UCLouvain

epl

École polytechnique de Louvain

Uni/Bi-manual Wrist Gesture Interaction for one/two Users: Application to Kinemic

Author: **Chloé Chauvaux, Sarah Steeman**
Supervisor: **Jean Vanderdonckt**
Readers: **Christophe De Vleeschouwer, Arthur Sluÿters**
Academic year 2021–2022
Master [120] in Computer Science

Abstract

Wrist-based gesture interaction enables end-users to interact by mid-air gestures while keeping hands and fingers free for other manual tasks. This thesis aims to investigate how wrist-based gesture interaction could be made feasible with one wrist (unimanual) or two wrists (bimanual) for one end-user (in isolation) or two (in collaboration) and how to integrate it into an interactive application. For this purpose, we adopted a development life cycle made up of the following stages: design of a set of gestures from different sources (elicitation and literature survey), acquire a dataset of these gestures to train a recognizer, compare several recognizers on this dataset with various conditions, and mapping gestures to commands into an interactive application so that they can be handled in real time. This life cycle is applied specifically to the Kinemic device and for a computer-assisted presentation application.

Acknowledgements

We would like to thank our supervisor Professor Vanderdonckt as well as Arthur Sluÿters and Medhi Ousmer for their time to answer our questions and help us. We also thank all the people who took the time to come and participate as well as our families and friends for the support and advice they were able to give us throughout this work. Finally, we thank in advance those who will take the time to read our work.

Contents

Chapter 1

Introduction

1.1 Context of The Problem

In our modern era, the amount of data and the number of computer-based systems, such as information systems, interactive applications, are constantly growing. Therefore, it is a major challenge to master the interaction between the end-user and the computer-based system. Numerous projects and research on Human-Computer Interaction (HCI) have allowed one to design and classify computer interfaces using quality properties such as usability, learning curve, and communication optimally with a system via this interface [16]. For example, the ISO 9241-Part 11 standard [20] defines the quality factor "Usability" that appears in ISO 25010 [21] as "*the effectiveness, the efficiency, and the satisfaction with which specified users achieve specified goals in particular environments*". In this standard, usability is itself decomposed into quality sub-factors as follows: appropriateness recognizability, learnability, operability, user error protection, user interface aesthetics, and accessibility. While these characteristics are all important, we focus mainly on three of them[1]: *appropriateness recognizability* ("degree to which users can recognize whether a product or system is appropriate for their needs"), *learnability* ("degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use"), and *operability* ("degree to which a product or system has attributes that make it easy to operate and control").

The works on HCI show the importance of the means of interaction chosen between the man and the machine, and gesturing is one of these means of interaction that we can use in a very effective way. This means of communication is very important from a human point of view and from a human-machine interaction point of view because it is a more natural and freer means of interaction that easily adapts to different situations.

It is necessary to follow a procedure during the creation of gesture interfaces which allows one to correctly select the gestures and to test them to then integrate them into a system and an interface [38]. It is also necessary to choose a device that acquires the gestures. This can be a device worn directly by a user or a device that observes and records it. Each has its advantages and disadvantages.

[1] See https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?start=3

Studies [35] show that *user-defined gestures* are more efficient, more memorable, and require less effort than designer-defined gestures (which are defined by designers alone) and than system-defined gestures (which are defined by the software/hardware vendor and shipped with the operating system).

The objective is to correctly interpret the gestures chosen and defined by users in a system. After their acquisition and labeling, it is necessary to integrate the most ergonomic gestures into the system and make them recognizable. For this, there are several techniques and algorithms such as machine learning or pattern matching. Sometimes, it is also necessary to use filters and segmenters on the data flows acquired in real or delayed time in order to exploit the data and integrate them in the best way. Once the gestures are defined and integrated, we can go further and think about integrating gestures that require several devices and several users who would then make the gestures in a collaborative way if they have a common goal to achieve.

1.2 Mission Statement

1.2.1 Objective

The main objective of this thesis is to address the research question:

RQ="To what extent could we support unimanual and bimanual wrist-based user-defined gestures for a single user and for two users in collaboration?".

Wrist-worn devices [10, 51] support a wrist-based interaction in which the hands are in principle free [6, 7] and could be used to manipulate other objects such as physical objects. An exception is [50], where a wrist-worn device captures gestures performed by fingers on the hand palm. Inspired by ring-based gestures [11], we define *wrist-based gestures* as any action performed with a wrist-worn wearable device that causes a detectable change in the position and / or orientation of the device in a reference system located on the user's body.

Based on Guiard's taxonomy of human movement [14] used in gestural interaction [41], *unimanual gestures* are those involving a single wrist and *bimanual gestures* involving two. In the *single-user* configuration, the user can perform gestures with one or two wrist-worn devices, whereas in the *two-user* configuration, a collaborative situation occurs where each user possesses one or two wrist-worn devices and acts in isolation or in collaboration with the other user.

These gestures will be performed with a device around the wrist, and one of the objectives will be to use these gestures to create and interact with an application using a selected wrist-worn device, the Kinemic device [Figure 1.1]. The use of this kind of device and application saves time because the person with the device does not have to switch between two different devices. Furthermore, the fact that the device is on the wrist allows the hands to be available for other activities at the same time. The wristband also does not require you to be in front of a screen to interact with it, which is the space for movement and activity. Nevertheless, this kind of interaction with the wrist is said to be more natural,

but from the user's point of view it is not always the case, and retrieving this kind of data for use in an application can be complicated for developers.

Figure 1.1: Kinemic device interacting with an application.

As far as interaction with the wrist is concerned, there is already a great deal of work done using devices such as the smartwatch with excellent results [49]. But these have some limitations compared to the device with which we want to achieve the objective of making uni- and bi-manual gestures with one or two users.

- Users typically own only one watch because it is not necessary or even useless to own two. The user may not want to buy a second one since it doubles the price and as the functionality provided by the second watch will duplicate the first one.
- While a smartwatch has more functionality requiring looking at its small screen, it can be cumbersome, tedious, and heavy when a smartwatch should be worn on each hand.
- Gesture recognizers working on smartwatches are based on machine learning or neural network systems in which the classifier is trained on a large number of gesture templates that is not changed afterwards: the gesture vocabulary is pre-defined and does not change over time, thus preventing the end-user from changing the gestures, or even personalising them for her own usage.
- A smartwatch is often unique to one person and is not intended to exchange data or commands with another smartwatch belonging to another user. It is therefore very difficult to achieve a collaborative situation unless the raw data are made accessible to each other and synchronised in some way.

1.2.2 Working Hypotheses

- In this thesis we will use the Kinemic band device because this wearable device fits the needs of our work. Indeed, it is an off-the-shelf, lightweight, and affordable

device making its raw data accessible.

- We will focus on a single-user configuration with one or two devices and on a double-user configuration as a starting point for investigating collaboration. Therefore, our practical experiences will be limited to two people, a number that could increase with more bracelets made available.
- We will consider gestures that can be performed with one or two hands in any configuration. Those that can come from one or two different people in a collaborative situation, i.e. one person with one or two wristbands or two people with one wristband each. The gestures we will deal with are not limited to system-defined gestures, i.e. those delivered as standard with the Kinemic or those given by the operating system, but to user-defined ones.
- To satisfy this requirement, we assume that pattern matching classifiers will be used because they allow patterns to be modified without having to redo everything as soon as one is added. This is in contrast to other approaches such as neural networks where modifying a single gesture requires a complete re-training, re-modelling, etc. which can be very costly in terms of time, memory space and complexity. An exception exists when this re-training can be performed in real-time, but this approach is only in its infancy and is very sophisticated.

1.2.3 Research Method

In this section, we will briefly describe the research methodology of our thesis. More details will of course be provided in the following chapters of this work.

State of the art. To begin with, we need to establish a state of the art of the existing concepts concerning our research and experiences. It should include information about the existing concepts of gesture-based user interface, the device we are using, gesture classification, collaboration and the framework we are going to work with.

Benchmark methodology. We will carry out our tests by successive analysis and by decomposition into sub-problems. By successive analysis we mean that we will test our results as we go along and adapt the rest of our tests according to them. We will also break down the tests into several sub-problems in order to be able to compare the results obtained between them.

1.3 Overview of the Thesis

This section contains a summary of the information that each chapter of this thesis addresses. This provides an overview and allows you to find the desired information more quickly and to better understand the structure of the thesis.

Chapter 2, entitled "State-of-the-Art of Gesture Recognition". This chapter contains an introduction to gesture-based user interfaces. It gives an overview of the existing concepts that we use in the thesis concerning gesture recognition to install relevant background knowledge. It develops more precisely in a theoretical way the device, the approaches, the classifications, the definitions as well as the algorithms that we used.

Chapter 3, entitled "Gestures Dataset and Algorithms". This chapter is dedicated to the technical and practical aspect. It contains explanations and interpretation of the data of the Kinemic device and also the adaptation of a recognizer to it. It develops the steps for the creation of the dataset, the justification of the chosen gestures as well as its acquisition and its complete and detailed description in English and their classification according to those considered relevant according to the different configurations explored. These configurations are also detailed according to the theory.

Chapter 4, entitled "Benchmarking of Kinemic Gesture recognizers". This chapter is dedicated to the comparative analysis of our dataset. It develops the different parameters used and provides the observed results as well as their interpretation. This allows to gather information about the general performance of the device as well as the performance of the different recognizers in order to make a comparative study and to select the best choices.

Chapter 5, entitled "Bimanual Part" This chapter explores the bimanual part of acquired gestures by developing from a theoretical point of view the different possible and relevant configurations to explore.

Chapter 6, entitled "Application" This chapter contains the description of the development of an application to support and put into practice our results. It develops our choices of algorithms and parameters as well as the technical details. It also describes and supports our choices concerning the gestures and the application. It is illustrated with screenshots of the developed interface

Chapter 7, the conclusion. This will be a summary of the observations and results achieved in this thesis. We will also identify the limitations of our work as well as suggestions for overcoming them, future work and possible improvements.

Chapter 2

State-of-the-Art of Gesture Recognition

Before diving into the heart of the matter, we first define the important concepts that we are going to develop as well as all that exists on them and what has already been done in the field. This will provide a solid foundation on which to build. To do this we will describe the main topic of gesture-based user interface as well as the main device used but also those that could compete with it like the smartwatch. We will also define gestures to avoid confusion on the subject and several of the classification categories that exist to differentiate them. Since we are going to analyze unimanual and bimanual gestures, we will also talk about collaboration and the configurations that already exist to classify them. Finally, we will talk about the QuantumLeap Framework and its recognizers and segmenters that will be useful for our different tests and implementations.

2.1 Gesture-based User Interfaces

Two/three-dimensional (2D/3D) gesture-based User Interfaces (UIs) [28] promise a natural and intuitive interaction [17] as they rely on movements performed by the human body, which are assumed to be more natural and easier to remember than artificially determined commands. Gesture-based UIs typically fall into three categories, depending on the number of gesture dimensions and the sensor used to capture and recognize the gesture:

- *2D (nearly-)touch-based gestures*: the gesture is performed on a sensitive surface, such as a trackpad [Figure 2.1], a touchscreen or a touchable surface (*e.g.,*in Gambit [43]), restricting movement to a series of temporally sequential points (x_i, y_i, t). Gestures performed on a spatial object, such as a tangible object, remain in this category even if the object is spatially moving. These gestures typically require a contact-based or a close-by sensor, when the gesture could stay close to the surface, but not touching, such as for a grazing gesture. Such gestures could also be augmented with a third dimension, such as pressure, contact surface, or temperature.

Figure 2.1: Trackpad.

- *3D vision-based gestures*: the motion gesture refers to any 3D movement that uses a 3D object, a pose, and a motion to create a gesture interaction by relying on a sensor equipped with computer vision algorithms, such as the Leap Motion Controller, the MS Kinect [Figure 2.2], the Intel RealSense cameras, or the webcam of a laptop [13].

Figure 2.2: MS Kinect.

- *3D wearable gestures*: most portable smart devices such as smartwatches [32, 37, 45], smartphones, tablets and laptops have build in sensors that measure the acceleration and orientation of the device in space. These sensors include accelerometers, gyroscopes, and magnetometers. In the most recent generation of portable smart devices, the output from these sensors has been combined into a single IMU (Inertial Measuring Unit) device, which provides methods for self-calibration and qualified motion data in a 3D space. The result is drift-free data that track the 9DoF of the device in 6 directions of movement (up/down/in/out/left/right) and 3 axes of rotation (x_i,y_i,z_i). With ring devices, w_i are even used for quaternions.

2.2 Kinemic

In this thesis, we will use a device that allows User Interfaces gestures based on the category 3D wearable gestures. We have chosen a wearable bracelet called the Kinemic Band that was created by the German Kinemic Industry [Figure 2.3]. This device is a completely freehand and mobile interaction system. It allows remote control through gestures and text input. It uses inertial sensors that are almost unaffected by the environment. Its sensors are based on accelerometers and gyroscopes, which provide direct and robust signals. However, it does not give access to raw data but provides high-level gesture detection. Currently, 12 gestures can already be used with the bracelet [Table 2.1] illustrated with designs from the Kinemic website [1], and the system can also recognize written text in the air. The company describes it as a product that frees you from the keyboard and touch screen and allows you to remotely control all kinds of digital devices such as tablets, PCs, smartphones, etc

Figure 2.3: Kinemic Band.

Gesture name	**Gesture description**	**Gesture Illustration**
Swipe up	Extend your index and middle fingers and then move your wrist up.	
Swipe down	Extend your index and middle fingers and then move your wrist down.	
Swipe left	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the left.	
Swipe right	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the right.	
Check mark	Make a V in the air.	
Cross mark	Make a cross in the air.	

Gesture name	Gesture description	Gesture Illustration
Circle clockwise	Make a circle clockwise.	
Circle counterclockwise	Make a circle counterclockwise.	
Rotation right-left	Start with a vertical orientation of your hand so that you can see the thumb on top of your hand. Turn your hand 90 ° to the left so that you can see the knuckles and then back to the starting position in one swift movement.	
Rotation left-right	Begins in a horizontal position so that your palm points to the floor. Rotate the wrist 90 ° clockwise so that your thumb points up and then back to the starting position, again all in one swift motion.	
Eartouch left	Begin the hand movement at the hip and simply move the hand to left ear.	
Eartouch right	Begin hand movement at the hip and simply move the hand to the right ear.	

Table 2.1: Description and illustration of existing Kinemic gestures.

More technically, the Kinemic bracelet is equipped with a MMS MetaMotionS developed by MBIENTLAB. This device contains different sensors, such as the BMI270 that contains the accelerometer and gyroscope. It also has memory and a rechargeable battery. This device allows the bracelet to be used with bluetooth and has a led light that indicates whether the bracelet is connected or not. This sensing solution is ideal for streaming data over long periods of time and can be retrieved in different formats and devices.

There are two main articles that talk about this device. The first one [3] is entirely dedicated to it and was written by four Kinemic GbR. It presents the bracelet and its creation and also explains its advantages compared to existing products. In particular, they highlight the fact that the bracelet will measure movements more directly compared to systems equipped with a camera that will capture a whole scene. In the article, they also explain a demonstration they did in which participants could interact with different

applications using the bracelet on a head-mounted display. This experiment could be easily performed as the wristband is user-independent, the participant only needs to put the wristband on like a watch, and no calibration or adaptation phase is required. The second article [13] that mentions the bracelet is more general and talks about experiences with Multi-Sensor Fusion in Industrial Assistance Systems. So, they also discuss other devices that enable human-machine interaction. They talk about the Kinemic in the section about gesture detection. They describe the device in a general way and provide a small evaluation of it. The latter mentions that it is easily integrated into a multi-sensor system and that it is rather efficient. On the Kinemic website, you can also find examples of projects that have already been implemented. For example, the KARL project [Figure 2.4] is a project in which a method was set up to count and verify the number of screws in a workpiece using Kinemic. The aim was to find the best way to interact with the computers and to facilitate the assistance system to also ensure quality.

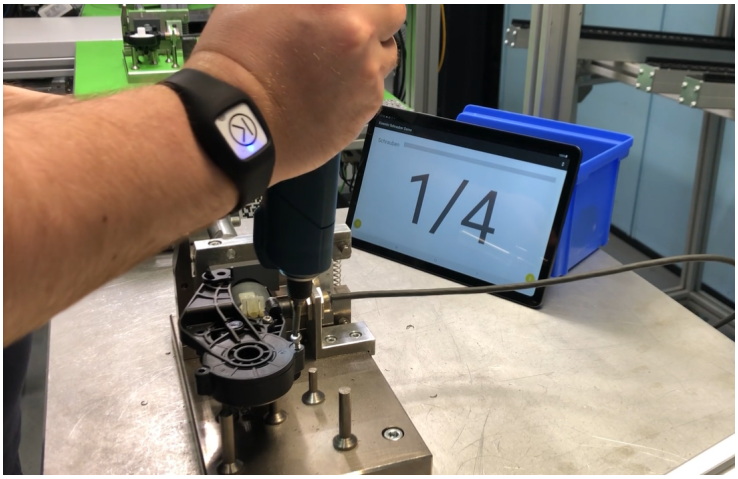


Figure 2.4: KARL Project.

2.3 Other devices

2.3.1 Smartwatch

There exist other devices very similar to the Kinemic that recognize gestures. One that can come closest is certainly the smartwatch [Figure 2.5]. Smartwatches are more and more present in our daily life and already allow to interact with many applications. Like the Kinemic band, they can allow a person to interact with a hands-free system. In the article [37], a Systematic Review of Literature was conducted to identify which gesture recognition methods already existed with this device and what their advantages were. Many gesture recognition methods have already been proposed for the smartwatch. Some are equipped with a camera, others allow for object recognition, text input or even hands-free interaction using arm gestures. Some methods use Dynamic Time Warping or Linear Discriminant Analysis. They are based on acceleration and neural networks, or they use an accelerometer and gyroscopic data like Kinemic.

Figure 2.5: Smartwatch.

The advantages of such devices are that unlike those using cameras, light or the user's position are no longer constraints as smartwatches are essentially based on sensors such as accelerometer, gyroscope and pressure sensors. They therefore allow a more free and natural interaction and can be easily adapted for people with reduced abilities [32].

2.4 Gestures

A gesture can be explained in many different ways but for the purposes of this paper it will be defined as a movement of the body in 1D, 2D or 3D that can be detected by a digital device without the use of a traditional tool such as a mouse or stylus [42].

2.5 Gestures Classification

There are different ways of classifying gestures into different types. For the purpose of our dissertation we decided to use Kendon's, Neill's and a more recent and adapted one by Aigner.

2.5.1 Kendon's continuum

Gesticulation. Gesticulation is one of the most common types of gesture. The gesture can be made by any part of the body and usually accompanies a speech. For example, the gestures made by the professor during a university speech or when a person tells a story [15, 24].

Speech-framed gestures. Speech-framed gestures are gestures that are part of a sentence and occupy a place in it. This type of gesture completes the structure of a sentence. This can be done by replacing the complement of the verb, for example, by pointing to the object concerned by our speech and with our speech.

Emblems. Emblems are gestures that represent conventional signs, such as a thumbs up to approve something. They are therefore gestures that may differ from culture to culture.

Pantomime. Pantomime gestures are a gesture or sequence of gestures that build a narrative line and represent a story. For example, by pointing to an object but this time without saying anything.

Signs. Signs gestures are words that correspond to a sign language. They have their own linguistic structure and grammar.

2.5.2 McNeill type

Deictic. Deictic gesture represents any gesture where something is pointed, usually with his index finger. It can point to something physical present in the place or something invisible that is referred to [15].

Beat. Beat gestures are biphasic movements so named because they represent a beating time. These are rhythmic movements that do not particularly represent anything but accompany the dialogue. For example, a teacher who snaps his fingers when something important needs to be emphasized.

Iconic. Iconic gestures usually represent a concrete object or action by its resemblance to them. They are gestures that often have a conventional meaning, which makes them recognizable. For example, support your point by imitating an object bent "this way".

Metaphoric. Metaphoric gestures represent abstract content or ideas. Therefore, they are not limited to concrete things. The principle is not to represent the object, but rather the idea behind it. For example, a person presents an idea that is not concrete by pretending to handle it.

2.5.3 Aigner type

Pointing. Pointing gestures are used as the name suggests to point to objects or directions. Things do not necessarily have to be pointed with the index finger, they can be pointed with any finger or even the whole hand [2].

Semaphoric. Semaphoric gestures are associated with specific meanings that are often dependent on the author of the gesture. There are three subcategories, the first is static semaphorics which represent a specific posture. The second is dynamic semaphorics, which construct information through their temporal aspect. The last one is the semaphoric strokes, which are like the dynamic ones represented by a movement of the hand, but which are a simple movement.

Pantomimic. Pantomimic gestures represent a specific task that is performed. The task is carried out without the necessary objects but with everything as if they were present. They are usually made up of several steps.

Iconic. Iconic gestures are intended to represent objects by a certain position or movement. There are two subcategories: static and dynamic. Static iconic gestures are represented by a spontaneous posture. Dynamic iconic gestures form a path or shape by slow movement.

Manipulation. Handling gestures form a relationship between the movement of the author and the object. They build up movement in a feedback loop.

Examples are shown below in [Figure 2.6]

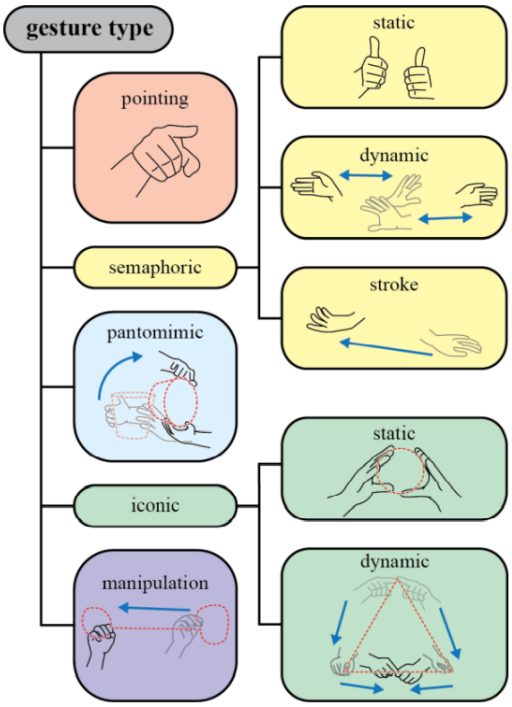


Figure 2.6: Aigner gestures classification, from [2].

2.5.4 Jahani's gesture vocabulary

Jahani has defined a vocabulary of gestures used in the design of human-computer interfaces. He developed a gesture code described in the table below [22].

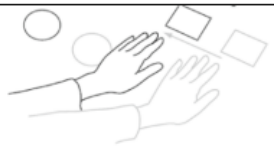
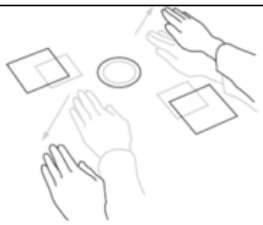
Gesture code	Gesture name	Gesture function	Example
1	Static Pose	Hand pose is held in one location.	
2	Dynamic Pose	Hand pose changes in one location.	
3	Static Pose and Path	Hand pose is held as hand moves.	
4	Dynamic Pose and Path	Hand pose changes as hand moves.	
5	One-point hold	Static Pose with One Finger.	
6	One-point Path	Static Pose and Path with One Finger.	

Figure 2.7: Jahani's gesture code, from [22].

He also proposes more elaborate codes that are a combination of the basic codes he defined earlier. This allows to better represent the practical gestures made in real life.

Gesture code	Gesture name	Gesture function
1-3	Co-Static and Static Path	Hand pose is held then it changes in one location and moves
2-1	Co-Dynamic and Static Pose	Hand pose changes in one location then it is held there
2-3	Co-Dynamic and Static Path	Hand pose changes, is held in one location then moves
2-4	Co-Dynamic and Dynamic Path	Hand pose changes in one location, then moves and changes again
3-2	Co-Static and Dynamic Path	Hand pose changes as hand moves, then pose changes and hold there
4-1	Co-Dynamic Pose-Path and Static Hold	Hand pose changes as hand moves, then hand stops and pose is held
4-2	Co-Dynamic Pose-Path-Pose	Hand pose changes as hand moves, then hand stops and pose changes
4-3	Co-Dynamic Pose-Path and Static Pose-Path	Hand pose changes as hand moves, then pose is held and hand keeps moving
6-4	Co-Finger and Hand Static Path	Static Pose and Path with one finger then changes to hand pose and keeps moving
2-5	Co-Finger Dynamic Pose and Hold	Hand pose changes in one location then changes to one finger Static Pose
2-6	Co-Finger Dynamic Pose and Path	Hand pose changes in one location then changes to one finger and moves

Figure 2.8: Jahani's combined gesture code, from [22].

2.6 Collaboration

Collaboration between two or more people means that they work together to achieve a common goal. Collaboration should not be confused with cooperation, as cooperation also means working towards the same goal, but not necessarily at the same time, and the different parties can carry out the task separately.

2.6.1 Collaborative Gesture

A collaborative gesture must always involve at least two people, because it is not supposed to be possible to carry it alone. These two people must work together to achieve a common goal. It is the totality of the gestures of the different participants that will allow the action to be carried out and not just one of the gestures [33].

In collaborative gestures there is the concept of a gesture combination which consists of having one gesture describing, for example, the context and the second one aiming at a command. In the article [8], we have the example of a gesture that represents light and the second that allows turning it on. We could also have a combination of an action and a parameter. For example, the first gesture represents the rotation and the second represents the angle of rotation.

2.6.2 Allen Temporal Algebra

Allen's interval algebra represents thirteen possible configurations between two time intervals as can be seen in the [Figure 5.2], the first six can be used symmetrically. They allow to order the gestures between two participants within the framework of collaborative gestures [11].

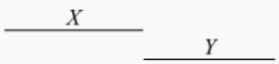
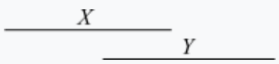
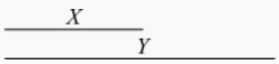
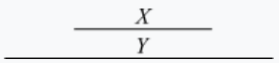
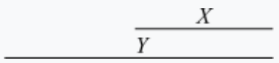
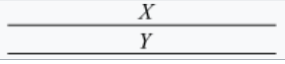
Relation	Illustration	Interpretation
$X < Y$ $Y > X$		X precedes Y Y is preceded by X
$X\ \mathbf{m}\ Y$ $Y\ \mathbf{mi}\ X$		X meets Y Y is met by X (*i* stands for *inverse*)
$X\ \mathbf{o}\ Y$ $Y\ \mathbf{oi}\ X$		X overlaps with Y Y is overlapped by X
$X\ \mathbf{s}\ Y$ $Y\ \mathbf{si}\ X$		X starts Y Y is started by X
$X\ \mathbf{d}\ Y$ $Y\ \mathbf{di}\ X$		X during Y Y contains X
$X\ \mathbf{f}\ Y$ $Y\ \mathbf{fi}\ X$		X finishes Y Y is finished by X
$X = Y$		X is equal to Y

Figure 2.9: Allen temporal algebra configurations.

2.7 Recognizer

2.7.1 Jackknife

Jackknife is a recognizer that can be used for 2D or 3D gestures and can be performed with only the hand or by the whole body. The different coordinates retrieved from the gestures are converted into different direction vectors of unit length. This recognizer then uses dynamic time warping to find out which template corresponds to each candidate. Dynamic time warping is defined as a measure of dissimilarity. It works by warping time series to find the best alignment. After this step, the recognizer will apply correction factors to adjust the score for additional dissimilarities [19, 40].

2.7.2 $P3+

$P3+ is a recognizer that is a more accurate version of the $P3 recognizer which is itself an adaptation of Vatavu's $P+ [48]. It can be used for multi-stroke gestures in 3D. Within the algorithm the gestures are represented as sets of unordered points, called point clouds. Gesture recognition takes place in two phases: normalization and cloud matching. Normalization is the same process used by other $-algorithms. Cloud matching consists of associating the points of a gesture to the points of the point cloud of a template by associating each point of the template to one or more points of the gesture. The resulting distance is a sum of the Euclidean distances between all pairs of matching points and takes into account the connections between consecutive points. The recognized template is then the template that minimizes this resulting distance. The recognizer also applies the principle of early abandonment of templates as soon as the distance exceeds the shortest distance currently stored [47].

2.7.3 μV

μV is an Articulation-, Rotation-, Scale-, and Translation-invariant multi-stroke recognizer [31]. It combines several existing recognizers, $P+ and !FTL and uses respectively the cloud matching and the local shape distance of these. As explained in $P3+, cloud matching is the principle of matching the points of a first cloud with the closest from the second cloud. The local shape distances give a measure of dissimilarity and the sum of these distances for each pair of vectors gives the dissimilarity between two gestures. The algorithm returns the class in the model that had the lowest overall dissimilarity score [29].

2.7.4 μF

μF is a (Rotation-), Scaling- and Translation-invariant multimodal recognizer that better handles the addition of new templates and classes in interactive time. It was introduced and developed by Nathan Magrofuoco. The algorithm uses the local distance shape dissimilarity function and derives a series of distances, also called features, from it. Their accuracy is then used to compute the dissimilarity score between two gestures. The recognizer is configured in runtime to define the relevant paths, called articulations, which will be the paths that achieve the highest possible accuracy [29].

2.8 Segmenter

One of the challenges with this type of device, such as the Kinemic is that the data received are continuous data. To recognize the different gestures, it is necessary to be able to segment these data to be able to differentiate the gestures among all these data received. To do this, a segmenter must be used to differentiate gestures from parasite data in the data stream [47].

2.8.1 Threshold

Threshold-based segmentation consists of sending the received data to the recognizer as soon as one or more properties of the point are outside a certain limit [4].

2.8.2 Sliding Window

The sliding window segmentation technique consists of creating windows, i.e. buffers of fixed size at regular intervals. The data frames are added to this buffer and all the fixed frame numbers of the window data are sent to the recognizer [4, 19, 47].

2.8.3 Machete

Machete is also a recognition tool that can handle continuous streams of data. It has the advantage that it works independently of the device and is effective with a single training sample. It uses traditional continuous dynamic programming with the addition of a new dissimilarity measure to align the incoming data. It is one of the best recognizers to achieve high and fast recognition of personalized gestures [18].

2.9 QuantumLeap Framework

To carry out our various tests with the Kinemic and to allow the bracelet to interact with an application we used QuantumLeap [47]. The latter is a framework allowing the design of user interfaces based on the Leap Motion Controller. Thanks to its interface, it is possible to configure a pipeline with different modules allowing us to choose the recognizers, segmenters to be applied as well as to manage their correspondence with the functions of an application. It contains everything necessary for gesture recognition and provides a JavaScript API that allows developers to associate gestures with actions. QuantumLeap provides the ability to use already implemented modules but also to add new ones that better match what the developer needs.

2.10 Summary

Therefore, we have reviewed the various important concepts that already exist on the subject and that will be necessary for the rest of this work. Thanks to this part, we will be able to better justify the choice of the device we have chosen. We will also be able to choose the most relevant gestures and classify them thanks to the different classifications that exist on the subject. Finally, we will be able to use the different recognition and segmentation techniques to test our different gestures, as well as the application that will be created and realized using the QuantumLeap framework.

Chapter 3

Gestures Dataset and Algorithms

This chapter will be dedicated to describing the dataset that we recorded, as well as the context in which this was done and the device used for the recording. For this we would describe the Kinemic, as well as the structure of the data it returns and how we processed it. We will then explain the whole process of creating the dataset and its description, including the environment in which it was recorded, the people who participated in it, the gestures we recorded, and their different possible configurations. The adaptations that had to be made to certain algorithms will also be discussed in this section.

3.1 Kinemic

As an explanation for this work, we decided to use the Kinemic band device. Although there are many other devices for human-computer interaction, we decided to focus on the Kinemic because it is a small, easily transportable and usable device. In addition, it is lightweight and not at all intrusive as it is worn like a watch so it is quite possible to do other things while wearing it. Its choice is therefore justified by the fact that it is space-saving and flexible and that the raw data can be easily accessed. So, it is possible to acquire gestures without having to go through the manufacturer. In addition, there is the possibility of purchasing and using several per person.

3.2 Data Structure

The Kinemic's sensors are based on accelerometer and gyroscope. For each point the device returns 7 data. The first 3 are from the accelerometer, the next 3 are from the gyro sensor and the last one is the timestamp [3]. These sensors are widely used in motion detection, as they can identify a change in movement or orientation.

Accelerometer. The accelerometer is used to retrieve data from x, y, and z. It measures the acceleration of the device itself, i.e. the acceleration of the movement in its own frame and not in a coordinate system. In our case the acceleration is therefore given by 3 vector components which represent the net acceleration. The accelerometer reacts to gravity and can therefore also identify certain rotational changes. This sensor is also used on many devices, such as smartphones. One of its advantages is that it does not require adaptation between users [9, 26, 27].

Gyroscope. The gyroscope sensor also allows for the recovery of x, y, and z data. It measures and maintains the orientation, inclination and angular velocity of the device. This sensor allows us to measure the movements of an object and, in our case, the wrist. It uses gravity to determine the orientation of the object and measures the angular velocity, which is the change in angle of rotation per unit of time. Gyroscopes which, as in our case, use 3 axes are generally also accompanied by a 3-axis accelerometer as in the Kinemic. This makes for the most complete system possible. The gyroscope itself is nevertheless more accurate than the accelerometer, which only measures linear motion.

Timestamp. The timestamp is represented by an integer. Each set of coordinates has its own timestamp. It represents a numerical counter and allows us to know the order in which the coordinates have been recorded.

The data of the Kinemic that are retrieved during the registrations of the database are stored in a JSON file that has the following structure:

```
{
    "name": string,
    "subject": integer,
    "paths": [
        {
            "label": string,
            "strokes": [
                {
                    "id": integer,
                    "points": [
                        {
                            "coordinates": [
                                xa,
                                ya,
                                za,
                                xg,
                                yg,
                                zg
                                ],
                                "t": integer
                        }
                    ]
                }
            ]
        }
    ]
}
```

The name field contains the name of the gesture, the subject contains the number of the person performing the gestures, the label contains 'Kinemic_sensor' for all of them, and the id contains 0 for all gestures as well. In the coordinates we find xa, ya and za which are the 3 axes of acceleration and xg, yg and zg which are those of the gyroscope. The coordinates are followed by the integer t, which contains the timestamp. For each point, new coordinates and timestamp fields are created to store the data. A new JSON file is therefore created for each new gesture recorded.

3.3 $PN+

In the Chapter 2, we briefly described what some of the recognizers such as $P3+ consist of. Unfortunately, this last one only allows to process three data (x, y and z), nevertheless, in our case the Kinemic returns six data for each coordinate. Therefore, we decided to test $P3+ on the three acceleration data and the three gyroscopic sensor data, but to push the experiment to the end we created a second algorithm that we named $PN+. This recognizer works exactly the same way as $P3+ with the only difference being that it allows us to use the six data returned by the bracelet and not only three. Its full code can be found in the Appendix A.1.

3.4 Creation of the Dataset

Environment. The experience took place for the majority of participants in the same quiet office. They sat in front of a desk with a computer in front of them [Figure 4.1]. On this computer, we showed them a presentation containing a video and an explicit name for each gesture so that each person had the same model on which to reproduce each gesture [Figure 3.2]. On the same desk was a second computer whose screen they could not see and which allowed the gestures to be recorded.

Figure 3.1: Reconstruction of the recording environment with a participant.

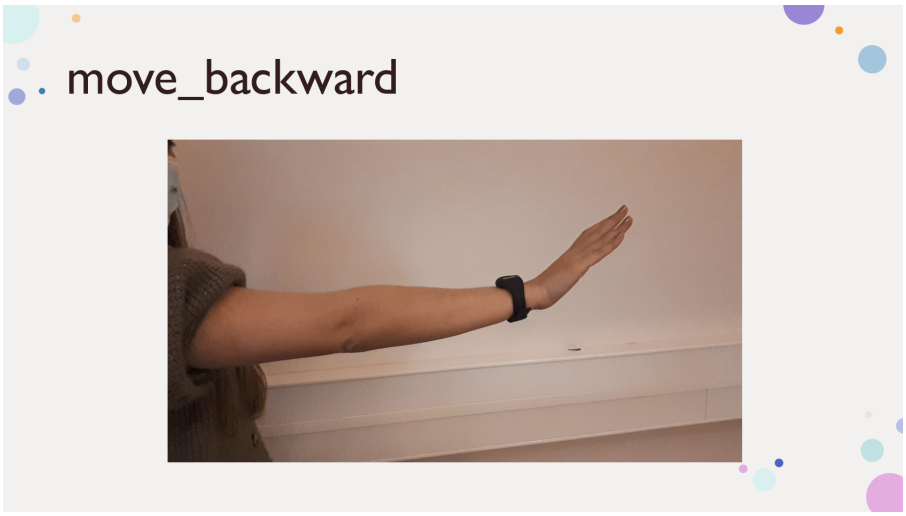


Figure 3.2: Example of a slide presented to participants for the move backward gesture.

Consent. Before the experiment started, participants were asked to fill in two documents. The first was a consent form to participate in the experiment. The second was a questionnaire in which they were asked to fill in a number of details about themselves. All these forms can be found in Appendix E. First of all, they had to provide their first name, last name, age and gender. Secondly, they had to provide their profession, field of activity, and study. Finally, they had to evaluate their frequency of use on a scale of 1 (disagree) to 7 (agree) concerning the computer, the smartphone, the tablet, the console, and the Kinect and specify whether or not they were familiar with the Kinemic and if so, the use they made of it on a scale of 1 (rarely) to 7 (frequently).

Procedure. The Kinemic device was first introduced to participants who were not familiar with it. Then, a device was first given to attach to their right wrist. The first slide was then shown to them, and once they had understood the movement, it was recorded twice for all unimanual gestures. Then, a second wristband was given to them to put on their left wrist so that they could perform the bimanual gestures in the same way. At the end of the experiment, participants were asked to complete a post-study system usability questionnaire in which they evaluated their satisfaction on a scale of 1 (totally disagree) to 7 (totally agree) on different points of the experiment and if they wished to list three positive and three negative aspects. This form can be found in Appendix E. Each experiment took about 30 minutes per participant.

3.5 Description of the Dataset

Participants. A sample of 30 participants volunteered to record the dataset. The minimum age was 19 years and the maximum age was 75 years and the mean age was 33 years (SD = 14.38). The gender distribution is 57% women and 43% men. 47% of the participants have a technical profile, that is, they work or study in the field of IT. They evaluated their use of technology by indicating on a scale of 1 (disagree) to 7 (agree) whether they use it frequently or not and the average of their use of the phone and computer was 5.9. Finally, 27% of the participants indicated that they were aware with the Kinemic device prior to the recording.

Gestures. We have defined the gestures according to three configurations: a unimanual configuration, a bimanual configuration and a unimanual collaborative configuration. In the associated table above there is a description of the gestures and their classification according to the classifications of Kendor, McNeill and Aigner. And the last column corresponds to the laterality [14] of the gestures. For the unimanual gestures, we did not specify laterality because they were all performed with the right hand and therefore there is no laterality to specify. The Kinemic comes with 12 predefined system gestures that we have recorded and called: "swipe left", "swipe right", "swipe up", "swipe down", "check mark", "cross mark", "circle clockwise", "circle counterclockwise", "rotation left right", "rotation right left", "eartouch left", "eartouch right".

We have invented the other gestures and we have chosen them in order to diversify the types of gestures according to the chosen classifications to be able to exploit all the possibilities of the Kinemic bracelet. We also based ourselves on the study of gesture elicitation conducted by Benjamin Alaerts in the context of his thesis [Figure 3.3]. This study allows us to justify the usefulness of swipes because they were performed by the subjects during the survey. This is also the case for the drawing of shapes, as well as the hand clapping.

Referent	Gesture	Agreement rate
Turn TV on	Press remote button with thumb	0.206
Turn TV off	Press remote button with thumb	0.142
Start player	Push button with index	0.305
Turn up the volume	Move hand up palm up	0.103
Turn down the volume	Move hand down palm down	0.112
Go to next item in a list	Swipe down	0.161
Go to previous item in a list	Swipe up	0.174
Turn Air Conditioning on	Move hand to the ceiling	0.082
Turn Air Conditioning off	Move hand to the ceiling	0.071
Turn Light on	Clap hands	0.097
Turn Light off	Close hand	0.095
Brighten lights	Move hand up palm up	0.316
Dim lights	Move hand down palm down	0.303
Accept	Thumbs up	0.17
Reject	Thumbs down	0.123
Draw a particular letter	Draw letter or shape	0.935
Draw a particular shape	Draw letter or shape	0.935
Write "Ok"	Make "OK" sign with hand	0.237
Write "KO"	Draw letter or shape	0.131

Figure 3.3: Final gestures selected from the gesture elicitation survey, from Benjamin Alearts.

The recorded data set contains 43 classes of gestures (29 unimanual and 14 bimanual) that can be used and combined to obtain all the gestures we have defined. The 29 classes of unimanual gestures are defined in [Table 3.1] and [Table 3.3] according to their collaborative or non-collaborative configuration depending on their relevance in these contexts. For example, "handshake" is a unimanual gesture relevant only in the configuration with two people each wearing a wristband and is then only defined in [Table 3.3]. The 14 classes of bimanual gestures are the classes defined in [Table 3.2] and that are not composed of unimanual gestures already defined individually in the 29 mentioned above. Each class

contains 60 templates because each participant recorded each gesture twice.

Gesture name	**Gesture description**	**Kendon type**	**McNeill type**	**Aigner type**
Swipe up	Extend your index and middle fingers and then move your wrist up.	Gesticulation	Iconic	Semaphoric
Swipe down	Extend your index and middle fingers and then move your wrist down.	Gesticulation	Iconic	Semaphoric
Swipe left	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the left.	Gesticulation	Iconic	Semaphoric
Swipe right	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the right.	Gesticulation	Iconic	Semaphoric
Check mark	Make a V in the air.	Emblems	Iconic	Iconic
Cross mark	Make a cross in the air.	Emblems	Iconic	Iconic
Circle clockwise	Make a circle clockwise.	Gesticulation	Iconic	Iconic
Circle counter-clockwise	Make a circle counterclockwise.	Gesticulation	Iconic	Iconic
Rotation right-left	Start with a vertical orientation of your hand so that you can see the thumb on top of your hand. Turn your hand 90 ° to the left so that you see the knuckles and then back to the starting position in one swift movement.	Gesticulation	Iconic	Semaphoric
Rotation left-right	Begins in a horizontal position so that your palm points to the floor. Rotate the wrist 90 ° clockwise so that your thumb points up and then back to the starting position, again all in one swift motion.	Gesticulation	Iconic	Semaphoric
Eartouch left	Begin the movement of the hand on the hip and simply move the hand to the left ear.	Gesticulation	Deictic	Pointing
Eartouch right	Begin hand movement on the hip and simply move the hand to the right ear.	Gesticulation	Deictic	Pointing
Move forward	Press a pedal with your hand forward.	Pantomine	Iconic	Semaphoric
Move backward	Press a pedal with your hand backward.	Pantomine	Iconic	Semaphoric
Hello	Wave hand to say hello.	Emblems	Iconic	Semaphoric

Gesture name	**Gesture description**	**Kendon type**	**McNeill type**	**Aigner type**
Windshield wiper	Wave your hand, with your thumb facing your face, from right to left to imitate a windshield wiper.	Pantomine	Iconic	Semaphoric
Shoulder-touch left	Begin hand movement at the hip and simply move the hand to the left shoulder.	Gesticulation	Deictic	Pointing
Shoulder-touch right	Begin the hand movement at the hip and simply move the hand to the right shoulder.	Gesticulation	Deictic	Pointing
Knock	Knock on a door.	Pantomine	Beat	Pantomimic
Mix	Mix soup in a large pot.	Pantomine	Iconic	Pantomimic
Forward rotation	Move your arm forward and rotate it as if you were screwing.	Pantomine	Iconic	Pantomimic
Backward rotation	Move your arm backward and rotate it as if you were unscrewing.	Pantomine	Iconic	Pantomimic

Table 3.1: Description of the classes of unimanual gestures and their classification according to Kendon's, McNeill's and Aigner's classification.

Gesture name	**Action by 1st Hand**	**Action by 2nd Hand**	**Kendon type**	**McNeill type**	**Aigner type**	**Laterality**
Swipe up	Extend your index and middle fingers and then move your wrist up.	Extend your index and middle fingers and then move your wrist up.	Gesticulation	Iconic	Semaphoric	Asymmetric
Swipe down	Extend your index and middle fingers and then move your wrist down.	Extend your index and middle fingers and then move your wrist down.	Gesticulation	Iconic	Semaphoric	Asymmetric
Swipe left	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the left.	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the left.	Gesticulation	Iconic	Semaphoric	Asymmetric
Swipe right	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the right.	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the right.	Gesticulation	Iconic	Semaphoric	Asymmetric
Circle clockwise	Make a circle clockwise.	Make a circle clockwise.	Gesticulation	Iconic	Iconic	Asymmetric
Circle counterclockwise	Make a circle counterclockwise.	Make a circle counterclockwise.	Gesticulation	Iconic	Iconic	Asymmetric
Rotation right-left	Start with a vertical orientation of your hand so that you can see the thumb on top of your hand. Turn your hand 90 ° to the left so that you see the knuckles and then back to the starting position in one swift movement.	Start with a vertical orientation of you hand so that you can see the thumb on top of your hand. Turn your hand 90 ° to the left so that you see the knuckles and then back to the starting position in one swift movement.	Gesticulation	Iconic	Semaphoric	Asymmetric

Gesture name	**Action by 1st Hand**	**Action by 2nd Hand**	**Kendon type**	**McNeill type**	**Aigner type**	**Laterality**
Rotation left-right	Begin in a horizontal position so that your palm points to the floor. Rotate the wrist 90 ° clockwise so that your thumb points up and then back to the starting position, again all in one swift motion.	Begin in a horizontal position so that your palm points to the floor. Rotate the wrist 90 ° clockwise so that your thumb points up and then back to the starting position, again all in one swift motion.	Gesticulation	Iconic	Semaphoric	Asymmetric
Eartouch	Begin the movement of the hand on the hip and simply move the hand to the left ear.	Begin the hand movement at the hip and simply move the hand to right ear.	Gesticulation	Deictic	Pointing	Asymmetric
Stretch	Both hands moving away horizontally.		Pantomine	Iconic	Pantomimic	Symmetric
Fusion	Both hands coming together horizontally.		Pantomine	Iconic	Pantomimic	Symmetric
Zoom in	Both hands moving away diagonally.		Pantomine	Iconic	Pantomimic	Symmetric
Zoom out	Both hands coming together diagonally.		Pantomine	Iconic	Pantomimic	Symmetric
Applause	Clap your hands.		Pantomine	Beat	Semaphoric	Symmetric
Receive	Hold out both hands.		Pantomine	Iconic	Pantomimic	Symmetric
Take	Both hands that take over.		Pantomine	Iconic	Pantomimic	Symmetric
Wrist Check	Wrists that hit each other.		Gesticulation	Iconic	Semaphoric	Symmetric
Puppet	Both wrists that turn vertically on each side of the head.		Pantomine	Iconic	Semaphoric	Symmetric
Cross Arm	Begin the hand movement at the hip and move the hand to the opposite shoulder.	Begin the hand movement at the hip and move the hand to the opposite shoulder.	Emblems	Deictic	Pointing	Symmetric
Pedal forward	Put your arms horizontally and rotate them around each other forward.		Pantomine	Iconic	Semaphoric	Symmetric
Pedal backward	Put your arms horizontally and rotate them around each other backward.		Pantomine	Iconic	Semaphoric	Symmetric

Gesture name	Action by 1st Hand	Action by 2nd Hand	Kendon type	McNeill type	Aigner type	Laterality
Right Hand Turn	Put your arms horizontally and rotate your right arm around your left arm.		Pantomine	Iconic	Semaphoric	Asymmetric
Left Hand Turn	Put your arms horizontally and rotate your left arm around your right arm.		Pantomine	Iconic	Semaphoric	Asymmetric

Table 3.2: Description of the classes of bimanual gestures, their classification according to Kendon's, McNeill's and Aigner's classification and their laterality.

Gesture name	Action by 1st Person	Action by 2nd Person	Kendon type	McNeill type	Aigner type	Laterality
Swipe up	Extend your index and middle fingers and then move your wrist up.	Extend your index and middle fingers and then move your wrist up.	Gesticulation	Iconic	Semaphoric	Asymmetric
Swipe down	Extend your index and middle fingers and then move your wrist down.	Extend your index and middle fingers and then move your wrist down.	Gesticulation	Iconic	Semaphoric	Asymmetric
Swipe left	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the left.	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the left.	Gesticulation	Iconic	Semaphoric	Asymmetric
Swipe right	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the right.	Extend your index and middle fingers and use a rotation of your wrist to move the fingers to the right.	Gesticulation	Iconic	Semaphoric	Asymmetric
Circle clockwise	Make a circle clockwise.	Make a circle clockwise.	Gesticulation	Iconic	Iconic	Asymmetric
Circle counterclockwise	Make a circle counterclockwise.	Make a circle counterclockwise.	Gesticulation	Iconic	Iconic	Asymmetric

Gesture name	**Action by 1st Person**	**Action by 2nd Person**	**Kendon type**	**McNeill type**	**Aigner type**	**Laterality**
Rotation right-left	Start with a vertical orientation of your hand so that you can see the thumb on top of your hand. Turn your hand 90 ° to the left so that you see the knuckles and then back to the starting position in one swift movement.	Start with a vertical orientation of you hand so that you can see the thumb on top of your hand. Turn your hand 90 ° to the left so that you see the knuckles and then back to the starting position in one swift movement.	Gesticulation	Iconic	Semaphoric	Asymmetric
Rotation left-right	Begin in a horizontal position so that your palm points to the floor. Rotate the wrist 90 ° clockwise so that your thumb points up and then back to the starting position, again in one swift motion.	Starts in a horizontal position so that your palm points to the floor. Rotate the wrist 90 ° clockwise so that your thumb points up and then back to the starting position, again all in one swift motion.	Gesticulation	Iconic	Semaphoric	Asymmetric
Eartouch	Begin the movement of the hand in the hip and simply move the hand to the ear.	Begin the hand movement at the hip and simply move the hand to ear.	Gesticulation	Deictic	Pointing	Asymmetric
Hand shake	Shake the hand of the other participant.	Shake the hand of the other participant.	Emblems	Iconic	Semaphoric	Symmetric
High five	Raise hand in air and slap the palm of the other participant.	Raise hand in air and slap the palm of the other participant.	Emblems	Iconic	Semaphoric	Symmetric
Hello	Wave hand to say hello.	Wave hand to respond.	Emblems	Iconic	Semaphoric	Symmetric
Stretch	Pull your hand towards you.	Pull your hand towards you.	Pantomime	Iconic	Pantomimic	Symmetric
Fusion	Push your hand forward.	Push your hand forward.	Pantomime	Iconic	Pantomimic	Symmetric
Agreement	Check mark.	Check mark.	Emblems	Iconic	Iconic	Asymmetric
Disagreement	Cross mark.	Cross mark.	Emblems	Iconic	Iconic	Asymmetric
Mitigate	Check mark/Cross mark.	Cross mark/Check mark.	Emblems	Iconic	Iconic	Asymmetric

Gesture name	**Action by 1st Person**	**Action by 2nd Person**	**Kendon type**	**McNeill type**	**Aigner type**	**Laterality**
Transfer	Transfers an object to the 2nd participant.	Receives object.	Pantomime	Iconic	Pantomimic	Asymmetric
Throw	Throws a small object to the 2nd participant.	Catches object.	Pantomime	Iconic	Pantomimic	Asymmetric
Carry	Carry object together with the other participant.		Pantomime	Iconic	Pantomimic	Symmetric
Push	Push object together with the other participant.		Pantomime	Iconic	Pantomimic	Symmetric
Pull	Pull object together with the other participant.		Pantomime	Iconic	Pantomimic	Symmetric

Table 3.3: Description of the classes of unimanual collaborative gestures, their classification according to Kendon's, McNeill's and Aigner's classification and their laterality.

We can also classify the gestures in [Table 3.2] according to the vocabulary defined by Jahani [22]. We chose the combined gesture codes for the "cross arm" and "eartouch" gestures because these gestures must mark a pause at the end of the movement of the hand position to mark the gesture.

Gesture name	**Jahani's gesture code**
Swipe up	Left and Right hand : 4
Swipe down	Left and Right hand : 4
Swipe left	Left and Right hand : 4
Swipe right	Left and Right hand : 4
Circle clockwise	Left and Right hand : 4
Circle counter-clockwise	Left and Right hand : 4
Rotation right-left	Left and Right hand : 2
Rotation left-right	Left and Right hand : 2
Eartouch	Left and Right hand : 3-1
Stretch	Left and Right hand : 3
Fusion	Left and Right hand : 3
Zoom in	Left and Right hand : 4
Zoom out	Left and Right hand : 4
Applause	Left and Right hand : 4
Receive	Left and Right hand : 4
Take	Left and Right hand : 4
Wrist check	Left and Right hand : 4
Puppet	Left and Right hand : 4
Cross Arm	Left and Right hand : 4-1
Pedal forward	Left and Right hand : 4
Pedal backward	Left and Right hand : 4
Right hand turn	Left : 1 Right : 4
Left hand turn	Left : 4 Right : 1

Table 3.4: Description of the classes of bimanual gestures and their classification according to Jahani's gesture code

3.6 Configurations

Given that we have two bracelets and that it would be possible to imagine having even more as each person can use between 0 and two bracelets. We decided to represent in the form of a graph and various tables, the configurations that would be possible if we used 1, 2, 3 or 4 bracelets with 1 or 2 people. In the [Figure 3.4], it can be seen that there are five different main cases and each of them can be divided into several configurations depending on who owns the bracelet(s) and whether it is placed on the dominant hand or not. On the y-axis, the number of Kinemic bracelets per person, there are 3 possibilities, either the bracelet(s) are used unimanually, or one person has 1 bracelet and the other 2, or they are used bimanually. The x-axis represents the number of people in the configuration.

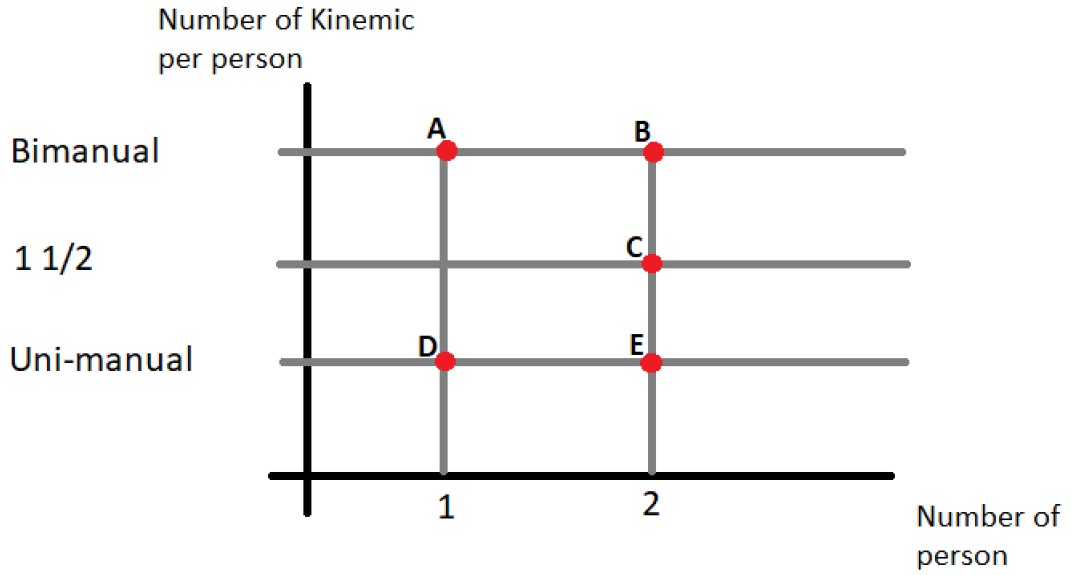


Figure 3.4: Graph of possible configuration.

In configuration A [Table 3.5], two wristbands are used by one person, i.e. bimanually. This is the configuration in which all gestures in [Table 3.2] will be found.

A	**Total : 2 Kinemic**
Person 1	2 Kinemic

Table 3.5: Configuration A

In configuration B [Table 3.6], two people use four bracelets, that is, bimanually for each. As we only have two wristbands, we cannot practically test this configuration and it is not represented in a table, but it would represent two people with gestures in [Table 3.2] and different combinations could be imagined between them.

B	**Total : 4 Kinemic**
Person 1	2 Kinemic
Person 2	2 Kinemic

Table 3.6: Configuration B

In configuration C [Table 3.7], three wristbands are used by two people, i.e. one unimanually and the other bimanually. As with configuration B, these are configurations that we cannot practically implement. Here, one person has two wristbands, and the other has only one, which she can put on her dominant hand or not. This makes a total of four possible configurations.

C	Total : 3 Kinemic	Hand
Person 1	2 Kinemic	/
Person 2	1 Kinemic	Dominant
Person 2	1 Kinemic	Non-dominant
Person 1	1 Kinemic	Dominant
Person 1	1 Kinemic	Non-dominant
Person 2	2 Kinemic	/

Table 3.7: Configuration C

In configuration D [Table 3.8], a wristband is used by a single person, i.e. in a one-handed manner. He or she can use it on his dominant or non-dominant hand. In this configuration, we will find all gestures in the [Table 3.1]. In our case, when we recorded the gestures, the participants all decided to use the bracelet on their right hand, whether it was their dominant hand or not.

D	Total : 1 Kinemic	Hand
Person 1	1 Kinemic	Dominant
Person 1	1 Kinemic	Non-dominant

Table 3.8: Configuration D

In configuration E [Table 3.9], two bracelets are used by two people, that is, unimanually for each. They can use it on their dominant or non-dominant hand. This will create four different configurations. In these configurations, we will find all the gestures in [Table 3.3].

E	Total : 2 Kinemic	Hand
Person 1	1 Kinemic	Dominant
Person 1	1 Kinemic	Non-dominant
Person 2	1 Kinemic	Dominant
Person 2	1 Kinemic	Non-dominant

Table 3.9: Configuration E

3.7 Summary

In this section, we have been able to justify the choice of the Kinemic as the device used for our experiment, as well as its data structure. We have also described the whole process we followed to create our dataset, as well as the dataset itself. The set of gestures that we recorded was described in detail and their choice was justified by an elicitation study, as well as by the use of different categories that we had defined in Chapter 2. Finally, we were able to describe the different configurations in which we had recorded the gestures. All the recorded unimanual gestures will then be tested and all the gestures could potentially be used in an application.

Chapter 4

Benchmarking of Kinemic Gesture Recognizers

This chapter will contain all the tests we have carried out. To ensure the rigorousness of the tests, the criteria and parameters of the tests will be described in detail. The tests are performed in a user-independent and user-dependent manner. For these two parts, the precise steps of the process and the adaptations made will be explained in their respective sections.

4.1 Benchmark

4.1.1 Criteria

First of all, let's define the objectives of our different tests: the goal is to compare different recognizers to identify which one would work best to recognize the Kinemic's gestures. To carry out these tests, we decided to use pattern matching algorithms rather than neural networks because they are much more adjustable when it comes to adding templates or even to add a gesture, it is not necessary to run everything from the beginning. Here is the list of the different recognizers we have decided to use:

- Jackknife : it is a very flexible recognizer and independent of the number of dimensions [19].
- $P3+ : according to the study in the article *Recognizing 3D Trajectories as 2D Multi-stroke Gestures*. Being a recognizer based on 3 data and the Kinemic providing us 6, we will test the recognizer using either the 3 acceleration data or the 3 gyroscopic data. It is one of the best recognizers compared to many others [40].
- $PN+: this is a generalization of $P3+ which we introduced because the Kinemic retrieves 6 data and $P3+ only retrieves 3 data. It therefore allows us to use all the data from the bracelet to recognize the gesture.
- μF and μV: these two recognizers have not yet been really tested, so we thought it would be interesting to try them as well [29].

4.1.2 Parameters

To test the different recognizers, we varied the different parameters available in QuantumLeap. We worked by freezing all but one of the parameters until we found the best result, and so on for all the parameters of each recognizer.

- Number of repetitions: The number of repetitions for each combination of parameters is in the range of 10 to 200 repetitions. A minimum of 10 was taken to test with few repetitions when this improved the score and a maximum of 200 when a plateau was not reached before in the results.
- Minimum number of templates: The minimum number of training templates used for the training of the recognizers which has always been left at 1.
- Maximum template: The maximum number of training templates used for the training of the recognizers, its value depending on the dataset used and the type of test. Overall, it will vary from 1 to 58. As the minimum number of templates registered per gesture class is 2, for user-dependent tests the maximum possible number to be taken for one person is 1 template. Then the maximum number of templates in the dataset is 60, so for user-independent tests, since there are two templates per user, this gives a maximum of 58 templates.
- Number of sampling points: The number of points in the gesture after resampling, ranging from 8 to 256, increasing or decreasing by multiples of 8 each time. A minimum of 8 was taken to test with few sampling points when this improved the score or simply to start with a small number and a maximum of 256 when a plateau was not reached before in the results.
- Number of participants: Number of different people making the gestures which can vary from 2 to 30 people, as the maximum number of people we have recorded in the database is 30.

The μV recognizer does not use sampling points but shapes :

- Number of shapes: The number of shapes into which gestures are converted, ranging from 8 to 256, increasing or decreasing by multiples of 8 each time. A minimum of 8 was taken to test with few shapes when this improved the score or simply to start with a small number and a maximum of 256 when a plateau was not reached before in the results.

The μF recognizer has additional special parameters:

- Weights: These are the weights of the 4 features that allow to calculate the similarity between a candidate gesture and the template set. We used the default values which are 1 for Alpha, 4 for Beta, 0 for Range and 0 for Delta.
- Articulations: The gesture articulation.

4.1.3 User-Independent Procedure

First we tested the different recognizers to evaluate their performance and to help us make our choice. We carried out this first part of the test in a user-independent (UI)

manner, i.e. the template of one user is compared to the templates of other users. The maximum number of templates will always be at most the number of templates made by the other participants [40]. For example, if there are 30 participants who have each performed each gesture twice, the maximum number of template will be set to 58, which is the total number of templates for a gesture minus those performed by one user. This user-independent test procedure is, in fact, a special case of the general procedure called "cross-validation". It is in fact a 1-fold cross-validation in which, at each iteration, a gesture (or a class of gestures) is extracted from the complete data set to serve as a test template and the system is trained with the rest. The choice of the number 1 is due to the fact that we have few data so we cannot choose a k too big otherwise the k-fold cross validation procedure will not be relevant.

Figure 4.1: K-fold cross-validation method with k=1.

Step 1 : Basic Gestures We first tested the 12 basic gestures that the Kinemic offers because they are simple, basic gestures that have already proven themselves with the bracelet. The dataset then has 12 classes of gestures that each contain 60 gesture templates since we recorded the gestures with 30 participants who repeated each gesture 2 times. The complete tables with all detailed test results can be found in Appendix D.1.1. For each recognizer, there are three different tables with the results for 10, 20 and 30 participants, respectively. The maximum number of templates is determined according to the number of people and therefore corresponds to 18 for 10 people, 38 for 20 and 58 for 30, i.e., the complete dataset. In each table, the number of repetitions and the number of sampling points are varied.

- Jackknife : First of all, with this recognizer, we can see that the best results are obtained when we take about 16 sampling points, if we take more or less, the accuracy decreases slightly. If we decrease the number of repetitions compared to 100 we get on average a better result except for 20 participants where 150 repetitions give the

best score. Finally, the best score of 70,2% is given when we make 50 repetitions and we take only 10 participants and decreases slightly if we take more.

- $PN+ : Just like Jackknife this recognizer gets its best result for 16 sampling points and decreases if you increase these except for the whole database where it remains broadly the same. The number of repetitions does not really influence the accuracy returned for 10 and 30 participants. When we take 20 people decreasing the number of repetitions increases the score. The best result of 63,3% is given when making 10 repetitions and taking 20 participants, around that it decreases.

- μV : Unlike other recognizers, better results are obtained by increasing the number of sampling points, and increasing it to just 128 or 256 results in stagnant scores. On the other hand, the number of repetitions does not really influence the accuracy. The best results are obtained with 10 and 20 participants, but the accuracy decreases when using 30 participants. With 10 participants and 128 sampling points, we obtain the score of 49% with 50 repetitions and 48% with 100 and with 20 participants the accuracy is of 49,6% with 100 repetitions and 256 sampling points.

- $P3+ with acceleration data : This recognizer reaches best performance with 64 and 128 sampling points, if you decrease or increase them the performance decreases, so we can consider that there is a plateau. The number of repetitions influences in the same way, the best is 100 repetitions, if you decrease or increase it, the performance decreases. The best result of 57,8% is obtained with 30 people, 100 repetitions and 128 sampling points.

- $P3+ with gyroscopic data : The observations are the same as for the alternative with the acceleration concerning the number of sampling points and the repetitions. In general, the recognizer with gyro data is slightly worse than the one with acceleration, except that it improves with decreasing number of persons and exceeds for 10 persons the performance of acceleration for the same number. Furthermore, we notice no significant difference between using only acceleration data or using gyroscopic data. The rates are very similar, so both data are relevant.

- μF : It is more efficient with few sampling points (8 or 16) on few people. But when we use the complete dataset of 30 people, the best results are reached with 64 sampling points. The best performance with 30 people is of 59,6% with 100 repetitions and 64 sampling points, with 10 people the accuracy is 68,9% with 100 repetitions and 8 sampling points.

The best result is achieved with the Jackknife recognizer which obtains an accuracy of 70,2% with the following parameters: 50 repetitions, 16 sampling points and 10 persons. It is also the best on the complete dataset (30 persons) with an accuracy of 67,7% on average with 10 repetitions and 32 sampling points. The worst recognizer is μV with a maximum accuracy of 49,6% using the following parameters: 100 repetitions, 256 sampling points and 20 participants.

Step 2 : Unimanual Gestures Once we had tested the different recognizers on the existing gestures of the Kinemic, we decided to test them on all the gestures we wanted to integrate into the bracelet in addition to the basic ones, i.e. the 12 basic ones and the 17 we had invented. The dataset has now 29 gesture classes that each contain 60 gesture templates.

Unfortunately, we soon realized after a few tests that the results dropped drastically and that the results became really poor. For some recognizers like Jackknife, the results could go from 60% accuracy to less than 50%, so the recognizer did not recognize even one out of two times correctly.

Rather than continue testing with unconvincing results, we decided to review the different gestures that we wanted to add to the Kinemic.

Step 3 : Gestures Selection To improve our results, we have analyzed the different gestures that we had imagined to keep the most relevant for use with the Kinemic bracelet. A gesture is relevant when it is not confused with another gesture and when it is recognizable. It was assumed that it was recognizable if a rate of at least 70% between the two gestures could be achieved.

To do so, we separated the gestures into two groups to group the gestures most likely to be similar and indistinguishable. For example, Move forward and High Five are both in the first group because they are visually very similar. Then, we compared their recognition rate and the confusion matrix. We also took into account the fact that symmetrical gestures (example: ”shoulder right” and ”shoulder left”) have a lower accuracy, but nonetheless we keep the two symmetries of the gestures to remain consistent. The two tables containing all the comparisons can be found in the Appendix C.1 and C.2

We decided to remove the gestures "High Five", "Throw", "Catch", "Knock", "Shouldertouch right", "Shouldertouch left" and "Windshield wiper" because they were strongly confused with several gestures. Indeed, the accuracy did not exceed 60% between one of the gestures and another one of the dataset.

For example, for the first group between Move Forward and High Five the recognition rate is 53% and similarly between High Five and Throw the rates vary between 50% and 60%. Moreover, when we take one of the confusion matrices for 100 repetitions with the best recognizer obtained so far, i.e. Jackknife, the diagonal of the matrix represents out of 100 the number of times the gesture has been recognized, so we can analyze it as a percentage. We can see very well for the first group in [Figure 4.1] that High Five is only recognized at 21% and that it is very often confused with several other gestures such as Catch, Move Forward, Move Backward and that it is even more recognized as Throw. The same kind of example can be found for the second group, where rates of 60% are obtained between Knock and Windshield Wiper. We can also analyze the confusion matrix [Figure 4.2] carried out in the same context as that of the first group, where we see that Windshield Wiper is only recognized at 33% for example and that it is often confused with Knock and Mix.

We also removed the "Carry" gesture because although it was differentiable from the gestures in its group, when we put all the gestures back together, we realized that it was difficult to differentiate with some of the gestures in group 1. Moreover, the dataset that does not contain them gives better results.

Gesture name	Catch	Hand Shake	High Five	Move Backward	Move Forward	Receive	Throw	Transfer
Catch	50	2	15	15	12	2	4	0
Hand Shake	0	75	5	6	1	4	2	7
High Five	13	4	21	13	19	0	22	8
Move Backward	21	1	18	45	3	7	5	0
Move Forward	6	2	33	4	33	0	15	7
Receive	0	6	0	7	3	56	3	25
Throw	4	2	14	4	11	2	57	6
Transfer	0	7	3	2	0	13	3	72

Table 4.1: Confusion matrix for the first group with 100 repetitions.

Gesture name	Backward Rotation	Carry	Forward Rotation	Hello	Knock	Mix	Shoulder Left	Shoulder Right	Windshield Wiper
Backward Rotation	72	0	26	2	0	0	0	0	0
Carry	0	61	3	14	1	1	10	7	3
Forward Rotation	29	0	66	1	0	2	0	0	2
Hello	3	0	4	90	0	0	0	1	2
Knock	4	1	1	11	46	19	0	0	18
Mix	3	1	6	5	16	52	1	0	16
Shoulder Left	4	9	4	2	2	6	38	28	7
Shoulder Right	0	2	0	1	2	0	14	77	4
Windshield Wiper	1	0	0	15	25	20	5	1	33

Table 4.2: Confusion matrix for the second group with 100 repetitions.

Step 4 : Selected Gestures After selecting the gestures that we thought were the most complete and that the recognizers were able to differentiate the best, we re-run tests on the selected gestures only. The dataset then has 21 classes of gestures that each contain the number of people chosen in parameters times 2 templates, because each person has recorded a gesture twice. The complete results of these tests are also presented in the Appendix D.1.2. As for the tests on basic gestures, there are also three tables per recognizer showing the results for 10, 20 and 30 people.

During this test phase we could observe that the different recognizers evolved with the change of parameter in the same way as in the first step on the 12 basic gestures. In this phase, $P3+ gave us very similar if not slightly better results for all of our new gestures and basic gestures compared to testing only the 12 basic gestures. The others, i.e. Jackknife, $PN+, μF and μV all gave better results on the 12 basic gestures alone. Indeed, the best result for Jackknife goes from 70,2% to 62.3%. Therefore, we lose almost 10% of accuracy compared to basic gestures. For $PN+, we go from a recognition rate of 63,3% to 60,5%, which is less decreasing than for Jackknife, but remains a lower score than for basic gestures. μV keeps an extremely bad accuracy given that it always stays below 50% going from 49.6% for basic gestures to 44,5% for these.

On this set of gestures, the best results we obtained were given by the Jackknife recognizer as well as for the 12 basic gestures. But here it gave us an accuracy of 63% with 100 repetitions, 32 sampling points and a total of 20 participants used. The worst results were returned by μV also with a better score of 44,5% when the recognizer used the same parameters as in step 1, i.e. 100 repetitions, 256 sampling points with 20 people.

Step 5 : Analyze of the Result After doing all these tests and despite the fact that we tried to find the best settings and test several recognizers as to well as remove gestures that the device could not differentiate, the final result surprised us. Indeed, we expected results with accuracy in the 80-90% range. Therefore, we investigated the origin of these low results by looking at several possibilities, such as the inefficiency of the recognizers, the poor registration of the device or too variable data. We soon realized that the last possibility was probably the most likely when we looked at some of the gestures made by the users. For example, the results for Swipe left were observed by comparing what the graph of the 3 acceleration data should look like with one of the graphs of data recorded by a participant. In the [Figure 4.2] we can see the normal evolution of the acceleration when a Swipe Left is performed while in the [Figure 4.3] this is what was recorded. It can be clearly observed that there was a problem during the recording of certain gestures, perhaps due to inaccuracy or the sensitivity of the bracelet.

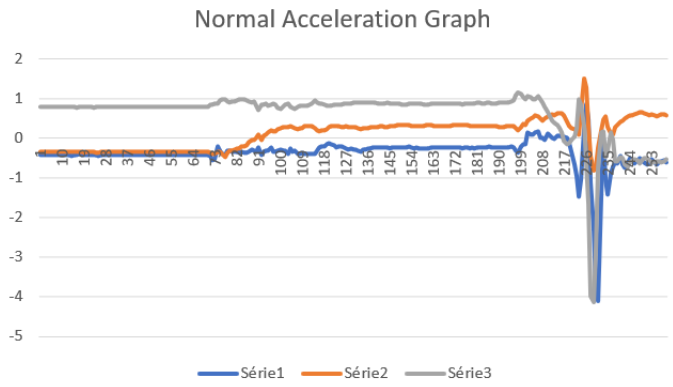


Figure 4.2: Normal acceleration evolution for swipe left.

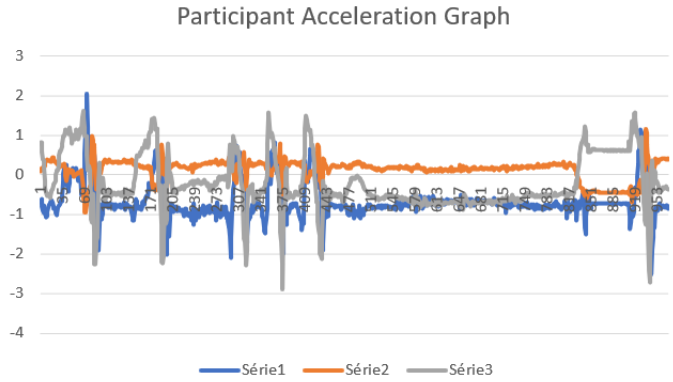


Figure 4.3: Participant acceleration evolution for swipe left.

Step 6 : Testing New Dataset In order to try and improve the results by getting accurate gestures, we decided to test running the recognizer only on our recorded gestures and quickly realized that these were better. All results are given in the tables in Appendix D.1.3. Therefore we can assume that the quality of the dataset and the quality of the recorded gestures greatly influence our results so we decided to recreate a dataset with only us as two people and we each repeated each gesture 10 times to obtain 20 templates per gesture class. This dataset contains the same gestures as in step 4. Since the number of participants is 2, the maximum number of templates will be 10. We have thus obtained this kind of results for the different recognizers:

- Jackknife : With this recognizer the best results are obtained with 16 sampling points, increasing these decreases the accuracy. The number of repetitions has a slight influence and the best result of 75,5% is obtained with 150 repetitions but with 100 repetitions the score is still 74%.
- $PN+ : As for Jackknife, the optimal number of sampling points is 16, but the number of repetitions to obtain the best result for this recognizer is 100 for an accuracy of 55%. Doing more or fewer repetitions decreases the score.
- μV : With μV we still get poor results with a score of 39,4% for 16 sampling points and 100 repetitions. Since the results seem identical and still bad, more tests have not been done for this recognizer.

- $P3+ with acceleration data : The optimal number of sampling points for this recognizer with the 3 acceleration data is 16, if you decrease or increase it, the performance decreases. The best results are obtained with 100 and 150 repetitions and is 57%.
- $P3+ with gyroscopic data : Using gyro data instead of acceleration data, the best performance is measured with 8 sampling points and 100 repetitions. The best score is then 60% on average
- μF : With this recognizer, the optimal number of sampling points is 16, and if you decrease or increase it, the performance decreases. The optimal number of repetitions is 150. The best score achieved is 60%.

With this new dataset, the best result is still given by the Jackknife recognizer with 74.5% accuracy which is about 10% more than with the dataset of the previous steps. $PN+ gives a worse result as it goes from 60% to 55%. The μV recognizer remains as poor as ever with rates of 40%. In general, the results for the $P3+ recognizer are better than with the old dataset, on average about ten percent better considering 30 peoples. But the results remain low because 60% is the best recognition rate achieved. For μF the recognition rate does not improve, it is still around 60% at best, but the parameter to reach it is different with our new dataset, it is 16 sampling instead of 64 before.

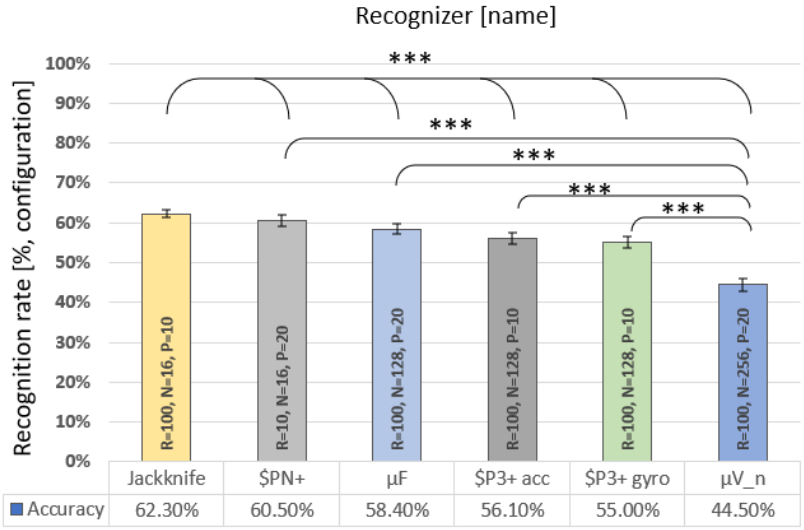


Figure 4.4: Configurations of recognizers leading to the best accuracy in the user-independent scenario with the old dataset. Error bars show a confidence interval of 95% computed on respective configurations.

Fig. 4.4 shows the configurations of the recognizers which lead to their best accuracy in the user-independent settings. For most cases, a repetition factor R=100 and a number of sampling points N=128 lead to the best recognition rates of the compared recognizers, with some exceptions, such as the best rate for the best recognizer, which is obtained with N=16. Since individual recognition rates are not normally distributed, we performed

a Kruskall-Wallis statistical test to determine whether there is a significant difference between recognition rates (H=91.49, df=5, p^{**}=$3.22E-18$, α=0.05). A Tukey post hoc test reveals that Jackknife is significantly more accurate than all other recognizers ($p^{***} \leq 0.001$) with a large effect size: with respect to \$$PN+$ (Cohen's d=1.79), to μF (Cohen's d=1.2), to \$$P^3+$ acc (Cohen's d=1.91), to \$P3+ gyro (Cohen's d=1.8), and to μV_n (Cohen's d=5.97). μV_n is significantly less accurate than all remaining recognizers with a large effect size: \$PN+ (p^{**}=0.028, Cohen's d=4.17), μF (p^{**}=0.029, Cohen's d=4.76), \$P3+ acc (p^{**}=0.03, Cohen's d=4.06), and \$P3+ gyro (p^{*}=0.03, Cohen's d=4.17).

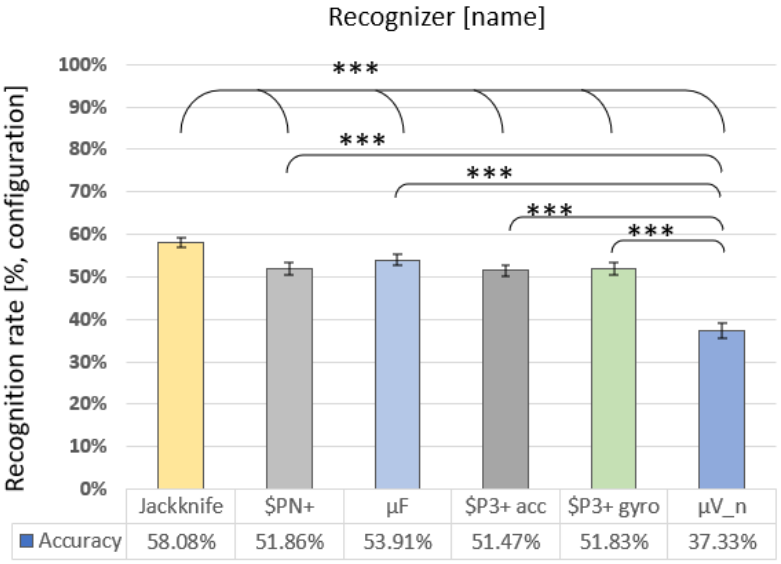


Figure 4.5: Configurations of recognizers leading to the mean accuracy in the user-independent scenario with the old dataset. Error bars show a confidence interval of 95% computed on respective configurations.

Fig. 4.5 shows the average accuracy achieved between all our tested configurations in the user-independent settings. The Kruskall-Wallis statistical test and the Tukey post hoc test bring the same observations as before for Fig. 4.6. Jackknife is the most accurate than all other recognizers and μV is the least accurate one compared to all others.

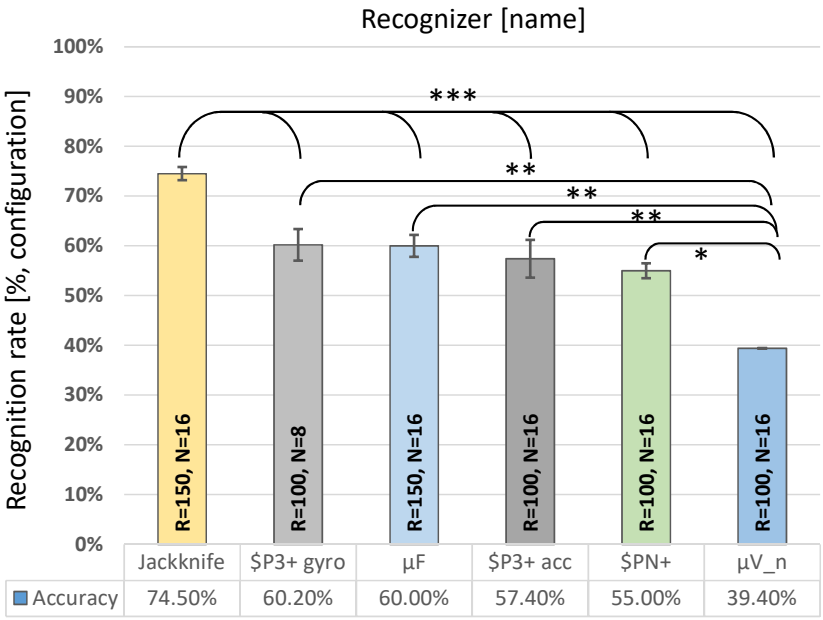


Figure 4.6: Configurations of recognizers leading to the best accuracy in the user-independent scenario with the new dataset. Error bars show a confidence interval of 95% computed on respective configurations.

Fig. 4.6 shows the configurations of the recognizers that lead to their best accuracy in user-independent settings. For most cases, a repetition factor R=100 and a number of sampling points N=16 lead to the best recognition rates of the compared recognizers, with some exceptions. Since the individual recognition rates are not normally distributed, we performed a Kruskall-Wallis statistical test to determine whether there is a significant difference between recognition rates (H=15.59, df=5, p^{**}=0.00806, α=0.05). A Tukey post hoc test reveals that Jackknife is significantly more accurate than all other recognizers ($p^{***} \leq 0.001$) with a large effect size: with respect to \$P3+ gyro (Cohen's d=5.07), to μF (Cohen's d=4.55), to \$$P^3$+ acc (Cohen's d=5.44), to \$$PN$+ (Cohen's d=6.02), and to μV_n (Cohen's d=9.83). μV_n is significantly less accurate than all remaining recognizers with a large effect size: \$P3+ gyro (p^{**}=0.0031, Cohen's d=4.76), μF (p^{**}=0.00108, Cohen's d=5.28), \$P3+ acc (p^{**}=0.0077, Cohen's d=4.39), and \$PN+ (p^{*}=0.028, Cohen's d=3.81).

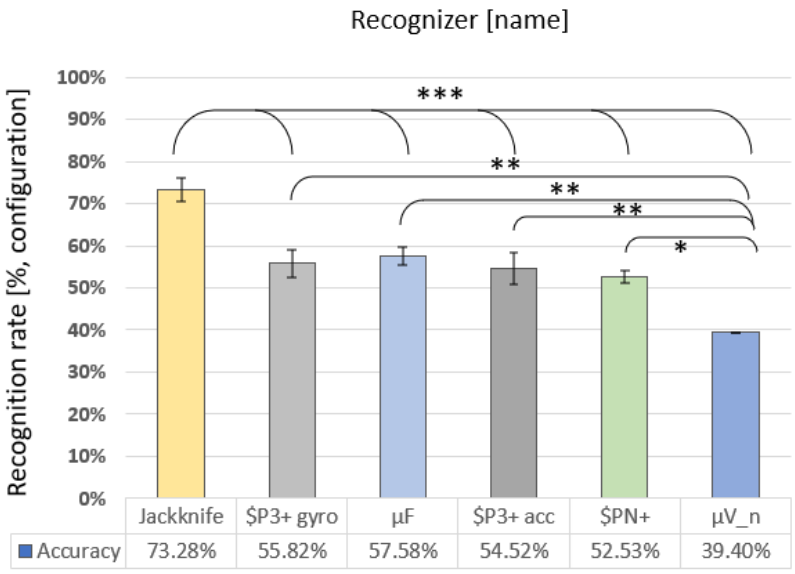


Figure 4.7: Configurations of recognizers leading to the mean accuracy in the user-independent scenario with the new dataset. Error bars show a confidence interval of 95% computed on respective configurations.

Fig. 4.7 shows the average accuracy achieved between all our tested configurations in the user-independent scenario. The Kruskall-Wallis statistical test and the Tukey post hoc test bring the same observations as before for Fig. 4.6. Jackknife is the most accurate than all other recognizers and μV is the least accurate one compared to all others.

4.1.4 User-Dependent Procedure

Next, we carried out user-dependent (UD) tests, i.e. the user's template is compared with the other templates he or she has made for the gesture [40]. If each user has made 2 templates per gesture, the maximum number of templates will be 1. We only performed these tests on the dataset from step 4 of the UI tests and on the dataset from step 6 of the UI tests.

Old Registered Dataset with 21 Selected Gestures First of all, we did the tests in a user-dependent way on the basic database with the 21 gestures which had been selected in the end as being quite differentiable. The tables showing the full details of the various tests are in Appendix D.2.1.

- Jackknife : For this first recognizer, the ideal number of sampling points is 16, increasing this does not improve the score. Depending on the number of participants selected, the number of repetitions can slightly improve the score. The best accuracy of 79 % is obtained with 50 repetitions and the first 10 participants selected only.
- $PN+ : In contrast to Jackknife, increasing the number of sampling points will increase the score, the optimal number is 64 after which the score will stagnate. The number of repetitions can vary the score, but only slightly. The best accuracy of 87,7% is achieved with 100 repetitions and the first 10 participants selected only.

- μV : As with the previous recognizer, increasing the number of sampling points will increase the score, but the score quickly stagnates at 32 sampling points. The number of repetitions can vary the score depending on the number of people selected. The best accuracy of 55.4% is obtained with 50 repetitions and the first 20 participants selected only.

- $P3+ with acceleration data : With this recognizer a high peak performance is reached with 128 sampling points except with the 20 person dataset where the best choice is 64 sampling points. It consistently achieves the best score recorded with 150 repetitions, and this best score is 89% with 10 people, but with the full dataset (30 peoples) the result is also very good, it reaches 87%.

- $P3+ with gyroscopic data : Using gyro data instead of acceleration data does not change the behavior of the results with respect to the parameters. A better result is also obtained when the number of sampling points is 128 and the optimal number of repetitions is 150 for 30 and 10 persons, but the performance is slightly different and not as good as with the acceleration data, the best result obtained being 88.8% on the complete dataset with 30 persons. With only 10 people the accuracy decreases by 2% compared to the results with the acceleration data.

- μF : With this recognizer, the best results are obtained with 64 sample points with 30 people, but with 10 or 20 people the best value is 32 sample points. The number of repetitions does not always influence the results in the same way so it is difficult to observe the best value for this parameter. The best accuracy reached is 64% in the 10 person dataset with 32 sampling points and 100 repetitions. On the complete dataset of 30 persons the highest result is 58%.

In comparison with the user-independent tests we carried out in the previous section, we can see that the results are much better. This seems quite logical, as the user-dependent tests compare a user's gesture with another one that they have performed themselves, so there is a greater chance that the two gestures are similar as they are performed by the same person. The same kind of observation can be found in the article [40], where he also explains that recognizing a 3D gesture that is not the same as the models used during training is more complex than identifying an already existing trajectory.

Thanks to the results, it can be seen that Jackknife, which used to give the best results, has increased from 63% to 79%. For $PN+ the difference is even higher, going from 60% to 87% which is a very good recognition score. The μV recognizer remains at a very low level even though the user-dependent tests are 10% higher than the user-independent tests. $P3+ also improves greatly, accuracy increases by 30% on average for the same parameter values. The μF recognizer also improves, by about 6% on the full dataset of 30 people and by about 10% when only 10 people are considered.

We can observe that in contrast to the user-independent tests the best recognizer is no longer Jackknife but $P3+ with a record accuracy of 89%, followed closely by $PN+ which reaches 87%.

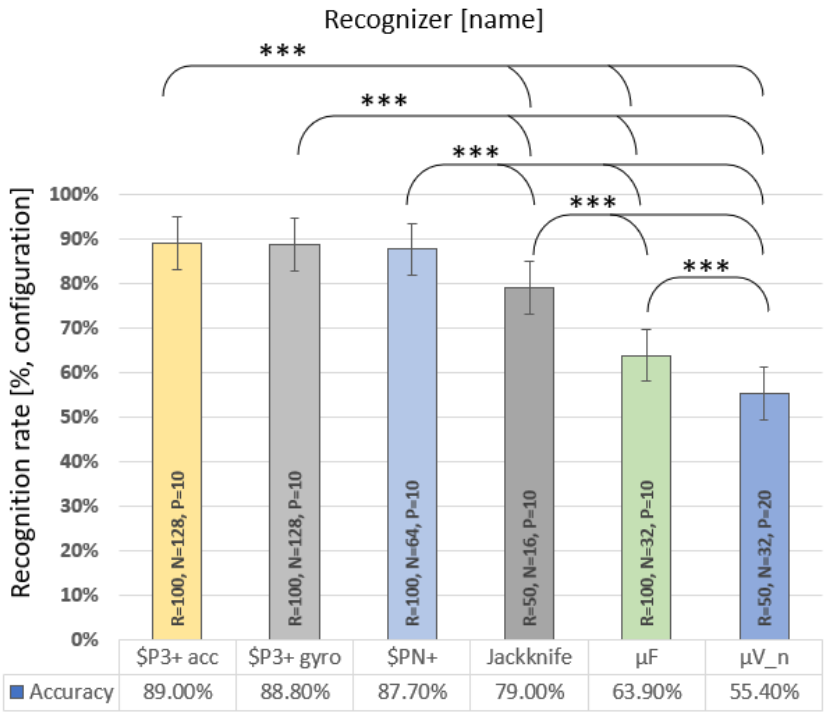


Figure 4.8: Configurations of recognizers leading to the best accuracy in the user-dependent scenario with the old dataset. Error bars show a confidence interval of 95% computed on respective configurations.

Fig. 4.8 shows the configurations of the recognizers which lead to their best accuracy in the user-dependent settings. For most cases, a repetition factor R=100 and a number of sampling points N that varies between 16 and 128 lead to the best recognition rates of the compared recognizers. Since the individual recognition rates are not normally distributed, we ran a Kruskall-Wallis statistical test to determine whether there is a significant difference between the recognition rates (H=76.66, df=5, p^{**}=4.15771$E-15$, α=0.05). A Tukey post-hoc test reveals that \$P3+ acc is significantly more accurate than Jacknife, μF and μV ($p^{***}\leq$0.001) with a large effect size: with respect to Jacknife (Cohen's d=3.23), to μF (Cohen's d=7.08) and to μV (Cohen's d=9.98). \$P3+ gyro is also significantly more accurate than Jacknife, μF and μV ($p^{***}\leq$0.001) with a large effect size: with respect to Jacknife (Cohen's d=2.98), to μF (Cohen's d=9.74) and to μV (Cohen's d=9.98). Then the test reveal than \$PN+ is significantly more accurate than Jacknife, μF and μV ($p^{***}\leq$0.001) with a large effect size: with respect to Jacknife (Cohen's d=2.87), to μF (Cohen's d=6.72) and to μV (Cohen's d=9.62). Jackknife, on the other hand, is significantly more accurate than μF and μV ($p^{***}\leq$0.001) with a large effect size: with respect to μF (Cohen's d=3.85) and to μV (Cohen's d=6.75). And finally, μF is significantly more accurate than μV ($p^{***}\leq$0.001) (Cohen's d=2.9).

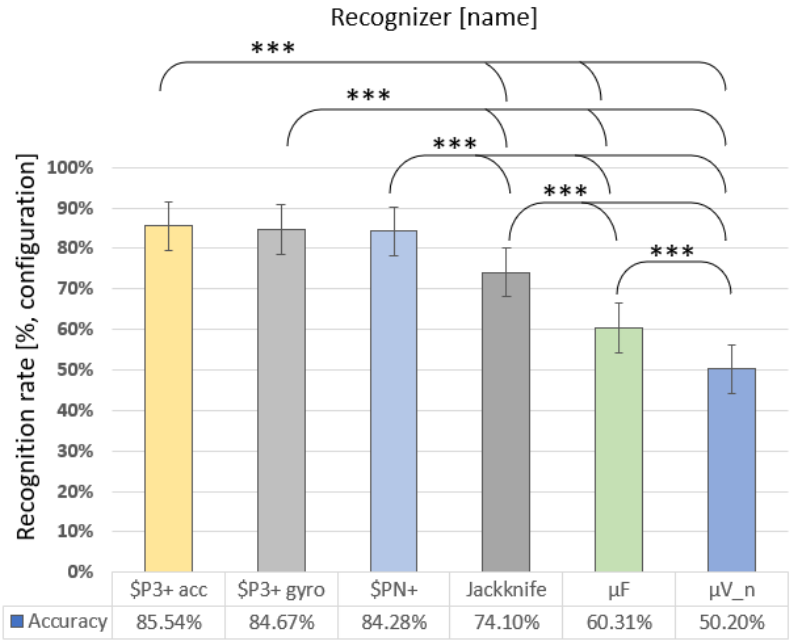


Figure 4.9: Configurations of recognizers leading to the mean accuracy in the user-dependent scenario with the old dataset. Error bars show a confidence interval of 95% computed on respective configurations.

Fig. 4.9 shows the average accuracy achieved between all our tested configurations in the user-dependent scenario. The Kruskall-Wallis statistical test and the Tukey post hoc test bring the same observations as before for Fig. 4.8.

New Dataset with 2 People Only and 21 Selected Gestures We also carried out user-dependent tests on the new dataset that we had recorded with only 2 people. The complete test results can be found in Appendix D.2.2.

- Jackknife : For this first recognizer, the ideal number of sample points is 16, increasing this number does not improve the score. Decreasing the number of repetitions gives a better result. The best accuracy of 86.3% is obtained with 50 repetitions and 16 sampling points.
- $PN+ : In contrast to the Jackknife, increasing the number of sample points increases the score, the optimal number being 64, after which the score stagnates. Changing the number of repetitions does not improve the score. The best accuracy of 85% is obtained with 100 repetitions and 64 sampling points
- μV : As in the previous case, increasing the number of sample points increases the score, but it soon stagnates at 32 sample points. Decreasing the number of repetitions increases the score and thus achieves a better accuracy of 60,3% with 50 repetitions and 32 sampling points.
- $P3+ with acceleration data : With this recognizer, the performance reaches a peak with 32 and 64 sampling points, by decreasing or increasing this number, the performance decreases. The optimal number of repetitions is 100. The best score achieved is 97% with 32 or 64 sampling points and 100 repetitions.

- $P3+ with gyroscopic data : With gyro data compared to observations with acceleration data, the plateau is longer and is reached with 32, 64 and 128 sampling points. Indeed the performance with these values reaches 98% and the number of repetitions does not influence much, the best value is usually 100 repetitions.
- μF : With this recognizer, the optimal number of sampling points is 64, and if you decrease or increase this number, the performance decreases. The optimal number of repetitions is 100. The highest accuracy achieved is 98% with 64 sampling points and 100 repetitions.

As with the other dataset, when analyzing the results, one can also observe an increase in scores compared to the user-independent test, with several recognizers reaching scores of more than 80%. The best recognizer is also $P3+ as for the other user-dependent tests on the earliest recorded datasets. Indeed, the highest accuracy rate has been achieved with this recognizer and is 98,75%. It is also achieved by μF results which also give accuracy of 98,75%.

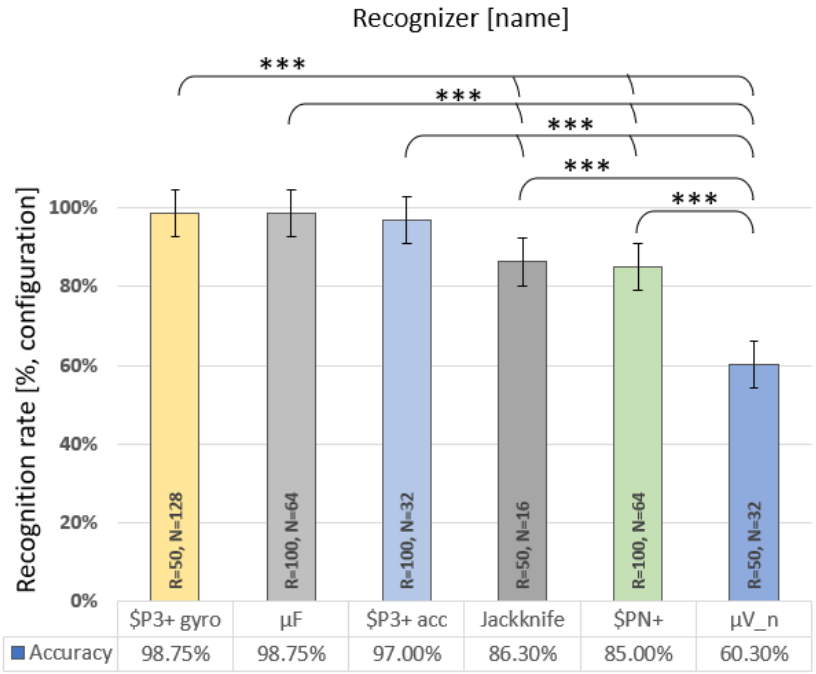


Figure 4.10: Configurations of recognizers leading to the best accuracy in the user-dependent scenario with the new dataset. Error bars show a confidence interval of 95% computed on respective configurations.

Fig. 4.6 shows the configurations of the recognizers which lead to their best accuracy in the user-dependent settings. For most cases, a repetition factor R=50 or R=100 and a number of sampling points varying from N=16 to N=128 lead to the best recognition rates of the compared recognizers, apart a few exceptions. Since the individual recognition rates are not normally distributed, we ran a Kruskall-Wallis statistical test to determine whether there is a significant difference between the recognition rates (H=26.98, df=5, p^{**}=5.64311$E-05$, α=0.05). A Tukey post hoc test reveals that $P3+ gyro is significantly more accurate than Jackknife (Cohen's d=7.56), $PN+ (Cohen's d=8.79) and μV (Cohen's d=23.83) ($p^{***}\leq$0.001) with a large effect size. It also reveal that μF is significantly more

accurate than Jackknife (Cohen's d=7.21), $PN+ (Cohen's d=8.44) and μV (Cohen's d=23.48) ($p^{***} \leq 0.001$) with a large effect size. In the Figure we can also see that $P3+ acc is also significantly more accurate than Jackknife (Cohen's d=6.6), $PN+ (Cohen's d=7.83) and μV (Cohen's d=22.87) ($p^{***} \leq 0.001$) with a large effect size. Finally we can observe than μV is significantly less accurate than Jackknife (Cohen's d=16.27) and $PN+ (Cohen's d=15.04) ($p^{***} \leq 0.001$) with a large effect size.

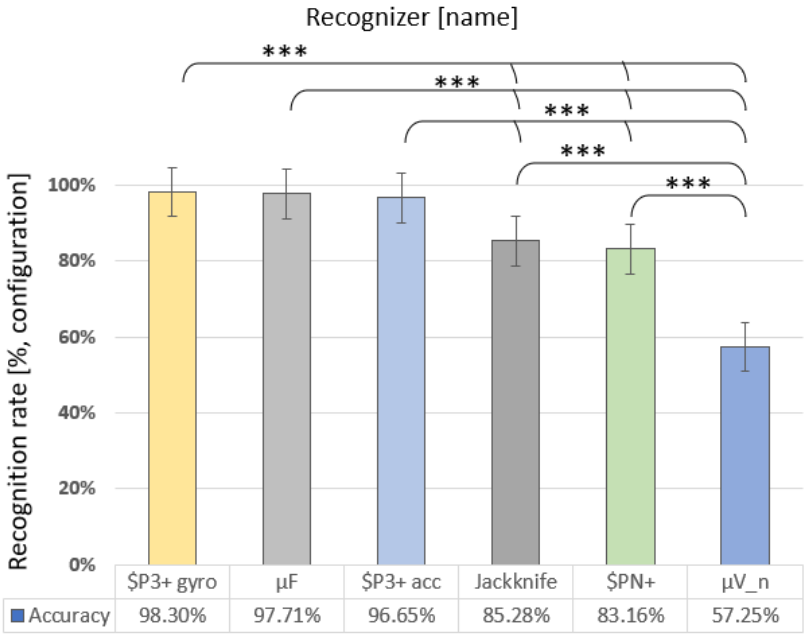


Figure 4.11: Configurations of recognizers leading to the mean accuracy in the user-dependent scenario with the new dataset. Error bars show a confidence interval of 95% computed on respective configurations.

Fig. 4.11 shows the average accuracy achieved between all our tested configurations in the user-dependent scenario. The Kruskall-Wallis statistical test and the Tukey post hoc test bring the same observations as before for Fig. 4.10.

4.2 Summary

In this chapter, we can find all the tests that were carried out on the gestures and the dataset that we created and described in Chapter 3. These tests allowed us to realize that some gestures were not differentiable enough to be kept and so we make a selection to keep only 21. They also allowed us to realize that the results obtained were lower than expected. Then, we analyzed the results and realized that there was a lot of variation for some participants. Therefore, we recorded a new dataset which allowed us to obtain better recognition results. The tests were carried out user-independent and user-dependent and the best results are returned by the user-dependent tests. Thanks to these different tests and analyses, we can conclude that we would use the Jackknife recognizer for our further experiment and application.

Chapter 5

Bimanual Part

This chapter will be dedicated to the bimanual part of the thesis. It will be explained the limits which we will be able to face and the solutions if there are any which could be brought. We will also speak separately about the bimanual gestures performed by one person with two bracelets and the bimanual gestures performed by two people in a collaborative manner, each with a device.

5.1 Why Create and Use Bimanual Gestures?

As defined in Chapter 2, collaborative gestures include at least two people because they are not supposed to be able to be performed alone. The purpose of these gestures is indeed that people work together for a common goal. Several studies already exist on collaborative tasks such as the one of Ou [39] performed on a Tablet PC but in most cases the gestures are not really collaborative because the gestures are performed one after the other individually. So, we decided to add real collaborative gestures to Kinemic. Moreover, by adding collaborative gestures, it allows to extend the number of gestures that can be performed with the bracelet and consequently the number of different actions that can be performed with it. For the same reason, we also decided to add bimanual gestures performed by a single person. Indeed, due to two bracelets, a single person would have access to an even wider and more diversified range of gestures. Therefore, we will explore these two different cases in the rest of this chapter.

5.2 Bimanual Testing Limitation and Hypothesis

For all the part carried out in bimanual with two bracelets, whether for the configuration where two people each have a bracelet or where one person wears both bracelets, we were unable to carry out tests to test the effectiveness of the recognizers and the differentiation between the new gestures. This is because the framework we are using for our tests and experiments, QuantumLeap, has not yet been adapted at the test level to support two data streams.

Nevertheless, since unimanual tests have been carried out, we can expect equivalent results for the collaborative part, since each instance has been tested individually. However, it will be possible to obtain combined results, if recognizer 1 recognizes gesture A at 80% and recognizer 2 recognizes gesture B at 70%, knowing that A and B are collaborative, it

is likely that the overall accuracy will not exceed the minimum, i.e. 70% or even a little less.

5.3 Bimanual with one Person

In this section we will develop our observations in relation to the use of 2 Kinemic by the same person, i.e. one bracelet on each wrist. In relation to this part we have faced 2 different cases. First, the unimanual gestures that could be used on the left and on the right at the same time, for example I perform a left swipe with my left hand and a left swipe with my right hand. Second, new gestures that only exist bimanually on one person, such as applause.

5.3.1 Unimanual gestures used in a bimanual way

In the first case, where the unimanual gestures were to be reused in a bimanual way, we used most of the unimanual gestures we devised and described in [Table 3.1] in Chapter 3. We selected the most relevant ones, those that seemed to us the most feasible by both hands at the same time and that made the most sense. For example, we did not choose gestures such as hello, because doing it twice would not have made much sense. On the other hand, we have kept gestures such as swipe right which, when done with both hands at the same time, could have a different intensity than a swipe right done with only one hand. We also grouped together in a single gesture those that were grouped together, such as eartouch right and eartouch left, which became eartouch. The set of gestures that we have imagined for this configuration can be found at the beginning of the [Table 3.2] in Chapter 3.

Unfortunately, the results we were able to obtain manually were not conclusive. Indeed, we could observe that it did not recognize the gesture made with the left hand, for example by making a swipe right, it would most of the time confuse it with the swipe left. This can be explained by the fact that all the gestures in the dataset were recorded on the right hand of the participants and that using the other hand to perform a gesture makes the coordinates vary in a different way. In order to overcome this problem, it would have been necessary to either record the gestures on the left hand as well, or to record the gestures that were to be used bimanually on both hands of the participants in addition to just the right hand.

5.3.2 New two-bracelet gestures on one person

In the second situation, we decided to create, in addition to unimanual gestures, bimanual gestures that can only be performed in this configuration. We made this choice in order to be able to take advantage of all the possibilities of this configuration and to add other relevant gestures in other contexts or allowing to activate other functions. These are also gestures that can only be performed with both hands at the same time, so they must also be possible without too much difficulty. We have therefore imagined gestures such as applause that cannot be done with one hand. The end of the [Table 3.2] of Chapter 3 shows all the gestures we have imagined. Among these we also find gestures such as zoom in or zoom out which could allow us to interact with a photo software, for example to

zoom in and out on an image. In the same context of image processing, gestures such as stretch and fusion could also be very relevant.

Compare to the previous configuration, we were able to make quite different observations. First, we realized that the position of the bracelets was important. By position, we mean putting this bracelet on the left wrist and that bracelet on the right wrist. If the bracelets were put on in the opposite order, the desired gesture would never be recognised. To solve this problem and so that the gestures are always recognised regardless of the order in which the bracelets are put on, we added the possibility of recognising the gesture in one direction or the other.

5.4 Bimanual in Collaboration

In a second phase we also decided to use the bracelets collaboratively to extend its use to several people. In our case, as we only have two wristbands we will limit ourselves to two people collaborating but later on, the wristbands can be adapted to be used with more people.

The use of wristbands in collaborative mode allows us to explore other types of gestures and other contexts of use. As briefly mentioned in Chapter 2, we can find articles like Delamare's on gesture combination [8] in which we can see that collaborative gestures bring something extra compared to unimanual gestures. In the article, they explain in particular that such a use brings the possibility to reuse gesture definitions. They give the example already mentioned: *"a first gesture - a signifier gesture - could signify a context (e.g., 'Light') while a second gesture - an operator gesture - could trigger a command (e.g., 'Switch on')"*[8]. In the same way, in our collaborative gestures that we can find in the [Table 3.3] of Chapter 3, we can imagine, for example, the context of a slide presentation with one of the two people using the swipe to move from one slide to the next and the other one using the drag to zoom in or out directly afterwards. In our table we have grouped the gestures by two but we can obviously imagine many more combinations depending on the situation and the context of use. To create these different gestures we have taken for the first part unimanual gestures that can be used in collaboration like the swipe and the move mentioned in the previous example but we have also imagined other gestures like hand shake that can only make sense if they are used in a collaborative configuration.

In addition, there are many other benefits of collaborative use, Morris's study on collaborative gestures [34] explains different motivating scenarios in which collaborative gestures can be used. Several of these benefits can also be applied to collaborative gestures. First, allowing several people to interact together with the same system and whose collaboration is necessary for its proper functioning can improve the cohesion of a group as well as the participation of its different members. Second, these kinds of gestures used in an important context can also allow for better reflection before making a decision, as the action to be taken will involve several people. Similarly, in a document management system where some documents cannot be modified by other users without the owner's permission, collaborative gestures can help to require the presence of the owner and the person wishing to modify in order to access the document. Finally, collaborative gestures

could also be used in the world of video games, for example, to allow players to interact together in the same game.

As in the other configurations, we sought to create the best possible gestures. In Morris's article, we can also find the three criteria they used to create and design their gestures *"(1) using postures and movements based on analogy to "real-world" actions when possible, (2) creating gestures distinct enough to be accurately recognised by our system given the limitations of the DiamondTouch as a recognition device, and (3) including gestures that involved several styles of cooperation"*[34]. In our case, we also applied these three criteria, adapted to our technologies. The first criterion was applied so that the gestures could be remembered by the users but also so that they made sense in the context, for example we can find a gesture representing the handshake. Then, we had to apply the second criteria so that our different recognizers could differentiate them since in Chapter 4 Benchmarking we had already failed to obtain the expected results for unimanual gestures. Finally, we applied the third criterion, so that our gestures would be relevant in collaboration, otherwise they could simply be used alone in an unmanual way, and the collaboration would not be interesting.

5.4.1 Allen

In this section we present possible configurations of our collaborative gestures using Allen's interval algebra for temporal calculus. These configurations allow us to show which gestures can or should take place at the same time, which ones can be executed sequentially, and to express certain precedence orders. In the [Table 5.2] we list all our collaborative bimanual gestures and the configuration(s) that are most relevant to each.

Nevertheless, this table mainly represents the configurations that are theoretically possible. In practice, all configurations that start or end at exactly the same time are impossible because two people cannot synchronize so perfectly to perform a gesture unless a time interval is added over which the gesture is still considered to have been performed at the same time. All Allen configurations such as X s Y, Y if X, X f Y, Y fi X and X = Y are therefore practically impossible. In practice, the configuration X = Y for example, will be considered rather as X d Y or Y di X.

Gesture	**Allen Configuration(s)**
Swipe Up	X=Y
Swipe left	X=Y
Swipe right	X=Y
Circle Clockwise	X=Y
Circle Counterclockwise	X=Y
Rotation Right-Left	X=Y
Rotation Left-Right	X=Y
Eartouch	X=Y
Hand Shake	X=Y, X o Y, X oi Y
Hello	X=Y, X o Y, X oi Y, X<Y, X>Y, X m Y, Y mi X, X s Y, Y si X, X d Y, Y di X, X f Y, Y fi X
Stretch	X=Y
Fusion	X=Y
Agreement	X=Y
Disagreement	X=Y
Mitigate	X=Y
Transfer	X < Y, X m Y
Carry	X=Y
Push	X=Y
Pull	X=Y

Table 5.2: Allen configurations for collaborative gestures.

5.5 Summary

So in this chapter we could talk about all the bimanual parts of this experiment. There are no tests as such in this part as we explained in the first section, but we could manually test their efficiency and realize the problems we could encounter. The gestures that cannot be used for the moment will not be used in the future, but we have set out some possible solutions to avoid the problems we encountered.

Chapter 6

Application

In order to test whether the Kinemic bracelet was effective with the recognizer, we selected from the many tests we carried out, we decided to create an application representing a PowerPoint simulation. This allowed us to test whether the device was effective in a situation in which it could be used regularly. We chose the demonstration of a PowerPoint for unimanual gestures because it seemed to us to be one of the most obvious and basic areas in which it could be useful. In addition, the presentation allowed us to use our imagination to employ various existing or invented gestures to interact with it.

6.1 Integration with QuantumLeap

6.1.1 Development

We started by developing an application representing a fake PowerPoint, containing different images and videos to be able to use the Kinemic to interact with [Figure 6.1]. We initially developed the application using just HTML and Javascript but when we wanted to integrate it with the QuantumLeap framework we had problems using the API. Therefore, we decided to develop our application using React. We also used several libraries such as React Slideshow Image and ReactPlayer to represent the scrolling of the different slides and the interaction with images and videos.

Figure 6.1: Example of slides from the application.

6.1.2 Segmenter

The segmenters already implemented in QuantumLeap (see Section 2.9) are generic and were not adapted to Kinemic. Indeed, we tried Machete but it did not segment correctly between each gesture. Using a sliding window is also complicated because our gestures do not have the same frame length in time, so it was difficult to use a window of a precise size. This can be seen by looking at the difference between swipe down [Figure 6.2] and forward rotation [Figure 6.3], the former contains about 80 frames while the latter contains just over 140 frames.

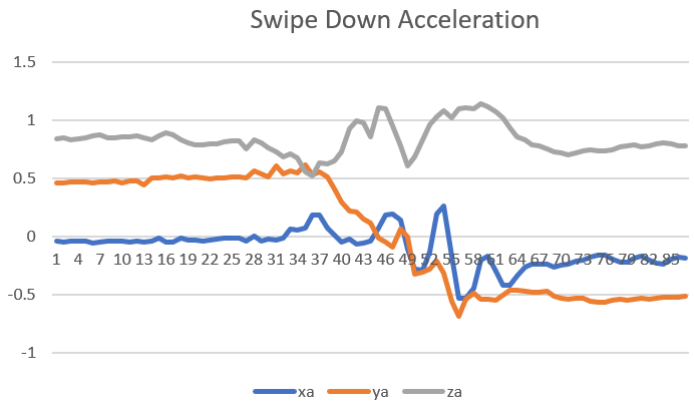


Figure 6.2: Swipe Down acceleration registered in the database graph.

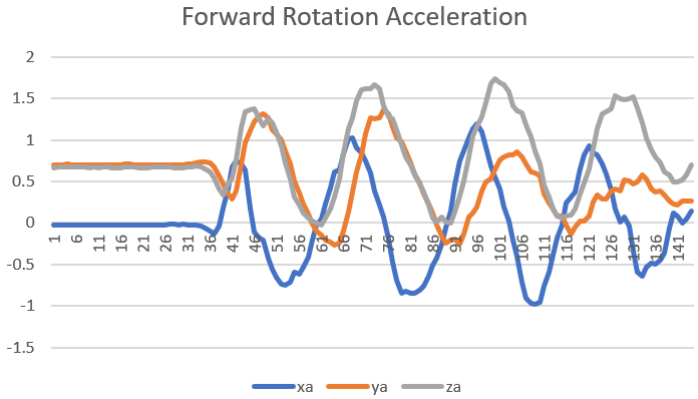


Figure 6.3: Forward Rotation acceleration registered in the database graph.

The threshold based segmenter is not adapted to our points because we can't find threshold values consistent with our acceleration and gyroscope data due to their type. Indeed, as can be seen in [Figure 6.3] for forward rotation the acceleration only goes up and down and it is the same for the gyroscope. They increase back to their stable value and then drop lower to rise again. With a threshold gesture such as forward and backward rotation are divided into several gestures.

So, we implemented a segmenter especially for our use of Kinemic that we called Kinemic Window. The code is in the Appendix B.1. It works on the basis of a threshold, but is calculated on the last 5 points recorded by the Kinemic. Indeed, the beginning of a gesture is characterized by the fact that the acceleration points vary. We then observe the standard deviation of one of the acceleration points (xa) on the last five sent points. If the latter exceeds 0.5, then a gesture is started and detected. The value was chosen after our observations, we observed the standard deviation when we did not move and then when we started a movement. Then, the gesture continues as long as the standard deviation of the acceleration points of the last 5 points is above 0.07. This value is useful because some of our gestures are longer in time, such as forward rotation, and during this gesture the standard deviation decreases with each circle drawn in the air and the segmenter should not cut off the gesture.

Once the gesture is detected, we add to the beginning the last 10 frames recorded by the bracelet before the beginning of the gesture. This is due to the fact that during the recording of the dataset we did not filter the beginning of the gestures and that there is always a flat moment at the beginning of each gesture template as we can see in [Figure 6.2] and [Figure 6.3]. The addition allows the recorded gesture to be better recognized and associated with the recorded gestures of the dataset because we are sure to integrate the frames of the beginning of the gesture and not only at the beginning of the detection of the variation and thus get closer to the dataset. This new recognizer allowed us to recover the data that can be seen in [Figure 6.4] and whose resemblance to [Figure 6.3], which is the gesture present in the database, can be clearly seen.

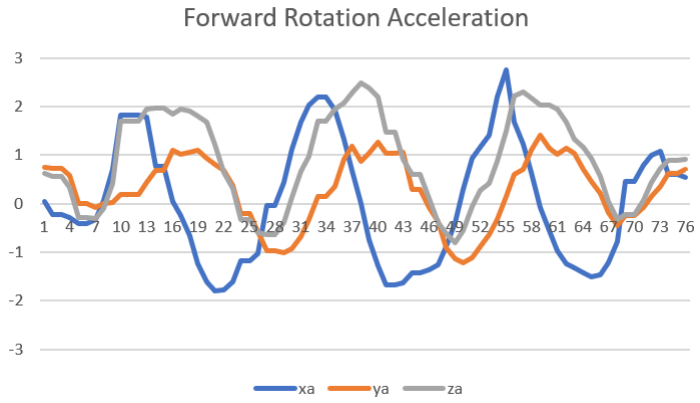


Figure 6.4: Forward Rotation acceleration registered by the segmenter.

We have also created a segmenter adapted to two bands called "Kinemic Window 2 bands" to be able to configure the pipeline for two bands and thus be adapted to **bimanual** and **collaborative** situations. The difference lies in the fact that the start of recording conditions are adapted to two bands exactly. Segmentation starts when one of the two wristbands starts to move and ends when both wristbands are not moving anymore, they have finished their movement.

6.1.3 Configuration

In order to be able to use QuantumLeap with our application, as well as integrate the recognizers and segmenters, we had to configure the different parts of the pipeline that we were interested in, i.e. the sensor(s), the segmenter, the dynamic dataset(s) and finally the dynamic recognizer.

Sensor(s)

For the sensor, you need to provide the framework with the name of the bracelet you want to use and an identifier that will be used later to differentiate it. If you want to use the application in unimanual mode, you only need to configure one module as shown in [Figure 6.5]. On the other hand, if you want to use it in bimanual, you must keep the configuration of the [Figure 6.5] and add a second module as shown in the [Figure 6.6].

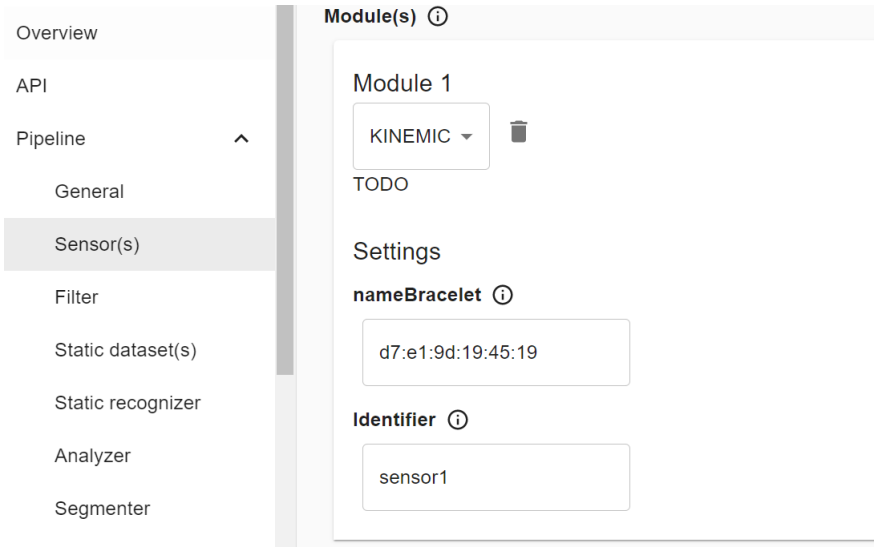


Figure 6.5: Module 1 of the sensor.

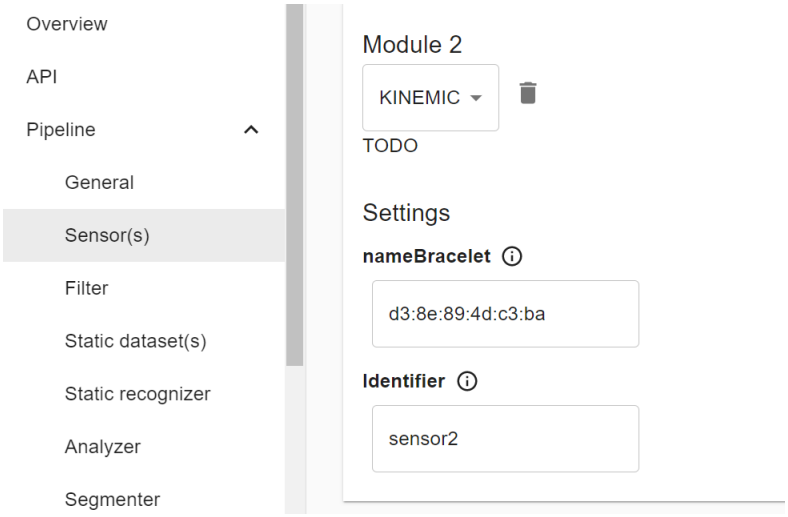


Figure 6.6: Module 2 of the sensor.

Segmenter

For the segmenter, it is enough to select Kinemic Segmenter as well as the points of sensor 1 if you want to use the bracelet in unimanual mode as shown in [Figure 6.7]. To use it in bimanual mode, select the other segmenter, i.e. Kinemic Segmenter 2 and select the points of sensor 1 and sensor 2 as shown in [Figure 6.8]. For the motion threshold box, it is not used in either segmenter so the value is not taken into account and is therefore not important.

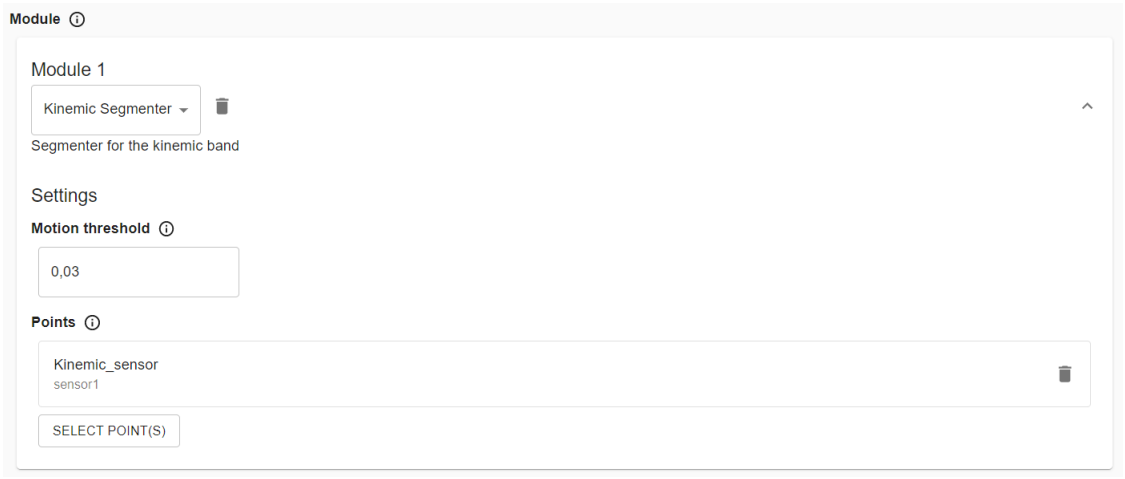


Figure 6.7: Unimanual Segmenter.

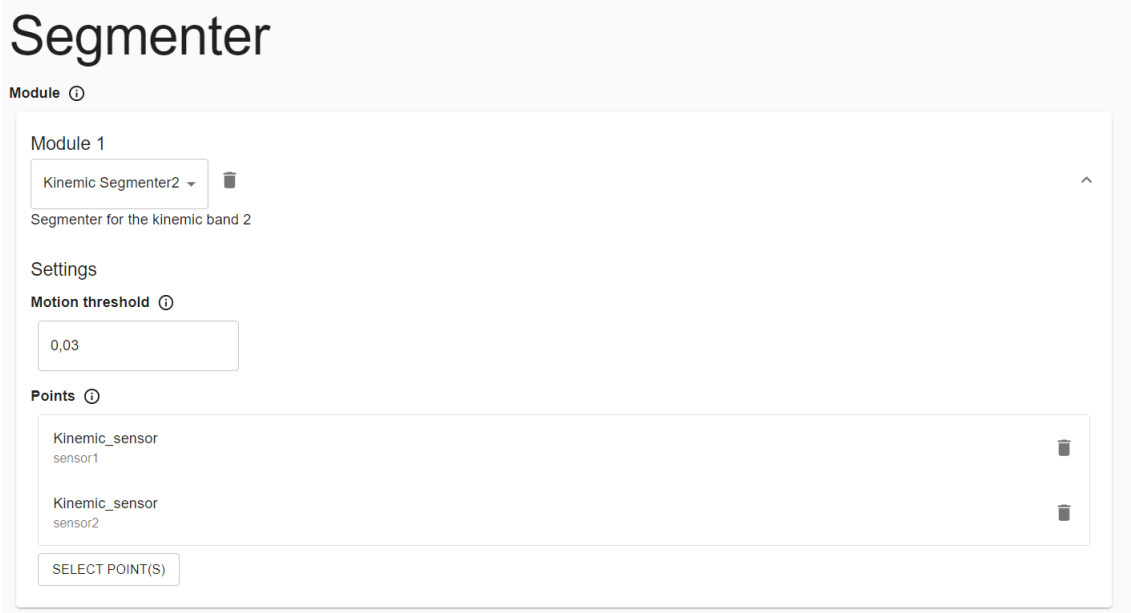


Figure 6.8: Bimanual Segmenter.

Dynamic Dataset(s)

To select the dataset(s) we selected the Unified Dataset Loader. As for the sensor, if we want to use the application in unimanual mode, we only need to create a module and select the dataset we want to use and fill in the identifiers with the same ones as the sensors [Figure 6.9]. To switch to bimanual, you have to create a second module in the same way with the identifiers of the second sensor as in the [Figure 6.10].

Figure 6.9: Module 1 of dataset.

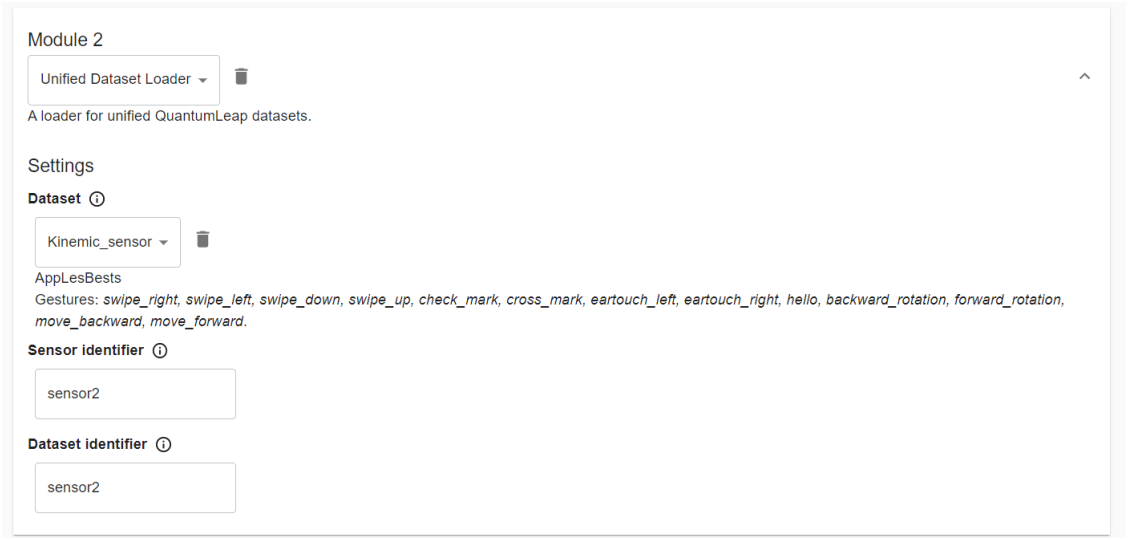


Figure 6.10: Module 2 of dataset.

Dynamic recognizer

To configure the recognizer, it also depends on whether you want to use one or both wristbands. In the case of unimanuel use, you have to configure a module by choosing the recognizer, in our case Jackknife with 16 sampling points, and select the points of sensor 1 configured in the sensor part. It is possible not to select any training gestures so that all of them are used as in [Figure 6.11]. In the case of bimanual use, you have to select the Multiple Recognizers (concat result) and then select in two different modules the two recognizers you want to use, in our case Jackknife with always 16 sampling points. You also have to select the sensor 1 points for the first one and sensor 2 for the second one and choose the corresponding sensor points each time in the Training gestures part as in the [Figures 6.12 and 6.13]. Finally, as in the bottom of [Figure 6.13], you must enter the separator / and the name that will be returned when no gesture is found, NO_GESTURE.

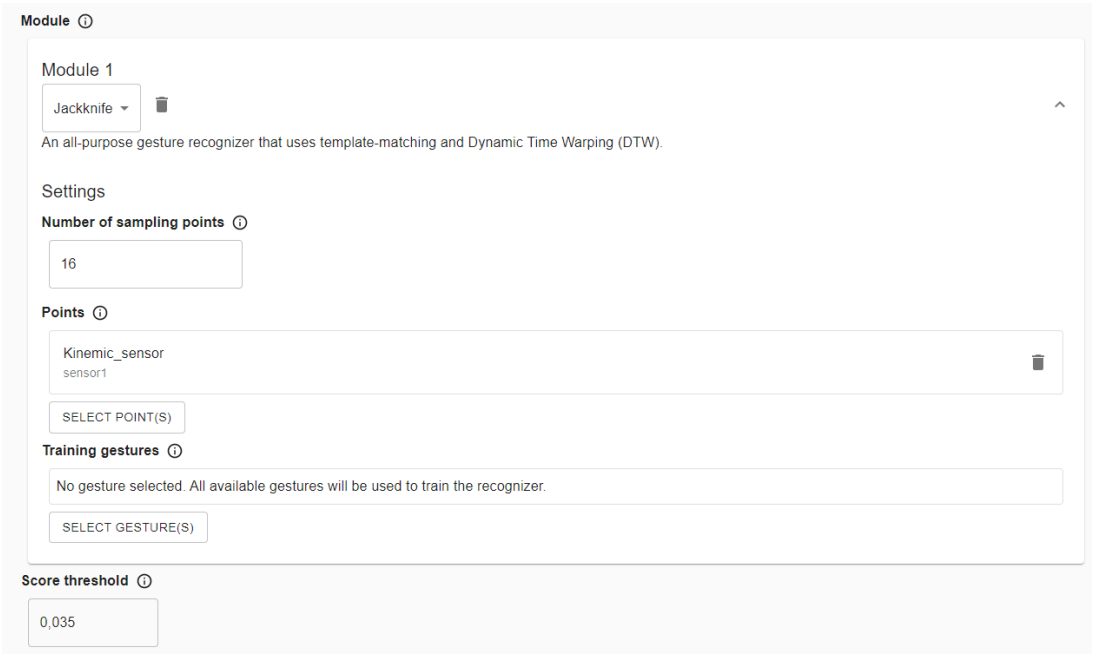


Figure 6.11: Unimanual Recognizer.

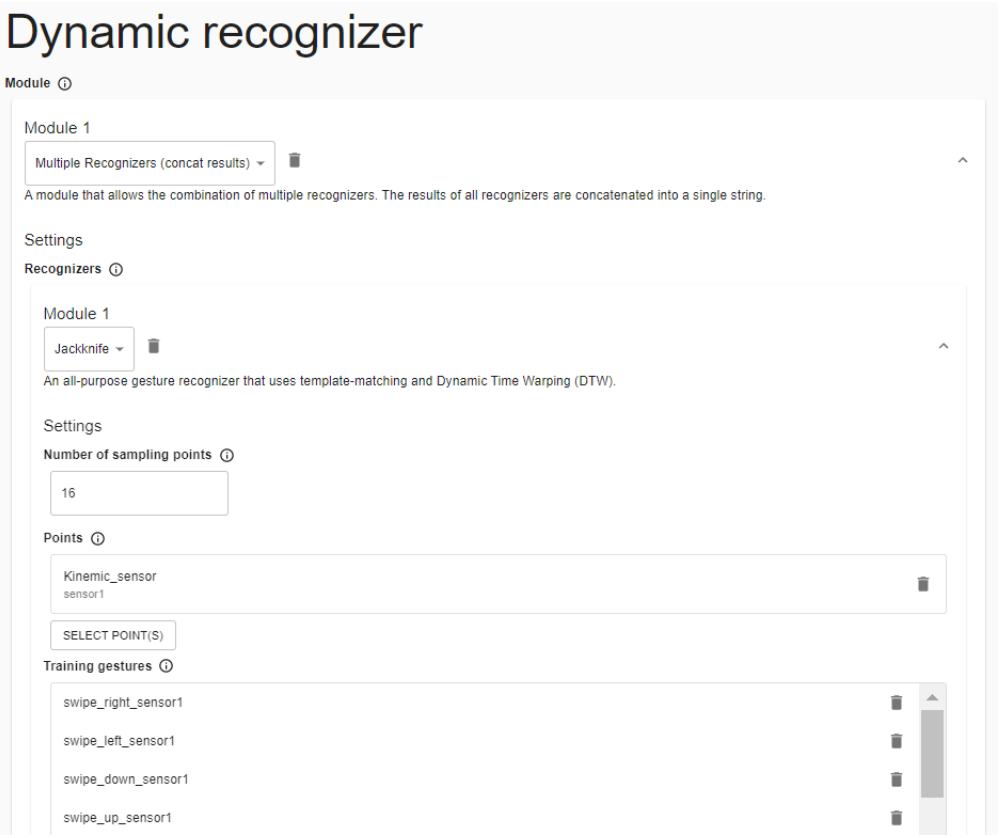


Figure 6.12: Module 1 of bimanual recognizer.

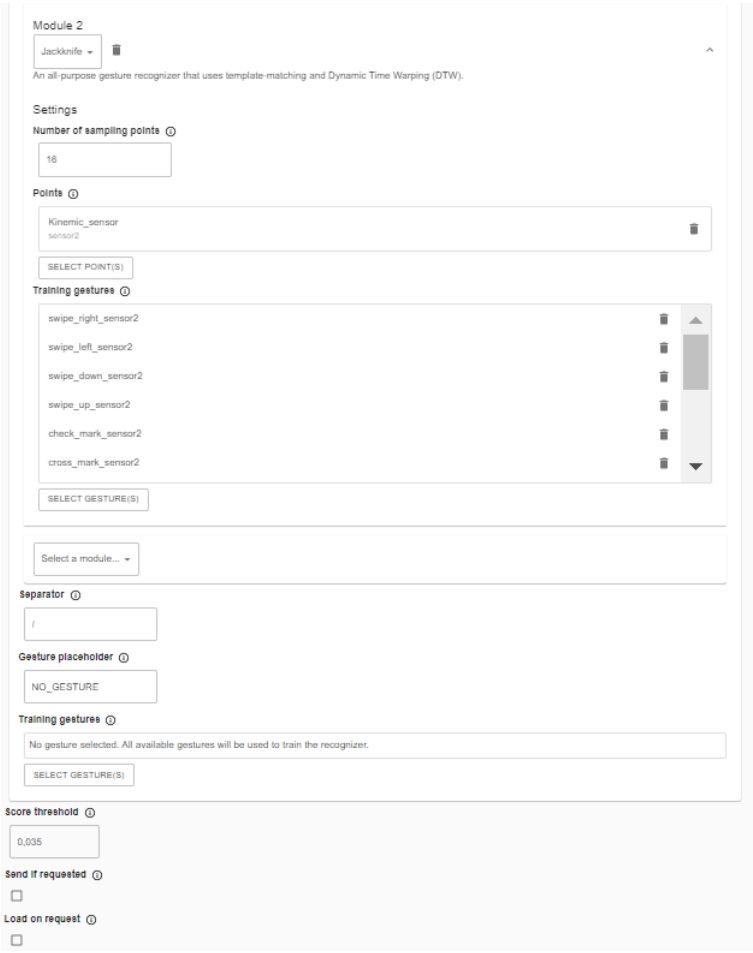


Figure 6.13: Module 2 of bimanual recognizer.

6.2 Unimanual Gestures Selection

To use our application we decided to use only thirteen of our gestures at first to interact and test the effectiveness of Kinemic. We took existing gestures with the bracelet, but also new ones that we had imagined to vary and see if they could really be effective in a real case. We were also careful to take gestures that made sense in relation to the action, either based on examples from everyday life or on previous studies, such as the one of Benjamin Alearts mentioned in Section 3.5. So, we recreated a dataset with only these 13 gestures:

- Swipe right : Allows the user to move to the next slide. We chose this gesture because it is already a common movement for navigating on different devices, such as the tablet. In addition, in the study by Benjamin Alaerts, the swipe principle is also used to browse elements.
- Swipe left : Unlike the previous gesture, it allows you to return to the previous slide, and therefore we have chosen it for the same reasons and by logic to use the opposite gesture for an opposite action.
- Swipe up : This gesture increases the sound of a running video. We chose this gesture because it represents the act of increasing something and we also based it on Benjamin's study where people had made an upward hand movement with the palm of the hand upwards to increase the volume.

- Swipe down : Allows the sound to be turned down. We have justified its choice for the same reasons as the previous gesture.
- Eartouch right : Touching your right ear mutes the sound of a video. We chose this gesture for this action because the ear is the organ directly related to the ear. Furthermore, in this article [5] exploring gestures for ear-based interactions, they explain: *"an average of 68% of participants performed gestures involving touching the ear for tasks that adjust sound, i.e., volume up/down and mute/unmute speaker".*
- Eartouch left : The exact opposite of the previous gesture, touching your left ear activates the sound of a video. We can therefore justify this choice in the same way as the gesture that activates it.
- Check mark : This would start the presentation. It is a gesture used a lot to choose or select elements. In our case, it would be used to select a slideshow to start from a list of slideshows or to select a slide to start.
- Cross mark : Would allow you to exit or stop the presentation. We have chosen the cross to represent this action in order to correspond to the opposite of the check mark which allows to start it.
- Forward rotation : This gesture allows the user to start a video present on the slide.
- Backward rotation : Conversely, it can be used to stop a video in progress as it represents the opposite gesture to the previous one.
- Hello : Saying hello allows the user to return to the beginning of the presentation, i.e. to slide number 1. We chose this gesture intuitively because in everyday life it represents the first thing you do when you meet someone and therefore a certain introduction. Saying hello would therefore allow us to always go back to the beginning.
- Move forward : Pushing your hand forward allows you to zoom in on the slide that contains an image or text. In this article [36] we can find the same design choice justified in this way: *"As in desktop applications such as Google Maps or NASA's WorldWind, linear techniques zoom in by moving forward towards the display and zoom out by moving backwards;".* This article [44] also includes a gesture that allows you to zoom in on a map by moving towards the map. This is quite similar to our gesture.
- Move backward : Unlike move forward, it allows you to zoom out on slides containing images or text. Its choice can be justified with the same articles as for the previous movement and because it is its opposite.

6.3 Bimanual Adaptation

The application we created was mainly designed to be used with a single wristband, but we also tested it with the 2 Kinemic. Indeed, it could be used either in the two-handed configuration with one person or in the collaborative two-handed configuration with two people. In the first case, we could only use the gestures that have been directly recorded with the two bracelets as explained in Section 5.2.1. but we could imagine using applause

to go to the end of the presentation or even more visual gestures such as fusion, stretch, zoom in or zoom out to manipulate images of the presentation, it would be enough to add them to the application and launch it for two bracelets. In the second case, all the unimanual gestures could be used and we could imagine combinations such as the first person making the Swipe right gesture to go to the next slide and at the same time the other person making the Move forward gesture to zoom in on this next slide.

6.4 Summary

In this chapter, we can therefore find the integration of the use of Kinemic in the QuantumLeap framework, as well as the adaptations we had to make so that the Kinemic data could be correctly recognized. Thanks to the frameword we were able to create an application demonstrating a fake Powerpoint presentation and we were able to interact with it thanks to the gestures of the bracelet, both unimanual and bimanual.

Chapter 7

Conclusion and Future Work

7.1 Review

In today's world with the amount of computer based system data, human-computer interaction is becoming increasingly important. Therefore, the aim of this thesis was to answer the question: "To what extent could we support unimanual and bimanual wrist-based user-defined gestures for a single user and for two users in collaboration?" In order to answer this question we decided to use the Kinemic device, a wristband that allows interaction with an application. After much research, we did not find much work done with this device. Therefore, we decided to analyze its unimanual and bimanual recognition capabilities, as we had two wristbands. As described in Chapter 3, we decided to create a dataset with the already existing gestures and by adding new gestures in the different configurations we wanted to explore. To create this dataset we called on 30 people of varying age, gender and education to perform the different gestures. Once this stage was completed, we then carried out numerous tests to identify the best recognizer to use with the Kinemic. In Chapter 4, we analyzed all the results we had obtained and all the changes we had to make to get the best possible results. These numerous tests were performed for the unimanual part of this thesis; it is in Chapter 5 we explain the different parts of the bimanual part. In the latter we manually analyzed the results that could be obtained by using the two bracelets on the same person or by using the bracelets in a collaborative way, i.e. with two people each having a bracelet and interacting with each other. Finally, to be able to apply all the results we had obtained, we decided to create an application and link the bracelet to it using the QuantumLeap framework. In Chapter 6, we go through the whole process of adapting the framework to our device and its data, as well as the different configurations that could be used in the framework and in the application. In summary, we support unimanual gestures from the wrist with the Kinemic wristband as a result of our study and implementation. We have also studied and explored bimanual gestures for one user and unimanual collaborative gestures, for which more research remains to be done because of the limitations encountered due to the framework used. We hope that our observations, modifications, and tests will be useful for future work in the field of human-computer interaction, whether with the Kinemic or other devices.

7.2 Critical Analysis

In this section, we will discuss the different limitations that we have encountered during this thesis and that we have presented in the different chapters.

Test Results The first results we obtained with the basic gestures present on the Kinemic were not very high, but we went on to test all the gestures we had recorded in the dataset. Unfortunately, we soon realized that the recognition rate was between 50% and 60% in the best cases, which is very low.

QuantumLeap Framework Several adaptations had to be made with the QuantumLeap framework in order to be able to test and create a usable application with the Kinemic. For the bimanual part, the framework had to be adapted so that it could receive two data streams. This adaptation could be done in the pipeline to use gestures with an application, but not for the test part. Therefore, we were unable to perform tests to see if the wristbands worked correctly for all bimanual gestures.

Bimanual Gestures For the bimanual part of this thesis, we realized that bimanual gestures performed by one person were not symmetrical and, therefore, not recognized if they were only recorded on one hand. Only gestures recorded on both hands at the same time were recognizable. Furthermore, the choice of putting a particular wristband on the left- or right-hand side of the wrist also matters, so that with our basic code the gestures were not recognized if the wristbands were not put on the right wrist.

7.3 Improvements and Future Works

Test Results As the results we obtained with the different recognizers did not exceed 60% at the beginning, we searched for a possible source of this to obtain better scores. To try and improve the results, we carried out different steps; first, we removed the gestures that were not very differentiable from the others to re-test the whole thing, but this was not enough to improve the results by any percentage. Then, we realized that the data varied enormously between different users when we looked at some of the gestures. So, we re-recorded a dataset of only 2 people to improve the results. Nevertheless, to obtain a really interesting dataset it would have been necessary to re-record the gestures for another thirty people, checking assiduously that each gesture had been correctly performed and recorded.

QuantumLeap Framework As tests could not be carried out for the bimanual part, as this functionality is not yet available in the QuantumLeap framework. In the future, it would be interesting to continue to develop the framework, in particular by allowing bimanual tests to be carried out. In this way, further research could be done in more depth on the use of two bracelets at the same time.

Bimanual Gestures As mentioned in Chapter 5, the second problem related to the order of the bracelets that was important was solved by allowing the application to recognize gestures with the bracelets placed in the two possible orders. Regarding the fact that unimanual gestures are not recognizable when they are performed with the left hand,

we explain that this could be solved by also recording these gestures with both hands so that they are recognizable when a person carries two bracelets and uses them.

In conclusion, re-recording gestures would be a first step. It would be necessary to rerecord with a new sample of people in a more assiduous and precise way and also to record the unimanual gestures in the configuration of the dominant hand of the person and in the opposite configuration to allow a greater adaptability to the user and to allow the complete functioning of the bimanual gestures. Afterwards, in order to conduct a complete study of the results via tests, it would be necessary to develop the QuantumLeap framework further so that it can adapt to bimanual and collaborative configurations. This will allow to complete our work and our study and to fully use the capabilities of the Kinemic. Moreover, we based ourselves on pattern matching algorithms because they adapt more quickly to the addition of a gesture or a template but also because in the context of our thesis we had a limited number of data. In the future, it would be possible to acquire more data and thus to consider the use of machine learning algorithms and more particularly those which allow a training in real time to keep the advantage of a fast adaptation.

There is still a lot of work and research in the field of gesture recognition and human-machine interaction because the amount of data is constantly increasing and new devices and new technologies to study and test are constantly emerging. Moreover, new recognition algorithms are emerging, especially machine learning algorithms that are trained in real time and bring promising results for the future of research [12, 23].

List of Figures

Glossary

API	Application Programming Interface.
HCI	Human-Computer Interaction.
IMU	Inertial Measuring Unit.
IT	Information Technology.
JSON	JavaScript Object Notation.
RQ	Research Question.
SD	Standard Deviation.
UCLouvain	Universite catholique de Louvain.
UD/UI	User Dependent / Independent.
UIs	User Interfaces.

Appendix A

Recognizers Code

A.1 $P3+

p3dollarplus.js

```
const { performance } = require('perf_hooks');

//
// Point class
//
function Point(x, y, z, id, angle = 0.0) {
  this.X = x;
  this.Y = y;
  this.Z = z;
  this.ID = id;
  this.Angle = angle; // normalized turning angle, $P+
}
//
// PointCloud class: a point-cloud template
//
function PointCloud(name, points) // constructor
{
  this.Name = name;
  this.Points = Resample(points, NumPoints);
  this.Points = Scale(this.Points);
  this.Points = TranslateTo(this.Points, Origin);
  this.Points = ComputeNormalizedTurningAngles(this.Points); // $P+
}
//
// Result class
//
function Result(name, score, ms) // constructor
{
  this.Name = name;
  this.Score = score;
  this.Time = ms;
}
//
// PDollarPlusRecognizer constants
//
var NumPoints = 32;
const Origin = new Point(0, 0, 0, 0);
//
// PDollarPlusRecognizer class
```

```
//
function P3DollarPlusRecognizer(numPoints) // constructor
{
  this.PointClouds = new Array();
  NumPoints = numPoints;
  //
  // The $P+ Point-Cloud Recognizer API begins here -- 4 methods:
   Recognize(), AddGesture(), RemoveGesture(), DeleteUserGestures()
  //
  this.Recognize = function (points) {
    var t0 = performance.now();
    var candidate = new PointCloud("", points);

    var u = -1;
    var b = +Infinity;
    for (var i = 0; i < this.PointClouds.length; i++) // for each point-
   cloud template
    {
      var d = Math.min(
        CloudDistance(candidate.Points, this.PointClouds[i].Points, b),
        CloudDistance(this.PointClouds[i].Points, candidate.Points, b)
      ); // $P+
      if (d < b) {
        b = d; // best (least) distance
        u = i; // point-cloud index
      }
    }
    var t1 = performance.now();
    return (u == -1) ? new Result("No match.", 0.0, t1 - t0) : new
   Result(this.PointClouds[u].Name, b > 1.0 ? 1.0 / b : 1.0, t1 - t0);
  }
  this.AddGesture = function (name, points) {
    this.PointClouds[this.PointClouds.length] = new PointCloud(name,
   points);
    var num = 0;
    for (var i = 0; i < this.PointClouds.length; i++) {
      if (this.PointClouds[i].Name == name)
        num++;
    }
    return num;
  }
  this.RemoveGesture = function (name) {
    this.PointClouds = this.PointClouds.filter(pointCloud => pointCloud.
   Name !== name);
  }
  this.DeleteUserGestures = function () {
    this.PointClouds.length = NumPointClouds; // clears any beyond the
   original set
    return NumPointClouds;
  }
}
//
// Private helper functions from here on down
//
function CloudDistance(pts1, pts2, minSoFar) // revised for $P+
{
  var matched = new Array(pts1.length); // pts1.length == pts2.length
  for (var k = 0; k < pts1.length; k++)
```

```
    matched[k] = false;
  var sum = 0;
  for (var i = 0; i < pts1.length; i++) {
    var index = -1;
    var min = +Infinity;
    for (var j = 0; j < pts1.length; j++) {
      var d = DistanceWithAngle(pts1[i], pts2[j]);
      if (d < min) {
        min = d;
        index = j;
      }
    }
    matched[index] = true;
    sum += min;
    if (sum >= minSoFar) return sum; // early abandoning
  }
  for (var j = 0; j < matched.length; j++) {
    if (!matched[j]) {
      var min = +Infinity;
      for (var i = 0; i < pts1.length; i++) {
        var d = DistanceWithAngle(pts1[i], pts2[j]);
        if (d < min)
          min = d;
      }
      sum += min;
      if (sum >= minSoFar) return sum; // early abandoning
    }
  }
  return sum;
}
function Resample(points, n) {
  var I = PathLength(points) / (n - 1); // interval length
  var D = 0.0;
  var newpoints = new Array(points[0]);
  for (var i = 1; i < points.length && newpoints.length <= n; i++) {
    if (points[i].ID == points[i - 1].ID) {
      var d = Distance(points[i - 1], points[i]);
      if ((D + d) >= I) {
        var qx = points[i - 1].X + ((I - D) / d) * (points[i].X - points
    [i - 1].X);
        var qy = points[i - 1].Y + ((I - D) / d) * (points[i].Y - points
    [i - 1].Y);
        var qz = points[i - 1].Z + ((I - D) / d) * (points[i].Z - points
    [i - 1].Z);
        var q = new Point(qx, qy, qz, points[i].ID);
        newpoints[newpoints.length] = q; // append new point 'q'
        points.splice(i, 0, q); // insert 'q' at position i in points s.
    t. 'q' will be the next i
        D = 0.0;
      }
      else D += d;
    }
  }
  if (newpoints.length == n - 1) // sometimes we fall a rounding-error
    short of adding the last point, so add it if so
    newpoints[newpoints.length] = new Point(points[points.length - 1].X,
     points[points.length - 1].Y, points[points.length - 1].Z, points[
    points.length - 1].ID);
```

```
  return newpoints;
}
function Scale(points) {
  var minX = +Infinity, maxX = -Infinity, minY = +Infinity, maxY = -Infinity, minZ = +Infinity, maxZ = -Infinity;
  for (var i = 0; i < points.length; i++) {
    minX = Math.min(minX, points[i].X);
    minY = Math.min(minY, points[i].Y);
    minZ = Math.min(minZ, points[i].Z);
    maxX = Math.max(maxX, points[i].X);
    maxY = Math.max(maxY, points[i].Y);
    maxZ = Math.max(maxZ, points[i].Z);
  }
  var size = Math.max(maxX - minX, maxY - minY, maxZ - minZ);
  var newpoints = new Array();
  for (var i = 0; i < points.length; i++) {
    var qx = (points[i].X - minX) / size;
    var qy = (points[i].Y - minY) / size;
    var qz = (points[i].Z - minZ) / size;
    newpoints[newpoints.length] = new Point(qx, qy, qz, points[i].ID);
  }
  return newpoints;
}
function TranslateTo(points, pt) // translates points' centroid
{
  var c = Centroid(points);
  var newpoints = new Array();
  for (var i = 0; i < points.length; i++) {
    var qx = points[i].X + pt.X - c.X;
    var qy = points[i].Y + pt.Y - c.Y;
    var qz = points[i].Z + pt.Z - c.Z;
    newpoints[newpoints.length] = new Point(qx, qy, qz, points[i].ID);
  }
  return newpoints;
}
function ComputeNormalizedTurningAngles(points) // $P+
{
  var newpoints = new Array();
  newpoints[0] = new Point(points[0].X, points[0].Y, points[0].Z, points[0].ID); // first point
  for (var i = 1; i < points.length - 1; i++) {
    var dx = (points[i + 1].X - points[i].X) * (points[i].X - points[i - 1].X);
    var dy = (points[i + 1].Y - points[i].Y) * (points[i].Y - points[i - 1].Y);
    var dz = (points[i + 1].Z - points[i].Z) * (points[i].Z - points[i - 1].Z);
    var dn = Distance(points[i + 1], points[i]) * Distance(points[i], points[i - 1]);
    var cosangle = Math.max(-1.0, Math.min(1.0, (dx + dy + dz) / dn)); // ensure [-1,+1]
    var angle = Math.acos(cosangle) / Math.PI; // normalized angle
    newpoints[newpoints.length] = new Point(points[i].X, points[i].Y, points[i].Z, points[i].ID, angle);
  }
  newpoints[newpoints.length] = new Point( // last point
    points[points.length - 1].X,
    points[points.length - 1].Y,
```

```
    points[points.length - 1].Z,
    points[points.length - 1].ID);
  return newpoints;
}
function Centroid(points) {
  var x = 0.0, y = 0.0, z = 0.0;
  for (var i = 0; i < points.length; i++) {
    x += points[i].X;
    y += points[i].Y;
    z += points[i].Z;
  }
  x /= points.length;
  y /= points.length;
  z /= points.length;
  return new Point(x, y, z, 0);
}
function PathLength(points) // length traversed by a point path
{
  var d = 0.0;
  for (var i = 1; i < points.length; i++) {
    if (points[i].ID == points[i - 1].ID)
      d += Distance(points[i - 1], points[i]);
  }
  return d;
}
function DistanceWithAngle(p1, p2) // $P+
{
  var dx = p2.X - p1.X;
  var dy = p2.Y - p1.Y;
  var dz = p2.Z - p1.Z;
  var da = p2.Angle - p1.Angle;
  return Math.sqrt(dx * dx + dy * dy + dz * dz + da * da);
}
function Distance(p1, p2) // Euclidean distance between two points
{
  var dx = p2.X - p1.X;
  var dy = p2.Y - p1.Y;
  var dz = p2.Z - p1.Z;
  return Math.sqrt(dx * dx + dy * dy + dz * dz);
}

module.exports = {
  Point,
  P3DollarPlusRecognizer
};
```

index.js

```
const AbstractDynamicRecognizer = require('../../../../framework/modules
    /recognizers/dynamic/abstract-dynamic-recognizer').
    AbstractDynamicRecognizer;
const P3DollarPlusRecognizer = require('./p3dollarplus/p3dollarplus').
    P3DollarPlusRecognizer;
const Point = require('./p3dollarplus/p3dollarplus').Point;
const { parsePointsNames } = require('../../../../framework/utils');

class Recognizer extends AbstractDynamicRecognizer {

  static name = "P3DollarPlusRecognizer";
```

```
9
10  constructor(options, dataset) {
11    super();
12    this.samplingPoints = options.samplingPoints;
13    this.selectedPoints = parsePointsNames(options.points);
14    this.recognizer = new P3DollarPlusRecognizer(this.samplingPoints);
15    if (dataset !== undefined) {
16      dataset.getGestureClasses().forEach((gesture) => {
17        gesture.getSamples().forEach(sample => {
18          this.addGesture(gesture.name, sample);
19        }
20        );
21      });
22    }
23  }
24
25  addGesture(name, sample) {
26    let points = convert(sample, this.selectedPoints);
27    this.recognizer.AddGesture(name, points);
28  }
29
30  removeGesture(name) {
31    this.recognizer.RemoveGesture(name);
32  }
33
34  recognize(sample) {
35    let points = convert(sample, this.selectedPoints);
36    if (points.length === 0) {
37      return { name: "", score: 0.0, time: 0 };
38    }
39    let result = this.recognizer.Recognize(points);
40    return (result.Name === "No match.") ? { name: "", score: result.
    Score, time: result.Time } : { name: result.Name, score: result.Score
    , time: result.Time };
41  }
42
43  toString() {
44    return `${Recognizer.name} [ samplingPoints = ${this.samplingPoints
    }, points = ${this.selectedPoints} ]`;
45  }
46 }
47
48 function convert(sample, selectedPoints) {
49  let points = [];
50  selectedPoints.forEach((articulation, articulationID) => {
51    sample.paths[articulation].strokes.forEach((stroke, strokeId) => {
52      stroke.points.forEach((point) => {
53        // If multipoint, one stroke per articulation, otherwise, keep
    original strokes
54        let index = selectedPoints.length > 1 ? articulationID :
    strokeId;
55        coords = point.getCoordinates()
56        points.push(new Point(coords[0], coords[1], coords[2], index));
    //0,1,2 are for acceleration and 3,4,5 are for using gyroscope
57      });
58    });
59  });
60  return points;
```

```
61 }
62
63 module.exports = Recognizer;
```

Appendix B

Segmenter Code

B.1 Kinemic Window

index.js

```
const AbstractSegmenter = require('../../../framework/modules/segmenters/abstract-segmenter').AbstractSegmenter;

class Segmenter extends AbstractSegmenter {
    constructor(options, dataset) {
        super(options);
        this.lastFivePoints = [];
        this.index = 0;
        // Init frame buffers
        this.frameBuffer = [];
        this.beforeBuffer = [];
        this.saveBefore = [];


        this.checkLastFivePoints = (frame) => {
          let point = frame.getArticulation('Kinemic_sensor_sensor1').point; //id of the bracelet
          if(this.index > 4){
            if (ecartFun(this.lastFivePoints)>(0.5)){ //0.5 to start a gesture
              var ret = false;
              this.lastFivePoints = Object.values(shift(this.lastFivePoints, point));
              return ret;
            }
            else{
              var ret = true;
              this.lastFivePoints = Object.values(shift(this.lastFivePoints, point));
              return ret;
            }
          }
          this.lastFivePoints[this.index] = point;
          this.index = this.index + 1;
          var ret = false;
          return ret;
        }

```

```
34        this.checkLastFivePoints2 = (frame) => { //when the gesture
  started we change the variation accepted
35          let point = frame.getArticulation('Kinemic_sensor_sensor1').
  point; //id of the bracelet
36          if(this.index > 4){
37            if (ecartFun(this.lastFivePoints)>(0.04)){
38              var ret = false;
39              this.lastFivePoints = Object.values(shift(this.
  lastFivePoints, point));
40              return ret;
41            }
42            else{
43              var ret = true;
44              this.lastFivePoints = Object.values(shift(this.
  lastFivePoints, point));
45              return ret;
46            }
47          }
48          this.lastFivePoints[this.index] = point;
49          this.index = this.index + 1;
50          var ret = false;
51          return ret;
52        }
53
54      }
55
56     addGesture(name, sample) {
57     // Do nothing
58     }
59
60     removeGesture(name) {
61     // Do nothing
62     }
63
64     computeSegments(frame) {
65       if (this.beforeBuffer.length > 9){ //number of last frames to
  stock
66         this.beforeBuffer.shift();
67       }
68
69       this.beforeBuffer.push(frame);
70
71       if (!this.checkLastFivePoints(frame) || (this.windowstarted && !
  this.checkLastFivePoints2(frame))) {
72
73         if(!this.windowstarted){
74           console.log("start")
75           this.saveBefore = this.beforeBuffer.slice();
76           this.windowstarted = true;
77         }
78         this.frameBuffer.push(frame);
79       } else if (this.frameBuffer.length > 0 ) {
80         // The value(s) no longer reach the check last five points
81         console.log("end");
82         let frames = this.saveBefore.concat(this.frameBuffer); //concat
  with the last frames before the start of the gesture
83         //saveFrame(frames);
84         this.frameBuffer = [];
```

```
85          this.saveBefore = [];
86          this.windowstarted = false;
87
88          return [ frames ];
89        }
90        return [];
91      }
92
93      notifyRecognition() {
94      // Do nothing
95      }
96
97  }
98
99  // Return true if the difference between the two number is greater than
        the threshold
100 function difference(a, b, threshold){
101   if(Math.abs(a-b) > threshold){
102     return true;
103   }
104   return false;
105 }
106
107 //Shift the 4 points and add the fifth
108 function shift(tab, newPoint){
109   newtab = []
110   newtab[0] = tab[1];
111   newtab[1] = tab[2];
112   newtab[2] = tab[3];
113   newtab[3] = tab[4];
114   newtab[4] = newPoint;
115   return newtab
116 }
117
118 //Calculate the variance between the last 5 points
119 function ecartFun(tab){
120   mean = (tab[0].xa +tab[1].xa +tab[2].xa +tab[3].xa +tab[4].xa)/5;
121   tabbis = [];
122   tabbis[0] = Math.pow(tab[0].xa-mean, 2);
123   tabbis[1] = Math.pow(tab[1].xa-mean, 2);
124   tabbis[2] = Math.pow(tab[2].xa-mean, 2);
125   tabbis[3] = Math.pow(tab[3].xa-mean, 2);
126   tabbis[4] = Math.pow(tab[4].xa-mean, 2);
127   sum = tabbis[0]+tabbis[1]+tabbis[2]+tabbis[3]+tabbis[4];
128   variance = sum/(4);
129   ecart = Math.sqrt(sum);
130   return ecart;
131 }
132
133 module.exports = Segmenter;
```

Appendix C

Selected Gesture

C.1 First Group

Gesture name	Move Forward	Move Backward	High Five	Transfer	Receive	Throw	Catch	Hand Shake
Move Forward	/	/	/	/	/	/	/	/
Move Backward	OK	/	/	/	/	/	/	/
High Five	NOT OK (53%)	Middle (60%)	/	/	/	/	/	/
Transfer	OK	OK	OK	/	/	/	/	/
Receive	OK	OK	OK	OK	/	/	/	/
Throw	Middle (60%)	OK	NOT OK (55%)	OK	OK	/	/	/
Catch	OK	NOT OK (55%)	Middle (60%)	OK	OK	OK	/	/
Hand Shake	OK	OK	OK	OK	OK	OK	OK	OK

Table C.1: Table with the gestures of the first groups showing whether the gestures are differentiable (OK) or not (NOT OK) with their accuracy rate in %.

C.2 Second Group

Gesture name	Hello	Wind-Shield	Shoulder Right	Shoulder Left	Knock	Mix	Forward Rotation	Backward Rotation	Carry
Hello	/	/	/	/	/	/	/	/	/
Wind-Shield	OK	/	/	/	/	/	/	/	/
Shoulder Right	OK	OK	/	/	/	/	/	/	/
Shoulder Left	OK	OK	Middle (65%)	/	/	/	/	/	/
Knock	OK	Middle (60%)	OK	OK	/	/	/	/	/
Mix	OK	OK	OK	OK	OK	/	/	/	/
Forward Rotation	OK	OK	OK	OK	OK	OK	/	/	/
Backward Rotation	OK	OK	OK	OK	OK	OK	Middle (65%)	/	/
Carry	OK	OK	OK	OK	OK	OK	OK	OK	/

Table C.2: Table with the gestures of the second groups showing whether the gestures are differentiable (OK) or not (NOT OK) with their accuracy rate in %.

Appendix D

Tests Results

D.1 User Independent Tests

D.1.1 Basic Gestures

Jackknife

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	8	16	32	16	16
Accuracy	65,8%	69,7%	67,6%	**70,2%**	67,2%

Table D.1: Result for Basic Gestures with Jackknife Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	16	32	64	32	32
Accuracy	62,7%	63,8%	62,6%	65%	66%

Table D.2: Result for Basic Gestures with Jackknife Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	10	50	150
Sampling Points	16	32	64	32	32	32
Accuracy	59%	60%	58,9%	67,7%	64,6%	57,5%

Table D.3: Result for Basic Gestures with Jackknife Recognizer and 30 participants.

$PN+

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	50	150	200
Sampling Points	16	32	64	16	16	16
Accuracy	58,5%	57,6%	55%	58%	58,5%	59,4%

Table D.4: Result for Basic Gestures with $PN+ Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	10	50	150
Sampling Points	16	32	16	16	16
Accuracy	58,7%	55,9%	**63,3%**	59,6%	57%

Table D.5: Result for Basic Gestures with $PN+ Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	32	64	128	64	64
Accuracy	54%	55%	54,8%	56%	55%

Table D.6: Result for Basic Gestures with $PN+ Recognizer and 30 participants.

μV

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	128	128
Accuracy	39%	43%	46%	48%	49%	47,9%

Table D.7: Result for Basic Gestures with μV Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100	100	50	150
Sampling Points	16	32	64	128	256	128	128
Accuracy	42,7%	40,9%	44,5%	45%	**49,6%**	45,3%	46%

Table D.8: Result for Basic Gestures with μV Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	128	128
Accuracy	29,5%	33%	37%	38,8%	40,5%	38,6%

Table D.9: Result for Basic Gestures with μV Recognizer and 30 participants.

$P3+ (acceleration)

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	100
Sampling Points	8	16	32	64	128
Accuracy	42,5%	52,8%	47,4%	52%	50,5%

Table D.10: Result for Basic Gestures with $P3+ (acceleration) Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100
Sampling Points	16	32	64	128
Accuracy	51,4%	48,5%	50%	50,8%

Table D.11: Result for Basic Gestures with $P3+ (acceleration) Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100
Sampling Points	128
Accuracy	**57,8%**

Table D.12: Result for Basic Gestures with $P3+ (acceleration) Recognizer and 30 participants.

$P3+ (gyroscope)

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	100
Sampling Points	8	16	32	64	128
Accuracy	43%	50,6%	49%	53,8%	53,5%

Table D.13: Result for Basic Gestures with $P3+ (gyroscope) Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100
Sampling Points	16	32	64	128
Accuracy	49,7%	50,5%	47,9%	49,2%

Table D.14: Result for Basic Gestures with $P3+ (gyroscope) Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100
Sampling Points	128
Accuracy	**56%**

Table D.15: Result for Basic Gestures with $P3+ (gyroscope) Recognizer and 30 participants.

μF

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	100
Sampling Points	8	16	32	64	128
Accuracy	**68,5%**	68,5%	65,3%	67,1%	67,3%

Table D.16: Result for Basic Gestures with μF Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100
Sampling Points	16	32	64	128
Accuracy	65,2%	64,2%	61%	59,5%

Table D.17: Result for Basic Gestures with μF Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	50	150
Sampling Points	32	64	64	64
Accuracy	58%	59,6%	54,5%	57,1%

Table D.18: Result for Basic Gestures with μF Recognizer and 30 participants.

D.1.2 Selected Gestures

Jackknife

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	8	16	32	64	16	16
Accuracy	51%	**62,3%**	59,8%	58,7%	59,8%	59,8%

Table D.19: Result for Selected Gestures with Jackknife Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	32	32
Accuracy	58,3%	61,6%	59,8%	58,1%	60%	60,1%

Table D.20: Result for Selected Gestures with Jackknife Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	100	10	50	150	200
Sampling Points	16	32	64	128	32	32	32	32
Accuracy	55,4%	57,2%	55,3%	54,4%	52,8%	56,3%	57,9%	56,5%

Table D.21: Result for Selected Gestures with Jackknife Recognizer and 30 participants.

$PN+

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	50	150	200
Sampling Points	8	16	32	64	16	16	16
Accuracy	51,5%	54,6%	51,6%	47%	53,7%	55,2%	54,3%

Table D.22: Result for Selected Gestures with $PN+ Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100	10	50	150
Sampling Points	8	16	32	64	16	16	16
Accuracy	48,4%	54,9%	52,7%	52,6%	**60,5%**	55%	56,1%

Table D.23: Result for Selected Gestures with $PN+ Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	32	64	128	64	64
Accuracy	48,6%	49,1%	48,9%	48,3%	49,9%

Table D.24: Result for Selected Gestures with $PN+ Recognizer and 30 participants.

μV

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	100	50	150
Sampling Points	8	16	32	64	128	128	128
Accuracy	32,2%	27,6%	32,8%	37%	40,1%	39%	40,2%

Table D.25: Result for Selected Gestures with μV Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100	100	50	150
Sampling Points	16	32	64	128	256	128	128
Accuracy	31,5%	33%	39,5%	42,4%	**44,5%**	41,3%	42%

Table D.26: Result for Selected Gestures with μV Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	128	128
Accuracy	28,6%	33,6%	35,4%	38,9%	37,8%	37%

Table D.27: Result for Selected Gestures with μV Recognizer and 30 participants.

$P3+ (acceleration)

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	100	50	150
Sampling Points	16	32	64	128	256	128	128
Accuracy	50%	50%	54%	**56,1%**	53%	51,7%	53%

Table D.28: Result for Selected Gestures with $P3+ (acceleration) Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100
Sampling Points	16	32	64	128
Accuracy	50%	52,7%	52,3%	54,4%

Table D.29: Result for Selected Gestures with $P3+ (acceleration) Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100
Sampling Points	128
Accuracy	51%

Table D.30: Result for Selected Gestures with $P3+ (acceleration) Recognizer and 30 participants.

$P3+ (gyroscope)

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	100	50	150
Sampling Points	16	32	64	128	256	128	128
Accuracy	49%	49,1%	52%	**55%**	54,9%	52,2%	54,5%

Table D.31: Result for Selected Gestures with $P3+ (gyroscope) Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100
Sampling Points	16	32	64	128
Accuracy	50%	50%	54,8%	**55,1%**

Table D.32: Result for Selected Gestures with $P3+ (gyroscope) Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100
Sampling Points	128
Accuracy	52,4%

Table D.33: Result for Selected Gestures with $P3+ (gyroscope) Recognizer and 30 participants.

μF

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	100	50	150
Sampling Points	16	32	64	128	256	128	128
Accuracy	56,6%	55,6%	53%	56,8%	55,4%	55%	55%

Table D.34: Result for Selected Gestures with μF Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100	100
Sampling Points	16	32	64	128	256
Accuracy	55%	57,9%	56,6%	**58,4%**	53,6%

Table D.35: Result for Selected Gestures with μF Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100
Sampling Points	128	128
Accuracy	54,5%	52,9%

Table D.36: Result for Selected Gestures with μF Recognizer and 30 participants.

D.1.3 New Dataset

Jackknife

- For 2 participants and so 10 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	16	16
Accuracy	74%	71%	73,6%	**74,5%**

Table D.37: Result for New Dataset with Jackknife Recognizer.

$PN+

- For 2 participants and so 10 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	16	16
Accuracy	**55%**	50,9%	51,7%	52,5%

Table D.38: Result for New Dataset with $PN+ Recognizer.

μV

- For 2 participants and so 10 maximum templates :

Repetition	100
Sampling Points	16
Accuracy	39,4%

Table D.39: Result for New Dataset with μV Recognizer.

$P3+ (acceleration)

- For 2 participants and so 10 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	16	16
Accuracy	**57%**	55,3%	56,9%	**57,4%**

Table D.40: Result for New Dataset with $P3+ (acceleration) Recognizer.

$P3+ (gyroscope)

- For 2 participants and so 10 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	8	16	32	8	8
Accuracy	**60,2%**	54,6%	50%	58,5%	59,9%

Table D.41: Result for New Dataset with $P3+ (gyroscope) Recognizer.

μF

- For 2 participants and so 10 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	8	16	32	16	16
Accuracy	52,8%	58,8%	57,9 %	58,7%	**60%**

Table D.42: Result for New Dataset with μF Recognizer.

D.2 User Dependent Tests

D.2.1 Selected Gestures

Jackknife

- For 10 participants and so 18 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	16	16
Accuracy	78,5%	75%	**79%**	77,6%

Table D.43: Result for Selected Gestures with Jackknife Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	16	16
Accuracy	75,3%	73,9%	73%	77%

Table D.44: Result for Selected Gestures with Jackknife Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	16	16
Accuracy	71%	68,4%	70,6%	69,9%

Table D.45: Result for Selected Gestures with Jackknife Recognizer and 30 participants.

$PN+

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	64	64
Accuracy	79%	86,7%	**87,7%**	87,8%	87,3%	87,2%

Table D.46: Result for Selected Gestures with $PN+ Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	16	32	64	64	64
Accuracy	80%	82,8%	86%	87%	86%

Table D.47: Result for Selected Gestures with $PN+ Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	16	32	64	64	64
Accuracy	75,7%	82,6%	84,8%	83%	84,8%

Table D.48: Result for Selected Gestures with $PN+ Recognizer and 30 participants.

μV

- For 10 participants and so 18 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	32	32
Accuracy	44%	52%	50,7%	53,3%

Table D.49: Result for Selected Gestures with μV Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	32	32
Accuracy	44,8%	53%	**55,4%**	52,8%

Table D.50: Result for Selected Gestures with μV Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	32	32
Accuracy	44,6%	50%	51,8%	50%

Table D.51: Result for Selected Gestures with μV Recognizer and 30 participants.

$P3+ (acceleration)

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	128	128
Accuracy	81,6%	84,3%	87,8%	**89%**	88,6%	**89,2%**

Table D.52: Result for Selected Gestures with $P3+ (acceleration) Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	64	64
Accuracy	79%	85%	88%	87,6%	88%	87,9%

Table D.53: Result for Selected Gestures with $P3+ (acceleration) Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	64	64
Accuracy	76,8%	81,3%	85,7%	86,8%	85,8%	87,3%

Table D.54: Result for Selected Gestures with $P3+ (acceleration) Recognizer and 30 participants.

$P3+ (gyroscope)

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	128	128
Accuracy	80,4%	86,73%	88%	**88,8%**	88,5%	88%

Table D.55: Result for Selected Gestures with $P3+ (gyroscope) Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	64	64
Accuracy	76,7%	85,7%	87,7%	86,6%	85,2%	86,6%

Table D.56: Result for Selected Gestures with $P3+ (gyroscope) Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	128	128
Accuracy	75%	81,2%	84,4%	84,5%	84,6%	85,4%

Table D.57: Result for Selected Gestures with $P3+ (gyroscope) Recognizer and 30 participants.

μF

- For 10 participants and so 18 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	16	32	64	32	32
Accuracy	61,8%	**63,9%**	62,9%	61,6%	63,3%

Table D.58: Result for Selected Gestures with μF Recognizer and 10 participants.

- For 20 participants and so 38 maximum templates :

Repetition	100	100	100	100	50	150	50	150
Sampling Points	16	32	64	128	32	32	64	64
Accuracy	61,37%	62,2%	62,1%	60,3%	61,3%	63%	63%	62,1%

Table D.59: Result for Selected Gestures with μF Recognizer and 20 participants.

- For 30 participants and so 58 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	64	64
Accuracy	56,2%	55,4%	58%	56%	56%	58,4%

Table D.60: Result for Selected Gestures with μF Recognizer and 30 participants.

D.2.2 New Dataset

Jackknife

- For 2 participants and so 10 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	16	16
Accuracy	**85,8%**	83,7%	86,3%	85,3%

Table D.61: Result for New Dataset with Jackknife Recognizer.

$PN+

- For 2 participants and so 10 maximum templates :

Repetition	100	100	100	50	150
Sampling Points	16	32	64	64	64
Accuracy	79,8%	83%	**85%**	83,5%	84,5%

Table D.62: Result for New Dataset with $PN+ Recognizer.

μV

- For 2 participants and so 10 maximum templates :

Repetition	100	100	50	150
Sampling Points	16	32	32	32
Accuracy	51,7%	57,2%	**60,3%**	59,8%

Table D.63: Result for New Dataset with μV Recognizer.

$P3+ (acceleration)

- For 2 participants and so 10 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	32	32
Accuracy	95,6%	**97%**	**97%**	96,9%	96,4%	96,9%

Table D.64: Result for New Dataset with $P3+ (acceleration) Recognizer.

$P3+ (gyroscope)

- For 2 participants and so 10 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	128	128
Accuracy	97,2%	98%	**98,75%**	**98,5%**	**98,75%**	98,5%

Table D.65: Result for New Dataset with $P3+ (gyroscope) Recognizer.

μF

- For 2 participants and so 10 maximum templates :

Repetition	100	100	100	100	50	150
Sampling Points	16	32	64	128	64	64
Accuracy	98%	98,5%	**98,75%**	97,5%	96,5%	97%

Table D.66: Result for New Dataset with μF Recognizer.

Appendix E

Forms

E.0.1 Consent Form

UCLouvain
Louvain Research Institute
in Management and Organizations

LOU RIM

CONSENTEMENT A PARTICIPER A UNE EXPERIENCE SCIENTIFIQUE

Titre de l'étude: Elicitation de gestes
Professeur: Prof. Jean Vanderdonckt, UCLouvain
Expérimentateurs: Chloé Chauvaux & Sarah Steeman
Email de contact: jean.vanderdonckt@uclouvain.be

Nous vous remercions de contribuer à la recherche scientifique en participant à cette expérience. L'expérimentateur vous expliquera le but de l'expérience et les questionnaires à remplir. Posez vos questions éventuelles à l'expérimentateur avant de débuter l'expérience. Vous pouvez obtenir une copie de ce formulaire de consentement et vous y référer à tout moment.

Par la présente, j'accepte de prendre part à l'expérience qui m'a été expliquée. Je peux l'arrêter à tout moment en notifiant l'expérimentateur sans conséquence. Une fois ce consentement signé, un identifiant anonyme sera assigné de sorte que mon nom et mon prénom n'apparaitront nulle part ailleurs que sur ce formulaire. Je consens à ce que les données recueillies dans cette expérience soient traitées à des fins scientifiques, de manière strictement confidentielles et en accord avec la règlementation du respect de la vie privée de l'UCLouvain et du Règlement général de protection des données (RGPD). Toute photo ou vidéo enregistrée lors de l'expérience ne permettra pas d'identifier le participant, notamment en évitant la tête ou en la floutant.

□ Veuillez cocher cette case si vous acceptez d'être éventuellement contacté ultérieurement pour d'autres expériences menées à l'UCLouvain ou pour une interview.
Le cas échéant, adresse e-mail: ……..

Déclaration du Participant:
Je, soussigné ……., atteste que l'expérience susmentionnée m'a été expliquée, que je l'ai comprise, et que j'accepte d'y prendre part. J'ai lu les termes ci-dessus et les formulaires accompagnant et je comprends ce que l'expérience va analyser.

Signature: Date:

Déclaration de l'Expérimentateur:
Je, soussigné ……., confirme avoir expliqué le but de l'expérience, sa procédure et ses questionnaires et, le cas échéant, les risques prévisibles de l'expérience proposée au participant.

Signature: Date:

Identifiant assigné au participant : ……..

E.0.2 Questionnaire

UCLouvain
Louvain Research Institute
in Management and Organizations

LOU RIM

QUESTIONNAIRE POUR PARTICIPER A L'EXPERIENCE

Titre de l'étude: Elicitation de gestes
Professeur: Prof. Jean Vanderdonckt, UCLouvain
Expérimentateurs: Chloé Chauvaux & Sarah Steeman
Email de contact: jean.vanderdonckt@uclouvain.be

Données personnelles (qui seront anonymisées)
Code anonyme: …… (à attribuer par l'expérimentateur)
Prénom : …….. (anonymisés dans la base de données)
Nom de famille : ……..
Genre : [ ] Femme [ ] Homme
Age : …… (années)
Profession : [ ] Etudiant [ ] Cadre [ ] Employé
[ ] Indépendant [ ] Pensionné [ ] Chômeur
[ ] Autre : …… (préciser s.v.p.)
Domaine d'activité: …… (p. ex., administration, enseignement)
Etudes : …… (domaine de votre dernier diplôme)

J'utilise un ordinateur fréquemment:	Pas d'accord	1	2	3	4	5	6	7	D'accord
J'utilise un smartphone fréquemment:	Pas d'accord	1	2	3	4	5	6	7	D'accord
J'utilise une tablette fréquemment:	Pas d'accord	1	2	3	4	5	6	7	D'accord
J'utilise une console de jeux fréque[t].:	Pas d'accord	1	2	3	4	5	6	7	D'accord
J'utilise une Kinect fréquemment :	Pas d'accord	1	2	3	4	5	6	7	D'accord

---

Données expérimentales

Connaissez-vous l'appareil utilisé dans l'expérience [ ] Oui [ ] Non
Combien de fois l'utilisez-vous? Rarement 1 2 3 4 5 6 7 Fréquemment

E.0.3 Post-study system usability questionnaire

UCLouvain
Louvain Research Institute
in Management and Organizations

LOU RIM

Questionnaire Post-test sur l'utilisabilité (PSSUQ)

Titre de l'étude: Elicitation de gestes
Professeur: Prof. Jean Vanderdonckt, UCLouvain
Expérimentateurs: Chloé Chauvaux & Sarah Steeman
Identifiant :

Ce questionnaire a pour but d'exprimer votre satisfaction relative à l'utilisabilité du système que vous avez testé. Vos réponses nous aideront à comprendre quels aspects du système vous avez appréciés ou non. Dans la mesure du possible, pensez aux tâches ou actions que vous avez réalisées avec le système lorsque vous répondrez aux questions. Veuillez lire chaque énoncé et indiquer dans quelle mesure vous êtes totalement d'accord ou en désaccord en suivant une échelle de réponse à 7 positions (1=totalement en désaccord, 7=totalement d'accord). Merci beaucoup à vous!

		Totalement en désaccord						Totalement d'accord	Non approprié
ID	**Enoncé**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	
1	Globalement, je suis satisfait de la facilité d'utilisation de ce système	○	○	○	○	○	○	○	
2	Le système était simple à utiliser	○	○	○	○	○	○	○	
3	J'ai pu réaliser les taches et les scenarios rapidement avec ce système	○	○	○	○	○	○	○	
4	Je me suis senti à l'aise avec ce système	○	○	○	○	○	○	○	
5	Le système était facile à apprendre	○	○	○	○	○	○	○	
6	Je pense devenir rapidement productif en utilisant ce système	○	○	○	○	○	○	○	
7	Le système m'a fourni des messages d'erreur clairs pour résoudre les problèmes	○	○	○	○	○	○	○	
8	J'ai pu corriger chaque erreur simplement et rapidement	○	○	○	○	○	○	○	
9	L'information fournie par le système, telle que l'aide en ligne, les messages, la documentation, était clairement fournie par le système	○	○	○	○	○	○	○	
10	J'ai trouvé facilement l'information dont j'avais besoin	○	○	○	○	○	○	○	
11	L'information m'a été utile pour réaliser les tâches et scénarios	○	○	○	○	○	○	○	
12	L'information était clairement présentée à l'écran	○	○	○	○	○	○	○	
13	L'interface du système était agréable	○	○	○	○	○	○	○	
14	J'ai aimé utiliser l'interface du système	○	○	○	○	○	○	○	
15	Le système disposait de toutes les fonctions que je souhaitais	○	○	○	○	○	○	○	
16	Globalement, je suis satisfait du système	○	○	○	○	○	○	○	

LOUVAIN-LA-NEUVE | BRUXELLES | MONS | TOURNAI | CHARLEROI | NAMUR
Place des Doyens, 1 bte L2.01.02, 1348 Louvain-la-Neuve, Belgique
Tél. +32 (0)10 47 85 25 – jean.vanderdonckt@uclouvain.be – www.uclouvain.be/jean.vanderdonckt

Bibliography

[1] Control digital devices using various gestures, https://kinemic.com/en/kinemic-band/gestures/, Oct 2019.

[2] Roland Aigner, Daniel Wigdor, Hrvoje Benko, Michael Haller, David Lindbauer, Alexandra Ion, Shengdong Zhao, and JTKV Koh. Understanding mid-air hand gestures: A study of human preferences in usage of gesture types for hci. *Microsoft Research TechReport MSR-TR-2012-111*, 2:30, 2012.

[3] Christoph Amma, Marcus Georgi, Tomt Lenz, and Fabian Winnen. Kinemic wave: A mobile freehand gesture and text-entry system. In Jofish Kaye, Allison Druin, Cliff Lampe, Dan Morris, and Juan Pablo Hourcade, editors, *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016, Extended Abstracts*, pages 3639–3642. ACM, 2016.

[4] F. M. Caputo, S. Burato, G. Pavan, T. Voillemin, H. Wannous, J. P. Vandeborre, M. Maghoumi, E. M. Taranta II, A. Razmjoo, J. J. LaViola Jr., F. Manganaro, S. Pini, G. Borghi, R. Vezzani, R. Cucchiara, H. Nguyen, M. T. Tran, and A. Giachetti. Online Gesture Recognition. In Silvia Biasotti, Guillaume Lavoué, and Remco Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2019.

[5] Yu-Chun Chen, Chia-Ying Liao, Shuo-Wen Hsu, Da-Yuan Huang, and Bing-Yu Chen. Exploring user defined gestures for ear-based interactions. *Proc. ACM Hum. Comput. Interact.*, 4(ISS):186:1–186:20, 2020.

[6] Victor Cheung, Alex Keith Eady, and Audrey Girouard. Exploring eyes-free interaction with wrist-worn deformable materials. In *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '17, page 521–528, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Andrew Crossan, John Williamson, Stephen Brewster, and Rod Murray-Smith. Wrist rotation for interaction in mobile contexts. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, page 435–438, New York, NY, USA, 2008. Association for Computing Machinery.

[8] William Delamare, Chaklam Silpasuwanchai, Sayan Sarcar, Toshiaki Shiraki, and Xiangshi Ren. On gesture combination: An exploration of a solution to augment gesture interaction. In Bongshin Lee, Geehyuk Lee, Stacey Scott, Melanie Tory, and Jeonghyun Kim, editors, *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces, ISS 2019, Daejeon, South Korea, November 10-13, 2019*, pages 135–146. ACM, 2019.

[9] Adrian Fernandez and Dung Dang. Chapter 8 - analog: The infinite shades of gray. In Adrian Fernandez and Dung Dang, editors, *Getting Started with the MSP430 Launchpad*, pages 81–125. Newnes, Oxford, 2013.

[10] Jutta Fortmann, Erika Root, Susanne Boll, and Wilko Heuten. Tangible apps bracelet: Designing modular wrist-worn digital jewellery for multiple purposes. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, DIS '16, page 841–852, New York, NY, USA, 2016. Association for Computing Machinery.

[11] Bogdan-Florin Gheran, Radu-Daniel Vatavu, and Jean Vanderdonckt. Ring x2: Designing gestures for smart rings using temporal calculus. In Ilpo Koskinen, Youn-Kyung Lim, Teresa Cerratto Pargman, Kenny K. N. Chow, and William Odom, editors, *Companion Publication of the 19th International ACM SIGACCESS Conference on Computers and Accessibility, DIS 2018, Hong Kong, China, June 09-13, 2018*, pages 117–122. ACM, 2018.

[12] Nicholas Gillian and Joseph A Paradiso. The gesture recognition toolkit. *The Journal of Machine Learning Research*, 15(1):3483–3487, 2014.

[13] Benedikt Gollan, Michael Haslgrübler, Alois Ferscha, and Josef Heftberger. Making sense: Experiences with multi-sensor fusion in industrial assistance systems. In Juan-Manuel Belda-Lois, Chen Wang, Manuel Domínguez-Morales, Alan Pope, and Hugo Plácido da Silva, editors, *Proceedings of the 5th International Conference on Physiological Computing Systems, PhyCS 2018, Seville, Spain, September 19-21, 2018*, pages 64–74. SciTePress, 2018.

[14] Yves Guiard. Asymmetric division of labor in human skilled bimanual action. *Journal of Motor Behavior*, 19(4):486–517, 1987. PMID: 15136274.

[15] F. Hamm. Frame semantics. In *The Cambridge Encyclopedia of the Language Sciences*. Cambridge University Press, 2009.

[16] Jean-Michel Hoc. From human – machine interaction to human – machine cooperation. *Ergonomics*, 43(7):833–843, 2000. PMID: 10929820.

[17] Jinmiao Huang, Prakhar Jaiswal, and Rahul Rai. Gesture-based system for next generation natural and intuitive interfaces. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 33(1):54–68, 2019.

[18] Eugene M. Taranta II, Corey R. Pittman, Mehran Maghoumi, Mykola Maslych, Yasmine M. Moolenaar, and Joseph J. LaViola Jr. Machete: Easy, efficient, and precise continuous custom gesture segmentation. *ACM Trans. Comput. Hum. Interact.*, 28(1):5:1–5:46, 2021.

[19] Eugene M. Taranta II, Amirreza Samiei, Mehran Maghoumi, Pooya Khaloo, Corey R. Pittman, and Joseph J. LaViola Jr. Jackknife: A reliable recognizer with few samples and many modalities. In Gloria Mark, Susan R. Fussell, Cliff Lampe, m. c. schraefel, Juan Pablo Hourcade, Caroline Appert, and Daniel Wigdor, editors, *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017*, pages 5850–5861. ACM, 2017.

[20] ISO. ISO/IEC 9241, Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts. standard, International Standard Organization, Geneva, 2018.

[21] ISO. ISO/IEC 25010 - Software Quality Product Standard. standard, International Standard Organization, Geneva, 2019.

[22] Hessam Jahani and Manolya Kavakli. Exploring a user-defined gesture vocabulary for descriptive mid-air interactions. *Cogn. Technol. Work*, 20(1):11–22, feb 2018.

[23] Andrés G Jaramillo and Marco E Benalcázar. Real-time hand gesture recognition with emg using machine learning. In *2017 IEEE second ecuador technical chapters meeting (ETCM)*, pages 1–5. IEEE, 2017.

[24] CONTINUUM KENDON'S. Gesture: A psycholinguistic approach1.

[25] Jungsoo Kim, Jiasheng He, Kent Lyons, and Thad Starner. The gesture watch: A wireless contact-free gesture based wrist interface. In *11th IEEE International Symposium on Wearable Computers (ISWC 2007), October 11-13, 2007, Boston, MA, USA*, pages 15–22. IEEE Computer Society, 2007.

[26] Ismail Kirbas. Developing a signal similarity analyse software for accelerometer sensor data. *J. Int. Sci. Publ*, 13:172–179, 2019.

[27] Kanak Kumar, Aanchal Sharma, and Suman Lata Tripathi. Sensors and their application. In *Electronic Devices, Circuits, and Systems for Biomedical Applications*, pages 177–195. Elsevier, 2021.

[28] Joseph J. LaViola. 3d gestural interaction: The state of the field. *International Scholarly Research Notices*, 2013, 2013.

[29] Nathan Magrofuoco. *Multi-context template-based gesture recognition for fast prototyping*. PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2021.

[30] Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. Two-dimensional stroke gesture recognition: A survey. *ACM Computing Surveys*, 54(7), jul 2021.

[31] Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. μv: An articulation, rotation, scaling, and translation invariant (arst) multi-stroke gesture recognize. *Proc. ACM Hum.-Comput. Interact.*, 6(EICS), may 2022.

[32] Meethu Malu, Pramod Chundury, and Leah Findlater. Exploring accessible smartwatch interactions for people with upper body motor impairments. In Regan L. Mandryk, Mark Hancock, Mark Perry, and Anna L. Cox, editors, *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, page 488. ACM, 2018.

[33] David McNeill. *Language and gesture*, volume 2. Cambridge University Press Cambridge, 2000.

[34] Meredith Ringel Morris, Anqi Huang, Andreas Paepcke, and Terry Winograd. Cooperative gestures: multi-user gestural interactions for co-located groupware. In Rebecca E. Grinter, Tom Rodden, Paul M. Aoki, Edward Cutrell, Robin Jeffries, and Gary M. Olson, editors, *Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*, pages 1201–1210. ACM, 2006.

[35] Miguel A. Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 1099–1108, New York, NY, USA, 2013. Association for Computing Machinery.

[36] Mathieu Nancel, Julie Wagner, Emmanuel Pietriga, Olivier Chapuis, and Wendy E. Mackay. Mid-air pan-and-zoom on wall-sized displays. In Desney S. Tan, Saleema Amershi, Bo Begole, Wendy A. Kellogg, and Manas Tungare, editors, *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 177–186. ACM, 2011.

[37] Thamer Horbylon Nascimento, Cristiane B. R. Ferreira, Wellington Galvão Rodrigues, and Fabrízzio Alphonsus A. M. N. Soares. Interaction with smartwatches using gesture recognition: A systematic literature review. In *44th IEEE Annual Computers, Software, and Applications Conference, COMPSAC 2020, Madrid, Spain, July 13-17, 2020*, pages 1661–1666. IEEE, 2020.

[38] Michael Nielsen, Moritz Stoerring, Thomas Moeslund, and Erik Granum. A procedure for developing intuitive and ergonomic gesture interfaces for hci. pages 409–420, 01 2003.

[39] Jiazhi Ou, Xilin Chen, and Jie Yang. Gesture recognition for remote collaborative physical tasks using tablet pcs. In *Proc. of IX IEEE Intl. Conf. on Computer Vision, Work. on Multimedia Technologies in E-Learning and Collaboration (Nice, 2003)*. Citeseer, 2003.

[40] Mehdi Ousmer, Arthur Sluÿters, Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. Recognizing 3d trajectories as 2d multi-stroke gestures. *Proc. ACM Hum. Comput. Interact.*, 4(ISS):198:1–198:21, 2020.

[41] Thammathip Piumsomboon, Adrian J. Clark, Mark Billinghurst, and Andy Cockburn. User-defined gestures for augmented reality. In Paula Kotzé, Gary Marsden, Gitte Lindgaard, Janet Wesson, and Marco Winckler, editors, *Human-Computer Interaction - INTERACT 2013 - 14th IFIP TC 13 International Conference, Cape Town, South Africa, September 2-6, 2013, Proceedings, Part II*, volume 8118 of *Lecture Notes in Computer Science*, pages 282–299. Springer, 2013.

[42] Dan Saffer. *Designing gestural interfaces - touchscreens and interactive devices*. O'Reilly, 2008.

[43] Ugo Braga Sangiorgi, François Beuvens, and Jean Vanderdonckt. User interface design by collaborative sketching. In *Proceedings of the Designing Interactive Systems Conference*, DIS '12, page 378–387, New York, NY, USA, 2012. Association for Computing Machinery.

[44] Kadek Ananta Satriadi, Barrett Ens, Maxime Cordeil, Bernhard Jenny, Tobias Czauderna, and Wesley Willett. Augmented reality map navigation with freehand gestures. In *IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019, Osaka, Japan, March 23-27, 2019*, pages 593–603. IEEE, 2019.

[45] Shaikh Shawon Arefin Shimon, Courtney Lutton, Zichun Xu, Sarah Morrison-Smith, Christina Boucher, and Jaime Ruiz. Exploring non-touchscreen gestures for smartwatches. In Jofish Kaye, Allison Druin, Cliff Lampe, Dan Morris, and Juan Pablo Hourcade, editors, *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016*, pages 3822–3833. ACM, 2016.

[46] Arthur Sluÿters, Sébastien Lambot, and Jean Vanderdonckt. Hand gesture recognition for an off-the-shelf radar by electromagnetic modeling and inversion. In *27th International Conference on Intelligent User Interfaces*, IUI '22, page 506–522, New York, NY, USA, 2022. Association for Computing Machinery.

[47] Arthur Sluÿters, Mehdi Ousmer, Paolo Roselli, and Jean Vanderdonckt. Quantumleap, a framework for engineering gestural user interfaces based on the leap motion controller. *Proc. ACM Hum. Comput. Interact.*, 6(EICS):1–47, 2022.

[48] Radu-Daniel Vatavu. Improving gesture recognition accuracy on touch screens for users with low vision. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 4667–4679, New York, NY, USA, 2017. Association for Computing Machinery.

[49] Tran Huy Vu, Archan Misra, Quentin Roy, Kenny Choo Tsu Wei, and Youngki Lee. Smartwatch-based early gesture detection 8 trajectory tracking for interactive gesture-driven applications. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(1), mar 2018.

[50] Yuki Yamato, Yutaro Suzuki, Kodai Sekimori, Buntarou Shizuki, and Shin Takahashi. *Hand Gesture Interaction with a Low-Resolution Infrared Image Sensor on an Inner Wrist*. Association for Computing Machinery, New York, NY, USA, 2020.

[51] Hui-Shyong Yeo, Juyoung Lee, Hyung-il Kim, Aakar Gupta, Andrea Bianchi, Daniel Vogel, Hideki Koike, Woontack Woo, and Aaron Quigley. Wrist: Watch-ring interaction and sensing technique for wrist gestures and macro-micro pointing. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '19, New York, NY, USA, 2019. Association for Computing Machinery.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**