 and a weight decay coefficient of 10^{-7}. The optimization was done using Stochastic Gradient Descent (SGD) with batches of 32 and a learning rate of 0.001. Furthermore, in order to prevent overfitting, we used early stopping when there is a minimum on the validation loss with a tolerance of 10 epochs. The hyperparameters of the remaining classifiers were optimized using manual search (detailed parameters and Python code available in supplementary material).

The performance of the trained classifiers is shown in table 6.1. The best performance on the test set was 95.6% on the trained users (92.4% in the untrained), obtained with the RF. The ANN was slightly worse, with 94.6% and 87.9% accuracy on the trained and untrained users, respectively, leading to the conclusion that in this case, the RF is generalizing better to new users than the ANN. The next best performance was the SVM, with 95.2% accuracy . The other models performed very well on the trained users, but clearly overfitted the dataset, since their accuracy on untrained users is below 70%.

The accuracy of the ANN model for each individual subject varies between 87.9% (untrained user) and 100.0%, while one of the trained users reached only 88.2%. This subject was one of the authors and was involved in the definition of the gesture library, which may have generated samples that are significantly different from those of the other users. The distribution of errors per class was nearly random, with no gestures being mixed consistently.

We also present the test results of the feasibility of removing poorly classified gestures by their score, Fig. 6.4. This analysis was done with the ANN classifier, since it outputs a probability distribution over the classes. We present the score of the winning class $p(\mathbf{y}^i = \mathbf{t}^i \mid \mathbf{z}^i)$, i.e., the probability of the ANN output $\mathbf{y}^i$

Table 6.2: Feature sets considered for the experiments. CI refers to cubic interpolation, PV to principal vector's and RAW to no feature extraction.

Feature set	Description
CI-FULL	CI applied to raw preprocessed data describing a full DG sample
PV-FULL	PVs applied to raw preprocessed data describing a full DG sample
PV-TS	PVs applied sequentially to a DG sample, starting from its first frame to an arbitrary timestep.
RAW-LSTM	Raw preprocessed data classified by LSTMs
RAW-CNN	Raw preprocessed data classified by CNNs

Table 6.3: Classification accuracy for the full DG experiments. The test scores are divided into the scores of the trained and untrained (other) users.

	CI-FULL			PV-FULL		
Model	Train	Val	Test (other)	Train	Val	Test (other)
ANN	100.0	98.5	99.3 (96.2)	99.7	91.3	94.4 (66.0)
LDA	100.0	98.0	97.2 (92.5)	67.2	60.7	68.8 (50.9)
KNN	97.9	94.4	96.5 (86.8)	90.0	81.1	84.7 (62.3)
RF	100.0	98.0	97.2 (86.8)	100.0	92.3	88.9 (67.9)
SVM	99.8	98.5	97.9 (96.2)	87.9	82.1	79.2 (60.4)

being the expected target $\mathbf{t}^i$ given the feature vector $\mathbf{z}^i$. It is impossible to define a score threshold to exclude misclassifications without excluding also some good ones. The trade-off is demonstrated in Fig. 6.4 on the right, via the true and false negative ratio. If we agree that a 5% False Negative Ratio (FNR) is acceptable, the threshold 0.696 reduces miss-classifications by 71%. On the other hand, if we want 95% of miss-classifications to be discarded, we lose 22% of valid classifications with a score threshold of 0.923. The latter is not a particularly good trade-off. In this context, another solution should be found, such as generating false samples so that the classifier learns how to better separate them.

6.3.2 Dynamic Gestures

From the UC2017 dataset four sets of data contemplating different features were considered for the experiments, Table 6.2. CI-FULL and PV-FULL output a single classification per DG, while PV-TS, RAW-LSTM and RAW-CNN output a classification for each timestep of a DG.

The results for the experiments using **CI-FULL** features are shown in Table 6.3. Multiple classifiers were tested and the results report the accuracy achieved by the best ones. The hyperparameters were chosen by manual search for all the classifiers

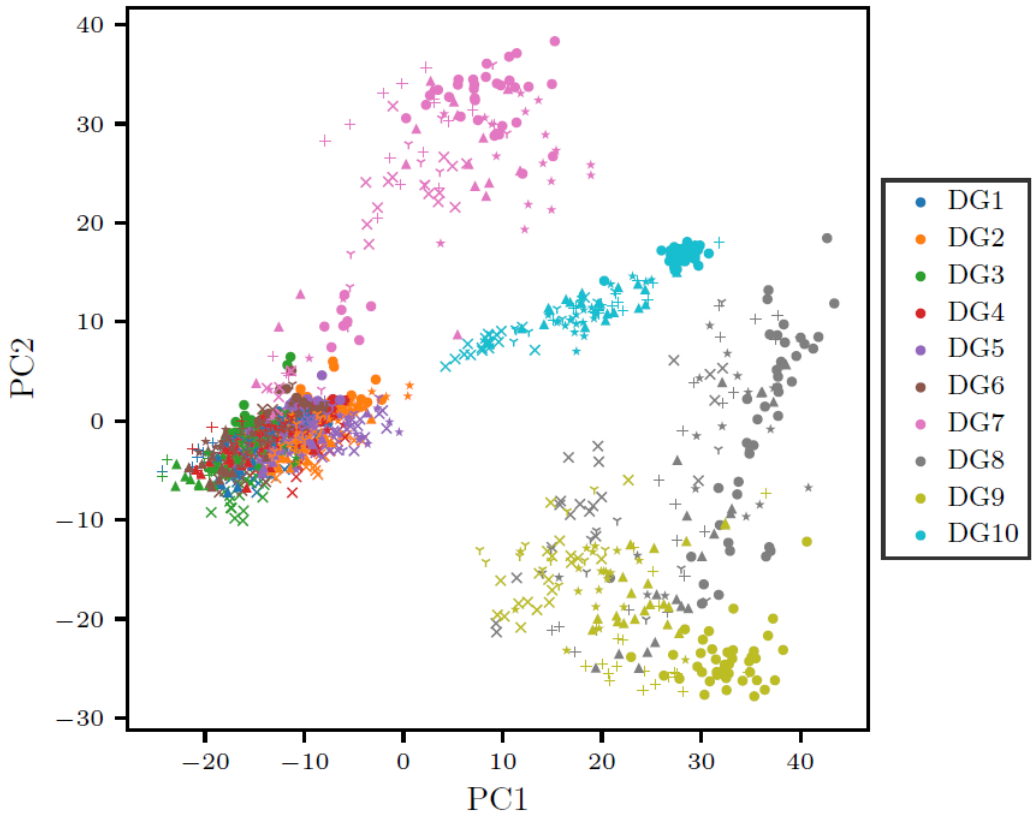


Figure 6.5: DGs projected on the plane defined by the first two principal components of the entire dataset - training, validation and test data). The ten DG classes are represented by different colours and markers. From DG1 to DG6 we can observe a cluster and then for DG7, DG8, DG9 and DG10 we have other clusters. This is because the gestures from DG1 to DG6 are all performed with the hand open (no variations in finger angle data).

(detailed parameters and Python code available in supplementary material). As an example, the ANN has two hidden layers of 100 and 200 nodes each, their activation function is linear and rectified, and the output is the *softmax* function. The weights were regularized using the L2 distance. For optimization, SGD was used with a batch size of 128 and a learning rate of 0.01.

The accuracy of the classifiers is generally excellent, around 97%, in the test set for trained users and up to 96.2% for untrained. The KNN and RF classifiers did not generalize as well to new users, reaching an accuracy of just 86.8%. This is most likely explained by the size of the feature vector (560) and comparatively low number of samples. On the other hand, the SVM performed nearly as well as the ANN on untrained users (96.2%), but worse on trained users (99.3% vs 97.9%).

The results for the experiments using **PV-FULL** features are shown in Table 6.3. Fig. 6.5 shows the ten DGs projected on the plane defined by the first two principal components of the data. We have tested the same classifiers as in CI-FULL, so that we can establish a comparison between the two feature sets. The accuracy is

Table 6.4: DG classification sequential accuracy for the time-series based experiments PV-TS, RAW-CNN and RAW-LSTM at 25, 50, 75 and 100% of DG completion. The test scores are divided into the scores of the trained and untrained (other) users.

	Time (s)		Train accuracy (%)				Validation accuracy (%)			
Model	Train	Test	0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00
ANN	94.7	0.2	95.3	96.9	95.4	91.1	74.5	86.7	88.3	85.2
KNN	25.6	3.4	99.9	99.9	100.0	99.5	67.9	82.1	83.2	78.6
RF	54.6	0.2	100	100.0	100.0	100.0	74.5	88.8	91.3	86.7
SVM	102.4	8.8	97.8	97.8	97.1	93.1	73.0	83.7	87.2	83.2
LSTM	390.5	69.5	87.9	96.8	99.8	99.8	78.1	92.9	97.4	98.5
CNN	66.4	2.7	86.6	96.5	97.1	88.7	81.1	92.9	97.1	88.7

	Test accuracy (%)			
Model	0.25	0.50	0.75	1.00
ANN	77.8 (49.1)	91.0 (75.5)	88.9 (71.7)	84.7 (62.3)
KNN	70.8 (43.4)	83.3 (69.8)	81.2 (69.8)	81.9 (54.7)
RF	70.1 (54.7)	88.9 (75.5)	87.5 (67.9)	80.6 (67.9)
SVM	75.7 (41.5)	89.6 (71.7)	88.9 (69.8)	81.2 (60.4)
LSTM	81.7 (50.9)	95.1 (74.5)	97.2 (87.3)	96.5 (89.1)
CNN	84.7 (60.4)	94.4 (84.9)	92.4 (73.6)	81.9 (54.7)

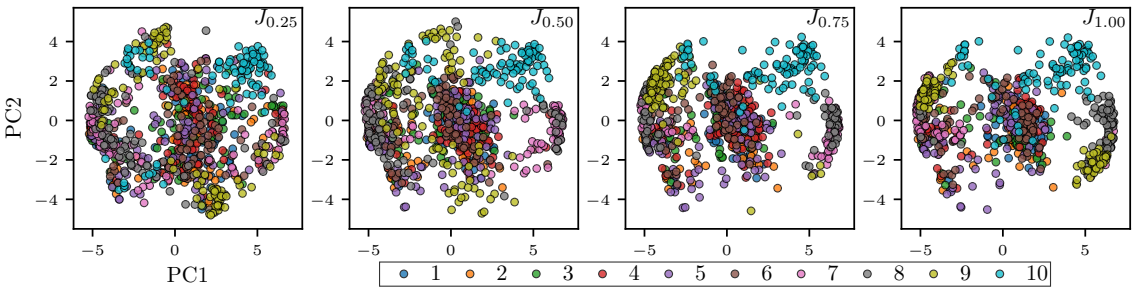


Figure 6.6: Plots of the features of the DG training set, with 25%, 50%, 75% and 100% of the data, in a reduced 2D principal component space. Each color represents a different class.

generally and markedly below of those obtained by CI-FULL. The ANN accuracy decreased by 4.9% for trained users, even with an updated architecture composed by two hidden layers with 300 units each. The generalization to untrained users was poor, with just 66.0% accuracy. The RF achieved slightly better results on the training and validation sets than the ANN, but the test accuracy was lower (88.9%). All of the other models performed significantly worse. As a conclusion, the PV features lose more information about the DGs than the CI features, making classification harder. However, the PV features can be calculated with an arbitrary number of frames of data, without the full gesture data, therefore allowing sequential (online) classification.

The results for the experiments using **PV-TS** features are shown in Table 6.4. For this case we present the results for a feed-forward ANN model, KNN, RF and SVM. The classifiers hyperparameters were optimized again by manual search, but they remained nearly the same as for PV-FULL, since the problem is similar. However, the ANN architecture was updated to two hidden layers of 512 and 256 nodes. There is also a 0.1σ Gaussian noise layer after each hidden layer and a dropout layer (50% rate) in between. These noisy layers are needed to help the network generalize better at all timesteps, since there is now a feature vector for each timestep of each DG sample, which in turn originates much more variability in the features.

We present the training time for each model and the total inference time for the whole dataset. The accuracy results for each dataset split are presented and the test split was divided into trained and untrained users. According to our designation for incomplete data, the columns "0.25" correspond to the PV feature vectors calculated with the first 25% of timesteps of each DG sample. The same logic applies to "0.50" , "0.75" and "1.00". The column "1.00" corresponds to all of the DG timesteps being used, which is the same case as PV-FULL. However, in PV-TS we use precisely the same classification model for all timesteps.

The best classification accuracy on the test set was obtained with the ANN at all timesteps of the test set. The RF and SVM models reached nearly the same accuracy as the ANN, with 90% at the middle of the DGs and 89% accuracy at 75% of DG completion. At the end of the DGs (100% completion), the accuracy is lower for these models than for the ANN (81% vs 85%). The KNN model performed worse at most timesteps, reaching only 83.3%, 81.2% and 81.9% accuracy at 50%, 75% and 100% of DG completion. Interestingly, for all of the classifiers, the accuracy is better at 50% and 75% of DG completion than at 100%. This is likely due to the way most of the gestures of the library have a motion in one direction and then move back to the initial position. This would mean that the first half of the gesture is a better predictor of the whole gesture than the second half. For example, the second half of DG1 is very similar to the first half of DG2. To aid in visualization, the features are shown in a 2D PC space (for the test set) in Fig. 6.6. There is considerable noise at $J_{0.25}$, which is unfavorable for classification. However, after that, we see stable clusters, such as classes 8 and 10. On the other hand, class 9 sees many samples flipping their position at $J_{1.00}$, which may help explain the drop in accuracy.

RAW-LSTM experiments use a LSTM recurrent neural network that is com-

Figure 6.7: The human collaborates with the robot to prepare the breakfast meal. The video is available in supplementary material.

posed of cells that have memory which can be kept or forgotten over-time, depending on the sequence of input data. The LSTM structure and hyperparameters were obtained by manual search. There is a densely connected layer with 512 units and a 0.4σ Gaussian noise layer, which is followed by a LSTM layer of 256 cells. After that we implement a 50% dropout layer. The output layer has a *softmax* activation function. All the other layers have the hyperbolic tangent as transfer function. Additionally, we increase the weight of the timesteps after 50% of sample completion, so that the model optimizes accuracy at later stages of the gesture. Detailed parameters and Python code is available in supplementary material. In terms of accuracy on the trained users of the test split, the LSTM outperforms all the other models at 50%, 75% and 100% of gesture completion, with 95.1%, 97.2% and 96.5% accuracy, respectively. In respect to generalization to new users, the LSTM is significantly better than all other models when 75% or more data is available. At 75% and 100%, the accuracy on new users is 87.3% and 89.1%, respectively. It compares favorably to the second best performer (ANN) at the cost of a significant increase in training and inference times, due to the large number of parameters of the LSTM.

We performed the **RAW-CNN** experiment in the same conditions as the previous RAW-LSTM. The CNN used had an initial dense hidden layer with 512 nodes and a *tanh* transfer function. Afterwards, there are two convolutional (1D) layers with 100 filters each, windows of 5 timesteps and rectified linear units as transfer function. Each convolutional layer is followed by Gaussian noise layers of strength 0.2σ. The model is trained by SGD with a learning rate of 0.001. While oftentimes a CNN can be used to model sequential data with performance similar to that of a LSTM, in this case it was worse in accuracy at the later stages of the DGs, and also in terms of generalization to new users. Considering 50% of gesture data, the accuracy on trained users (94.4%) is close to the LSTM's 95.1%. The CNN also generalizes better at that timestep than the LSTM (84.9% vs 74.5%). However, for 75% of gesture completion, it is significantly worse than the LSTM for all users, and at 100% it is worse than the feed-forward ANN with just 81.9% and 54.7% accuracy for trained and untrained users, respectively. A decrease of this magnitude was unexpected and it is possibly due to the data padding that occurs during the convolution operations. The last timestep of a DG is evaluated by a convolution operation on a window centered on that timestep. However, since the window has length 5, that means that 2 timesteps correspond to CNN padding, which is generated, unreliable data, leading to the low score.

The LSTM and CNN approaches have the advantage that the raw sensor data can be fed directly into the model after scaling, unlike all of the others, which require carefully chosen feature extraction methods. For LSTMs, the disadvantage is that the training and inference time are one to two orders of magnitude higher than the other classifiers, due to model complexity. The latter is more concerning because classification must be done online for an efficient human-robot interactive process. The experimental setup demonstrated that the inference time per frame for the LSTM model is about 0.16 ms (about 6300 Hz) on a GPU, so it could be an issue for its implementation in embedded systems.

6.3.3 Robot Interface

We have implemented a gesture-based human-robot interface composed by gestures from the UC2017 dataset. Since there are delays in data acquisition, data stream segmentation, candidate sample preprocessing, classification, decision making and robot communication, we estimate that the total delay between the end of a gesture performed by the human and the robot reaction is about 300 ms.

The collaborative robot is a 7 DOF KUKA iiwa equipped with the Sunrise controller and interfaced using the KUKA Sunrise Toolbox for MATLAB [176]. The attempted collaborative robotic task consists in preparing a breakfast meal, composed of subtasks such as grabbing a cereal and yogurt bottle, and pouring into a bowl, Fig. 6.7. These tasks were performed by direct robot teleoperation, being the robot actions controlled by the human gestures and the collaborative process managed by a task manager [165]. The task manager can be setup with a number of required validations so that when a gesture is wrongly classified the system actuates to avoid any potential danger for the human or the equipment. From the library of 24 SGs and 10 DGs, three subjects were taught the mapping between gestures and specific robot commands, such as: stop motion, move along X, Y or Z in Cartesian space, rotate the robot end-effector in turn of X, Y or Z, open/close the gripper, and teleoperate the robot in joystick mode. Anytime the user is not performing a given gesture the system is paused.

All users indicated that the interaction process is very natural, since they can easily select the desired operation modes and the system is intuitive to use. During the interactive process, the reached target points can be saved and used in future robot operations. The impedance controlled robot compensates positioning inaccuracies, i.e., the users can physically interact with the robot to adjust positioning. Concerning safety, the subjects indicated that they feel safe in interacting with this robot due to the fact that the KUKA iiwa is a sensitive robot that is able to stop its motion when a pre-defined contact force is reached.

6.4 Conclusion

This work presented an online static and dynamic gesture recognition framework for HRI. Experimental results using the UC2017 dataset showed a relatively high

classification accuracy on SGs without feature extraction. For DGs, the use of CI features resulted in high offline classification accuracy using a regular ANN model. The achieved accuracy compares favourably with standard classifiers and with deep learning LSTM on online classification with PCA features. The LSTM uses scaled raw data, therefore being more easily extendable to new datasets. Nevertheless, The LSTM degrades when we compare the training and inference time, which is critical for online implementation. The sequential classification of DGs with either PCA features or raw data showed an accuracy that is higher with partial gesture data (50% or 75% of the initial frames of a DG sample) than with the full DG data. In this context, DGs can be accurately classified in anticipation, even before the user finishes the gesture in real world, thus allowing faster and more efficient gesture-based control of a robot by cutting the processing time overheads.

The human-robot interactive process demonstrated that is feasible to associate the recognized gesture patterns to robot commands and by this way teleoperate the collaborative robot in an intuitive fashion.

Future work will be dedicated to testing the proposed solution with other interaction technologies (vision, IMUs, and EMG). The promising results obtained with the classification from incomplete data will be explored as a way to anticipate robot reaction to human commands.

Chapter 7

Online Classification of Gestures with Long Short-Term Memory Networks

Abstract

Previously, we presented an online classification framework is based on unsupervised segmentation in order to divide the data stream into static and dynamic segments for individual classification. However, this step requires calibration and adds complexity to the framework, thus becoming a failure point. An alternative is the sequential (dynamic) classification of the data stream. In this work we propose the use of an LSTM neural network to improve the online classification of hand gestures with EMG signals acquired from the forearm muscles. A methodology from data source to classification output is presented.

The methodology was evaluated on the UC2018 DualMyo data set. A Feed-Forward Neural Network (FFNN) model was defined and trained with windows of data, to serve as a comparison baseline. Additionally, an alternative performance index, the gesture detection accuracy, is proposed to evaluate the performance of the model during online classification. Both the baseline and LSTM approaches achieved above 96% accuracy in nearly all data splits. The proposed detection accuracy was higher than the regular accuracy due to the decrease of the overlap ratio (truth and prediction) requirement from 100% to 50%. The detection accuracy was around 99% in all cases, except on the validation split of the static model. In this case, the LSTM was better than the FFNN (99% vs 91%, respectively). Although both models had similar accuracies, the LSTM model has a third of the parameters, thus also having smaller training and inference times. Therefore, the dynamic model appears to be better suited to online and portable recognition systems.

Keywords: Collaborative Robotics, Online Gesture Recognition, LSTM Networks

7.1 Introduction

This thesis presents an online classification framework that is based on an unsupervised segmentation method to divide the data stream into static and dynamic segments, Paper 4. However, this method requires the calibration of the motion detection algorithm to each specific user in order to maximize its performance. This calibration is done through the adjustment of a motion sensitivity factor, which can be performed online when the user wears the devices for the first time. It is an extra step that adds a failure point to the system. Therefore, it is worth exploring an alternative through the sequential (dynamic) classification of the data stream.

Sequential classification is the name given to the process of classification of a data stream, in which each new frame of data – where all channels are synchronized at the same time step – originates a new prediction. In this way, given n new time steps, the classification model outputs n predictions. Furthermore, the classification of frame i can only use the information available in that frame and the previous, i.e., $i,\ i-1,\ i-2, \ldots$. As a consequence, the classification predictions are available in real-time and there is no need to wait for a gesture to end.

It is possible to achieve sequential classification in two ways: (1) using a dynamic classification model which takes as input raw or processed data in a sequential fashion; (2) a static model whose input are features determined from a window of frames ending at each time step of the input sequence. In both cases, the model must have some type of memory of previous events. The dynamic model has the "memory" provided by the parameters that define its state at each time step, while the static model is limited to the window of time steps that it is fed.

A dynamic model has some advantages relative to a static one. It is easier to implement online, as new data can be fed directly into the model one-by-one or by batches, without affecting its output. On the contrary, the static model's data must be split into windows of constant or variable size. Therefore, a window size must be specified, which can cause a decrease in accuracy of the model. Gestures can vary in length due to the speed of the user and/or his/her range of motion. So, a small window will split the data of longer gestures and a large window will attach unimportant data to shorter gestures. Depending on the problem, a balance might be difficult to find, so it is worth pursuing the use of a dynamic model to avoid this problem of gesture time scale.

In this work, we propose the use of LSTM networks – a type of RNN – to find and classify gesture patterns in EMG data obtained from the forearm's muscles. This is, to the best of our knowledge, a novel application of LSTMs and RNNs in general, and presents advantages through the simplification of the data chain from the sensors to the classification model and output.

7.2 Methodology

In this section we discuss the data pipeline for the fitting and test of the model, its architecture, training methodology and performance indicators.

7.2.1 Long Short-Term Memory Networks

An RNN is an extension of FFNNs with loops in the hidden layers. This allows the model to take as input a sequence of samples and find time-relationships between them. However, they have been found to have issues in learning long-term relationships [177]. LSTM networks [178] solve this issue by adding parameters to the hidden node loops so that they can acquire and release states depending on the input sequence. Therefore, states are activated according to short-term events, while the network can keep those states active indefinitely, providing long-term memory to the network. LSTMs have been found to be better at learning sequences than classic RNNs.

The network can be generalized by equation (7.1). Input $\boldsymbol{X} \in \mathbb{R}^{t \times d}$ is a sequence of t steps and d channels. Output $\boldsymbol{Y} \in \mathbb{R}^{t \times n_c}$ is a sequence of the same length t and with dimensionality n_c. The parameters of the network are represented by Θ. This form represents a one-to-one configuration in which each time step of the input data generates a time step of the output.

$$\boldsymbol{Y} = \mathcal{L}_\Theta(\boldsymbol{X}) \tag{7.1}$$

While a regular RNN node has a single weight and bias, an LSTM network has four times that value. There are four weight/bias pairs: (1) forget gate layer, (2) input gate layer, (3) output gate layer, (4) state gate layer. The forward pass equations of an LSTM cell are described in (7.2). The input and forget gates, i_t (7.2a) and f_t (7.2b) respectively, control how much of the previous hidden state h_{t-1} and current input x_t contribute to the cell state c_t. The forget, input and output gates' activation is scaled by a sigmoid function σ and the hidden state is the output filtered with the hyperbolic tangent function *tanh*.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{7.2a}$$

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{7.2b}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{7.2c}$$

$$\tilde{c}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{7.2d}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{7.2e}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{7.2f}$$

Equations (7.2) show that the activation value of a LSTM cell requires knowledge of the preceding value in time. Therefore, the optimization of the network's parameters, through Stochastic Gradient Descent (SGD), is done on sequences of input data time steps. Typically, the input data is split into shorter windows of 200-300 time steps in order to reduce the BP time depth.

The structure of the network is similar to FFNNs. The input layer has a fixed time-length and number of variables. Fully connected nodes and LSTM layers compose the hidden layers, and lastly, there is an output layer with a *softmax* transfer

function for classification. The time-length of the input and output layers is the same, due to the sequence-to-sequence classification configuration. The state of the LSTM cells is kept between consecutive training sequences.

The network is trained with SGD with early stopping when the classification loss stops decreasing on a non-trained data set. Early stopping prevents the model from over-fitting on the training data, but we must ensure a good local solution is reaching through hyper-parameter tuning. The hyper-parameters of interest are the architecture of the network (layers and size), the learning rate, momentum, batch size and sequence length. There is no method to define these hyper-parameters, so they must be set through manual or stochastic search. However, since LSTM networks are resource intensive careful manual search is preferred. The most important parameters to manually tune are the architecture and learning rate, while the others are set to commonly used values.

7.2.2 Data Pipeline

We assume that we have a labelled dataset such that:

$$\mathcal{D} = \left\{ \left(\boldsymbol{X}^{(i)}, \boldsymbol{\iota}^{(i)} \right) \ : \quad i = 1, 2, \ldots, n_{samples} \right\} \tag{7.3}$$

where $\boldsymbol{X}^{(i)} \in \mathbb{R}^{t \times d}$ represents the sample data of d channels (variables) and t time steps, and $\boldsymbol{\iota}^{(i)} \in \mathbb{N}_0^t$ is the vector of target class' indexes for each of the sample's time step. Index (i) corresponds to the sample's index in the data set. The dataset is split into three subsets: training, validation and testing sub-sets for development and evaluation purposes.

Depending on the classifier model, features must then be extracted from the data. Generally, it is defined by $\boldsymbol{F}^{(i)} = \mathcal{F}\left(\boldsymbol{X}^{(i)}\right)$, where $\mathcal{F}$ represents the extraction function and $\boldsymbol{F} \in \mathbb{R}^{n_f}$ are the output features, which is a matrix of length equal to the number of features per sample, n_f. In this solution the features cannot be calculated with time steps from the future. Feature vector $\boldsymbol{F}_t$, at time step t, can only be determined with the sample time steps $\boldsymbol{X}_m$ where $m \leq t$.

The literature review on EMG signal processing presented on chapter 2 showed us that there is not a single best solution for feature extraction. Previous work and the review showed that the standard deviation of the EMG signals correlates well with force being exerted by the muscle, so we chose to use the standard deviation as the single feature. It is calculated in a window of w time steps ending at step i by:

$$\boldsymbol{F}_{ij} = \sqrt{\frac{1}{w-1} \sum_{k=0}^{w-1} \left(\boldsymbol{X}_{(i-k)j} - \overline{\boldsymbol{X}}_{ij} \right)^2}, \tag{7.4a}$$

$$\overline{\boldsymbol{X}}_{ij} = \frac{1}{w} \sum_{k=0}^{w-1} \boldsymbol{X}_{(i-k)j} \tag{7.4b}$$

where $j = 1, 2, \ldots n_c$ with n_c being the number of channels in the EMG data. Since a single feature is extracted, the dimensionality of $\boldsymbol{F}$ is the same as $\boldsymbol{X}$, except the

skipped time steps at the beginning of a sequence due to the features being calculated on a fixed window size. This is not an issue because most sequences begin in a rest state.

The features are then normalized, i.e., assuming the variables follow normal distributions, whose parameters are calculated in the training set. All of the subsets are normalized with these parameters and every new sample is normalized with these same parameters. Finally, the targets are one-hot encoded.

As previously mentioned, each time step in $\boldsymbol{X}_i$ has a matching target ι_i. Given that, the features are calculated on a window of frames, the choice of target label for each window is ambiguous. Therefore, we determined that it is given by the mode of the targets within that window ending at i:

$$\iota_i = \text{mode}\left(\{\iota_{i-k} : k = 0, 1, \ldots, w-2, w-1\}\right) \tag{7.5}$$

The output class sequences may be noisy due to noise in the source EMG data, user mistakes or class uncertainty in the transitions between gestures. These sources lead to small portions of a gesture, mainly in its boundaries, to be miss-classified. We propose some post-processing steps in order to prevent errors due to these miss-classifications. Firstly, gestures with length below an arbitrary value are disregarded. Then, consecutive gestures of the same class, broken by the previous filter, are merged, e.g., 1-1-2-1-1-1 is turned into 1-1-1-1-1-1.

7.2.3 Gesture Detection Criterion

A common performance indicator in continuous gesture classification is the Jaccard similarity index, also known as intersection over union. For a gesture spanning the true targets A and prediction B, it is defined by:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{7.6}$$

The co-domain is between 0 and 1, with 0 meaning that there is no overlap between the reference value and the prediction, while 1 means that there is perfect overlap. However, for gesture recognition, there is no difference whether there is 50% or perfect overlap. For this reason, we propose a different criterion for performance measurement of the classifier's output. We consider that there are four types of outcome in multi-class gesture detection:

1. True positive (TP) – good classification;
2. Miss-classification (MC);
3. False positive (FP) – gesture detected when there is none;
4. False negative (FN) – gesture not detected.

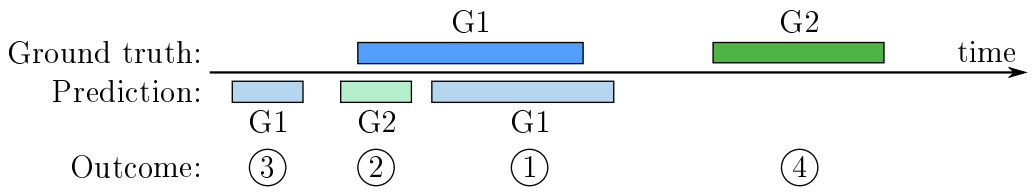


Figure 7.1: Types of possible outcomes of the classification of gesture sequences. In this example, there are two gesture classes G1 and G2 interwoven by pauses.

This list is derived from the four types of outcome in binary classification with the difference that true negatives are not considered and miss-classifications (wrong gesture detected) are added. An example of each outcome is shown in figure 7.1.

Good classifications are correctly predicted gestures that have an overlap of at least 50% with the ground truth. Miss-classifications occur when there an overlap between a predicted gesture and a ground truth gesture but its classification is wrong. False positives occur when a gesture is detected during a pause (rest state) and false negatives occur when there is no overlap between a predicted and a ground truth gesture. The final score is given by:

$$\text{score} = \frac{TP}{TP + MC + FP + FN} \tag{7.7}$$

A score of 1.0 means that no errors occurred, while a classifier that did not identify any gesture correctly.

7.3 Results and Discussion

7.3.1 Test Setup

The presented methodology was tested on the synthetic sequences of the UC2018 DualMyo data set, presented in section 3.2, a data set for EMG-based hand gesture recognition. The data acquisition methodology is described there. All of the tests use the same fixed data split and the same data samples: 60% for the training set, 20% for validation and 20% for testing. The training set is used to train the classifier, while the validation set's performance is used to optimize the hyperparameters of the classification model, i.e., neural network structure and training hyper-parameters. Finally, the test set is used to test the generalization capability of the model and is only used when the training and validation sets provide desirable metrics. Therefore, the model is not optimized for the test set and the metrics calculated on it should provide a good measurement of the model performance in other conditions.

The UC2018 DualMyo data set acquisition setup has a total of 16 EMG channels and 8 gesture classes with 110 samples each. There are a total of 95 synthetic sequences, each one composed by 8 gesture samples. Each sequence has a length of about 50 seconds, i.e., about 10000 time steps. We seek to compare the performance between a static model operating on windows of frames to a dynamic model that predicts on the time steps sequentially, such as an LSTM network.

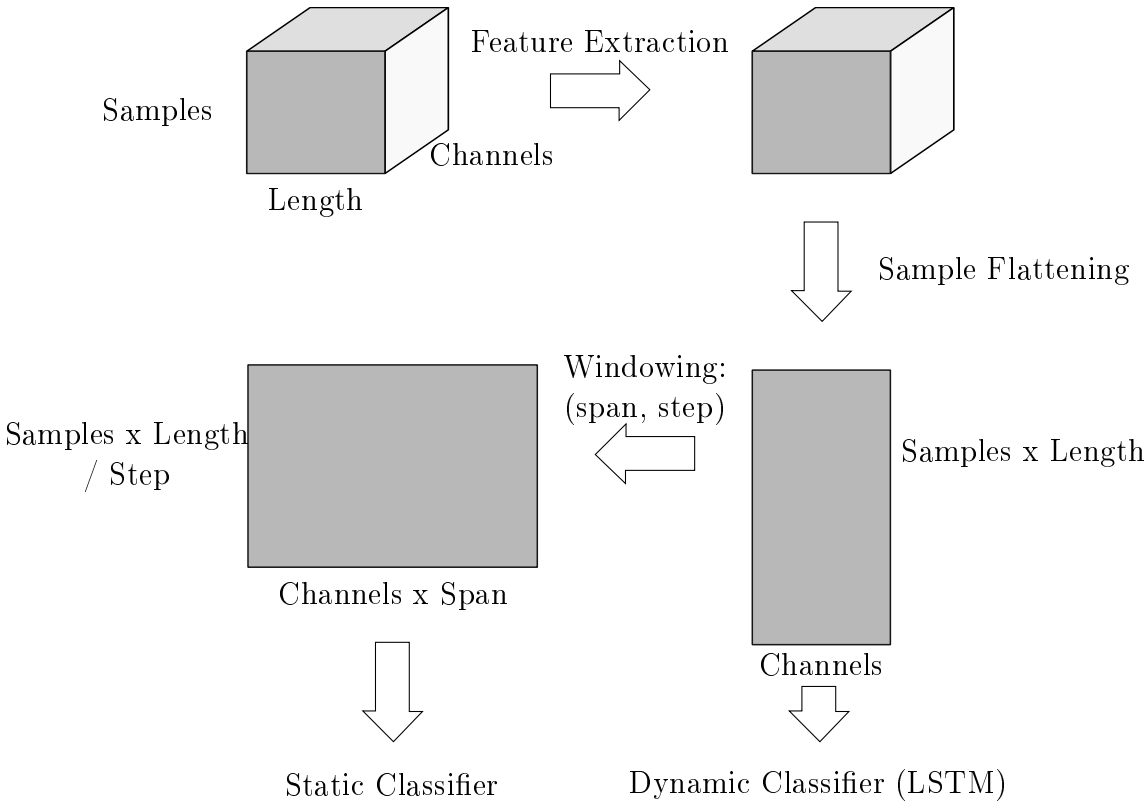


Figure 7.2: Data processing chain for the LSTM network performance analysis.

The data processing chain is slightly different for the static and dynamic classification models, as shown in figure 7.2. We start by extracting features from the 95 sequences, denominated samples. Their length is the number of time steps of the data set's sequences (10000), while the number of EMG channels is 16. The shape of the input data is maintained after feature extraction. As previously mentioned, the feature chosen is the standard deviation (σ) along time of a window of 100 time steps (0.5 seconds), for each channel. Therefore, the number of variables is maintained. The step of the window feature extraction is a single frame, in order to keep the same time scale between the raw input data and their features.

The following step in the data processing chain is concatenating every sample into the same master sequence. Every sample starts and begins during a pause, so there are no discontinuities in the master sequence. Afterwards, this sequence can be either fed directly into the LSTM or windowed again for the static classifier. In the latter case, the chosen window span was 200 frames and step 1. The span of 200 time steps is specially chosen since it corresponds to one second of data, which is about the typical minimum length of a gesture. The windows of time steps are concatenated into a single vector, which then serves as the classification predictor for that time step. The target of that time step is the mode of targets for the same time window.

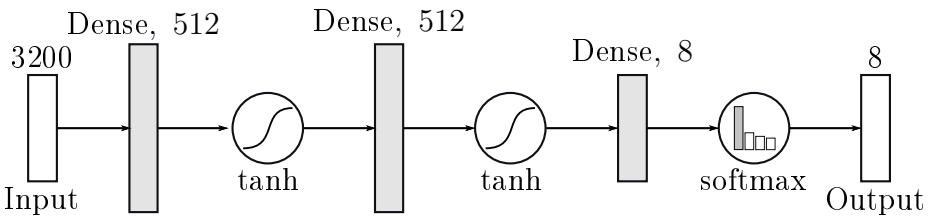


Figure 7.3: Structure of the FFNN used as the static classifier model.

Static Model

The static classifier chosen is a FFNN, which has faster training and inference times for the large amount of data, compared to other machine leaning methods. There are about 933 thousand time steps, of which 561 thousand are for used for training and the remaining are evenly split into the validation and testing sets. Since the selected features are a concatenated window of 200 time steps, the feature vector has length 3200.

The chosen model has an input layer with 3200 nodes, two fully-connected hidden layers with 512 units and a *softmax* output layer with 8 nodes, equal to the number of gesture classes in the data set. The transfer function of the hidden layers is the hyperbolic tangent. A representation of the network structure is shown in figure 7.3. This structure was determined by manual search but the output is not very sensitive to the hyper-parameters on this problem. There are almost 2 million trainable parameters, which were optimized with the ADAM variant of SGD [179], a learning rate of 0.01 and a batch size of 256. The optimization is halted with the early stopping method, to prevent over-fitting, when the model loss of the validation set stops decreasing during 12 iterations. The model is then tested on all data splits (training, validation and testing) and the following metrics are calculated:

1. Training time;
2. Inference time;
3. Frame-wise accuracy;
4. Gesture detection accuracy.

Dynamic Model

The dynamic model is the LSTM network defined in section 7.2.1. The 933 thousand time steps of the data set were split in the same way as for the static model test. However, in this model, the input features are a single frame of data from the 16 EMG channels, instead of the 3200 features the static model uses. This has great advantages for the size of the model and the total memory used.

The LSTM model, represented in figure 7.4, has 16 input nodes followed by a fully-connected (dense) layer of 400 units. Afterwards, there is a recurrent layer

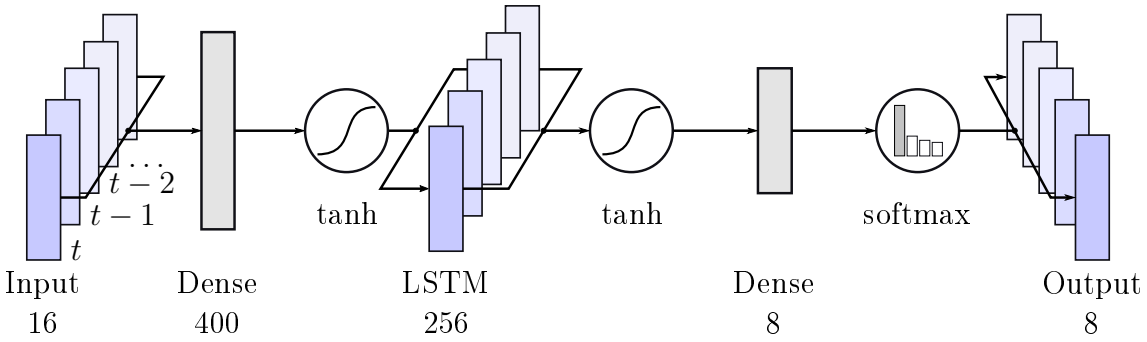


Figure 7.4: Structure of the LSTM used as the dynamic classifier model, with a one-to-one input-output classification configuration.

with 256 LSTM cells, which feeds a dense layer of 8 units. Finally, a *softmax* transfer function provides the classification output probability distribution. The dense and LSTM hidden layers have the hyperbolic tangent as their activation function. This structure was also determined by manual search, with the performance having a low sensitivity to the hyper-parameters. This structure has just 683 thousand parameters, 66% less than the static model mainly due to the low dimensionality of the input layer. The optimization of the parameters is also done with ADAM's SGD, with a learning rate of 0.001, batch size of 10 and sequence length of 200 time steps. The training process is also halted after 12 epochs without improvement of the validation loss.

7.3.2 Results

The networks were defined and trained with the Keras library (python) using the Tensorflow backend [180, 181]. The hardware used was a computer with a Nvidia GTX970M GPU (6GB VRAM), an Intel i7-6700HQ CPU and 32GB of RAM. The classification models were trained according to the methodology previously described. The training and inference processes are performed on the GPU.

The training and inference times are shown in table 7.1. The number of training time steps changes slightly between the two models due to the padding added to the end of the sequences in order to allow fixed-size batches of data to be fed into the dynamic model. The padding is typically filled by zeros and does not influence the training loss and the parameter optimization. In this setup, the training time of the dynamic model is 161 seconds, while the static model takes 284 seconds. The difference is mainly due to the discrepancy between the model's sizes, since the learning rate used to train the LSTM is an order of magnitude lower than that of the static model. For that reason, the LSTM is also faster in inference, classifying about 930 thousand time steps (about 1 hour and 20 minutes of data) in less than 5 seconds, while the static model takes 13 seconds. In any case, inference time is well below the acquisition time, but the hardware requirements are high and still need optimization for embedded and portable devices. Another disadvantage of the static model is that it requires much more memory during training, since each time

Table 7.1: Training and inference times for the static and dynamic models, tested on the UC2018 DualMyo data set.

Model	Training Steps	Training Time	Testing Steps	Testing Time	Frames/ second
Static	561,408	283.6	933,120	13.0	71,778.5
Dynamic	556,000	161.0	928,000	4.9	189,387.8

Table 7.2: Classification accuracies of the static model tested on the UC2018 DualMyo synthetic sequences.

Data Split	Framewise Accuracy	Detection Accuracy	Outcomes			
			Type 1	Type 2	Type 3	Type 4
Train	97.16	99.10	453	2	0	1
Val	92.67	91.10	151	4	10	0
Test	96.59	99.34	151	0	0	1

Table 7.3: Classification accuracies of the dynamic (LSTM) model tested on the UC2018 DualMyo synthetic sequences.

Data Split	Framewise Accuracy	Detection Accuracy	Outcomes			
			Type 1	Type 2	Type 3	Type 4
Train	97.18	98.68	455	1	0	0
Val	95.85	99.24	152	0	0	0
Test	96.68	99.67	152	0	0	0

step requires 200 frames to be stored. This is a significant constraint for this data set and larger ones.

The accuracy of the models are presented in tables 7.2 and 7.3, for the static and dynamic models, respectively. We present the frame-wise accuracy (proportion of correctly classified time steps) of the models in the 3 data splits. We also present the detection accuracy, as defined in (7.7). Additionally, we show what types of detection errors that occurred in each data split: (1) true positives, (2) miss-classifications, (3) false positives, and (4) false negatives.

In terms of frame-wise accuracy, both models present a good classification accuracy (97%) on the training and test sets, while the validation performance is slightly lower, particularly for the static model, which presents 93% and 96% for the static and dynamic models, respectively. The discrepancy between data splits is due to particularities in the sequences of each split rather than over-fitting, since the performance on the testing set is close to that of the training set.

While the frame-wise classification accuracy is an indicator of overall performance of the model, it does not tell us if there are miss-classifications in the middle of

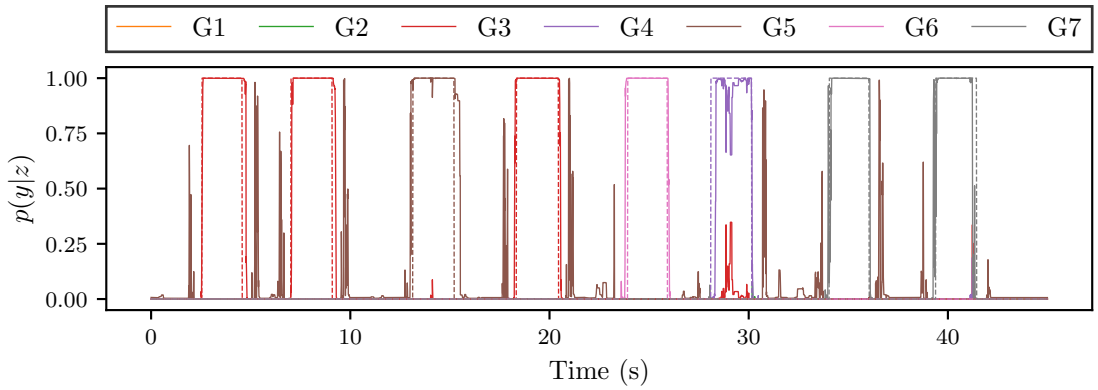


Figure 7.5: Raw output of the static model on the first sequence of the testing set and the corresponding targets in dashed lines.

gestures, causing the model to momentarily and wrongly announce a poor gesture classification. The detection accuracy measurements indicate whether gestures as a whole are correctly identified, or if the model is finding gestures when it should not (false positives). The gesture detection accuracy was determined on the post-processed output of the models. Gestures below a specific length (0.5 seconds) are disregarded in order to reduce the classification noise of the models. The detection accuracy was generally better than the frame-wise accuracy, tables 7.2 and 7.3. It exceed 99% in all data splits, for the static model, except in the validation split, where it reached just 91%. The dynamic model performed better, with over 98% detection accuracy in all cases. In this case, the validation and test splits presented no errors, but the truth/prediction overlap was lower than 50% in a few gesture samples.

Most of the errors occurred with the static model on the validation split. In this case, there were 4 outcomes of type 2 (miss-classifications), and 10 of type 3 (false positives). Outcomes of type 2 are the result of a gesture being totally or partially misclassified. Examples of type 3 outcomes are visible in figure 7.5, where often times gesture 5 is detected in the transitions between the gesture and the surrounding rest states. This gesture (double-tap) has a relatively weak signal compared to the others, so it is harder to differentiate from the rest state. During the transition between rest and a gesture, the feature value increases or decreases slowly since it is the standard deviation of the signal in a window of frames. Therefore, there are some frames during the transition whose features can be similar to gesture 5. Despite these errors, gesture 5 is correctly classified when it appears isolated. In figure 7.6, the raw signal output of the LSTM model is shown in the same conditions and for the same sequence. There is considerably less classification noise and less false positives.

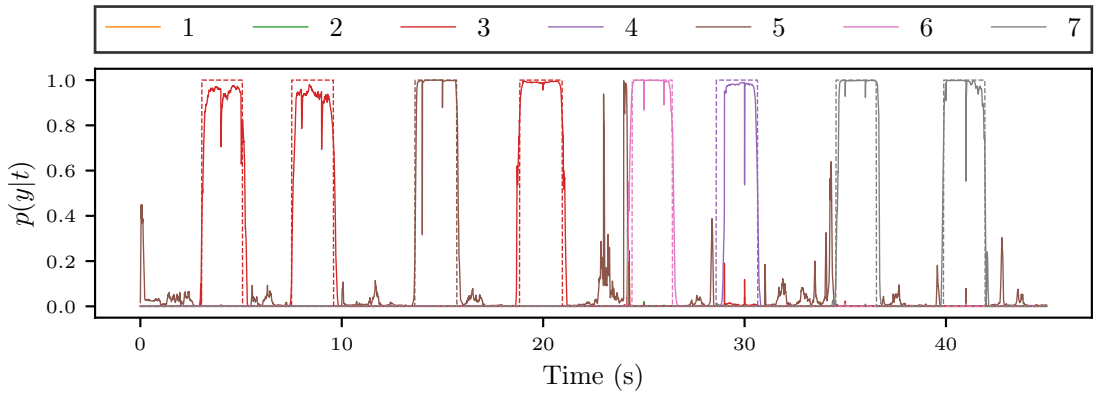


Figure 7.6: Raw output of the LSTM model on the first sequence of the testing set and the corresponding targets in dashed lines.

7.4 Conclusion

In this work we proposed the use of an LSTM neural network to improve the online classification of hand gestures with EMG signals acquired from the forearm muscles. A FFNN model was defined and trained with windows of data, in order to establish a comparison baseline. Additionally, an alternative performance index, the gesture detection accuracy, is proposed to evaluate the performance of the model during online classification.

The methodology was evaluated on the UC2018 DualMyo data set. Both the baseline and LSTM approaches achieved above 96% accuracy in all data splits but one. The proposed detection accuracy criterion was higher than the regular accuracy due to the decrease of the overlap ratio (truth and prediction) requirement from 100% to 50%. The detection accuracy was around 99% in all cases, except on the validation split of the static model. In this case, the LSTM performed better than the FFNN (99% vs 91%, respectively). Most errors originated from scenarios where gesture 5 was detected during transition periods, and the miss-classification errors were marginal. Although both models had similar accuracies, the LSTM model has significantly less parameters (1.9 vs 0.7 million), thus also having smaller training and inference times. Therefore, the dynamic model appears to be better suited to online and portable recognition systems.

In the future, further optimization should be done in both models in order to decrease the number of parameters to make it better suited to embedded systems. Additionally, the methodology should be validated in larger data sets, which should also use other types of signals.

Chapter 8

Improving Outlier Detection with Generative Adversarial Networks on Gesture Data

Abstract

A major issue in gesture detection is the limited scope of a data set, when compared to the real problem. Independently of the resources available, a data set does not normally include a large fraction of real-world scenarios. In those cases, there is no way of confidently predicting the performance of a classifier. In this chapter, we propose a novel way of solving the issue of classification of out-of-vocabulary gestures. Very often, these gestures are classified as an existing class and are difficult to remove with a classification threshold. This solution uses a generative model (GAN) to augment the data set with new samples, and noisy labels.

The approach was evaluated on the UC2018 DualMyo data set with 7 trained classes and 1 non-trained class. A baseline test was established for the generative and classification models. In terms of sample generation quality, the GAN is significantly better than a random distribution (noise) in mean distance, for all classes. However, the dispersion within samples of the same class, is generally significantly lower than that of the data set, even though the GAN has not collapsed into generating a single type of sample. In the classification tests, the baseline neural network was not capable of detecting any untrained gestures. When the proposed methodology was introduced, we found that there is a trade-off between the detection of trained and untrained gestures, with some trained samples being mistaken as untrained. Nevertheless, an outlier detection rate of 95.4% was achieved with just 5% loss of trained samples.

Keywords: Collaborative Robotics, Semi-Supervised Learning, Generative Adversarial Networks

8.1 Introduction

Often-times the performance of a classifier trained offline on a data set is not indicative of the online performance. This may happen due to missing elements on the data processing pipeline, such as proper data scaling. However, the major issue is the limited scope of a data set, when compared to the real problem. Independently of the resources available, a data set does not normally include a large fraction of real-world scenarios. In those cases, there is no way of confidently predicting the performance of a classifier.

In a gesture recognition data set, there are training patterns of a predefined number of gesture classes. It is also possible to include gesture patterns that do not match any of the classes but that can occur in real-world conditions [122]. These would belong to the class *others*. Hereinafter, we will call these non-gestures *others*, i.e., gestures that do not belong to the predefined classes. However, the diversity of non-gestures is almost infinite, or at least, much greater than that of the predefined gestures.

Non-gestures appear in real-world conditions on every type of interaction, particularly in human-human and human-machine interaction. They may occur due to, albeit not limited to the following reasons:

1. The user is uneducated in respect to the human-machine interface and performs out of vocabulary gestures;
2. The user is distracted and is moving in a way that does not make sense in the context of the interface, e.g., talking to somebody else;
3. The user is forced to move in response to other elements in the surroundings, e.g., moving machines or falling objects;
4. The user is from the end of an interaction to the start of the next, i.e., ME.

The naive way to exclude non-gestures is to set a threshold to the output probability of a classifier:

$$\text{class} = \begin{cases} \tau, & p(y_\tau|z) \geq \text{threshold} \\ \text{none}, & p(y_\tau|z) < \text{threshold} \end{cases} \tag{8.1}$$

where y is the classifier's output given the feature vector z, and p the probability distribution over the problem's classes. We have previously shown that for the most widely used classifier – ANNs –, the output probability is not a good measure of classification certainty [8]. This means that many correct classifications may have low probabilities and incorrect ones have high likelihood, as predicted by the ANN. If we were to set a threshold on the class probability using 8.1, many good classifications would be discarded, while an equal proportion of bad classifications would pass. Therefore, the threshold method is not effective for its purpose.

It is possible to exclude non-gestures and bad classification using context clues, such as limiting the quantity of possible outputs. However, this is a limited approach

and we are interested in exploring new methodologies that may help a classifier discriminate non-gestures.

A non-gesture is an "abnormal" occurrence for the classification model, which is trained with a restricted number of classes. While non-gestures could belong to an extra class containing all of the possible non-gestures, training it is a challenge because of the lack of examples. Because of the large gesture domain, it is unfeasible to get data samples from every possible non-gesture. This problem is seldom addressed and most data sets do not include such patterns.

The problem of detecting "abnormal" patterns from a predefined number of gesture classes is generally known in the literature as ND. In this type of problem, the predefined classes have considerably more training examples, while the "abnormal" patterns are under-represented. In [182], the authors concluded that currently, there is no optimal solution for the ND problem because if depends on the type of data and the application domain. Distance methods such as k-NN have been shown to be superior, but their computational efficiency decreases with data set size, therefore making them unsuited for larger data sets and real-time applications [183].

We propose the use of a semi-supervised methodology which uses the labelled samples of a data set and generated unlabelled samples which correspond to either gestures or non-gestures. This is an approach that is currently used in deep learning where very large data sets are required but labels are not always available [184, 185]. This is a methodology that has been used successfully on data sets for image recognition. In this chapter, we present the results of its application to hand-gesture recognition with GANs.

8.1.1 Generative Adversarial Networks

The name GAN describes a framework for the training of generative neural networks that was introduced by Goodfellow et *al.* in [186]. In this framework, there are two competing nets which are trained simultaneously: a generative net G and a discriminative net D. The objective of the discriminator D is calculating the probability that a sample came from the real data set rather than from the generator G. On the other hand, G is trained to produce samples that maximize the probability of D classifying them as real. D and G have competing objectives and should normally improve one another. A diagram representing a variant of this framework is shown in figure 8.1.

Current applications of GANs are essentially in the field of image to image translation, i.e., generation of new images with specified features or increased resolution [187, 188, 189, 190, 191]. Very few authors have studied applications on other domains, such as speech [192, 162] and text generation [193]. The flexibility of GANs gave rise to a plethora of network structures and training methods, of which some notable ones are: Auxiliary Conditional Generative Adversarial Network (AC-GAN) [194], Cycle-Consistent GAN (CycleGAN) [195] and Wasserstein GAN [196]. In terms of performance, it is difficult to evaluate these models quantitatively, since these are generative models and they are purpose built. Nevertheless, the generated

outputs are often indistinguishable from real data in state of the art implementations.

The original GANs [186] had a discriminator D whose output was the binary classification of the source of a sample (real or generated). If the original data set was also divided in N classes, the D would also be able to classify generated data, thus appearing an extension to semi-supervised learning [194]. These are the auxiliary conditional GANs.

For the evaluation of the generated samples, an intra-class diversity measure was proposed in [194], specifically the multi-scale structural similarity (MS-SSIM). This metric aims to account only for the same features as a human would perceive, rather than calculate a pixel to pixel distance. Thus, it is more indicated for image similarity. Lack of similarity is a symptom of an important failure mode during GAN training. It happens when the generator collapses and generates a single pattern that maximally confuses the discriminator. A generator model that only outputs a single pattern is very limited and not useful, so we should avoid a collapsed generator.

In this chapter, we propose small modifications in the structure the AC-GAN:

1. A softmax layer as the second output of the discriminator;
2. A one-hot encoded second input in the generator instead of the class number;
3. Training with noisy generated labels.

These changes aim to allow the use of the discriminator as an online classifier and the generation of samples with any given class likelihood.

8.2 Methods and Methodology

In this section we discuss the data pipeline for the fitting and test of the model, the architecture of the model, its training methodology and the definition of the tests.

8.2.1 Data Pipeline

We assume that we have available a labelled data set, which is defined as:

$$\mathcal{D} = \left\{ \left(\boldsymbol{X}^{(i)}, \iota^{(i)} \right) \ : \quad i = 1, 2, \ldots, n_{samples} \right\} \tag{8.2}$$

in which $\boldsymbol{X} \subset \mathbb{M}^{t \times d}$ represents the sample data of d channels (variables) and t time steps, and $\iota^{(i)} \in \mathbb{N}_0$ is the target class for that sample. For development and testing purposes, the data set is split into three subsets: training, validation and testing subsets.

The following step is feature extraction, which depends on the data set, classifier model and objectives. Generally speaking, it is defined by $\boldsymbol{f}^{(i)} = \mathcal{F}\left(\boldsymbol{X}^{(i)}\right)$, where $\mathcal{F}$ represents the extraction function and $\boldsymbol{f} \subset \mathbb{M}^{n_f}$ is the output features, which is a vector of length equal to the number of features per sample, n_f.

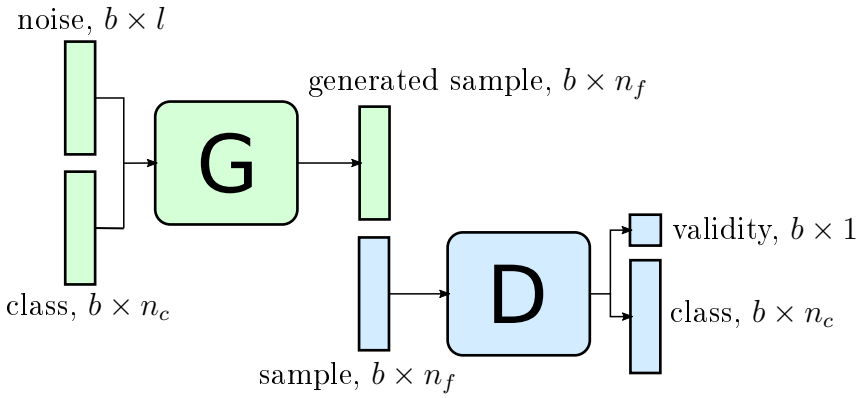


Figure 8.1: Diagram representing the custom AC-GAN framework. The generator G has two inputs: noise with a latent size l and class, a one-hot encoded vector for n_c classes. Its output is the generated sample, a vector with n_f variables – the same as the number of features of a real sample. The discriminator D takes as inputs a sample, real or generated, and determines a validity scalar for binary classification of the sample's source. Its second output is a vector with the classification of the sample. b corresponds to the batch size for the gradient descent optimization.

The features are then normalized, i.e., assuming the variables follow normal distributions, whose parameters are calculated in the training set. All of the subsets are normalized with these parameters and every new sample is normalized with these same parameters. Finally, the targets are one-hot encoded.

8.2.2 Custom-Built Generative Adversarial Networks Model

The structure of the custom GAN is similar to the AC-GAN, figure 8.1. There are still separated generator and discriminator networks, like in the vanilla GAN [186]. The first input of the generator G is a noise vector z with latent size l that is sampled from a normal distribution $\mathcal{N} \sim (\mu = 0, \sigma^2 = 1)$. The second input of G is a one-hot vector that represents the class to be generated. The generator network's structure is relaxed, but it must be a feed-forward neural network. The structure is highly dependent on the available data and the training hyperparameters.

The discriminator D takes samples as a vector of length n_f (number of features), which can either come from a data set or the generator. These serve as input for the discriminator network, whose structure is also free, depending on the type of data. D has two outputs, being the first the validity of the input sample. The validity is a scalar $v \in [0, 1] \subset \mathbb{R}$, i.e., a real number between 0 and 1. Validity below 0.5 denotes a generated sample and if it is above or equal to 0.5, it corresponds to a real sample. The discriminator has second output, which is the classification of the sample as a one-hot vector.

The notation used in this study is now explained. The data set samples are represented by a 3-element tuple x_r, t_r, s_r, which are the sample data, the target class and its source, respectively. These are also called real samples, as opposed

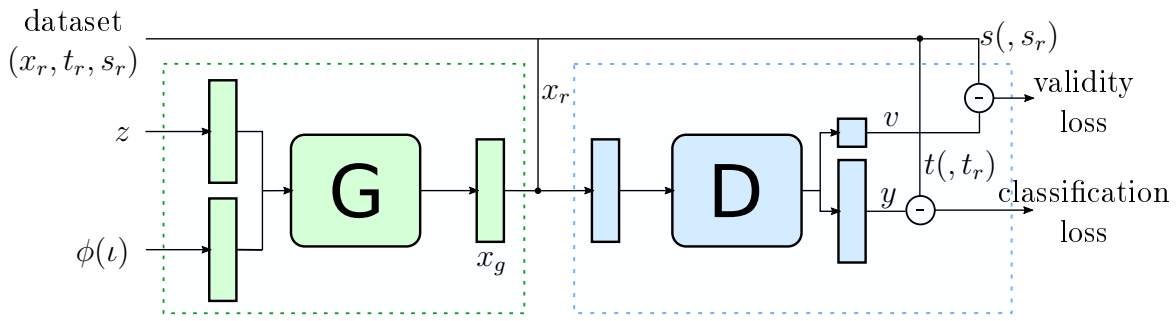


Figure 8.2: Diagram representing the training process of the custom GAN.

to generated samples, known as x_g, t_g, s_g. x_r and x_g have the same shape as $\boldsymbol{f}$ in section 8.2.1 and represent the features obtained from the sample. The targets t_r and t_g, for real and generated samples respectively, are one-hot encoded vectors of the classes they represent. Assuming the classification problem has n_c classes, the one-hot vector t for class ι is a horizontal vector of size $1 \times n_c$ composed of zeros, except $t_\iota = 1$. The source scalar, s, is either 1 for real samples, or 0 for generated samples.

8.2.3 Model Training

The two networks are trained by SGD simultaneously, in two interwoven stages (D-G-D-G-D-G ...) for a number of epochs. A diagram that applies to both stages is shown in figure 8.2. In SGD, the weights of the model are updated according to the gradient of the model's loss on a batch of b samples. This means that instead of vectors, the inputs and outputs of the networks are matrices. In these, the first dimension corresponds to the sample and the second to the variables (features, target index, among others).

First stage: discriminator

The discriminator is trained with both real and generated samples. Given a pre-selected batch size b, $b/2$ samples are extracted from the data set, being denoted by x_r. An equal amount of samples is generated from G, x_g. These samples are generated by running G with two inputs:

$$x_g = G(z, \phi(\iota)) \tag{8.3}$$

The first input is the noise matrix z, which has the following definition:

$$z = \{z_{ij} \leftarrow \mathcal{N}(0,1),\ \forall i \in [1,n],\ j \in [1,l],\ i,j \in \mathbb{N}\} \tag{8.4}$$

where n is the number of samples to be generated (in this case $b/2$) and l is the latent dimension of the generator. Basically, a noise matrix is sampled from the normal distribution.

The second input are the one-hot vectors of the classes ι of the samples to be generated, which are sampled from a discrete uniform distribution:

$$\iota = \{\iota_i \leftarrow \mathcal{U}\{1, n_c\},\ \forall i \in [1, n] \subset \mathbb{N}\} \tag{8.5}$$

where n_c is the total number of classes in the problem and n is the number of samples. The one-hot encoded input is $\phi(\iota)$. For the noisy labels, the target vector is a vector where the element $\boldsymbol{t}_{k=\iota}$ has a certain value p' between 0 and 1, while the other elements sum up to 1:

$$\boldsymbol{t}_k = \begin{cases} p', & k = \iota \\ \dfrac{1 - p'}{n_c - 1}, & k \neq \iota \end{cases} \tag{8.6}$$

Until now we have defined all the data required to train the discriminator. There are two analogue tuples of data: (x_r, t_r, s_r) and (x_g, t_g, s_g). The samples x_r and x_g are fed into the discriminator D:

$$(v, y) = D(x) \tag{8.7}$$

There are now two losses to be calculated, as seen at the end of figure 8.2: the validity and classification loss. The validity is a binary classification problem, thus the loss function chosen is the binary cross-entropy:

$$L_1 = -(s \log(v) + (1 - s) \log(1 - v)) \tag{8.8}$$

where L_1 is the validity loss on a sample, s is the value of the source of the sample (0 or 1) and v is the predicted probability of the sample belonging to the correct class, which is the first output of D.

The actual classification of a sample is a multi-class problem, so the multi-class cross-entropy is used:

$$L_2 = -\sum_{c=1}^{n_c} t_{i,c} \log(y_{i,c}) \tag{8.9}$$

where L_2 is the classification loss of a sample i, $t_{i,c}$ is the value of element c of the sample's target t, and $y_{i,c}$ is the output of D for the same sample.

The final loss is a weighted average of L_1 and L_2. Finally, the discriminator's weights are updated.

Second stage: generator

While the discriminator is trained with both real and generated samples, the generator is trained without real samples. The process is the same as shown in figure 8.2, but the data set is not used.

The noise z and indexes ι are sampled according to (8.4) and (8.5). Analogously to discriminator training, a batch of b samples x_g is obtained from the generator so that $x_g = G(z_g, \phi(\iota_g))$. We then calculate the validity (8.8) and classification (8.9) losses with the discriminator. Hence, the loss function of the discriminator is also used with the generator. However, the discriminator weights are frozen during this stage, so only the generator's weights are updated to minimize these losses.

8.2.4 Classification Decision

A trained discriminator can be used to classify new samples. However, the output class provided by the discriminator is rarely taken as the final classification. There is often a decision method that provides the final classification, possibly context-based information, such as data from other sensors.

Most of the neural network classifier models have a final layer which is a softmax transfer function. This function provides a probability distribution over the possible classes. This is the second output of D shown in (8.3). The probability of a given sample x belonging to class i is given by:

$$\boldsymbol{y}_i = p\left(i \mid \boldsymbol{x}\right), \quad \text{for } i = 1, 2, ..., n_c \tag{8.10}$$

where n_c is the number of possible classes.

Given the probability distribution $\boldsymbol{y}$, it makes sense to set a threshold τ on $\boldsymbol{y}_i$ so that if its value drops below a pre-defined value, the output class is disregarded as *others*:

$$\text{output class} = \begin{cases} \text{class } i, & \max\ \boldsymbol{y} \geq \tau \\ \text{others}, & \max\ \boldsymbol{y} < \tau \end{cases} \tag{8.11}$$

This definition introduces the problem of determining an adequate value for the threshold.

The use of this method results in more false negatives than using no threshold (or $\tau = 0$). As a consequence, increasing the threshold yields lower recall of the classes i. Therefore, it is possible define a threshold value such that the recall does not decrease further than, e.g., 5 or 10%. In our view, false positives are worse than false negatives in most applications, the exception being in critical conditions such as a request for an emergency stop. These cases are rare and should be specially handled to increase safety.

8.3 Test Definition

This section describes the test methodologies followed in this chapter. We are interested in evaluating the performance of both the generator and the discriminator. The generator should find patterns in the data set samples and create new samples based on those patterns.

8.3.1 Generator Performance

Firstly, we tested the quality of the samples created by the GAN's generator. The quality is determined by the similarity between generated and real samples. There are plenty of similarity measures that can be used, but since we have small feature vectors, we opted by simply using the L2 distance between samples. The concept of distance is opposite to similarity, thus he distance must be minimized to improve similarity.

Formally, we are interested in knowing the distance between two sets of samples, $\boldsymbol{X}$ and $\boldsymbol{Y}$. These sets of data are matrices of shape $(n_{samples} \times n_f)$, where $n_{samples}$ may or may not be the same. The L2 distance between the i-th sample of $\boldsymbol{Y}$ and $\boldsymbol{X}$ is thus given by:

$$\boldsymbol{l}_i = \frac{1}{N} \sum_{j=1}^{N} \sqrt{\sum_{k=1}^{n_f} (\boldsymbol{X}_{jk} - \boldsymbol{Y}_{ik})^2} \tag{8.12}$$

where N is the number of samples in $\boldsymbol{X}$ and n_f the number of features. In short, the distance between a sample $\boldsymbol{Y}_i$ and the set $\boldsymbol{X}$ is the mean L2 distance between $\boldsymbol{Y}_i$ and all of the samples of $\boldsymbol{X}$.

The following tests are presented:

1. **Data set baseline distance:**
   Mean distance between data set samples of the same class.
   Standard deviation of the distance between data set samples of the same class.

2. **Generated data distance:**
   Mean distance between real and generated samples of the same class.
   Standard deviation of the distance between real and generated samples of the same class.

3. **Gaussian noise distance:**
   Mean distance between real and random noise generated from Gaussian distributions.
   Intra-class standard deviation of the distance between real and noise samples.

Two baseline distances are established: a data set and a random noise baseline. The data set baseline establishes the ideal distance, i.e., the dispersion of the real data set. The generated data distance measures how far the generated data are from the real data, which should tend to the ideal metric when these mimic perfectly the real data. The Gaussian noise distance is the worst-case scenario that would occur if the generator were to diverge from the real data. In all cases, the distance metric is computed for sets of samples within the same class. The randomly generated data are sampled from Gaussian distributions with mean and standard deviations calculated from the real data, for each class individually.

8.3.2 Discriminator Performance

The objective of the presented methodology is to improve the real-world performance of the discriminator. The performance is strongly tied to the accuracy of the classifier model, i.e., the number of successful classifications over the total number of gesture samples. However, the accuracy does not reflect the rate of non-gestures classified as gestures, which may happen in real-world conditions. The metrics of interest for this type of problems are the accuracy, precision and recall of the predictions.

We propose the analysis of the test results as a two-step problem. The first step is determining whether a sample belongs to one of the trained classes or not, which is a binary classification problem. In this case, the positives are when a prediction belongs to the trained classes group, and the negative cases correspond to other unknown classes. The second is the multi-class classification problem, where a prediction is performed to find what the class of a sample is, within the trained classes set. The metrics used are the classification accuracy, as well as precision and recall. They are described by the following equations:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{8.13a}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8.13b}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8.13c}$$

where TP, FP and FN stand for true positive, false positive and false negative, respectively. In the multi-class problem, the TPs are considered to be the cases when a pattern of class τ is classified as such and FPs occur when τ is classified as any other class.

The working hypothesis is that the data set augmentation with the previously described GANs improves the classification of gestures and non-gestures. The baseline performance is established by training a discriminator model with the data set samples. Following that, since we found that presence of noisy labels on the data set helps training the GAN, we test their specific contribution without using the GAN framework. Finally, we test the performance of the discriminator trained within the GAN framework and retrained with real and generated samples:

Baseline: Discriminator trained regularly

Test 1: Discriminator trained with noisy labels

Test 2: Trained GAN discriminator

Test 3: Trained GAN discriminator retrained with real and generated samples

Additionally, these tests are repeated without the threshold defined in (8.11), and with the threshold tuned so that the average accuracy over the original classes is 95% and 90%.

8.4 Results

8.4.1 Validation Split

All of the tests use the same data split and the same data samples: 60% for the training set, 20% for validation and 20% for testing. The split was fixed at the beginning of data analysis, so that it is not optimized for proposed methodology. We consider this hold-out split to be better than a k-fold cross-validation split in regards to deployment-oriented analysis. On one hand, a k-fold method requires the classifier to be trained k times, thus having a training time roughly k times larger than a hold-out split. On the other hand, GANs are remarkably difficult to train and it is highly unlikely that the same hyper-parameter set will allow the GAN training process to converge.

The training set is used to train the classifier and does not include any sample of non-gestures. The purpose of the validation set is to optimize the hyperparameters of the classification model, i.e., neural network structure, added noise, normalization, among others. Additionally, the model fitting process is controlled by the model loss that is calculated online on the validation set, in order to prevent over-fitting on the training data. This set does not include any non-gesture sample, so that there is no leakage of these data into the fitting process. Finally, the test set is used to test the generalization capability of the model and is only used when the training and validation sets provide desirable metrics. Therefore, the model is not optimized for the test set and the metrics calculated on it should provide a good measurement of the model performance in other conditions.

8.4.2 UC2018 DualMyo Data set

The UC2018 DualMyo data set comprises 8 classes of patterns with 110 repetitions each. For these tests, class 7 (see figure 3.5) was set aside to become the class *others*, or non-gestures. This means that the classifier is trained on classes 0 through 6, and tested on all 8 classes. This class was selected because it is not trivially separated from the others in an unsupervised manner, as can be seen in the larger cluster of figure 3.8. This is proven by the baseline test.

Each data sample is a matrix $\boldsymbol{X} \in \mathbb{M}^{t \times d}$, where the length t is 200 frames and the second dimension d consists of the 16 EMG channels. The feature extraction function chosen $\mathcal{F}$ is the standard deviation of the sample along time, i.e., one standard deviation per channel. Therefore, the feature vector extracted from each sample $\boldsymbol{f}^{(i)}$ is a vector of length equal to the number of channels. This feature is often proportional to muscle contraction strength, thus providing a muscle activation map around the forearm. calc The 60/20/20 split results in 462 samples being used for training, 154 for hyper-parameter validation and 155 for testing. Additionally, the 110 samples of class 7 constitute a second test split to evaluate the performance of the nets on the class *others*.

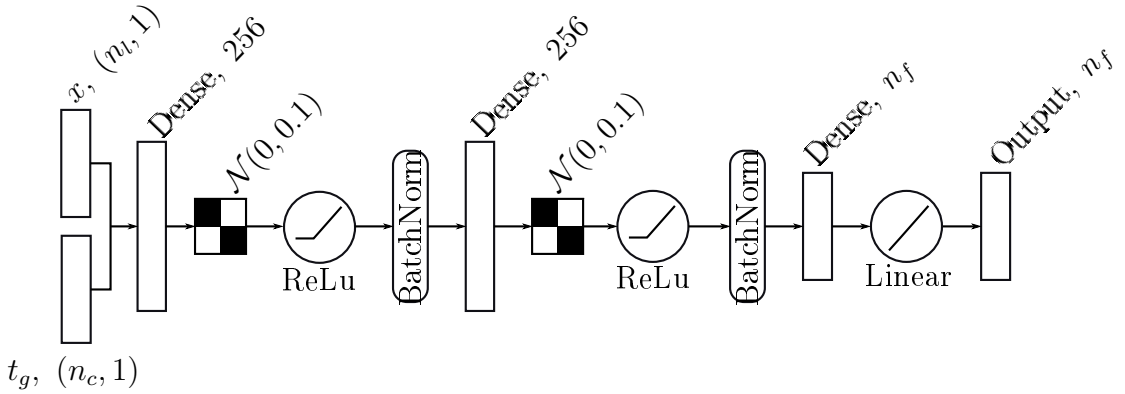


Figure 8.3: Structure of the generator used on the UC2018 DualMyo data set for the evaluation of the GAN methodology.

GAN Structure

The discriminator and generator networks may take many different shapes, under the presented framework. Furthermore, there is no method to initialize a network structure for a given problem. Generally, it depends on the size of the data set, type and quality of the data and number of features, among others. The initial structure is found by random grid search on a varying number of layers and respective nodes in large steps. The network's performance is evaluated on the validation set. When a good structure is found, it is then optimized by manually fine-tuning the number of nodes, transfer functions and applying generalization aids.

The structure of the generator is shown in figure 8.3. The depth of the network was kept at 2 fully-connected (dense) hidden layers with 256 nodes each. There are two inputs, the first being the noise vector x with a latent size of $n_l = 8$. The second input is the noisy label vector. These are fully-connected to the first dense layer. Gaussian noise from a distribution $\mathcal{N}(0, 0.1)$ is added to the dense layer's activations, followed by a rectified linear unit (ReLU). The ReLU activations are then normalized in the training batch, i.e., centred and scaled. These 4 elements form a block which is repeated once again. Finally, the output is reduced to a vector of length $n_f = 16$ (one feature per data channel) by a dense layer of that same size, after which a linear transfer function provides the network output. The length n_f corresponds to the number of features of a data set sample, so the output is equivalent to the features of a generated sample.

The introduction the the Gaussian noise layers is typically recommended in generator networks and helps increase the variance of the generator samples. A failure mode of GANs is the mode collapse, in which the generator learns and outputs an unique sample. Adding noise during the training process, whether through noise layers or noisy labels, helps prevent this issue.

The discriminator is shown in figure 8.4. It has a single input, which can be either generated or real samples, therefore having length n_f. A Gaussian noise layer is added next, with a standard deviation of 0.4 and mean 0. Following that, there is a

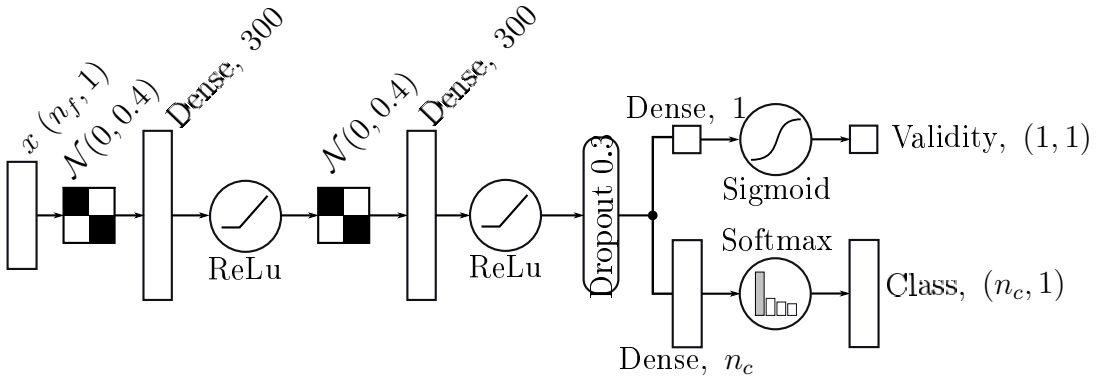


Figure 8.4: Structure of the discriminator used on the UC2018 DualMyo data set for the evaluation of the GAN methodology.

dense layer with 300 nodes and a ReLU as activation function. This 3 element block is repeated one more time with the same parameters. To aid generalization, there is a dropout layer next, where 30% of randomly chosen connections are dropped in each training iteration. Afterwards, the networks splits into two branches corresponding to its two outputs. The first output, validity, has a single fully-connected node, whose activation is then transformed by the sigmoid function is order to output a value between 0 and 1. The second output is also fully-connected with $n_c = 8$ nodes, which corresponds to the number of classes of the problem. Afterwards, a softmax activation function is applied in order to output a distribution of probabilities over the number of classes.

Training Parameters

In the methodology section we mentioned that the discriminator D and generator G are trained separately, one after the other, since they are different networks. Therefore, they may have distinct training parameters. The success of a GAN framework is strongly dependant of these parameters. The learning process must be balanced so that one does not learn much faster than the other, causing the mode collapse failure mode. The failure mode occurs when the generator is trained to a point where it always outputs the same value, independently of the input, thus bringing the learning process to a halt.

The networks are balanced through the tuning of the learning rates of D and G, and the relative weights of the two losses of the generator. As a reminder, the generator is trained with the losses calculated on the discriminator. These are the validity, (8.8), and classification losses, (8.9). If the classification losses decrease faster than the validity, the generator will focus on generating n_c different classes that are more easily separable, not necessarily resembling the original data. Therefore, we increase the validity loss weight, so that G learns to generate samples closer to the real data, rather than more separable data.

Other relevant parameters include the training momentum, number of epochs,

the latent dimension of G, batch size and label noise strength. Additionally, there are the learning rates for D and G, and the loss weights of D. There are some recommended values for these parameters, but since there is no established optimization process, they were optimized by trial and error as a function of the network losses and sample visualization.

The networks were trained for a fixed amount of 300 epochs, while the batch size was kept at 32 samples. The G latent size was set to 8, which is half of the feature vector dimensionality. The limits of the label noise were set to $[0.9, 1.0]$. In early trials, it was set to $[0.8, 1.0]$, but the lower bound of 0.9 achieved better results in combination with the remaining parameters.

The SGD optimization algorithm followed the Adam update rule [179]. The learning rates were set firstly to the typical value of 0.0002 for both networks, but the rate for G ended up being incrementally increased to 0.001, in order to speed up the generator training. The momentum was kept at 0.5 for both networks. The weight decay was set to 10^{-7} and 10^{-6} to D and G, respectively. Finally, the G loss weights were set to 1.3 for the validity loss and 0.8 for the classification loss. The training time for this setup is about 63 seconds in a Tensorflow-based Keras implementation. The nets are trained on a Nvidia GTX970M GPU with 6GB of memory.

We show the GAN losses on figure 8.5. The discriminator and generator losses are very close, which is expected since the loss functions are the same, despite the G loss being calculated from generated samples, while the D loss is obtained from equal parts of generated and real samples. Both losses are still decreasing by epoch 300. However, looking at the generator loss components individually, we see that the classification loss is decreasing, but the validity loss has plateaued. This means that the network is improving the separation of the generated samples but their similarity to the real samples is not improving. The slight peaks found every 50 epochs are likely to be numerical errors caused by sampling G at the checkpoints that occur at the same frequency.

Generator Performance

Examples of generated and real samples for each gesture class are shown in figure 8.6, where the lowest signal is represented in dark blue and the highest in yellow. A first look shows that generally, the intensity of low state signals is higher in generated samples than in real ones. Nevertheless, that is not an issue since there is still a visible gap between low and high signal states. The last class, G7 or *others*, is a class created by the generator that was not trained, therefore there is no equivalent in real samples.

While visualizing the samples provides a subjective evaluation of the quality of the generated samples, the first generator test is comprised by a similarity measurement between data set samples, generated samples and samples drawn from a Gaussian distribution. The results are shown in table 8.1. The table shows that the GAN is significantly better than the random distribution (noise) in mean distance, for all classes. However, the standard deviation of the distance, which measures the

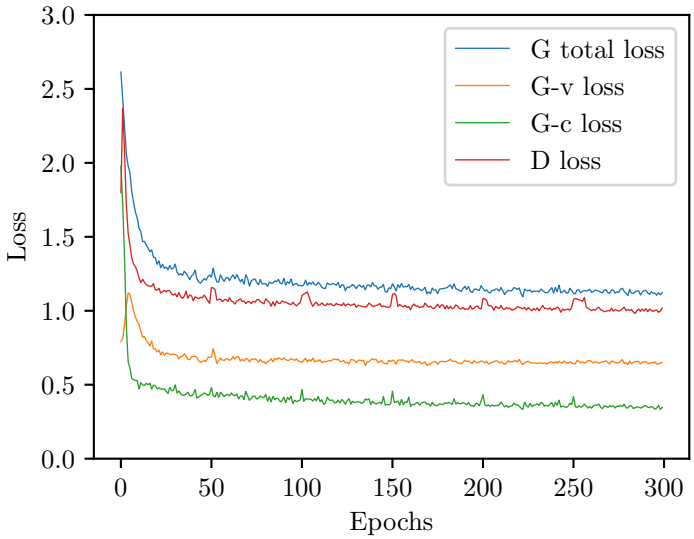


Figure 8.5: Plot of the training losses of discriminator D and generator G validity loss (G-v) and classification (G-c) loss components for each training epoch. All losses are monotonically decreasing and the G-v loss has plateaued by epoch 300.

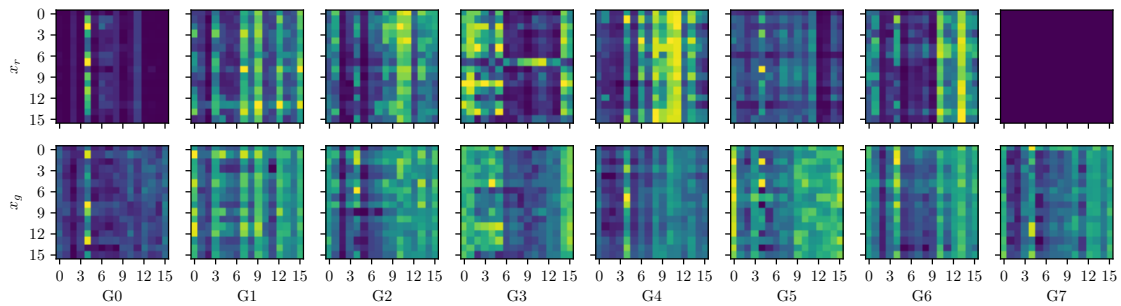


Figure 8.6: Comparison of real and generated samples of each class in a scale of colours. Dark blue represents the minimum signal while yellow corresponds to the maximum. G7 does not have a real class equivalent since it is a class created by the GAN.

Table 8.1: Similarity performance indicators between samples of the data set, Gaussian noise samples and GAN-generated samples. The mean and standard deviation of the distances between samples is shown for each class of gestures. The difference Δ between values and the baseline is shown as a percentage.

	Baseline		GAN				Noise			
Class	Mean	Std	Mean	Δ(%)	Std	Δ(%)	Mean	Δ(%)	Std	Δ(%)
0	1.43	0.63	1.44	1.0	0.33	−47.1	4.74	232.7	0.20	−67.7
1	3.20	1.40	3.55	11.0	0.86	−38.9	5.06	58.2	1.58	12.8
2	2.28	0.69	2.10	−7.8	0.57	−17.3	2.47	8.4	0.58	−15.6
3	4.03	1.66	4.09	1.5	0.93	−44.1	6.36	57.8	1.89	14.1
4	2.32	0.71	2.18	−6.0	0.63	−11.3	4.02	73.1	0.70	−1.7
5	1.95	0.77	2.00	2.2	0.85	9.7	2.55	30.6	0.46	−40.2
6	2.44	0.83	2.38	−2.5	0.81	−2.8	3.24	32.8	0.47	−44.0

dispersion within samples of the same class, is generally significantly lower than that of the data set. While the dispersion is comparatively low, it is significant, as seen in figure 8.6. It also indicates that the training of the GAN has not collapsed into generating a single type of sample.

Owing to the similarity between generated and trained samples, and the reasonable intra-class dispersion of the generated samples, we can conclude that the generator is successfully trained.

Discriminator Performance

The discriminator performance was measured in several settings. To ensure that the differences between them are due to the proposed methodology, the structure of the discriminator, figure 8.4, is kept the same for all tests. The networks are always initialized with the same weights. The data set is fixed as well, with classes 0-6 being used for the training, validation and test splits, while samples of class 7 are present only in the test split and are considered to be the class *others*.

The baseline test consists of the performance measurements calculated with the outputs of a discriminator that was trained as a single network. The training hyperparameters were optimized for this purpose. The Adam optimizer was used with a learning rate of 0.01. The training process was halted using the early stopping technique, were the loss on the validation data is monitored online. The training is stopped when the validation loss stops decreasing in 12 consecutive epochs to prevent over-fitting on the training data. Since this is a rather small training data set in a simple network, the fitting process takes just a few seconds.

In Test 1, the discriminator performance is equal to the baseline test, except that the training targets are noisy, as described in (8.6). The maximum value of the target vector of each sample was sampled before training from a uniform distribution $\mathcal{U}\sim(0.8, 1.0)$.

Table 8.2: Accuracy of the predictions of the discriminator on the test split. The *Class* and *Others* columns correspond to the trained and *others* classes, respectively. No threshold $\tau = 0$ was applied on the classification probability distribution.

$\tau = 0$	Accuracy (%)			
	Class	Others	Mean	Threshold
Baseline	98.1	0.0	57.4	0.00
Test 1	100.0	0.0	58.5	0.00
Test 2	98.1	0.0	57.4	0.00
Test 3	100.0	0.0	58.5	0.00

The second and third tests use the trained GAN with the setup described in the training parameters. For Test 2, the discriminator is used as trained in the GAN framework. For Test 3, it is retrained afterwards with noisy labels and the training set is augmented with more 224 newly generated samples, which is a data set size increase of about +50% in number of samples.

All of the tests were repeated with different values of thresholds τ (8.11) for the final classification decision. The no-threshold case is equivalent to setting $\tau = 0$. In the remaining cases, the threshold was optimized so that the accuracy on trained classes does not drop below 95% ($p = 0.95$) and 90% ($p = 0.90$), arbitrarily set values.

The accuracy on the test split of the no-threshold case, $p = 0.95$ and $p = 0.90$ for all tests is shown in tables 8.2 and 8.3. When no threshold is applied, no samples of the class *others* are correctly classified. The accuracy on the trained classes varies between 98.1 and 100.0%, but no conclusions can be made about the use of noisy labels or the GAN. However, when a threshold is used, the accuracy on *others* increases substantially. In the baseline case it increases to just 17.1% with a threshold $\tau = 0.95$. When noisy labels are added to the baseline test, it increases to 90.4% ($\tau = 0.63$). The best result is the case where the discriminator of the GAN is used, achieving 94.5% on *others*. However, when this discriminator is retrained, it decreases to 72.7%, meaning that the data augmentation that occurs every iteration of the GAN fitting leads to a better solution than when samples from the best generator are used.

If we accept a decrease of the trained classes accuracy to about 90% ($p = 0.90$), the tendency is similar to that of $p = 0.95$, but the accuracy on *others* is higher due to the increase of the optimized threshold τ. Because of the higher threshold, more samples of the trained classes are rejected. The accuracy on *others* is the highest (100.0%) for test 2, where we use the discriminator fitted by the GAN. The trade-off between the trained classes and *others* class test accuracies is shown on figure 8.7.

The precision measurements provide the fraction of samples predicted to be of class i that were actually correct. On the other hand, recall identifies the fraction of samples of class i that were classified as such. Therefore, recall can be seen as the accuracy of the samples of a certain class. However, if the classifier were to predict

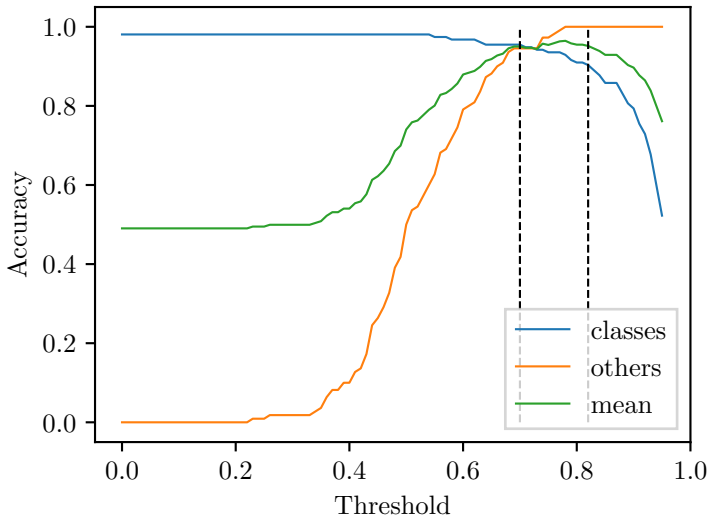


Figure 8.7: Trade-off between test split accuracy on the trained classes and *others* as a function of the decision threshold. The two vertical lines correspond to the $p = 0.95$ and $p = 0.90$ thresholds, from left to right.

Table 8.3: Accuracy of the predictions of the discriminator on the test split. The *Class* and *Others* columns correspond to the trained and *others* classes, respectively. The thresholds were optimized so that the class accuracy is about p.

	$p = 0.95$				$p = 0.90$			
	Accuracy (%)				Accuracy (%)			
	Class*	Others	Mean	Threshold	Class*	Others	Mean	Threshold
Baseline	96.1	17.3	63.4	0.95	96.1	17.3	63.4	0.95
Test 1	95.5	90.9	93.6	0.63	91.0	95.5	92.9	0.71
Test 2	95.5	94.5	95.1	0.70	90.3	100.0	94.3	0.82
Test 3	95.5	72.7	86.0	0.69	90.3	88.2	89.4	0.82

*Closest accuracy value to p after threshold optimization.

every sample to be of class i, its recall would be 100% while the precision would be the poor. So, we present these two metrics for all classes and tests that were performed, including all decision threshold levels, on tables 8.4, 8.5 and 8.6.

When no threshold is set, all the tests show 0 precision and recall for class 7 (*others*). This means that no samples are predicted to be of this class. The class 7 samples are confused as class 4 and 5, as seen by their low precision value. Notably, the recall is nearly perfect in all tests, but the prediction is low on classes 4 and 5. This reinforces the conclusion that out-of-vocabulary classes are often difficult to categorize. The discriminator trained with noisy labels (test 1) showed the best precision and recall.

Previously, we saw that the accuracy of the discriminator without a threshold is between 98.1 and 100.0% on the trained classes, and 0.0% on *others*. This happens because the network node for class 7 sees nearly no activation, since samples of class 7 are purposely excluded from the fitting process. However, setting a classification threshold below which the samples are considered to be of class *others* helps their recognition considerably. Tables 8.5 and 8.6 show the results of the application of these thresholds.

Tuning the threshold for a minimum trained class accuracy of 95%, increases the recall of the class *others* to just 13.6% in the baseline test. The use of noisy labels ($\mathcal{U}(0.8, 1.0)$ – test 1) increases it to 87.3%. This happens because the networks "learns" that the prediction probability does not have to be always 1.0 like the target vectors of the baseline test. Therefore, the activation scores are generally lower and it is easier to find a good threshold. Further improvement is seen on test 2, where the recall of *others* increases to 94.5%, but test 3 shows a slight decrease to 92.7%. Overall, the precision and recall improved significantly between the baseline and the remaining tests. The GAN had the best performance, relative to the baseline and the use of noisy labels. However, there is no significant improvement when the discriminator network is retrained with new generated samples (test 3).

The tendency is the same for the case where $p = 0.90$. However, there is a small difference between tests 2 and 3. In this case, the mean precision seen in test 2 is 2 percentage points higher than that of test 3. This is caused by an increase of the number of errors in the trained classes when the GAN discriminator is retrained.

8.5 Conclusion

In this work we implemented a novel way of solving the issue of classification of out-of-vocabulary gestures. Very often, these gestures are classified as an existing class and are difficult to remove with a classification threshold. The proposed solution has two components: (1) the use of a generative model (GAN) to augment the data set with new generated samples, (2) the use of noisy labels to decrease the average probability of predictions, thus facilitating threshold tuning.

To evaluate the proposed approach, we defined a baseline test where a neural network is used as a classification model without augmentation or noisy labels. Test 1 added noisy labels to the targets of the baseline test data set. The second test

Table 8.4: Precision and recall values for each class, on the test split, when no threshold is set for the final classification decision.

$\tau = 0$	Baseline		Test 1		Test 2		Test 3	
Class	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0	0.880	1.000	0.957	1.000	0.846	1.000	0.957	1.000
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	0.957	1.000	1.000	1.000	1.000	1.000	0.957	1.000
3	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
4	0.207	1.000	0.622	1.000	0.359	1.000	0.383	1.000
5	0.500	0.909	0.188	1.000	0.218	0.864	0.237	1.000
6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Mean	0.693	0.864	0.721	0.875	0.678	0.858	0.692	0.875

Table 8.5: Precision and recall values for each class, on the test split, when the classification decision threshold is optimized to achieved at least 0.95 accuracy on the trained classes.

$p = 0.95$	Baseline		Test 1		Test 2		Test 3	
Class	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.864
2	1.000	1.000	1.000	0.909	1.000	0.955	1.000	0.955
3	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
4	0.217	1.000	1.000	1.000	0.852	1.000	0.852	1.000
5	0.586	0.773	0.548	0.773	0.889	0.727	0.840	0.955
6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.955
7	0.750	0.136	0.932	0.873	0.937	0.945	0.944	0.927
Mean	0.819	0.864	0.935	0.944	0.960	0.953	0.955	0.957

Table 8.6: Precision and recall values for each class, on the test split, when the classification decision threshold is optimized to achieved at least 0.90 accuracy on the trained classes.

$p = 0.90$	Baseline		Test 1		Test 2		Test 3	
Class	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	1.000	1.000	0.955	1.000	1.000	1.000	0.636
2	1.000	1.000	1.000	0.773	1.000	0.773	1.000	0.909
3	1.000	1.000	1.000	0.955	1.000	0.909	1.000	1.000
4	0.217	1.000	1.000	1.000	1.000	1.000	0.880	0.957
5	0.586	0.773	0.667	0.727	1.000	0.636	0.952	0.909
6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.955
7	0.750	0.136	0.887	0.927	0.880	1.000	0.883	0.964
Mean	0.819	0.864	0.944	0.917	0.985	0.915	0.964	0.916

added the GAN approach, with its discriminator being used as the classification model. In the third and final test, the previous discriminator is retrained with a 50% augmented training split where the new samples are generated with the GAN's generator and with noisy labels. The discriminator shape and starting point are kept static in all tests.

For comparison between tests, the performance of the different discriminators is measured through the classification accuracy, precision and recall on a held-out test split. In a first approach, these metrics were calculated when there is no classification threshold. In two other cases, the threshold tuned so that there is a maximum loss of trained class accuracy of 5 and 10%. Threshold tuning is a trade-off of between accuracy on trained and non-trained classes.

The tests were performed on the UC2018 DualMyo data set with 7 trained classes and 1 non-trained class (*others*). The baseline results showed that without a threshold, all of the samples of *others* were confused with another class. Considering a tuned threshold, only 17.3% of *others* were excluded correctly. Without a threshold, all of the samples of class *others* were excluded. Considering a threshold tuned to a loss of 5% ($p = 0.95$) and 10% ($p = 0.90$), the best rejection rate of *others* was 95.4% and 100.0%, respectively, with the GAN discriminator (test 2).

The goals of this work were successfully achieved, despite the successful training of the GAN being a major challenge, similarly to other projects. Despite the success, further validation of the methodology should be done on richer data sets. Further work should also be done on the generation of time-series.

Chapter 9

Conclusion

The work presented in this thesis demonstrates that the objectives proposed at the beginning were achieved. The main goal was the development of a novel framework for the online recognition of hand gestures from data streams, which is divided in several modules that were evaluated individually. The most important modules are the gesture database, stream segmentation and gesture classification. Furthermore, the classification module has the capability of excluding out-of-vocabulary gestures and perform predictive classification, i.e., before the gesture is completed.

The presented methodologies were tested on two new hand gesture datasets that are publicly available: (1) UC2017 SG/DG dataset, based on a dataglove and hand tracking, and (2) UC2018 DualMyo dataset, based on EMG sensing. Initially, a deep literature review was done focusing on recent studies about EMG signal acquisition and pattern recognition. This review concluded that the current efforts on EMG PR improvement are targeted at increasing the reliability of the classification by anticipating and correcting sources of EMG disturbances, such as muscle fatigue, varying skin impedance, electrode shift and lift-off. However, a definite solution has not been found and future work should focus on increasing the number of sensors with deep learning for feature extraction and classification.

In terms of data stream segmentation, the proposed method enabled motion detection for any kind of gesture in a sequence of unsegmented and unbounded data, in an unsupervised fashion without prior training. A quantitative analysis indicates an over-segmentation error of 2% with most of the segments having a shift-right and extend behaviour, i.e., with a small delay in real-time streams. The segmentation method accurately determines the boundaries of a gesture, thus improving its subsequent classification. Any kind of sequential positional or intensity data can be used as input and good results were obtained with the dataglove and tracker combo, IMU and EMG sensors. Further work should be done vision-based systems.

The segmentation module simplifies the classification problem by separating static and dynamic gestures. The UC2017 SG/DG dataset, with a total of 34 distinct gesture classes, was used to compare the traditional classification methods with engineered features, and deep learning methods such as CNNs and LSTM neural nets. Experimental results showed a classification accuracy above 95% on SGs and 99% for DGs (after the gesture is finished). On online frame-by-frame classification, the

deep learning methods outperformed the engineered features based on PCA. The LSTM net and CNN use scaled raw data, therefore being more easily extendable to new datasets. Furthermore, the sequential classification of DGs with either PCA features or raw data showed an accuracy that is higher with partial gesture data (50% or 75% of the initial frames of a DG sample) than with the full DG data. In this context, DGs can be accurately classified in anticipation, even before the user finishes the gesture, thus allowing faster and more efficient gesture-based control of a robot by cutting the processing time overheads.

The last significant contribution of the thesis is the improvement of rejection rates of out-of-vocabulary gestures through semi-supervised learning with a modified AC-GAN. The introduction of one-hot encoded target vectors with noisy activation labels improved the rejection rate from 17.3% in the baseline to 94.5% on the UC2018 DualMyo dataset. In this case, just 3% of in-vocabulary samples were rejected. These results were obtained on SGs, so further validation of this method should be done on larger datasets, such as the UC2017 SG/DG dataset.

Qualitative analysis showed that a HRI system based on the proposed framework is adequate to associate recognized gesture patterns to robot commands and by this way teleoperate the collaborative robot in an intuitive fashion. Future work will be dedicated to testing the proposed solution quantitatively against traditional programming methods. The promising results obtained with predictive classification will be explored as a way to anticipate robot reaction to human behaviour.

Bibliography

[1] Guang-Zhong Yang et al. "The grand challenges of Science Robotics". In: *Science Robotics* 3.14 (2018). DOI: 10.1126/scirobotics.aar7650. eprint: http://robotics.sciencemag.org/content/3/14/eaar7650.full.pdf. URL: http://robotics.sciencemag.org/content/3/14/eaar7650.

[2] J Rey et al. *Learning motions from demonstrations and rewards with time-invariant dynamical systems based policies.* Autonomous Robots. 2017.

[3] Sara Sheikholeslami, AJung Moon, and Elizabeth A Croft. "Cooperative gestures for industry: Exploring the efficacy of robot hand configurations in expression of instructional gestures for human-robot interaction". In: *The International Journal of Robotics Research* 36.5-7 (2017), pp. 699–720.

[4] M. Simão, P. Neto, and O. Gibaru. "Natural control of an industrial robot using hand gesture recognition with neural networks". In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society.* Oct. 2016, pp. 5322–5327. DOI: 10.1109/IECON.2016.7793333.

[5] Ruiduo Yang, Sudeep Sarkar, and Barbara Loeding. "Handling movement epenthesis and hand segmentation ambiguities in continuous sign language recognition using nested dynamic programming." In: *IEEE transactions on pattern analysis and machine intelligence* 32.3 (Mar. 2010), pp. 462–77. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2009.26.

[6] Miguel Simão, Pedro Neto, and Olivier Gibaru. *UC2017 Static and Dynamic Hand Gestures.* 2018. DOI: 10.5281/zenodo.1319659. URL: https://doi.org/10.5281/zenodo.1319659.

[7] Miguel Simão, Pedro Neto, and Olivier Gibaru. *UC2018 DualMyo Hand Gesture Dataset.* 2018. DOI: 10.5281/zenodo.1320922. URL: https://doi.org/10.5281/zenodo.1320922.

[8] M. A. Simão, P. Neto, and O. Gibaru. "Unsupervised Gesture Segmentation by Motion Detection of a Real-Time Data Stream". In: *IEEE Transactions on Industrial Informatics* in press.99 (2016), pp. 1–1. ISSN: 1551-3203. DOI: 10.1109/TII.2016.2613683.

[9] Miguel Simão, Pedro Neto, and Olivier Gibaru. "Using data dimensionality reduction for recognition of incomplete dynamic gestures". In: *Pattern Recognition Letters* 99 (2017). User Profiling and Behavior Adaptation for Human-Robot Interaction, pp. 32–38. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2017.01.003. URL: http://www.sciencedirect.com/science/article/pii/S016786551730003X.

[10] M. A. Simão, P. Neto, and O. Gibaru. "Unsupervised gesture segmentation of a real-time data stream in MATLAB". In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. Oct. 2016, pp. 809–814. DOI: 10.1109/IECON.2016.7793517.

[11] M. Simão, P. Neto, and O. Gibaru. "Taking Advantage of Data Dimensionality Reduction for Dynamic Gesture Recognition from Incomplete Data". In: *Workshop on Behavior Adaptation, Interaction and Learning for Assistive Robotics, IEEE RO-MAN 2016, NYC, USA*. 2016.

[12] B. Burger et al. "Two-handed gesture recognition and fusion with speech to command a robot". In: *Autonomous Robots* 32.2 (Dec. 2011), pp. 129–147. ISSN: 0929-5593. DOI: 10.1007/s10514-011-9263-y.

[13] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. "A Gesture Based Interface for Human-Robot Interaction". In: *Autonomous Robots* 9.2 (Sept. 2000), pp. 151–173. ISSN: 1573-7527. DOI: 10.1023/A:1008918401478.

[14] Pedro Neto, J. Norberto Pires, and António Paulo Moreira. "3-D position estimation from inertial sensing: Minimizing the error from the process of double integration of accelerations". In: *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, Nov. 2013, pp. 4026–4031. ISBN: 978-1-4799-0224-8. DOI: 10.1109/IECON.2013.6699780.

[15] Michael T. Wolf et al. "Decoding static and dynamic arm and hand gestures from the JPL BioSleeve". In: *IEEE Aerospace Conference Proceedings* (2013). ISSN: 1095323X. DOI: 10.1109/AERO.2013.6497171.

[16] Giorgio Biagetti et al. "Homomorphic Deconvolution for MUAP Estimation from Surface EMG Signals". In: *IEEE Journal of Biomedical and Health Informatics* 2194.c (2016), pp. 1–1. ISSN: 2168-2194. DOI: 10.1109/JBHI.2016.2530943. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7409930.

[17] Todd R. Farrell and R. F Ff Weir. "A comparison of the effects of electrode implantation and targeting on pattern classification accuracy for prosthesis control". In: *IEEE Transactions on Biomedical Engineering* 55.9 (2008), pp. 2198–2211. ISSN: 00189294. DOI: 10.1109/TBME.2008.923917.

[18] Lauren H. Smith and Levi J. Hargrove. "Comparison of surface and intramuscular EMG pattern recognition for simultaneous wrist/hand motion classification". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (2013), pp. 4223–4226. ISSN: 1557170X. DOI: 10.1109/EMBC.2013.6610477. arXiv: NIHMS150003.

[19] A Botter, M Gazzoni, and R Merletti. "Surface Electromyogram Detection". In: *Introduction to Neural Engineering for Motor Rehabilitation*. Wiley-IEEE Press, 2013, pp. 600–. ISBN: 9781118628522. DOI: 10.1002/9781118628522.ch6. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6583350.

[20] Roberto Merletti and Dario Farina. *Surface Electromyography: Physiology, Engineering, and Applications*. Wiley-IEEE Press, 2016. ISBN: 9781118987025. DOI: 10.1002/9781119082934. arXiv: arXiv:1011.1669v3.

[21] Dario Farina et al. "The extraction of neural information from the surface EMG for the control of upper-limb prostheses: Emerging avenues and challenges". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 22.4 (2014), pp. 797–809. ISSN: 15344320. DOI: 10.1109/TNSRE.2014.2305111.

[22] E A Clancy, E L Morin, and R Merletti. "Sampling, noise-reduction and amplitude estimation issues in surface electromyography." eng. In: *Journal of electromyography and kinesiology : official journal of the International Society of Electrophysiological Kinesiology* 12.1 (Feb. 2002), pp. 1–16. ISSN: 1050-6411 (Print).

[23] G Piervirgili, F Petracca, and R Merletti. "A new method to assess skin treatments for lowering the impedance and noise of individual gelled Ag-AgCl electrodes". In: *Physiological Measurement* 35.10 (2014), pp. 2101–2118. URL: http://stacks.iop.org/0967-3334/35/i=10/a=2101.

[24] Norden E. Huang et al. "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis". In: *Proc. R. Soc. Lond.* 454 (1998), pp. 903–950.

[25] Xu Zhang and Ping Zhou. "Filtering of surface EMG using ensemble empirical mode decomposition." In: *Medical engineering & physics* 35.4 (2013), pp. 537–42. ISSN: 1873-4030. DOI: 10.1016/j.medengphy.2012.10.009. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3769943%7B%5C&%7Dtool=pmcentrez%7B%5C&%7Drendertype=abstract.

[26] Araceli Guadalupe Santana Rayo et al. "Design and Manufacturing of a Dry Electrode for EMG Signals Recording with Microneedles". In: *Improved performance of materials*. Ed. by Andreas Öchsner and Holm Altenbach. Springer International Publishing, 2018. Chap. 22, pp. 259–267. ISBN: 978-3-319-59590-0. DOI: 10.1007/978-3-319-59590-0_22. URL: http://link.springer.com/10.1007/978-3-319-59590-0%7B%5C_%7D22.

[27] Ki-Hee Park, Heung-Il Suk, and Seong-Whan Lee. "Position-Independent Decoding of Movement Intention for Proportional Myoelectric Interfaces". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* X.X (2015), pp. 1–1. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2015.2481461. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7275160.

[28] F. Riillo et al. "Optimization of EMG-based hand gesture recognition: Supervised vs. unsupervised data preprocessing on healthy subjects and transradial amputees". In: *Biomedical Signal Processing and Control* 14 (2014), pp. 117–125. ISSN: 17468108. DOI: 10.1016/j.bspc.2014.07.007.

[29] Arash Ajoudani, Nikolaos G. Tsagarakis, and Antonio Bicchi. "Tele-impedance: Teleoperation with impedance regulation using a body-machine interface". In: *The International Journal of Robotics Research* (2012). DOI: 10.1177/0278364912464668. eprint: http://ijr.sagepub.com/content/early/2012/10/31/0278364912464668.full.pdf+html. URL: http://ijr.sagepub.com/content/early/2012/10/31/0278364912464668.abstract.

[30] Federico Nicolás Guerrero, Enrique Mario Spinelli, and Marcelo Alejandro Haberman. "Analysis and Simple Circuit Design of Double Differential EMG Active Electrode". In: (2015), pp. 1–9.

[31] S H Park and S P Lee. "EMG pattern recognition based on artificial intelligence techniques." In: *IEEE transactions on rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society* 6.4 (1998), pp. 400–405. ISSN: 1063-6528. DOI: 10.1109/86.736154.

[32] Yi Hung Liu and Han Pang Huang. "Towards a high-stability EMG recognition system for prosthesis control: A one-class classification based non-target EMG pattern filtering scheme". In: *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics* October (2009), pp. 4752–4757. ISSN: 1062922X. DOI: 10.1109/ICSMC.2009.5346086.

[33] Jie Liu and Ping Zhou. "A novel myoelectric pattern recognition strategy for hand function restoration after incomplete cervical spinal cord injury". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 21.1 (2013), pp. 96–103. ISSN: 15344320. DOI: 10.1109/TNSRE.2012.2218832.

[34] Zhijun Li et al. "SEMG-based joint force control for an upper-limb power-assist exoskeleton robot". In: *IEEE Journal of Biomedical and Health Informatics* 18.3 (2014), pp. 1043–1050. ISSN: 21682194. DOI: 10.1109/JBHI.2013.2286455.

[35] John A Spanias et al. "Effect of Additional Mechanical Sensor Data on an EMG-based Pattern Recognition System for Powered Leg Prosthesis". In: *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)* (2015), pp. 639–642. DOI: 10.1109/NER.2015.7146704.

[36] Antonietta Stango, Francesco Negro, and Dario Farina. "Spatial Correlation of High Density EMG Signals Provides Features Robust to Electrode Number and Shift in Pattern Recognition for Myocontrol". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 23.2 (2015), pp. 189–198. ISSN: 15344320. DOI: 10.1109/TNSRE.2014.2366752.

[37] David T. Mewett, Homer Nazeran, and Karen J. Reynolds. "Removing power line noise from recorded EMG". In: *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Vol. 3. IEEE, 2001, pp. 2190–2193. ISBN: 0-7803-7211-5. DOI: 10.1109/IEMBS.2001.1017205. URL: http://ieeexplore.ieee.org/document/1017205/.

[38] D. T. Mewett, K. J. Reynolds, and H. Nazeran. "Reducing power line interference in digitised electromyogram recordings by spectrum interpolation". In: *Medical and Biological Engineering and Computing* 42.4 (July 2004), pp. 524–531. ISSN: 0140-0118. DOI: 10.1007/BF02350994.

[39] Chavdar Levkov et al. "Removal of power-line interference from the ECG: a review of the subtraction procedure". In: *BioMedical Engineering OnLine* 4.1 (Aug. 2005), p. 50. ISSN: 1475925X. DOI: 10.1186/1475-925X-4-50.

[40] Mojtaba Malboubi et al. "Power line noise elimination from EMG signals using adaptive laguerre filter with fuzzy step size". In: *2010 17th Iranian Conference of Biomedical Engineering, ICBME 2010 - Proceedings*. IEEE, Nov. 2010, pp. 1–4. ISBN: 9781424474844. DOI: 10.1109/ICBME.2010.5704932. URL: http://ieeexplore.ieee.org/document/5704932/.

[41] Angkoon Phinyomark et al. "Feature extraction of the first difference of EMG time series for EMG pattern recognition". In: *Computer Methods and Programs in Biomedicine* 117.2 (2014), pp. 247–256. ISSN: 01692607. DOI: 10.1016/j.cmpb.2014.06.013.

[42] Alberto Botter and Taian M. Vieira. "Filtered virtual reference: A new method for the reduction of power line interference with minimal distortion of monopolar surface EMG". In: *IEEE Transactions on Biomedical Engineering* 62.11 (2015), pp. 2638–2647. ISSN: 15582531. DOI: 10.1109/TBME.2015.2438335.

[43] Maciej Niegowski et al. "Unsupervised learning technique for surface electromyogram denoising from power line interference and baseline wander". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* 2015-Novem (2015), pp. 7274–7277. ISSN: 1557170X. DOI: 10.1109/EMBC.2015.7320071.

[44] Babak Afsharipour et al. "Spatial distribution of surface EMG on trapezius and lumbar muscles of violin and cello players in single note playing". In: *Journal of Electromyography and Kinesiology* 31 (Dec. 2016), pp. 144–153. ISSN: 18735711. DOI: 10.1016/j.jelekin.2016.10.003. URL: http://www.sciencedirect.com/science/article/pii/S1050641116302280?via%7B%5C%%7D3Dihub.

[45] Dennis C. Tkach et al. "Real-time and offline performance of pattern recognition myoelectric control using a generic electrode grid with targeted muscle reinnervation patients". In: *IEEE Transactions on Neural Systems and*

Rehabilitation Engineering 22.4 (2014), pp. 727–734. ISSN: 15344320. DOI: 10.1109/TNSRE.2014.2302799.

[46] Nicolo Celadon et al. "Individual finger classification from surface EMG: Influence of electrode set". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* 2015-Novem (2015), pp. 7284–7287. ISSN: 1557170X. DOI: 10.1109/EMBC.2015.7320073.

[47] Yong Ning et al. "Surface EMG decomposition based on K-means clustering and convolution kernel compensation". In: *IEEE Journal of Biomedical and Health Informatics* 19.2 (2015), pp. 471–477. ISSN: 21682194. DOI: 10.1109/JBHI.2014.2328497.

[48] Mark Ison et al. "High-Density Electromyography and Motor Skill Learning for Robust Long-Term Control of a 7-DoF Robot Arm." In: *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society* 4320.c (2015), pp. 1–10. ISSN: 1558-0210. DOI: 10.1109/TNSRE.2015.2417775. URL: http://www.ncbi.nlm.nih.gov/pubmed/25838524.

[49] Vojko Glaser, Ales Holobar, and Damjan Zazula. "Real-time motor unit identification from high-density surface EMG". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 21.6 (Nov. 2013), pp. 949–958. ISSN: 15344320. DOI: 10.1109/TNSRE.2013.2247631. URL: http://ieeexplore.ieee.org/document/6475191/.

[50] Sarbast Rasheed, Daniel W Stashuk, and Mohamed S Kamel. "Classifier Fusion Interactive Software Toolbox for EMG Signal Decomposition". In: (2015), pp. 806–811.

[51] Maoqi Chen and Ping Zhou. "A Novel Framework Based on FastICA for High Density Surface EMG Decomposition". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 4320.c (2015), pp. 1–1. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2015.2412038. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7058391.

[52] Angkoon Phinyomark, Pornchai Phukpattaranont, and Chusak Limsakul. "Feature reduction and selection for EMG signal classification". In: *Expert Systems with Applications* 39.8 (2012), pp. 7420–7431. ISSN: 09574174. DOI: 10.1016/j.eswa.2012.01.102.

[53] Xiao Hu and Valeriy Nenov. "Multivariate AR modeling of electromyography for the classification of upper arm movements". In: *Clinical Neurophysiology* 115 (2004), pp. 1276–1287. ISSN: 13882457. DOI: 10.1016/j.clinph.2003.12.030.

[54] Takamitsu Matsubara and Jun Morimoto. "Bilinear modeling of EMG signals to extract user-independent features for multiuser myoelectric interface". In: *IEEE Transactions on Biomedical Engineering* 60.8 (2013), pp. 2205–2213. ISSN: 00189294. DOI: 10.1109/TBME.2013.2250502.

[55] Zhijun Li et al. "Boosting-based EMG patterns classification scheme for robustness enhancement". In: *IEEE Journal of Biomedical and Health Informatics* 17.3 (2013), pp. 545–552. ISSN: 21682194. DOI: 10.1109/JBHI.2013.2256920.

[56] Zhaojie Ju and Honghai Liu. "Human Hand Motion Analysis With Multisensory Information". In: 19.2 (2014), pp. 456–466.

[57] Max Ortiz-catalan, Bo Håkansson, and Rickard Brånemark. "Real-Time and Simultaneous Control of Artificial Limbs Based on Pattern Recognition Algorithms". In: 22.4 (2014), pp. 756–764.

[58] H. Kawasaki et al. "Learning system for myoelectric prosthetic hand control by forearm amputees". In: *The 23rd IEEE International Symposium on Robot and Human Interactive Communication* (2014), pp. 899–904. DOI: 10.1109/ROMAN.2014.6926367.

[59] An-Chih Tsai, Jer-Junn Luh, and Ta-Te Lin. "A novel STFT-ranking feature of multi-channel EMG for motion pattern recognition". In: *Expert Systems with Applications* 42.7 (2015), pp. 3327–3341. ISSN: 09574174. DOI: 10.1016/j.eswa.2014.11.044. URL: http://linkinghub.elsevier.com/retrieve/pii/S095741741400743X.

[60] André L.V. Coelho and Clodoaldo a.M. Lima. "Assessing fractal dimension methods as feature extractors for EMG signal classification". In: *Engineering Applications of Artificial Intelligence* 36 (2014), pp. 81–98. ISSN: 09521976. DOI: 10.1016/j.engappai.2014.07.009. URL: http://linkinghub.elsevier.com/retrieve/pii/S0952197614001778.

[61] Rami N. Khushaba, Ali Al-Timemy, and Sarath Kodagoda. "Influence of multiple dynamic factors on the performance of myoelectric pattern recognition". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* 2015-Novem (2015), pp. 1679–1682. ISSN: 1557170X. DOI: 10.1109/EMBC.2015.7318699.

[62] Jianwei Liu et al. "Enhanced robustness of myoelectric pattern recognition to across-day variation through invariant feature extraction". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* 2015-Novem.1 (2015), pp. 7262–7265. ISSN: 1557170X. DOI: 10.1109/EMBC.2015.7320068.

[63] Haitham M. Al-Angari et al. "Distance and mutual information methods for {EMG} feature and channel subset selection for classification of hand movements". In: *Biomedical Signal Processing and Control* 27 (2016), pp. 24–31. ISSN: 1746-8094. DOI: 10.1016/j.bspc.2016.01.011. URL: http://www.sciencedirect.com/science/article/pii/S1746809416300040.

[64] G. R. Naik, K. G. Baker, and H. T. Nguyen. "Dependence Independence Measure for Posterior and Anterior EMG Sensors Used in Simple and Complex Finger Flexion Movements: Evaluation Using SDICA". In: *IEEE Journal of Biomedical and Health Informatics* 19.5 (Sept. 2015), pp. 1689–1696. ISSN: 2168-2194. DOI: 10.1109/JBHI.2014.2340397.

[65] James Cannan and Huosheng Hu. "Using forearm circumference for automatic threshold calibration for simple EMG control". In: *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics: Mechatronics for Human Wellbeing, AIM 2013* (2013), pp. 1476–1481. ISSN: 2159-6247. DOI: 10.1109/AIM.2013.6584303.

[66] Chris Thornton. "Separability is a learner's best friend". In: *Proceedings of the Fourth Neural Computation and ...* (1997), pp. 40–47. DOI: 10.1007/978-1-4471-1546-5{_}4. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.139.4361%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[67] G. R. Naik et al. "Principal Component Analysis Applied to Surface Electromyography: A Comprehensive Review". In: *IEEE Access* 4 (2016), pp. 4025–4037. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2016.2593013.

[68] Ganesh Naik and Hung Nguyen. "Non Negative Matrix Factorization for the Identification of EMG finger movements: Evaluation using matrix analysis". In: *IEEE Journal of Biomedical and Health Informatics* PP.c (2014), pp. 1–1. ISSN: 2168-2194. DOI: 10.1109/JBHI.2014.2326660. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6824741.

[69] Dd Lee and Hs Seung. "Algorithms for non-negative matrix factorization". In: *Advances in neural information processing systems* 1 (2001), pp. 556–562. ISSN: 10987576. DOI: 10.1109/IJCNN.2008.4634046. URL: http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.

[70] Rami N. Khushaba, Ahmed Al-Ani, and Adel Al-Jumaily. "Orthogonal fuzzy neighborhood discriminant analysis for multifunction myoelectric hand control". In: *IEEE Transactions on Biomedical Engineering* 57.6 (2010), pp. 1410–1419. ISSN: 00189294. DOI: 10.1109/TBME.2009.2039480.

[71] G. R. Naik, A. H. Al-Timemy, and H. T. Nguyen. "Transradial Amputee Gesture Classification Using an Optimal Number of sEMG Sensors: An Approach Using ICA Clustering". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 24.8 (Aug. 2016), pp. 837–846. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2015.2478138.

[72] Ahmet Alkan and Mücahid Günay. "Identification of EMG signals using discriminant analysis and SVM classifier". In: *Expert Systems with Applications* 39.1 (2012), pp. 44–47. ISSN: 09574174. DOI: 10.1016/j.eswa.2011.06.043.

[73] Xiaorong Zhang, He Huang, and Qing Yang. "Real-time implementation of a self-recovery EMG pattern recognition interface for artificial arms". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (2013), pp. 5926–5929. ISSN: 1557170X. DOI: 10.1109/EMBC.2013.6610901.

[74] Ali H. Al-Timemy et al. "Classification of finger movements for the dexterous hand prosthesis control with surface electromyography". In: *IEEE Journal of Biomedical and Health Informatics* 17.3 (2013), pp. 608–618. ISSN: 21682194. DOI: 10.1109/JBHI.2013.2249590.

[75] Ercan Gokgoz and Abdulhamit Subasi. "Comparison of decision tree algorithms for EMG signal classification using DWT". In: *Biomedical Signal Processing and Control* 18 (2015), pp. 138–144. ISSN: 17468094. DOI: 10.1016/j.bspc.2014.12.005. URL: http://linkinghub.elsevier.com/retrieve/pii/S1746809414002006.

[76] P. Geethanjali and K. K. Ray. "A Low-Cost Real-Time Research Platform for EMG Pattern Recognition-Based Prosthetic Hand". In: *IEEE/ASME Transactions on Mechatronics* 20.4 (2014), pp. 1948–1955. ISSN: 10834435. DOI: 10.1109/TMECH.2014.2360119.

[77] Long Peng et al. "Feasibility of NeuCube Spiking Neural Network Architecture for EMG Pattern Recognition". In: (2015), pp. 22–24.

[78] Qichuan Ding et al. "Missing-Data Classification with the Extended Full-Dimensional Gaussian Mixture Model: Applications to EMG-Based Motion Recognition". In: *IEEE Transactions on Industrial Electronics* 62.8 (2015), pp. 4994–5005. ISSN: 02780046. DOI: 10.1109/TIE.2015.2403797.

[79] Lizhi Pan et al. "Continuous estimation of finger joint angles under different static wrist motions from surface EMG signals". In: *Biomedical Signal Processing and Control* 14 (2014), pp. 265–271. ISSN: 17468094. DOI: 10.1016/j.bspc.2014.08.004. URL: http://linkinghub.elsevier.com/retrieve/pii/S1746809414001256.

[80] Han Li, Xi Chen, and Pengfei Li. "Human-computer Interaction System Design Based on Surface EMG Signals". In: (2014), pp. 94–98.

[81] Geethanjali Purushothaman and K. K. Ray. "EMG based man-machine interaction-A pattern recognition research platform". In: *Robotics and Autonomous Systems* 62.6 (2014), pp. 864–870. ISSN: 09218890. DOI: 10.1016/j.robot.2014.01.008.

[82] Varadach Amatanon. "Sign Language-Thai Alphabet Conversion Based on Electromyogram (EMG) L1x (W) 1 L (x (W)". In: (2014).

[83] Marco Paleari et al. "On optimal electrode configuration to estimate hand movements from forearm surface electromyography". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* 2015-Novem (2015), pp. 6086–6089. ISSN: 1557170X. DOI: 10.1109/EMBC.2015.7319780.

[84] Samanta Rosati et al. "Muscle activation patterns during gait: A hierarchical clustering analysis". In: *Biomedical Signal Processing and Control* 31 (2017), pp. 463–469. ISSN: 1746-8094. DOI: 10.1016/j.bspc.2016.09.017. URL: http://www.sciencedirect.com/science/article/pii/S1746809416301434.

[85] Yusuke Yamanoi et al. "Development of myoelectric hand that determines hand posture and estimates grip force simultaneously". In: *Biomedical Signal Processing and Control* 38 (2017), pp. 312–321. ISSN: 1746-8094. DOI: 10.1016/j.bspc.2017.06.019. URL: http://www.sciencedirect.com/science/article/pii/S1746809417301313.

[86] R. Menon et al. "Study on interaction between temporal and spatial information in classification of EMG signals in myoelectric prostheses". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* PP.99 (2017), pp. 1–1. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2017.2687761.

[87] Jianwei Liu et al. "Quantification and solutions of arm movements effect on sEMG pattern recognition". In: *Biomedical Signal Processing and Control* 13 (2014), pp. 189–197. ISSN: 1746-8094. DOI: 10.1016/j.bspc.2014.05.001. URL: http://www.sciencedirect.com/science/article/pii/S1746809414000792.

[88] Dario Farina. "Interpretation of the Surface Electromyogram in Dynamic Contractions". In: *Exercise and Sport Sciences Reviews* 34.3 (2006). ISSN: 0091-6331. URL: http://journals.lww.com/acsm-essr/Fulltext/2006/07000/Interpretation%7B%5C_%7Dof%7B%5C_%7Dthe%7B%5C_%7DSurface%7B%5C_%7DElectromyogram%7B%5C_%7Din.6.aspx.

[89] Catherine Disselhorst-Klug, Thomas Schmitz-Rode, and Günter Rau. "Surface electromyography and muscle force: Limits in sEMG - force relationship and new approaches for applications". In: *Clinical Biomechanics* 24.3 (Feb. 2015), pp. 225–235. DOI: 10.1016/j.clinbiomech.2008.08.003. URL: http://www.clinbiomech.com/article/S0268-0033(08)00265-9/abstract.

[90] P.K. Artemiadis and K.J. Kyriakopoulos. "EMG-Based Control of a Robot Arm Using Low-Dimensional Embeddings". In: *IEEE Transactions on Robotics* 26.2 (Apr. 2010), pp. 393–398. ISSN: 1552-3098. DOI: 10.1109/TRO.2009.2039378.

[91] Christian Fleischer and Günter Hommel. "A human-exoskeleton interface utilizing electromyography". In: *IEEE Transactions on Robotics* 24.4 (2008), pp. 872–882. ISSN: 15523098. DOI: 10.1109/TRO.2008.926860.

[92] Omid Haddad and Gary a. Mirka. "Trunk muscle fatigue and its implications in EMG-assisted biomechanical modeling". In: *International Journal of Industrial Ergonomics* 43.5 (2013), pp. 425–429. ISSN: 01698141. DOI: 10.1016/j.ergon.2013.08.004.

[93] Javad Hashemi et al. "Enhanced Dynamic EMG-Force Estimation Through Calibration and PCI Modeling." In: *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society* 4320.1 (2014), pp. 1–10. ISSN: 1558-0210. DOI: 10.1109/TNSRE.2014.2325713. URL: http://www.ncbi.nlm.nih.gov/pubmed/24860036.

[94] Jessica Beltran Ullauri et al. "On the EMG-based torque estimation for humans coupled with a force-controlled elbow exoskeleton". In: *2015 International Conference on Advanced Robotics (ICAR)* (2015), pp. 302–307. DOI: 10.1109/ICAR.2015.7251472. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7251472.

[95] Riccardo Valentini et al. "Processing of sEMG signals for online motion of a single robot joint through GMM modelization". In: *IEEE International Conference on Rehabilitation Robotics* 2015-Septe (2015), pp. 943–949. ISSN: 19457901. DOI: 10.1109/ICORR.2015.7281325.

[96] Armin Ehrampoosh and Aghil Yousefi-koma. "Estimation of Elbow Joint Abgle by NARX Model using EMG data". In: (2015), pp. 450–455.

[97] Liang Peng, Zeng Guang Hou, and Weiqun Wang. "A dynamic EMG-torque model of elbow based on neural networks". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* 2015-Novem (2015), pp. 2852–2855. ISSN: 1557170X. DOI: 10.1109/EMBC.2015.7318986.

[98] Jianda Han et al. "A State-Space EMG Model for the Estimation of Continuous Joint Movements". In: *IEEE Transactions on Industrial Electronics* 62.7 (2015), pp. 4267–4275. ISSN: 0278-0046. DOI: 10.1109/TIE.2014.2387337. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7001065.

[99] Zhichuan Tang, Hongnian Yu, and Shuang Cang. "Impact of Load Variation on Joint Angle Estimation from Surface EMG Signals". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 4320.c (2015), pp. 1–1. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2015.2502663. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7332977.

[100] Agamemnon Krasoulis, Sethu Vijayakumar, and Kianoush Nazarpour. "Evaluation of regression methods for the continuous decoding of finger movement from surface EMG and accelerometry". In: *Neural Engineering (NER), 2015 7th International IEEE/EMBS Conference on* (2015), pp. 631–634. DOI: 10.1109/NER.2015.7146702.

[101] Wennan Chang et al. "A Hierarchical Hand Motions Recognition Method Based on IMU and sEMG Sensors". In: (2015), pp. 1024–1029.

[102] Jian Wu et al. "Real-time American Sign Language Recognition Using Wrist-worn Motion and Surface EMG Sensors". In: (2015), pp. 1–6.

[103] Roberto Meattini et al. "Experimental Evaluation of a sEMG-based Human-Robot Interface for Human-Like Grasping Tasks". In: (2015), pp. 1030–1035.

[104] J. L. Betthauser et al. "Limb Position Tolerant Pattern Recognition for Myoelectric Prosthesis Control with Adaptive Sparse Representations from Extreme Learning". In: *IEEE Transactions on Biomedical Engineering* PP.99 (2017), pp. 1–1. ISSN: 0018-9294. DOI: 10.1109/TBME.2017.2719400.

[105] C. Hartmann et al. "Closed-Loop Control of Myoelectric Prostheses With Electrotactile Feedback: Influence of Stimulation Artifact and Blanking". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 23.5 (Sept. 2015), pp. 807–816. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2014.2357175.

[106] John Spanias, Eric Perreault, and Levi Hargrove. "Detection of and Compensation for EMG Disturbances for Powered Lower Limb Prosthesis Control". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 4320.c (2015), pp. 1–1. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2015.2413393. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7070694.

[107] David Hofmann et al. "Bayesian Filtering of Surface EMG for Accurate Simultaneous and Proportional Prosthetic Control". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* (2015), pp. 1–1. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2015.2501979. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7332757.

[108] Erik Scheme et al. "Motion normalized proportional control for improved pattern recognition-based myoelectric control". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 22.1 (2014), pp. 149–157. ISSN: 15344320. DOI: 10.1109/TNSRE.2013.2247421.

[109] Ghulam Rasool et al. "Real-Time Task Discrimination for Myoelectric Control Employing Task-Specific Muscle Synergies." In: *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society* 24.1 (2016), pp. 98–108. ISSN: 1558-0210. DOI: 10.1109/TNSRE.2015.2410176. URL: http://www.ncbi.nlm.nih.gov/pubmed/25769166.

[110] Abdulhamit Subasi. "Classification of EMG signals using PSO optimized SVM for diagnosis of neuromuscular disorders". In: *Computers in Biology and Medicine* 43.5 (2013), pp. 576–586. ISSN: 00104825. DOI: 10.1016/j.compbiomed.2013.01.020.

[111] Jie Liu, Dongwen Ying, and Ping Zhou. "Wiener filtering of surface EMG with a priori SNR estimation toward myoelectric control for neurological injury patients". In: *Medical Engineering & Physics* 36.12 (2014), pp. 1711–1715. ISSN: 13504533. DOI: 10.1016/j.medengphy.2014.09.008. URL: http://linkinghub.elsevier.com/retrieve/pii/S1350453314002410.

[112] S.W.R. Nijmeijer et al. "EMG coherence and spectral analysis in cervical dystonia: Discriminative tools to identify dystonic muscles?" In: *Journal of the Neurological Sciences* 347.1-2 (2014), pp. 167–173. ISSN: 0022510X. DOI: 10.1016/j.jns.2014.09.041. URL: http://linkinghub.elsevier.com/retrieve/pii/S0022510X1400639X.

[113] Winston De Armas, Khondaker a. Mamun, and Tom Chau. "Vocal frequency estimation and voicing state prediction with surface EMG pattern recognition". In: *Speech Communication* 63-64 (2014), pp. 15–26. ISSN: 01676393. DOI: 10.1016/j.specom.2014.04.004. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167639314000296.

[114] M.W. Kadous, Mohammed Waleed Kadous, and Supervisor Claude Sammut. "Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series". In: *PhD Thesis, School of Computer Science and Engineering, University of New South Wales* (2002).

[115] CyberGlove Systems LLC. *CyberGlove II*. http://www.cyberglovesystems.com/cyberglove-ii/. Accessed: 2018-07-23.

[116] Polhemus. *Polhemus Liberty Tracking System*. https://polhemus.com/motion-tracking/all-trackers/liberty. Accessed: 2018-07-23.

[117] Thalmic Labs. *Myo Gesture Control Armband*. https://www.myo.com/. Accessed: 2018-07-23.

[118] Stefano Pizzolato et al. "Comparison of six electromyography acquisition setups on hand movement classification tasks". In: *PLOS ONE* 12.10 (Oct. 2017), pp. 1–17. DOI: 10.1371/journal.pone.0186132. URL: https://doi.org/10.1371/journal.pone.0186132.

[119] Michael T. Wolf et al. "Gesture-based robot control with variable autonomy from the JPL BioSleeve". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, May 2013, pp. 1160–1165. ISBN: 978-1-4673-5643-5. DOI: 10.1109/ICRA.2013.6630718.

[120] Yale Song, David Demirdjian, and Randall Davis. "Continuous body and hand gesture recognition for natural human-computer interaction". In: *ACM Transactions on Interactive Intelligent Systems* 2.1 (Mar. 2012), pp. 1–28. ISSN: 21606455. DOI: 10.1145/2133366.2133371.

[121] Daniel Kelly, John Mc Donald, and Charles Markham. "Weakly supervised training of a sign language recognition system using multiple instance learning density matrices." In: *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society* 41.2 (Apr. 2011), pp. 526–41. ISSN: 1941-0492. DOI: 10.1109/TSMCB.2010.2065802.

[122] Pedro Neto et al. "Real-time and continuous hand gesture spotting: An approach based on artificial neural networks". In: *2013 IEEE International Conference on Robotics and Automation* (2013), pp. 178–183. DOI: 10.1109/ICRA.2013.6630573.

[123] Elham Shahinfard, Maher A. Sid- Ahmed, and Majid Ahmadi. "A motion adaptive deinterlacing method with hierarchical motion detection algorithm". In: *2008 15th IEEE International Conference on Image Processing*. IEEE, 2008, pp. 889–892. ISBN: 978-1-4244-1765-0. DOI: 10.1109/ICIP.2008.4711898.

[124] Emel Demircan et al. "Human Movement Understanding [TC Spotlight]". In: *IEEE Robotics & Automation Magazine* 22.3 (Sept. 2015), pp. 22–24. ISSN: 1070-9932. DOI: 10.1109/MRA.2015.2452171.

[125] Daehwan Kim, Jinyoung Song, and Daijin Kim. "Simultaneous gesture segmentation and recognition based on forward spotting accumulative HMMs". In: *Pattern Recognition* 40.11 (Nov. 2007), pp. 3012–3026. ISSN: 00313203. DOI: 10.1016/j.patcog.2007.02.010.

[126] Jinyoung Song and Daijin Kim. "Simultaneous Gesture Segmentation and Recognition based on Forward Spotting Accumulative HMMs". In: *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 1. IEEE, 2006, pp. 1231–1235. ISBN: 0-7695-2521-0. DOI: 10.1109/ICPR.2006.1056.

[127] Hee-Deok Yang, Stan Sclaroff, and Seong-Whan Lee. "Sign language spotting with a threshold model based on conditional random fields." In: *IEEE transactions on pattern analysis and machine intelligence* 31.7 (July 2009), pp. 1264–77. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.172.

[128] J.W. Deng and H.T. Tsui. "An HMM-based approach for gesture segmentation and recognition". In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 3. IEEE Comput. Soc, 2000, pp. 679–682. ISBN: 0-7695-0750-6. DOI: 10.1109/ICPR.2000.903636.

[129] J.H. Kim. "An HMM-based threshold model approach for gesture recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.10 (1999), pp. 961–973. ISSN: 01628828. DOI: 10.1109/34.799904.

[130] Shiguo Lian. "Automatic video temporal segmentation based on multiple features". In: *Soft Computing* 15.3 (Nov. 2009), pp. 469–482. ISSN: 1432-7643. DOI: 10.1007/s00500-009-0527-9.

[131] Bo Yao et al. "A fuzzy logic-based system for the automation of human behavior recognition using machine vision in intelligent environments". In: *Soft Computing* (Apr. 2014). ISSN: 1432-7643. DOI: 10.1007/s00500-014-1270-4.

[132] Hyun Kang, Chang Woo Lee, and Keechul Jung. "Recognition-based gesture spotting in video games". In: *Pattern Recognition Letters* 25.15 (Nov. 2004), pp. 1701–1714. ISSN: 01678655. DOI: 10.1016/j.patrec.2004.06.016.

[133] Oresti Banos et al. "Human activity recognition based on a sensor weighting hierarchical classifier". In: *Soft Computing* 17.2 (July 2012), pp. 333–343. ISSN: 1432-7643. DOI: 10.1007/s00500-012-0896-3.

[134] Maja J. Mataric. "Sensory-Motor Primitives as a Basis for Imitation: Linking Perception to Action and Biology to Robotics". In: ().

[135] Jounghoon Beh, David Han, and Hanseok Ko. "Rule-based trajectory segmentation for modeling hand motion trajectory". In: *Pattern Recognition* 47.4 (Apr. 2014), pp. 1586–1601. ISSN: 00313203. DOI: 10.1016/j.patcog.2013.11.010.

[136] Xianbin Cao et al. "Selecting Key Poses on Manifold for Pairwise Action Recognition". In: *IEEE Transactions on Industrial Informatics* 8.1 (Feb. 2012), pp. 168–177. ISSN: 1551-3203. DOI: 10.1109/TII.2011.2172452.

[137] Cuong Tran and Mohan Manubhai Trivedi. "3-D Posture and Gesture Recognition for Interactivity in Smart Spaces". In: *IEEE Transactions on Industrial Informatics* 8.1 (Feb. 2012), pp. 178–187. ISSN: 1551-3203. DOI: 10.1109/TII.2011.2172450.

[138] Holger Junker et al. "Gesture spotting with body-worn inertial sensors to detect user activities". In: *Pattern Recognition* 41.6 (June 2008), pp. 2010–2024. ISSN: 00313203. DOI: 10.1016/j.patcog.2007.11.016.

[139] Rung-Huei Liang and Ming Ouhyoung. "A real-time continuous gesture recognition system for sign language". In: *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE Comput. Soc, 1998, pp. 558–567. ISBN: 0-8186-8344-9. DOI: 10.1109/AFGR.1998.671007.

[140] Zhiyuan Lu et al. "A Hand Gesture Recognition Framework and Wearable Gesture-Based Interaction Prototype for Mobile Devices". In: *IEEE Transactions on Human-Machine Systems* 44.2 (Apr. 2014), pp. 293–299. ISSN: 2168-2291. DOI: 10.1109/THMS.2014.2302794.

[141] Daniel Kelly, John McDonald, and Charles Markham. "A person independent system for recognition of hand postures used in sign language". In: *Pattern Recognition Letters* 31.11 (Aug. 2010), pp. 1359–1368. ISSN: 01678655. DOI: 10.1016/j.patrec.2010.02.004.

[142] 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, June 2010, pp. 233–240. ISBN: 978-1-4244-7911-5. DOI: 10.1109/INSS.2010.5573462.

[143] Thomas Stiefmeier et al. "Wearable Activity Tracking in Car Manufacturing". In: *IEEE Pervasive Computing* 7.2 (Apr. 2008), pp. 42–50. ISSN: 1536-1268. DOI: 10.1109/MPRV.2008.40.

[144] Long-Van Nguyen-Dinh, Alberto Calatroni, and Gerhard Tröster. "Robust online gesture recognition with crowdsourced annotations". In: *The Journal of Machine Learning Research* 15.1 (Jan. 2014), pp. 3187–3220. ISSN: 1532-4435.

[145] Barbara Bruno et al. "Analysis of human behavior recognition algorithms based on acceleration data". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, May 2013, pp. 1602–1607. ISBN: 978-1-4673-5643-5. DOI: 10.1109/ICRA.2013.6630784.

[146] Barbara Bruno, Fulvio Mastrogiovanni, and Antonio Sgorbissa. "Wearable Inertial Sensors: Applications, Challenges, and Public Test Benches". In: *IEEE Robotics & Automation Magazine* 22.3 (Sept. 2015), pp. 116–124. ISSN: 1070-9932. DOI: 10.1109/MRA.2015.2448279.

[147] Gang Yu, Zicheng Liu, and Junsong Yuan. "Discriminative Orderlet Mining for Real-Time Recognition of Human-Object Interaction". English. In: *Computer Vision – ACCV 2014 SE - 4*. Ed. by Daniel Cremers et al. Vol. 9007. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 50–65. ISBN: 978-3-319-16813-5. DOI: 10.1007/978-3-319-16814-2{_}4.

[148] L. S. Lasdon et al. "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming". In: *ACM Transactions on Mathematical Software* 4.1 (Mar. 1978), pp. 34–50. ISSN: 00983500. DOI: 10.1145/355769.355773.

[149] M. Burke and J. Lasenby. "Pantomimic Gestures for Human–Robot Interaction". In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1225–1237. ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2475956.

[150] S. Calinon and A. Billard. "Incremental learning of gestures by imitation in a humanoid robot". In: *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*. Mar. 2007, pp. 255–262. DOI: 10.1145/1228716.1228751.

[151] D. Kulic, W. Takano, and Y. Nakamura. "Online Segmentation and Clustering From Continuous Observation of Whole Body Motions". In: *IEEE Transactions on Robotics* 25.5 (Oct. 2009), pp. 1158–1166. ISSN: 1552-3098. DOI: 10.1109/TRO.2009.2026508.

[152] J. F. S. Lin, V. Joukov, and D. Kulic. "Full-body multi-primitive segmentation using classifiers". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. Nov. 2014, pp. 874–880. DOI: 10.1109/HUMANOIDS.2014.7041467.

[153] Z. Lai et al. "Approximate Orthogonal Sparse Embedding for Dimensionality Reduction". In: *IEEE Transactions on Neural Networks and Learning Systems* 27.4 (Apr. 2016), pp. 723–735. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2015.2422994.

[154] P. Lin, J. Zhang, and R. An. "Data dimensionality reduction approach to improve feature selection performance using sparsified SVD". In: *2014 International Joint Conference on Neural Networks (IJCNN)*. July 2014, pp. 1393–1400. DOI: 10.1109/IJCNN.2014.6889366.

[155] Alex Shyr, Raquel Urtasun, and Michael I. Jordan. "Sufficient dimension reduction for visual sequence classification". In: *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*. 2010, pp. 3610–3617. DOI: 10.1109/CVPR.2010.5539922.

[156] Ho Gi Jung. "Support vector number reduction by extending iterative preimage addition using genetic algorithm-based preimage estimation". In: *Pattern Recognition Letters* 84 (2016), pp. 43–48. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2016.08.003. URL: http://www.sciencedirect.com/science/article/pii/S0167865516301970.

[157] H. S. Koppula and A. Saxena. "Anticipating Human Activities Using Object Affordances for Reactive Robotic Response". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.1 (Jan. 2016), pp. 14–29. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2015.2430335.

[158] R. Keys. "Cubic convolution interpolation for digital image processing". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.6 (Dec. 1981), pp. 1153–1160. ISSN: 0096-3518. DOI: 10.1109/TASSP.1981.1163711.

[159] B. Ni, P. Moulin, and S. Yan. "Order Preserving Sparse Coding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.8 (Aug. 2015), pp. 1615–1628. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2362935.

[160] F. Dalvi, H. Cate, and Z. Hussain. "Sign Language Recognition using Temporal Classification". In: *Stanford University* (2015).

[161] Y. Guo, Y. Li, and Z. Shao. "DSRF: A flexible descriptor for effective rigid body motion trajectory recognition". In: *2016 IEEE International Conference on Mechatronics and Automation*. Aug. 2016, pp. 1673–1678. DOI: 10.1109/ICMA.2016.7558815.

[162] X. Yuan et al. "Weighted Linear Dynamic System for Feature Representation and Soft Sensor Application in Nonlinear Dynamic Industrial Processes". In: *IEEE Transactions on Industrial Electronics* 65.2 (Feb. 2018), pp. 1508–1517. ISSN: 0278-0046. DOI: 10.1109/TIE.2017.2733443.

[163] E. Zahedi et al. "Gesture-Based Adaptive Haptic Guidance: A Comparison of Discriminative and Generative Modeling Approaches". In: *IEEE Robotics and Automation Letters* 2.2 (Apr. 2017), pp. 1015–1022. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2660071.

[164] T. Ende et al. "A human-centered approach to robot gesture based communication within collaborative working processes". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2011, pp. 3367–3374. DOI: 10.1109/IROS.2011.6094592.

[165] Nuno Mendes, Mohammad Safeea, and Pedro Neto. "Flexible programming and orchestration of collaborative robotic manufacturing systems". In: *IEEE 16th International Conference on Industrial Informatics INDIN 2018* (2018), pp. 913–918.

[166] Jonathan Alon et al. "A unified framework for gesture recognition and spatiotemporal gesture segmentation." In: *IEEE transactions on pattern analysis and machine intelligence* 31.9 (Sept. 2009), pp. 1685–99. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.203.

[167] D. Wu et al. "Deep Dynamic Neural Networks for Multimodal Gesture Segmentation and Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP.99 (2016), pp. 1–1. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2537340.

[168] Francisco Javier Ordonez and Daniel Roggen. "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition". In: *Sensors* 16.1 (2016). DOI: 10.3390/s16010115.

[169] Matthew Field et al. "Recognizing human motions through mixture modeling of inertial data". In: *Pattern Recognition* 48.8 (2015), pp. 2394–2406. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2015.03.004.

[170] Jixu Chen et al. "Switching Gaussian Process Dynamic Models for simultaneous composite motion tracking and recognition". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 2655–2662. DOI: 10.1109/CVPR.2009.5206580.

[171] Cristian A. Torres-Valencia et al. "Dynamic Hand Gesture Recognition Using Generalized Time Warping and Deep Belief Networks". In: *Advances in Visual Computing*. Ed. by George Bebis et al. Cham: Springer International Publishing, 2015, pp. 682–691. ISBN: 978-3-319-27863-6.

[172] X. Yuan et al. "Deep Learning-Based Feature Representation and Its Application for Soft Sensor Modeling With Variable-Wise Weighted SAE". In: *IEEE Transactions on Industrial Informatics* 14.7 (July 2018), pp. 3235–3243. ISSN: 1551-3203. DOI: 10.1109/TII.2018.2809730.

[173] Camille Monnier, Stan German, and Andrey Ost. "A Multi-scale Boosted Detector for Efficient and Robust Gesture Recognition". In: *Computer Vision - ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part I*. Springer International Publishing, 2015, pp. 491–502. ISBN: 978-3-319-16178-5. DOI: 10.1007/978-3-319-16178-5_34.

[174] Kuizhi Mei et al. "Training more discriminative multi-class classifiers for hand detection". In: *Pattern Recognition* 48.3 (2015), pp. 785–797. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.09.001.

[175] Geoffrey E. Hinton. "A Practical Guide to Training Restricted Boltzmann Machines". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 599–619. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_32. URL: https://doi.org/10.1007/978-3-642-35289-8_32.

[176] M. Safeea and P. Neto. "KUKA Sunrise Toolbox: Interfacing Collaborative Robots with MATLAB". In: (2018). URL: https://github.com/Modi1987/KST-Kuka-Sunrise-Toolbox.

[177] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (Mar. 1994), pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181.

[178] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[179] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[180] François Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015.

[181] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[182] Marco A.F. Pimentel et al. "A review of novelty detection". In: *Signal Processing* 99 (June 2014), pp. 215–249. ISSN: 0165-1684. DOI: 10.1016/J.SIGPRO.2013.12.026. URL: https://www.sciencedirect.com/science/article/pii/S016516841300515X.

[183] Xuemei Ding et al. "An experimental evaluation of novelty detection methods". In: *Neurocomputing* 135 (July 2014), pp. 313–327. ISSN: 0925-2312. DOI: 10.1016/J.NEUCOM.2013.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0925231213011314.

[184] Diederik P. Kingma et al. *Semi-supervised Learning with Deep Generative Models*. 2014. URL: http://papers.nips.cc/paper/5352-semi-supervised-learning-with-deep-generative-models.

[185] Jost Tobias Springenberg. "Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks". In: (Nov. 2015). arXiv: 1511.06390. URL: http://arxiv.org/abs/1511.06390.

[186] Ian J Goodfellow et al. "Generative Adversarial Nets". In: (2014). URL: https://arxiv.org/pdf/1406.2661.pdf.

[187] Wei Han et al. "A semi-supervised generative framework with deep learning features for high-resolution remote sensing image scene classification". In: *ISPRS Journal of Photogrammetry and Remote Sensing* (Nov. 2017). ISSN: 0924-2716. DOI: 10.1016/J.ISPRSJPRS.2017.11.004. URL: https://www.sciencedirect.com/science/article/pii/S0924271617303428.

[188] Bin Huang et al. "High-quality face image generated with conditional boundary equilibrium generative adversarial networks". In: *Pattern Recognition Letters* 111 (Aug. 2018), pp. 72–79. ISSN: 0167-8655. DOI: 10.1016/J.PATREC.2018.04.028. URL: https://www.sciencedirect.com/science/article/pii/S016786551830148X.

[189] Jie Li et al. "WaterGAN: Unsupervised Generative Network to Enable Real-time Color Correction of Monocular Underwater Images". In: *IEEE Robotics and Automation Letters* (2017), pp. 1–1. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2730363. URL: http://ieeexplore.ieee.org/document/7995024/.

[190] Xiaofeng Mao et al. "Semantic invariant cross-domain image generation with generative adversarial networks". In: *Neurocomputing* 293 (June 2018), pp. 55–63. ISSN: 0925-2312. DOI: 10.1016/J.NEUCOM.2018.02.092. URL: https://www.sciencedirect.com/science/article/pii/S0925231218302728.

[191] Yuan Yuan, Chunlin Tian, and Xiaoqiang Lu. "Auxiliary Loss Multimodal GRU Model in Audio-Visual Speech Recognition". In: *IEEE Access* 6 (2018), pp. 5573–5583. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2796118. URL: http://ieeexplore.ieee.org/document/8279447/.

[192] Yuki Saito, Shinnosuke Takamichi, and Hiroshi Saruwatari. "Statistical Parametric Speech Synthesis Incorporating Generative Adversarial Networks". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.1 (Jan. 2018), pp. 84–96. ISSN: 2329-9290. DOI: 10.1109/TASLP.2017.2761547. URL: http://ieeexplore.ieee.org/document/8063435/.

[193] Yang Li et al. "A Generative Model for category text generation". In: *Information Sciences* 450 (June 2018), pp. 301–315. ISSN: 0020-0255. DOI: 10.1016/J.INS.2018.03.050. URL: https://www.sciencedirect.com/science/article/pii/S0020025518302366.

[194] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional Image Synthesis With Auxiliary Classifier GANs". In: (Oct. 2016). arXiv: 1610.09585. URL: http://arxiv.org/abs/1610.09585.

[195] Jun-Yan Zhu et al. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: (Mar. 2017). arXiv: 1703.10593. URL: http://arxiv.org/abs/1703.10593.

[196] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: (Jan. 2017). arXiv: 1701.07875. URL: http://arxiv.org/abs/1701.07875.

SEGMENTATION ET RECONNAISSANCE DES GESTES POUR L'INTERACTION HOMME-ROBOT COGNITIVE

RESUME :

CETTE THESE PRESENTE UN CADRE FORMEL POUR L'INTERACTION HOMME-ROBOT (HRI), QUI RECONNAITRE UN IMPORTANT LEXIQUE DE GESTES STATIQUES ET DYNAMIQUES MESURES PAR DES CAPTEURS PORTATIFS. GESTES STATIQUES ET DYNAMIQUES SONT CLASSES SEPAREMENT GRACE A UN PROCESSUS DE SEGMENTATION. LES TESTS EXPERIMENTAUX SUR LA BASE DE DONNEES DE GESTES UC2017 ONT MONTRE UNE HAUTE PRECISION DE CLASSIFICATION. LA CLASSIFICATION PAS A PAS EN LIGNE UTILISANT DES DONNEES BRUTES EST FAIT AVEC DES RESEAUX DE NEURONES PROFONDS « LONG-SHORT TERM MEMORY » (LSTM) ET A CONVOLUTION (CNN), ET SONT PLUS PERFORMANTS QUE LES MODELES STATIQUES ENTRAINES AVEC DES CARACTERISTIQUES SPECIALEMENT CONÇUES, AU DETRIMENT DU TEMPS D'ENTRAINEMENT ET D'INFERENCE. LA CLASSIFICATION EN LIGNE DES GESTES PERMET UNE CLASSIFICATION PREDICTIVE AVEC REUSSIT. LE REJET DES GESTES HORS VOCABULAIRE EST PROPOSE PAR APPRENTISSAGE SEMI-SUPERVISE PAR UN RESEAU DE NEURONES DU TYPE « AUXILIARY CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS ». LE RESEAU PROPOSE A ATTEINT UNE HAUTE PRECISION DE REJET DE LES GESTES NON ENTRAINES DE LA BASE DE DONNEES UC2018 DUALMYO.

Mots clés : Robotique collaborative, segmentation non supervisée, apprentissage supervisée, apprentissage en profondeur

GESTURE SEGMENTATION AND RECOGNITION FOR COGNITIVE HUMAN-ROBOT INTERACTION

ABSTRACT :

THIS THESIS PRESENTS A HUMAN-ROBOT INTERACTION (HRI) FRAMEWORK TO CLASSIFY LARGE VOCABULARIES OF STATIC AND DYNAMIC HAND GESTURES, CAPTURED WITH WEARABLE SENSORS. STATIC AND DYNAMIC GESTURES ARE CLASSIFIED SEPARATELY THANKS TO THE SEGMENTATION PROCESS. EXPERIMENTAL TESTS ON THE UC2017 HAND GESTURE DATASET SHOWED HIGH ACCURACY. IN ONLINE FRAME-BY-FRAME CLASSIFICATION USING RAW INCOMPLETE DATA, LONG SHORT-TERM MEMORY (LSTM) DEEP NETWORKS AND CONVOLUTIONAL NEURAL NETWORKS (CNN) PERFORMED BETTER THAN STATIC MODELS WITH SPECIALLY CRAFTED FEATURES AT THE COST OF TRAINING AND INFERENCE TIME. ONLINE CLASSIFICATION OF DYNAMIC GESTURES ALLOWS SUCCESSFUL PREDICTIVE CLASSIFICATION. THE REJECTION OF OUT-OF-VOCABULARY GESTURES IS PROPOSED TO BE DONE THROUGH SEMI-SUPERVISED LEARNING OF A NETWORK IN THE AUXILIARY CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS FRAMEWORK. THE PROPOSED NETWORK ACHIEVED A HIGH ACCURACY ON THE REJECTION OF UNTRAINED PATTERNS OF THE UC2018 DUALMYO DATASET.

Keywords : Collaborative robotics, unsupervised segmentation, supervised learning, deep learning

ARTS ET MÉTIERS ParisTech

ParisTech INSTITUT DES SCIENCES ET TECHNOLOGIES PARIS INSTITUTE OF TECHNOLOGY