

Louvain Research Institute
in Management and Organizations



A Systematic Procedure for Comparative Testing of 3D Hand Gesture Template-Based Recognizers in Multiple Contexts of Use

by

Mehdi OUSMER

Thesis submitted in fulfillment
of the requirements for the degree of

Doctor in Engineering and Technology

of the Université catholique de Louvain
Ecole polytechnique de Louvain

Commitee in charge:

Prof. Jean Vanderdonckt (Co-advisor)	Université catholique de Louvain, Belgium
Prof. Paolo Roselli (Co-advisor)	University of Rome "Tor Vergata", Italy
Prof. Rachid Gherbi	Université de Paris-Saclay, France
Prof. Suzanne Kieffer (Secretary)	Université catholique de Louvain, Belgium
Prof. Giovanni Saggio	University of Rome "Tor Vergata", Italy
Prof. Charles Pecheur (President)	Université catholique de Louvain, Belgium

"It is not because things are difficult that we do not dare, it is because we do not dare that they are difficult."

—Seneca, Epistuale Morales

Abstract

The advent of embedded and wearable sensors and devices with precise tracking capabilities has paved the way for 3D gesture recognition. Designing usable gesture-based user interfaces is already challenging to address both gesture usability and recognition. This problem is even more challenging when such user interfaces need to be deployed in multiple contexts of use, i.e., for different users, devices, and environments. While the conceptualization of gesture recognition systems for multiple contexts of use, primarily multiple devices, is possible, their technological implementation is limited to a few devices, that are hard to transpose from one device to another. This limitation occurs because it is currently impossible to develop it for all types of devices due to their variety. To address these challenges, this thesis defines, and justifies a systematic procedure for comparative testing of 3D hand gesture recognition in multiple contexts of use. This procedure leads to a method for comparative testing of multi-device gesture recognition that is applied to two use cases: one use case with unipath dynamic gestures and one with multipath dynamic gestures. The applicability of this procedure is supported by the development of a software tool that extends a gesture recognition framework that demonstrates the potential of the comparative testing method and provides valuable insights for designers seeking a reliable and efficient recognizer to meet their needs.

Keywords: Human-Computer Interaction, Gesture recognition, 3D hand gestures, Comparative testing, User interface development, Template-based matching

Acknowledgements

First of all, I would like to express my sincere gratitude to my thesis supervisor, Prof. Jean Vanderdonckt, for providing me with the opportunity to complete this thesis. He has always supported me throughout my Ph.D. and related research. His patience, motivation, and extensive knowledge have been incredibly valuable. In addition, his advice helped me throughout the research and writing of this thesis.

I would like to express my gratitude to my co-supervisor, Professor Paolo Roselli, for his invaluable collaboration and assistance in navigating the complex knowledge of geometric algebra and its application in gesture recognition. His guidance and encouragement have played a pivotal role in shaping the direction of my research.

I would like to express my gratitude to the members of my Ph.D. committee for reviewing the manuscript and providing valuable comments. To Prof. Rachid Gherbi, whom I met during my master's degree in Algeria and who was a source of inspiration for his dedication to transmitting his knowledge about the VR/AR field by giving constructive advice. I would like to thank Prof. Suzanne Kieffer and Prof. Giovanni Saggio for their expertise and contribution to the quality of this thesis. I would also like to thank Prof. Charles Pecheur for being the president of my thesis committee.

Special thanks to Prof. Radu Vatavu, from Stefan cel Mare University of Suceava, who shared his deep knowledge in gesture interaction [121] with me at different stages of my research.

In addition, I would like to thank my colleagues at the Louvain Research Institute in Management and Organizations (LouRIM) Institute and the assistants and researchers for their fruitful discussions, comments, and support throughout this research and teaching work. I would also like to thank the administrative team for their invaluable advice and excellent work in facilitating our tasks and creating a favorable working environment.

Finally, I am deeply grateful to my family and friends for their unconditional love, understanding, and support throughout this Ph.D. journey. I want to express my profound gratitude for their constant source of motivation, strength, and invaluable advice.

Publications

- Mehdi Ousmer, Jean Vanderdonckt, and Sabin Buraga. "An Ontology for Reasoning on Body-Based Gestures". In: *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '19. Valencia, Spain: Association for Computing Machinery, 2019. ISBN: 9781450367455. DOI: 10.1145/3319499.3328238. URL: https://doi.org/10.1145/3319499.3328238.
- Mehdi Ousmer, Arthur Sluÿters, Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. "Recognizing 3D Trajectories as 2D Multi-Stroke Gestures". In: *Proceedings of the ACM on Human-Computer Interaction* 4.ISS (Nov. 2020), pp. 1–21. DOI: 10.1145/3427326. URL: https://doi.org/10.1145/3427326.
- Djamel Eddine Benrachou, Mostefa Masmoudi, Oualid Djekoune, Nadia Zenati, and Mehdi Ousmer. "Avatar-Facilitated Therapy and Virtual Reality: Next-Generation of Functional Rehabilitation Methods". In: *2020 1st International Conference on Communications, Control Systems and Signal Processing (CCSSP)*. El Oued, Algeria: IEEE, 2020, pp. 298–304. DOI: 10.1109/CCSSP49278.2020.9151528.
- Arthur Sluÿters, Mehdi Ousmer, Paolo Roselli, and Jean Vanderdonckt. "QuantumLeap, a Framework for Engineering Gestural User Interfaces based on the Leap Motion Controller". In: *Proceedings of the ACM on Human-Computer Interaction* 6.EICS (June 2022), 161:1–161:47. DOI: 10.1145/3532211. URL: https://doi.org/10.1145/3532211.
- Mehdi Ousmer, Arthur Sluÿters, Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. "A Systematic Procedure for Comparing Template-Based Gesture Recognizers". In: *HCI International 2022 - Late Breaking Papers. Multimodality in Advanced Interaction Environments*. Ed. by Masaaki Kurosu, Sakae Yamamoto, Hirohiko Mori, Dylan D. Schmorrow, Cali M. Fidopiastis, Norbert A. Streitz, et al. Cham: Springer Nature Switzerland, 2022, pp. 160–179. ISBN: 978-3-031-17618-0. DOI: 10.1007/978-3-031-17618-0_13. URL: https://doi.org/10.1007/978-3-031-17618-0_13.

Contents

Abstract iv

Acknowledgements v

Publications vii

List of Figures xiii

List of Tables xvii

1 Introduction 1
- 1.1 Context . . . 1
- 1.2 Research Question . . . 3
- 1.3 Working Hypotheses . . . 5
- 1.4 List of Contributions . . . 7
- 1.5 Manuscript Overview . . . 8

2 Related Work 11
- 2.1 Gesture . . . 11
  - 2.1.1 Stroke Gesture . . . 12
  - 2.1.2 3D Gesture . . . 12
  - 2.1.3 Gesture Taxonomies and Classifications . . . 13
  - 2.1.4 Gesture Segmentation . . . 16
  - 2.1.5 Gesture Representation and Notation . . . 17
  - 2.1.6 Motion Sensors . . . 17
- 2.2 Related Work . . . 18
  - 2.2.1 Gesture Recognition . . . 18
  - 2.2.2 2D Stroke Gesture Recognizers . . . 19
  - 2.2.3 3D Gesture Recognizers . . . 21
- 2.3 A Design Space for Hand Gesture Recognizers . . . 22
  - 2.3.1 Rapid Prototyping Features . . . 25
- 2.4 Descriptive Analysis . . . 25
  - 2.4.1 uWave (2009) . . . 26
  - 2.4.2 Rubine3D (2009) . . . 27
  - 2.4.3 $3 (2010) . . . 29

2.4.4 Hoffman's Linear Classifier (2010) . . . 32
2.4.5 Protractor3D (2011) . . . 33
2.4.6 $P3D (2016) . . . 35
2.4.7 3 Cent (2017) . . . 37
2.4.8 JackKnife (2017) . . . 39
2.4.9 DeepGRU (2019) . . . 41
2.5 Comparative Analysis . . . 43
2.6 Generative Analysis . . . 45
2.6.1 Classification Approaches . . . 46
2.6.2 Invariance Properties . . . 46
2.6.3 Data Type . . . 47
2.6.4 Gesture Type . . . 47
2.6.5 Gesture Nature . . . 48
2.6.6 Device Type . . . 48
2.7 Motivations for Multi-context gesture recognizers . . . 49

3 Concepts for Comparative Testing of 3D Hand Gestures in Multiple Contexts of Use 51
3.1 Terminology . . . 51
3.2 Definitions . . . 52
3.3 Temporal Relationships in the "Cross" Configuration . . . 56
3.4 Procedure for Single Gesture Recognition Testing . . . 57
3.5 Workflow . . . 59
3.5.1 Stage1: Gesture Elicitation Study . . . 59
3.5.2 Stage2: Data Acquisition . . . 60
3.5.3 Stage3: Training . . . 62
3.5.4 Stage4: Testing . . . 63
3.6 Comparative Testing . . . 64
3.6.1 Artefacts for Comparative Testing . . . 65
3.6.2 Recognizers . . . 65
3.6.3 Techniques . . . 65
3.6.4 Scenarios . . . 66
3.6.5 Datasets . . . 66
3.6.6 Parameters . . . 67
3.6.7 Procedure for Systematic Comparative Testing . . . 67
3.7 Conclusion . . . 67

4 Use Case 1: Comparative Testing on Unipath Dynamic Gestures 69
4.1 Recognizers . . . 71
4.1.1 Rubine3D, a 3D Extension of Rubine . . . 71
4.1.2 Rubine-Sheng, Another 3D Extension of Rubine . . . 72
4.1.3 $\$P^3$, a 3D Extension of $\$P$ . . . 73

4.1.4 $\$F$, a Flexible Variant of $\$P^3$ . . . 74
4.1.5 FreeHandUni, a 3D Extension of $P . . . 74
4.1.6 $\$P+^3$, a 3D Extension of $\$P+$ . . . 75
4.1.7 $\$Q^3$, a 3D Extension of $\$Q$ . . . 75
4.2 Datasets . . . 76
4.2.1 SHREC2019 . . . 76
4.2.2 3DTCGS . . . 76
4.2.3 MadLabSD . . . 76
4.3 Design . . . 78
4.3.1 Experiment . . . 78
4.3.2 Procedures and Quantitative Measures . . . 79
4.4 Results . . . 80
4.4.1 User-Independent Scenario . . . 80
4.4.2 User-Dependent Scenario . . . 84
4.4.3 Impact on Recognition Rate and Execution Time . . . 90
4.5 Conclusion . . . 93

5 Use Case 2: Comparative Testing on Multipath Dynamic Gestures 97
5.1 Recognizers . . . 98
5.1.1 P^3+X Recognizer . . . 98
5.1.2 PennyPincher3D Recognizer . . . 98
5.2 Datasets . . . 98
5.2.1 SHREC2019 . . . 98
5.2.2 Jackknife-LM . . . 99
5.3 Design . . . 99
5.3.1 Experiment . . . 100
5.3.2 Procedure and Quantitative Measures . . . 101
5.4 Results . . . 102
5.4.1 SHREC2019 Dataset . . . 102
5.4.2 Jackknife-LM Dataset . . . 112
5.5 Conclusion . . . 114

6 QUANTUMLEAP Extension: Testing Module 117
6.1 Evaluation Procedure . . . 118
6.1.1 User-dependent Scenario . . . 118
6.1.2 User-independent Scenario . . . 118
6.2 Testing Module Implementation . . . 119
6.2.1 Configuration . . . 120
6.2.2 Recognizers . . . 120
6.2.3 Testing . . . 121
6.2.4 Datasets . . . 121
6.2.5 Results . . . 122

6.3 Conclusion 123

7 Conclusion **125**
7.1 Advantages and Limitations 127
7.2 Future Work 128

List of Figures

1.1 The AMMPT Conceptual framework applied for the gesture recognition [6]. 4
1.2 The categorization of our work within the diverse classes of evaluation methods. [137] 7

2.1 A diagram that summarizes the various phases of gesture recognition in an interaction system. 18
2.2 A design space for hand gesture recognizer. 22
2.3 The space design instantiation for uWave. 28
2.4 The space design instantiation for Rubine3D. 30
2.5 The space design instantiation for $3. 32
2.6 The space design instantiation for Hoffman's linear classifier. . . 34
2.7 The space design instantiation for Protractor3D. 36
2.8 The space design instantiation for $P3D. 38
2.9 The space design instantiation for 3 Cent. 40
2.10 The space design instantiation for JackKnife. 42
2.11 The space design instantiation for DeepGRU. 44
2.12 The space design instantiation for all the selected recognizers. . 45

3.1 Taxonomy of configurations for input devices used in the gesture recognition process. 55
3.2 Taxonomy of input devices from [20] extended. 56
3.3 A subset of the SPEM notation relevant to this work. 58
3.4 Gesture recognition testing procedure workflow. 59
3.5 Activity diagram for Gesture Elicitation Study. 61
3.6 Activity diagram for Data acquisition. 62
3.7 Activity diagram for Training. 63
3.8 Activity diagram for Testing. 64

4.1 Configuration of Input Devices in Use Cases. 70
4.2 The SHREC2019 dataset [24]. 76
4.3 The 3DTCGS dataset [23]. 77
4.4 The four domains of the MadLabSD dataset. [49] 78

4.5 Recognition rates of all recognizers aggregated for all datasets in the user-independent scenario: individual rates (top) for N=4, ..., 64 (top) and global rates (bottom) $\forall N$. Error bars show their respective standard deviation. 82
4.6 Execution times of all recognizers for two datasets (left: SHREC2019 [24], right: 3DTCGS [23]): individual times for N=4, ..., 64. 85
4.7 Recognition times of all recognizers for two datasets from MadLabSD (left: Domain 1, right: Domain 2 [49]): individual times for N=4..64 for the User-dependent scenario. 88
4.8 Recognition times of all recognizers for two datasets from MadLabSD (left: Domain 3, right: Domain 4 [49]): individual times for N=4..64 for the User-dependent scenario. 89
4.9 Difference of percentage in recognition rate loss for all recognizers on the MadLabSD dataset [49], all domains, when restricting the scenario from user-dependent to user-independent. Recognizers are sorted in decreasing order of their global recognition rate. 91
4.10 The $-like recognizers instantiation. 94
4.11 The Rubine3D and Rubine-Sheng instantiation. 95

5.1 The SHREC2019 gesture classes and samples. [24] 99
5.2 The Jackknife-LM gesture classes. [116] 100
5.3 Overview of the LMC Hand model and joints. 101
5.4 Recognition rates of all recognizers for the SHREC2019 for all conditions. Error bars show a confidence interval of 95%. 103
5.5 Recognition rates of all recognizers for the SHREC2019 dataset [24], the plot shows average rates by joint $A = \{1\ (P), ..., 6\ (P + AF)\}$. 105
5.6 Ranking of the best individual recognition rates above 80% by the number of joints (A) and the number of points (N) for the SHREC2019 dataset. 106
5.7 Recognition rate (a) and the number of recognized gestures per class over 100 trials (b) of all recognizers for the user-independent scenario, for the optimal conditions defined by the de Borda ranking: A=2 $(T + I)$, N=32, T=16. Error bars show a confidence interval of 95%. 108
5.8 Average execution times by recognizer. Error bars show a confidence interval of 95%. 110
5.9 Average execution times by recognizer per number of joints (A). 111
5.10 Execution times of all recognizers for the Jackknife-LM for the user-independent scenario. Error bars show a confidence interval of 95%. 113

5.11 Recognition rates of all recognizers for the Jackknife-LM dataset [116], the plot shows average rates by joint $A = \{1\,(P), ..., 6\,(P + AF)\}$. 114
5.12 The $-like recognizers instantiation. 115
5.13 The JackKnife and PennyPincher3D instantiation. 116

6.1 The UI configuration page. 120
6.2 The structure of a result file in user-dependent evaluation. . . . 123

7.1 ZeroG testing tool mockups UI. 130

List of Tables

2.1 Rapid prototyping properties of the 3D recognizers 26

3.1 Allen's interval algebra relations in "cross-device" configuration. 57

4.1 Description of selected datasets. 79
4.2 Summary of recognition rates and execution times for all datasets and per individual dataset. 81
4.3 ANOVAs computed for the 7 RECOGNIZERS in the user-independent scenario: G1=group 1, G2=group 2. 83
4.4 ANOVAs computed for the 7 RECOGNIZERS in the user-dependent scenario: G1=group 1, G2=group 2. 86
4.5 The effect of dataset on the recognition rate and the execution time with the effect size 92
4.6 The effect of number of templates and points on the recognition rate and the execution time with the effect size 93

5.1 Global position of the recognizers for each condition and overall, according to de Borda method across all recognizers (the higher, the better). 107

Chapter 1

Introduction

1.1 Context

The era of ubiquitous computing [135] has promoted the use of computer-based systems anywhere, at any time, by anybody, in any type of business where the user interface is predominant [112]. The development of its underlying technology has enhanced the human ability to process information by developing various sensors to capture a large amount of data present in the environment. It also enabled the development of traditional means of communication between humans, such as verbal and non-verbal communication [88], which is one of the natural and efficient means of communication in human-to-human interaction. This has been achieved by incorporating new gestures resulting from the increasing accessibility [15] of affordable mobile and embedded devices [65].

Furthermore, the development of new work tools using computer-based systems and robots introduced new methods of communication between humans and machines. This is why researchers in the field of Human-Computer Interaction (HCI) are committed to designing systems that should be appropriate and usable for users and that ensure smooth and natural interaction while being adapted to the context of use as much as possible [37]. The most common user interface paradigm remains the Graphical User Interface (GUI), which has slightly evolved, especially the WIMP (Windows, Icons, Menus, Pointer) interaction style, which mainly uses the keyboard and mouse. This interaction style has evolved to Post-WIMP, in particular through the paradigm of Speech, Images, language, Knowledge (SILK) paradigm [34]. However, many newly developed devices are equipped with sensors, enabling the development of many interaction systems that rely on gestures and non-verbal communication for interaction. These systems give users the freedom to produce a large number of gestures with different parts of their body to manipulate Gesture-Based User Interfaces (GBUIs) [121].

In some contexts of use, it is important to consider that users may have different preferences for producing gestures [63, 100, 119, 139, 142]. The context of use strongly influences the range of gestures that users can produce. It is defined as a triplet formed by **(**1**) the end user(s) and his (their) tasks**, **(**2**) the hardware and software platform used**, and **(**3**) the physical environment in which the interaction takes place** [21]. This situation involves using various body parts and performing gestures with different amplitudes within the user gesture space. It requires the utilization of one or more motion devices to accurately capture the specific details of a movement. This is in line with the observation that many works have particular configurations that are specific to the context of use. The literature contains a significant body of knowledge [65, 121] regarding gesture recognizers, gesture sets, devices, number of users, and other specific configuration variables due to the variability of contexts of use such as in the 2D $Family recognizers (*e.g.*, $Q for low-resources devices [125] and $P+ for users with low vision [122]).

Designing intuitive and usable GBUIs can be challenging, especially for multiple contexts of use. One particular challenge that encapsulates this difficulty is the issue of gesture recognition. Designers and developers often struggle to create a recognizer module that can accurately identify gestures performed by users in multiple contexts of use, a situation that we will hereafter refer to as *"multi-context interaction"*. The goal is to implement a precise and easy-to-train gesture recognizer capable of handling numerous gestures that can be used in different contexts.

Over the last few years, the prevalence of touch-screen technology and the emergence of consumer mobile devices relying on this technology (*e.g.*, smartphones, tablets, touch-based and pen-based devices) has led to the development of a large number of 2D gesture recognizers [32] based on different techniques. In addition, the rise of embedded and wearable devices with precise tracking capabilities has opened up prospects for developing multiple applications using 3D gesture recognizers.

A significant number of research works have been produced about developing gesture recognizers, many of which are characterized by being very specific to a context of use [5, 64, 80]. A common characteristic of all these works is reusability: it is challenging to make the gesture recognizer reusable in contexts of use that are different from those initially envisioned, expected, and trained for. Two categories of solutions have received the most attention on this topic: machine learning solutions and template-based solutions. In the case of machine learning solutions, these models are highly capable of recognizing complex gestures, even when dealing with noisy data [50, 114]. In

particular, recognizers relying on deep learning are widespread [11, 82, 106]. These models require large datasets of gestures for training and validation to produce accurate results with complex gestures. However, they have drawbacks such as slow learning times, retraining the model for a new gesture, and the need to involve a large number of users in the process of acquiring this data. For example, the same gesture set can be handled by template-based matching [110] and by Convolutional Neural Networks (CNNs) [29], but CNNs require much more gestures samples than in template-matching, for which only a few already achieve good results [111]. This problem could be overcome by generating synthetic data [66, 77, 81]. This solution may not capture all possible variations a user could produce when performing a gesture, even though many attempts to develop reusable gesture recognizers for different contexts exist [74, 116]. Many of these works are related to template-based gesture recognizers, which rely on the template-matching technique. This technique provides a fast, simple, and accurate classifier. The ease of use and performance of these recognizers have led designers and developers to adopt them for rapid prototyping quickly. One of the key benefits of template-based recognizers is their ability to be independent of the dataset. This means that you can easily add new gesture classes without the need for retraining, providing a significant advantage in terms of customizability.

1.2 Research Question

Many input devices, such as game controllers, smartphones, and cameras, can be integrated into a 3D GBUI. These input devices often include one or more motion sensors and measure various physical quantities, expanding the usage of gesture recognizers to a wide range of contexts. The ability of gesture recognizers to function flawlessly in various contexts of use should be thoroughly studied. Therefore, we state the research question as follows:

This thesis approaches the problem of comparative testing of 3D hand gesture recognition in multiple contexts of use by defining and applying a systematic procedure using template-based recognizers on multi-device gestures.

The gesture recognition procedure consists of several stages: acquisition, training, and testing. In the acquisition stage, we collect gesture data. In the training stage, we use this data to build a recognition model. Finally, in the testing stage, we evaluate the performance of the recognizer using new data. In this context, the meaning of **"Multi-device"** refers to the ability to train the recognizer and classify candidate gestures using data produced by

different input devices at the same or different stages of the gesture recognition procedure. Furthermore, this thesis employs the AMMPT framework, an acronym standing for Approach, Methodology, Method, Procedure, and Technique [6] to distinguish the different meanings as each one refers to one level of abstraction. The term "approach", as defined in this framework, refers to a philosophy or belief about a subject matter. This belief guides problem-solving based on certain assumptions [47]. While the "method" refers to the practical implementation of an approach [47]. It outlines how a research problem would be solved [19]. In addition, based on the AMMPT framework, the "procedure" consists of a sequence of techniques that produce immediate results. Finally, "Technique" refers to specific, observable, and measurable activities performed by users [19] that are essential components of a procedure. Moreover, In Human-Computer Interaction, the performance evaluation of a system involves assessing it using various methods relying on the user and the computer component, which is assessed [137]. To conclude, we will use the last three levels described in the framework in a materialization relationship, which will be used in the rest of the thesis [89].

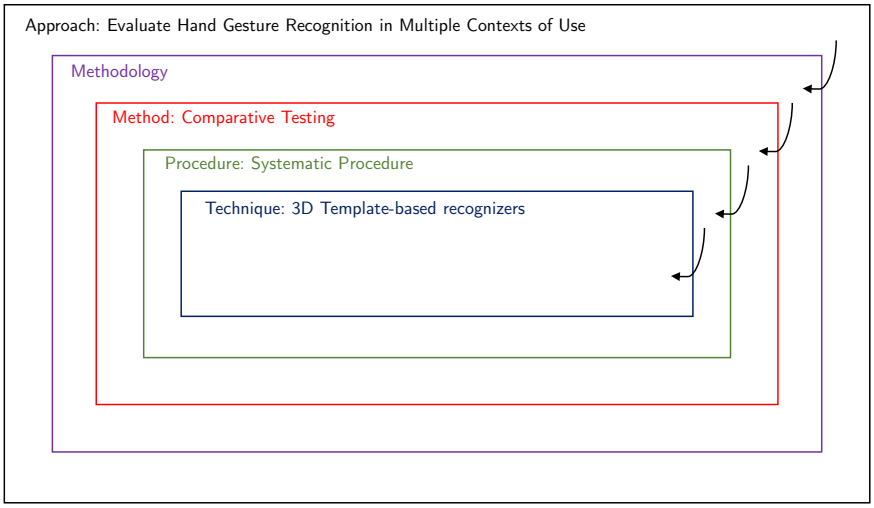


FIGURE 1.1: The AMMPT Conceptual framework applied for the gesture recognition [6]. Reproduced with permission from Springer Nature.

1.3 Working Hypotheses

Researchers focus on various aspects of gesture recognition. Conceptually, some research approaches gesture recognition from a multi-context perspective. This means taking into account the different contexts in which gestures are performed and designing gestures that are adapted to different scenarios. Gesture-based user interface design involves the collection of gestures and the creation of gesture sets that are easy for users to learn and use. These gestures are provided to the gesture recognition system for processing. They are mainly proposed by designers or users. In addition, in the recognition process, devices provide a continuous data stream containing gestures that need to be determined using algorithmic techniques. Several works have tried to tackle these issues to improve the robustness of gesture recognition systems and enhance the naturalness and usefulness of gestural interaction.

While it is possible to conceptualize a gesture recognition system for multiple devices, its technological implementation is limited to a few devices. This limitation arises because it is currently impossible to develop it for all types of devices due to the wide range of available devices. Most template-based 3D gesture recognition work, such as the $ family of recognizers, focuses primarily on hand gestures. For example, $3 [58] and Protractor-3D [57] were evaluated using a handheld device, while $P3D [31] and 3 Cent [23] were evaluated using freehand gestures. It is important to note that these recognizers were tested on gestures captured by specific devices. For example, $3 claims to be a device-independent recognizer, but it was only tested with a game controller that has an accelerometer that measures a specific set of physical quantities. There are also more complex recognizers, such as Jackknife [116]. This section covers several aspects of gesture recognition in different contexts. It starts with template-based gesture recognizers, followed by multi-device configuration for gesture recognition. Finally, it presents their main focus: systematic procedure for a comparative testing method for multi-device gesture recognizers.

In the area of multi-context gesture recognition, this thesis will focus on designing and applying a systematic procedure for comparative testing. Due to the potentially confusing heterogeneity of terminology, we consistently use the AMMPT framework [6] throughout this manuscript, for terms such as recognition techniques and testing techniques. In this thesis, the application of the AMMPT framework is demonstrated. Its implementation is shown in Figure 1.1. The "approach" specifically refers to **the evaluation of hand gesture recognition in multiple contexts of use**. The **"comparative testing"** is a method that helps evaluate the performance of a new software product

against its competitors, helping to develop a higher quality product with improved performance and functionality [132]. Comparative testing in software development provides several benefits, such as identifying areas for software improvement, assessing market positioning, and contributing to product promotion strategies. However, it also presents certain challenges, including the complexity of integrating changes after a series of development and testing phases, and the possibility of customer reluctance if product limitations become readily apparent [2]. In addition, we refer to a **"systematic procedure"** as a structured sequence of steps followed in a regular, defined order to achieve a desired result.

A systematic approach to comparative testing is essential. Such an approach ensures the comparability and reliability of the results. Consider the scenario of comparing two recognition algorithms in two experiments. Using different procedures would make the results incomparable. However, following a systematic procedure ensures the comparability of results. In the wide spectrum of all techniques of gesture recognition, this thesis will focus on template-based gesture recognizers such as the $ family with multiple devices. We primarily focus on mid-air gestures performed with the hands. The systematic procedure can handle the transition from model-based techniques to machine learning and deep learning techniques. This adaptability maintains the consistency of the procedure.

Template-based recognizers have several advantages. On one hand, they are device-independent, which means that new gesture classes can be easily added without retraining, making them highly customizable. They are suitable for rapid prototyping and are easy to train, understand, and use, which is advantageous for developers without extensive machine-learning expertise. On the other hand, deep learning recognizers, offer the possibility of real-time or near-real-time retraining, eliminating the need for complete retraining. This is especially beneficial when new data is introduced. The trade-off, however, is that they can be complex to implement and maintain. They also present challenges such as "catastrophic forgetting", where the neural network forgets previously learned information when new data is introduced [102]. Among all classes of evaluation methods, in the use cases applying the systematic procedure, we will rely on the analytic methods [137] for the comparative testing, which indicates that both the user interface (UI) and user are representational during the evaluation as depicted in Figure 1.2. As a first step, we consider devices that provide user gestures that are 3D trajectories represented by a series of 3D points in a Cartesian coordinate system, performed with wearable and embedded devices that provide physical quantities such as accelerations and orientations. In the end, as a reminder, the main objective is to implement a systematic procedure for comparative testing for template-based hand gesture recognizers adapted to rapid prototyping.

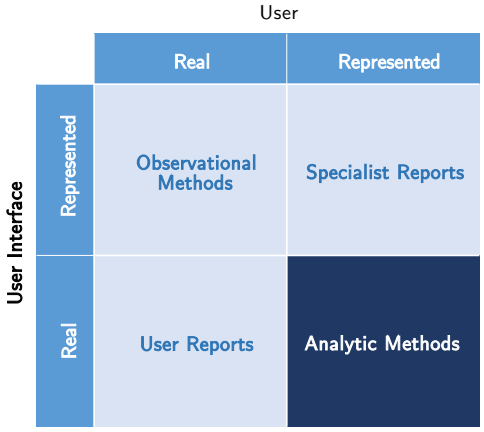


FIGURE 1.2: The categorization of our work within the diverse classes of evaluation methods. [137]

1.4 List of Contributions

This thesis has provided several substantial contributions to the field of the evaluation of 3D hand gesture recognition.

- Creation of a design space for hand gesture recognizers, especially for that adapted to rapid prototyping.
- Analysis of recognizers resulting from a targeted literature review on template-based 3D gesture recognizers.
- Establishment of definitions and a taxonomy for multi-device gesture recognition.
- Definition of a procedure model for testing of 3D gesture recognition and the systematic comparative testing for gesture recognizers.
- Creation of a support tool testing module.
- The application of a systematic procedure for comparative testing. This procedure is applied to template-based 3D gesture recognizers. Two use cases are considered: "unipath dynamic" gestures and "multipath dynamic" gestures.

1.5 Manuscript Overview

There are 7 chapters in the thesis. It starts with the introduction **(Chapter 1)**, followed by the related work **(Chapter 2)** and an explanation of the concepts related to the systematic procedure for comparative testing in different contexts of use **(Chapter 3)**. The following chapters focus on the validation of the systematic procedure for comparative testing of 3D hand gestures by applying it through two use cases **(Chapter 4)**. This chapter evaluates the efficiency of seven template-based gesture recognizers for "unipath dynamic" gestures. **Chapter 5** presents a comparative testing of six state-of-the-art template-based gesture recognizers on two datasets of "multipath dynamic" gestures, measuring accuracy and responsiveness. **Chapter 6** introduces a testing tool designed to implement the systematic procedure for comparison testing of various recognizers across diverse datasets and produce the results. Finally, **Chapter 7** concludes the thesis.

- The **"Related Work"** chapter presents a three-level analysis to understand the state of the art of rapid prototyping gesture recognition. First, it provides some basic definitions to define a gesture, some taxonomies, and classifications and describes some segmentation techniques and recognizers used in 2D gesture recognition. Next, we define a design space for hand gesture recognizers that are suitable for rapid prototyping, which primarily encompasses template-based recognizers and their various characteristics. The three-level analysis begins with a descriptive analysis of the most impactful gesture recognizers for 3D hand gestures. Next, it proceeds with a comparative analysis of 3D gesture recognizers across different dimensions of the design space. Lastly, a generative analysis is conducted to precisely identify what has already been achieved and what still needs to be achieved in 3D rapid prototyping gesture recognition. The chapter concludes by highlighting the importance and motivation for multi-context gesture recognizers in the work.
- The **"Concepts for Comparative Testing of 3D Hand Gestures in Multiple Contexts of Use"** provides fundamental definitions for this work and examines various activities and concepts. It is based on an analysis of state-of-the-art recognizers done in related works. It classifies gesture recognition for multiple contexts based on the findings from this analysis, specifically considering single and multiple-device scenarios. The chapter outlines the stages involved in designing an effective method for testing hand gesture recognizers. It also defines the main components of a gesture recognition comparative testing method. Finally, the introduction of a testing procedure model ensures a clear and efficient systematic procedure.

- The **"Use Case 1: Comparative Testing on Unipath Dynamic Gestures"** chapter presents a comparative testing of seven template-based recognizers to evaluate their efficiency. They are evaluated in a specific intra-device scenario, which implies the use of a unique device during the gesture recognition procedure. Under predefined conditions, using a challenging set of 3D trajectories known as "unipath dynamic" gestures. Recognition rates and execution times are measured for two scenarios, one user-independent and one scenario that includes both user-dependent and user-independent cases. Some recognizers perform similarly to existing 3D recognizers for these gestures. The implications of selecting a recognizer based on the context of use, such as datasets, are also discussed.
- The **"Use Case 2: Comparative Testing on Multipath Dynamic Gestures"** chapter presents a comparative testing of six state-of-the-art template-based gesture recognizers on two datasets. The goal is to compare the recognizers evaluated in a specific intra-device scenario, which implies the use of a unique device during the gesture recognition procedure, using a series of test cases with controlled variables. Accuracy is measured by recognition rate and responsiveness by execution time. The procedure is applied to the SHREC2019 dataset and the challenging JACKKNIFE-LM dataset composed of "multipath dynamic" gestures. The goal is to accurately compare gesture recognizers and investigate the impact of various factors such as number of templates, sampling points, fingers, and hand parameters (*e.g.*, joints, palm, fingertips) on performance. The results will show the most accurate or fastest configurations for each recognizer.
- The **"QUANTUMLEAP Extension: Testing Module"** chapter provides a comprehensive overview and detailed explanation of the implementation of an extension to the QUANTUMLEAP framework. The goal of this extension is to automate the evaluation procedure of gesture recognizers, allowing the use of multiple gesture sets. In addition to discussing the technical aspects of this implementation, the chapter delves into the pipeline used for the evaluation procedures of real-world use cases for gesture recognizers. Finally, the chapter presents the structure of the result files.
- Finally, the **"Conclusion"** provides an overview of the contributions and describes the advantages and limitations of the solution. A subsection entitled "Future work" aims to provide a concise list of short- and medium-term goals for further development of multi-context gesture recognition procedure.

Chapter 2

Related Work

This chapter provides a three-level analysis to understand the existing state of the art in template-based gesture recognition. It commences with introducing elemental definitions for gestures, taxonomies, and classifications in **Section 2.1**, which also covers segmentation techniques. The following section, **Section 2.2**, briefly describes the details of recognizers used in 2D gesture recognition. Lastly, **Section 2.3** defines a design space for rapid-prototyping recognizers and explores their various features.

In this chapter, we rely on the Generative Theories of Interaction model described by Beaudouin-Lafon et al. [14]. This framework helps researchers understand current technology use and propose future interactive systems through three successive lenses: analytical, evaluative, and constructive. In this chapter, we will conduct a descriptive analysis of the most impactful rapid-prototyping template-based gesture recognizers for 3D gestures in **Section 2.4**, which will be followed by a comparative analysis of these 3D recognizers in **Section 2.5**. Finally, **Section 2.6** presents a generative analysis on multiple aspects.

2.1 Gesture

- *"Gestures are expressive, meaningful body motions involving physical movements of the fingers, hands, arms, head, face, or body with the intent of 1) conveying meaningful information or 2) interacting with the environment."* is the definition given by Mitra for a gesture [80].
- In a more general explanation, Stam & Ishino characterized gestures as *"As all visible bodily actions employed intentionally and meaningfully"* [113].

In the Human-Computer Interaction field, a gesture can be defined as any physical movement that a digital system can detect and respond to without the use of a traditional pointing device such as a mouse or stylus [101]. In

the field of Human-Computer Interaction (HCI), there are two main types of interaction: 2D interaction and 3D interaction.

- In 2D interaction, the focus is initially on stroke gestures. According to Li [68], stroke gestures are defined as *"2D trajectories drawn by users with their finger on a touch screen or with a pen so that a computer system can act based on recognition results"*.
- On the other hand, 3D spatial interaction [65] is defined as an interaction where users are tracked in space over time using different kinds of sensors, whether obtrusive or unobtrusive [17]. Thence, it allows the user to be immersed in a 3D virtual environment and provide commands as a virtual character. Additionally, it enables users to issue commands and perform activities by generating specific patterns over time recognized by the computer and defined as 3D gestures.

2.1.1 Stroke Gesture

- First, in the parametric recognizer developed by Rubine, a single-stroke gesture is defined as *"hand markings entered with a stylus or a mouse, that indicate scope and commands: (1) the operation (e.g., move text), (2) the operand (e.g., the text to be moved), and (3) additional parameters (e.g., the new location of the text)"* [97].
- Vatavu defined a 2-dimensional stroke gesture as *"any path traced onto a touch-sensitive surface by a user with a finger or a stylus"* [125].

Some interesting types of stroke gestures are defined in the literature:

- [97] made a distinction between "single-path" and "multi-path" gestures. "Single-path" gestures involve articulating a single stroke, while "multi-path" gestures involve articulating several strokes simultaneously (*e.g.*, drawing with 2 or more fingers at the same time).
- Finally, [105] distinguished between "single-stroke" and "multi-stroke" gestures such that *"each set of candidate points sampled between the time it contacts the surface (e.g., mouse, pen or finger down) and the time it breaks contact (e.g., mouse, pen or finger up) belongs to a stroke"*.

2.1.2 3D Gesture

- Liu defined a gesture in his personalized gesture recognition system. He identified a gesture as *"free-space hand movements that physically manipulate the interaction device. Such movements include not only gestures as we*

commonly know but also any physical manipulations like shaking and tapping of the device." [69].

- La Viola gave a technical definition of a 3D gesture as *"a specific pattern that can be extracted from a continuous data stream that contains 3D position, 3D orientation, and/or 3D motion information. In other words, a 3D gesture is a pattern that can be identified in space, whether it can be a device moving in the air, such as a mobile phone or game controller, or a user's hand or whole body"* [65].

Some interesting simple types of 3D gestures are presented:

- [80] distinguished gestures into *"hand and arm gestures"*,*"head and face gestures"*, *"body gestures"*, based on the body parts used to perform them.
- [12] classified gestures into "Static" and "Dynamic": *"If the user adopts a certain pose, it is called a static gesture. Dynamic gestures are defined as tracked motions over time."*

2.1.3 Gesture Taxonomies and Classifications

Many taxonomies are defined based on a set of dimensions that classify gestures and are domain-dependent. These taxonomies include various gesture variations and provide designers with a detailed view of the gesture space. Furthermore, a *Gesture Elicitation Study* (GES) [139] is a process of understanding user preferences and behavior in relation to a new interactive technology. This study provides designers with valuable information for designing product features to improve usability.
Based on a taxonomy and a notation expressing 3D hand gestures captured with a Kinect, a Gesture Elicitation Study (GES) resulted in 43 user-defined gestures for 6 TV actions [30]. These actions include turning the TV on/off, going to the previous/next channel, and opening/closing blinds. The fine-grained hand gestures are represented by six intrinsic properties: hand location, shape, movement, and orientation.
An ontology for Kinect gestures has already been devised to categorize such gestures [38], which also introduces a formal notation for reasoning on gestures. Other notations come from linguistics [44] or from multimodal interaction modeling [91]. There are many taxonomies that exist and are briefly described:

The first taxonomies were closely related to linguistics.

- We cite Efron [40], who classified human gestures into *"Objective"* gestures, which have a meaning independent of speech and multiple subcategories that are *"Deictic"*, *"Physiographic"*, *"Emblematic"*. Also, in *"Logical-discursive"* gestures, which go along with the speech, it regroups two sub-categories *"Batons" and "Ideographic"*.

- Kendon [52] introduced a continuum of categories ranging from the least formal and most speech-related gestures (Gesticulation) to the formal and speech-independent gestures (Sign Languages).

Gesticulation $\longrightarrow$ *Language-like Gestures* $\longrightarrow$ *Pantomimes* $\longrightarrow$ *Emblems* $\longrightarrow$ *Sign Languages*

- Inspired by the precedent works, Mc Neill [78] provided a significant classification based on four types of gestures *"Deictic"*, *"Iconic"*, *"Metaphoric"*, *"Beat"*.

Although the above classifications are effective, they are still limited, especially in an interaction context, because of their initial discourse-related purpose. For this reason, many taxonomies appeared to describe gestures for computer interactions:

- The categorization proposed by Karam and Schraefel [51] based on the work of Quek [94] and the literature classifies gestures into five categories: The *"gesticulation"* (1) defined as coverbal gestures that support speech. The *"manipulations"* (2) are gestures that link the movement of the hand to the manipulated object. These gestures can take place in 2D or 3D by manipulating an object or using bare hands. The *"semaphores"* (3) are communicative gestures based on a dictionary of static or dynamic gestures and even stroke gestures. The *"deictic"* (4) gestures point to identify an object or its position in space. The last ones are *"language gestures"* (5), which have grammar and are self-sufficient as a communication tool. They are mainly used in sign languages.

- Vatavu, classified gesture commands within the context of interaction regarding their structural pattern, as given by the degree of posture or movement in the execution of the command [127], into four different types. *Static Simple Gestures* (1) described as a posture (*i.e., a set of values that describe the pose of the body or parts of the body at a given moment*) held for a period of time. *Static Generalized Gestures* (2) described as gestures that are represented by a series of consecutive postures that are maintained for certain amounts of time. *Dynamic Simple Gestures* (3) described as gestures where the primary information is the motion trajectory and not the posture. Finally, *Dynamic Generalized Gestures* (4) described as gestures where both the trajectory and the posture have the same importance.

- The Guessability study conducted by [100] resulted in a taxonomy that divides gestures into two classes of taxonomy dimensions, the *"Gestures mapping"* and *"Physical Characterstics"* classes. The first group *"Gestures mapping"* (1) contains several dimensions that detail the link that the user creates between the gestures motion and the controls of the device, encompasses three dimensions: the *"Nature"* (a), which describes the correspondence of the gesture performed with the physical items. This link can be of the *"Metaphor"*, *"Physical"*, *"Symbolic"*, or *"Abstract"* type. The *"temporal"* dimension (b) indicates whether the action performed on an object takes place at the same time as the gesture performed by the participant or describes the action that is triggered at the end of the gesture. The last dimension described is the *"context"* (c), which indicates whether the gesture needs a defined context or not. The second class of "physical characteristics" (2) is based on the kinematic properties of the gestures, such as *"kinematic impulse"* (a), defined as the range that the jerk can have during the performance of the gesture. In addition, the second property is *"dimension"* (b), which expresses the number of axes on which the movement is performed. Finally, the *"complexity"* (c) defines whether a gesture is *"simple"* or *"complex"* if it is decomposed into *"simple"* gestures.

- In their study [3] used a more extensive classification than that proposed by Karam and Schraefel [51] by detailing some gesture types and by omitting sign languages. This improved classification separates the *"gesticulation"* into two categories represented by *"iconic"* (1) gestures, which transmit information about concepts or objects as letters, digits, symbols, drawings. Moreover, this type of gestures are divided into *"static"* and *"dynamic"*. The second category is *"pantomimic"* gestures (2), which describe the performance of a task relying on a set of gestures. Another gesture type is *"pointing"* (3), which is used to point out a specific object or indicate a direction using different parts of the hand. The *"semaphoric"* gestures (4) are movements that communicate specific information and can be *"static"* through defined hand poses, *"dynamic"*, which are defined by taking into considerations of time and *"strokes"*, which are dynamic hand gestures defined by a stroke motion. Finally, *"manipulation"* gestures (5) are performed by the actor and are linked to the movement of the object he is manipulating with a feedback loop.

- Choi's taxonomy [30] classified three-dimensional hand gestures based on elements from the literature into three detailed categories and are composed of sub-elements providing more details : The *"gesture type"* (1), which defines the form of the mono or bimanual gesture. The category of *"static gesture"* (2) which are gestures defined with a single image

consisting of four elements: *"Hand Location"*, *"Hand Shape"*, *"Hand Orientation"* and *"Arm Shape"*. The last category is *"dynamic gesture"* (3), which are gestures represented as a set of static gestures (*i.e.*, hand movements over time). This class contains three types of gestures *"Hand Shape Transition"*, *"Hand Transition"* and *"Hand Shape Transition"*.

2.1.4 Gesture Segmentation

One of the main challenges of recognizing dynamic gestures is determining the beginning and end of an expressive gesture in the data stream recorded by a motion sensor. In particular, if the user cannot specify them by pressing a button or maintaining a specific posture. In the context of human-computer interaction, this involves determining when a user is performing a command gesture. This problem is known as the Midas problem [104] or as gesture segmentation, and sometimes it is cited as gesture spotting. There has been considerable work done to address it, based on many approaches.

Zoning approach

An approach where the user performs gestures in front of a device, while delimited by a set of thresholds gesture space, defines the gesture record area. The gesture recognition process is triggered when the bounded area contains one or more points. Once the selected points have been recorded, the gestures are sent to the recognizer. Among the works that have used this approach is the hand gesture-based interface developed by [60] with a Kinect sensor. They have segmented the data by defining an "input zone" where the movements are performed to segment the continuous gesture.

Threshold approach

This segmentation approach defines the start and end points of a gesture by determining whether the values of one or more characteristics of the gesture reach a defined condition, such as a change in acceleration, velocity, energy. Most of the time, the values of these conditions are determined empirically. An example is Pactolus [26] where the authors segmented the real-time EMG signal provided by the sensor using two features, energy and root mean square (RMS).

Window Segmenter

Window segmentation is the simplest segmentation process that divides the motion stream into many equal-length segments called windows. Each time the sensor provides a new frame, it is added to the window(s) while the oldest

one is deleted. When the window reaches the maximum number of frames, it sends the gesture data it contains to the recognizer. [70] used this approach in an algorithm that adapts the window size using training data from a data glove.

Other more advanced segmentation methods exist such as the one used by [126] to reject candidate gestures by designing rejection rules, and they use a scale-invariant method based on the absolute curvature integral for 2D gestures. There is also the Machete technique [117] which is a fast gesture segmenter based on continuous dynamic programming (CDP) with many feature improvements.

2.1.5 Gesture Representation and Notation

This subsection will develop some interesting gesture representations and notation methods used in different contexts to analyze gestures and facilitate their interpretation. These notation methods are dedicated for different parts of the body, such as for hand gestures [30] from sign language notations, or used for the body in a dance context like the Labanotation or the Eshkol Wachman Movement Notation (EWMN).

2.1.6 Motion Sensors

In the field of HCI, interactive systems are composed of several parts that allow users to interact with the system naturally. Each part has a specific function as shown in Figure 2.1. The first part captures the user's data through an input device consisting of one or more sensors, the captured data is analyzed and processed, and then the results are presented to the user. In their review, [12] summarize the evolution of interaction devices developed through time, ranging from the *keyboard* in the early systems that were first used with CLI (Command Line Interface) in (1952), this was followed by the invention of the *light pen* in the same year. The development of *mouse* device took place in 1965, due to the development of Graphical User Interfaces (GUI), but also at the same period, the first *game controllers* appeared. In the 1980s, new natural interaction devices emerged, such as *data gloves* that map the finger movements to GUI interactions, and new vision-based interaction systems with a *camera*. The next decade saw the emergence of *touch screen* technology with many devices such as *multi-touch displays* and *smartphones*. *3D mice* came on during this period, while a few years later, an improved version appeared in the gaming context, with the *Wii Remote* and *PlayStation Move*, and in the VR/MR context with the *Oculus Touch* and *Playstation Move*. Since 2010 and the appearance of the *Kinect* device with its two versions for the general public using a depth

sensor to track full-body motion, many other depth-based devices appeared, for instance, the *Intel Real Sense* or the *Leap Motion controller* for finger and hand gestures. Finally, the *Myo Armband* is a device based on EMG and IMU sensors and is worn on the user's wrist.

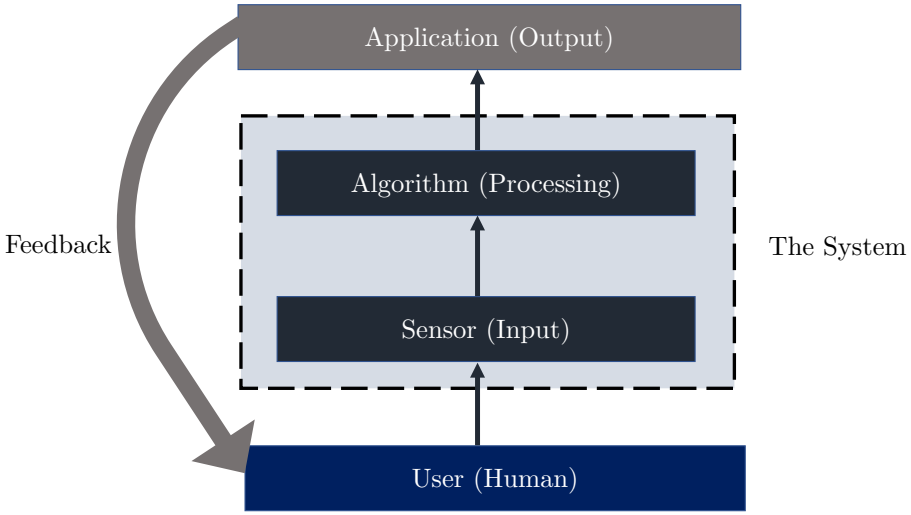


FIGURE 2.1: A diagram that summarizes the various phases of gesture recognition in an interaction system.

2.2 Related Work

2.2.1 Gesture Recognition

The domain of gesture recognition has expanded quickly, resulting in a large body of work on gesture recognition. We conducted a Targeted Literature Review (TLR), which is a non-systematic, in-depth, and informative literature review that keeps only the references maximizing rigorousness and relevance while minimizing selection bias [62]. The 2D stroke gesture recognizers selected in the next subsection and resulting from the TLR will be extended to support 3D gesture recognition in a subsequent chapter, an extended description of these and other 2D stroke gesture recognizers are presented in the survey of [73], and also in the extensive literature that covers this topic. As for the 3D recognizers, they will be discussed later in this chapter in a descriptive, comparative, and generative analysis based on the different defined criteria of the design space for hand gesture recognizers adapted to rapid prototyping.

A gesture recognizer is a module that distinguishes the gestures expected by the system [97]. The gesture classes have labels that define the type of gesture. They constitute a set called a "gesture set". Firstly, a gesture recognizer works in such a way that first, it has to be trained with provided gestures (*i.e.*, training examples). Then, it waits for the gestures provided by the users (*i.e.*, candidate gestures) in order to classify them into one of the predefined gesture classes [45]. There are many metrics that exist to evaluate the performance of a gesture recognizer, but the recognition rate is the main metric to evaluate the ratio of successfully classified candidate gestures.

2.2.2 2D Stroke Gesture Recognizers

Several algorithms recognize 2D stroke gestures [143] by *template matching* [18]: they compare a *candidate gesture* against pre-recorded *template gestures* gathered by *gesture classes* in a *training set* that was used to train the recognizer beforehand. This assumes calculating a (dis-)similarity distance between the candidate gesture and template classes. While formerly aimed at user interface prototyping, the $-family initiated several applications and recognizers [23, 45, 59, 61, 95, 120].

GRANDMA (1991)

Rubine's recognizer [98], creates a vector of 13 geometric features for each template and candidate, which is classified with respect to gesture classes by a linear evaluation. This offers a high recognition accuracy for a low computational cost, manual opportunistic programming is no longer required, and its understanding is simple and facilitated by a geometric interpretation. These three advantages form a common motto for several other recognizers.

$1 (2007)

The $1 recognizer [140] recognizes 2D uni-stroke gestures into four steps: resampling, rotation, scaling, and moving. The candidate gesture is resampled into a fixed number of points that are evenly spaced along the path. Then the path is rotated so that the line from the centroid of the path to the first point of the path is parallel to the x-axis. Non-uniform scaling is used to fit the path in a reference square. Finally, the path moves its centroid to the origin $(0, 0)$. The mean value of the Euclidean distance between the corresponding points is computed as a similarity distance.

PROTRACTOR (2010)

Instead of the Euclidean distance, PROTRACTOR [68] minimizes the cosine distance between two gestures by calculating the best rotation angle, but no scaling is performed.

$\$N$ (2010)

The $\$N$ recognizer [7] generalizes $\$1$ to 2D multi-stroke gestures. Each gesture sample is considered as a uni-stroke where part of the gesture is made away from the sensing surface (*i.e.*, by following the user's hand through the air), and the algorithm automatically generates all possible uni-stroke permutations of the gesture.

$\$P$ (2012)

The $\$P$ recognizer [124] overcomes the combinatorial explosion of $\$N$ by representing gestures as unordered point-clouds, therefore making them invariant to order, number, and direction.

Penny Pincher (2015)

Penny Pincher [115] converts resampled gesture points into a series of between-point vectors and calculates the sum of the angles between their corresponding vectors. Penny Pincher, due to this simplicity, is reported as a super-fast algorithm with a high accuracy (>90%).

$\$P+$ (2017)

The $\$P+$ algorithm [122] increases the $P flexibility such that any point from the first cloud can be matched with any point from the second cloud. Although this recognizer was tested for low-vision users, it turns out to be the most accurate recognizer of the $-family.

$\$Q$ (2018)

The $\$Q$ recognizer [125] improves $\$P$ with several algorithmic revisions: the classification of multi-stroke gestures is faster when the size of the training set is large. Aligned with the accuracy of $\$P$, $\$Q$ is announced as the fastest recognizer of the $-family.

!FTL (2018)

The !FTL recognizer [118] converts resampled gesture points into a series of pairs of vectors called "shapes" and computes the Local Shape Distance (LSD) between each pair of shapes in sequence to assess how two gestures are dissimilar.

2.2.3 3D Gesture Recognizers

3D gesture recognition is a large field of research and development, which considers several aspects such as the proliferation of devices, whether wearable or not, and the parts of the human body considered for the gesture; and finally the techniques used for the recognition. There exist many techniques that are used [27, 141]: general purpose toolkits [136], Artificial Neural Networks (ANN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), k-Nearest Neighbor [39, 123], Support Vector Machine (SVM), Hidden Markov Model (HMM), Dynamic Time Warping (DTW) [116], Naive Bayes classifier, Principal Component Analysis (PCA), Deep Learning [72].

One such technique is the AdaBoost algorithm [43]. In gesture recognition, Sheng [107] used the AdaBoost to solve a multi-class recognition problem for 3D hand gestures, using a Vicon motion tracking system to track body movement. The principle of a boosting algorithm is to focus on incorrectly classified data by increasing its importance each time the AdaBoost calls the weak classifier, resulting in a series of hypotheses that are combined into a single hypothesis at the end. The author used the AdaBoost.M1, an extension for solving multi-class classification problems, which only considers correctly labeled samples and maximizes the sum of their weights. For the data, instead of using 3D point sequences, the author extracted a set of 16 features similar to those selected in [98] but extended to 3D and following defined criteria. AdaBoost was tested against three weak classifiers (*i.e.*, 1NN, NaiveBayes and LDA) on a dataset composed of 6 gestures, and for each gesture, a total of 200 samples were recorded (*i.e.*, 120 were used for the training and 80 for the testing) showing good results although it was sensitive to noisy data, particularly for the NaiveBayes and LDA weak classifiers in comparison to the base classifiers.

A common pattern identified in this portion of the literature is the overfitting in the data sets for artificial intelligence and machine learning algorithms: they exhibit an accuracy superior to 90% and a performance inferior to 100 msec., a threshold for system response time [84], but they are entirely optimized for a given dataset, thus leaving little or no space for keeping the same performance if the dataset is changed. This occurs when samples and/or

classes need to be modified, thus leading to a re-training of the algorithm, although this is now achievable in real-time.

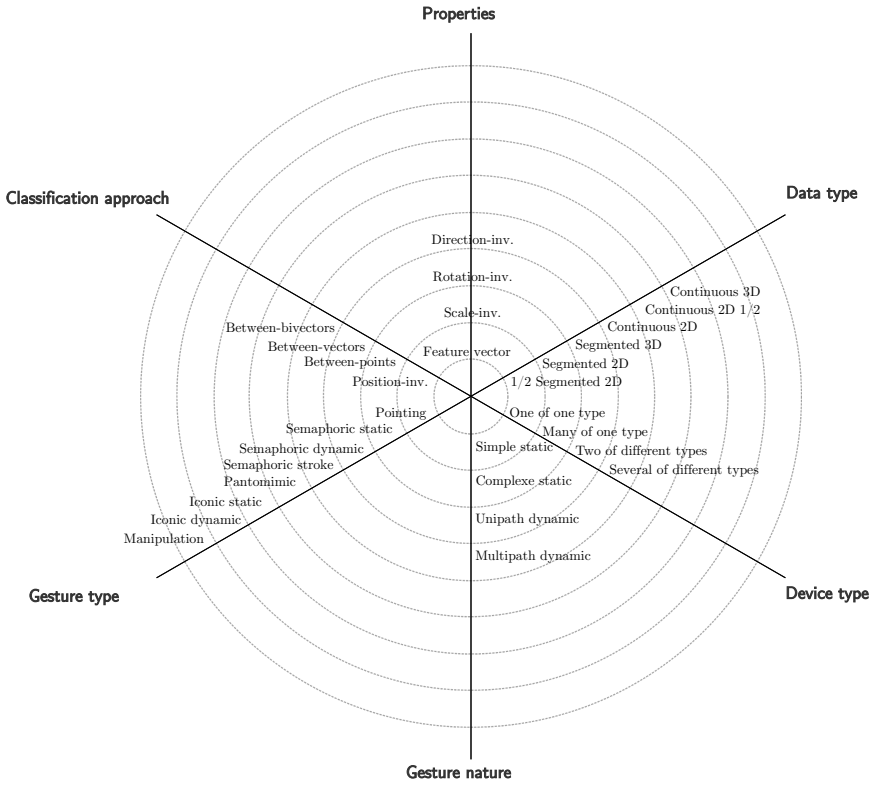


FIGURE 2.2: A design space for hand gesture recognizer.

2.3 A Design Space for Hand Gesture Recognizers

From the references cited before, we determined common features between them independently of their domain, as long as rapid prototyping is involved to a certain degree. This resulted in the design space for hand gesture recognizers adapted to rapid prototyping. It is based on six dimensions represented in Figure 2.2: recognizer properties, data type, device type, gesture nature,

gesture type, and classification approach. Every dimension is arranged in a progressive degree of complexity: each step starts from the simplest value found to the most complex one. We have considered the radar diagram as the most appropriate to introduce the three main aspects to characterize a recognizer as a model that can be evaluated: *descriptive* (the design space should be able to describe any 3D gesture recognizers based on these six features), *comparative* (the design space should be able to compare two or more recognizers to catch their similarities and their differences); *generative* (After doing a comparison the design space should be able to show unexplored possibilities and to propose new ideas and configurations). These six dimensions are not designed to be independent of each other. Instead, they aim to serve these three values. The Radar chart can be used to represent the values along the multiple dimensions formed by the features. The steps on each dimension are not intended to be aligned or to be corresponding by concentric level. Therefore, steps joined on the same circle are not necessarily dependent; they only represent some progression.

Properties

A recognizer compares two gestures in the form of data provided by devices. The data are subject to variations that make recognition more difficult. That's why the different recognizer (pre-)processing stages provide a set of insensitivities or invariances to the recognizers along one or multiple properties which are sorted from the most frequent to the least covered, as **1) scale** (*by scaling gesture points to a unified scale, the "small" and "big" gestures are unified*), **2) position** (*Gestures performed in different location of the space are moved to the same location by translating the gesture points to a defined point as the origin*), **3) rotation** (*Gestures can be performed with different angles in the 3D space and can be adjusted to the same orientation by using angular adjustments techniques*) and **4) direction** (*Gestures can be performed in different directions, the direction can be omitted by using points cloud representation*). From the literature, all the selected recognizers are sampling-invariant. That's why we didn't include the value as one of the property values.

Data Type

The data type is a characteristic inherent to the device, determining the dimensionality and format of the data transmitted by devices and received by the recognizer for processing. Gesture data can be continuous as provided by the sensor or segmented, as seen in (see Subsection 2.1.4). Moreover, it can be in 2-Dimensions when the gesture is performed on a surface. or a 2D 1/2 when gestures are performed in 3D space then it is projected in a 2D single

plane where one of the coordinates is zero. Finally, gesture data also could be in 3-Dimensions provided by sensors from mid-air gestures. The values are ordered from the simplest to process (*i.e.*, 2D segmented) to the most complex (*i.e.*, 3D continuous).

Device Type

The device type is a feature that provides information on the number of devices that are used to evaluate the recognizer. Furthermore, this feature indicates if the used devices during the evaluation have the same type or are of different types [20]. The values are detailed along the axis, starting from the most basic configuration (*i.e.*, One of one type) to the most complex one (*i.e.*, several different types).

Gesture Nature

The gesture nature gives information about how a gesture is performed and its complexity. The values represented alongside the axis are taken from the taxonomy of [127] which are described in Subsection 2.1.3: **Simple static**, **Complex static**, **Simple dynamic** and **Complex dynamic**. However, we replaced the dynamic values with new values which are more accurate and correspond to the basic representation of gesture data, **Unipath dynamic** (The gesture is represented as a serie of points which represents the motion of articulation) and **Multipath dynamic** (The gesture is more complex, as it is represented as several series of points defining the movement of multiple articulations).

Gesture Type

The gesture type dimension provides information on the classification of gestures that have been evaluated by the recognizer based on the taxonomy that has been defined by Aigner *et al.* [3] and described in the Subsection 2.1.3). the values are sorted in an ascendant way from the simplest, which is the *pointing* class, to the most complex gesture class which is the *Manipulation*.

Classification Technique

In order to classify the gestures, the recognition machine has to compare two gestures, the comparison of the gestures can be direct between the points that constitute the gesture, or converted and preprocessed into defined structures such as feature sets, vectors or bi-vectors. The classification approach values are ordered by the date of appearance of the approach, from oldest to most recent **a) Features vector:** Compute a feature vector from the points to describe the gesture and process the feature vector for the classification. **b)**

between-points: The approach is based on the direct processing of points. **c) between-vectors:** The approach converts the set of points into a set of vectors by creating a vector between each pair of points, followed by direct processing of vectors. **d) between-bivectors:** the approach converts the set of points into a set of bivectors (*i.e.*, a bivector is created by two vectors resulting from the preprocessing of three consecutive points), followed by direct processing of the bivectors for classification.

2.3.1 Rapid Prototyping Features

In this thesis, we focus mainly on template-based recognizers adapted to rapid prototyping and designed to retrieve the gesture class of a candidate gesture using several samples called templates. We present a summary of selected recognizers in Table 2.1that meet the three principal criteria for a rapid prototyping recognizer. The three criteria are:

- **Easy to understand.** The recognizer is designed to be easy to understand, meaning that the concepts used are simple and require only basic knowledge of geometry and trigonometry.
- **Easy to implement.** The deployment of the recognizer relies on small, comprehensible functions for the designer. The first function adds new templates to the training set, and the second function classifies gestures by relying on the training set. Some recognizers have a third function that allows the algorithm to be trained using the templates to recognize candidate gestures.
- **Easy to use.** The use of the recognizer does not depend on a specific device and is easy to adapt. It only requires a small number of templates to achieve a high recognition rate.

2.4 Descriptive Analysis

Next, we describe the main developed template-matching 3D hand gesture recognizers and focus on those suitable for rapid prototyping by instantiating the design space described above which relies on a set of features extracted from the different recognizers selected from the literature. Based on these features, other recognizers based on techniques other than template matching can be added, such as those based on statistical matching or deep learning.

TABLE 2.1: Rapid prototyping properties of the 3D recognizers (○: criteria not satisfied, ◐: criteria partially satisfied, ●: criteria satisfied).

Recognizer	Rapid prototyping criteria		
	Easy to understand	Easy to implement	Easy to use
uWave	●	●	●
Rubine3D	●	●	◐
$3	◐	●	●
Hoffman's Lin. clf.	●	●	◐
Protractor3D	●	●	●
$P3D	◐	●	●
3 Cent	●	●	●
Jackknife	●	◐	●
DeepGRU	○	◐	●

2.4.1 uWave (2009)

In 2009, Liu *et al.* raised the problem of the lack of a standard vocabulary in gesture recognition compared to other pattern recognition. Users should enter their gestures or customized gestures. Furthermore, with this type of gesture, it is hard to create a large training set to use statistical methods such as the Hidden Markov Model, as the platforms used for personalized gesture recognition are very limited in resources. They proposed a new recognizer, uWave [69] is a template-matching recognizer that requires only one training sample per gesture, which is appropriate for resource-constrained platforms.

Design Space Instantiation

1. **The recognizer.** The uWave classifies gestures by first performing a two-stage quantization of the acceleration data to reduce the rapid changes in values resulting from noise and slight shake or tilt of the hand. Starting with a temporal compression by an averaging window that progresses at a time step. Next, the acceleration data are non-linearly quantized in one of 33 levels, resulting in a set of discrete values. The recognition process relies on matching a quantized time series represented as points with another time series by Dynamic Time Warping (DTW), leading to Euclidean distance calculation as a cost function of the DTW. The basic assumption of the uWave author on gesture representation is that human gestures can be described as a time series of forces being applied to a handheld device. The raw data is represented as a set of temporal samples where each sample is a three-dimensional vector characterizing the acceleration along the three axes, this representation implies that the recognizer is position invariant. Furthermore, to cope with sample variations of gestures performed by the same user after a number of days, uWave uses a

template adaptation through the time following two schemes, by keeping two samples by gesture generated in different days by the same user, each time the user input a gesture is recognized or not well recognized it replaces the oldest template.

2. **The device.** The recognizer was evaluated using only acceleration provided by a WiiRemote; the acceleration-based gestures were recorded as segmented gestures by pressing a button.

3. **The gestures.** The evaluation of the recognizer was done on a user-dependent scenario only. The database, composed of iconic gestures, comprises 8 dynamic gestures representing the paths of hand movement with more than 4000 samples collected from 8 users over several weeks using the WiiRemote.

Rapid Prototyping Criteria

The uWave uses many concepts such as the quantization of the input values and the Dynamic Time Warping (DTW) for the matching which is a comparison between two time series to find the best alignment given a local cost function, as well as optional optimization like the rejection criteria applied on the input gesture and temporal adaptation which are not very intuitive for designers. The recognizer is easy to deploy with simple functions to start and end gesture recording and to detect the gesture. The recognizer requires only one training sample per gesture and the evaluation showed high accuracy.

Performance

The uWave showed good results in terms of accuracy and recognition speed. It achieved an overall recognition rate of 93.5% without template adaptation and increased to 98.6% with the adaptation, using a single sample in the training set. While for the recognition speed of uWave, it was evaluated on multiple platforms. On a Lenovo $T60$ with 1.6 GHz Core 2 Duo, it took less than 2 ms for a template library of eight gestures. On a T-Mobile MDA Pocket PC with Windows Mobile 5.0 and 195 MHz TI OMAP processor, it took about 4 ms for the same vocabulary. It was also tested uWave on a 16-bit micro-controller for which the delay was about 300 ms.

2.4.2 Rubine3D (2009)

Rubine3D of iGesture [93] is an extension of the framework iGesture [108] developed to extend its capability to process 3D gestures, along the 2D gestures that are supported by the framework. The Rubine3D recognition process in

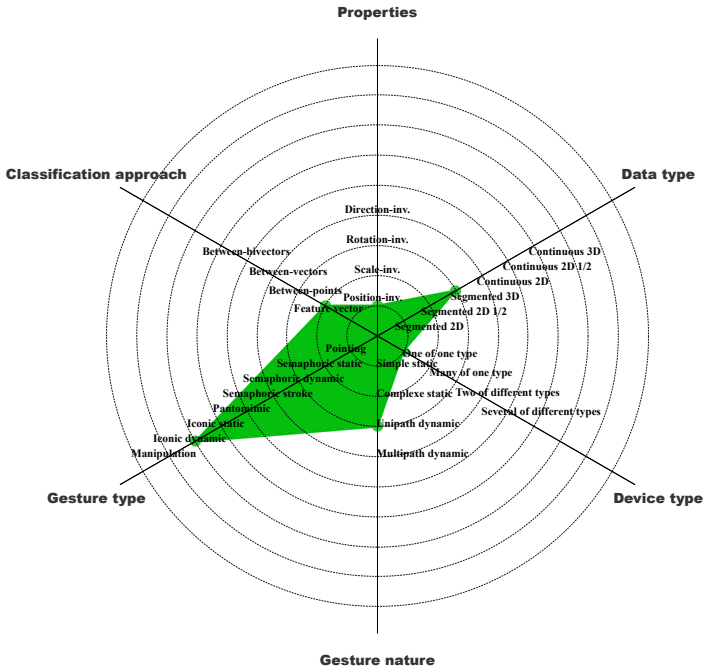


FIGURE 2.3: The space design instantiation for uWave.

iGesture is carried out by projecting the 3D gesture to the three 2D planes, *i.e.*, XY, YZ, and ZX, and applying the Rubine algorithm [98] on each plane with a weight.

Design Space Instantiation

1. **The recognizer.** The Rubine algorithm [98] which is a statistical classifier based on examples as described in the description of 2D recognizers part.2.2.2. The captured 3D gestures are projected into three planes XY, YZ and ZX and the Rubine algorithm is applied using a linear evaluation of the feature vector defined by Rubine. The algorithm is trained with the templates of each class to determine the weights of the features for each class. Then the classifier return the corresponding class by computing the distance between the candidate gesture and the mean of each classes and return the class with the lowest distance. Each recognizer on each plane provides a result and they are put together knowing that each plane has

its own weight factor which the sum gives 1. These factors permit to configure the recognizer if some gesture are more significant on a particular plane than others that allows to increase the accuracy of the recognizer. As for the 2D Rubine [73] the trained 3D Rubine are position invariant but are scale and rotation sensitive, they are also direction sensitive.

2. **The device.** In addition, a module was added to support 3D input devices such as the Wii Remote controller, which provides linear acceleration. The data are recorded using a trigger to delimit the start and the end of a gesture.

3. **The gestures.** An evaluation of the Rubine3D algorithm was made with two experiments on two gesture sets containing acceleration data that are converted to position data to create 3D gesture representation. The first one is composed of vertical gestures performed in a single plane in the air. The set included 7 gestures (*e.g.*, circle or an X), the test required 10 samples per class out of a total of 15 samples. Like for the first gesture set included 7 gesture classes (*e.g.*, handshake, and tennis forehand), and the test set included 10 samples per class out of a total of 15 samples.

Rapid Prototyping Criteria

The Rubine 3D uses a feature vector of 13 mathematical features to describe the shape of the gesture on each of the 2D planes, which has a weight. The difficulty is to understand the statistical evaluation in the classification part. The recognizer relies on two main functions: one to train the recognizer on the loaded gesture set and the other to recognize the candidate gesture.

Performance

The average recognition accuracy for this test was 98.5%. The second gesture set tested used "true" three-dimensional gestures with a lot of variations in the three planes. For this type of gesture, a higher weight is assigned to the plane in which the gesture is made. The recognition accuracy for this test is less significant (95.7%). The configuration where the weights of the three planes are more or less equal provided the best results.

2.4.3 $3 (2010)

The emergence of mobile devices incorporating 3D acceleration sensors requires developing new methods adapted to the mobile platform. At the same time, gesture input for such devices can help address the limitation of miniature input devices and small screens, as the device's size does not limit the

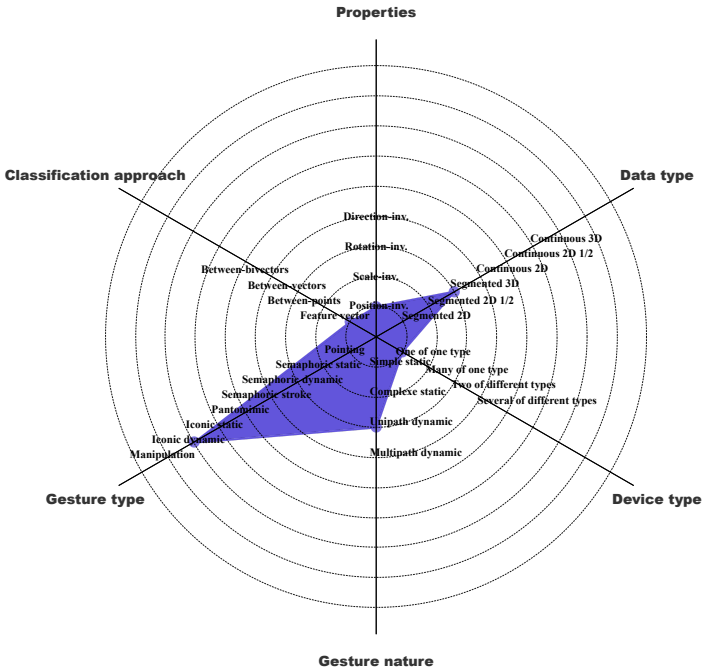


FIGURE 2.4: The space design instantiation for Rubine3D.

range of motion. Consequently, [58] developed $3 A 3D gesture recognizer that works with 3D acceleration data as an extension of the $1 [140].

Design Space Instantiation

1. **The recognizer.** As for $1, the $3 recognizer is based on template-matching method, *i.e.*, the challenge is to find for a an input gesture I, the best matching gesture class G in the training set L. Recognition goes through the same pre-processing steps: (1) isometric resampling of the candidate and template gestures with the same number of points. The resampling uses a piece-wise linear interpolation with (N=150). The aim is to ensure an equal distance between points to guarantee the sampling invariance. Furthermore, the number of points affects the recognition precision and the computation time. (2) Rotation of the resampled gesture trace along the "indicative angle" which is the angle between the first gesture point

and the centroid. The angle is calculated from the arcus cosine of the normalized scalar product of the two points. (3) Scaling the rotated gesture to overcome scale differences between gestures. The gestures are scaled to fit into a normalized $100 : 3$ units. Moreover, same as Wobbrock in $1, the rotation-invariance of path distance in $3 is ensured by the GSS approach but adapted to three dimensions, for the rotation around the three axes, a cut-off angle is defined to guarantee that the approximate minimum angle is found after a limited number of iterations (11 iterations). The preprocessing ensures the $3 main invariances (*i.e.*, position, scale, rotation). For each training gesture, a score is computed against the candidate gesture by adapting the equation used by Wobbrock in $1 to three dimensions, and the result is a score table grouping the scores between gestures. Finally, a heuristic is applied to reduce false positives, compared to the strategy of $1, which consists of selecting the highest matching score as the recognized gesture.

2. **The device.** Some limitations of the $3 recognizer are that it is only compatible with segmented gestures. Moreover, the imperfection of the WiiRemote device must be added to the limitations of the acceleration data, considered as lower quality data subject to noise, and to the drift generated by the integration of the gesture path.

3. **The gestures.** The authors introduced a new representation of the raw data acceleration without modifying or preprocessing them. They defined a *gesture trace* which is the total sum of the previous *acceleration deltas* (*i.e.*, the current change in acceleration obtained by subtracting the current acceleration value from the previous one). There is also the complexity of gestures that are not simple 2D shapes but also "true" 3D gestures.

Rapid Prototyping Criteria

This recognizer extends the $1 [140], a touchscreen-based 2D recognizer that follows the same motivation for rapid prototyping, which involves minimal parameter adjustment and minimal training and is based on simple geometric and trigonometric calculations.

Performance

The $3 recognizer evaluation gave a recognition rate of 80% with 10 gesture classes with 12 participants. The recognizer was trained with 5 templates per

gesture class. Results between subjects varied between 58% and 98%. Although the results are considered inferior to those obtained by [103], they are acceptable in view of the limitations.

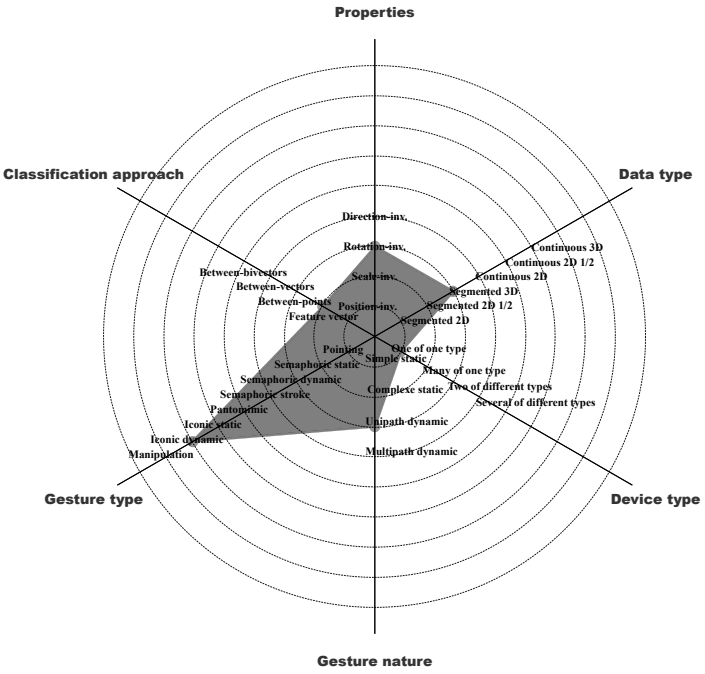


FIGURE 2.5: The space design instantiation for $3.

2.4.4 Hoffman's Linear Classifier (2010)

With the advent of convenient spatial input devices, the possibility of using 3D gesture recognition in virtual environments has become widespread, particularly in the field of video games. These devices detect motion using accelerometers and angular rate gyroscopes instead of spatial positions and orientations. For this reason, Hoffman *et al.* [46] used a linear classifier based on Rubine's algorithm [98].

Design Space Instantiation

1. **The recognizer.** The algorithm extracts a 29 features vector for the WiiRemote (*i.e.*, acceleration data) and 41 features for the WiiRemote coupled

with the WiiMotionPlus (*i.e.*, acceleration data + angular velocity) from the candidate gesture. Both feature sets are extended from Rubine's feature set designed for 2D gestures, considering the acceleration and angular velocity data as position information in 3D space. This feature vector is used for classification via a linear machine similar to the Rubine algorithm and also used in Rubine3D, and the algorithm outputs the gesture class assigned to the candidate.

2. **The device.** The evaluation examined the performance of 3D gesture recognition using the WiiRemote, which is a common acceleration-based input device combined with the angular velocity-based WiiMotionPlus device.
3. **The gestures.** Classification accuracy was defined as the metric to be evaluated. A gesture database composed of 25 gestures was created, and 17 participants performed 8500 samples. The data generated by these devices can suffer from numerous cumulative errors caused by drift, lack of a reference frame, and impact of motion rate on gesture profile. These issues make the task of 3D gesture recognition with these devices even more difficult.

Rapid Prototyping Criteria

As with the Rubine3D recognizer, the classifier relies on a set of 29 and 41 features, each with a meaning; the classifier needs to be trained on a set of gestures with many samples.

Performance

They explored the effect of different variables such as the number of gestures, the size of the training set, and the evaluation scenario's impact (*i.e.*, user-dependent or user-independent) on the recognition accuracy in comparison with another machine learning algorithm, the AdaBoost classifier, both of which were tested in user-dependent and user-independent scenarios using the WiiRemote and the pairing of the WiiRemote and WiiMotionPlus. The evaluation of the Linear classifier indicates that its overall accuracy outperforms that of the AdaBoost classifier with a rate over 99% in the user-dependent scenario for the coupled sensors. At the same time, it reaches 98% for the user-independent scenario.

2.4.5 Protractor3D (2011)

Gesture recognition based on the template-matching approach is usually performed on a per-user basis by matching the user's gestures entered with the

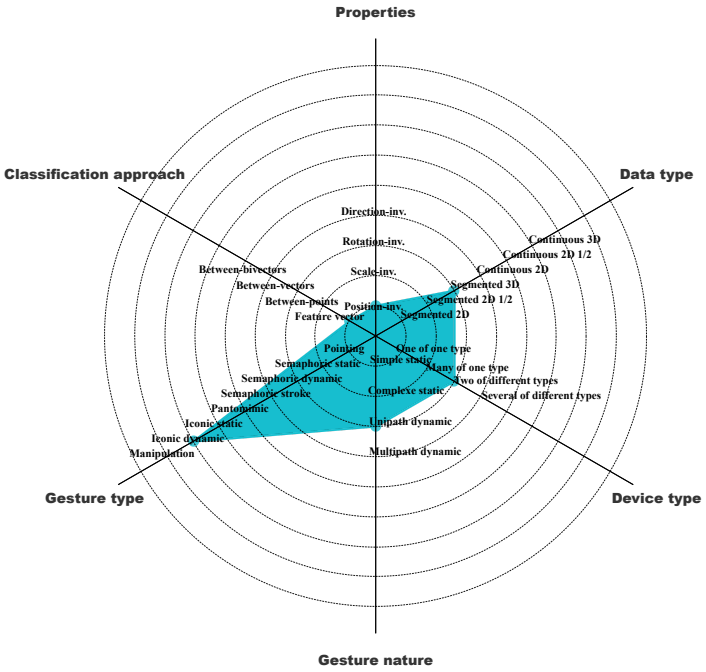


FIGURE 2.6: The space design instantiation for Hoffman's linear classifier.

previously recorded templates. One of the main problems with recognition techniques is that the rotated input gesture cannot be matched accurately with training templates that were captured with a different posture of the device, which is referred to as the template-gesture rotation problem. In order to solve this problem [57] developed *Protractor3D*, a gesture recognition algorithm based on the closed-form solution of the absolute orientation problem for measurements in two 3D coordinate systems.

Design Space Instantiation

1. **The recognizer.** The Protractor3D algorithm starts with a preprocessing of the gestures. First, the algorithm subsamples each input gesture to a fixed number of points (n=32). After that, it scales it to fit into a cube of a fixed side length. Lastly, it centers the gesture around the origin of the

coordinate system. The input gesture is compared to every saved gesture template using a rotation correction to solve the gesture-template rotation problem. The recognizer uses Horn's technique [48], which can be summarised as the idea of finding the rotation R that minimizes the sum of squares error between the input gesture and the template entered by the user, which is tantamount to maximize the dot product of the template and the rotated input gesture. Moreover, the technique uses Quaternions to find the optimal rotation, and the absolute rotation angle between the gestures decides whether the gesture template should be rejected as a possible candidate for recognition by setting a cut-off angle. The recognizer declares the recognized gesture class as the gesture class of the template with the smallest Euclidean distance to the entered gesture.

2. **The device.** The gestures were recorded using an accelerometer sensor, which let users record delimited gestures by holding a button.

3. **The gestures.** The authors evaluated the Protractor3D With 11 gestures recorded by ten subjects and a training set of 5 samples per gesture class.

Rapid Prototyping Criteria

As for $1 and $3 the protractor 3D is based on simple geometric notions, it doesn't require many samples to achieve great results.

Performance

The algorithm's accuracy varied between 83.3% and 98.9% for the different gesture classes. Finally, an evaluation showed the effect of rotational correction; Protractor3D has the same accuracy under different rotations compared to the unrotated matching using a mean square error, thus proving its rotation invariance.

2.4.6 $P3D (2016)

To achieve the objective of determining when a user performs a gesture, based on discrete and continuous gestures, to provide the user with a range of possibilities. Moreover, hand gesture interaction needs to be simple enough to be understood but distinct enough for the user to understand which gesture has been made. All this requires developing a customized gesture recognition system with the idea of achieving a maximum recognition rate while using a small dataset.

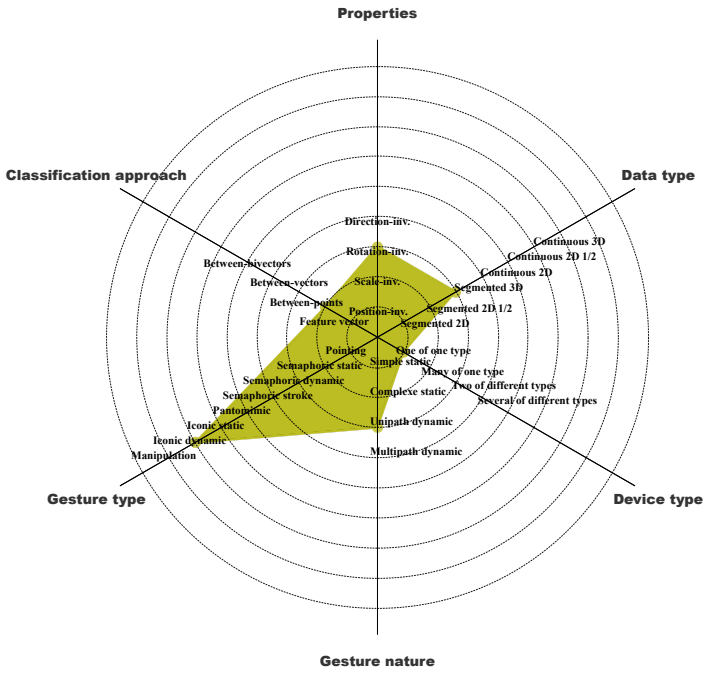


FIGURE 2.7: The space design instantiation for Protractor3D.

Design Space Instantiation

1. **The recognizer.** The main algorithm,$P3D, is a 3D extension of the point-cloud based 2D recognizer $P [124]. The $P3D manages both 2D and 3D gesture recognition, so the algorithm changes the creation method to gather and assign points based on the data (*i.e.*, it takes the Z position of the set of points where it is appropriate). For preprocessing, the data points are first resampled with a fixed number of points (N=32) as in Protractor3D [57] using linear interpolation. Then, the algorithm translates the centroid of the data points at the origin of the coordinate system. In addition, the authors implemented $P3D+, an improved version of the $P3D recognizer, whose effectiveness depends on data. Its main improvements are (1) the addition of a set of classifiers to limit the complexity of searching for all gestures according to three criteria: a) the dimension of the space (2D or 3D gesture); b) the type of gesture (pose or dynamic gesture); c) the handedness (gesture executed with the right or left hand). (2) Another improvement was to assign and order stroke IDs to the hand

so that each joint from the palm to the fingertips has its stroke ID and order, in order to avoid very flexible cloud matching and to reduce inaccuracies (*i.e.*, to prevent matching of points from different strokes). $P3D is a 3-dimensional gesture recognizer based on the $P, which is position-, scale-, and direction-invariant, but it is sensitive to rotation, which means that the gesture is not rotated when comparing point clouds. The authors compensated for this problem by adding gestures with different angles to the training set.

2. **The device.** Therefore, [31] has developed a system that includes new gesture algorithms to provide effective natural interaction at various distances and screen resolutions using the Leap Motion controller device.
3. **The gestures.** The training dataset consists of 15 gestures using two sample databases. The first database contains all gestures (*i.e.*, Iconic shapes, 2D shapes, 3D poses and 3D gestures); it also includes many variations for each gesture. The second database focuses only on 3D gestures and poses and expands them with many variations but excludes 2D gestures. The developed system is expected to be fast and adapted to multi-stroke gesture recognition, which could handle multiple start and stop points while ignoring the stroke order. For the evaluation scoring method, the scores were calculated using a polynomial formula and mapping the distance to defined percentage values.

Rapid Prototyping Criteria

The recognizer is easy to implement and understand; in particular, it extends the $\$P$ recognizer. However, the evaluation scoring method introduced in this recognizer is complicated to calculate. The recognizer achieves good results by requiring only a small number of templates per gesture.

Performance

The evaluation showed numerous results for both datasets, mainly demonstrating the superiority of the $P3D over the 2D recognizer $P, with a recognition rate for the first dataset of 95% ($P3D) versus 91% ($P). For the second dataset, the recognition rates were great for the $P3D for 3D gestures and the $P for the 3D poses. The evaluation results between $P3D and its enhanced version $P3D+ showed a slight improvement for the latter.

2.4.7 3 Cent (2017)

Low-cost devices can provide pre-processed input data represented as 3D trajectories, which permit the reduction of the recognition task to a geometric

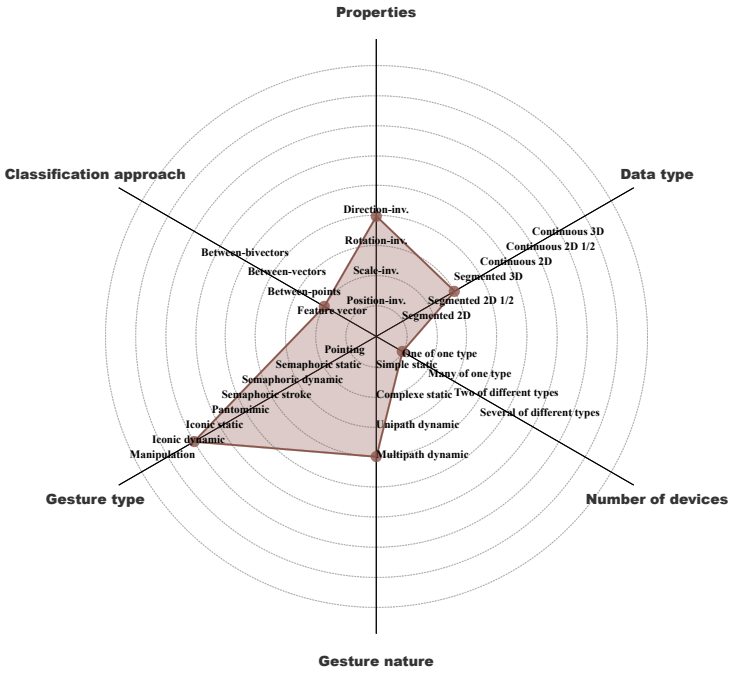


FIGURE 2.8: The space design instantiation for $P3D.

problem. Also, this can provide tools to design custom gesture interfaces using a small amount of data, similar to popular 2D stroke recognizers such as $1 [140]. [23] designed a 3D gesture recognizer based solely on single 3D trajectories using sampled trajectories comparison but adopting different and simpler processing than the 3D adaptations of $1 (*i.e.*, $3 and Protractor3D).

Design Space Instantiation

1. **The recognizer.** The $3Cent$ recognizer processes segmented gestures and first applies a three-step preprocessing: (1) resampling the gesture into a fixed number of equidistant points; (2) scaling the gesture uniformly to a standard unit length; (3) translating the centroid of the point cloud to the origin, (*i.e.*, (0,0,0)). The applied preprocessing ensures the position and scale invariance properties, but neither in rotation nor direction. To be recognized, each input gesture must be performed in the same way as

the template gestures, with fixed direction and orientation, or all the variations must be represented in the training set. The recognizer calculates the distance between this gesture and each template as the sum of the squared distances of the corresponding points. It returns the template with the shortest distance.

2. **The device.** To evaluate the recognizer, the authors used two datasets of segmented gestures, using a Leap Motion controller device. The second dataset is from the Shrec'17 track [36] recorded with an Intel RealSense Depth.

3. **The gestures.** The first dataset is composed of 26 gestures representing command interface gestures and are 3D trajectories of the dominant forefinger performed by 14 subjects. While the second dataset consists of coarse gestures (*i.e.*, not characterized by finger movements) and fine gestures (*i.e.*, characterized by finger movements).

Rapid Prototyping Criteria

The $3Cent$ recognizer is a rapid prototyping recognizer, particularly it extends the $\$1$ recognizer, it is easy to understand and doesn't require a lot of samples to train the recognizer and to achieve great results.

Performance

The evaluation was done using a K-nn classification (k=1,3,5) with the gestures of 5 users selected as a training set and tested with the gestures of the eight remaining users. For Shrec'17, the evaluation was separated into two tests. The first considered only the coarse gestures in the dataset, characterized by the palm trajectory. Then, the second dataset was composed of fine gestures. The results showed that the $3Cent$ recognizer without rotation performed best for the different tests; the classification accuracy of $3Cent$ on the 26 classes dataset ranged from 95.2% to 96.9%, while it varied less, ranging from 90.2% to 92% for the coarse gestures in the Shrec'17 dataset. Furthermore, the $3Cent$ applied to the index finger for the fine gestures dataset reached almost 78%.

2.4.8 JackKnife (2017)

To solve the problem of designing a general rapid prototyping recognizer for dynamic gestures that can be trained with few samples, that not only works with segmented data but can also work with continuous data; and is modality-agnostic. [116] created Jackknife, a multitool recognizer composed of a number of techniques to achieve high recognition rates through different modalities.

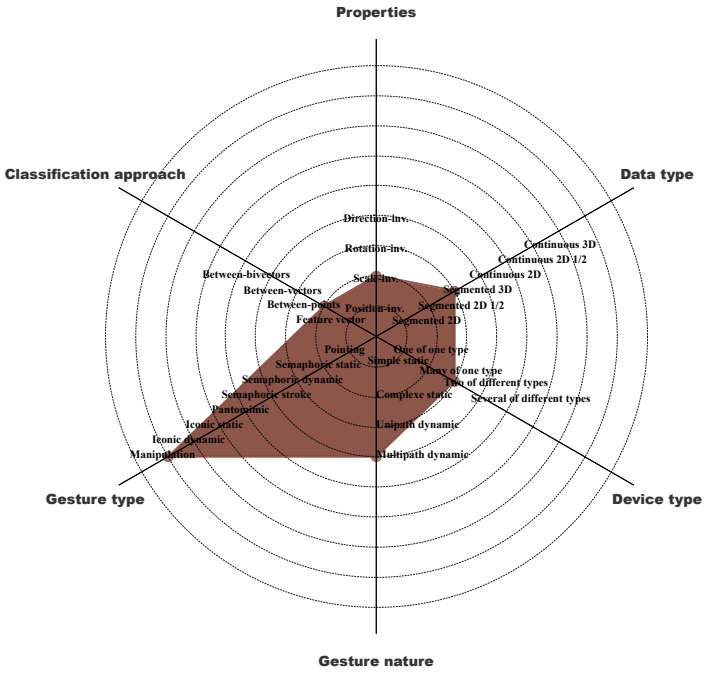


FIGURE 2.9: The space design instantiation for 3 Cent.

Design Space Instantiation

1. **The recognizer.** The Jackknife recognition system relies on dynamic time warping (DTW) as a measure of dissimilarity based on the inner product of the direction vectors of the gesture trajectory, combined with a simple 1-NN pattern matching recognition algorithm. Besides, they apply a correction factor to the DTW to disambiguate some gestures and separate negative from positive samples. They implemented a method that selects a rejection threshold using one training sample per class for continuous gestures. The threshold is determined by learning a per-template rejection threshold based on synthetically generated samples from the user gesture samples. Synthetic data are generated to obtain a sufficient number of training data to construct the distribution of scored negative (*i.e.*, non-gesture) and positive (*i.e.*, gesture) synthetic samples. The evaluation was done on several rejection thresholds for continuous gestures using a Kinect and a Leap Motion. The gesture classification required segmentation using a sliding window segmenter. The excellent results

for the different parameters tested suggest that JackKnife is suitable for continuous gestures. The technique is neither direction- nor rotation-invariant, and the sampling rate for synthetic gestures must be defined manually. It showed that the inner product is a better local cost function than the squared euclidean distance in almost all evaluations.

2. **The device.** The recognizer was evaluated on different modalities using different sensor datasets for segmented and continuous gestures. The evaluation of JackKnife with gestures made with a WiiRemote was done in a user-dependent scenario, and the data were integrated to obtain 3D position points.

3. **The gestures.** The authors consider the processed gestures as a temporal series of trajectories in a multidimensional space (*i.e.*, an ordered set of points, and represented as a set of direction vectors of unit length). Also, a limitation of JackKnife is that it does not consider static poses. The authors point out that the recognizer achieved competitive results in 2D and 3D gesture recognition and is a robust gesture recognizer.

Rapid Prototyping Criteria

For the evaluation of segmented gestures, the recognizer was evaluated on many public datasets using only one or two samples as the training set. While the Jackknife recognizer needs to be trained each time a new gesture is added to the training set, it also relies on some complex techniques to improve its recognition rate.

Performance

In terms of accuracy for gestures produced with a WiiRemote, JackKnife achieved high results compared to other popular recognizers (*i.e.*, $3 and Protractor3D), but its performance was lower than the uWave recognizer (*i.e.*, DTW with quantization). The evaluation of Kinect gestures also showed great results with one sample in user-dependent and user-independent scenarios (*i.e.*, 99% and 96% respectively).

2.4.9 DeepGRU (2019)

Recognition methods that can handle multiple devices, and, in particular, those that can effectively capture and distinguish between different gestures, are needed. At the same time, rapid prototyping requires that these methods can be easily integrated into the recognition process, even though they are generally more complex and less flexible to work with different devices. Therefore,

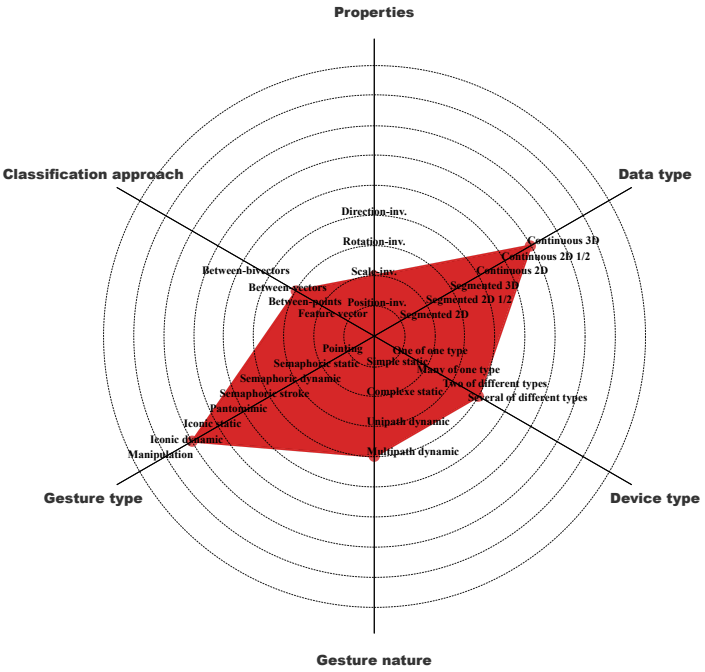


FIGURE 2.10: The space design instantiation for JackKnife.

Maghoumi *et al.* [72] introduced a general gesture recognition utility relying on a deep learning model that takes as input different types of data.

Design Space Instantiation

1. **The recognizer.** The model relies on gate recurrent units (GRU) as the principal block to build the network. The latter is composed of three parts: the encoder network, which compresses the input gesture sample; the attention module to refine the set of features; and finally, the two layers that classify the gesture based on the probability distribution of the class labels. Numerous experiments were conducted to evaluate the DeepGRU and measure its recognition accuracy with different segmented gesture datasets of various sizes that cover many configurations (*e.g.,* gesture interactions, number of actors, view-point variations and input devices.). The model supports invariances in a limited way by creating synthetic samples of real samples. The generated samples are

characterized by limited variations (position, scale, rotation) to train the model on them.

2. **The device.** Two small datasets of segmented gestures were tested to show the usefulness of the utility for a limited amount of data in rapid prototyping with two hand gesture datasets, one containing acoustic data and the other containing the Wii Remote data. The DeepGRU was evaluated with 2 and 4 training samples per gesture class, respectively, in a user-dependent scenario.

3. **The gestures.** The raw data provided to the model is heterogeneous as a temporal sequence, raw 3D skeleton composed of multiple articulations, pose or other vector features (*e.g.* acceleration, angular velocity, etc) and doesn't depend on a specific device, they were tested with existing datasets [25, 90] composed of dynamic semaphoric and iconic gestures.

Rapid Prototyping Criteria

The authors assert that DeepGRU is a compact network achieving excellent results, comparable to state-of-the-art recognition accuracy on both small and large datasets. Its suitability for rapid prototyping is particularly noteworthy, given the impressive results with just four training samples per class and a reasonable training time using only the CPU. However, the code model provided is difficult to adapt and seems rather non-obvious. Despite the authors' claims about its suitability for rapid prototyping, understanding and implementing the model does require prior knowledge of the deep learning field for developers and practitioners.

Performance

The evaluation results on small datasets showed that the DeepGRU performed well (*i.e.*, $> 90\%$), and showed a higher accuracy compared to other existing recognizers (*e.g.,* Protractor3D [57];$3 [58] and Jackknife for 4 training samples) this results concern the Wii Remote dataset, while Jackknife [116]) was better for 2 training samples. The model training times were small compared to larger datasets, but still quite high (*i.e.*, 6.9 minutes with a 12-thread CPU device for the Wii Remote dataset with 4 samples per gesture class).

2.5 Comparative Analysis

The Figure 2.12 gives an overview of the instantiation of all the selected recognizers that meet or are very close to the rapid-prototyping criteria, the radar

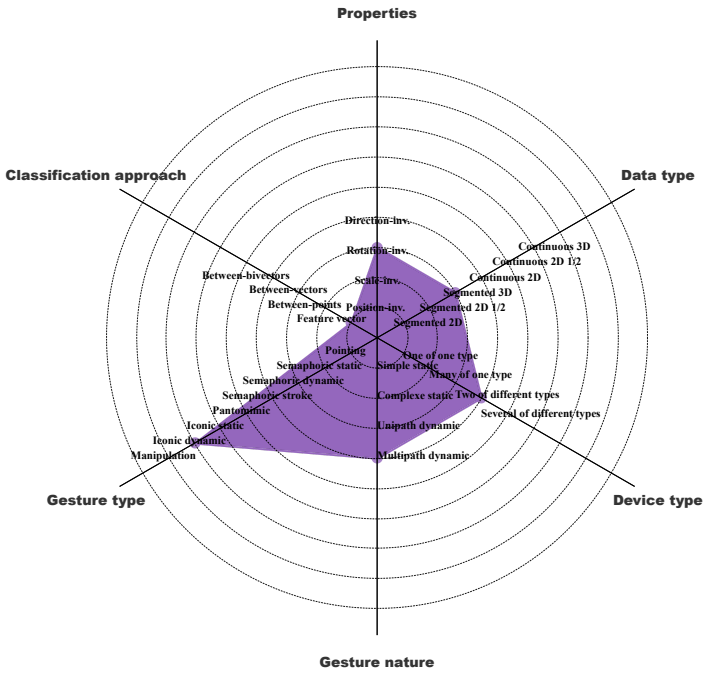


FIGURE 2.11: The space design instantiation for DeepGRU.

diagram shows interesting features of the 3D gesture recognizers described in the subsection 2.2.3. At the same time, they show the area covered by these different recognition systems, which lets us distinguish the similarities and differences between recognizers. Not in order to designate the best recognizer, but to display the usefulness of the representation of design space in the comparative aspect between recognizers along different features.

- The **DeepGRU** and **JackKnife** recognizers cover more complex values on different axes, in particular along the "Gesture Nature" and the "Number of devices".
- The "uWave" and "Rubine3D" cover the same area on almost all dimensions except for the classification approach where "uWave" cover a more complex value than Rubin3D.
- The "Rubine3D" and "Hoffman's classifier" also have the same characteristics on all dimensions except for the number of devices where the

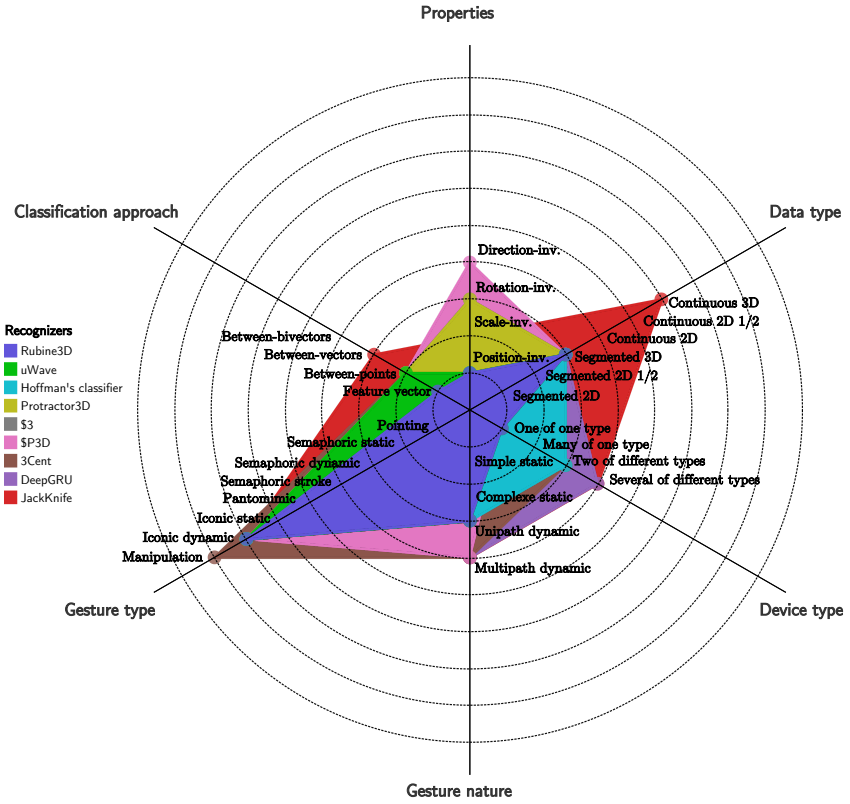


FIGURE 2.12: The space design instantiation for all the selected recognizers.

later one cover an advanced value as an evaluation of the recognizer with "Two devices of different types".

- the "$3" and "Protractor3D" covers the same design space area.

2.6 Generative Analysis

The above analysis that resulted in the creation of the figure illustrating the design space of all recognizers indicates that certain dimensions are adequately covered. This analysis will summarize and highlight the main features addressed by 3D hand gesture recognizers adapted to rapid prototyping, which

mainly use the template-based technique. The third level of analysis, known as generative analysis, helps create diverse and enriched design spaces for innovative solutions. This level is particularly useful in fostering new ideas as opposed to criticism [13]. It also highlights unexplored areas that could improve the design of future gesture recognizers for rapid prototyping.

2.6.1 Classification Approaches

Several recognizers, particularly those belonging to the $-family, have employed the between-point approach to assess the similarity between gestures. The parametric approach was used by Rubine3D and Hoffman's classifier, which are adaptations of the classifier originally developed by Rubine for stroke gestures [98].

Between-point classification is the most commonly used approach, while parametric classification has been tested in Rubine3D and Hoffman's classifier using accelerations and angular velocities. The between-vector approach has only been used in Jackknife, and between-bivector approaches have not been utilized in rapid-prototyping 3D gesture recognizers. It would be valuable to study these approaches to demonstrate their advantages and highlight their effects on the 3D gesture recognition process.

2.6.2 Invariance Properties

The invariance properties of rapid prototyping recognizers vary between different recognizers. All of the recognizers studied have position invariance in common. The scale-invariance property is also widespread, except for the uWave and Hoffman classifier. On the other hand, rotation-invariance is not very common and is only found in 3D extensions of the $-family recognizer $1, as well as the two recognizers $3 and Protractor 3D. The same applies to direction-invariance, where only the $P3D recognizer has preserved it through the implementation of the cloud of points.

- First, all the recognizers provide "position-invariance", which means that a recognizer is independent of the place where a gesture is performed in the air (*e.g.*, a swipe left at the head is equivalent to a swipe left at the chest).
- The "scale-invariance" property describes the recognizer's independence from the scale and the size of gestures (*e.g.*, a large gesture performed is equivalent to a smaller one). All the recognizers share this property except the uWave [69] and Hoffman's classifier [46].

- Then, the property of "rotation-invariance" refers to the independence of a recognizer towards the rotation of a gesture (*e.g.*, if we consider the cardinal points as a reference, a smash to the north-east cannot be distinguished from a smash gesture to the north-west). The recognizers that support this rotation-invariance property are $3 and Protractor3D.
- Finally, $P3D [31] provides "direction-invariance" by using clouds of points to recognize a gesture. It refers to the independence of a recognizer towards the direction where the gestures are articulated.

The invariance properties of 3D gesture recognizers vary greatly. These properties depend mainly on the characteristics of the 2D recognizers they are inspired by and the requirements these recognizers need to fulfill. Most 3D rapid prototyping gesture recognizers maintain strict scale-, position-invariance but only a few are rotation-, direction-invariant.

2.6.3 Data Type

The JackKnife recognizer [116] demonstrated the importance of having a recognizer that can recognize gestures using different types of devices. However, this recent recognizer has only focused on the uni-device recognition scenario, by using segmented and continuous gestures (*i.e.*, recognition accuracy $> 90\%$).

- The recognizers primarily processed segmented gestures, and many of them used pre-segmented gestures, while the devices provided continuous data. The JackKnife recognizer demonstrated impressive performance when evaluated on both segmented and continuous gestures.
- Among the mentioned recognizers, most of them classified various classes of 3D gestures, with a focus on segmented gestures ranging from uWave to DeepGRU. It is worth noting that very few recognizers have successfully recognized 2D gestures [31, 116].

2.6.4 Gesture Type

The most common type of gesture is the dynamic iconic gesture, which represents a shape or path. The second most common type is the stroke semaphoric gesture. However, we observe that the other types are poorly integrated and could benefit from further study. Furthermore, based on *Aigner's taxonomy* [3], all the recognizers described were used to classify mainly iconic dynamic gestures, followed by the semaphoric stroke gestures.

The latest studied 3D gesture recognizers primarily classify dynamic iconic gestures and stroke semaphoric gestures.

2.6.5 Gesture Nature

Based on the comparative analysis, it is evident that "unipath dynamic" gestures have been extensively studied in the analysis of recognizers. However, "multipath dynamic" gestures have not been thoroughly investigated, and static gestures have not been widely used.

Furthermore, unipath dynamic gestures, which can be represented as a set of measurements, are the most recognized gestures. However, only a small number of recognizers, namely [23, 31, 116], have explored the multipath dynamic gestures [98] (*i.e.*, movements where each finger has its trajectory). These gestures are more interesting and convey more detail. However, the lack of recognition of the static gesture is remarkable, except for $\$P3D$, which was tested with this type of gesture.

The latest 3D gesture recognizers studied can classify both unipath and multipath dynamic gestures.

2.6.6 Device Type

The evaluation of template-based 3D gesture recognizers has been conducted using different types and numbers of devices:

- First, as we have described, there are many types of devices that provide gesture data. We can study the impact of these different types on the recognizer.
- All the recognizers were primarily evaluated using 3D devices. They provided measurements of various physical quantities, including position in Cartesian space, accelerations, and angular velocities. However, the $P3D and the JackKnife recognizers were also assessed using 2D data.
- The number of devices used in the evaluation varies depending on the recognizer. Most recognizers that track only the hand use a single type of device. However, Hoffman's classifier uses two different types of devices that provide acceleration and angular velocity values. The $3Cent$ recognizer is trained on multiple devices of the same type. Finally, in JackKnife, the recognizer is trained on data from several different types of devices, including both hand and full-body gestures.

- Finally, we can introduce multi-device testing, which involves training the recognizer with examples from one device and testing it with candidate gestures produced on another device. It is well detailed in the Chapter 3. This additional testing configuration expands the number of testing scenarios that can be considered when evaluating a recognizer. We can now distinguish four basic configurations: (1) user-dependent uni-device, (2) user-dependent uni-device, (3) user-independent multi-device, and (4) user-independent multi-device.

The template-based gesture recognizers, adapted to rapid prototyping, have been extensively evaluated with one type of 3D device for hand gestures using segmented data. However, some recognizers have also been evaluated with different types of devices using continuous data.

2.7 Motivations for Multi-context gesture recognizers

The three-level analysis in this chapter demonstrated the necessity of investigating diverse dimensions of the defined design space for hand gesture recognizers. It emphasized the importance of exploring multi-context 3D hand gesture recognizers. The focus should be on those using template-based recognition algorithms that are adapted to rapid prototyping.

This work aims to explore various aspects that have not been extensively researched. Our primary focus will be on two dimensions: the "Classification" and the "Device". The "Classification approach" feature has been underexplored because it currently focuses mainly on one approach per recognizer. By exploring new values for "Classification Approach" and comparing them, we can explore the use of recognizers in multiple contexts of use, such as with multiple devices. This exploration also necessitates new testing scenarios that specifically focus on device independence and the ability to test gestures.

Chapter 3

Concepts for Comparative Testing of 3D Hand Gestures in Multiple Contexts of Use

Developing multi-context comparative testing requires a thorough understanding of the subject matter. Important concepts related to the subject matter must be defined and explained in detail to achieve this. Moreover, it requires a specific analysis of state-of-the-art works, as it was accomplished in chapter 2, to gain valuable insights and knowledge. This chapter examines the results of the analysis to identify the main principles for creating an effective comparative testing systematic procedure for hand gesture recognizers. It also describes a model that outlines the main stages for single gesture recognition testing, including Gesture Elicitation Study (GES), data acquisition, training, and testing. This approach underscores the importance of structuring the multi-context gesture recognition approach with well-defined components. At the end of the chapter, we describe the main elements of comparative testing, and we denote the systematic nature of the procedure, ensuring a seamless and efficient procedure that enhances the overall reliability and quality of the comparative testing.

3.1 Terminology

We define a set of fundamental terms as follows:

Sensor: is a device that converts a physical signal into an electrical signal that can be processed by a system [138].

Device: is a physical object that enables interaction between a user and a system. A device can be an input or output device [9].

Input Device: is a device that provides human actions (outputs) to a machine controlled or manipulated by a user.

Motion Input Device: is a device that includes at least a motion sensor and transfers raw motion data from a sensor to a system.

Device Family: is a set of input devices that share the same values for a predefined set of properties, such as:

- *Sensing*: is the ability of the device to sense a type of signal input sent to the system.
- *Processing*: is the ability of the system to execute different stages and methods to transform the input raw signal from the device into a digital signal.
- *Configurability*: is the ability of the device to be configured by user input
- *Programmability*: is the ability of the device to accept a new set of instructions that change how it operates.

Input Device Category: is a collection of different device families grouped according to one or more common criteria defined for this purpose, which differ from the device family criteria.

3.2 Definitions

First, we define the prefixes used in this terminology:

- *Intra-* means "within" and indicates that the elements used in gesture recognition belong to the same class.
- *Inter-* means "between" and indicates that the elements used in gesture recognition belong to different classes.
- *Cross-* means "extending over" and indicates that the elements used in gesture recognition combine elements of different classes.
- *Trans-* means "across" and indicates that the elements used in gesture recognition belong to different classes for each stage as defined above.

- *Uni-* means "One" and indicates that the elements used in gesture recognition belong to a single class.
- *Multi-* means "Multiple" and indicates that the elements used in gesture recognition belong to multiple classes.

The following definitions refer to the terms that indicate the configurations of the devices used during the gesture recognition process and are illustrated in Figure 3.1. In the context of this thesis, the gesture recognition process includes two stages common to all machine learning processes: training and testing, which were defined above [55].

- **Inter-input device category** means that the devices used in the gesture recognition process belong to different input device categories. Many examples follow this configuration.
- **Intra-input device category** means that the devices used in the gesture recognition process belong to the same input device category.
- **Cross-input device category** means that the devices used in the gesture recognition process are a combination of two devices from different input device categories for each stage.
- **Trans-input device category** means that the devices used in the gesture recognition process belong to different input device categories for each stage.
- **Uni-Category** means that the devices used in the gesture recognition process belong to one input device category.
- **Multi-Category** means that the devices used in the gesture recognition process belong to multiple input device categories.
- **Inter-input device family** means that the devices used in the gesture recognition process belong to the same input device category but to different input device families.
- **Intra-input device family** means that the devices used in the gesture recognition process belong to the same input device category and the same input device family.
- **Cross-input device family** means that the devices used in the gesture recognition process are a combination of two devices from the same input device category but from different input device families for each stage.
- **Trans-input device family** means that the devices used in the gesture recognition process belong to the same input device category but to different input device families for each stage.

- **Uni-Family** means that the devices used in the gesture recognition process belong to one input device category and to the same input device family.
- **Multi-Family** means that the devices used in the gesture recognition process belong to one input device category and to the different input device families.
- **Inter-device** means that the devices used in the gesture recognition process are different but belong to the same input device category and input device family.
- **Intra-device** means that the devices used in the gesture recognition process are exactly the same and belong to the same category and family of input devices.
- **Cross-device** means that the devices used in the gesture recognition process are a combination of two devices from the same category and family of input devices for each stage.
- **Trans-device** means that different devices are used in the gesture recognition process, but belong to the same category and family of input devices for each stage.
- **Uni-device** means that there is only one device used in the gesture recognition process.
- **Multi-device** means that the devices used in the gesture recognition process are different [9].

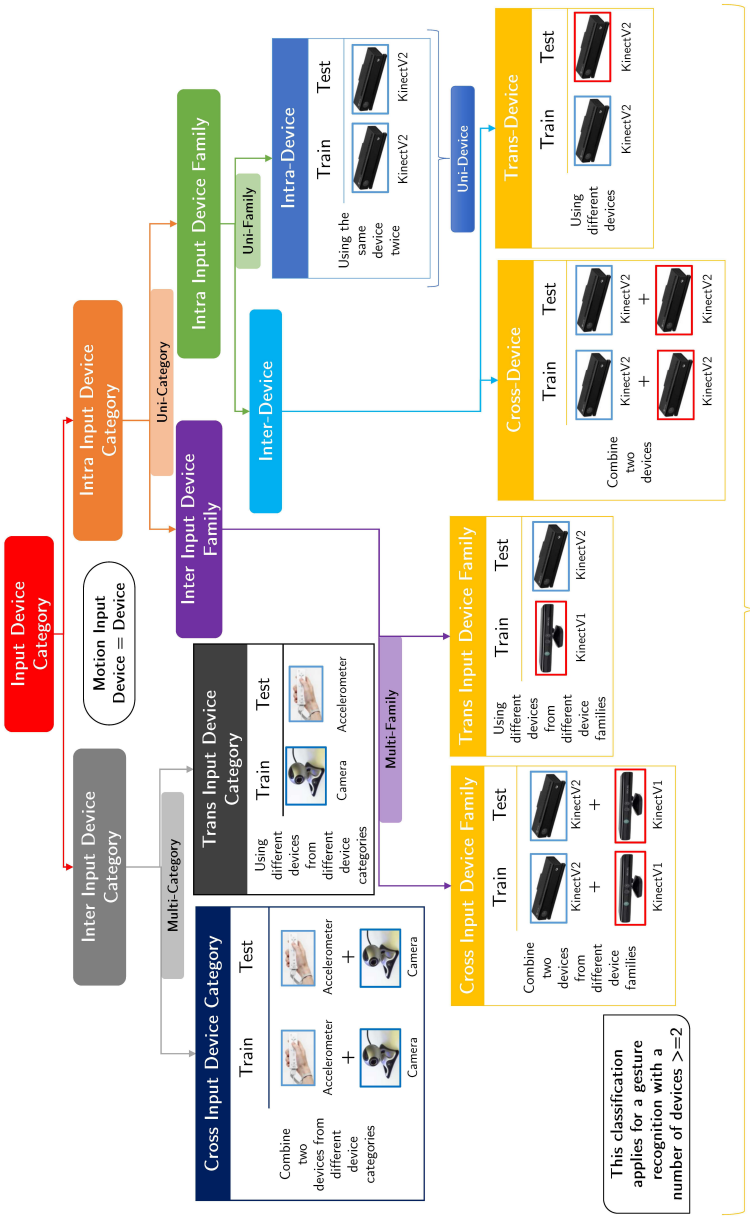


FIGURE 3.1: Taxonomy of configurations for input devices used in the gesture recognition process.

Based on the classification described above, we can provide an example that expands Buxton's taxonomy of input devices [20] by including the terminology defined in this chapter. We will introduce two examples. The first example defines two criteria for the category. As a result, it uses a motion mechanical device for training and a motion sensing device for testing. This describes an inter-device category relation for the recognition process. The second example shown is using a ring zero for training and the Wii Remote for testing, which represents a trans-device family relation. These examples are shown in the following extended Buxton's taxonomy in Figure 3.2.

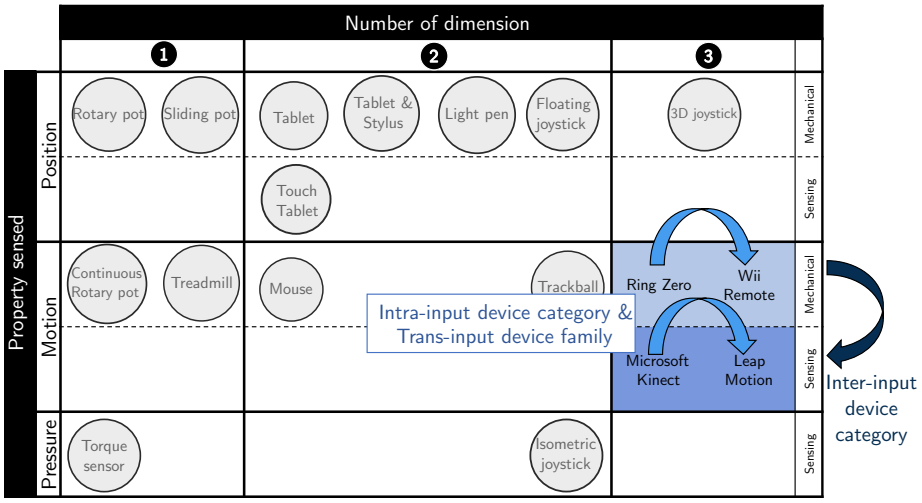
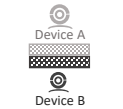
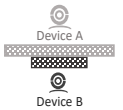
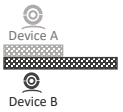
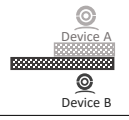


FIGURE 3.2: Taxonomy of input devices from [20] extended.

3.3 Temporal Relationships in the "Cross" Configuration

The aforementioned "cross" configurations require the use of two different devices at different stages of the gesture recognition process. The recording of the various possible relationships described by Allen's interval algebra are grouped into two relational properties: the sequentiality and the simultaneity of the gesture as it is captured by the devices as it is shown in Table 3.1, where G_{DevA} and G_{DevB} represent the gesture recorded by device A and device B, respectively.

TABLE 3.1: Allen's interval algebra relations in "cross-device" configuration.

Allen Statements	Visual Example	Chronological Sequence
G_{DevA} before G_{DevB}		$Start(G_{DevA}) < End(G_{DevA}) < Start(G_{DevB}) < End(G_{DevB})$
G_{DevA} equals G_{DevB}		$Start(G_{DevA}) = Start(G_{DevB}) < End(G_{DevA}) = End(G_{DevB})$
G_{DevA} meets G_{DevB}		$Start(G_{DevA}) < End(G_{DevA}) = Start(G_{DevB}) < End(G_{DevB})$
G_{DevA} overlaps G_{DevB}		$Start(G_{DevA}) < Start(G_{DevB}) < End(G_{DevA}) < End(G_{DevB})$
G_{DevA} during G_{DevB}		$Start(G_{DevA}) < Start(G_{DevB}) < End(G_{DevB}) < End(G_{DevA})$
G_{DevA} starts G_{DevB}		$Start(G_{DevA}) = Start(G_{DevB}) < End(G_{DevA}) < End(G_{DevB})$
G_{DevA} finishes G_{DevB}		$Start(G_{DevB}) < Start(G_{DevA}) < End(G_{DevA}) = End(G_{DevB})$

3.4 Procedure for Single Gesture Recognition Testing

The gesture recognition testing procedure involves several stages: elicitation, data acquisition, training, and testing. These stages are commonly used in the literature for gesture recognition evaluation. However, some literature works include certain stages, while others are limited to others [65, 55, 141]. During the Gesture Elicitation Study stage, participants collaborate to create gestures for specific tasks that reflect the end user's preferences, as briefly mentioned in subsection 2.1.3.

The data acquisition stage involves capturing gesture data using various devices. The training stage is dedicated to developing the recognizer model, depending on the chosen classification technique. Finally, the test stage evaluates the performance of the trained model by measuring its ability to recognize new gestures. Similarly to ASPLe work [1] that provides systematic process support for the development of self-adaptive software systems with reuse.

We use the Software & Systems Process Engineering Meta-Model (SPEM) notation [85] (Figure 3.3) to illustrate the gesture recognition testing procedure.

As a specification of a Unified Modeling Language (UML) metamodel provided by the Object Management Group (OMG), SPEM is used to represent a wide range of software development processes and their components. Essentially, it functions as an "ontology" for software development processes, that provides a minimum set of process modeling elements that are applicable to any software development process. It does not impose specific models or constraints for particular fields such as software engineering, requirements engineering, or project management.

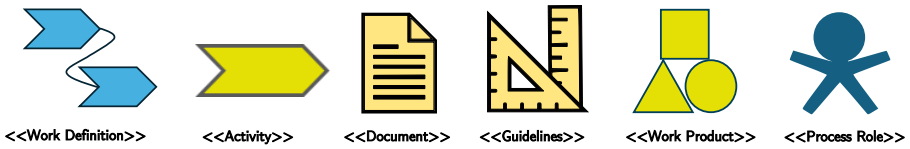


FIGURE 3.3: A subset of the SPEM notation relevant to this work.

- "Work Definition" refers to the specific type of operation that describes the work executed in a process.
- "Work Product" denotes any tangible information either produced, consumed, or altered by a process.
- "Process Role" outlines the responsibilities related to specific Work Products.
- "Activity" signifies a task performed by a single Process Role.
- "Document" is a specific category (a stereotype) of Work Product.
- "Guidelines" provides detailed information regarding a resource.

The roles considered in this procedure are related to experimental design [28]. The *Researcher/Experimenter* is responsible for creating and testing the gesture interaction system. Their goal is to obtain accurate results that either support or challenge a given theory about the product they've designed and developed. This includes conducting the Gesture Elicitation Study, acquiring data when necessary, and carrying out the training and testing of the gesture recognizer(s).

The participant role is responsible for providing data and feedback throughout the Gesture Elicitation Study and the Acquisition data stages, both by helping with collaboratively constructing the gesture set and providing and validating gestures to the recognizer(s) being tested.

The procedure includes four disciplines corresponding to the previously mentioned stages. This procedure is designed to support the different stages of hand gesture recognition evaluation. In the following sections, each stage and its interactions are described in detail as a workflow.

3.5 Workflow

The workflow that links all the disciplines together is shown in Figure 3.4. All *Work Products*, *Documents*, and *Guidelines* are also shown as either resources (if they are used in the discipline) or outcomes (if they are produced). We indicate only the most logical or frequently used shortcuts on the diagrams, rather than listing all possible shortcuts. The workflow shows that we can initiate the procedure from three of the four stages, namely: Gesture Elicitation Study, Data Acquisition, and Training. It's important to note that we can iterate on each stage and progress through them sequentially, the whole workflow forming a loop.

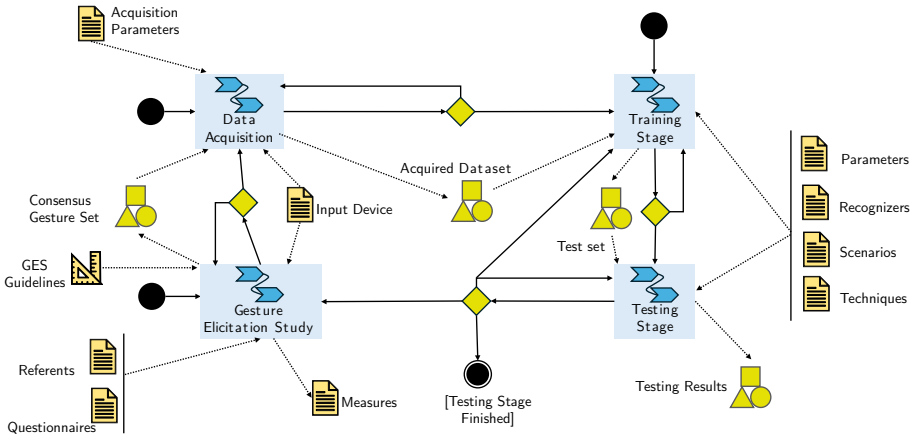


FIGURE 3.4: Gesture recognition testing procedure workflow.

3.5.1 Stage1: Gesture Elicitation Study

Gesture Elicitation Studies (GESs) are commonly used methods in HCI to design intuitive gesture commands for controlling interactive devices, applications, and systems [63, 100, 119, 139, 142]. These studies cover different contexts of use [128], spanning from classical devices to more modern ones [130, 131], and are thoroughly studied in the literature (see [129] for a systematic literature review).

According to Wobbrock, Morris, and Wilson [139], Magrofuoco and Vanderdonckt [75], and Villarreal-Narvaez et al. [129], the GES workflow is structured in several steps:

1. Define a study by determining the parameters and specifications for a GES. This includes defining the task set, referents, context of use, and data collection.
2. Conduct the study by conducting the GES according to the defined parameters. The result is a collection of elicited gestures and associated data.
3. Classify the gestures by categorizing them using a method such as clustering criteria. This organizes similar gestures together.
4. Measure the gestures by calculating various measurements using a measurement method. These measurements provide insight into the characteristics and properties of the gestures.
5. Discuss the gestures by interacting with stakeholders to reach a consensus. This ensures that the selected gestures are suitable for the intended use.
6. Export the gestures by converting the consensus data into a format compatible with gesture recognizers or modules that support gesture recognition. This enables effective use in practical applications.

Workflow The workflow in Figure 3.5 begins with the pre-test phase. In this phase, the *Researcher/Experimente*r defines the structure of the study, following the GES Guidelines. After this, the study is presented to the participant, who then completes questionnaires and undergoes tests. The *Researcher/Experimenter* then introduces the study apparatus, including the input device, and presents an example and the different referents for the study. For each referent, participants suggest gestures using the device, which are observed and recorded. Following the gesture proposal, participants provide feedback based on their experience. If unsatisfied, they can suggest alternative gestures. Afterward, they will take a post-test quiz in the post-test phase. The *Researcher/-Experimenter* then classifies the gestures, forms a consensus gesture set, and consolidates the results using this set as a reference.

3.5.2 Stage2: Data Acquisition

The stage of gesture acquisition is critical for evaluating gesture recognizers during the training and testing stages. Evaluators can develop effective methods by carefully collecting and analyzing datasets of gestures and choosing one

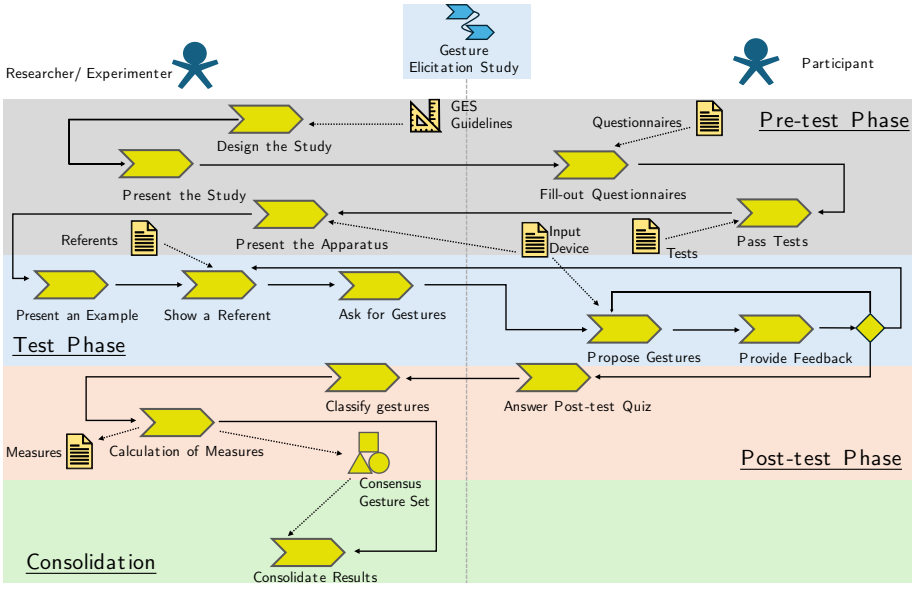


FIGURE 3.5: Activity diagram for Gesture Elicitation Study.

or more appropriate input devices to capture gesture data [25, 69, 141], such as electromyography (EMG) armbands, Time-of-Flight cameras, and inertial measurement units. Additionally, video recording is essential when collecting gestures, whatever the device used. It enables outliers to be detected and variances to be understood, improving data quality and reliability. Furthermore, they should consider different types of gestures. As mentioned earlier, conducting a study to elicit gestures can provide a dataset of gestures that define users' preferences. Other methods, such as challenges and contests, can also be used as datasets of gestures for evaluation [24, 36]. An evaluation dataset should ideally include at least two samples for each class of gesture, with one sample used for training and the other for testing purposes [46, 69]. Participants involved in the data collection process should perform two or more samples per gesture class, and these gesture classes could be augmented with synthetic data to ensure consistency and accuracy throughout the evaluation process [66, 81].

Workflow The data acquisition stage of gesture recognition involves several activities described in Figure 3.6. First, the *Researcher/Experimenter* designs the study, defining its scope and objectives. The *Acquisition Parameters* document is important for this activity. Next, the *Researcher/Experimenter* presents the

study to the participant, explaining its purpose, design, and expected results. They then introduce the input device that will be used to capture the gestures, detailing its specifications. The *Researcher/Experimenter* then presents examples of the gestures to be acquired, followed by a specific gesture class stimulus. Participants are asked to imitate these gestures using the input device. The next activities involve participants performing the gestures and providing feedback on their gestures. The *Researcher/Experimenter* collects these gesture samples for further analysis. Finally, the researcher consolidates the samples into a dataset and reviews it to ensure its integrity and usability.

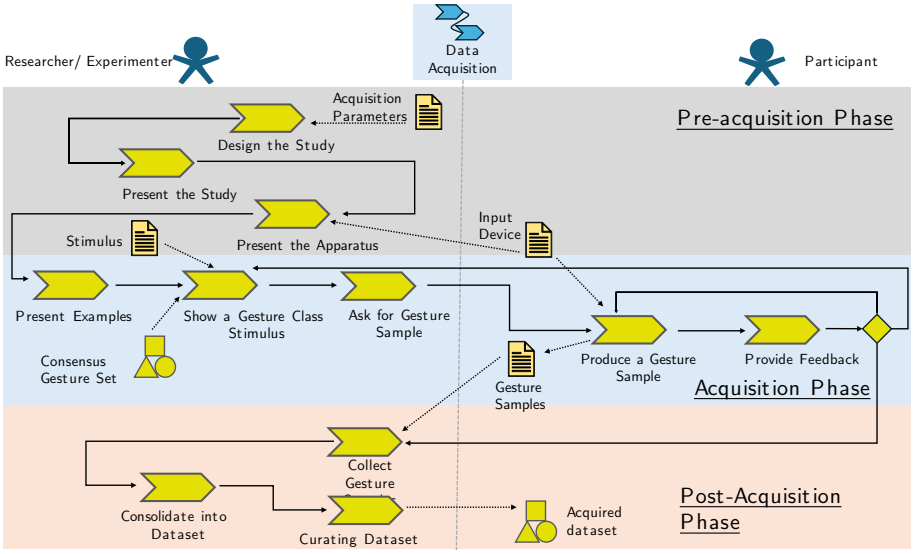


FIGURE 3.6: Activity diagram for Data acquisition.

3.5.3 Stage3: Training

The training stage of gesture recognition evaluation focuses on creating a gesture model [55, 141]. The process consists of several steps. First, a training data set is created from the collected gesture data. Then, a data pre-processing stage is applied to each sample of the gesture classes that compose the training set to remove noise and standardize the data. Finally, during the training stage, different training methods, such as feature extraction and normalization, are used based on the recognizer's classification technique. The training stage may have specific parameters, such as the number of templates for each class in the training set.

Workflow Developing an effective gesture recognizer involves several activities for the training stage shown in Figure3.7. Firstly, the *Researcher/Experimenter* designs a training model by setting the necessary parameters relying on the *Parameters* document. Following this, *Create Datasets* produces three work products (*Training Set, Testing Set, and Validation Set*). These are created using appropriate *Scenarios* and *Techniques* and may be segmented if required. Two of these datasets are then preprocessed (*Preprocess Datasets*) for the training phase. Afterward, the *Recognizer(s)* are trained on the *Training Set* produced work. Next, to validate, a gesture template is tested for validation (*Test a Gesture Template for Validation*) and the performance is evaluated on the *Validation Set*. Ultimately, the overall performance of the recognizer(s) is validated, resulting in recognizer(s) that can efficiently identify gestures.

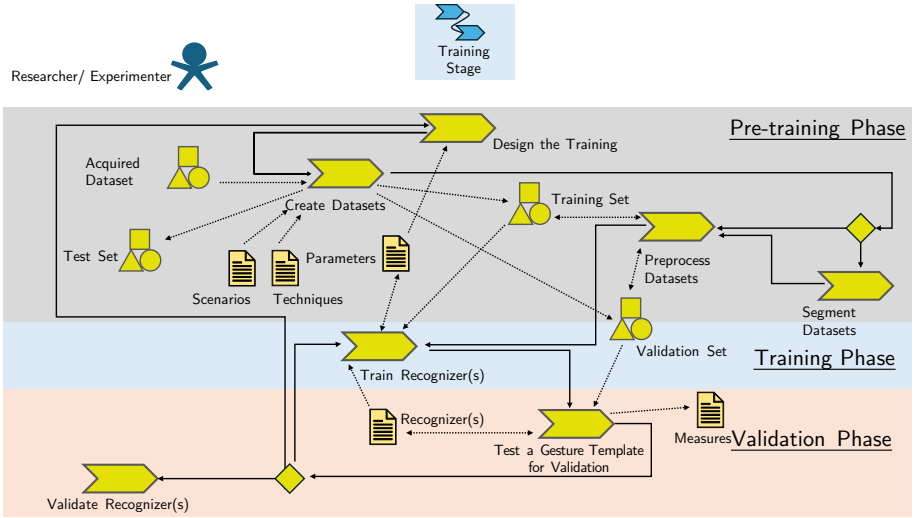


FIGURE 3.7: Activity diagram for Training.

3.5.4 Stage4: Testing

The testing stage evaluates the performance of the gesture recognizer by predicting the class label of a candidate gesture using the learned gesture model [55, 141]. Unlike the offline training stage, the testing stage is considered an online phase. In this case, a gesture template is selected from the gesture set, which is not part of the training set. Or it could result from segmenting a continuous stream of data using various methods (See Subsection 2.1.4) to extract the important segment of the gesture.

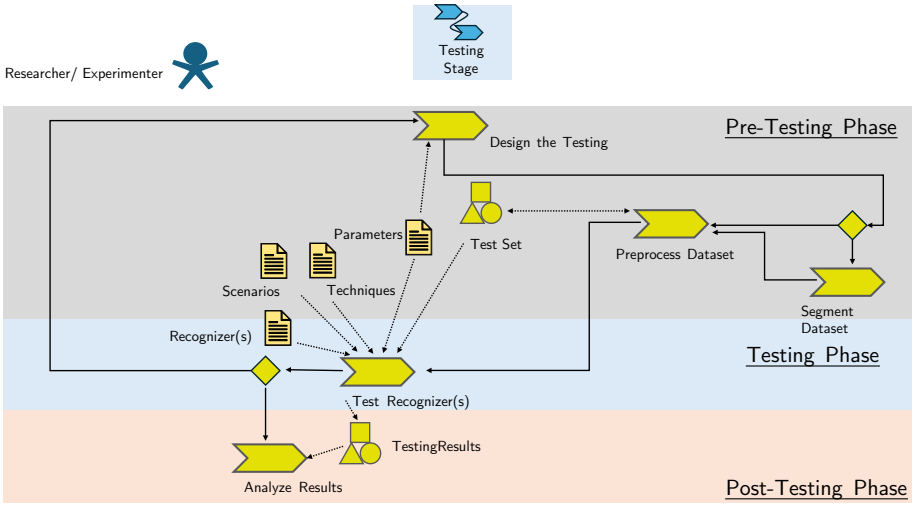


FIGURE 3.8: Activity diagram for Testing.

Workflow The testing discipline for 3D hand gesture recognizers is described in the Figure 3.8 and is performed by the *Researcher/Experimenter*. The *Design the Testing* is an essential activity that involves setting the test parameters using the document *Parameters*. This is followed by the segmentation of the *Test Set* work product, an activity that is part of the Pre-testing phase. This is followed by the preprocessing of the test set, which requires the use of the *Test Set* work product as both input and output. Testing itself involves various documents (*Scenarios*, *Recognizer(s)*, *Techniques*, *Parameters*) and culminates in the generation of specific work product *Testing Results*. The final activity is to analyze these results to extract meaningful insights from the data.

3.6 Comparative Testing

Numerous comparative testing studies have been conducted in the field of gesture recognition. For instance, the study by Khan *et al.* [54] examines various vision-based hand gesture recognition systems, focusing on segmentation, feature detection, and extraction phases.

Another study assesses the effectiveness of a proposed feature selector method, comparing it with three other methods based on four performance measures and prediction time. The aim is to understand the impact of the feature selector on the performance of data fusion in activity recognition [133].

Ferrer *et al.* [42] conducted a comparative study on various features for predicting human motion based on minimum curvature variance.

Popov *et al.* [92] Compared the performance of the developed two-stage real-time hand image-based gesture recognition system in complex backgrounds with MobileNets, a state-of-the-art neural network for real-time mobile vision applications, one of the concerns expressed in their work is the lack of comparative testing methods.

A different study compares various devices within a game design context [53]. The work done by Erazo *et al.* [41] who compared five 2D stroke gesture recognition algorithms on multiple datasets, described briefly the experimentation procedure, which is an instantiation of the procedure described in Section 3.4. Despite numerous comparative studies in the field of gesture recognition, many of them were carried out opportunistically or were ad hoc methods [96, 10, 144], none have focused on defining a systematic procedure for 3D hand gesture recognition using template-based recognizers.

3.6.1 Artefacts for Comparative Testing

The goal of the gesture recognition comparative testing method is to assess the performance of various template-based gesture recognizers in multiple contexts of use. These evaluations usually involve comparing the accuracy or recognition rate of the recognizers and considering factors such as computational complexity, real-time capability, and feasibility on affordable hardware. Comparative testing is conducted by applying a set of predefined test cases to determine the most accurate gesture recognizer. It relies on a framework to compare different template-based gesture recognizers. From the literature, we extracted the key components of comparative testing for gesture recognition.

3.6.2 Recognizers

The recognizers serve as the central component of the comparative testing framework. The evaluator includes a collection of template-based recognizers that are capable of identifying a candidate gesture that matches a gesture class defined by a limited number of training templates. These recognizers should be implemented in the same programming language and be compatible with datasets consisting of 3D hand gestures.

3.6.3 Techniques

There are several techniques used to evaluate the performance of gesture recognizers [7, 8, 55, 140, 125, 124]. One common technique is K-fold cross-validation, where a dataset is divided into k groups and each group is used as

a test set during the evaluation process. Other techniques described in [16] are the 50%-50% technique, where the dataset is divided into equal training and test sets. The leave-one-out cross-subject evaluation trains models on data from all but one subject and tests them on the subject not included in the training set. The leave-one-out cross-session evaluation focuses on hand gestures recorded from a single subject and tests the models in the remaining session after training on data from all other sessions.

3.6.4 Scenarios

The procedure of comparative testing aims to handle the 3D hand gesture recognizer in multiple contexts. The number of devices used and the different configurations during the gesture recognition process determine the system's ability to handle both single-device and multi-device configurations. As explained in Section 3.2, for instance, in the Cross-input Device family scenario, the recognizer is trained using samples performed from two or more devices of the same device family and category and then tested using the same device.

When studying the user variable, two commonly used scenarios inspired by the k-fold cross-validation technique are user-dependent and user-independent.

Unlike the cross-validation technique, these techniques randomly select training and test samples from the dataset, without dividing it into k groups. In the user-dependent scenario, the evaluation involves training and testing a recognizer with gestures produced by the same end user. Randomly selected samples per gesture class are used for training, and one remaining sample per gesture class is used for testing. In the user-independent scenario, the recognizers are trained with gestures produced by a set of independent participants who are randomly selected to train the recognizer. The recognizer is tested with gestures produced by an independent end user. For each participant, one sample per gesture class is randomly selected for testing.

3.6.5 Datasets

The datasets used to train and test the recognizers can consist of various types of gestures performed by one or multiple users. The selection of datasets depends on factors such as the context of use, the diversity and variability of gestures, and the availability of reference data. It is advisable to use publicly available datasets that allow for reproducibility.

The evaluator can choose to include one or more datasets from one or more devices. These datasets may differ in terms of gesture classes, participants, and equipment and can impact the configuration and scenario of the comparative testing.

3.6.6 Parameters

Comparative testing requires the configuration of numerous parameters within the testing procedure. These parameters are context-dependent and vary based on the test scenario.

In the user-dependent scenario, the recognizer is trained and tested using gestures produced by the same end user. The main variables in this scenario are the number of templates (T) and the number of tests (repetitions) (R).

In the user-independent scenario, the recognizer is trained with gestures produced by one or more end users and tested with gestures produced by an independent end user. This scenario includes an additional variable, the number of participants (P).

We can adjust the parameter values by assigning them any value. These values can be derived from experience, previous studies, or constraints from the dataset. The number of additional parameters may increase depending on the devices used during the gesture recognition procedure, especially when considering the different scenarios brought about by the use of multiple devices.

3.6.7 Procedure for Systematic Comparative Testing

Based on the detailed testing procedure outlined in Section 3.4, it was determined that the single gesture recognition testing procedure shares artifacts with comparative testing, as detailed in Subsection 3.6.1. The introduction of a unified process for single gesture recognition testing into the workflow, along with the capability to repeat each stage multiple times, offers a systematic approach to this procedure through clearly defined activities. The primary function of the comparative testing procedure is to test the recognizers on a set of predefined test cases, which utilize the same components as the single testing procedure and the same activities. The reuse of a coherent set of activities highlights its completeness and underscores the systematic nature of comparative testing. For example, if we aim to compare two recognition algorithms across two experiments and use different procedures for each, the results won't be comparable. Conversely, by following the stages of the systematic procedure, we can ensure the results are comparable.

3.7 Conclusion

In conclusion, this chapter provides an overview of the definitions and concepts involved in the comparative testing of gesture recognizers for multiple contexts of use. It emphasizes the importance of a thorough understanding of concepts and well-defined testing procedures to ensure the development

of robust and accurate gesture recognizers in multiple contexts of use. In the upcoming chapters, this procedure will be applied using specific examples, with a focus on two different scenarios: comparing template-based gesture recognizers on 3D hand unipath dynamic gestures and template-based gesture recognizers on 3D hand "multipath dynamic" gestures.

Chapter 4

Use Case 1: Comparative Testing of Recognizers on Unipath Dynamic Gestures

The work presented in this chapter was initially published in *Proceedings of the ACM on Human-Computer Interaction, Volume 4, Issue ISS, Article No.: 198, pp 1–21*, published in 2020.[87]

In this work, we use the term "use case" in its general definition "**Use Case** *n. A use to which something (such as a proposed product or service) can be put*"(Merriam-webster: Use Case)[79]. It is an application of the comparative testing systematic procedure defined in Chapter 3. Two experiments will be conducted in the context of specific use cases focusing on "unipath dynamic" gestures (Chapter 4) and "multipath dynamic" gestures (Chapter 5), respectively. These experiments will be evaluated under an "intra-device configuration", where a single device will be used during different stages of the gesture recognition process, as shown in Figure 4.1. This configuration was determined by the datasets used for the experiments.

In this chapter, we perform an experiment on seven template-based recognizers. In order to assess the efficiency of these recognizers for interaction using the comparative testing method. This interaction requires a low response time (*e.g.*, 0.1 s according to Nielsen [84]) and high accuracy (*e.g.*, a recognition rate of $\geq 90\%$ [76, 134]). Under predefined conditions (*i.e.*, parameter values), we will use a challenging set of 3D trajectories known as "unipath dynamic" gestures which are divided into six datasets from different input device categories, including "LeapMotionController" for SHREC2019 and 3DTCGS, and "SoftKinetic DepthSense DS325" for the 3DMadLabSD. The experiment will be conducted to measure recognition rates and execution times for two datasets in user-independent scenarios. Additionally, we include one dataset that includes both user-dependent and user-independent scenarios.

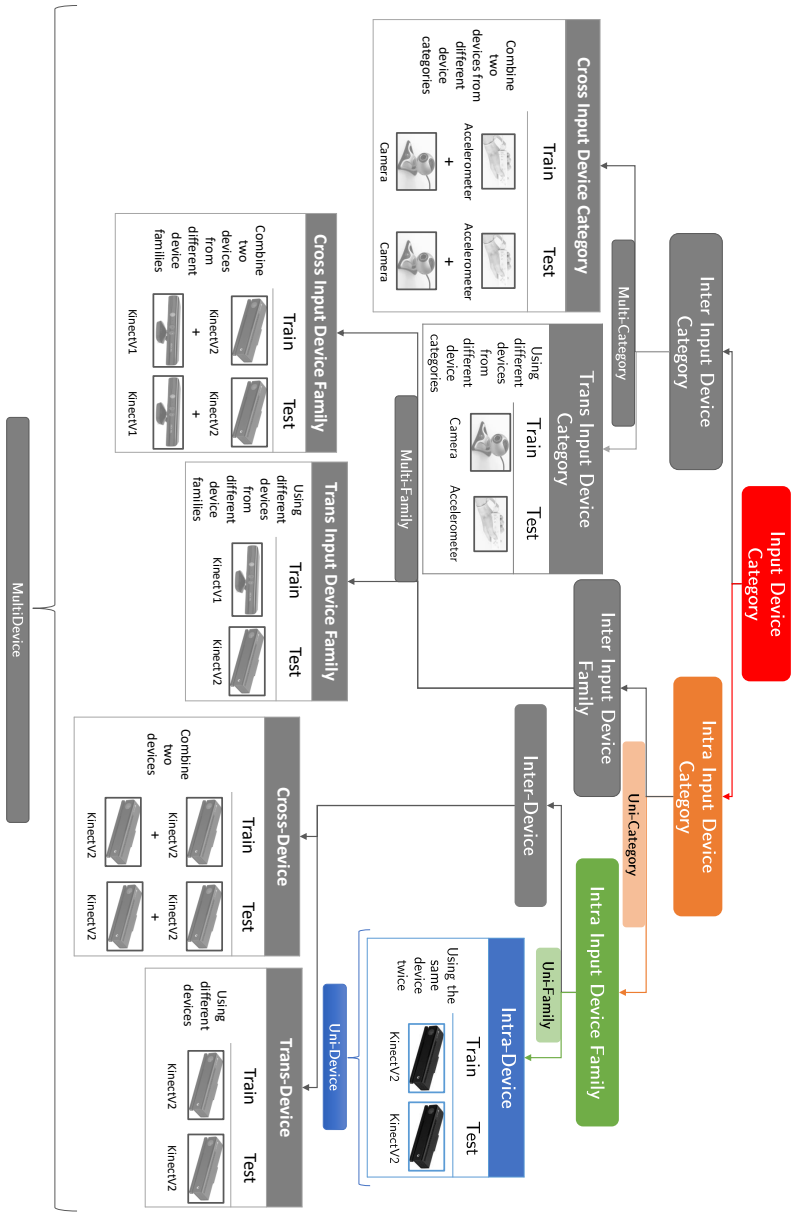


FIGURE 4.1: Configuration of Input Devices in Use Cases.

Some recognizers perform similarly to existing 3D recognizers for these gestures. We also discuss the implications of selecting a recognizer based on context-specific conditions, such as sampling and number of points, to determine the impact on recognition rate and execution time.

4.1 Recognizers

Based on the TLR in Subsection 2.2.2, four recognizers were selected for comparative testing. Rubine [98] pioneered 2D stroke recognition using statistical matching on weighted feature vectors. $\$P$ [124] introduced the cloud matching principle, while $\$P+$ [122] has the best accuracy and $\$Q$ [125] has the fastest execution time.

$\$1$[140] is not included because several aforementioned recognizers outperform it [115, 118]. Similarly, $\$N$ is also not included due to a combinatory explosion in both memory and execution time despite two-speed optimizations.

$\$P$ addresses this limitation by treating gestures as unordered sets of points. Other extensions and local optimizations discussed in Subsection 2.2.2 are excluded. $\$3$ and similar algorithms are potential candidates but are not included because they rely on additional data from a wearable device, such as an accelerometer, which we want to avoid. Vector-based recognizers [115, 118] are not retained as they may require recomputation of vectors between points or vectors between vectors for each plane, leading to increased computational cost. Other 3D recognizers belonging to classifier classes with different computational complexities to cover a broader range of 3D gestures were not included.

The selected recognizers undergo a three-dimensionalization process, adding the Z dimension as an independent dimension to the existing X and Y dimensions [83].

4.1.1 Rubine3D, a 3D Extension of Rubine

Rubine3D (*R3D*) extends the feature-Based 2D gesture recognizer Rubine [98] to handle 3D gesture. It is inspired by the Rubine3D implemented in the iGesture framework [108] (Subsection 2.4.2). The Rubine3D recognizer combines three individual 2D Rubine recognizers for each plane: XY, YZ, and ZX. The projection on each plane is achieved by transforming each point of the 3D trajectory to the three coordinate system planes (XY, YZ, ZX). A plane in an orthogonal coordinate system is represented by the linear equation $Ax + By + Cz + D = 0$, where $n(A, B, C)$ are the coordinates of the normal

vector to a plane. If the plane passes through the origin, the equation has constant term D=0. The XY, YZ, and ZX planes passing through the origin have equations $z = 0$, $x = 0$, and $y = 0$ respectively. For a point $M(i, j, h)$ in the 3D trajectory, P, Q, R are the 2D points where the coordinates of M intersect the (XY, YZ, ZX) planes. These points are the orthogonal projections of M onto each plane with vectors parallel to the X, Y, and Z axes. Thus, $P(i, j)$, $Q(j, h)$, and $R(h, i)$ are the coordinates of the 2D projections on (XY, YZ, ZX).

First, the raw data are preprocessed by scaling and filtering points. Thus, a point is discarded if the distance from the previous point is under a threshold of d=.005. Next, for each training gesture projected on each plane, the thirteen original features $(f_1, .., f_{13})$ [98] are calculated.

After that, every class mean feature vector and covariance matrix are computed for all planes. We then calculate the common covariance matrix and the inverse matrix, of which the gesture class weights are estimated. Thus, several actions are required to determine the possible class of the candidate gesture. To begin, the gesture is projected on each plane.

Next, the feature vectors are extracted from the three gestures projections. After that, the gestures projections are defined as one of the possible gesture classes with a linear function. This function evaluates which class maximizes the evaluation function result.

Finally, a heuristic inspired by iGesture is used to determine the final result class of the 3D gesture. If all three projections have the same result class, this is called the final result. However, if there are two or three different classes, we calculate the scores of these classes for the three planes and multiply each score result by a weight factor. The weight factors for the planes are $W_{XY} = 0.4, W_{YZ} = 0.3, W_{ZX} = 0.3$, based on the ease of producing gestures in the sagittal plane, then in the transversal plane, and finally in the frontal plane, which is particularly observed for head and shoulder gestures [119]. We found that using only two planes decreased accuracy. The final gesture class is determined by summing the scores.

However, the gesture rejection part of Rubine has not been implemented. This decision was made to ensure consistency in computational complexity with other recognizers and to focus solely on the classification problem.

4.1.2 Rubine-Sheng, Another 3D Extension of Rubine

Rubine-Sheng (RS) uses a vector of sixteen features computed from a 3D gesture $G = \{p_t = (x_t, y_t, z_t) | \forall t = 1, ..., n\}$. These features extend Rubine's original feature vector to 3D by incorporating three additional features proposed in the AdaBoost recognizer and applied to 3D gestures [107]. Similar to its 2D counterpart, the Rubine-Sheng recognizer classifies a candidate gesture by identifying its corresponding gesture class. First, each class's mean feature

vectors and covariance matrix are estimated using the feature vectors of the training samples. Then, the common covariance matrix is computed and its inverse is used to determine the weights of the gesture classes. Finally, the class of a candidate gesture is the class that maximizes the result of the discrimination function.

4.1.3 $\$P^3$, a 3D Extension of $\$P$

The $\$P$ algorithm matches a candidate gesture C with each template T in the training set. This is done using a function M that associates each point $C_i \in C$ with one point $T_j \in T$, where $T_j = M(C_i)$. The classification result determines the closest template T to the candidate C by calculating the matching distance: $C \in$ class of T where $T = argmin_T\{\$P(C,T)\}$. This principle is extended to 3D by defining a set of points with three-dimensional coordinates: $C = \{p_i = (x_i, y_i, z_i) | \forall i = 1, ..., n\}$.

$\$P^3$ is the 3D extension of $\$P$, inspired by the implementation of the $P3D recognizer by Cook *et al.* [31]. However, unlike $P3D, the P^3 recognizer does not support 3D static poses and 2D dynamic gesture recognition.

To minimize gesture variations, the recognizer performs three pre-processing steps:

- Resampling the point clouds of gestures to ensure they have the same number of points n for matching.
- Scaling the gesture points to a non-uniform reference box.
- Translating the centroid of the gesture (Gx, Gy, Gz), to the origin $O = (0, 0, 0)$.

The matching score is defined as the sum of Euclidean distances of all pairs of points from M. This calculation is generalized into 3D by incorporating the third coordinate, z. Let's assume that point C_i from the candidate gesture C is matched to point T_j from the template T. The score is then given by:

$$\sum_{i=1}^{n} \parallel C_i - T_j \parallel = \sum_{i=1}^{n} \sqrt{(C_i.x - T_j.x)^2 + (C_i.y - T_j.y)^2 + (C_i.z - T_j.z)^2} \quad (4.1)$$

In order to compute the dissimilarity score between two clouds of points, we performed a one-to-one time-free alignment between points, inspired by Vatavu's matching heuristic named Greedy-5 because it gave the best results among the tested methods [124]. The heuristic matches each point from the first cloud with one point in the second cloud, with the condition that it has not been matched before. Next, it matches points from the second cloud that

have not been matched yet can be matched to one point in the first cloud resulting in a complexity of $\mathcal{O}(n^2)$. The algorithm runs several times with different starting points considered circularly through all points and returns the minimum matching of all runs. The returned matching sum is multiplied by a weight representing a confidence degree on the matching.

$$w = \sum_i w_i \cdot \parallel C_i - T_j \parallel \tag{4.2}$$

The ϵ threshold controls the number of runs and affects the complexity of the heuristic to $\mathcal{O}(n^{2+\epsilon}))$. We take into account that the direction of matching impacts the result of the heuristic.

4.1.4 $\$F$, a Flexible Variant of $\$P^3$

$\$F$ extends the previously mentioned $\$P^3$ with the flexible cloud matching of $\$P+$ [122]. As usual, both the candidate and template points are resampled to equidistantly-spaced points, scaled within a unit box, and translated so that their centroid is at the origin $(0, 0, 0)$. The template with the lowest dissimilarity score is considered the best matching template for the candidate.

The $\$P+$'s cloud matching process involves matching the points from the first cloud with their closest point from the second cloud, and then matching the remaining points from the second cloud with their closest point from the first cloud. To investigate the possibility of a more flexible matching, $\$F$ allows the matching of two points that have already been matched, hence its name $\$F$.

4.1.5 FreeHandUni, a 3D Extension of $P

The FreeHandUni (*FH*) recognizer is based on the pseudocode of the FreeHand recognizer, which extends the 2D gesture recognizer $P++ [33]. Additionally, it uses full hand information for free-hand gesture recognition. FreeHandUni is adapted to recognize the unipath character of 3D trajectories by replacing the hand pose structure with a 3D point structure (x, y, z). With this modification, FreeHandUni can be seen as an improvement of $\$P^3$, using a flexible cloud matching approach based on one-to-many alignment between points [122]. The pre-processing step remains the same, and the matching process uses Euclidean distance (Eq. 4.1) with increased flexibility. Each point from the template cloud is matched to the closest point from the candidate cloud, and each remaining point from the candidate cloud is matched to the closest point from the template cloud. The gesture class is determined based on the template cloud with the lowest dissimilarity score compared to the candidate cloud, and

vice versa. Unlike $\$F$, FreeHandUni does not implement early abandoning to keep the computational complexity close to $\$P^3$.

4.1.6 $\$P+^3$, a 3D Extension of $\$P+$

One of the major improvements with respect to $\$P$ lies in expanding the matching from one-to-one to one-to-many points, which clears the weights of the starting point used in $\$P$. Based on the 3D points, we compute the turning angle at each point with points coordinates gesture:

$$C_{i.\theta} = \frac{1}{\pi} \arccos(\gamma) \tag{4.3}$$

where

$$\gamma = \frac{(C_{i+1}.x - C_i.x)\cdot(C_i.x - C_{i-1}.x) + (C_{i+1}.y - C_i.y)\cdot(C_{i+1}.y - C_i.y) + (C_i.z - C_{i-1}.z)\cdot(C_i.z - C_{i-1}.z)}{||C_{i+1} - C_i||\cdot||C_i - C_{i-1}||} \tag{4.4}$$

To optimize the execution time, the early abandoning used in $\$Q$ is added, as well as the third coordinate in the computation of the point distance beside the angle as in $\$P+$:

$$D(C_i, T_j) = \sqrt{(T_j.x - C_i.x)^2 + (T_j.y - C_i.y)^2 + (T_j.z - C_i.z)^2 + (T_j.a - C_i.a)^2} \tag{4.5}$$

4.1.7 $\$Q^3$, a 3D Extension of $\$Q$

$\$Q^3$, is the 3D extension of $Q [125]. It performs the same preprocessing as $\$P+^3$, except for the calculation of a 3D Look-Up-Table (LUT) offline for each template and its storage with the cloud points. The look-up point-matching technique is generalized to three dimensions using a $16\times16\times16$ 3D grid of equidistant points. Each cloud point C is positioned within this grid. The index of the row, column, and layer of the closest point in the grid is stored for each point, resulting in a computational complexity of $\mathcal{O}(n \cdot m^3)$. To recognize a candidate, the closest point from a template to C_i is determined by iterating through the cloud points and summing the Euclidean distances of Eq. 4.1 between each pair of points. This computation is stopped when the sum exceeds the minimum dissimilarity score.

4.2 Datasets

4.2.1 SHREC2019

The **SHREC2019** dataset [24] was used as a benchmark for the Eurographics 2019 SHape Retrieval Contest (SHREC) track on online gesture recognition. It consists of 195 3D trajectories performed by 13 participants using their whole hand, representing five iconic gestures: "Cross" (X), "Circle" (O), "V-mark" (V), "Caret" (/\), and "Square" ([]). The dataset is relevant because it is recent and contains simple gestures.

To detect command gestures from hand movements in a virtual reality context, the training set and testing set were combined to create a single dataset. Unnecessary hand movements were filtered by clipping the sequences before and after the main section [56].

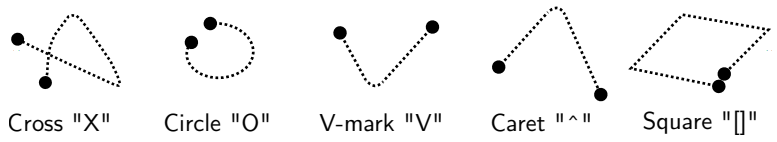


FIGURE 4.2: The SHREC2019 dataset [24].

4.2.2 3DTCGS

The **3DTCGS** dataset [23] is used to test the classification performance of the 3¢ [23] recognizer on interface command gestures. The data is provided in a segmented form, making it easier to exploit during experimentation for the classification task. This dataset contains gestures of varying complexity, ranging from simple ones like a "3D swipe" to more complex ones like a "3D spiral". Similar trajectories also include different directions (*e.g.*, "left-swipe" vs. "right-swipe", "arc3Dleft" vs. "arc3Dright"), to test the direction-invariance property.

Each subject produces one sample per gesture, which makes the user-independent evaluation particularly challenging.

Participants were asked to perform short iconic gestures with their dominant hand forefinger. The collected data consists of 347 sequences of 3D coordinates, combined with a timestamp. These sequences describe 26 gesture classes performed by 14 subjects and recorded with a Leap Motion (see Figure 4.3).

4.2.3 MadLabSD

The **MadLabSD** (MadLab Sketch Dataset) dataset [49] is publicly available online since 2018. It was used to assess a new gesture-based system [49]. This

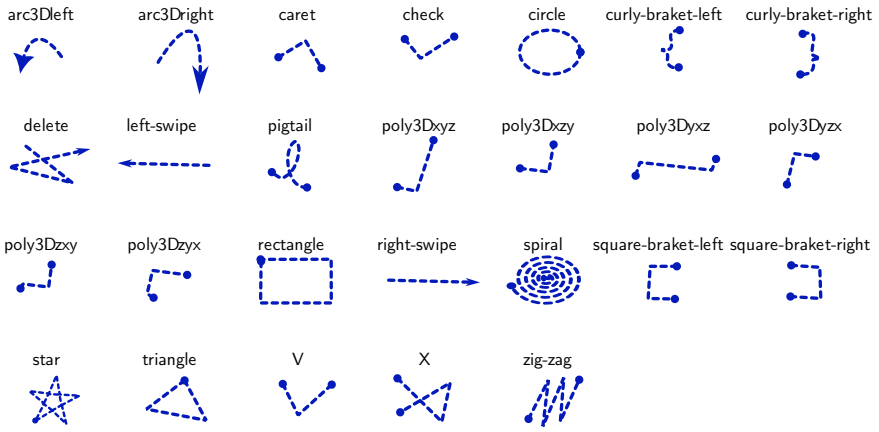


FIGURE 4.3: The 3DTCGS dataset [23].

dataset consists of mid-air single stroke gestures, which are palm-centered dynamic motion trajectories. It offers a wide range of gestures, from common symbols (*i.e.*,, alphabets and numerals) to uncommon ones (*e.g.*, CAD primitives), making it particularly attractive.

The dataset was recorded by a group of 10 users who captured 40 segmented symbols and sketches. These were split into four domains, as shown in Figure 4.4, resulting in a total of 4,000 samples. Each user recorded 10 samples per gesture, which included the 3D position coordinates of the center of their palm along with a timestamp.

The dataset also includes the extraction of the center palm from depth grayscale images, which were recorded using a SoftKinetic DepthSense DS325. The gesture was recorded based on the static hand pose, allowing the user to switch between active (hand open) and inactive (closed hand) states.

It is important to note that the raw data from the depth camera is not preprocessed, potentially containing spurious data, such as noisy points at the start of the sketch, due to user latency after mode switching [49].

The dataset comprises four domains:

- **(D1) Arabic numerals**: *0-9*
- **(D2) English alphabets (Handwritten)**: *a-j*
- **(D3) Simulation symbols**: *Spring-Mass-Wheel-Pulley-Hinge-Fast Forward-Rewind-Play-Pause-Delete*
- **(D4) CAD primitives**: *Cuboid-Cylinder-Sphere-Rectangular Pipe-Hemisphere-Cylindrical Pipe-Pyramid-Tetrahedron-Cone-Toroid*

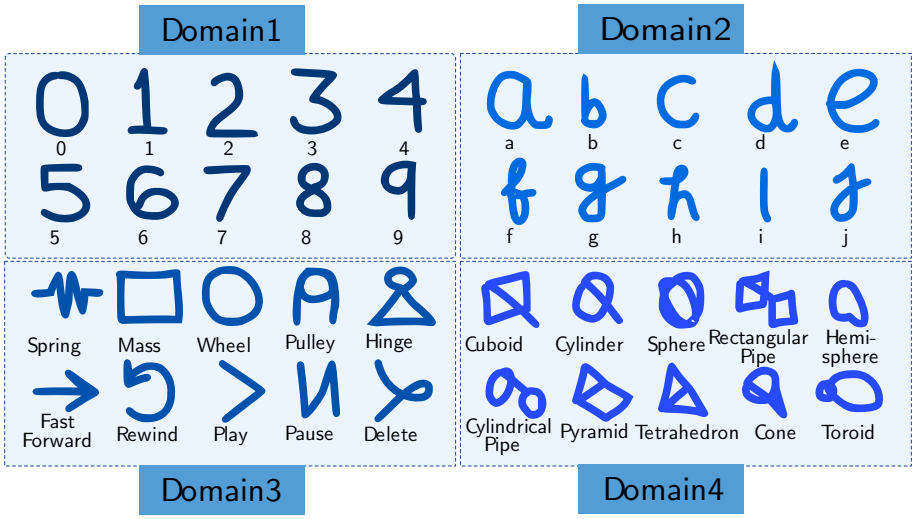


FIGURE 4.4: The four domains of the MadLabSD dataset. [49]

4.3 Design

We evaluated the seven recognizers using both traditional user-dependent and user-independent scenarios [7, 8, 124, 122] on 3D trajectories from six datasets. The goal was to demonstrate the efficiency of these recognizers under different conditions (*i.e.*, parameters values). We focused solely on evaluating the classification task, not the preprocessing, as this task can be done offline.

4.3.1 Experiment

Our study was within-factors with four independent variables:

- RECOGNIZER: This nominal variable with 7 conditions, representing the various recognizers implemented to recognize 3D trajectories: $\$P^3$, $\$P+^3$, $\$Q^3$, $\$F$, FH (FreeHandUni), $R3D$ (Rubine3D), and RS (Rubine-Sheng).
- DATASET: This nominal variable with 6 conditions (see Table 4.1 and Figure 4.2,4.3,4.4), representing two datasets considered as a whole, namely SHREC2019 [24] and 3DTCGS [23], and four domains of 3DMadLabSD [49]-D1, 2, 3, and 4.
- NUMBER OF TEMPLATES: This numerical variable with 5 conditions, representing the number of templates per gesture class used to train the recognizer: $T=\{1, 2, 4, 8, 16\}$.

- SAMPLING: This numerical variable with 5 values representing the number of points per gesture: $N=\{4, 8, 16, 32, 64\}$.

TABLE 4.1: Description of selected datasets. Notation for the Ground truth information: Timestamp (T), Label (L), Hand joints (J).

Name	Sensor	Subjects	Classes	Instances	Ground truth information
SHREC2019 [24]	Leap Motion	13	5	195	T, L, J
3DTCGS [23]	Leap Motion	13	26	347	T, L
3DMadLabSD [49]	SoftKinetic DepthSense DS325	10	40 (10 x 4 Domains)	4000	T, L

Apparatus

We used a sexa-core Intel Core i7 2.20 GHz CPU and running a Windows 10 Home Edition operating system. The RAM was 16 GB DDR4 memory with 2400 MHz. We ran the framework described in Chapter 6 with the seven recognizers on the six individual datasets.

4.3.2 Procedures and Quantitative Measures

We compute the *recognition rate* (Which is the ratio of positive recognitions divided by the total number of trials) and the *execution time* (measured in milliseconds as the time taken to preprocess a candidate gesture and recognize its result class) for the 7 (RECOGNIZER) $\times$ 6 (DATASET) = 42 basic configurations following the typical procedure used in the literature to evaluate gesture recognizers [4, 7, 8, 124, 125, 140]. In the "user-independent scenario", recognition is evaluated on gestures produced by users other than those used to train the recognizer. In the "user-dependent scenario", recognition is evaluated on gestures produced by the same users used to train the recognizer. In total, 665,000 trials were performed.

User-Independent Scenario

For the user-independent scenario, one template is randomly selected for each gesture class from all participants and saved for testing. Then, a training set is created by randomly choosing T templates for each gesture class across all users. These templates should be different from those selected for testing. The recognizer is then trained using the resulting training set.

To evaluate the performance, the recognizer is tested 100 times for each T value: $T=\{1, 2, 4, 8, 16\}$ for SHREC2019 and MadLabSD, and $T=\{1, 2, 4, 8, 11\}$ for 3DTCGS. For the Rubine's recognizer 3D extensions ($R3D$ and RS), $T=\{2, 3, 4, 8, 16\}$ for the SHREC2019 and MadLabSD datasets, and $T=\{2, 3, 4, 8, 11\}$ for the 3DTCGS dataset. The number of correctly recognized

gestures is counted, and this is repeated for each T value. Finally, the average recognition rate is calculated by averaging the number of recognized gestures and formatting it as a percentage for each N and for each recognizer. In total,105,000 recognition trials were performed by running 6 (DATASET) $\times$ 5 (SAMPLING) $\times$ 5 (NUMBER OF TEMPLATES) $\times$ 100 (Repetitions) $\times$ 7 (RECOGNIZER).

User-Dependent Scenario

For the user-dependent scenario, one template from each gesture class is randomly selected for each user. Then, for each user, a set of randomly selected T templates is created for each gesture class, excluding the templates selected for testing by the same user. This training dataset is used to train the recognizer. Similar to the user-independent scenario, the recognizer is tested 100 times for each user and for each T value, depending on the recognizer: T=$\{2, 3, 4, 8\}$ for $R3D$ and RS, and T=$\{1, 2, 4, 8\}$ for the others. The number of correctly recognized gestures is counted, and the average recognition rate is calculated by averaging the results for each user and T value. The recognition rate is then presented in percentage format for each N and for each recognizer. Only the MadLabSD dataset was used because SHREC2019 does not identify the emitter of the gesture, and 3DTCGS contains only one template per user for each gesture class. In total, we performed 560,000 recognition trials by running 4 (DATASET) $\times$ 10 (Users) $\times$ 5 (SAMPLING) $\times$ 4 (NUMBER OF TEMPLATES) $\times$ 100 (Repetitions) $\times$ 7 (RECOGNIZER).

4.4 Results

Table 4.2 provides a summary of these results for all datasets as well as for each individual dataset.

All reported results are averaged on all datasets, ranging from the simplest one to the most complex one (*e.g.*, Domain 4).

4.4.1 User-Independent Scenario

Recognition Rate

Recognizers are ranked in descending order according to their average recognition rate on all datasets, as shown in the following Figure 4.5): $\$P+^3$ is superior to $\$F$, then FH, $\$Q^3$, $\$P^3$, followed by the two Rubine recognizers $R3D$ and RS. $\$P+^3$ is 9.98% more accurate than its successor $\$F$, which is roughly in the same interval as FH, $\$Q^3$, and $\$P^3$, which is in turn 15.29% more accurate than $R3D$.

TABLE 4.2: Summary of recognition rates and execution times for all datasets and per individual dataset. Recognizers are sorted in decreasing order of their average recognition rate for all datasets.

Recognizer		Datasets											
		Overall		SHREC	3DTCGS	Domain 1		Domain 2		Domain 3		Domain 4	
		UD	UI	UI	UI	UD	UI	UD	UI	UD	UI	UD	UI
$\$P+^3$	rate (M)	74.40	87.48	86.94	84.28	98.45	88.08	98.81	87.87	99.26	94.45	93.86	66.36
	rate (SD)	1.73	13.68	0.75	2.22	4.13	2.34	3.67	1.27	2.78	0.76	0.92	1.65
	time (M)	0.53	0.77	0.36	1.24	0.54	0.76	0.53	0.77	0.49	0.69	0.54	0.81
	time (SD)	1.43	0.85	0.64	2.00	0.87	1.37	0.87	1.39	0.78	1.24	0.88	1.46
$\$F$	rate (M)	68.13	79.54	79.36	78.83	96.22	76.38	96.78	75.45	97.67	85.66	91.76	57.94
	rate (SD)	1.89	16.08	3.55	2.76	6.98	1.63	6.70	3.28	5.26	1.28	1.07	1.68
	time (M)	0.42	0.59	0.29	0.91	0.41	0.57	0.42	0.58	0.38	0.53	0.45	0.67
	time (SD)	1.04	0.63	0.48	1.40	0.63	0.96	0.64	1.00	0.57	0.91	0.69	1.15
FH	rate (M)	68.46	79.49	79.28	78.84	96.21	76.05	96.69	75.78	97.70	85.55	91.96	57.75
	rate (SD)	1.89	16.28	0.93	2.80	6.97	3.20	6.79	3.24	5.28	1.26	1.04	1.70
	time (M)	0.94	1.23	0.49	2.41	0.92	1.07	0.92	1.08	0.93	1.14	0.97	1.19
	time (SD)	2.55	1.65	0.94	4.22	1.62	2.04	1.62	2.07	1.66	2.13	1.70	2.30
$\$Q^3$	rate (M)	67.36	79.09	79.36	78.45	95.79	75.20	96.28	74.55	97.51	84.88	91.31	57.02
	rate (SD)	1.91	16.54	0.93	2.94	7.22	3.21	7.05	3.22	5.47	1.31	1.09	1.72
	time (M)	3.12	3.76	3.01	4.81	3.14	3.55	2.97	3.53	3.13	3.42	3.25	4.27
	time (SD)	4.16	3.14	2.94	5.39	3.11	3.70	2.89	3.67	3.19	3.51	3.36	4.92
$\$P^3$	rate (M)	65.90	77.50	78.10	77.43	95.28	72.57	95.70	71.55	97.18	82.92	90.68	54.98
	rate (SD)	1.93	16.88	0.96	2.99	7.70	3.19	7.54	3.28	5.93	1.40	1.12	1.73
	time (M)	8.71	10.94	4.80	20.79	8.59	9.83	8.65	10.19	8.72	9.87	8.88	10.16
	time (SD)	24.45	16.76	10.02	39.79	16.49	20.69	16.62	21.38	16.76	20.87	17.15	21.12
$R3D$	rate (M)	60.80	67.22	52.51	74.48	75.60	72.57	75.94	70.33	76.93	76.52	72.99	54.40
	rate (SD)	2.98	24.93	1.61	2.29	38.46	6.49	38.48	6.45	38.83	3.43	3.74	2.69
	time (M)	0.08	0.08	0.05	0.12	0.06	0.07	0.06	0.06	0.07	0.08	0.10	0.10
	time (SD)	0.027	0.016	0.013	0.018	0.004	0.009	0.005	0.005	0.005	0.009	0.002	0.007
RS	rate (M)	50.04	57.45	40.82	68.82	69.07	58.37	70.76	58.10	73.36	67.98	65.04	43.49
	rate (SD)	2.63	23.56	1.51	2.50	36.18	5.77	36.86	5.66	38.01	3.24	3.48	2.35
	time (M)	0.03	0.03	0.02	0.05	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.04
	time (SD)	0.012	0.007	0.006	0.011	0.002	0.002	0.005	0.005	0.003	0.003	0.004	0.002

The overall difference between the recognizers was statistically very highly significant ($F_{6,10459} = 352.89$, $^{***}p<.001$, $\eta^2=.019$ (–)). Table 4.3 details the results of ANOVAs for all datasets (column "Overall") and per dataset. For example, $\$P+^3$ is more accurate than $\$F$ with a very high significant difference (q=21.18) and a small effect size, more accurate than FH with a very high significant difference but without effect size, more accurate than $\$Q^3$ without any effect size, and so forth. In short, $\$P+^3$ is more accurate than all other recognizers with a very highly significant difference, the effect size ranging from none to medium. The difference between $\$F$ and FH, $\$Q^3$, is not significant but becomes significant over $\$P^3$ with a medium effect size, over $R3D$, and RS with a large effect size.

Execution Time

Figure 4.6 depicts the execution time in milliseconds for all recognizers in all conditions for the 3DTCGS and SHREC2019 datasets.

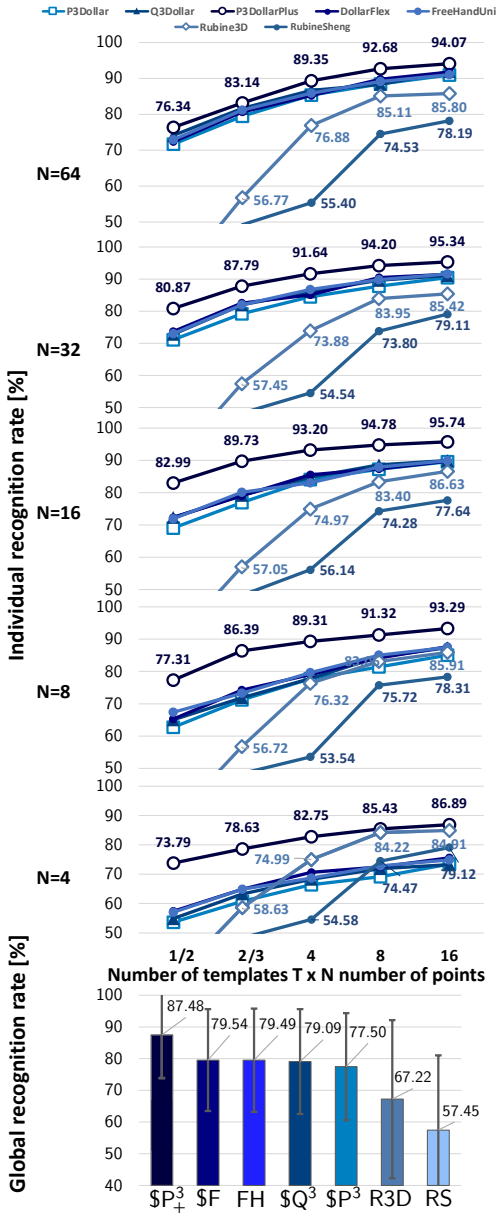


FIGURE 4.5: Recognition rates of all recognizers aggregated for all datasets in the user-independent scenario: individual rates (top) for N=4, ..., 64 (top) and global rates (bottom) $\forall N$. Error bars show their respective standard deviation.

- Overall, $R3D$ (M=.047, SD=.013) and RS (M=.017, SD=.006) have the lowest and the most constant times in all configurations.
- On the contrary, the $-like recognizers are the most time-consuming: $\$F$ (M=.289, SD=.490) is the fastest recognizer among them, followed by $\$P+^3$ (M=.363, SD=.649), FH (M=.492, SD=.956), $\$Q^3$ (M=3.005, SD=3.005) and $\$P^3$ (M=4.798, SD=10.224).
- This difference suggests that feature-based recognizers, like $R3D$ and RS, are not influenced by the same parameters as template-based ones (*i.e.*, $\$P^3$, $\$P+^3$, $\$Q^3$, $\$F$, and FH).
- RS is the fastest recognizer in most N and T conditions, except when N=4 and $T\leq$4.
- The times for all $-like recognizers increase while N and T grow.
- $\$Q^3$ is the slowest recognizer for N=4, 8. Other curves are under the $\$Q^3$ one but from N=16, $\$P^3$ execution times increase quickly and exceed the execution time of $\$Q^3$ for T=16 and it occurs even quicker for N=32 and N=64, where $\$P^3$ exceeds $\$Q^3$ immediately after T=4, resp., T=2.

TABLE 4.3: ANOVAs computed for the 7 RECOGNIZERS in the user-independent scenario: G1=group 1, G2=group 2. For each dataset, three data are provided: the q value resulting from the ANOVA, the significance of the p value if any (***$p\leq$), and Cohen's d coefficient for effect size ((S)mall when $d\geq$.02, (M)edium when $d\geq$.05, (L)arge when $d\geq$.08, and (–) when there is no significant effect size ($d<$.02)).

Recognizer		Dataset (q-stat, p value, Cohen's d)						
G1	G2	Overall	SHREC2019	3DTCGS	Domain 1	Domain 2	Domain 3	Domain 4
$\$P+^3$	$\$F$	21.18,***, S	16.80,***, S	26.64,***, M	27.48,***, M	29.11,***, M	21.30,***, M	21.35,***, S
	FH	21.30,***, –	16.97,***, S	26.63,***, M	28.15,***, M	28.33,***, M	21.57,***, M	21.83,***, S
	$\$Q^3$	22.89,***, –	16.80,***, S	28.50,***, M	30.23,***, M	31.22,***, M	23.19,***, S	23.70,***, M
	$\$P^3$	27.50,***, S	19.58,***, S	36.49,***,M	36.41,***, M	38.24,***, M	27.96,***, M	28.85,***, M
	$R3D$	37.75,***, S	76.28,***, L	47.90,***, L	36.42,***, M	41.11,***, L	43.47,***, L	30.32,***, M
	RS	61.51,***, M	102.19,***, L	75.64,***, L	69.78,***, L	69.77,***, L	64.17,***, L	58.02,***, L
$\$F$	FH	0.11, *n.s.*	0.17, *n.s.*	0.01, *n.s.*	0.77, *n.s.*	0.77, *n.s.*	0.27, *n.s.*	0.47, *n.s.*
	$\$Q^3$	1.70, *n.s.*	0.01, *n.s.*	1.85, *n.s.*	2.75, *n.s.*	2.10, *n.s.*	1.89, *n.s.*	2.34, *n.s.*
	$\$P^3$	6.31,***, M	2.78,***, S	6.86,***, –	8.93,***, –	9.13,***, –	6.66,***, –	7.50,***, –
	$R3D$	16.56,***, L	59.48,***, L	21.28,***, S	8.93,***, –	12.00,***, S	22.17,***, S	8.97,***, –
	RS	40.32,***, L	85.39,***, L	48.99,***, L	42.29,***, L	40.65, ***, L	42.87,***, L	36.67,***, M
FH	$\$Q^3$	1.58, *n.s.*	0.17, *n.s.*	1.87, *n.s.*	1.98, *n.s.*	2.88, *n.s.*	1.61, *n.s.*	2.34, *n.s.*
	$\$P^3$	6.19,***, M	2.60, *n.s.*	6.86,***, –	8.14,***, –	9.91, ***, –	6.39,***, –	7.02,***, –
	R	16.44,***, L	59.30,***, L	21.29,***, S	8.16,***, –	12.78,***, S	21.90,***, S	8.49,***, –
	RS	40.22,***, L	85.21,***, L	49.01,***, L	41.52,***, L	41.43,***, L	42.60,***, L	36.19,***, M
$\$Q^3$	$\$P^3$	4.61,***, S	2.78, *n.s.*	4.98,**, –	6.18,***, –	7.02,***, –	4.77,**, –	5.15,**, –
	$R3D$	14.86,***, L	59.48,***, L	19.42,***, S	6.18,***, –	9.89, ***, –	20.28,***, S	6.62,***, –
	RS	36.61,***, L	85.39,***, L	47.14,***, L	39.54,***, M	38.54,***, M	40.98,***, L	34.32,***, M
$\$P^3$	$R3D$	10.24,***, L	56.59,***, L	14.43,***, S	0.01, *n.s.*	2.86, ***, –	15.50,***, S	1.47, *n.s.*
	RS	34.01,***, L	82.61,***, L	42.15,***, L	33.36,***, M	31.52,***, M	36.20,***, M	29.17,***, M
$R3D$	RS	23.75,***, L	25.91,***, M	27.71,***, M	33.36,***, M	28.65,***, M	20.69,***, S	27.69,***, M

- RS is three-time faster than $R3D$: the ratio of their averaged execution times is $\frac{t_R S}{t_R 3D}$=2.82. This ratio makes sense considering that $R3D$ computes 39 features of the candidate (3 times 13 features for the three planes XY, YZ, and ZX) while RS computes 16 features once.

For the feature-oriented recognizers, the candidate is preprocessed and its features are extracted. These features are then multiplied by precomputed weights for each class and summed. The candidate's class is designated by the larger sum resulting from this calculation, which confirms a constant time influenced by the number of classes. Among the $-like recognizers, $\$F$, $\$P+^3$, and FH show shorter execution times. Comparatively, $\$F$ and $\$P+^3$ outperform FH due to the early abandoning strategy employed by these two recognizers. All conditions for this dataset have execution times below 100 ms, which is the limit for a user to perceive real-time execution of a system [84], as mentioned above.

The overall picture for 3DTCGS remains the same as for SHREC2019. The execution times increase for all recognizers and conditions, causing $\$P^3$ to exceed the 100 millisecond limit. This is due to the increase of the number of classes and samples per class in the dataset (26 classes compared with 5 gestures for SHREC2019). We observe a rapid increase of the $\$P^3$ curve that exceeds $\$Q^3$ just after T=4 for N=8.

The four 3DMadLab domains results are analogous to the first two datasets. Rubine-Sheng remains the fastest recognizer (M_{D1}=.26 ms, M_{D2}=.28 ms, M_{D3}=.31 ms, and M_{D4}=.41ms), whereas the $\$P^3$ stays the slowest one (M_{D1}=9.83 ms, M_{D2}=10.19 ms, M_{D3}=9.87 ms, M_{D4}=10.16 ms). For the fourth domain containing the CAD symbols, the execution times are slightly above the other domains, probably because their shape complexity with more crossing points and angle variations require more comparisons.

4.4.2 User-Dependent Scenario

Recognition Rate

The recognizers are sorted in decreasing order of their recognition rate averaged on *all* datasets as follows: $\$P+^3$ is superior to FH, then $\$F$, $\$Q^3$, $\$P^3$, ended by the two Rubine conditions $R3D$ and RS. $\$P+^3$ is only 2.04% more accurate than its successor FH, which is roughly in the same interval as $\$F$, $\$Q^3$, and $\$P^3$, which is in turn 25.66% more accurate than $R3D$.

The overall difference between the recognizers was again statistically very highly significant. Table 4.4 details the results of ANOVAs for the four 3DMadLab domains (column "Overall") and per domain. For example, $\$P+^3$ is more accurate than $\$F$ (q=14.38), FH (q=13.62), $\$Q^3$ (q=16.13), and $\$P^3$ (q=19.50), all with a very high significant difference and a small effect size. In short,

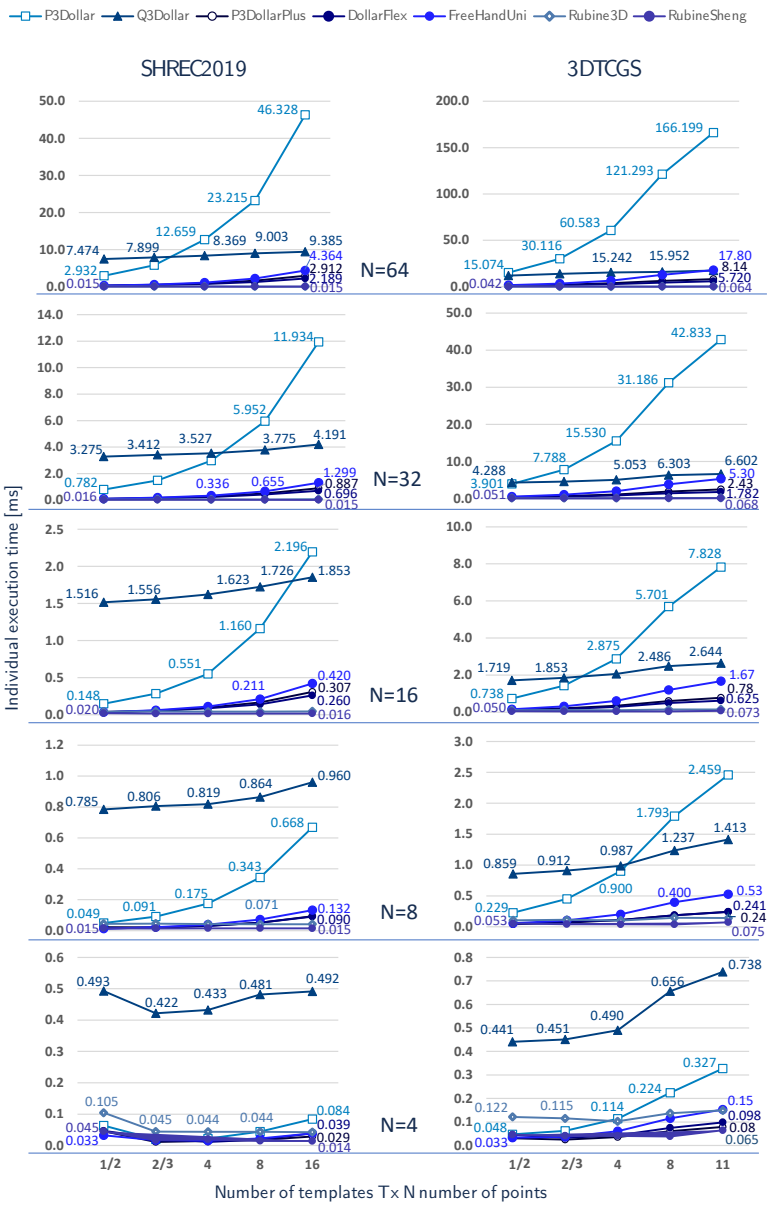


FIGURE 4.6: Execution times of all recognizers for two datasets (left: SHREC2019 [24], right: 3DTCGS [23]): individual times for N=4, ..., 64.

$\$P+^3$ is more accurate than all other recognizers with a very high significant difference, the effect size ranging from small to large. Similarly, the difference between $\$F$ and FH, $\$Q^3$ is not significant, but becomes significant over $\$P^3$ without any effect size, over $R3D$ and RS with a small and large effect size, and so forth.

TABLE 4.4: ANOVAs computed for the 7 RECOGNIZERS in the user-dependent scenario: G1=group 1, G2=group 2. For each dataset, three data are provided: the q value resulting from the ANOVA, the significance of the p value if any (***$p\leq$), and Cohen's d coefficient for effect size ((S)mall when $d\geq.02$, (M)edium when $d\geq.05$, (L)arge when $d\geq.08$, and (–) when there is no significant effect size ($d<.02$)).

Recognizer		**Dataset (q-stat, *p* value, Cohen's *d*)**				
G1	G2	Overall	Domain 1	Domain 2	Domain 3	Domain 4
$\$P+^3$	$\$F$	14.38,***, S	15.21,***, S	13.78,***, –	10.73,***, –	13.99,***, –
	FH	13.62,***, S	15.27,***, S	14.39,***, –	10.54,***, –	12.66,***, –
	$\$Q^3$	16.13,***, S	18.15,***, S	17.15,***, –	11.78,***, –	16.98,***, –
	$\$P^3$	19.50,***, S	21.63,***, S	21.10,***, –	14.04,***, –	21.16,***, –
	$R3D$	31.19,***, M	155.71,***, L	154.95,***, L	150.50,***, L	138.79,***, L
	RS	55.87,***, L	200.24,***, L	190.04,***, L	174.59,***, L	191.70,***, L
$\$F$	FH	0.76, *n.s.*	0.05, *n.s.*	0.61, *n.s.*	0.18, *n.s.*	1.32, *n.s.*
	$\$Q^3$	1.75, *n.s.*	2.94, *n.s.*	3.36, *n.s.*	1.05, *n.s.*	2.99, *n.s.*
	$\$P^3$	5.11,**, –	6.41,***, –	7.32,***, –	3.30, *n.s.*	7.17,***, –
	$R3D$	16.81,***, S	140.49,***, L	141.16,***, L	139.77,***, L	124.80,***, L
	RS	41.48,***, L	185.02,***, L	176.26, ***, L	163.85,***, L	177.70,***, L
FH	$\$Q^3$	2.51, *n.s.*	2.88, *n.s.*	2.75, *n.s.*	1.24, *n.s.*	4.31,*, –
	$\$P^3$	5.87,***, –	6.36,***, –	6.70, ***, –	3.49, *n.s.*	8.49,***, –
	$R3D$	17.57,***, S	140.44,***, L	140.55,***, L	139.96,***, L	126.13,***, L
	RS	42.25,***, L	184.97,***, L	175.64,***, L	164.04,***, L	179.03,***, L
$\$Q^3$	$\$P^3$	3.36, *n.s.*	3.47, *n.s.*	3.95, *n.s.*	2.25, *n.s.*	4.18,*, –
	$R3D$	15.06,***, S	137.55,***, L	137.80, ***, L	138.72,***, L	121.81,***, L
	RS	39.73,***, L	182.08,***, L	172.89,***, L	162.80,***, L	174.71,***, L
$\$P^3$	$R3D$	11.69,***, S	134.08,***, L	133.84, ***, L	136.46,***, L	117.63, ***, L
	RS	36.37,***, M	178.61,***, L	168.93,***, L	160.54,***, L	150.53,***, L
$R3D$	RS	24.67,***, S	44.52,***, S	35.09,***, S	24.08,***, –	52.90,***, S

Execution Time

The execution time curves of the four domains are depicted in Figure 4.7 and Figure 4.8, resp. In contrast to the user-independent scenario, RS remains the fastest recognizer in all conditions (M_{D1}=.030 ms, M_{D2}=.027 ms, M_{D3}=.030 ms, and M_{D1}=.042ms). The $\$P^3$ curve is located under the $\$Q^3$ for N=4 ($M_{\$P^3}$=.058 ms and for $M_{\$Q^3}$=.447 ms) and N=8 ($M_{\$P^3}$=.444 ms and $M_{\$Q^3}$=.853 ms).

The $\$P^3$ execution time curve progressively increases as N increases and it intersects with the $\$Q^3$ at different points, the intersection point is determined by T.

The execution time for this scenario is larger than for the user-independent scenario because of the method used in JavaScript to return the timestamp during the recognition. For the user-independent scenario, we used performance.now(), which is a high-performance method returning a value representing the time spent since the beginning of the program with a precision of up to one microsecond, whereas in the user-dependent scenario, we used date.now() which is a method returning the number of milliseconds elapsed since UNIX epoch, which limits the precision by rounding it to 1 millisecond.

User-dependent *vs.* User-independent Scenario

- The $\$P+^3$ recognizer remains the most accurate for the first domain (*i.e.,* the ten digits) in both user-dependent (M=98.45%) and user-independent scenarios (M=88.08%).
- For all values of N, a high recognition rate is obtained as soon as with one template and the best recognition rate for more templates.
- With two or more templates (T>2) used for training, $R3D$'s curve surpasses all the other curves except the $\$P+^3$ in the user-independent scenario and is competitive with other $-like recognizers in the user-dependent scenario.
- With less than three templates, $R3D$ is inaccurate, which explains why its overall rate is inferior to the others, except for RS.
- In the user-independent scenario, $-like recognizers belong to an envelope that progressively grows when N grows.
- In the user-dependent scenario, their curves are confounded most of the time.
- The recognition rates of the second domain, *i.e.,* the lowercase letters a-j, are quite similar to the first domain with a margin of about 1-2%.

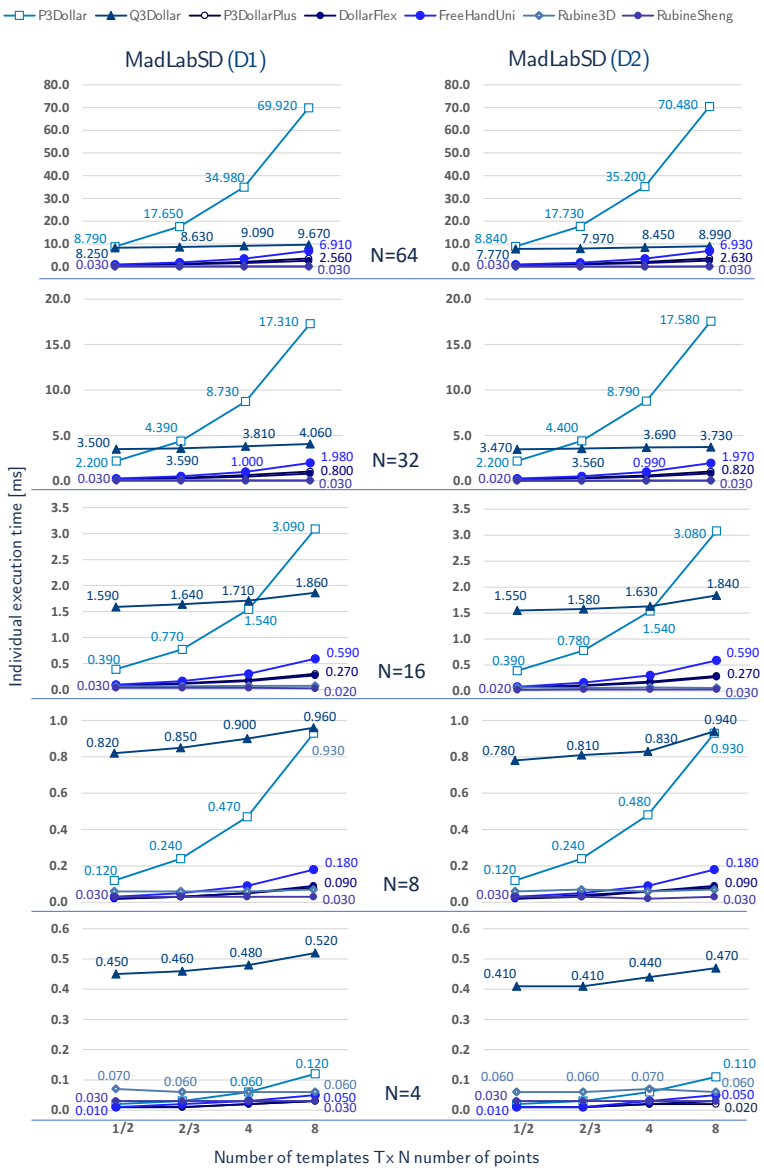


FIGURE 4.7: Recognition times of all recognizers for two datasets from MadLabSD (left: Domain 1, right: Domain 2 [49]): individual times for N=4..64 for the User-dependent scenario.

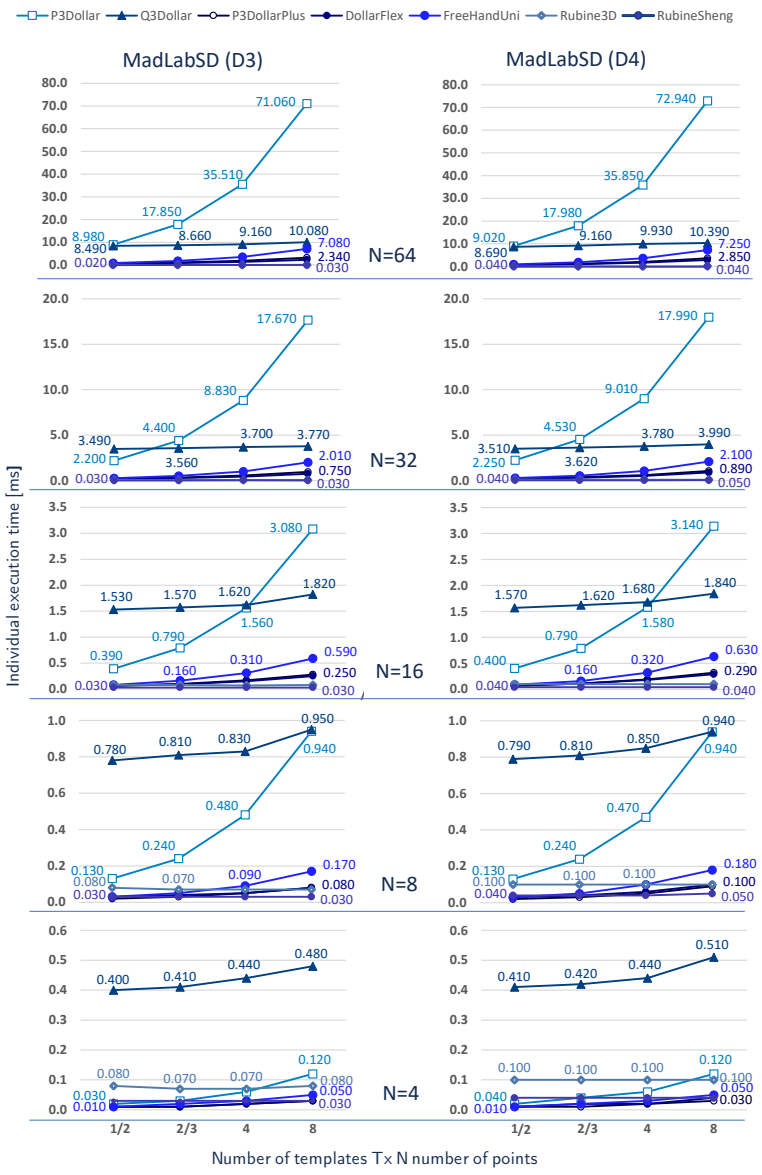


FIGURE 4.8: Recognition times of all recognizers for two datasets from MadLabSD (left: Domain 3, right: Domain 4 [49]): individual times for N=4..64 for the User-dependent scenario.

- For the third domain, *i.e.,* the simulation symbols, $-like recognizers are close to perfection in the user-dependent scenario with $\$P+^3$ being the champion in both cases (M=99.26% and M=94.45%, respectively).
- The results for the user-independent scenario are slightly superior to those obtained for the previous domains.
- However, the rates are very low for the fourth domain (*i.e.,* the CAD symbols) in the user-independent scenario (they are all below 66%), but still very good for the user-dependent scenario (between M=93.86% for $\$P+^3$ and M=90.68% for $\$P^3$).

For all four domains, the recognition rate significantly decreases when we are switching from the user-dependent scenario to the user-independent scenario, which was expected. Recognizing a 3D trajectory that is different from the templates used for training remains more demanding than re-identifying an already existing one. Figure 4.9 depicts the loss of recognition rate in terms of a difference of percentage from user-dependent to user-independent. $R3D$ wins the lowest averaged difference of percentage (M=11.75%), but its global recognition rate is below the $-like recognizers. The second place is occupied by $\$P+^3$, which undergoes the next lowest averaged difference of percentage (M=17.50% on four domains), followed by $\$F$, FH, $\$P+^3$, $\$P^3$, and RS.

In conclusion, $\$P+^3$ reasonably resists to user-independence while keeping the best recognition rate. The two first domains, *i.e.,* digits and letters, remain constant in terms of recognition loss for each recognizer, again with a loss of 12% for $\$P+^3$. Overall, the third domain benefits from the minimum loss, probably because of straightforward symbols, and the fourth domain suffers from the maximum loss, probably because of the most uncommon symbols.

4.4.3 Impact on Recognition Rate and Execution Time

We conducted a series of one-way ANOVAs using IBM SPSS V27 to evaluate the effect of the recognizer, number of templates T, and sampling N on recognition rate and execution time. When Levene's test [67] indicated unequal variances, we used Games-Howell post-hoc analysis. Otherwise, we used Tukey's HSD. Before analysis, the data underwent a Bonferroni type I correction.

Impact of Datasets on Recognition Rate and Execution Time

To determine the potential impact of the datasets (*i.e.,* SHREC2019, 3DTCGS, MadLabSD Domains 1–4 in the user-independent scenario and MadLabSD Domains 1–4 in the user-dependent scenario) on the recognition rate and the execution time, we computed another series of one-way ANOVAs (Table 4.5).

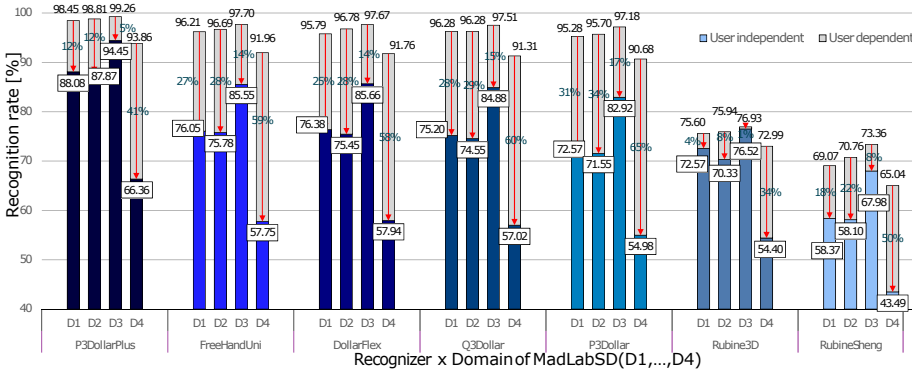


FIGURE 4.9: Difference of percentage in recognition rate loss for all recognizers on the MadLabSD dataset [49], all domains, when restricting the scenario from user-dependent to user-independent. Recognizers are sorted in decreasing order of their global recognition rate.

- The analyses showed a significant effect of the datasets on the recognition rate with a large effect size, but no significant effect on the execution time in both scenarios.
- A medium effect size in the user-independent scenario was observed on the $R3D$ recognition rate, but no significant effect on it and on RS recognition rate in the user-dependent scenario.
- Tukey's HSD and Games-Howell post-hoc analyses revealed that the recognizers got the lowest rate for Domain 4 in the two scenarios. For instance, there were significant differences for the $\$P+^3$ recognition rate in user-independent scenario between the different datasets, Domain 4 *vs.* SHREC2019 (M_{Diff}=-20.588,$^{***}p$<.001), Domain 4 *vs.* 3DTCGS (M_{Diff}=-17.923,$^{***}p$<.001), Domain 4 *vs.* Domain 1 (M_{Diff}= -21.720,$^{***}p$<.001), Domain 4 *vs.* Domain 2 (M_{Diff}= -21.512,$^{***}p$<.001), Domain 4 *vs.* Domain 3 (M_{Diff}=-28.092,$^{***}p$<.001), and Domain 4 *vs.* 3DMadLab (M_{Diff}=-20.588,$^{***}p$<.001).
- There was also a significant effect of the datasets on $R3D$ and RS execution time. The $R3D$ execution times are significantly different between each pair of datasets.

Impact of Number of Templates and Sampling on Recognition Rate and Execution Time

Both T and N have a significant impact on the recognition rate and execution time of most recognizers (Table 4.6):

TABLE 4.5: The effect of dataset on the recognition rate and the execution time with the effect size ((L)arge when $\eta^2 \geq .14$, (M)edium when $\eta^2 \geq .06$, (S)mall when $\eta^2 \geq .01$, and (—) when there is a null effect size).

Recognizer	**Effect**	**User independent**			**User dependent**		
		$F_{5,144}$	p	η^2	$F_{3,76}$	p	η^2
$\$P^3$	Datasets × Rates	21.75	***	.43 (L)	8.66	***	.25 (L)
	Datasets × Time	1.15	*n.s.*	.04 (–)	0	*n.s.*	0 (–)
$\$Q^3$	Datasets × Rates	21.87	***	.43 (L)	8.86	***	.26 (L)
	Datasets × Time	0.61	*n.s.*	.02 (–)	0	*n.s.*	0 (–)
$\$P+^3$	Datasets × Rates	41.48	***	.59 (L)	20.9	***	.45 (L)
	Datasets × Time	0.95	*n.s.*	.03 (–)	0	*n.s.*	0 (–)
$\$F$	Datasets × Rates	22.1	***	.43 (L)	8.45	***	.25 (L)
	Datasets × Time	0.93	*n.s.*	.03 (–)	0	*n.s.*	0 (–)
FH	Datasets × Rates	22.48	***	.44 (L)	8.28	***	.25 (L)
	Datasets × Time	1.55	*n.s.*	.05 (–)	0	*n.s.*	0 (–)
$R3D$	Datasets × Rates	3.64	**	.11 (M)	0	*n.s.*	0 (–)
	Datasets × Time	147.18	***	.84 (L)	357.48	**	.93 (L)
RS	Datasets × Rates	6.07	***	.17 (L)	.19 0	*n.s.*	.01 (–)
	Datasets × Time	89.39	***	.76 (L)	65.55	***	.72 (L)

- Regarding the $\$P^3$, $\$Q^3$, $\$P+^3$, $\$F$, and FH, we observe a significant effect of both factors, with a large effect size in the two scenarios. The Games-Howell analyses revealed that the execution time is significantly slower as the number of points increases.
- The $\$Q^3$ execution time is not significantly impacted by the number of templates in both scenarios. The effect of the number of templates on the execution time of $\$F$ is significant with a medium effect size, the Game-Howell post-hoc analysis revealed no significant difference between the execution times of each T in the user-dependent scenario.
- As per $R3D$ and RS, only T had a significant effect on the recognition rates with a large effect size. The Game-Howell confirms that the recognizers are more accurate with more templates.

TABLE 4.6: The effect of number of templates and points on the recognition rate and the execution time with the effect size ((L)arge when $\eta^2 \geq .14$,(M)edium $\eta^2 \geq .06$, (S)mall $\eta^2 \geq .01$, and (–) when null effect size).

Recognizer	Effect	User independent			User dependent		
		$F_{4,145}$	p	η^2	$F_{3,76}$	p	η^2
$\$P^3$	Templates × Rates	16.31	***	.31 (L)	16.80	***	.40 (L)
	Points × Rates	9.40	***	.21 (L)	5.98	***	.24 (L)
	Templates × Time	5.69	***	.14 (L)	4.04	***	.14 (L)
	Points × Time	25.27	***	.41 (L)	24.87	***	.57 (L)
$\$Q^3$	Templates × Rates	15.31	***	.30 (L)	15.44	***	.38 (L)
	Points × Rates	9.90	***	.21 (L)	6.22	***	.25 (L)
	Templates × Time	0.64	*n.s.*	.02 (–)	0.09	*n.s.*	0 (–)
	Points × Time	311.36	***	.90 (L)	1587.00	***	.99 (L)
$\$P+^3$	Templates × Rates	11.46	***	.24 (L)	7.99	***	.24 (L)
	Points × Rates	2.65	*	.07 (M)	2.16	*n.s.*	.10 (-)
	Templates × Time	6.26	***	.15 (L)	3.51	*	.12 (M)
	Points × Time	33.96	***	.48 (L)	36.02	***	.66 (L)
$\$F$	Templates × Rates	16.77	***	.32 (L)	13.89	***	.36 (L)
	Points × Rates	8.66	***	.19 (L)	6.88	***	.27 (L)
	Templates × Time	6.70	***	.16 (L)	3.52	*	.12 (M)
	Points × Time	34.95	***	.49 (L)	39.12	***	.68 (L)
FH	Templates × Rates	16.25	***	.31 (L)	13.83	***	.35 (L)
	Points × Rates	8.79	***	.20 (L)	7.26	***	.28 (L)
	Templates × Time	6.62	***	.15 (L)	4.76	**	.16 (L)
	Points × Time	22.65	***	.39 (L)	23.83	***	.56 (L)
$R3D$	Templates × Rates	89.56	***	.71 (L)	12047.47	***	1 (L)
	Points × Rates	0	*n.s.*	0 (–)	0	*n.s.*	0 (–)
	Templates × Time	0.42	*n.s.*	.01 (–)	0.13	*n.s.*	.01 (–)
	Points × Time	0.14	*n.s.*	0 (–)	0.1	*n.s.*	.01 (–)
RS	Templates × Rates	70.35	***	.66 (L)	2258.11	***	.99 (L)
	Points × Rates	0	*n.s.*	0 (–)	0.09	*n.s.*	0 (–)
	Templates × Time	1.21	*n.s.*	.03 (–)	0.37	*n.s.*	.01 (–)
	Points × Time	0.19	*n.s.*	.01 (–)	0.10	*n.s.*	.01 (–)

4.5 Conclusion

This chapter focuses on the comparative testing of template-based hand gesture recognizers on "unipath dynamic" gestures, specifically on 3D trajectories. These gestures involve single-point movements performed in the air. The goal is to instantiate the comparative testing systematic procedure to evaluate the recognition rate and execution time of recognizing these 3D trajectories using different gesture recognizers in different scenarios. The scenarios include an "intra-device" scenario, where the gestures are recorded with the same device

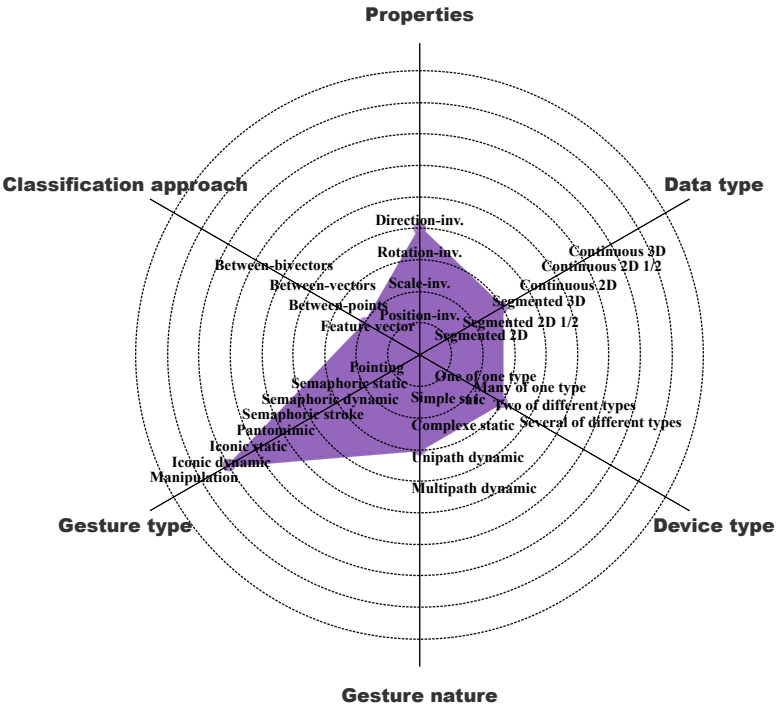


FIGURE 4.10: The $-like recognizers instantiation.

for different stages of the gesture recognition process. Additionally, there are user-independent and user-dependent scenarios.

To achieve the goal, four 2D stroke gesture recognizers ($\$P$, $\$P+$, $\$Q$, and Rubine) selected from the literature review are extended to the third dimension. The extended recognizers are: $\$P+^3$, $\$F$, FreeHandUni, $\$Q^3$, $\$P^3$, Rubine3D, and Rubine-Sheng.

Significant differences were observed among the recognizers by comparing individual recognition rates and execution times on different datasets, as well as aggregated results across all datasets. $\$P+^3$ is the top performer among the tested 3D trajectory recognizers, consistently achieving the highest recognition rate in almost all conditions. This statistical significance is only diminished in a few cases, such as when the number of templates is very limited. In addition to its robustness in user-independent scenarios, $\$P+^3$ also boasts excellent execution time in almost all scenarios. As a result, it is the preferred choice for efficiently recognizing 3D trajectories, particularly when compared to alternative recognizers. After the $\$P+^3$ recognizer, there is a recurrent pattern

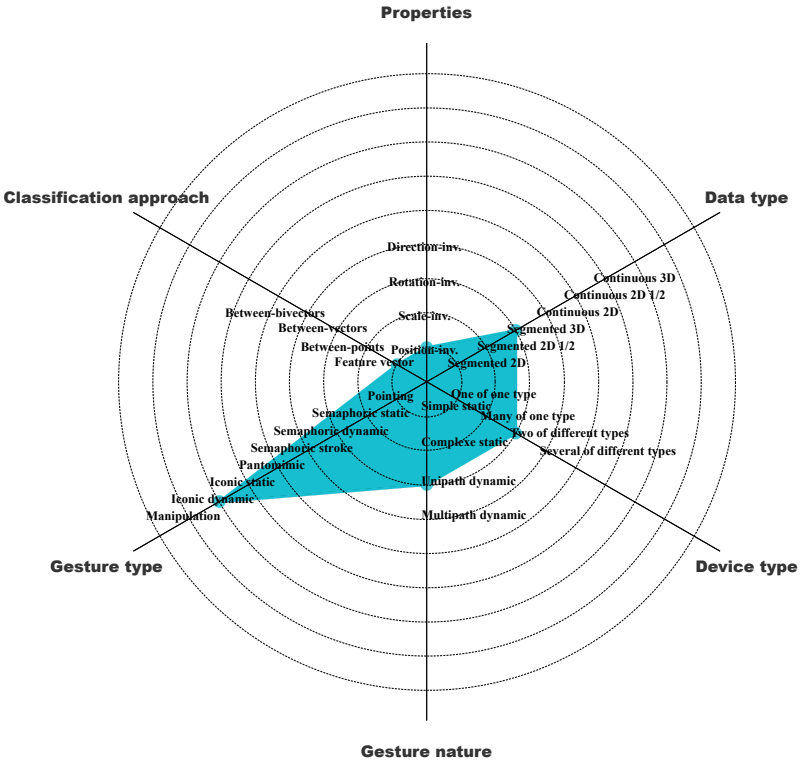


FIGURE 4.11: The Rubine3D and Rubine-Sheng instantiation.

in the ordering of subsequent recognizers both in terms of recognition rate and execution time: a first batch of other $-like recognizers appears with $\$F$, FH, $\$Q^3$, and $\$P^3$ and a second batch containing the two feature-oriented algorithms, *i.e.*, $R3D$ and RS. We notice that the recognizers share two common instantiations. The first instantiation, represented in Figure 4.11, is shared by the Rubine recognizers (Rubine3D and Rubine-Sheng) and uses the "Feature vector" classification approach. The second instantiation, represented in Figure 4.10, shows the values covered by the other $-like recognizers ($\$F$, FH, $\$Q^3$, $\$P^3$, and $\$P+^3$) and uses the "Between-points" classification approach.

Chapter 5

Use Case 2: Comparative Testing of Recognizers on Multipath Dynamic Gestures

The work discussed in this chapter was originally published in *HCI International 2022 - Late Breaking Papers. Multimodality in Advanced Interaction Environments.* Published in 2022 by Cham: Springer Nature Switzerland AG [86]. It is reproduced with permission from Springer Nature.

In the comparative testing of the "unipath dynamic" gestures, we observed the effectiveness of using comparative testing to evaluate recognizers in a *uni-device configuration*. The gesture sets used were created from different input device categories, including "Leap Motion controller" for SHREC2019 and 3DTCGS, and "SoftKinetic DepthSense DS325" for the 3DMadLabSD. This highlights the capabilities of the comparative testing method for template-based hand gesture recognizers. We further investigate this method by extending the exploration of the design space for hand gesture recognizer adapted to rapid prototyping (Section 2.3).

In this chapter, we will instantiate the procedure for comparative testing of a set of template-based gesture recognizers on "multipath dynamic" hand gestures recorded with a Leap Motion controller grouped into two gesture sets. We selected two recognizers from the set of recognizers we evaluated in Chapter 4 on "unipath dynamic" gestures, $\$P^3+$ (The same as in Section 4.1.6), $\$F$ (Section 4.1.4), we also selected two recognizers from the targeted literature review Jackknife [116] and $3\ Cent$ [23]. Jackknife has been also successfully used for dynamic gestures captured by a radar [109, 111], thus making it an eligible candidate for challenging gestures. In addition to two new state-of-the-art recognizers: $\$P^3$+X and PennyPincher3D, a recognizer with a "Between-Vectors" classification approach.

To ensure consistent comparison of gesture recognizers under identical conditions (*i.e.*, parameter values), comparative testing should examine the

impact of various factors on performance. These factors include the number of templates, the number of sampling points, the number of fingers, and their configuration with other hand parameters such as hand joints, palm, and fingertips. The procedure's results identify the configurations in which each recognizer is most accurate or fastest. Specifically, we focus on identifying the best gesture recognizers that have both high accuracy and low response time. We then define a method to determine the optimal conditions for designating these recognizers.

5.1 Recognizers

In this comparative evaluation procedure, we used template-based gesture recognizers, among them, 2 recognizers used in the comparative tests performed in Chapter 4: $\$P^3+$ ($\$P+^3$ adapted to multipath dynamic gestures), $\$F$, We added two recognizers from the targeted literature review in Chapter 2, Jackknife [116] and 3 $Cent$ [23]. Finally, we added two new state-of-the-art recognizers: P^3+X and PennyPincher3D.

5.1.1 $P^3+X Recognizer

A variant of *$P^3+* [122] that takes into account the direction-invariance by tracking conflicting templates (*i.e.*, templates of the same gesture but performed in different directions). If a gesture matches with a conflicting template, its direction is compared with the direction of each conflicting template, and the nearest one is chosen.

5.1.2 PennyPincher3D Recognizer

PennyPincher3D is an adaptation of the 2D recognizer *PennyPincher* [115]. The gestures are represented as a set of $N-1$ vectors linking between N equidistant points. The recognizer matches the candidate gesture with the template that maximizes a dissimilarity score, computed as the sum of the angles between the vectors. The computation relies on basic mathematical operations such as additions and multiplications. The gestures require just a resampling as prepossessing. This recognizer is scale- and position-invariant as most of the $-recognizers.

5.2 Datasets

5.2.1 SHREC2019

The **SHREC2019** dataset [24] includes a sequence of 3D points and a sequence of quaternions for each hand joint. However, we ignored the quaternions in

our experiments. The provided gestures represent one of five different gesture classes (Figure 5.1): “Cross”(X), “Circle”(O), “V-mark”(V), “Caret”(/\), and “Square”([]). It served in (SHREC) track, a contest on online gesture recognition to detect command gestures from hands’ movements in a virtual reality context. The proposed dataset consists of 195 3D movements performed by 13 participants with the whole hand. The dataset contains unsegmented gestures. The training set and the testing set were merged to create a unique dataset in which, unnecessary hand movements were removed from the gestures.

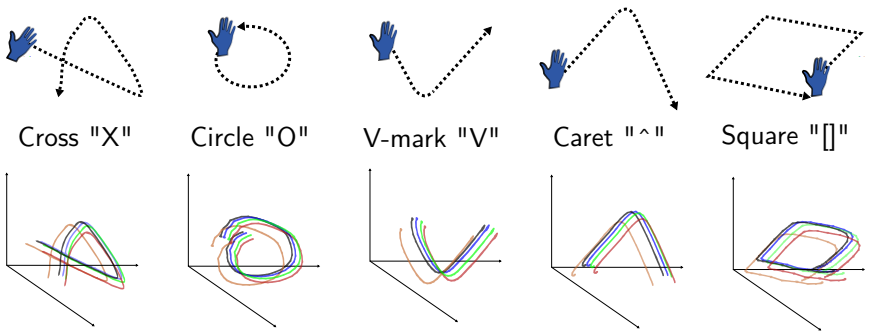


FIGURE 5.1: The SHREC2019 gesture classes and samples. [24]

5.2.2 Jackknife-LM

The **Jackknife-LM (Jackknife-LeapMotion)** dataset [116] contains 3D complex gestures of the hand and saved as 3D skeleton which is provided by the Leap Motion controller device. We used the segmented gestures composed of 360 samples of 9 different gesture classes, for example, “Fist Circles”, “Snip Snip”, “Explode”(Figure 5.2). It was used to test a rejection method of non-gesture sequences from a continuous data stream. While segmented gestures make up the training set, authors employ unsegmented sessions of samples in the test [116].

5.3 Design

The comparative testing performed aims at filling this gap in the literature. As such, we selected two available gesture sets for a number of reasons, including reproducibility. We evaluated the described recognizers following the typical method used in the literature to evaluate gesture recognizers [7, 8, 87, 124, 125, 140] in a user-independent scenario. We tested them on full hand gestures provided by the LMC skeletal hand model (Figure 5.3(a)) to show the efficiency of these recognizers under different conditions.

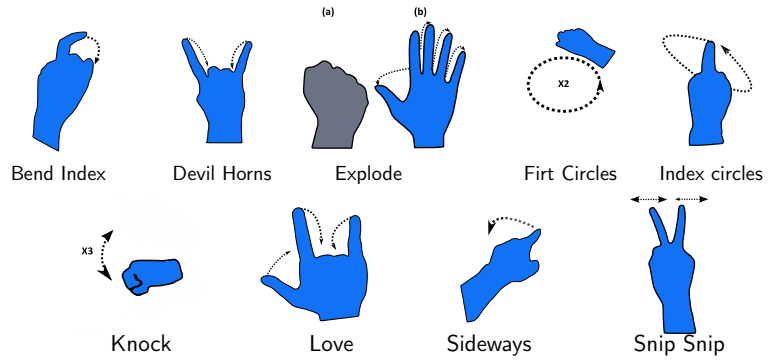


FIGURE 5.2: The Jackknife-LM gesture classes. [116]

5.3.1 Experiment

Our evaluation was within factors with five independent variables:

1. RECOGNIZER: nominal variable with 6 conditions, representing the various recognizers implemented for recognizing 3D gestures $\$P^3+$, $\$F$, Jackknife [116] and 3 $Cent$ [23]. And the two new recognizers which are described above: $\$P^3 + X$ and *PennyPincher3D*.

2. DATASET: nominal variable with 2 condition, representing the datasets considered, *i.e.*, SHREC2019 [24] and Jackknife-LM [116] described in Section 5.2.

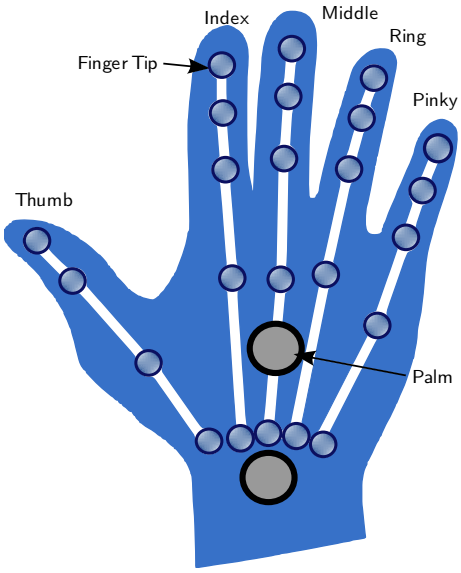
3. JOINTS: Nominal variable with 8 conditions, representing the hand joints used. Since the fingers contain several joints, we decided to use the information provided by the tips of the fingers (Figure 5.3(b)): **"1(P)"** = {Palm}, **"2(P+I)"** = {Palm, Index}, **"2(T+I)"**={Thumb, Index}, **"2(I+M)"** = {Index, Middle} , **"3(P+I+M)"** ={Palm, Index, Middle} , **"3(T+I+M)"** ={Thumb, Index, Middle}, **"5(AF)"** = {Thumb, Index, Middle, Ring, Pinky}, **"6(P+AF)"** = {Palm, Thumb, Index, Middle, Ring, Pinky}.

4. NUMBER OF TEMPLATES: numerical variable with 5 conditions, representing the number of templates per gesture for training: T=$\{1, 2, 4, 8, 16\}$.

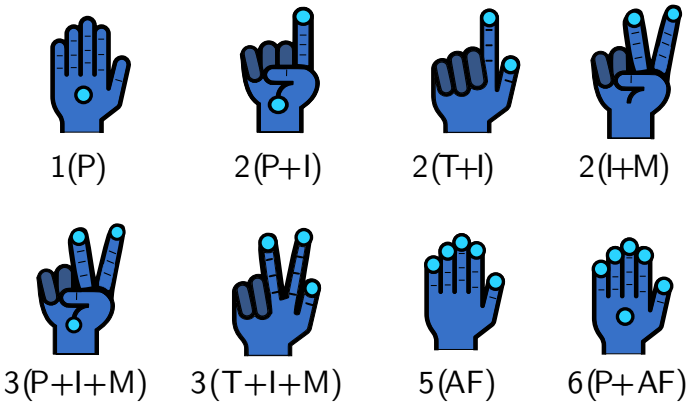
5. SAMPLING: numerical variable with 5 values representing the number of points per gesture: N=$\{4, 8, 16, 32, 64\}$.

Apparatus

We used a hexa-core Intel Core i7 2.20 GHz CPU and a Windows 10 Home Edition operating system. The RAM was 16 GB DDR4 memory with 2400 MHz.



(a) LMC hand model. [35]



(b) The hand joints selected for the testing.

FIGURE 5.3: Overview of the LMC Hand model and joints.

5.3.2 Procedure and Quantitative Measures

We compute the *recognition rate* (computed as the ratio of positive recognitions divided by the total number of trials) for the 6 (RECOGNIZER) $\times$ 2

(DATASET) = 12 basic configurations in the *user-independent scenario*. This scenario evaluates the recognition of gestures produced by users who are different from those used for training the recognizer. In this scenario, the basic configurations are refined depending on A, the number of joints, on T, the number of templates, and depending on N, the number of resampling points to train the recognizer. For each gesture class, a template is randomly selected from all participants and saved for testing. Then, a training set is created by randomly choosing T templates for each gesture class from the remaining users. These templates should be different from the ones previously selected for testing. The recognizer is then trained using this training set. This task is repeated R=100 times for each template in the T set.

5.4 Results

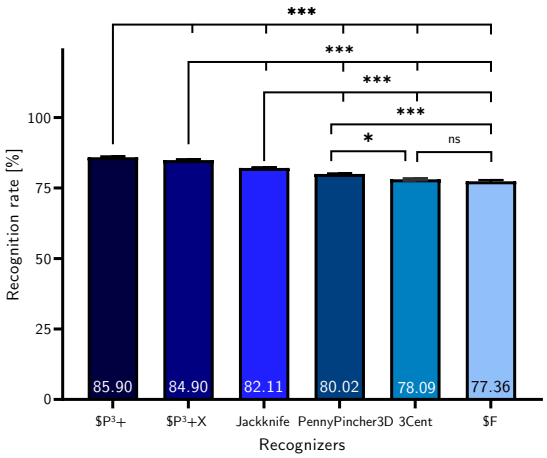
Overall, we conducted 2 (DATASET) × 8 (JOINTS) × 5 (SAMPLING) × 5 (NUMBER OF TEMPLATES) × 100 (repetitions) × 6 (RECOGNIZER) = 240,000 recognition trials for each dataset. Finally, the number of recognized gestures is averaged to calculate the recognition rate, which is then formatted as a percentage. Statistical computations are performed using GraphPad Prism.

5.4.1 SHREC2019 Dataset

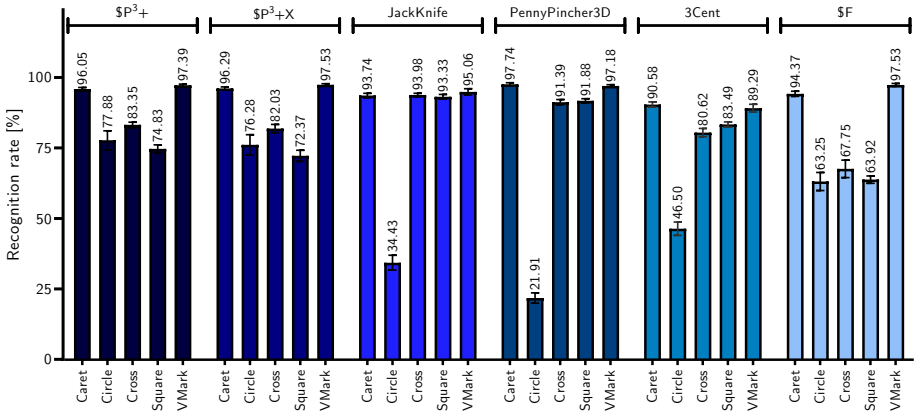
Overall Recognition Rate

Figure 5.4 displays the average recognition rate for all tests under all conditions, with the recognizers listed in descending order. The $\$P^3+$ has the highest average recognition rate (M=85.90%, SD=15.44%), followed by $\$P^3 + X$ and Jackknife with (M=84.90%, SD=16.45%) and (M=82.11%, SD=14.06%), respectively. The average recognition rate of *PennyPincher3D* is (M=80.02%, SD=12.38%). For the last two recognizers, the average rates do not exceed 80%: 3 $Cent$ has an average recognition rate of (M=78.09%, SD=18.12%) and $\$F$ has an average recognition rate of (M=77.36%, SD=19.27%).

We performed four normality tests on the RECOGNIZER variable: the Anderson-Darling K-sample test, D'Agostino's K2 test and Kolmogorov-Smirnov's KS. None of the recognition rates followed a normal distribution. Due to the large number of samples, the Shapiro-Wilk test could not be used to assess normality. Then, we performed a Kruskal-Wallis test with Dunn's multiple comparisons on the measures. The overall difference between the recognizers is very highly significant (p<.001***). Figure 5.4 shows that $\$P^3+$ is the most accurate Recognizer, significantly outperforming all other recognizers. $\$P^3+$ is better than $\$F$ with a highly significant difference (Z=48.84, p<.001***) and



(a) Average recognition Rate



(b) The overall recognition rate per gesture class

FIGURE 5.4: Recognition rates of all recognizers for the SHREC2019 for all conditions. Error bars show a confidence interval of 95%.

significantly better than Jackknife (Z=27.86, p<.001***). Also, \$$P^3$+ significantly outperforms \$$P^3 + X$ (Z=5.247, p<.001***). However, \$$F$ is not significantly different from 3 $Cent$ (Z=1.928, $n.s.$), while this later is significantly outperformed by *PennnyPincher3D* (Z=3.055, p<.05*).

Furthermore, the overall recognition rate per gesture class indicates that:

- The gestures "Caret"(/\) and "V-mark"(V) have average rates greater than or equal to 90%. However, similarly to the average recognition rates

of the other classes, these rates vary among the different recognizers.

- The "Cross" (X) gesture is better recognized than the "Square" ([]) and the "Circle" (O) by the $\$F$, $\$P^3+$ and $\$P^3 + X$ recognizers.
- The best average recognition rate for "Cross" (X) gesture class is achieved by Jackknife (M=93.98%); this value goes down respectively to (M=83.35%) for $\$P^3+$ and to (M=67.75%) for $\$F$ with a difference of 10.53% and 26.45%.
- For the Jackknife and *PennyPincher3D* recognizers, the "Square"([])) and "Cross"(X) classes have similar average recognition rates.
- On the contrary, the gesture class "Circle"(O) has a very low average recognition rate for Jackknife (M=34.43%), *PennyPincher3D* (M=21.91%) and 3 $Cent$ (M=46.50%).

From the detailed recognition rate for each condition, some recognizers have low or almost non-existent recognition rates that meet the expectation of end users ($\tau \geq 90\%$) [76, 134]. These include *PennyPincher3D*, 3 $Cent$, and $\$F$. On the other hand, $\$P^3+$ and $\$P^3 + X$ consistently achieve recognition rates of $\tau \geq 90\%$. Since the goal is to conduct a comparative test of recognizers using the SHREC2019 dataset, we relaxed the aforementioned constraint and refined the results to require a recognition rate of $\tau \geq 80\%$.

Recognition Rates by Number of Joints

Figure 5.5 illustrates the average recognition rates for each value of condition JOINTS (A). Each recognition rate represents the average of the recognition rates of all conditions (T) and (N). The values range from M=75.58% for $\$F$ in $A = 2\ (I + M)$ to M=87.04% for $\$P^3+$ in $A = 3\ (P + I + T)$. Generally, the recognition rates do not vary significantly across different values of (A). $\$P^3+$ achieves the highest average recognition rate for $A = 3\ : (P + I + T)$ (M=87.04%) while $\$F$ has the lowest average rate for $A = 2\ : (M + I)$ (M=75.58%).

Although, there are some conditions such as $A = 2\ (I + M)$ and $A = 3\ (P + I + T)$, where the average rate of $\$P^3 + X$ exceeds that of $\$P^3+$ (*e.g.* $A = 2\ (I + M)$ and $A = 3\ (P + I + M)$). Similarly, for some values of (A), 3 $Cent$ has lower recognition rates than $\$F$ (*e.g.* $A = 1\ (P)$ and $A = 2\ (T + I)$).

As a way to help understand the behavior of the recognizer with respect to both the number of points as well as the number of joints. Figure 5.6 is an overview of recognizers ranked by recognition rate. This ranking is presented along two axes, the number of joints (A) and the number of sampling points (N). It shows which recognizers are better than the others for each pair of

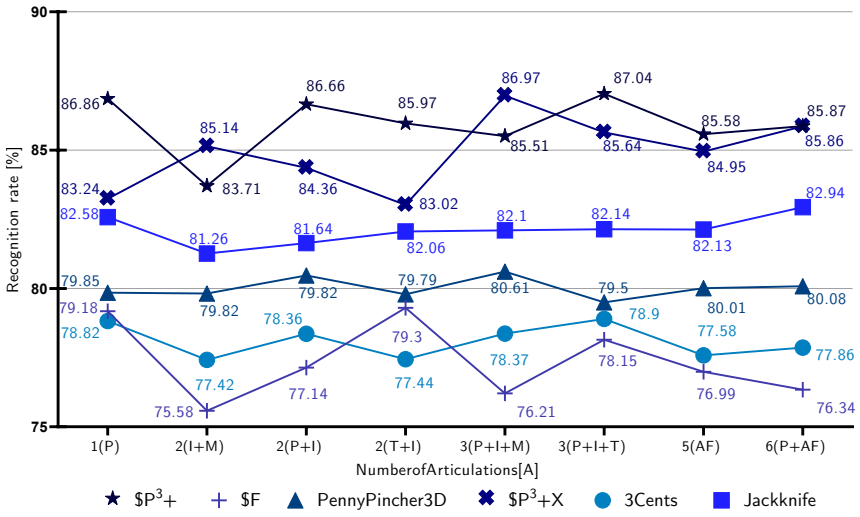


FIGURE 5.5: Recognition rates of all recognizers for the SHREC2019 dataset [24], the plot shows average rates by joint $A = \{1\,(P), ..., 6\,(P + AF)\}$.

conditions (A, N). To respect the restriction on the recognition rate that we have defined above, all recognizers in this figure comply with two criteria:

1. The recognizer must achieve a recognition rate $> 80\%$.
2. The recognition rate must be the highest value among the conditions of (T).

Globally, the position of the recognizers varies a lot according to the defined conditions, and few recognizers keep a constant ranking by varying the number of joints or the number of points (*e.g.*, *PennyPincher3D* for N=16 and N=32). For N=4, we observe that some recognizers do not appear, because of the first criterion defined, except for 3 $Cent$, which has at least a recognition rate that respects the criterion. We notice that for several conditions, the $\$P^3+$ recognizer takes the first place, except for the condition where $A = 2\,(P + I)$ and $A = 2\,(I + M)$ where $\$P^3 + X$ is designated as the best recognizer. In contrast to $\$P^3+$, the *PennyPincher3D* often achieves the lowest recognition rate and is ranked last.

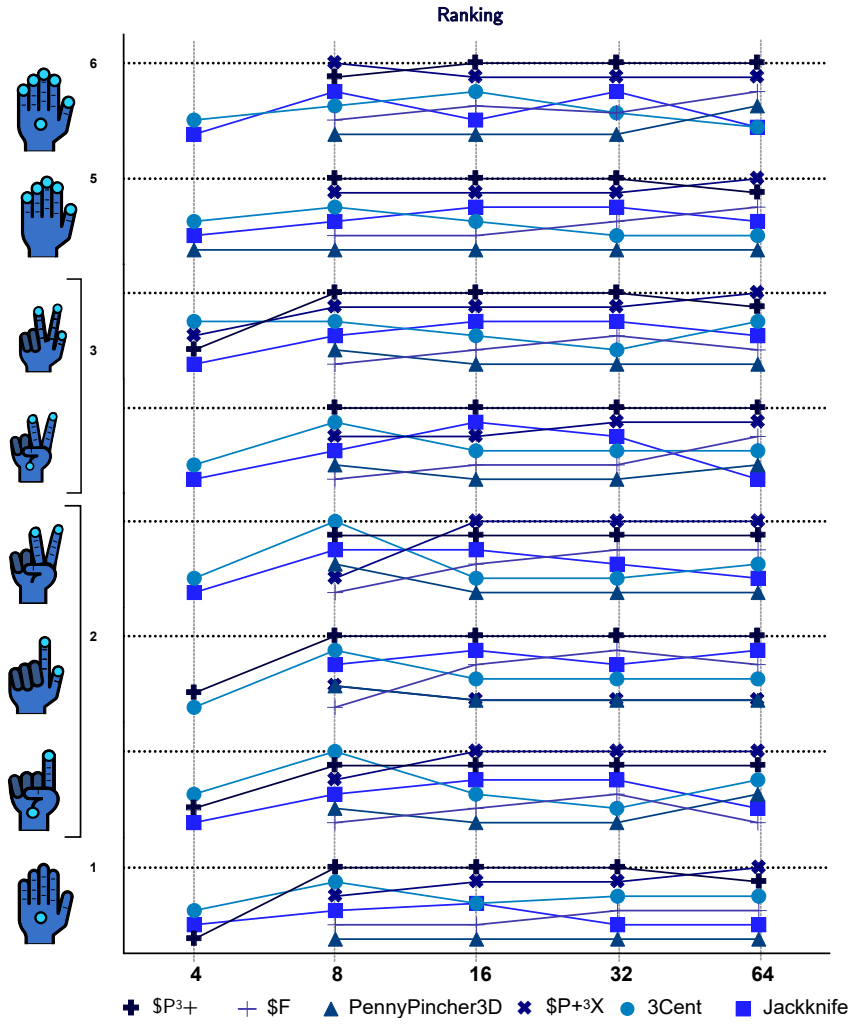


FIGURE 5.6: Ranking of the best individual recognition rates above 80% by the number of joints (A) and the number of points (N) for the SHREC2019 dataset.

Recognition Rate for the Optimal Conditions

To evaluate the efficiency of the recognizers, we planned to test them in a precise scenario where the conditions are well-defined. For this reason, we determine which values of the variables JOINTS (A), SAMPLING (N), and

NUMBER OF TEMPLATES (T) allow an optimal comparison of the different recognizers in the recognition rate measure. We completed a de Borda ranking of all combinations of conditions for each recognizer, and the result is reported in Table 5.1. The de Borda ranking selects the best combination of conditions, the one with the highest overall Borda score. This implies that the elected combination of values has a high recognition rate with many recognizers. Furthermore, we notice that many conditions share the same rank and score in the ranking for each recognizer. This situation is due to a defined sensitivity value of 2% in the recognition rate. We determine the case of a tie between the last ranked combination of conditions and the following combination to be ranked on a specific recognizer, if the difference between their recognition rates is less than the defined sensitivity value. According to Table 5.1, the result of the Borda ranking gives this combination of conditions ($A = 2\ (T + I)/\ N$=32/ T=16) as the best.

TABLE 5.1: Global position of the recognizers for each condition and overall, according to de Borda method across all recognizers (the higher, the better).

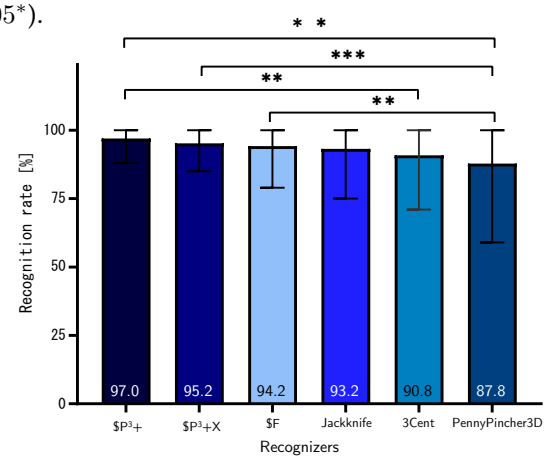
Condition				**Recognizer (#Rank, Score)**					
A	N	T	**Overall**	$\$P^3+$	$\$F$	*PP3D*	Jackknife	$\$P^3 + X$	3 $Cent$
2 (T + I)	32	16	#1, 1187	#1, 200	#1, 200	#2, 196	#1, 200	#1, 200	#2, 191
3 (P + I +M)	32	16	#2, 1180	#1, 200	#3, 184	#2, 196	#1, 200	#1, 200	#1, 200
2 (P + I)	32	16	#3, 1177	#1, 200	#2, 197	#2, 196	#2, 193	#1, 200	#2, 191
3 (P + I +M)	16	16	#4, 1175	#1, 200	#3, 184	#1, 200	#1, 200	#1, 200	#2, 191
2 (I + M)	32	16	#5, 1166	#2, 186	#1, 200	#2, 196	#2, 193	#1, 200	#2, 191
1 (P)	64	16	#6, 1161	#2, 186	#2, 197	#3, 185	#2, 193	#1, 200	#1, 200
2 (T + I)	16	16	#7, 1158	#1, 200	#1, 200	#3, 185	#1, 200	#2, 182	#2, 191
5 (AF)	32	16	#8, 1157	#1, 200	#2, 197	#2, 196	#2, 193	#1,200	#3, 171
6 (P + AF)	32	16	#9, 1153	#1, 200	#3, 184	#3, 185	#2, 193	#1, 200	#2, 191
3 (P + I + T)	32	16	#10, 1148	#1, 200	#2, 197	#3, 185	#2, 193	#2, 182	#2, 191
...	...	...	...	...	...	...	...	...	...

Based on de Borda's results, we evaluate the recognizers on the SHREC2019 dataset under the elected conditions. The upper part (a) of Figure 5.7 shows the average recognition rate for the defined conditions. According to these results, the $\$P^3+$ has the best average recognition rate (M=97.00%, SD=7.18%), followed by the $\$P^3 + X$ (M=95.20%, SD=8.59%) and the $\$F$ (M=94.20%, SD=9.55%), while *PennyPincher3D* is the least accurate recognizer (M=87.80%, SD=13.30%), for the other two recognizers, the average rate of Jackknife (M=93.2%, SD=10.34%) and 3 $Cent$ (M=90.80%, SD=13.16%).

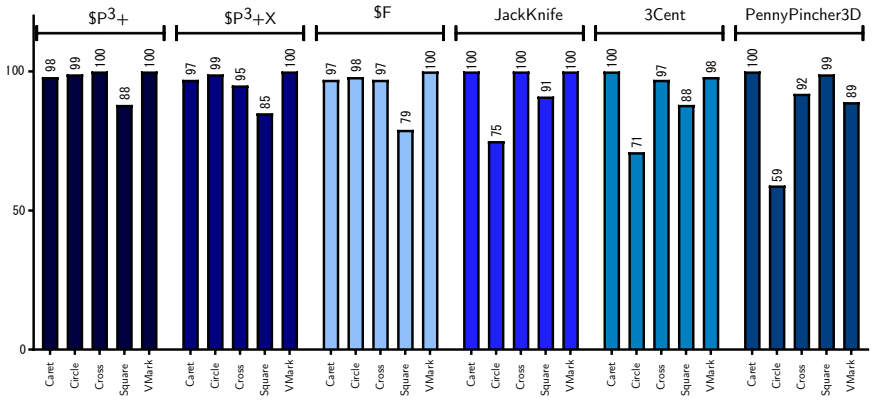
As for the overall recognition rate on all conditions in part 5.4.1, we calculated the four normality tests: K sample's Anderson-Darling test, Shapiro-Wilk W test, D'Agostino's K2 test and Kolmogorov-Smirnov's KS.

None of the recognition rates followed a normal distribution. Then, we calculated a Kruskal-Wallis test with Dunn's multiple comparisons. The high

difference between the recognizer $\$P^3+$ and *PennyPincher3D* (10%) is statistically highly significant (Z=5.736, p<.001***). Same thing for $\$P^3 + X$ that is significantly better than *PennyPincher3D* (Z=4.420, p<.001***). Moreover, $\$P^3+$ is significantly more accurate than 3 $Cent$ by 6.2% (Z=3.699, p<.01**). $\$F$ is superior by 6.4% to *PennyPincher3D* with a highly significant difference (Z=3.788, p<.01**). However, $\$P^3+$ is not significantly better than Jackknife (Z=2.581, *n.s.*), while this later is significantly better than *PennnyPincher3D* (Z=3.156, p<.05*).



(a) Average recognition Rate



(b) Number of recognized gestures per class over 100

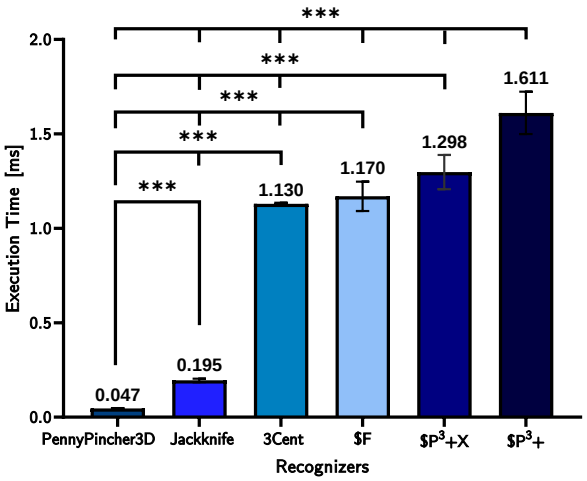
.

FIGURE 5.7: Recognition rate (a) and the number of recognized gestures per class over 100 trials (b) of all recognizers for the user-independent scenario, for the optimal conditions defined by the de Borda ranking: A=2 ($T + I$), N=32, T=16. Error bars show a confidence interval of 95%.

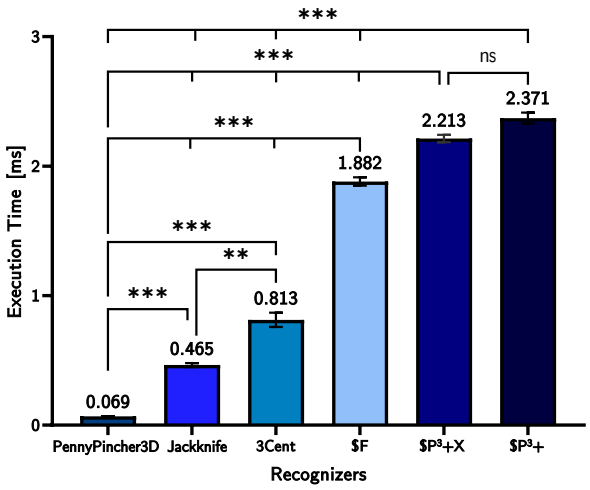
In the bottom part of the same figure (Figure 5.7(b)), the bar chart shows the number of recognized gestures per gesture class on 100 repetitions, noting that all recognizers achieve a perfect score for at least one gesture class. Among the gesture classes in question, there is the "Caret"(/\) for *PennyPincher3D* and 3 $Cent$ recognizers, and also the "V-mark"(V) gesture class for $\$P^3 + X$, $\$F$ recognizers. While $\$P^3+$ achieves an accuracy rate of ($\frac{100}{100}$) for two gesture classes : the"V-mark"(V) and the "Cross"(X). Although Jackknife achieves a flawless recognition for three gesture classes: "Caret"(/\) , "Cross"(X) and "V-mark"(V), its recognition of the "Circle"(O) gesture class is weak ($\frac{75}{100}$). This difficulty is also encountered by *PennyPincher3D* for the same gesture class ($\frac{59}{100}$) as for 3 $Cent$. However, the latter achieves the best accuracy rate for the "Square"([]) gesture class ($\frac{99}{100}$) which means that *PennyPincher3D* is well designed to recognize this gesture class, especially when the other recognizers perform less well in recognizing it.

Overall Conditions Execution Time

In Figure 5.8(a), the average execution times of the different recognizers for all conditions varied between (M=0.047 ms, SD=0.060 ms) for the *PennyPincher3D* and (M=1.611 ms, SD=3.328 ms) for the $\$P^3+$. The execution times do not follow a normal distribution. We calculated a Kruskal-Wallis test which indicates a significant difference in the execution times of the recognizers. After that, the Dunn's multiple comparisons test shows significant differences for all pairs of recognizers except between 3 $Cent$ and $\$P^3 + X$. The *PennyPincher3D* ($M$=0.047ms, SD=0.060ms) is significantly faster than other recognizers; It is significantly better than Jackknife (M=0.195 ms, SD=0.2457 ms) with a difference of $148\mu s$ (Z=73.62, p<.001***), and outperforms significantly the slowest recognizer $\$P^3+$ (Z=147.7, p<.001***).



(a) Execution times for all conditions.



(b) Exec. times for the optimal conditions.

FIGURE 5.8: Average execution times by recognizer. Error bars show a confidence interval of 95%.

Overall Execution Time by Number of Joints

The graphs in Figure 5.9 shows the execution times according to the values of the variable JOINTS (A). The smallest value in the graphic is the averaged execution time of *PennyPincher3D* for the condition $A = 1\ (P)$ (M=0.017 ms, SD=0.020 ms), while the longest execution time is achieved by $\$P^3+$ for $A = 6\ (P + AF)$ (M=3.204 ms, SD=5.936 ms). The *PennyPincher3D* is the fastest recognizer and is ahead of the Jackknife. According to the results, we distinguish two groups of recognizers with regard to their execution times. The one formed by Jackknife (M=0.195 ms, SD=0.246 ms) and *PennyPincher3D*, with a small slope, are not much affected by the variation in the number of articulations. The second group consisting of 3 $Cent$(M=1.130 ms, SD=0.683 ms), $\$F$ (M=1.170 ms, SD=3.328 ms), $\$P^3 + X$($M$=1.298 ms, SD=2.731 ms) and $\$P^3+$ in order from fastest to slowest recognizer; All the recognizers are significantly impacted by the variation.of Number of Joints (p<.001***).

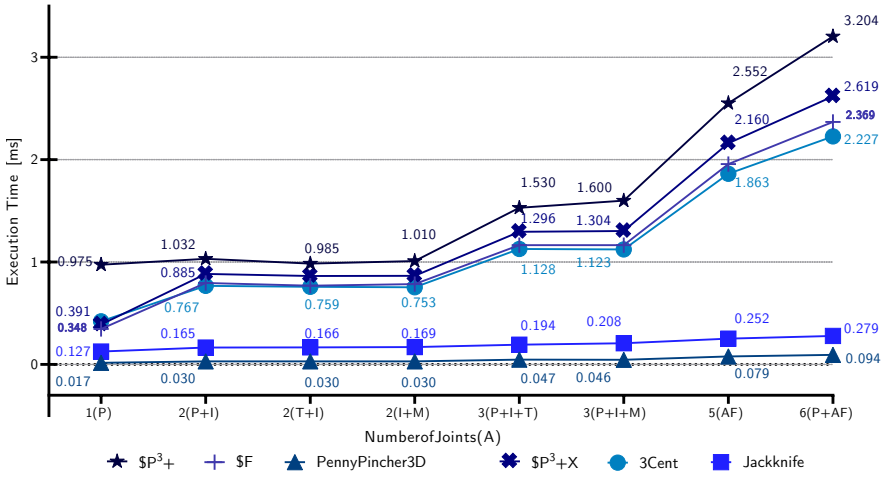


FIGURE 5.9: Average execution times by recognizer per number of joints (A).

Execution Time for the Best Condition

Figure 5.8(b) shows the execution time of the recognizers for the best condition defined by the de Borda method. The recognizers appear in the same order as in the Figure 5.8(a). *PennyPincher3D* remains the fastest (M=0.069 ms, SD=0.005 ms) and $\$P^3+$ the slowest (M=2.371 ms, SD=0.213 ms). The Kruskal-Wallis shows significant differences between all the recognizers except between the $\$P^3+$ and $\$P^3 + X$ (Z=1.910, $n.s.$).

5.4.2 Jackknife-LM Dataset

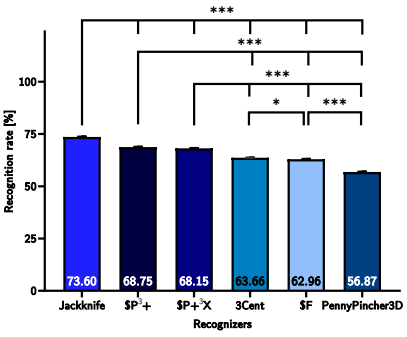
Overall Recognition Rate

The Jackknife-LM dataset has more gesture classes than SHREC2019, mainly complex gestures where different joints can move independently of the hand movement. Figure 5.10(a) sums up the averaged recognition results of each of the recognizers for all tests under all conditions. The results give an overview of the recognition with this dataset which are different from the results obtained for the SHREC2019. In general, none of the recognizers went beyond 80%. The Jackknife recognizer (M=73.60%, SD=19.75%) takes the lead and outperforms significantly other recognizers, followed by the $\$P^3+$ (M=68.75%, SD=19.73%), which is slightly better than $\$P^3 + X$ (M=68.15%, SD=20.51%) by (0.60%). They are followed by 3 $Cent$ and $\$F$ with respectfully (M=63.66%, SD=18.29%) and (M=62.96%, SD=20.12%) which are significantly better than *PennyPincher3D* the least efficient of the recognizers (M=56.87%, SD=19.99%).

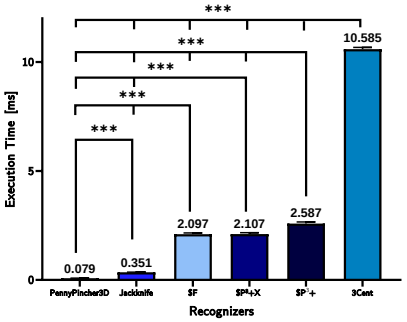
Three gesture classes ("FistCircles", "Knock" and "Sideways") are well recognized for all the recognizers (*i.e.*, $\tau \geq 80\%$ - Figure 5.10(c), bottom). An exception to this is *PennyPincher3D* where the "Knock" gesture has an average recognition rate of (M=62.77%). However, the "Love", "BendIndex", "DevilHorns", and "SnipSnip" have the worst average recognition rates for most recognizers (*i.e.*, $\tau \leq 60\%$), except for Jackknife, where the "SnipSnip" and "BendIndex" are above 70%. These results indicate that for many recognizers, many conditions perform very well with gestures where the fingers remain static, regardless of hand movements. Whereas the recognition rate drops for many conditions with gestures that include finger movements.

Recognition Rates by Number of Joints

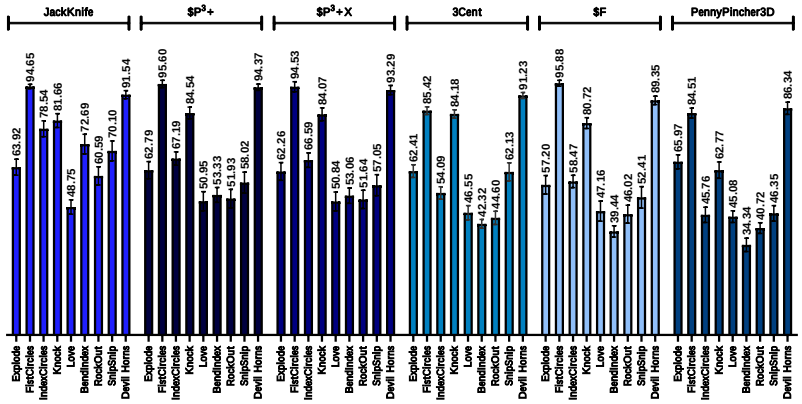
With regard to the average recognition rates by articulation in Figure 5.11, the lowest recognition rate is M=39.78% for $\$F$ in $A = 1\ (P)$, while the highest recognition rate is M=87.78% for $\$P^3+X$ in $A = 6(P+AF)$. The Jackknife performs better than other recognizers for the conditions with a reduced number of articulations ($A \leq 3$), and it is joined by $\$P^3+$ and $\$P^3+X$ at $A = 5(AF)$. The average recognition rate increases as the number of joints increases for all of the recognizers, except for some specific cases, such as for 3 $Cent$, *PennyPincher3D*, and $\$P^3+$, whose rate drops between $A = 5\ (AF)$ and $A = 6\ (P + AF)$ (*e.g.* a drop of 2.6% for 3 $Cent$). However, the recognition rates of $\$P^3+X$, $\$P^3+$, and $\$F$ under conditions with the same number of joints vary depending on how the joints are configured (*e.g.*, A=2$(P + I)$, A=2$(T + I)$, A=2$(I + M)$). Therefore, some joints are more relevant than others as they carry more information about gestures.



(a) Recognition rates for all conditions.



(b) Execution times for all conditions.



(c) Overall recognition rates for all conditions by gesture class.

FIGURE 5.10: Execution times of all recognizers for the Jackknife-LM for the user-independent scenario. Error bars show a confidence interval of 95%.

Overall Conditions Execution Time

Figure 5.10(b) shows the average execution times for all conditions, representing the execution time for the different recognizers. The execution time of *PennyPincher3D* (M=0.079 ms, SD=0.108 ms) differs significantly from that of 3 $Cent$ (M=10.585 ms, SD=5.911 ms). The execution time includes both preprocessing and recognition time. In the case of 3 $Cent$, the excessive execution time is attributed to the resampling function that uses the Cubic Spline interpolation method. The execution times of the other recognizers, $\$P^3+$ (M=2.587 ms, SD=4.371 ms), $\$F$ (M=2.097 ms, SD=4.184 ms), and $\$P^3 + X$ (M=2.107

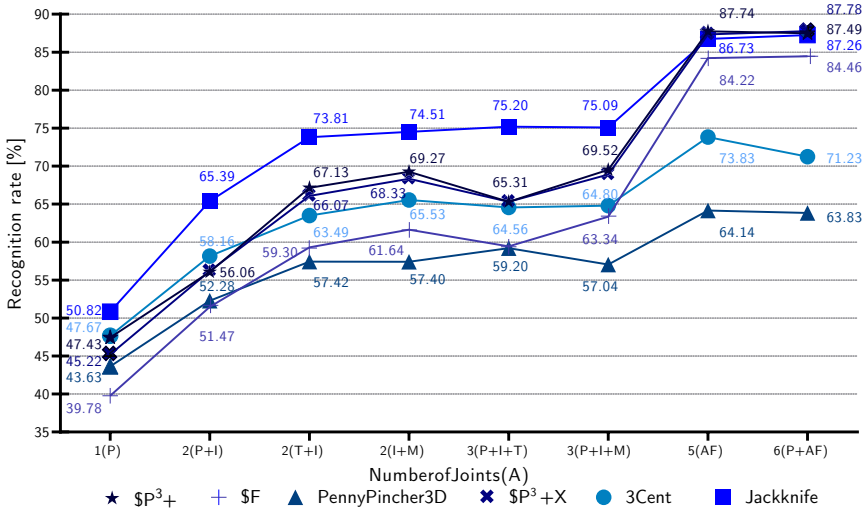


FIGURE 5.11: Recognition rates of all recognizers for the Jackknife-LM dataset [116], the plot shows average rates by joint $A = \{1\,(P), ..., 6\,(P + AF)\}$.

ms, SD=4.371 ms), are similar, with no significant difference between the latter two (Z=1.772, $n.s.$). Jackknife, which has the highest recognition rate, is also a fast recognizer (M=0.351 ms, SD=0.508 ms), making it suitable for this dataset.

5.5 Conclusion

The purpose of this use case is to compare template-based hand gesture recognizers on the "multipath dynamic" gestures, specifically focusing on the LMC-based gestures. These gestures are represented as a set of hand joint trajectories. The goal is to evaluate six gesture recognizers on two LMC-based datasets, which consist of simple and complex gestures (*i.e.*, SHREC2019 and Jackknife-LM). The evaluation is done in a user-independent scenario and similar to the comparative testing in Chapter 4 in an "intra-device" configuration where the gestures are recorded with the same device for different stages of the gesture recognition procedure.

Based on the design space defined in Section 2.3, we observe that for this comparative testing, the recognizers have two common instantiations. The first instantiation, shown in Figure 5.13, is shared by the JackKnife and PennyPincher3D recognizers, which are based on the "Between-Vectors" classification approach. The second instantiation, shown in Figure 5.12, shows the

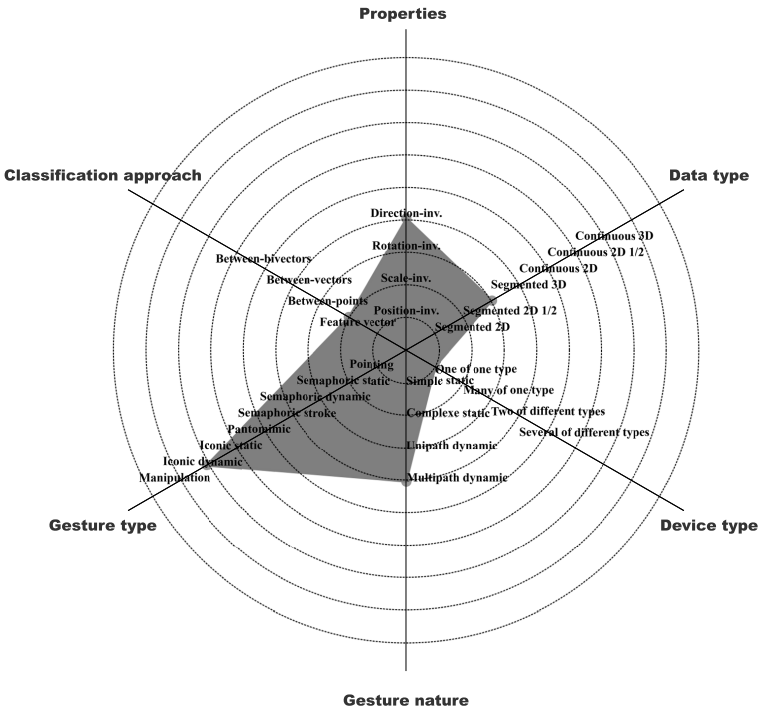


FIGURE 5.12: The $-like recognizers instantiation.

values covered by the other $-like recognizers ($\$F$,$\$P^3+$, *3 Cent*, and $\$P^3 + X$), which rely on the "Between-points" classification approach.

For the SHREC2019 dataset, $\$P^3+$ achieved the best recognition rate and outperformed the other recognizers. This was confirmed under particular optimal conditions. However, $\$F$ is the worst for this dataset. It was also observed that certain recognizers are better suited for recognizing specific gestures compared to others.

For the Jackknife-LM dataset, the Jackknife recognizer achieved good overall recognition rates under certain conditions but failed to meet the high accuracy requirement for many other conditions. The performance of recognizers was found to be influenced by the nature of the gesture being performed. Additionally, some recognizers were slower in processing longer gestures, making them less interesting.

Interestingly, the recognizer using the "Between-vectors" classification approach, PennyPincher3D, did not perform better than some other recognizers using the "Between-points" classification approach such as $\$P^3+$ or the other

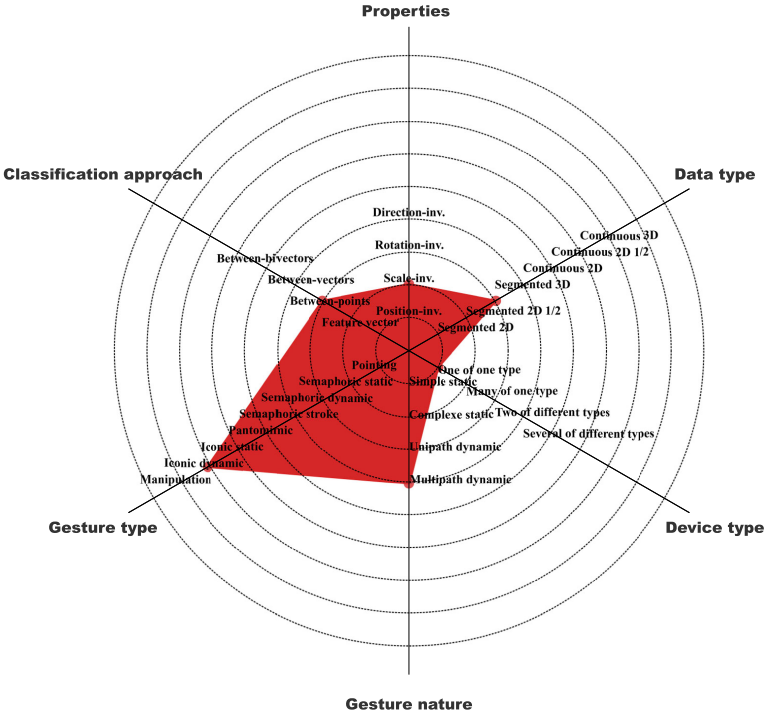


FIGURE 5.13: The JackKnife and PennyPincher3D instantiation.

recognizer relying on "Between-vectors", Jackknife, in terms of recognition rate. However, PennyPincher3D was the fastest recognizer among those evaluated.

This comparative testing demonstrates the potential benefits of the systematic procedure for the comparative testing method and provides valuable insights for designers and researchers seeking a reliable and efficient recognizer to meet their needs.

Chapter 6

QUANTUMLEAP Extension: Testing Module

In the previous chapters, two separate experiments were carried out to evaluate the recognition rate and execution time of template-based 3D hand gesture recognizers. A reasonable range of datasets was used to ensure a robust evaluation, including various classes of gestures of different natures (*i.e.*, "unipath dynamic" and "multipath dynamic" gestures). The selected recognizers' evaluation included user-dependent and user-independent scenarios for several gesture sets. The findings from these evaluations provide valuable insights for researchers and practitioners, enabling them to select the most suitable algorithm for their specific context of use while considering performance aspects. To accomplish this, there are few, if any, evaluation tools available that automate and support the systematic testing procedure of 3D hand gesture recognizers in multi-device configurations.

However, no tool with complex evaluation functions currently includes multiple recognizers, datasets, scenarios, and parameters. For this reason, we have implemented a testing module, a support tool that automates the evaluation of 3D gesture recognizers on multiple gesture sets. This module can be customized with different parameters. The core of this testing module is inspired by a part of the software architecture of the QuantumLeap Framework [110]. It uses some essential components to create a pipeline for automating the evaluation of gesture recognizers. The QuantumLeap Framework is designed to facilitate the development of gestural user interfaces using the Leap Motion controller. It simplifies the development process by providing a software architecture with a pipeline for acquiring, segmenting, recognizing, and mapping gestures to application functions.

6.1 Evaluation Procedure

The comparative testing method use cases in this work are classified as an analytic method of evaluation [137]. It indicates that the user interface (UI) and user are representational during the evaluation. Inspired by the gesture recognition testing procedure described in Section 3.4, we have already described a concrete application of the gesture recognition procedure on "unipath dynamic" gestures in Subsection 4.3.2. Two scenarios are defined: the user-dependent scenario and the user-independent scenario. In the user-dependent scenario, templates are randomly selected from each participant for testing, and a training set is created by randomly choosing templates from the same user. In the user-independent scenario, templates are randomly selected from a participant for testing, and a training set is created by randomly choosing templates from different users. The evaluation result is determined by calculating the execution time and the recognition rate of the gesture recognizers.

6.1.1 User-dependent Scenario

In this evaluation, a recognizer is trained and tested using gestures performed by the same user from the same dataset. The user-dependent evaluation introduces two important parameters: T, which represents the number of training samples used, and R, which represents the number of tests performed for each participant. The procedure randomly selects one sample per gesture class for testing and then selects T samples per gesture class from the remaining samples for testing.

6.1.2 User-independent Scenario

In this evaluation, a recognizer is trained using gestures from one dataset performed by one or more users and then tested with gestures performed by a different user from the same dataset. This evaluation introduces a parameter, denoted as P, which represents the number of independent participants used to train the recognizer. The procedure randomly selects one participant and then randomly selects one sample from each gesture class for testing. It then randomly selects a group of P independent participants to train the recognizer. For each participant in the training set, T samples per gesture class are randomly selected for training from the same dataset.

These scenarios are seen in this work and applied in the case of "intra-device " configuration. However, QuantumLeap's ability to handle multi-device configurations expands the number of possible scenarios. This includes using two datasets from two devices from the same input device family or

leveraging QuantumLeap's extension capabilities to other input device categories. In addition to the mentioned parameters, we can introduce new parameters relevant to different contexts of use of gesture recognition. For example, as discussed in Chapter 5, we can consider the articulation parameter A or the number of points during gesture resampling in the pre-processing step N.

The first main measure in these procedures is the accuracy of a gesture recognizer, which is represented by the average recognition rate of each recognizer, whether it is global (*i.e.*, grouping all conditions) or specific to a particular condition. More detailed results are provided, such as the number of recognized gestures for each trial and the confusion matrix. The second measure is the efficiency of the recognizers, which is determined by the average execution time of each recognizer, whether it is global (*i.e.*, grouping all conditions) or specific to a particular condition. The pre-processing time refers to the time it takes to preprocess the candidate before comparing it with the templates. The classification time refers to the time it takes to compare the candidate with all the templates. The execution time is the total time it takes for preprocessing and classification. All these times are measured in milliseconds (ms).

6.2 Testing Module Implementation

To perform comparative testing on 3D gesture recognizers, a testing module has been implemented in JavaScript within a Node.js environment `https://github.com/Mehous/TestingFramework_QL_UI`. Node.js is a cross-platform environment that fulfills the testing requirements. The tool is a single-page application designed to launch testing and provide evaluators with several options for different parameters, gesture sets, and recognizers. Installing the framework requires basic knowledge of command-line interfaces and Node.js, while configuration involves modifying the configuration file. The application requires storing a set of recognizers, datasets, and a configuration file locally. The recognizers should be in JavaScript, while the datasets should be in JSON format. The results are saved as JSON files, with a defined structure for each evaluation scenario. A large amount of data is returned in the results. They can be global, such as the average recognition rate for a participant, the average execution time for a particular condition of P, or a confusion matrix. They can also be condition-specific, such as the classification time per class or the number of recognized gestures from a set of gestures for a specific condition of parameters N, P, and T.

6.2.1 Configuration

The tool comes with a configuration UI shown in Figure 6.1 that allows evaluators to easily set the initial evaluation conditions according to their requirements. Before selecting the evaluation condition values on the configuration page, the configuration file needs to be customized. It includes a list of available recognizers, datasets for each recognizer, and various parameters with their respective values.

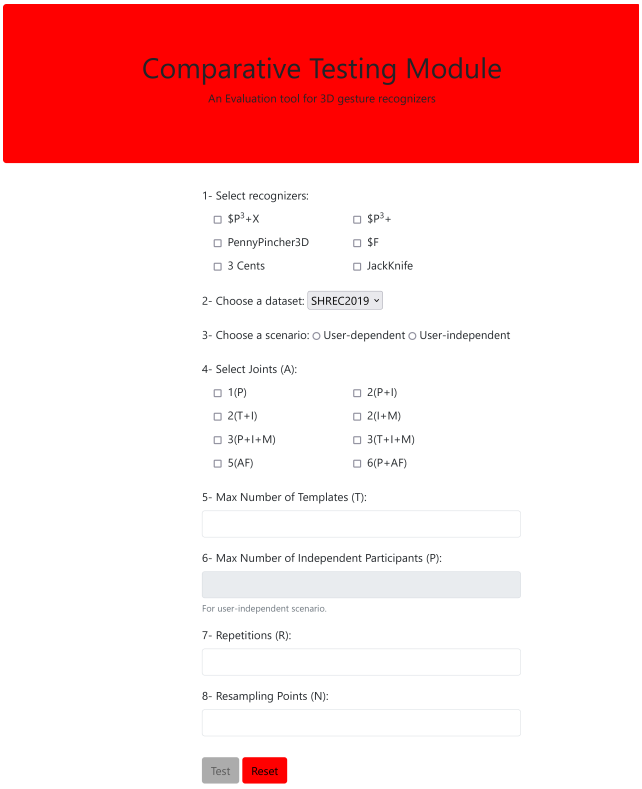


FIGURE 6.1: The UI configuration page.

6.2.2 Recognizers

Recognizers are the fundamental components of the test module. They are responsible for assigning a gesture class to a candidate gesture. Like in QuantumLeap framework [110], recognizers implement three methods:

- addGesture(name, sample, [parameters]): it adds a gesture template to the training set of the recognizer.
- convert(sample,[parameters]): This function converts the loaded sample in memory into an array of sets of points.
- recognize(sample,[parameters]): designate the name of the gesture class that corresponds to the candidate sample provided as an argument.

These methods are primarily used in conjunction with segmenters (seen in Subsection 2.1.4), which detect dynamic gestures from a continuous stream of data.

6.2.3 Testing

The testing is the central element of the module, which contains the different evaluation functions. It calculates the different results and writes them to JSON files located in predefined folders created based on the values of the different variables. the test item allows the evaluator to adjust the values of the parameters under evaluation before initiating the procedure Moreover, the test records logs of the evaluated conditions and saves the gesture templates used for training and testing, allowing other recognizers to be evaluated on the same training and test sets.

6.2.4 Datasets

The testing module must have at least one dataset to evaluate gesture recognizers. This gesture set is stored as files in folders that must follow a standardized structure. This structure is necessary to enable the loader methods to load data from memory into another suitable structure for use in a specific evaluation scenario. The structure is such that each folder contains subfolders representing the participants who performed the gestures stored in JSON format.

A gesture file stored in a dataset must adhere to a certain structure, to ensure optimum use of the data it contains. The gestures used in the testing module share a number of attributes, such as "$name$", which designates the name of the gesture class of the sample. The "$subject$" field, indicates the name or identifier of the user who performed the gesture. the "$date$" the gesture was created by the user. The gesture file contains a set of paths. The attribute "$paths$" includes a set of paths that have names similar to the articulations provided by the "multipath dynamic" hand gestures of the Leap Motion controller. These paths are named after the associated articulations and contain a list of strokes. A stroke is a set of coordinates of points ordered chronologically ($x, y, z, alpha, beta, gamma$), along with another attribute "$timestamp$" and the

"$stroke_id$". Finally, the sample defines an object named "$device$" with various properties related to the device's information. These properties include "$osBrowserInfo$","$resolutionHeight$","$resolutionWidth$" "$rwindowHeight$", "$rwindowWidth$" ,"$rpixelRatio$", "$mouse$", "pen", "$finger$", "$acceleration$" and "$webcam$".

6.2.5 Results

Once the tests are completed, the module generates multiple result files in JSON format. These files are stored in folders and subfolders, following a hierarchical structure of $scenario \rightarrow dataset \rightarrow recognizer$. In each recognizer folder, we decided to store the result files by aggregating the results according to the conditions under which the tests were performed, choosing a predefined order of importance for the parameters as shown in Figure 6.2. We cite the measures that are calculated for aggregated data according to different variables.

- Recognition Accuracy
- Execution Time
- pre-processing Time
- Classification Time
- Confusion Matrix
- Execution Time per Class
- pre-processing Time per Class
- Classification Time per Class
- Raw Recognition Rate
- Raw Execution Time
- Raw pre-processing Time
- Raw Classification Time



FIGURE 6.2: The structure of a result file in user-dependent evaluation.

6.3 Conclusion

The testing module is a support tool for the comparative testing of 3D gesture recognizers. It is designed to help researchers and practitioners. The module is implemented in JavaScript. This tool supports two evaluation scenarios: user-independent and user-dependent. It requires the recognizers to be implemented in JavaScript and the datasets to be imported in JSON format. The testing module generates detailed test results that are written in multiple JSON files. These files are stored in automatically created folders that correspond to different testing conditions. The results contain information such as the overall recognition rate and the average pre-processing time for a set of gesture candidates.

Chapter 7

Conclusion

Developers and practitioners face numerous challenges when developing interactive solutions incorporating hand gesture recognition. This is especially true with the rise and widespread use of mobile devices equipped with various sensors, opening up countless interaction possibilities. The design, testing, and validation of interactive systems are becoming increasingly complex, mainly due to the growing popularity of complex machine learning models in gesture recognition.

This work focuses on the definition and application of a comparative testing systematic procedure for template-based 3D hand gesture recognizers in multiple contexts of use, that permits developers and practitioners to choose a recognizer effectively for developing a gesture interface using rapid prototyping during the design stage. This systematic procedure allows for the comparison of gesture recognizers in multiple contexts of use, which is precisely tailored to 3D hand gesture recognizers adapted for the multi-device.

- First, we introduced the existing state-of-the-art 3D template-based gesture recognizers adapted for rapid prototyping. We also described and compared the most impactful recognizers based on various dimensions defined for rapid prototyping hand gesture recognizers derived from observations, such as gesture natures, data types, device types, classification approaches, gesture types, and invariance properties. At the end of the chapter, we concluded that there is a need for an extended exploration of classification and device dimensions.
- Second, we have introduced the terminology and defined the fundamental concepts for conducting gesture recognition comparative testing in multi-device configuration. This includes the design of a model determining the main stages of gesture recognition testing, identifying the main elements of the comparative testing method, and featuring the systematic nature of the testing procedure.

- We have implemented the systematic procedure of the comparative testing method to evaluate the accuracy and efficiency of 7 state-of-the-art recognizers on"unipath dynamic" gestures. To achieve this, four 2D stroke gesture recognizers ($\$P$, $\$P+$, $\$Q$, and Rubine) from the literature review were extended to 3D: $\$P+^3$, $\$F$, FreeHandUni, $\$Q^3$, $\$P^3$, Rubine3D, and Rubine-Sheng. Significant differences were observed among these recognizers regarding recognition rates and execution times on three datasets (*i.e.*, SHREC2019, 3DTCGS and MadLabSD). $\$P+^3$ consistently achieved the highest recognition rate and excellent execution time in most scenarios, making it the preferred choice for efficiently recognizing 3D trajectories. After $\$P+^3$, we observe that the subsequent recognizers, including $\$F$, FH, $\$Q^3$, $\$P^3$, have similar recognition rates except for $R3D$ and RS which have low recognition rates. There is a significant difference when transitioning from the user-dependent to the user-independent scenario. The number of templates has a significant effect on the recognition rates of all the recognizers, and the number of points has a significant effect on the $-like recognizers. Similarly, the datasets have a significant impact on the recognition rates of all the recognizers in the user-independent scenario.

- We have implemented the systematic procedure for the comparative testing method to evaluate the accuracy and efficiency of 6 recognizers on"multipath dynamic" gestures. This study compared six 3D template-based gesture recognizers from the literature. Our focus was on their recognition rate and execution time. We conducted experiments on two LMC-based datasets: SHREC2019 and Jackknife-LM. We defined independent variables with a limited selection of eight joint combinations. The SHREC2019 results indicated that $\$P^3+$ achieved the highest performance, followed by $\$P^3 + X$ and Jackknife. The $-like recognizers also demonstrated good performance for the same classes of gestures. In addition, all of the recognizers exhibited low response times, making them suitable for template matching. We used the de Borda ranking to select the best conditions among the gesture recognizers. Among the recognizers tested on the Jackknife-LM dataset, the Jackknife recognizer was the best, while PennyPincher3D and 3 Cent should be avoided.

- Finally, we have implemented and described the testing module, which is a tool that facilitates the integration of the systematic procedure for comparative testing of 3D hand gesture recognizers in multiple contexts of use. This offline tool extends the QuantumLeap Framework to evaluate 3D gesture recognizers using different datasets. The testing procedures assess the accuracy and efficiency of the recognizers in two evaluation

scenarios: user-dependent and user-independent, and in the intra-device configuration.

7.1 Advantages and Limitations

Several advantages are to be considered in this work:

- Introduction of terminology and extended definitions that consider the multi-device in the gesture recognition procedure.
- The work allows developers and practitioners to view the problem of gestural interface design from a perspective where the systematic procedure for the comparative testing can precisely meet their needs. This systematic procedure reduces time costs and workload, helping with rapid prototyping and easing the burden of the development process by designating the suitable recognizer in multiple contexts of use.
- The systematic procedure's ability to cover many possibilities enables it to evolve into more complex comparative testing methods, adding new recognizers, datasets, scenarios, and parameters. However, it is essential to note that we have only tested a limited number of instantiations for template-based 3D gesture recognizers.
- The testing module, which extends the QuantumLeap framework, is primarily dedicated to the Leap Motion controller. However, it can also integrate other data sets and devices for comparative testing in single and multi-device configurations. Its proven ability to support this comparative procedure is a key feature to ensure reproducibility in experiments. Moreover, the systematic nature of the comparative testing ensures consistency across different experimental conditions for different dimensions, by altering one component of the comparative testing while keeping all others constant. This method is frequently used in competitions and contests for evaluating the performances of new machine-learning models or algorithms as the SHREC contest series is organized every year [24, 22, 36].

However, there are some limitations and shortcomings in this work:

- The exploration is limited only to template-based recognizers, disregarding complex recognizers based on more complex machine learning techniques [27, 141].
- The conducted experiments within the two use cases do not include the entire values of the different dimensions of the design space, such as

"Simple static" and "Complex static", as well as the "Between-vectors" classification approach implemented in many gesture recognizers such as μ_V [74]. Given the abundance of datasets that are often flawed and take time to process, I'm building upon the work presented in Chapter 5. My focus is on processing seven out of the twelve available Leap-motion-based datasets, and evaluating the results of hundreds of thousands of tests.

- The application of comparative testing in this work is part of the analytic methods evaluation class that relies on theoretical representations of the user and computer. However, this method is limited because it is not a user-based evaluation because there is no real end-user, which is consistent with the desire to use the systematic procedure for rapid prototyping [71]. This classification results from the choice of conducting only the training and testing stages of the gesture recognition procedure.
- The experiments carried out involved only the "intra-device" configuration. Furthermore, the data used in the experiments are all of the same type, specifically Segmented3D, as outlined in the design space of the gesture recognizer defined in Chapter 2. While the datasets primarily provide gestures as sets of 3D positions, devices can provide a variety of data types, such as rotations and accelerations. However, due to a lack of sufficient publicly available datasets, these 3D hand gesture recognizers were not tested.

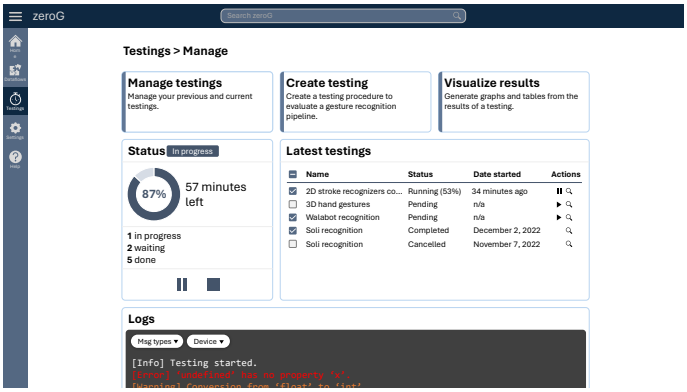
7.2 Future Work

Potential avenues for future research could be explored across a variety of dimensions.

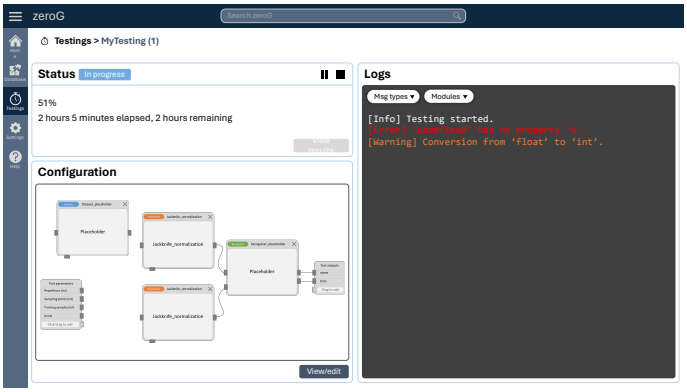
- Along the support tool: The development of the testing module, a tool that extended the QuantumLeap framework [110], encounters the limitations of QuantumLeap. Its primary limitation is that it is confined to the Leap Motion controller or a specific device family and does not support other multi-device gesture recognition configurations. To address its issues, we are currently working on a new framework named *zeroG*, which is a tool that helps users with different levels of expertise with the development of gesture-based applications (Figure 7.1). The framework also includes a testing tool for users who need to perform comparative testing or benchmarking to explore various configurations of gesture recognizers. The aim is to help them make informed decisions that best suit their needs. It allows the evaluation of one or more recognizers on different

datasets and using different parameters in automated sessions, with control over the testing procedure execution(Figure 7.1(a). The tool works by creating data dataflows for gesture recognition, which involves linking different standardized modules that can be customized, as shown in Figure 7.1(b). The testing configuration GUI will provide numerous opportunities to integrate various devices in multi-device configurations. It should allow gesture recording, acquisition of entire datasets, and their management. The GUI should also provide the ability to create new datasets by assembling gestures from different existing datasets stored in the framework's database. Furthermore, it should allow users to publish and share their own datasets and results, which could be used to provide new datasets and centralize them to create competition on gesture recognition. The results of the testing will be available for download or will be viewable using the results visualization tool (refer to Figure 7.1(c)). This tool will allow the generation of common charts, such as a confusion matrix, and will offer users the ability to manipulate these results. Users will be able to aggregate data based on a specific variable or filter results to show a particular testing configuration. This will provide the opportunity for preliminary analysis of various measures, such as recognition rate and execution time.

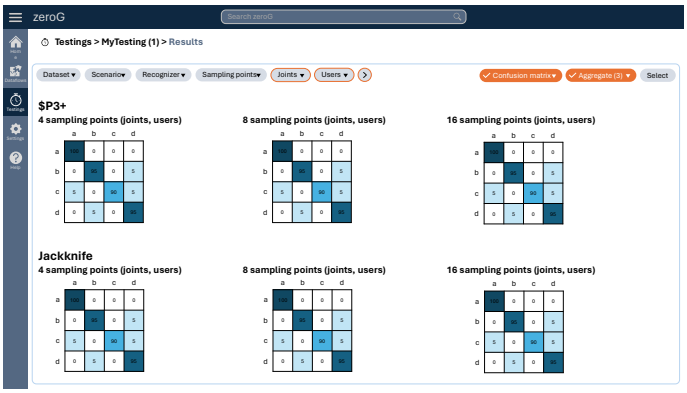
- Along the scope of the experiments: The shortcomings mentioned above suggest extending the systematic procedure to include more datasets and exploring new gestures that are relevant to specific application fields or performed with different parts of the human body, especially those used for gesture elicitation studies [119]. Additionally, more recognizers (such as DeepGRU [72]) and different conditions, such as more or fewer classes or templates per gesture class, could be considered. This would allow for a direct comparison of other recognizers to the best competitor emerging from the previously completed systematic procedure on a common dataset. Like in a competition. Generally, the practice of varying one aspect of comparative testing while keeping the rest constant is often seen in competition [24, 22, 36, 99]. There are many possible configurations at the first level. After that, parameters can also be varied, opening up countless possibilities:
  - **Vary**: Recognizer. **Fixed**: Dataset; Method; Scenario; Parameters; Technique
  - **Vary**: Dataset, **Fixed**: Recognizer; Method; Scenario; Parameters; Technique
  - **Vary**: Method, **Fixed**: Recognizer; Dataset; Scenario; Parameters; Technique



(a) zeroG Testing dashboard.



(b) zeroG Testing session and configuration.



(c) zeroG Testing results.

FIGURE 7.1: ZeroG testing tool mockups UI.

  - **Vary**: Scenario, **Fixed**: Recognizer; Dataset; Method; Parameters; Technique
  - **Vary**: Parameters, **Fixed**: Recognizer; Dataset; Method; Scenario; Technique
  - **Vary**: Technique, Fixed: Recognizer; Dataset; Parmeters; Method; Scenario;

- Along the hypothesis to be tested: The method defined in this work can be applied to other contexts of use, including different multi-device configurations. The hypothesis put forward in this respect is supportable and deserves to be explored further.

- Along the design guidance: The systematic procedure could help to define a process for prototyping a gesture-based user interface using the comparative testing method. Once a dataset is decided, the systematic procedure for comparative testing could automatically determine which recognizer is the most suitable depending on the conditions imposed. We envision an automatic generation of a decision tree that guides the practitioner on which recognizer to select for one or several datasets by asking questions at each decision tree node, such as: Do you want to maximize the recognition rate? The execution speed? Under which circumstances? The framework could be designed to integrate a complete testing system and allow for the direct integration of the recognizer resulting from the systematic procedure for comparative testing into a real-time application.

Bibliography

[1] Nadeem Abbas, Jesper Andersson, and Danny Weyns. "ASPLe: A methodology to develop self-adaptive software systems with systematic reuse". In: *Journal of Systems and Software* 167 (2020), p. 110626. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2020.110626. URL: https://www.sciencedirect.com/science/article/pii/S0164121220301047.

[2] Amaaz Ahmad. *What is Comparison Testing*. 2022. URL: https://www.educative.io/answers/what-is-comparison-testing.

[3] Roland Aigner et al. *Understanding Mid-Air Hand Gestures: A Study of Human Preferences in Usage of Gesture Types for HCI*. Tech. rep. MSR-TR-2012-111. Microsoft Research, Nov. 2012. URL: https://www.microsoft.com/en-us/research/publication/understanding-mid-air-hand-gestures-a-study-of-human-preferences-in-usage-of-gesture-types-for-hci/.

[4] Ahmad Akl and Shahrokh Valaee. "Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing". en. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. Dallas, TX, USA: IEEE, 2010, 2270–2273. ISBN: 978-1-4244-4295-9. DOI: 10.1109/ICASSP.2010.5495895. URL: http://ieeexplore.ieee.org/document/5495895/.

[5] Ahmad Sami Al-Shamayleh et al. "A systematic literature review on vision based gesture recognition techniques". en. In: *Multimedia Tools and Applications* 77.21 (Nov. 2018), pp. 28121–28184. ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-018-5971-z.

[6] Viknesh Andiappan and Yoke Kin Wan. "Distinguishing approach, methodology, method, procedure and technique in process systems engineering". English. In: *Clean Technologies and Environmental Policy* 22.3 (Apr. 2020), pp. 547–555. ISSN: 1618-954X. DOI: 10.1007/s10098-020-01819-w.

[7] Lisa Anthony and Jacob O. Wobbrock. "A Lightweight Multistroke Recognizer for User Interface Prototypes". In: *Proceedings of Graphics Interface 2010*. GI '10. Ottawa, Ontario, Canada: Canadian Information

Processing Society, 2010, pp. 245–252. ISBN: 978-1-56881-712-5. URL: http://dl.acm.org/citation.cfm?id=1839214.1839258.

[8] Lisa Anthony and Jacob O. Wobbrock. "$N-protractor: A Fast and Accurate Multistroke Recognizer". In: *Proceedings of Graphics Interface 2012*. GI '12. Toronto, Ontario, Canada: Canadian Information Processing Society, 2012, pp. 117–120. ISBN: 978-1-4503-1420-6. URL: http://dl.acm.org/citation.cfm?id=2305276.2305296.

[9] Nathalie Aquino et al. "Usability evaluation of multi-device/platform user interfaces generated by model-driven engineering". In: *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '10. Bolzano-Bozen, Italy: Association for Computing Machinery, 2010. ISBN: 9781450300391. DOI: 10.1145/1852786.1852826. URL: https://doi.org/10.1145/1852786.1852826.

[10] Sluÿters Arthur et al. "Consistent, Continuous, and Customizable Mid-Air Gesture Interaction for Browsing Multimedia Objects on Large Displays". In: *International Journal of Human–Computer Interaction* 39.12 (2023), pp. 2492–2523. DOI: 10.1080/10447318.2022.2078464. eprint: https://doi.org/10.1080/10447318.2022.2078464. URL: https://doi.org/10.1080/10447318.2022.2078464.

[11] Maryam Asadi-Aghbolaghi et al. "Deep Learning for Action and Gesture Recognition in Image Sequences: A Survey". In: *Gesture Recognition*. Ed. by Sergio Escalera, Isabelle Guyon, and Vassilis Athitsos. The Springer Series on Challenges in Machine Learning. Cham: Springer International Publishing, 2017, pp. 539–578. ISBN: 978-3-319-57020-4. DOI: 10.1007/978-3-319-57021-1_19. URL: http://link.springer.com/10.1007/978-3-319-57021-1_19.

[12] Daniel Bachmann, Frank Weichert, and Gerhard Rinkenauer. "Review of three-dimensional human-computer interaction with focus on the leap motion controller". In: *Sensors* 18.7 (2018), p. 2194. ISSN: 1424-8220. DOI: 10.3390/s18072194. URL: https://www.mdpi.com/1424-8220/18/7/2194.

[13] Michel Beaudouin-Lafon. "Designing interaction, not interfaces". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '04. Gallipoli, Italy: Association for Computing Machinery, 2004, 15–22. ISBN: 1581138679. DOI: 10.1145/989863.989865. URL: https://doi.org/10.1145/989863.989865.

[14] Michel Beaudouin-Lafon, Susanne Bødker, and Wendy E. Mackay. "Generative Theories of Interaction". In: *ACM Trans. Comput.-Hum. Interact.* 28.6 (Nov. 2021). ISSN: 1073-0516. DOI: 10.1145/3468505. URL: https://doi.org/10.1145/3468505.

[15] Abdo Beirekdar et al. "Flexible Reporting for Automated Usability and Accessibility Evaluation of Web Sites". In: *Human-Computer Interaction - INTERACT 2005*. Ed. by Maria Francesca Costabile and Fabio Paternò. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 281–294. ISBN: 978-3-540-31722-7.

[16] Abel Díaz Berenguer et al. "GestureVLAD: Combining Unsupervised Features Representation and Spatio-Temporal Aggregation for Doppler-Radar Gesture Recognition". In: *IEEE Access* 7 (2019), pp. 137122–137135. DOI: 10.1109/ACCESS.2019.2942305.

[17] Doug A. Bowman et al. "An Introduction to 3-D User Interface Design". In: *Presence: Teleoperators and Virtual Environments* 10.1 (Feb. 2001), pp. 96–108. DOI: 10.1162/105474601750182342. URL: https://direct.mit.edu/pvar/article-pdf/10/1/96/1623535/105474601750182342.pdf.

[18] Roberto Brunelli. *Template Matching Techniques in Computer Vision: Theory and Practice*. New York: John Wiley & Sons, Ltd, 2009. ISBN: 9780470744055. DOI: 10.1002/9780470744055. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470744055.

[19] John Burnham. "Approach - Method- - Technique: Making Distinctions and Creating Connections". In: *Human systems : the journal of systemic consultation and management* 3 (1999), pp. 3–26.

[20] William Buxton. "Lexical and Pragmatic Considerations of Input Structures". In: *SIGGRAPH Computer Graphics* 17.1 (Jan. 1983), pp. 31–37. ISSN: 0097-8930. DOI: 10.1145/988584.988586. URL: http://doi.acm.org/10.1145/988584.988586.

[21] Gaëlle Calvary et al. "A Unifying Reference Framework for multi-target user interfaces". In: *Interacting with Computers* 15.3 (2003). Computer-Aided Design of User Interface, pp. 289–308. ISSN: 0953-5438. DOI: 10.1016/S0953-5438(03)00010-9. URL: https://doi.org/10.1016/S0953-5438(03)00010-9.

[22] Ariel Caputo et al. "SHREC 2021: Skeleton-based hand gesture recognition in the wild". In: *Computers & Graphics* 99 (2021), pp. 201–211. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2021.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0097849321001382.

[23] Fabio M. Caputo et al. "A 3 Cent Recognizer: Simple and Effective Retrieval and Classification of Mid-Air Gestures from Single 3D Traces". In: *Proceedings of the Conference on Smart Tools and Applications in Computer Graphics*. STAG '17. Catania, Italy: Eurographics Association, 2017, pp. 9–15. DOI: 10.2312/stag.20171221. URL: https://doi.org/10.2312/stag.20171221.

[24] Fabio M. Caputo et al. "SHREC 2019 Track: Online Gesture Recognition". In: *Eurographics Workshop on 3D Object Retrieval*. Ed. by Silvia Biasotti, Guillaume Lavoué, and Remco Veltkamp. Genova, Italy: The Eurographics Association, May 2019. ISBN: 978-3-03868-077-2. DOI: 10.2312/3dor.20191067.

[25] Salman Cheema, Michael Hoffman, and Joseph J. LaViola. "3D Gesture classification with linear acceleration and angular velocity sensing devices for video games". In: *Entertainment Computing* 4.1 (Feb. 2013), pp. 11–24. ISSN: 18759521. DOI: 10.1016/j.entcom.2012.09.002. URL: https://www.sciencedirect.com/science/article/pii/S1875952112000146.

[26] Yineng Chen et al. "Pactolus: A Method for Mid-Air Gesture Segmentation within EMG". In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. San Jose, CA, USA: ACM, May 2016, pp. 1760–1765. ISBN: 978-1-4503-4082-3. DOI: 10.1145/2851581.2892492. URL: https://dl.acm.org/doi/10.1145/2851581.2892492.

[27] Hong Cheng, Lu Yang, and Zicheng Liu. "Survey on 3D Hand Gesture Recognition". In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.9 (Sept. 2016), pp. 1659–1673. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2015.2469551.

[28] David.N. Chin. "Empirical Evaluation of User Models and User-Adapted Systems". In: *User Modeling and User-Adapted Interaction* 11 (2001), 181–194. DOI: 10.1023/A:1011127315884.

[29] Stefano Chioccarello et al. "FORTE: Few Samples for Recognizing Hand Gestures with a Smartphone-attached Radar". In: *Proc. ACM Hum.-Comput. Interact.* 7.EICS (June 2023). DOI: 10.1145/3593231. URL: https://doi.org/10.1145/3593231.

[30] Eunjung Choi, Heejin Kim, and Min K. Chung. "A taxonomy and notation method for three-dimensional hand gestures". In: *International Journal of Industrial Ergonomics* 44.1 (Jan. 2014), pp. 171–188. ISSN: 01698141. DOI: 10.1016/j.ergon.2013.10.011. URL: https://www.sciencedirect.com/science/article/pii/S0169814113001285.

[31] Harisson Cook et al. "Enabling Gesture Interaction with 3D Point Cloud". In: *Proceedings of the 24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. Vol. 2602. Plzen, Czech Republic: Computer Science Research Notes CSRN, June 2016, pp. 59–68. ISBN: 978-80-86943-58-9. URL: https://opus.lib.uts.edu.au/handle/10453/45407.

[32] Adrien Coyette et al. "Trainable Sketch Recognizer for Graphical User Interface Design". In: *Human-Computer Interaction – INTERACT 2007*. Ed. by Cécilia Baranauskas et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 124–135. ISBN: 978-3-540-74796-3.

[33] Elena-Gina Craciun, Ionela Rusu, and Radu-Daniel Vatavu. *Free-Hand Gesture Recognizer Pseudocode*. Accessed: 2020-04-23. 2016. URL: http://www.eed.usv.ro/mintviz/projects/GIVISIMP/data/Pseudocode2.pdf.

[34] Andries van Dam. "Post-WIMP User Interfaces". In: *Commun. ACM* 40.2 (1997), pp. 63–67. DOI: 10.1145/253671.253708. URL: https://doi.org/10.1145/253671.253708.

[35] Alan Davis. *Getting Started with the Leap Motion SDK*. Feb. 2014. URL: https://blog.leapmotion.com/getting-started-leap-motion-sdk/.

[36] Quentin De Smedt et al. "SHREC'17 Track: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset". In: *3DOR - 10th Eurographics Workshop on 3D Object Retrieval*. Ed. by I. Pratikakis, F. Dupont, and M. Ovsjanikov. Lyon, France, Apr. 2017, pp. 1–6. ISBN: 978-3-03868-030-7. DOI: 10.2312/3dor.20171049. URL: https://hal.archives-ouvertes.fr/hal-01563505.

[37] Charles-Eric Dessart, Vivian Genaro Motti, and Jean Vanderdonckt. "Showing user interface adaptivity by animated transitions". In: *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '11. Pisa, Italy: Association for Computing Machinery, 2011, 95–104. ISBN: 9781450306706. DOI: 10.1145/1996461.1996501. URL: https://doi.org/10.1145/1996461.1996501.

[38] Natalia Díaz Rodríguez et al. "Understanding Movement and Interaction: An Ontology for Kinect-Based 3D Depth Sensors". In: *Ubiquitous Computing and Ambient Intelligence. Context-Awareness and Context-Driven Interaction*. Ed. by Gabriel Urzaiz et al. Vol. 8276. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2013, pp. 254–261. ISBN: 978-3-319-03176-7. DOI: 10.1007/978-3-319-03176-7_33.

[39] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. USA: Wiley-Interscience, 2000. ISBN: 0471056693.

[40] David Efron. *Gesture, Race and Culture: A Tentative Study of the Spatio-temporal and Linguistic Aspects of the Gestural Behavior of Eastern Jews and Southern Italians in New York City, Living Under Similar as Well as Different Environmental Conditions*. Approaches to Semiotics [AS]. Mouton De Gruyter, 1972. ISBN: 978-3-11-098103-2.

[41] Ana Belén Erazo and Jorge Luis Pérez Medina. "Algorithmic Efficiency of Stroke Gesture Recognizers: a Comparative Analysis". In: *International Journal on Advanced Science, Engineering and Information Technology* 10.2 (Mar. 2020), pp. 438–446. DOI: 10.18517/ijaseit.10.2.10807. URL: https://ijaseit.insightsociety.org/index.php/ijaseit/article/view/10807.

[42] Gonzalo Ferrer and Alberto Sanfeliu. "Comparative Analysis of Human Motion Trajectory Prediction Using Minimum Variance Curvature". In: *Proceedings of the 6th International Conference on Human-Robot Interaction*. HRI '11. Lausanne, Switzerland: Association for Computing Machinery, 2011, 135–136. ISBN: 9781450305617. DOI: 10.1145/1957656.1957698.

[43] Yoav Freund and Robert E Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.1997.1504. URL: https://www.sciencedirect.com/science/article/pii/S002200009791504X.

[44] Dafydd Gibbon. "Gesture Theory is Linguistics: On Modelling Multimodality as Prosody". In: *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*. Ed. by Olivia Kwong. Hong Kong: City University of Hong Kong, Dec. 2009, pp. 9–18. URL: https://aclanthology.org/Y09-1002.

[45] James Herold and Thomas F Stahovich. "The One Cent Recognizer: A Fast, Accurate, and Easy-to-implement Handwritten Gesture Recognition Technique". In: *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*. SBIM '12. Annecy, France: Eurographics Association, 2012, pp. 39–46. ISBN: 978-3-905674-42-2. DOI: 10.2312/SBM/SBM12/039-046. URL: http://dx.doi.org/10.2312/SBM/SBM12/039-046.

[46] Michael Hoffman, Paul Varcholik, and Joseph J. LaViola. "Breaking the status quo: Improving 3D gesture recognition with spatially convenient input devices". In: *2010 IEEE Virtual Reality Conference (VR)*. Boston, MA, USA: IEEE, Mar. 2010, pp. 59–66. ISBN: 978-1-4244-6237-7. DOI: 10.1109/VR.2010.5444813. URL: http://ieeexplore.ieee.org/document/5444813/.

[47] Donald B. Hofler. "Approach, method, technique: a clarification". In: *Reading World* 23.1 (1983), pp. 71–72. DOI: 10.1080/19388078309557742. URL: https://doi.org/10.1080/19388078309557742.

[48] Berthold K. P. Horn, Hugh M. Hilden, and Shahriar Negahdaripour. "Closed-form solution of absolute orientation using orthonormal matrices". In: *Journal of the Optical Society of America A* 5.7 (July 1988), pp. 1127–1135. ISSN: 1084-7529, 1520-8532. DOI: 10.1364/JOSAA.5.001127. URL: https://opg.optica.org/josaa/abstract.cfm?URI=josaa-5-7-1127.

[49] Jinmiao Huang, Prakhar Jaiswal, and Rahul Rai. "Gesture-based system for next generation natural and intuitive interfaces". In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 33.1 (2019), pp. 54–68. DOI: 10.1017/S0890060418000045.

[50] Hairina Mohd Jais, Zainal Rasyid Mahayuddin, and Haslina Arshad. "A review on gesture recognition using kinect". In: *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*. Denpasar, Bali, Indonesia: IEEE, Aug. 2015, 594–599. ISBN: 978-1-4673-6778-3. DOI: 10.1109/ICEEI.2015.7352569. URL: http://ieeexplore.ieee.org/document/7352569/.

[51] Maria Karam and m. c. schraefel. *A Taxonomy of Gestures in Human Computer Interactions*. Tech. rep. Jan. 2005. URL: https://eprints.soton.ac.uk/261149/.

[52] Adam Kendon. "How gestures can become like words." In: *Cross-cultural perspectives in nonverbal communication*. Ashland, OH, US: Hogrefe & Huber Publishers, 1988, pp. 131–141. ISBN: 3-8017-0278-2.

[53] Ahmed S. Khalaf et al. "A Comparative Study of Hand Gesture Recognition Devices in the Context of Game Design". In: *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*. ISS '19. Daejeon, Republic of Korea: Association for Computing Machinery, 2019, 397–402. ISBN: 9781450368919. DOI: 10.1145/3343055.3360758.

[54] Rafiqul Zaman Khan. "Comparative Study of Hand Gesture Recognition System". In: *Computer Science & Information Technology*. Vol. 2. July 2012, pp. 203–213. ISBN: 9781921987045. DOI: 10.5121/csit.2012.2320.

[55] Sotiris Kotsiantis, I. Zaharakis, and P. Pintelas. "Machine learning: A review of classification and combining techniques". In: *Artificial Intelligence Review* 26 (Nov. 2006), pp. 159–190. DOI: 10.1007/s10462-007-9052-3.

[56] Sven Kratz and Maribeth Back. "Towards Accurate Automatic Segmentation of IMU-Tracked Motion Gestures". In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '15*. Seoul, Republic of Korea: ACM Press, 2015, pp. 1337–1342. ISBN: 978-1-4503-3146-3. DOI: 10.1145/2702613.2732922. URL: http://dl.acm.org/citation.cfm?doid=2702613.2732922.

[57] Sven Kratz and Michael Rohs. "Protractor3D: A Closed-form Solution to Rotation-invariant 3D Gestures". In: *Proceedings of the 16th International Conference on Intelligent User Interfaces*. IUI '11. Palo Alto, CA, USA: ACM, 2011, pp. 371–374. ISBN: 978-1-4503-0419-1. DOI: 10.1145/1943403.1943468. URL: http://doi.acm.org/10.1145/1943403.1943468.

[58] Sven Kratz and Michael Rohs. "The $3 Recognizer: Simple 3D Gesture Recognition on Mobile Devices". In: *Proceedings of the 15th International Conference on Intelligent User Interfaces*. IUI '10. Hong Kong, China: ACM, 2010, pp. 419–420. ISBN: 978-1-60558-515-4. DOI: 10.1145/1719970.1720051. URL: http://doi.acm.org/10.1145/1719970.1720051.

[59] Per Ola Kristensson and Leif C. Denby. "Continuous Recognition and Visualization of Pen Strokes and Touch-Screen Gestures". In: *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*. SBIM '11. Vancouver, British Columbia, Canada: ACM, 2011, pp. 95–102. ISBN: 9781450309066. DOI: 10.1145/2021164.2021181. URL: https://doi.org/10.1145/2021164.2021181.

[60] Per Ola Kristensson, Thomas Nicholson, and Aaron Quigley. "Continuous recognition of one-handed and two-handed gestures using 3D full-body motion tracking sensors". In: *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces - IUI '12*. Lisbon, Portugal: ACM Press, 2012, p. 89. ISBN: 978-1-4503-1048-2. DOI: 10.1145/2166966.2166983. URL: http://dl.acm.org/citation.cfm?doid=2166966.2166983.

[61] Per-Ola Kristensson and Shumin Zhai. "SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers". In: *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*. UIST '04. Santa Fe, NM, USA: ACM, 2004, pp. 43–52. ISBN: 1-58113-957-8. DOI: 10.1145/1029632.1029640. URL: http://doi.acm.org/10.1145/1029632.1029640.

[62] Lynn Kysh. *Difference between a systematic review and a literature review*. Poster presentation. Aug. 2013. DOI: 10.6084/m9.figshare.766364.v1. URL: https://figshare.com/articles/Difference_between_a_systematic_review_and_a_literature_review/766364.

[63] Christine Kühnel et al. "I'm home: Defining and evaluating a gesture set for smart-home control". In: *International Journal of Human-Computer Studies* 69.11 (Oct. 2011), 693–704. ISSN: 10715819. DOI: 10.1016/j.ijhcs.2011.04.005. URL: https://www.sciencedirect.com/science/article/pii/S1071581911000668.

[64] Houssem Lahiani, Monji Kherallah, and Mahmoud Neji. "Vision Based Hand Gesture Recognition for Mobile Devices: A Review". In: *Proceedings of the 16th International Conference on Hybrid Intelligent Systems (HIS 2016)*. Ed. by Ajith Abraham et al. Vol. 552. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2017, 308–318. ISBN: 978-3-319-52940-0. DOI: 10.1007/978-3-319-52941-7_31. URL: http://link.springer.com/10.1007/978-3-319-52941-7_31.

[65] Joseph J. LaViola. "3D Gestural Interaction: The State of the Field". In: *ISRN Artificial Intelligence* 2013 (2013), pp. 1–18. ISSN: 2090-7443. DOI: 10.1155/2013/514641. URL: https://www.hindawi.com/journals/isrn/2013/514641/.

[66] Luis A. Leiva, Daniel Martín-Albo, and Réjean Plamondon. "Gestures à Go Go: Authoring Synthetic Human-like Stroke Gestures Using the Kinematic Theory of Rapid Movements". In: *ACM Transactions on Intelligent Systems and Technology* 7.2 (Nov. 2015). ISSN: 2157-6904. DOI: 10.1145/2799648. URL: https://doi.org/10.1145/2799648.

[67] Howard Levene. "Robust Tests for Equality of Variances". In: *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*. Ed. by Ingram Olkin and Harold Hotelling et al. Palo Alto, CA, USA: Stanford University Press, 1960, pp. 278–292.

[68] Yang Li. "Protractor: A Fast and Accurate Gesture Recognizer". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, GA, USA: ACM, 2010, pp. 2169–2172. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753654. URL: http://doi.acm.org/10.1145/1753326.1753654.

[69] Jiayang Liu et al. "uWave: Accelerometer-based personalized gesture recognition and its applications". In: *Pervasive and Mobile Computing* 5.6 (Dec. 2009), pp. 657–675. ISSN: 15741192. DOI: 10.1016/j.pmcj.2009.07.007. URL: https://www.sciencedirect.com/science/article/pii/S1574119209000674.

[70] Granit Luzhnica et al. "A sliding window approach to natural hand gesture recognition using a custom data glove". In: *2016 IEEE Symposium on 3D User Interfaces (3DUI)*. Greenville, SC, USA: IEEE, Mar. 2016, pp. 81–90. ISBN: 978-1-5090-0842-1. DOI: 10.1109/3DUI.2016.7460035. URL: http://ieeexplore.ieee.org/document/7460035/.

[71] M. MacLeod. *An Introduction to Usability Evaluation*. Tech. rep. 1992. URL: https://eprintspublications.npl.co.uk/437/1/DITC202.pdf.

[72] Mehran Maghoumi and Joseph J. LaViola. "DeepGRU: Deep Gesture Recognition Utility". In: *Advances in Visual Computing*. Ed. by George Bebis et al. Vol. 11844. ISVC 2019. Cham: Springer, 2019, pp. 16–31. ISBN: 978-3-030-33720-9. DOI: 10.1007/978-3-030-33720-9_2. URL: https://link.springer.com/chapter/10.1007/978-3-030-33720-9_2.

[73] Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. "Two-dimensional Stroke Gesture Recognition: A Survey". In: *ACM Computing Surveys* 54.7 (July 2021). ISSN: 0360-0300. DOI: 10.1145/3465400. URL: https://doi.org/10.1145/3465400.

[74] Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. "μV: An Articulation, Rotation, Scaling, and Translation Invariant (ARST) Multi-Stroke Gesture Recognizer". In: *Proceedings of the ACM on Human-Computer Interaction* 6.EICS (June 2022). DOI: 10.1145/3532200. URL: https://doi.org/10.1145/3532200.

[75] Nathan Magrofuoco and Jean Vanderdonckt. "Gelicit: A Cloud Platform for Distributed Gesture Elicitation Studies". In: *Proceedings of the ACM on Human-Computer Interaction* 3.EICS (June 2019). DOI: 10.1145/3331148. URL: https://doi.org/10.1145/3331148.

[76] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. "Hand Gesture Recognition with Jointly Calibrated Leap Motion and Depth Sensor". In: *Multimedia Tools Appl.* 75.22 (Nov. 2016), 14991–15015. ISSN: 1380-7501. DOI: 10.1007/s11042-015-2451-6.

[77] Mykola Maslych et al. "Effective 2D Stroke-Based Gesture Augmentation for RNNs". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI '23. Hamburg, Germany: ACM, 2023, pp. 1–13. ISBN: 9781450394215. DOI: 10.1145/3544548.3581358. URL: https://doi.org/10.1145/3544548.3581358.

[78] David McNeill. *Hand and mind: what gestures reveal about thought*. University of Chicago Press, 1992. ISBN: 978-0-226-56132-5.

[79] Merriam-Webster. *Use Case*. In: *Merriam-Webster.com dictionary*. URL: https://www.merriam-webster.com/dictionary/use%20case (visited on 12/08/2023).

[80] Sushmita Mitra and Tinku Acharya. "Gesture Recognition: A Survey". In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 37.3 (May 2007), pp. 311–324. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2007.893280. URL: https://ieeexplore.ieee.org/document/4154947.

[81] Javier Molina et al. "A Natural and Synthetic Corpus for Benchmarking of Hand Gesture Recognition Systems". In: *Machine Vision and Applications* 25.4 (May 2014), pp. 943–954. ISSN: 0932-8092. DOI: 10.1007/s00138-013-0576-z. URL: https://doi.org/10.1007/s00138-013-0576-z.

[82] Abdullah Mujahid et al. "Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model". In: *Applied Sciences* 11.9 (2021). ISSN: 2076-3417. DOI: 10.3390/app11094164. URL: https://www.mdpi.com/2076-3417/11/9/4164.

[83] Keith Ngan, David N. Straub, and Peter Bartello. "Three-dimensionalization of freely-decaying two-dimensional turbulence". In: *Physics of Fluids* 16.8 (July 2004), pp. 2918–2932. ISSN: 1070-6631. DOI: 10.1063/1.1763191. URL: https://doi.org/10.1063/1.1763191.

[84] Jakob Nielsen. *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. ISBN: 9780080520292.

[85] Object Management Group (OMG). *Software & Systems Process Engineering MetaModel Specification, Version 2.0*. http://www.omg.org/spec/SPEM/2.0/PDF/. Apr. 2008.

[86] Mehdi Ousmer et al. "A Systematic Procedure for Comparing Template-Based Gesture Recognizers". In: *HCI International 2022 - Late Breaking Papers. Multimodality in Advanced Interaction Environments*. Ed. by Masaaki Kurosu et al. Cham: Springer Nature Switzerland, 2022, pp. 160–179. ISBN: 978-3-031-17618-0. DOI: 10.1007/978-3-031-17618-0_13. URL: https://doi.org/10.1007/978-3-031-17618-0_13.

[87] Mehdi Ousmer et al. "Recognizing 3D Trajectories as 2D Multi-Stroke Gestures". In: *Proceedings of the ACM on Human-Computer Interaction* 4.ISS (Nov. 2020), pp. 1–21. DOI: 10.1145/3427326. URL: https://doi.org/10.1145/3427326.

[88] Vik Parthiban et al. "Gestural-Vocal Coordinated Interaction on Large Displays". In: *Companion of the 2022 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '22 Companion. Sophia Antipolis, France: Association for Computing Machinery, 2022, 26–32. ISBN: 9781450390316. DOI: 10.1145/3531706.3536457. URL: https://doi.org/10.1145/3531706.3536457.

[89] Alain Pirotte et al. "Materialization: A Powerful and Ubiquitous Abstraction Pattern". In: *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, 630–641. ISBN: 1558601538.

[90] Corey R. Pittman and Joseph J. LaViola. "Multiwave: Complex Hand Gesture Recognition Using the Doppler Effect". In: *Proceedings of the 43rd Graphics Interface Conference*. GI 2017. Edmonton, Alberta, Canada: Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 2017, pp. 97–106. ISBN: 978-0-9947868-2-1. DOI: 10.20380/GI2017.13.

[91] George V. Popescu, Helmuth Trefftz, and Grigore Greg Burdea. "Multimodal Interaction Modeling". English (US). In: *Handbook of Virtual Environments, 2nd ed.* CRC Press, Jan. 2014, pp. 411–434. ISBN: 9781466511842.

[92] P.A. Popov and R. Laganière. "Long Hands gesture recognition system: 2 step gesture recognition with machine learning and geometric shape analysis". In: *Multimed Tools Appl* 81 (2022), pp. 40311–40342. DOI: 10.1007/s11042-022-12870-8.

[93] Björn Puype. "Extending the iGesture Framework with Multi-modal Gesture Interaction Functionality". Master Thesis. Vrije Universiteit Brussels, 2010.

[94] Francis Quek et al. "Multimodal Human Discourse: Gesture and Speech". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 9.3 (Sept. 2002), 171–193. ISSN: 1073-0516. DOI: 10.1145/568513.568514. URL: https://doi.org/10.1145/568513.568514.

[95] J. Reaver, Thomas F. Stahovich, and James Herold. "How to Make a Quick$: Using Hierarchical Clustering to Improve the Efficiency of the Dollar Recognizer". In: *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*. SBIM '11. Vancouver, British Columbia, Canada: ACM, 2011, pp. 103–108. ISBN: 978-1-4503-0906-6. DOI: 10.1145/2021164.2021183. URL: http://doi.acm.org/10.1145/2021164.2021183.

[96] B. H. K. Reddy and K. Anitha. "A Comparative Analysis for Measuring Accuracy in Recognizing Hand Gestures using Feedforward Neural Network and CNN Method". In: *2022 International Conference on Business Analytics for Technology and Security (ICBATS)*. IEEE. Dubai, United Arab Emirates, 2022, pp. 1–8. DOI: 10.1109/ICBATS54253.2022.9759008.

[97] Dean Rubine. "Combining Gestures and Direct Manipulation". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '92. Monterey, CA, USA: ACM, 1992, pp. 659–660. ISBN: 0-89791-513-5. DOI: 10.1145/142750.143072. URL: http://doi.acm.org/10.1145/142750.143072.

[98] Dean Rubine. "Specifying Gestures by Example". In: *ACM SIGGRAPH Computer Graphics* 25.4 (July 1991), pp. 329–337. ISSN: 0097-8930. DOI: 10.1145/127719.122753. URL: https://doi.org/10.1145/127719.122753.

[99] Simon Ruffieux, Denis Lalanne, and Elena Mugellini. "ChAirGest: a challenge for multimodal mid-air gesture recognition for close HCI". In: *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*. ICMI '13. New York, NY, USA: Association for Computing Machinery, 2013, 483–488. ISBN: 9781450321297. DOI: 10.1145/2522848.2532590. URL: https://doi.org/10.1145/2522848.2532590.

[100] Jaime Ruiz, Yang Li, and Edward Lank. "User-Defined Motion Gestures for Mobile Interaction". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. Vancouver, BC, Canada: Association for Computing Machinery, 2011, pp. 197–206. ISBN: 9781450302289. DOI: 10.1145/1978942.1978971. URL: https://doi.org/10.1145/1978942.1978971.

[101] Dan Saffer. *Designing Gestural Interfaces: Touchscreens and Interactive Devices*. O'Reilly Media, Inc., 2008. ISBN: 0596518390.

[102] Monika Schak and Alexander Gepperth. "A Study on Catastrophic Forgetting in Deep LSTM Networks". In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Deep Learning*. Ed. by Igor V. Tetko et al. Cham: Springer International Publishing, 2019, pp. 714–728. ISBN: 978-3-030-30484-3.

[103] Thomas Schlömer et al. "Gesture Recognition with a Wii Controller". In: *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction*. TEI '08. Bonn, Germany: ACM, 2008, 11–14. ISBN: 9781605580043. DOI: 10.1145/1347390.1347395. URL: https://doi.org/10.1145/1347390.1347395.

[104] Julia Schwarz et al. "Combining Body Pose, Gaze, and Gesture to Determine Intention to Interact in Vision-Based Interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, 2014, 3443–3452. ISBN: 9781450324731. DOI: 10.1145/2556288.2556989. URL: https://doi.org/10.1145/2556288.2556989.

[105] Tevfik Metin Sezgin, Thomas Stahovich, and Randall Davis. "Sketch Based Interfaces: Early Processing for Sketch Understanding". In: *Proceedings of the 2001 Workshop on Perceptive User Interfaces*. PUI '01. Orlando, FL, USA: ACM, 2001, pp. 1–8. DOI: 10.1145/971478.971487. URL: http://doi.acm.org/10.1145/971478.971487.

[106] Sakshi Sharma and Sukhwinder Singh. "Vision-based hand gesture recognition using deep learning for the interpretation of sign language". In: *Expert Systems with Applications* 182 (2021), p. 115657. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115657. URL: https://www.sciencedirect.com/science/article/pii/S0957417421010484.

[107] Jia Sheng. *A Study of AdaBoost in 3D Gesture Recognition*. Tech. rep. Toronto, Ontario, Canada: Department of Computer Science, University of Toronto, 2003.

[108] Beat Signer, Ueli Kurmann, and Moira C. Norrie. "iGesture: A General Gesture Recognition Framework". In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*. Curitiba, Parana, Brazil: IEEE, Sept. 2007, pp. 954–958. ISBN: 978-0-7695-2822-9. DOI: 10.1109/ICDAR.2007.4377056. URL: http://ieeexplore.ieee.org/document/4377056/.

[109] Arthur Sluÿters, Sébastien Lambot, and Jean Vanderdonckt. "Hand Gesture Recognition for an Off-the-Shelf Radar by Electromagnetic Modeling and Inversion". In: *IUI 2022: 27th International Conference on Intelligent User Interfaces, Helsinki, Finland, March 22 - 25, 2022*. Ed. by Giulio Jacucci et al. ACM, 2022, pp. 506–522. DOI: 10.1145/3490099.3511107. URL: https://doi.org/10.1145/3490099.3511107.

[110] Arthur Sluÿters et al. "QuantumLeap, a Framework for Engineering Gestural User Interfaces based on the Leap Motion Controller". In: *Proceedings of the ACM on Human-Computer Interaction* 6.EICS (June 2022), 161:1–161:47. DOI: 10.1145/3532211. URL: https://doi.org/10.1145/3532211.

[111] Arthur Sluÿters et al. "RadarSense: Accurate Recognition of Mid-air Hand Gestures with Radar Sensing and Few Training Examples". In: *ACM Trans. Interact. Intell. Syst.* 13.3 (Sept. 2023). ISSN: 2160-6455. DOI: 10.1145/3589645. URL: https://doi.org/10.1145/3589645.

[112] Kênia Sousa et al. "User interface derivation from business processes: a model-driven approach for organizational engineering". In: *Proceedings of the 2008 ACM Symposium on Applied Computing*. SAC '08. Fortaleza, Ceara, Brazil: Association for Computing Machinery, 2008, 553–560. ISBN: 9781595937537. DOI: 10.1145/1363686.1363821. URL: https://doi.org/10.1145/1363686.1363821.

[113] Gale Stam and Mika Ishino. *Integrating Gestures: The interdisciplinary nature of gesture*. Vol. 4. Gesture Studies. John Benjamins Publishing Company, June 2011. ISBN: 978-90-272-2845-1. DOI: 10.1075/gs.4. URL: http://www.jbe-platform.com/content/books/9789027287205.

[114] Jesus Suarez and Robin R. Murphy. "Hand gesture recognition with depth images: A review". In: *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, Sept. 2012, 411–417. ISBN: 978-1-4673-4606-1. DOI: 10.1109/ROMAN.2012.6343787. URL: http://ieeexplore.ieee.org/document/6343787/.

[115] Eugene M. Taranta II and Joseph J. LaViola. "Penny Pincher: A Blazing Fast, Highly Accurate $-family Recognizer". In: *Proceedings of the 41st Graphics Interface Conference*. GI '15. Halifax, Nova Scotia, Canada: Canadian Information Processing Society, 2015, pp. 195–202. ISBN: 978-0-9947868-0-7. URL: http://dl.acm.org/citation.cfm?id=2788890.2788925.

[116] Eugene M. Taranta II et al. "Jackknife: A Reliable Recognizer with Few Samples and Many Modalities". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, May 2017, 5850–5861. ISBN: 9781450346559. DOI: 10.1145/3025453.3026002. URL: https://doi.org/10.1145/3025453.3026002.

[117] Eugene M. Taranta II et al. "Machete: Easy, Efficient, and Precise Continuous Custom Gesture Segmentation". In: *ACM Transactions on Computer-Human Interaction* 28.1 (Jan. 2021). ISSN: 1073-0516. DOI: 10.1145/3428068. URL: https://doi.org/10.1145/3428068.

[118] Jean Vanderdonckt, Paolo Roselli, and Jorge Luis Pérez-Medina. "!!FTL, an Articulation-Invariant Stroke Gesture Recognizer with Controllable Position, Scale, and Rotation Invariances". In: *Proceedings of the 20th ACM International Conference on Multimodal Interaction*. ICMI '18. Boulder, CO, USA: ACM, 2018, 125–134. ISBN: 9781450356923. DOI: 10.1145/3242969.3243032. URL: https://doi.org/10.1145/3242969.3243032.

[119] Jean Vanderdonckt et al. "Head and Shoulders Gestures: Exploring User-Defined Gestures with Upper Body". In: *Design, User Experience, and Usability. User Experience in Advanced Technological Environments*. Ed. by Aaron Marcus and Wentao Wang. Cham: Springer International Publishing, 2019, pp. 192–213. ISBN: 978-3-030-23541-3. DOI: 10.1007/978-3-030-23541-3_15. URL: https://doi.org/10.1007/978-3-030-23541-3_15.

[120] Radu-Daniel Vatavu. "1F: One Accessory Feature Design for Gesture Recognizers". In: *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*. IUI '12. Lisbon, Portugal: ACM, 2012, pp. 297–300. ISBN: 978-1-4503-1048-2. DOI: 10.1145/2166966.2167022. URL: http://doi.acm.org/10.1145/2166966.2167022.

[121] Radu-Daniel Vatavu. "Gesture-Based Interaction". In: *Handbook of Human Computer Interaction*. Ed. by Jean Vanderdonckt, Philippe Palanque, and Marco Winckler. Cham: Springer International Publishing, 2023, pp. 1–47. ISBN: 978-3-319-27648-9. DOI: 10.1007/978-3-319-27648-9_20-1. URL: https://doi.org/10.1007/978-3-319-27648-9_20-1.

[122] 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. Denver, CO, USA: ACM, 2017, pp. 4667–4679. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025941. URL: http://doi.acm.org/10.1145/3025453.3025941.

[123] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. "Gesture Heatmaps: Understanding Gesture Performance with Colorful Visualizations". In: *Proceedings of the 16th International Conference on Multimodal Interaction*. ICMI '14. Istanbul, Turkey: ACM, 2014, pp. 172–179. ISBN: 978-1-4503-2885-2. DOI: 10.1145/2663204.2663256. URL: http://doi.acm.org/10.1145/2663204.2663256.

[124] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. "Gestures As Point Clouds: A $P Recognizer for User Interface Prototypes". In: *Proceedings of the 14th ACM International Conference on Multimodal Interaction*. ICMI '12. Santa Monica, CA, USA: ACM, 2012, pp. 273–280. ISBN: 978-1-4503-1467-1. DOI: 10.1145/2388676.2388732. URL: http://doi.acm.org/10.1145/2388676.2388732.

[125] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. "$Q: A Super-quick, Articulation-invariant Stroke-gesture Recognizer for Low-resource Devices". In: *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI '18. Barcelona, Spain: ACM, 2018, 23:1–23:12. ISBN: 978-1-4503-5898-9. DOI: 10.1145/3229434.3229465. URL: http://doi.acm.org/10.1145/3229434.3229465.

[126] Radu-Daniel Vatavu, Laurent Grisoni, and Stefan-Gheorghe Pentiuc. "Multiscale Detection of Gesture Patterns in Continuous Motion Trajectories". In: *Gesture in Embodied Communication and Human-Computer Interaction*. Ed. by Stefan Kopp and Ipke Wachsmuth. Vol. 5934. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 85–97. ISBN: 978-3-642-12552-2. DOI: 10.1007/978-3-642-12553-9_8. URL: http://link.springer.com/10.1007/978-3-642-12553-9_8.

[127] Radu-Daniel Vatavu and Stefan Gheorghe Pentiuc. "Multi-Level Representation of Gesture as Command for Human Computer Interaction". In: *Computing and Informatics* 27.6 (Jan. 2008), 837–851. URL: https://www.cai.sk/ojs/index.php/cai/article/view/16.

[128] Santiago Villarreal-Narvaez et al. "A Systematic Review of Gesture Elicitation Studies: What Can We Learn from 216 Studies?" In: *Proceedings of the 2020 ACM Designing Interactive Systems Conference*. DIS '20. Eindhoven, Netherlands: Association for Computing Machinery, 2020, 855–872. ISBN: 9781450369749. DOI: 10.1145/3357236.3395511. URL: https://doi.org/10.1145/3357236.3395511.

[129] Santiago Villarreal-Narvaez et al. "Brave New GES World: A Systematic Literature Review of Gestures and Referents in Gesture Elicitation Studies". In: *ACM Comput. Surv.* 56.5 (Jan. 2024). ISSN: 0360-0300.

DOI: 10.1145/3636458. URL: https://doi.org/10.1145/3636458.

[130] Santiago Villarreal-Narvaez et al. "Informing Future Gesture Elicitation Studies for Interactive Applications that Use Radar Sensing". In: *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*. AVI 2022. Frascati, Rome, Italy: Association for Computing Machinery, 2022. ISBN: 9781450397193. DOI: 10.1145/3531073.3534475. URL: https://doi.org/10.1145/3531073.3534475.

[131] Santiago Villarreal-Narvaez et al. "Theoretically-Defined vs. User-Defined Squeeze Gestures". In: *Proc. ACM Hum.-Comput. Interact.* 6.ISS (Nov. 2022). DOI: 10.1145/3567805. URL: https://doi.org/10.1145/3567805.

[132] TutorialsPoint Vineet Nanda. *Comparison Testing in Software Engineering*. 2021. URL: https://www.tutorialspoint.com/comparison-testing-in-software-engineering.

[133] Aiguo Wang et al. "A Comparative Study on Human Activity Recognition Using Inertial Sensors in a Smartphone". In: *IEEE Sensors Journal* 16 (June 2016), pp. 1–1. DOI: 10.1109/JSEN.2016.2545708.

[134] Chong Wang, Zhong Liu, and Shing-Chow Chan. "Superpixel-Based Hand Gesture Recognition With Kinect Depth Camera". In: *IEEE Transactions on Multimedia* 17.1 (Jan. 2015), pp. 29–39. DOI: 10.1109/TMM.2014.2374357.

[135] Mark D. Weiser. "The Future of Ubiquitous Computing on Campus". In: *Commun. ACM* 41.1 (1998), pp. 41–42. DOI: 10.1145/268092.268108. URL: https://doi.org/10.1145/268092.268108.

[136] Tracy Westeyn et al. "Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition". In: *Proceedings of the 5th International Conference on Multimodal Interfaces*. ICMI '03. Vancouver, British Columbia, Canada: ACM, 2003, pp. 85–92. ISBN: 1581136218. DOI: 10.1145/958432.958452. URL: https://doi.org/10.1145/958432.958452.

[137] Any Whitefield, Frank Wilson, and John Dowell. "A framework for human factors evaluation". In: *Behaviour & Information Technology* 10.1 (1991), pp. 65–79. DOI: 10.1080/01449299108924272. URL: https://doi.org/10.1080/01449299108924272.

[138] Andy Wilson. "Sensor- and Recognition-Based Input for Interaction". In: *Human-Computer Interaction Fundamentals (Human Factors and Ergonomics)*. CRC Press, Mar. 2009. Chap. 10, pp. 153–176. ISBN: 978-1-4200-8881-6. URL: https://www.microsoft.com/en-us/research/publication/sensor-recognition-based-input-interaction/.

[139] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. "User-defined Gestures for Surface Computing". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 1083–1092. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518866. URL: http://doi.acm.org/10.1145/1518701.1518866.

[140] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. "Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes". In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST '07. Newport, RI, USA: ACM, 2007, pp. 159–168. ISBN: 978-1-59593-679-0. DOI: 10.1145/1294211.1294238. URL: http://doi.acm.org/10.1145/1294211.1294238.

[141] Mais Yasen and Shaidah Jusoh. "A systematic review on hand gesture recognition techniques, challenges and applications". In: *PeerJ Computer Science* 5 (Sept. 2019), e218. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.218. URL: https://doi.org/10.7717/peerj-cs.218.

[142] Ionuţ-Alexandru Zaiţi, Ştefan-Gheorghe Pentiuc, and Radu-Daniel Vatavu. "On free-hand TV control: experimental results on user-elicited gestures with Leap Motion". In: *Personal and Ubiquitous Computing* 19.5–6 (Aug. 2015), pp. 821–838. ISSN: 1617-4909, 1617-4917. DOI: 10.1007/s00779-015-0863-y.

[143] Shumin Zhai et al. "Foundational Issues in Touch-Surface Stroke Gesture Design — An Integrative Review". In: *Foundations and Trends in Human–Computer Interaction* 5.2 (2012), pp. 97–205. ISSN: 1551-3955. DOI: 10.1561/1100000012. URL: http://dx.doi.org/10.1561/1100000012.

[144] N. Zheng et al. "User-Independent EMG Gesture Recognition Method Based on Adaptive Learning". In: *Frontiers in Neuroscience* 16 (2022), p. 847180. DOI: 10.3389/fnins.2022.847180.