Turning Gesture Sequences into Gesture Composition by Macro-Instructions: Definition, Framework, and Application in Smart Home

NUWAN T. ATTYGALLE, Université catholique de Louvain, Belgium
JEAN VANDERDONCKT, Université catholique de Louvain, Belgium
VIK PARTHIBAN, Massachusetts Institute of Technology, MIT Media Lab, United States of America

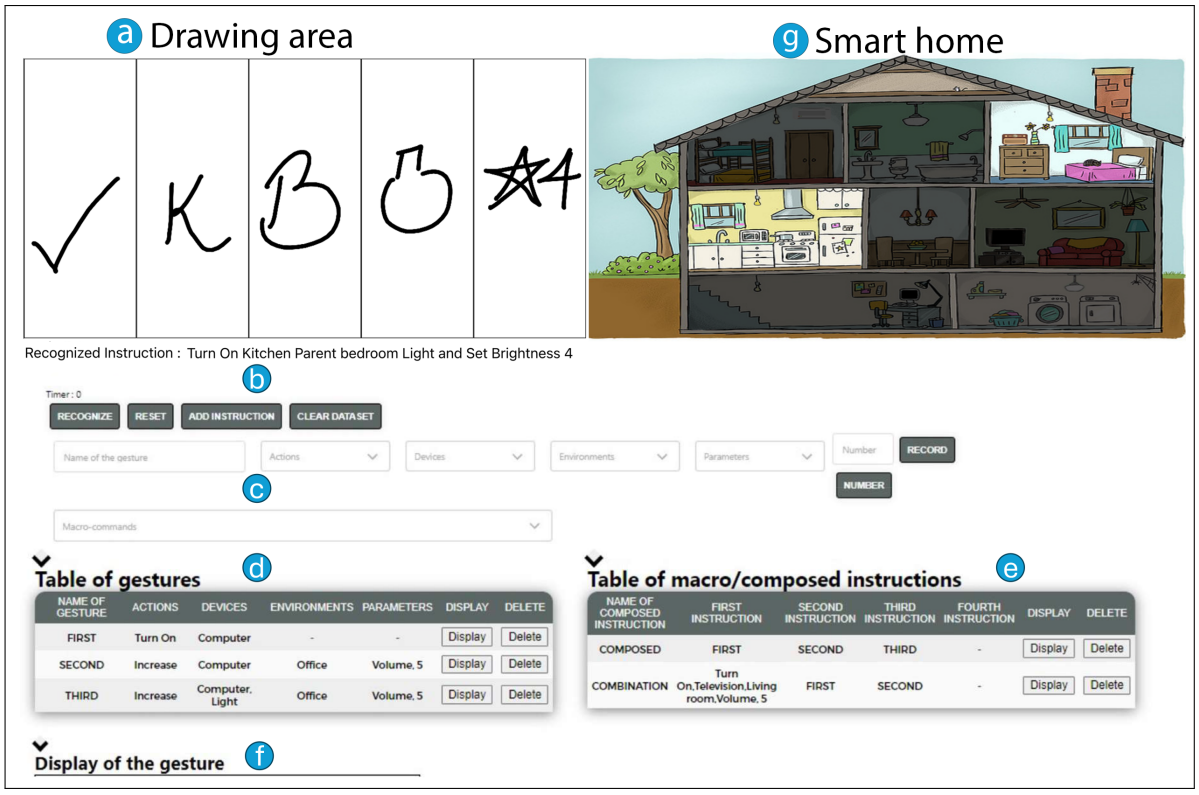

Fig. 1. Composition of elementary gestures into a compound gesture in EMERITI: (a) the end user performs one or many gestures (*e.g.,* 2D strokes in a drawing area), (b) elementary gestures are associated to an instruction consisting of an action (*e.g.,* "Turn on"), an object or a device (*e.g.,* "light"), an environment (*e.g.,* "kitchen", "bedroom"), any parameter (*e.g.,* "brightness") and its value (*e.g.,* "4"), (c) the gesture is recorded and associated to an instruction and instructions can be grouped into macro-instructions, (d) gestures appear in a table for editing, (e) instructions appear in a table for editing and control, (f) gestures are displayed in a drop-down list, and (g) a smart home receiving the execution of the instruction.

Authors' Contact Information: Nuwan T. Attygalle, Université catholique de Louvain, Louvain Research Institute in Management and Organizations (LouRIM), Louvain-la-Neuve, Belgium, nuwan.attygalle@uclouvain.be; Jean Vanderdonckt, jean.vanderdonckt@uclouvain.be, Université catholique de Louvain, Louvain Research Institute in Management and Organizations (LouRIM), Louvain-la-Neuve, Belgium; Vik Parthiban, vparth@mit.edu, Massachusetts Institute of Technology, MIT Media Lab, Cambridge, United States of America.

2D-3D gesture interaction most often associates a recognized gesture with a single command, such as "Swipe right" to navigate to the next screen. When gestures are performed one after another in a gesture sequence, each gesture still needs to be associated with one separate command at a time. Increasing the number of such gestures to realize more commands is not favorable to the end user, who memorizes and uses only a limited number of gestures. Instead of relying on gesture sequences, we can expand the range of executable commands to perform more complex tasks by gesture composition. To this end, we define an instruction as an action applied to a device in an environment, together with its valued parameters, such as "Increase the brightness of the bedroom lamp to 10". Elementary gestures, preferably congruent and hierarchical, are composed into a compound gesture. The instructions can be combined into macro-instructions to summarize gesture composition. To put these definitions in practice, we developed Emeriti, a framework for the recognition of elementary and compound gestures in both 2D (uni- or multistroke gestures recognized on an interactive surface) and 3D (point trajectories in space) by composition. We demonstrate the effectiveness and usefulness of turning gesture sequences into gesture composition using (macro-)instructions in an application simulating a smart home in which every device in every room can be controlled by compound gestures with parameters. This makes it possible to multiply the number of possible commands for more complex tasks without considerably increasing the number of gestures.

CCS Concepts: • **Human-centered computing** → **Graphical user interfaces**; Touch screens; **Gestural input**; User interface management systems; Command line interfaces; • **Software and its engineering** → *Software prototyping*; Macro languages; • **Theory of computation** → *Pattern matching*.

Additional Key Words and Phrases: 2D Stroke gestures, 3D Mid-air gestures, Compound gestures, Elementary gestures, Gesture-based interfaces, Gesture dataset, Gesture recognition, Interaction surfaces, Large display interfaces and multi-display environments, Software engineering methods and frameworks.

ACM Reference Format:
Nuwan T. Attygalle, Jean Vanderdonckt, and Vik Parthiban. 2026. Turning Gesture Sequences into Gesture Composition by Macro- Instructions: Definition, Framework, and Application in Smart Home. *Proc. ACM Hum.-Comput. Interact.* 10, 4, Article EICS016 (June 2026), 39 pages. https://doi.org/10.1145/3816768

1 Introduction

Gesture-based user interfaces (UIs) [109] and computing [35] today provide new opportunities for natural and intuitive interaction as they interpret human movements into computer commands using gesture recognition software [58, 126]. In the field of smart homes, for example, end-users can open their hand by performing a blossom gesture [83] to display the main menu, put their thumbs up/down to turn the light on/off [109]. The user experience of this gestural interaction primarily depends on the gesture vocabulary [121], its mapping to commands [124], and the gesture recognizer [10, 58, 71]. Several efficient algorithms exist today for recognizing a large number of two-dimensional (2D) stroke gestures [3, 4, 37, 64, 86, 102, 106–108, 110, 111] and three-dimensional (3D) mid-air gestures [22, 26, 39, 54, 58, 80, 96, 103].

The availability of such efficient recognizers suggests that a gesture vocabulary can be very rich, but this is rarely the case for several reasons. Firstly, in most cases, a gesture is mapped to a simple command [8, 9, 11, 33, 113], such as "Swipe right" and "Swipe left" to navigate to the next and previous item in a list [107]. There are few cases where the same gesture can be used in different contexts of use [49] or in different fields of application with different devices [41]. When a one-to-one mapping is established between a gesture and a command, more gestures must be introduced in order to perform more commands. One must introduce additional gestures proportional to the number of desired system commands or interactions.

Secondly, when multiple gestures are performed one after another in a *gesture sequence* [2, 48], each gesture still needs to be recognized individually to trigger a separate command. Gestures of a sequence are interpreted

ACM 2573-0142/2026/6-ARTEICS016
https://doi.org/10.1145/3816768

independently of each other, though they may form a chain leading to a larger task. For example, "Raise hand" should be performed repeatedly to reach a desired sound level. Instead, a gesture composition involves combining gestures together, typically simultaneously or overlapping, to produce a single, more complex command. The combination of gestures should be recognized as one compound gesture that triggers a single, higher-level command. Temporal relationship: gestures occur in time order. For example, "Raise hand" combined with five fingers should raise the sound level to 5. Gestures occur in time order in a gesture sequence (*i.e.,* a temporal relationship exists) while gestures are combined rather than ordered in a gesture composition (*i.e.,* a spatial or a concurrent relationship exists).

Thirdly, in practice, an end-user only memorizes and uses a limited number of gestures and therefore executes a limited number of commands, usually below the set of commands actually used. The number of gestures vary widely from one source to another: 3 or 4 [53], 7 [57], 7±2 if we rely on Miller's limit of short-term memory [53], 16 [113], and up to 21 [114] in Gesture Elicitation Studies (GESs) [119].

Consequently, increasing the number of gestures in a vocabulary with a one-to-one mapping to a command to perform more simple tasks and authorize more complex ones is not realistic for the end-user, who will not use these gestures anyway, and will restrict to a useful subset of gestures. To address these limitations, we propose Emeriti and, to evaluate our approach, we formulate research questions using the Goal-Question-Metric (GQM) framework [15, 101] in Table 1.

Table 1. Goal-Question-Metrics (GQM) formulation of our research question.

Goal	Purpose	Expand
	Issue	the expressivity of gestures to facilitate multi-step and parameterized commands
	Object	by turning gesture sequences into gesture compositions using Emeriti
	Viewpoint	from end users' perspective
Research Question	RQ_1	How to arrange elementary gestures of a gestures sequence into a gesture composition without proportionally expanding the size of a gesture vocabulary?
Metrics	M1	Gesture vocabulary size *vs.* command-set size (command expansion factor)
	M2	Interaction cost: required gesture submissions per task (*e.g.,* repetitions needed for parameter values)
Research Question	RQ_2	To what extent can variable-order gesture compositions be interpreted into the intended command at the task level?
Metrics	M3	Task Success Rate (strict command match)
	M4	Goal Completion Ratio (partial completion) and Task Completion Time (efficiency)
Research Question	RQ_3	To what extent does gesture composition impact subjective workload, usability, user experience, and perceived multi-step interaction quality?
Metrics	M5	Subjective workload, usability, and user experience scores (NASA-TLX, SUS, UEQ)
	M6	Custom post-study ratings on perceived multi-step interaction quality

Today, answering these questions remains an open and complex problem for several reasons: research and development into compound gestures is limited, the mapping of gestures to commands for performing simple and complex tasks is often limited to a one-to-one relationship, there is no particular study for investigating the potential composition of gestures into multiple levels of composition, ranging from simple compositions to high-level, possibly recursive, compositions, and there is no software environment specifically tailored for supporting compound gestures. For this purpose, this paper provides the following contributions:

- A definition of an elementary gesture, a sequence of elementary gestures, and a compound gesture based on two properties, *i.e.,* hierarchical and congruent gestures, together with the process of turning a gesture sequence into a compound gesture.
- A definition of new concepts useful for this transformation: an instruction as an action applied to a device in an environment, together with its valued parameters, and a macro-instruction, as an assembly of instructions.
- Emeriti (EnvironMent for composing gEstures InTo Instruction), a framework for the recognition of elementary and compound gestures in both 2D (uni- or multistroke gestures recognized on an interactive surface) and 3D (point trajectories in space) for composition. We also publicly release the code and the datasets used for this development, along with some more examples.
- Two application demonstrations (a virtual smart home and a coffee shop) illustrating the expressiveness of the proposed congruent hierarchical grammar and the ANTLR-based gesture composition pipeline in Emeriti.
- Two complementary user studies: (1) a formative pilot study (n=9) collecting exploratory feedback on Emeriti's interaction concept and interface (*e.g.,* Product Reaction Cards and open-ended comments), and (2) a controlled comparative study (n=18, nine participants per condition) evaluating task effectiveness, task completion time, gesture recognition performance, and subjective workload, usability, and user experience.

The remainder of this paper is structured as follows: section 2 reviews related work on gesture recognition, gesture composition, and gesture sequences; section 3 introduces the fundamental definitions underlying gesture composition in Emeriti; section 4 describes the implementation of Emeriti and its support for composing elementary gestures into compound gestures; section 5 illustrates the application of Emeriti in a simulated smart-home environment and a coffee-shop scenario, for which a full demonstration video is provided as supplementary material; section 6 reports two user studies evaluating Emeriti's gesture-composition approach against a sequencing baseline; section 7 discusses the limitations of the proposed approach and of the conducted user studies; and section 8 concludes the paper by highlighting the benefits of compound gestures over elementary gestures and outlining directions for future work.

2 Related Work

2D stroke gestures are those gestures performed by an end-user on a pen-[18], mouse-[125], or finger-sensitive interactive surface [119] to trigger functions, commands [124], their shortcuts [5], and to manipulate information [87], such as medical image annotation [24]. Similarly, three-dimensional (3D) mid-air gestures [1] are those gestures performed by an end-user in the air using various sensors, such as cameras [58], sensors [77, 99], wearables [23], wi-fi [59], or radars [7, 12, 34, 96, 112], worn on various body parts such as the head, the arm, the wrist, the hand, or the fingers [56]. Gesture recognition consists of classifying a *candidate* gesture based on a training set, *i.e.,* a series of samples called *templates*. The recognizers determine the template most similar to a metric and return its *gesture class*. Researchers distinguish *uni-* from *multistroke*, *i.e.,* when gestures are composed of multiple strokes, and *uni-* from *multi-touch*, *i.e.,* when gestures consist of multiple strokes performed at the same time (*e.g.,* by combining index and thumb).

2.1 Existing Recognizers and Frameworks for Gesture Recognition

There is a wide range of efficient **2D stroke gesture** recognizers [71]. For example, the $-family employs pattern matching [89] to classify a candidate after pre-processing all templates for satisfying *articulation*, *sampling*, *scaling*, and translation invariances [109], summarized as ARST-invariance [72]. $1 [3] and $N [120] use the golden section search to compare gestures at different angular alignments to ensure rotation invariance to recognize multistroke gestures with articulation invariance, *i.e.,* gestures are recognized regardless of variations in the number, order, and direction of their strokes. The $P [110], $P+ [108], and $Q [111] algorithms mitigate their issue of combinatorial explosion by employing cloud matching to compare any candidate with each template, *i.e.,* the gestures are considered as sets of points.

Protractor (Pro) [64] and $N-Protractor ($NP) [4] extend $1 and $N, respectively, by introducing a closed-form solution to compute the minimal angular distance between two gestures, thus outperforming their predecessors in accuracy, speed, and storage cost. Four variants exist depending on its extension with Protractor's closed-form solution and/or bounded rotation invariance: $N, $N-Protractor ($NP), $N-Rotation ($NR), and $N-Protractor-Rotation ($NPR), Penny Pincher (PP) [102] is a uni-stroke recognizer that reduces the pre-processing of the $-family to resampling and vectorizing of the gesture points. They satisfy sampling, translation, and scaling invariances. PP calculates the sum of the angles between pairs of corresponding vectors to assess the similarity between two gestures. !FTL [106] employs the local shape distance to assess the dissimilarity between two gestures.

The offer of efficient **3D gesture recognizers** is even wider. For example, Rubine3D generalizes the Rubine algorithm to 3D gestures as part of the iGesture workbench [95]. Jackknife [103] recognizes both 2D and 3D gestures in multimodal settings. It promotes dynamic time warping using the inner product (JIP) or the Euclidean distance (JED) as the local cost function in addition to correction factors and rejection criteria. The $3 recognizer [54] extends $1 to 3D by recording mid-air gestures using the accelerometer data of a single point issued by a Nintendo Wii Remote Controller device. $3 can be adapted to another sensor provided that acceleration data are available. From a reasonably small training set, $3 efficiently recognizes between 10 and 15 gestures. Protractor3D [55] adds rotation invariance to $3 by finding the rotation that minimizes the Euclidean distance. The uWave recognizer [67] exploits the data from a three-axis accelerometer, which is particularly appropriate for wearable devices such as smartwatches and smartphones. Raw data are compressed by an averaging window, are quantized non-linearly, and matched with another time series by Dynamic Time Warping (DTW) to end up with a calculation of the Euclidean distance. Very similarly to $3 and uWave, Madani et al. [70] recognize 3D unistroke arm gestures in thin air (*i.e.,* digits from 0 to 9 and mathematical operators for addition, subtraction, multiplication, and division) based on accelerometer data. More recently, the 3¢ algorithm [26] extends the 1¢ [47] with the same principles as in $3 and Protractor3D: comparing sampled 3D trajectories, but with different and simpler processing options. 3D gesture recognition is a wide field of research and development in view of the device proliferation, wearable or not, human body parts considered for gesturing, and techniques used for recognition. Many techniques are used [31, 123]. Several Integrated Development Environments (IDEs) offer various levels of support in developing gesture-based UIs using one or many recognizers. Cirelli and Nakamura [32] survey several frameworks for gesture recognition, which can have the form of a framework (*e.g.,* iGesture [95], QuantumLeap [97]), a toolkit (*e.g.,* the Gesture Recognition Toolkit [42], programming facilities [50], and real-time recognizers [27, 92]. All these recognizers and frameworks are efficient for recognizing one gesture at a time.

2.2 Gesture Composition

The literature has explored gesture composition in various ways, mainly by considering just two elementary gestures, one at a time. Billon et al. [19] introduce a multi-agent system that recognizes a gesture sequence formed

Table 2. Comparison of selected prior works related to the composition of gestures, sorted in increasing order of appearance (Gest. = Gestures, Cmd. = Commands, Obj. = Objects, Env. = Environments, Par. = Parameters, Val. = Values. An empty space represents an unsupported feature.

Reference	# Dim.	# People	# Hands	# Gest.	# Cmd.	# Obj.	# Env.	# Par.,Val.
Cooperative gestures (2006) [76]	2D	*n*	1	3	1	1		
Shortcuts (2009) [5]	2D	1	1	1	1			
UsiGesture (2012) [18]	2D/3D	1	1	2	1			
AugmentedLetters (2013) [88]	2D	1	1	2	1			
FingerCount (2014) [56]	3D	1	1	1-5	1			
GestIt (2012) [100]	2D/3D	1	1	*n*	*n*	1		
Summon & Select (2017) [44]	3D	1	1	2	1,2			
Cross-device interaction (2019) [84]	3D	1	1	2	1,2			
On Gesture Combination (2019) [38]	3D	1	1	2	1,2			
Mid-air interaction (2021) [115]	3D	1	1	2	1	1		
Address & Command (2022) [116]	3D	1	1	2	1	1		
Multi-type gestures (2023) [105]	3D	0	1	*n*	*n*	1		3
Consecutive gestures (2024) [122]	3D	1	1	2	2			
Emeriti (2026): this paper	2D/3D	1	2	*n*	*n*	*n*	*n*	*n*

by only two gestures, one gesture performed by a real actor in a theater context and one performed by a virtual actor. This work demonstrates the relevance of recognizing gesture sequences, even if the sequence is limited to two gestures. Cao and Zhai [25] present a quantitative model of human performance of one 2D uni-stroke gesture at a time in terms of production time based on the properties of curves, line segments, and corners. 2D elementary gestures [5] used as shortcuts for commands have cognitive advantages over keyboard shortcuts: an elementary 2D gesture is always mapped to a simple command (*e.g.,* drawing a "V" to launch the media player) or a single object (*e.g.,* issuing an "α" to select "Mushroom" in a menu). Zhai et al. [124] review foundational research on designing stroke gestures, including factors like motor control, feedback, and chunking, but does not specifically address composing elementary gestures into compound gestures. AirStroke [77] brings unistroke text entry to freehand gesture interfaces by combining mode selection and character entry with one hand and word completion with the other. Aussems et al. [13] uses a pattern-based learning system to identify sequences of speech-accompanying gestures and relates these to empathy empathy scores as meta-information to classify the gesture sequences. Again, pattern matching is appropriate, but for speech-accompanying gestures here, not for system command.

The existing body of literature has predominantly examined gesture-based UIs in which multiple gestures execute a single command. Morris et al. [76] examined how different people issue collective gestures to obtain a shared command. One common approach involves decomposing a command into two distinct gestures: one specifying the command itself and the other identifying its target. For example, the "Address and Command" [116] utilizes the non-dominant hand to specify the target device (*e.g.,* a smart TV or window blinds) while simultaneously executing a command gesture (*e.g.,* increasing or decreasing a parameter) with the dominant hand. A similar paradigm is adopted by Delamare et al. [38], where the target is conceptualized as an object of the command (*e.g.,* an audio file, a video, or a display) rather than a specific device. Summon and Select [44] follows a comparable strategy, employing a pantomimic target gesture followed by a deictic gesture with the same hand to manipulate a control. Basically, the end user performs two elementary gestures consecutively, one for the target and one for the command using a single hand [115], such as a first gesture to copy an element on

one device and a second gesture to paste it on another device [84]. FingerCount [56] exploits multiple gestures to navigate within a menu to execute a single command. Haptic feedback is also studied between two consecutive gestures [122], but how elementary gestures can be sequenced to execute more commands is left untreated.

The definition of a gesture is the subject of a compositional model based on Petri Nets [100] or Hidden Markov Models in G-Gene [28], allowing for flexible [6] and granular gesture recognition. These models address the composition of the gesture primitives (*e.g.,* a line, a semicircle, a circle) that constitute a 2D gesture to enable online partial stroke gesture recognition [27], but it does not cover the composition of gestures between them. The composition is located at a finer level of granularity than ours: Spano et al. [100] compositional model for gesture definition, notably implemented through GestIT [100] and DEICTIC [28], defines a complex 2D/3D gesture by combining basic input sensor traits using six temporal operators (*i.e.,* iteration, sequence, parallel, choice, disabling, and order independence) in declarative rules. In this way, it is possible to specify any individual gesture using a Petri Net involving these rules for any fine-grained action performed for a single gesture. GestureScript [69] enables creating accurate unistroke gesture recognizers by combining example-based learning with declarative guidance through rendering scripts and interactively trained parts. Goguey and Ortega [43] empirically evaluate 30 concurrent stroke gestures on touch devices, but not their composition.

In Human-Robot Interaction (HRI), a robot can interpret a sequence of gestures as a single complex composition [104]: for example, the sequence "Move cup on top of a can with 50% speed and rotate it by 45°" is composed of [105] action gestures (*e.g.,* a combination of a grab gesture (closed fist) and a swipe right gesture), deictic gestures (*e.g.,* pointing gestures used to identify the target objects: the cup and the can), and metric parameters (*e.g.,* specific physical constraints provided via hand gestures, indicating (likely height/clearance) and a rotation). The idea of structuring a vocabulary for gesture composition is similar to ours in terms of actions and parameters.

2.3 Gesture Sequences in Social Sciences

Research on *gesture sequences* [13, 65], defined as a sequence of individual gestures performed by a subject at once, in social sciences has explored their use in various contexts of use. In human communication, pattern-based sequence classification of gestures can predict empathy levels more effectively than analyzing isolated gestures [13]. Chimpanzees use gesture sequences, often repeating tactile gestures in play contexts, but do not strategically combine gestures to attract attention before visual communication [65]. In gesture recognition, shared parameter models can learn common sub-gestures across different gestures, improving recognition performance [73]. In design communication, archetypal gesture sequences (*e.g.,* compound reflective, mirroring) play crucial roles in idea evolution and developing shared understanding among team members [30]. Wang et al. [117] introduces a discriminative hidden-state approach for recognizing gesture sequences. Bobick and Wilson [20] presents a state-based approach for representing and recognizing gesture sequences. Gesture sequences contain additional information compared to isolated gestures and are important in various forms of communication, from human empathy to primate interaction and design conceptualization. Hinrichs and Carpendale [48] reported in a field study investigating multi-touch gesture sequences on interactive tabletop exhibits that "gestures are not executed in isolation but linked into gesture sequences where previous gestures influence the formation of subsequent gestures," and that gesture choice is influenced by both interaction and social context. Nothing is said about how the sequence can be formed though.

2.4 Summary

These aforementioned recognizers, frameworks, and approaches aim to improve gesture recognition, provide structural information, and facilitate the development of gesture-based UIs, but do not closely examine how elementary gestures can be composed to issue more simple tasks or more complex tasks. They are typically limited to one or two gestures that result in a single command. Table 2 gives an overview of the selected works

related to the composition of gestures in Human-Computer Interaction (HCI) and compares them against our bottom line, as addressed in this paper.

When combining elementary gestures into gesture sequences, arbitrary mappings can emerge, potentially increasing cognitive load and learning curves for end users [78]. In linguistics, commands typically structure an action verb with an object [75]. However, in HCI, two paradigms coexist [93]: the action+object approach, mirroring linguistic patterns (*e.g.,* "Print a document"), and the object+action approach [17], where the object is selected first, followed by the action (*e.g.,* dragging a file icon onto a printer icon), reflecting direct manipulation. Applying both paradigms to multiple actions and objects in any order leads to a combinatorial explosion of vocabulary. Assigning a unique elementary gesture to each combination becomes impractical. To fulfill this requirement, the system should be accessible to users regardless of their native language, enabling intuitive ordering based on familiar linguistic structures. Speakers of Subject-Verb-Object (SVO) languages like English or French may prefer action-first sequences (*e.g.,* "Increase brightness"), while those using Subject-Object-Verb (SOV) languages like Sinhala or Japanese opt for object-first permutations (*e.g.,* "Brightness increase") [94]. This flexibility reduces cognitive barriers and enhances cross-cultural usability.

In conclusion, while individual aspects of gesture combination—such as decomposing a command into two sequential gestures [38, 115] or structuring a vocabulary hierarchically for better recall [68, 88]—have been explored in prior work, no existing framework has unified *all* of the following: an instruction-level grammar supporting any permutation of Action, Device, Parameter, Environment, and Macro elements; direct encoding of numeric parameter values within a single compound gesture submission; macro-instruction grouping for multi-step routines; and applicability to both 2D stroke and 3D mid-air modalities. Our contribution is therefore not the concept of composing two gestures per se, but the formal, grammar-based extension of composition to an arbitrary number of heterogeneous elements—including numerically valued parameters—and its systematic empirical validation against a sequential baseline that directly measures the cost of *not* composing parameter values. The empirical evidence base and development experience for gesture composition at this level of expressivity remain modest but inspiring, and Emeriti represents a step toward closing this gap.

3 Fundamental Definitions

To address the aforementioned issues, this section distinguishes an elementary gesture from a sequence of such gestures and, after that, introduces our fundamental definitions to turn a sequence into a composition.

3.1 Initial Definitions

We hereby refer to an *elementary gesture* as an atomic unit of movement performed by any body part(s), such as the fingers, hands, head, or other feet, to indicate an intention of executing a command. Such a single, atomic "movement" starts from a rest position and returns to it, thereby delineating a *gesture unit*. Elementary gestures are often spontaneous and represent simple, immediate ideas (*e.g.,* a "yes" nod or a "bye" wave). They are typically categorized into deictic (pointing), representational/iconic (depicting shape), and beats (rhythmic movements). For example, a pointing finger, a shrug of the shoulders. The literature contains a huge body of such gestures.

In contrast, we hereby refer to a *sequence of gestures* as a suite of multiple elementary gestures performed in succession in a deliberate, ordered manner to organize a complex meaning. They represent more complex ideas, narratives, or procedural actions. They can involve transitions between different gestures, such as moving from a "palm-up" (open) to a "palm-down" (closed) position. For example, A "Palm Presenting" gesture followed immediately by a "Vertical Palm" gesture to represent an argument followed by a constraint or a "swipe-and-tap" sequence.

3.2 New Definitions: Instructions and Elements

We now introduce some fundamental definitions for expressing a set of actions on one or more objects in one or more contexts of use [40], each with its parameters and values, in any order: (For the simplicity of the user, from now on, Action, Device, Parameter, Environment and Macro are shown in these particular colors.)

- Action: an action to be performed on one or many targets, objects, or devices, *e.g.,* "Turn on", "Set" [75].
- Device: a device on which an action will be executed, *e.g.,* a television, radio [36].
- Parameter: a variable of any target, such as a device, with its value to control it, *e.g.,* "Channel on 5", "Volume on 3", "Brightness 10" [73].
- Environment: a location where the action will be executed, *e.g.,* a kitchen, a living room, a bedroom [36].
- Macro: a word or a sentence which includes one or more previously created instructions, *e.g.,* All (contains all created instructions), macro_lights (contains all instructions that include modifications to the lights).

Based on these definitions, we created an Extended Backus-Naur Form (EBNF) [79] expression for an instruction:

Instruction := Action {Device} [{Parameter}] [{Environment}] | {Macro }

This definition is expressed in a logical order that allows any permutation between the different elements of the statement, *e.g.,* "Switch channel TV to 5", "Switch TV channel 5", "TV channel switch 5", and "Channel 5 TV switch" are all equivalent from this viewpoint. We now add new definitions:

(1) Basic instruction = Action Device [{Parameter}][{Environment}]. A basic instruction is an action that optionally applies one or many parameters on a device, in one or many environments (Fig. 2).

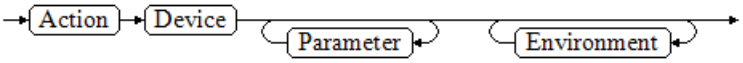

Fig. 2. Syntax diagram of a basic instruction.

Since the EBNF visualizer cannot represent the syntax diagram of all permutations of a statement, let us keep in mind that any permutation is possible, thus leading to other syntax diagrams (Fig. 3).

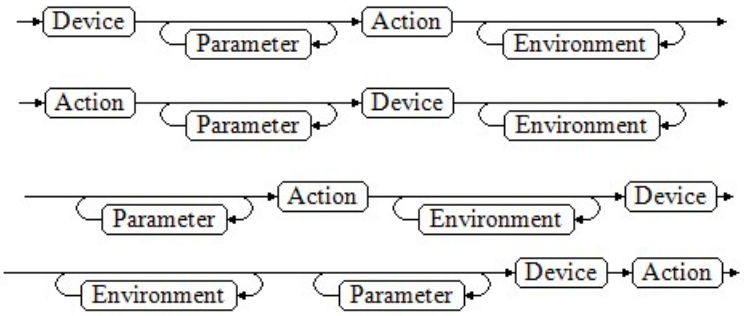

Fig. 3. Syntax diagram of some permutations of a basic instruction.

(2) Common Instruction = Action {Device} [{Parameter}] [{Environment}]. A common instruction is an action that applies zero to many parameters on one or many devices in one to many environments (Fig. 4).

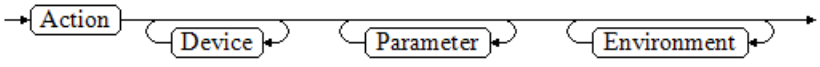

Fig. 4. Syntax diagram of a common instruction.

(3) Macro Instruction = Macro . A macro instruction is a shortcut for composing any other set of instructions, basic and common. To perform several instructions at once without having to make a gesture for each one, a macro instruction will combine the different instructions into a single one, that can be attached to an elementary gesture. For example, the macro instruction "Macro living room" combines the instructions "Turn on the lights in the living room" and "Turn on the television on channel 3".
(4) Composed Instruction = {Common instruction} | {Basic instruction} | {Macro instruction}. A composed instruction is composed of one or more common and/or basic and/or macro instructions (Fig. 5). Although a basic instruction is technically a specialisation of a common instruction (one Device vs. many), the distinction is maintained for two reasons: it provides a simpler entry point for the common single-device case, and the ANTLR parser encodes them as separate productions, reducing lookahead overhead when only one Device token is present.

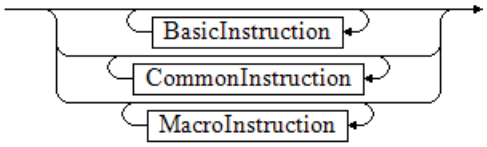

Fig. 5. Syntax diagram of a composed instruction.

(5) Element = Action | Device | Parameter | Environment. To enable the most flexible expression of an instruction (and later on of its associated gestures), we define an element as a unique action, device, parameter, or environment. The composition of these elements will be used later to create any composition of gestures that will be mapped on instructions, whatever their type is.
(6) Composed Element = ([Action] | [{ Device }]) [{Parameter}] [{Environment}]. A composed element is a composition of elements, having zero or one action, zero or many devices followed by optional parameters with their values and/or environments. The only condition is that a composed element cannot just have an action on a device, which would be otherwise a basic or a common instruction.

3.3 Composition of Elementary Gestures into Compound Gestures

We now normally define the difference between an elementary gesture and a compound one:

(1) Elementary Gesture = Basic instruction | Common instruction | Macro-instruction | Composed element
(2) Compound gesture = {Gesture}

An elementary gesture is therefore associated with either any instruction, basic, common, macro or a composed element. These gestures can be composed to create a combination of gestures and, therefore, new instructions. In this way, end users can create sets of gestures with full combinations of gestures with reusability. Furthermore, gesture composition can improve recall compared to simple gestures, Compound gestures are preferred to a set of simple gestures when they have an abstract gesture-command correspondence [38].

To visualize how an instruction is processed, Fig. 6 shows a Nassi-Schneiderman diagram [118], a graphical design representation for structured programming, of an instruction. A composed instruction includes a basic, a common, or a macro-instruction. For a basic instruction, we look until there are no more elements in the basic instruction. If an element still exists, we examine whether an action or device has not been considered because a basic instruction accepts only one action and one device. If a parameter or an environment element appears, we accept it. This same procedure is applied to a common instruction, with several devices. A command macro only accepts macros.

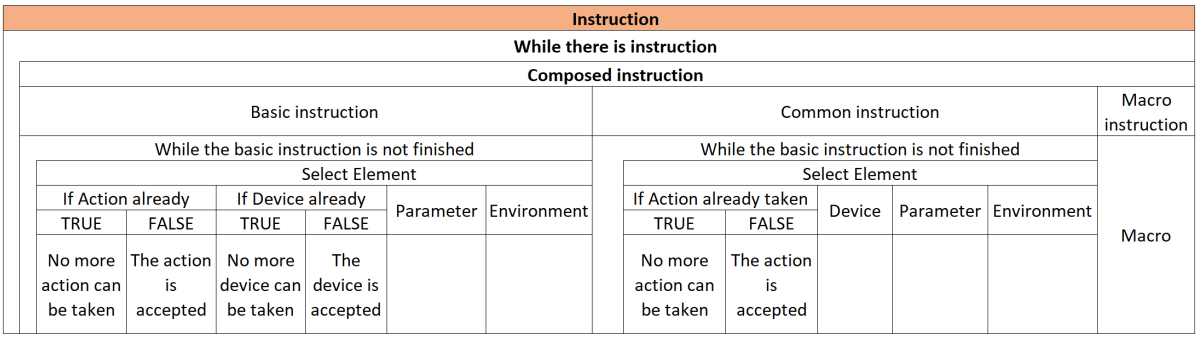

Fig. 6. Nassi-Schneiderman diagram of an instruction.

3.4 Composition of 2D Stroke Gestures by Congruence and Hierarchy

Before composing elementary gestures into compound ones, there is a need to study which gestures are appropriate for composition and how. Carroll [29] conducted a comparative analysis of four variations of a 16-command language focusing on two structural properties: *congruence*, which refers to the symmetry between pairs of semantically related commands, and *hierarchy*, which follows a structured "verb-object-qualifier" format. For instance, command pairs such as "Advance/Retreat" and "Right/Left" exhibit congruence, whereas "Go/Back" and "Turn/Left" do not. Commands adhering to both congruence and hierarchical organization yielded the highest subjective ratings, the best performance in problem-solving tasks, the lowest error rate, and the lowest omission rate. Thus, Carroll [29] hypothesized that the congruent and hierarchical structure facilitates recall.

Applying this principle to gesture-based UIs, particularly for 2D stroke gestures [98], necessitates an investigation into the process of *gesture composition*—how individual gestures can be systematically combined. Various composition strategies have been proposed, including spatio-temporal combination [38], concatenation [88], and assembly [116]. Hierarchical gestures are particularly well-suited for smartphone interaction in scenarios requiring a large set of commands, as they have been shown to enhance learnability, improve expressivity, and yield higher subjective satisfaction compared to non-hierarchical alternatives [68].

For instance, the "Augmented Letters" [88] employs a unistroke representation of the command's initial letter (*e.g.,* "T" for "Turn"), followed by a straight tail indicating a parameter (*e.g.,* a rightward flick for "Right"). This method enables the construction of extensive gesture vocabularies that enhance discoverability and recall without compromising efficiency [88]. Similarly, Delamare et al. [38] extended this principle to 3D mid-air gestures by developing a hierarchical gesture vocabulary. Their study demonstrated that hierarchical gestures lead to improved recall rates, as participants naturally structured gesture vocabularies for reusability, ultimately improving memorability compared to single gestures. Loch [68] did the same for shortcuts on a touchscreen.

Fig. 7 shows the difference between gestures that are congruent and hierarchical, and those which are not [98]. In the case of non-congruent, non-hierarchical gestures, a single gesture is mapped onto any single command (see fourth column in Fig. 7): the single gestures for "Turn TV on" and "Turn TV off" would be "Draw a circle in the clockwise direction" and "Draw a cross", respectively. Gestures mapped to related commands, such as symmetric commands, are independent. When they become hierarchical (see third column in Fig. 7), one gesture is mapped for the action and another one for the object, but they are still independent of the commands. Hierarchical gestures are gestures in which we map every single action to a sequence of gestures. For example, "Circle" represents "Turn", "Rectangle" is for "TV", and "V" for "on". Yet, these sequences remain independent as they do not necessarily reuse the same internal mappings. In the case of congruent and hierarchical gestures (see first

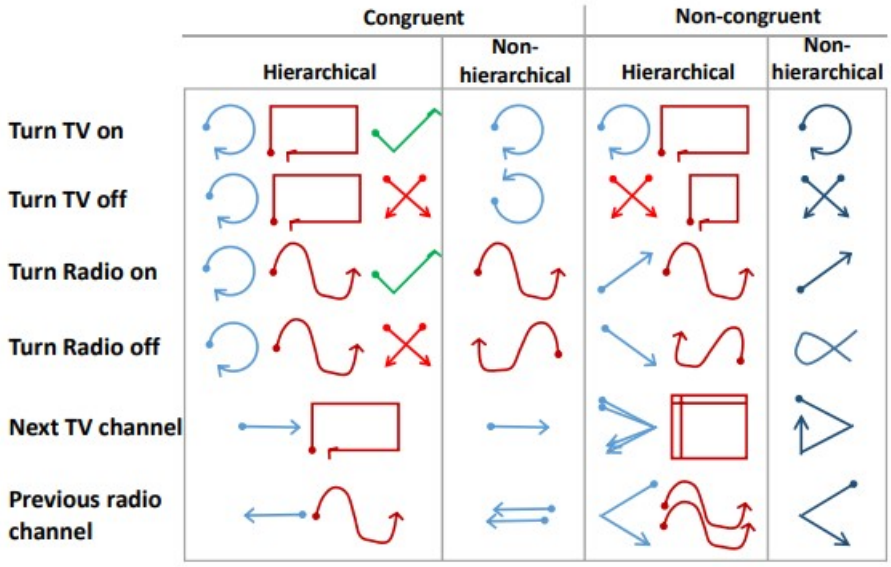

Fig. 7. Comparison of 2D stroke gestures by congruence and hierarchy.

column in Fig. 7), individual gestures are consistently mapped to any command element. For example, "Turn TV on" is decomposed into "Turn" (Draw a clockwise circle), "TV" (Draw a clockwise rectangle), and "on" (Draw a check"). Any individual gesture is then reused in any other command using the same command element.

4 The Emeriti Framework

This section describes our implementation of Emeriti (Environment for composing gestures into instruction) (Fig. 1), a framework for the recognition of elementary and compound gestures in both 2D and 3D for composition based on the concepts defined in the grammar in section 3. We first overview the parsing of this grammar to enable the end user to issue an instruction with its elements in any order to create any command and to support both paradigms.

4.1 Parser Generator

To correctly manipulate our EBNF grammar, we needed a parser generator like Xtext and ANother Tool for Language Recognition (ANTLR) [21]. Unfortunately, Xtext does not allow any permutation as it does not accept two permutations starting with the same element ending differently. For example, Xtext produces an error when parsing "Action Device Environment Parameter " and "Action Device Parameter Environment ". Therefore, we realized our grammar with ANTLR [82], a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It is widely used to build languages, tools, and frameworks. From our grammar, ANTLR generates a parser that can build and traverse parse trees, thus proving that our grammar is syntactically correct and checking, processing any instruction according to it. We created a rule for any instruction, which can be a basic, common, macro, or composed instruction. The composed instruction can recursively contain basic instructions, common instructions, and macro instructions. We defined the basic instruction as the set of possible permutations including one action, one device, one or many optional parameters, and one or many optional environments.

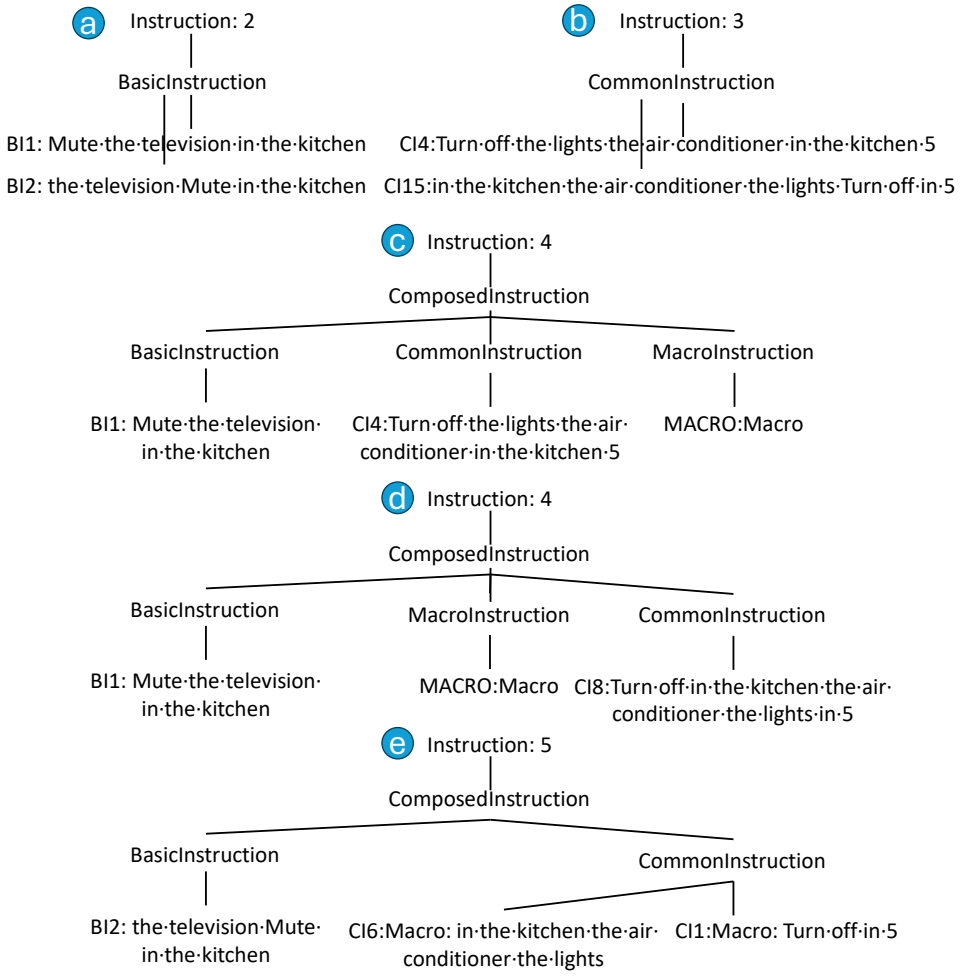

Fig. 8. Syntax trees of various instructions: (a) logical basic instruction with a permutation, (b) a common instruction with a permutation, (c) a composed instruction with a macro, and (d) a permuted composed instruction, and (e) a composed instruction with a basic and a common instruction.

Fig. 8-a, b show the correct syntax tree of a basic instruction "Mute the television in the kitchen " and of its permutation "the television Mute in the kitchen ", resp. The common instruction is similar to the basic instruction except that we can have two or more devices (Fig. 8-c): "Turn off the lights and the air conditioner in the kitchen in 5 minutes" and a permutation "in the kitchen the air conditioner the lights Turn off in 5 minutes " (Fig. 8-d).

A macro instruction is represented by a macro rule, its correct tree (Fig. 8-d) is found when a macro instruction arrives. To process a composed instruction, the most sophisticated composition, we needed to ensure that all the different types of instructions were well identified by processing two sentences: one in a logical sense and the other being any permutation. For example, "Mute the television in the kitchen || Turn off the lights and the air conditioner in the kitchen in 5 minutes || Macro " and " Mute in the kitchen the television|| Macro || Turn off in the kitchen the air conditioner the lights in 5". Since the syntax for basic, common, macro, and composed instructions is correct, our EBNF grammar is syntactically correct. After defining the three elements Action Device Parameter,

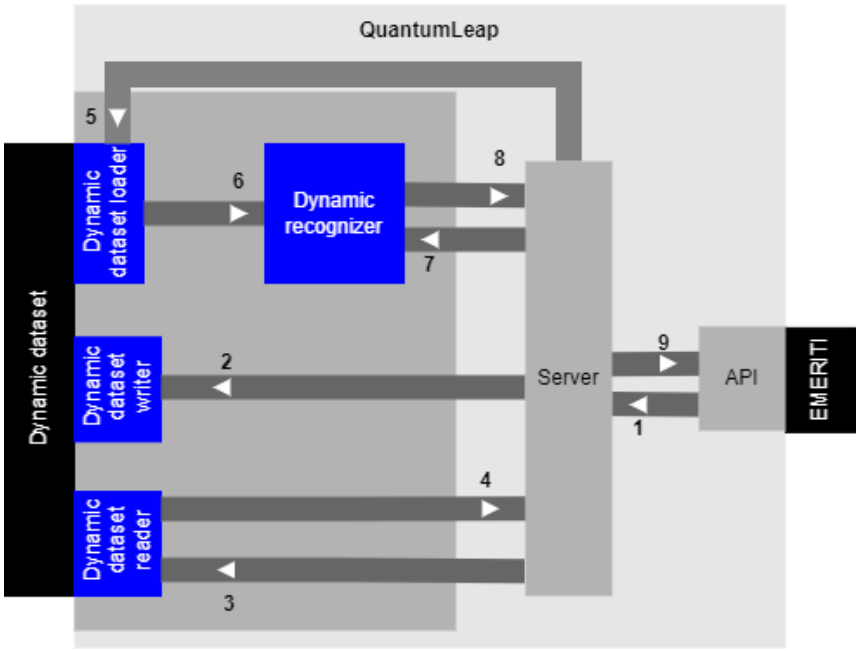

Fig. 9. Pipeline configuration of software components in Emeriti.

we created Environment and Macro as the fourth and fifth elements. When a composed instruction does not start with an action, the parsing could be confusing. For example, the composed instruction "Turn on the television Turn off the radio" permuted as "Turn on the television the radio Turn off " (Fig. 8-e) could lead to considering "Turn on the television the radio", " Turn off". To avoid this confusion, we added the rule of having to start with an action when a basic or common statement is used inside a composed instruction.

4.2 System Architecture and Data Flow

Emeriti is a framework implemented in JavaScript (ECMAScript 2019) [51] integrated with the parser generator (subsection 4.1) and QuantumLeap [97], a gesture recognition environment, via quantumleapjs, an interface that processes gestures following a structured pipeline configuring these software components (Fig. 9), which regulates the interactions between Emeriti and QuantumLeap using the following data structure:

(1) A standardized frame.
(2) A Sample object.
(3) A JavaScript object with three properties: the gesture type (*i.e.*,, static or dynamic, 2D/3D), the gesture name, and the data from the analyzer if the gesture is static.
(4) A WebSocket message of type Data containing frames, static, and dynamic gestures according to its WebSocket API, which is very convenient for designing interactions across devices in smart environments [90].
(5) A WebSocket message of type Operation.

These modules transform a gesture dataset into a GestureSet object, *i.e.*, a standardized format compatible with Emeriti. Datasets are then used to train the static and dynamic recognizers. Dynamic recognizers analyze sequences of frames instead of individual frames. Each recognizer returns a confidence score for each recognized gesture and implements three methods:

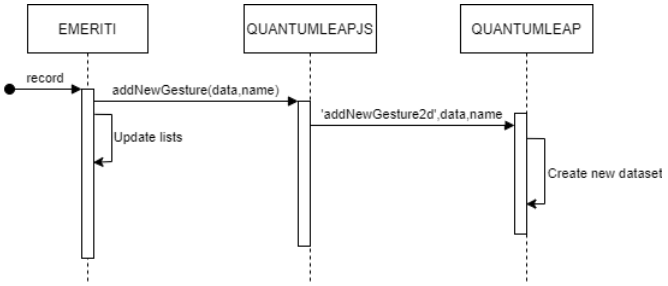

Fig. 10. The UML sequence diagram of the process of gesture recording.

- addGesture(name, sample): adds a gesture to the training set of this recognizer.
- removeGesture(name): removes a gesture from the training set of this recognizer.
- recognize(sample): determines the name of the gesture that corresponds to the candidate given as argument.

The QuantumLeap JavaScript API exposes the GestureHandler class, which is a member of the EventEmitter class of the Node.js core API. A GestureHandler object connects to a running instance of the QuantumLeap framework and then emits events based on the data received from QuantumLeap. From the recognizers reviewed in Section 2, we chose to equip Emeriti with the $P+ Point-Cloud classifier [108] to perform 2D multi-stroke gesture recognition. For previously trained gestures, this classifier has demonstrated 98% accuracy across 12 gesture classes and benefits from invariance properties. Furthermore, $P+ is associated with a P^3_+, a 3D extension of $P+ to recognizer gestures issues in 3D in terms of traces or trajectories of 3D points [81]. If required, a recorded gesture can be subjected to a procedure to increase the number of templates, in particular by 2D or 3D synthesis [61, 74]. The recording of gestures follows a structured sequence (Fig. 10):

(1) The user initiates gesture recording by executing the "Record" button in the Emeriti environment.
(2) Depending on the type of command, the user may create:
   - A basic instruction (single gesture-to-command mapping).
   - A composed instruction, constructed using lists and macro-instructions.
(3) Emeriti sends a request to QuantumLeapJS, which contains:
   - The gesture name selected by the user.
   - The gesture's data points collected from the gesture input (*e.g.*, a 2D drawing area, a 3D gesture recorder).
(4) Emeriti then updates the gesture database, ensuring that each instruction is correctly associated with its respective action, device, and optional parameters by parsing the EBNF grammar defined in subsection 4.1.
(5) The request is forwarded to QuantumLeap, where:
   - A new gesture dataset is generated in JSON format.
   - A gesture template is created and stored for subsequent recognition.

5 Application to a Smart Home

5.1 Prototype Smart Home Simulator

To demonstrate the applicability of Emeriti in a realistic multi-device setting, we implemented a virtual smart-home simulator (Fig. 11), inspired by the *Address and Command* model [116]. The simulator provides a controlled evaluation environment without the cost and logistics of deploying multiple physical IoT devices. It supports multi-device and multi-environment control and allows us to instantiate and test the instruction structures defined by our grammar in a scalable setting. The simulator is implemented in JavaScript on top of Emeriti. Interactive device feedback is rendered using opacity-based image swapping to animate state changes. A short video of the simulator is included in the supplementary material.

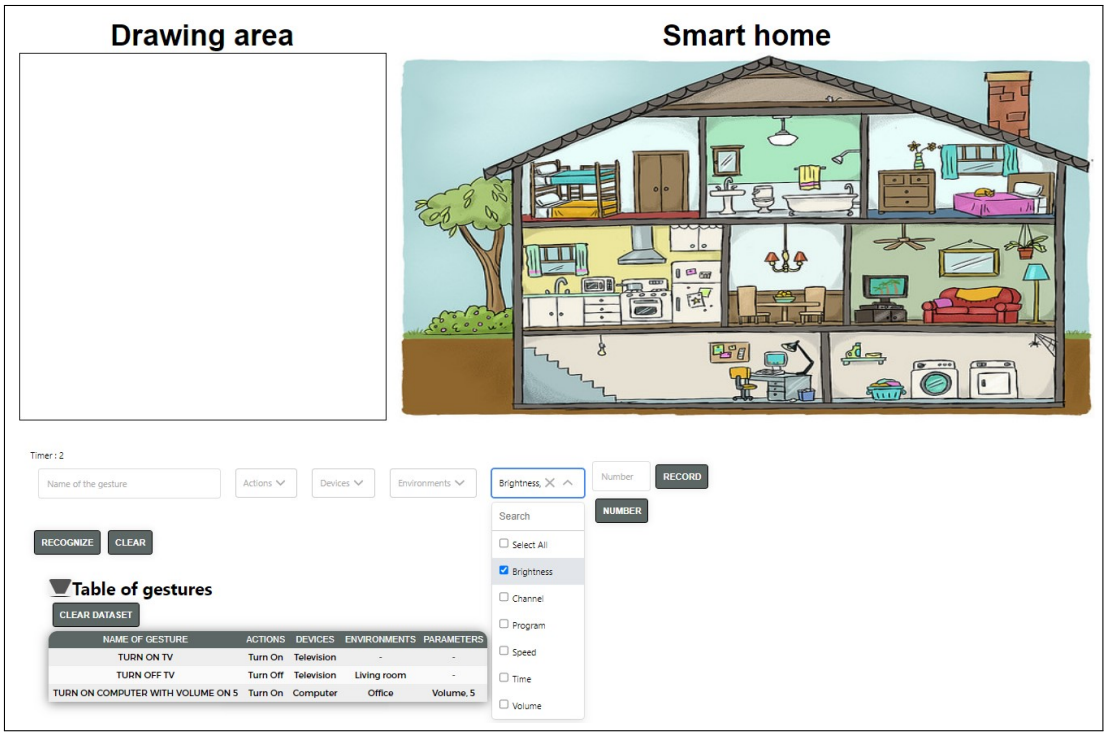

Fig. 11. Screenshot of the virtual smart home simulator.

5.2 Interaction Workflow

The simulator supports two phases: (i) *gesture authoring* and (ii) *gesture execution*. During authoring, a user draws an elementary or compound gesture, assigns it a name, and links it to one or more instruction components (Action, Device, Environment, Parameter, and value). The system records this mapping and stores it for later use. During execution, the user draws the gesture and submits it for recognition; the recognized label is mapped to the linked instruction and executed in the simulator. For example, a user may create an Action gesture (e.g., *Turn on*) and a Device gesture (e.g., *Television*) during authoring. During execution, drawing these gestures enables Action+Object and Object+Action formulations, supporting order-invariant composition.

5.3 System Integration and Data Management

Emeriti interfaces with the recognizer backend (QuantumLeapJS/QuantumLeap) to (1) submit stroke data for recognition and (2) retrieve the recognized gesture label used to trigger the associated instruction. Users can manage their gesture set through two operations: deleting individual gestures and clearing the full dataset (reset). For clarity and brevity, we summarize these interactions here; detailed sequence diagrams for deletion/reset are provided in the supplementary material.

5.4 Example Commands and Permutations

Table 3 lists the vocabulary used in our smart-home instantiation. Rather than enumerating all permutations, we illustrate representative examples of each instruction level below and emphasize that elements can be provided in flexible orders when valid under the grammar. Additional permutations and extended examples are provided in supplementary material.

Table 3. Vocabulary of elements used for the smart-home instantiation.

Action	Target	Environment	Parameter	Macro
Turn on	Television	Kitchen	Volume on NUMBER	MacroLiving
Turn off	Washing machine	Living room	Time in NUMBER	
Increase	Light	Office	Channel NUMBER	
Decrease	Radio	Parent bedroom	Program NUMBER	
Pause	Air Conditioner	Children bedroom	Brightness on NUMBER	
Play	Computer	Bathroom		
Mute	Fan	Dining room		
Unmute	Microwave	Laundry room		
Next				

Representative examples include:

- *Basic instruction:* Turn on the light in the dining room with brightness 5.
- *Common instruction:* Turn off the fan and the lights in 5 minutes.
- *Macro instruction:* MacroLiving (predefined multi-step routine).
- *Composed instruction:* Decrease radio volume to 1 and turn off the air conditioner in the children bedroom.

In all cases, when permitted by the grammar, users may provide components in different orders (*e.g.,* Object+Action vs. Action+Object), enabling order-flexible composition without expanding the base vocabulary.

5.5 Extending to Other Domains: Coffee Shop Use Case

Table 4 lists the vocabulary used in our coffee-shop instantiation. Rather than enumerating all possible permutations, we present a small set of representative examples for each instruction level and highlight that, when permitted by the grammar, command components can be provided in flexible orders. Additional permutations and extended examples are provided in the supplementary material.

Table 4. Example vocabulary for the coffee shop scenario.

Action	Target	Environment	Parameter	Macro
Order	Americano	Kitchen	Spoons	MacroCoffeeOrder
Add	Black coffee	Staff room	Cups	MacroPay
Remove	Espresso	Dining room	Cubes	
Increase	Latte	Indoor seating area	Pumps	
Decrease	Macchiato	Outdoor seating area	Size	
Pay	Mocha		Table	
Bring	Sugar			
Mix				

To illustrate that the same grammar can be instantiated beyond smart homes, we outline a coffee shop ordering scenario (Fig. 12) in which actions, targets, parameters, and macros represent ordering operations (*e.g.,* Order Latte size 2) rather than device control. This scenario is intended as a portability demonstration: the underlying composition mechanism remains unchanged while the vocabulary and scene geometry are swapped.

Fig. 12. A coffee shop use case illustrating portability of the grammar to other domains.

6 Evaluation Studies

We evaluated Emeriti against Ledo et al. [60]'s framework (Figure 13) in which our evaluation spans two complementary strategies: a *Usage*-oriented study (free exploration, qualitative feedback through Product Reaction Cards and discussions, demographic/experience questions, and debriefing) and a *Performance*-oriented controlled comparative study (fixed tasks with objective measures of usability metrics; i.e., effectiveness, efficiency and subjective experience.)

We report two user studies with non-overlapping participant groups. The first study was a formative pilot study (n=9) designed to gather early feedback on Emeriti's interaction concept and interface. Participants freely explored the system and provided feedback through Product Reaction Cards and open-ended comments. This study was exploratory: its purpose was to identify design-relevant usability concerns, such as memorability, repetition burden, and interaction clarity, rather than to provide confirmatory evidence of effectiveness.

Building on these formative observations, we conducted a controlled comparative study with a separate cohort of participants (n=18). Participants completed fifteen fixed tasks, allowing us to compare Emeriti's gesture-composition approach with a sequencing baseline. Following ISO 9241-11, we organize the evaluation around the usability dimensions of effectiveness, efficiency, and satisfaction [52]. Effectiveness refers to the accuracy and completeness with which participants achieved the specified task goals, whereas efficiency refers to the resources required to achieve those goals. We further report subjective outcomes through standardized post-study questionnaires and custom ratings, covering perceived workload, usability, user experience, and multi-step interaction quality.

6.1 Formative Pilot Feedback (n=9)

This subsection reports a formative pilot study conducted to gather early feedback on Emeriti's interaction concept and interface. Unlike the controlled task-based evaluation in subsection 6.2, this pilot was exploratory and aimed to collect design feedback (*e.g.,* clarity, usability concerns, and feature suggestions), rather than to establish effectiveness.

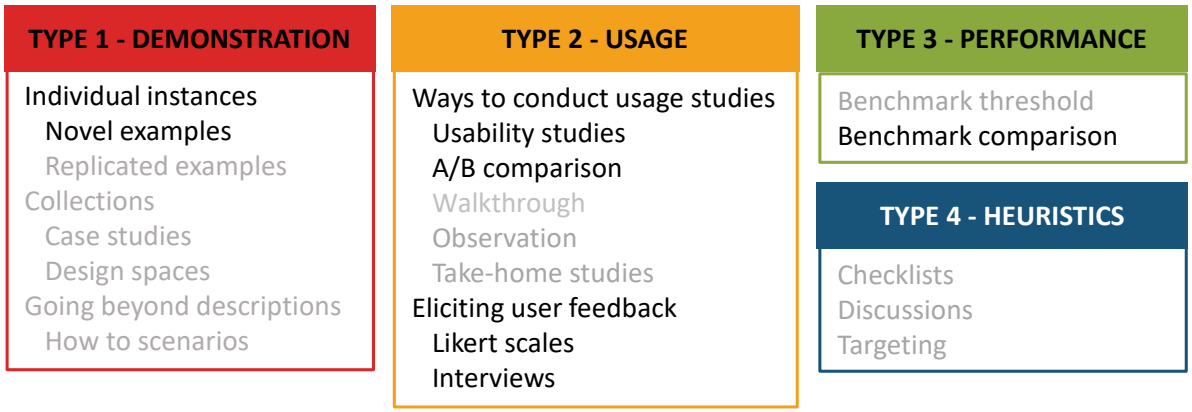

Fig. 13. Adapted from Ledo et al. [60]'s evaluation framework for HCI toolkits. The highlighted elements indicate the parts of the framework covered by our evaluation studies: a *Usage*-oriented formative pilot study (free exploration, elicited user feedback, and discussion-based feedback, including PRC responses) and a *Performance*-oriented controlled comparative study (fixed-task comparison with objective performance measures and questionnaire-based subjective ratings, *e.g.,* workload, usability, and user experience).

The pilot involved 9 participants (5 women, 4 men, M=31.2 years, SD=4.59 years) recruited through internal volunteer mailing lists. None had prior experience with gestural input for compound commands. All reported moderate to extensive use of desktop computers, smartphones, and tablets, and familiarity with common touch gestures (*e.g.,* flick, swipe, pinch-in/out). Participants signed a GDPR-compliant consent form before the study, that has been officially validated by the Ethics Committee in Psychology of our organization. No compensation was offered, which may have introduced self-selection bias.

Participants were introduced to the domain of gestural input for compound commands, including a short video (supplementary material), and then shown Emeriti as a prototype for composing gesture-based commands. After a demonstration, they freely explored the system. No fixed task set was imposed, as the goal was to elicit exploratory feedback on interaction strategies, difficulties, and potential improvements. At the end of the session, we conducted a short question-and-answer discussion and collected questionnaire responses covering demographics, open-ended comments/suggestions, and the Microsoft Product Reaction Cards (PRC) questionnaire [16]. The PRC presents 118 descriptive words with check boxes, and participants may select as many or as few words as they wish. Fig. 14 shows the frequency of the top 38 PRC selections, including *Intuitive*, *Creative*, and *Motivating*. These responses suggest generally positive impressions of the interaction concept, while also highlighting trade-offs and usability concerns. For example, *Time-Saving* and *Time-Consuming* were selected less frequently, indicating mixed perceptions regarding efficiency. Participants also selected words such as *Predictable* and *Contradictory*, suggesting variation in how clearly the system behavior was understood.

Among the critiques, suggestions, and comments, participants highlighted several recurring themes: progressive learning (*e.g.,* P1: "At first, I didn't understand why I had to introduce a gesture for each element each time, rather than one gesture for everything at once. But with practice, I understood how they could be combined to achieve the goal"), order independence of gestures (*e.g.,* P8: "When I was told that the elements could be introduced one by one in any order, I was confused, but in the end I found it flexible"), memorability challenges (*e.g.,* P1: "Even though I've understood that you can combine gestures to obtain more commands, I still have to learn and remember each gesture for each element"), and concerns about transfer to smaller surfaces (*e.g.,* P9: "On the large

Fig. 14. The frequency of top cards chosen by the participants.

touch-sensitive surface, it's fine for drawing several symbols, but I doubt I'll be able to do the same thing on my smartphone, which I use much more than a tablet").

Because this session was exploratory and included an upfront system introduction and demonstration, participant responses may have been influenced by framing and novelty effects. Accordingly, we use these findings as formative design feedback rather than confirmatory evidence. Several themes identified in the pilot (*e.g.,* memorability challenges, repeated articulation, and interaction clarity) also reappeared in post-study comments from the later controlled evaluation, suggesting that these concerns were not unique to the exploratory pilot setting and helping contextualize the controlled study results.

6.2 Controlled Comparative Study (n=18)

Building on the formative pilot feedback, we conducted a controlled comparative study to evaluate Emeriti under fixed tasks and standardized measures. Unlike the pilot, this study assessed usability of Emeriti's gesture-composition approach in comparison with a gesture-sequencing baseline. Following ISO 9241-11, we considered effectiveness, efficiency, and subjective experience [52]. Effectiveness was operationalized through Task Success Rate, which measures whether the final executed command correctly matched the target task, and Goal Completion Ratio, which captures partial completion when participants stopped before finishing a task. Efficiency was operationalized through Task Completion Time. Because the Sequence condition included a 2-second automatic reset after each gesture, we report timeout-corrected task time by subtracting 2 seconds per submitted gesture.

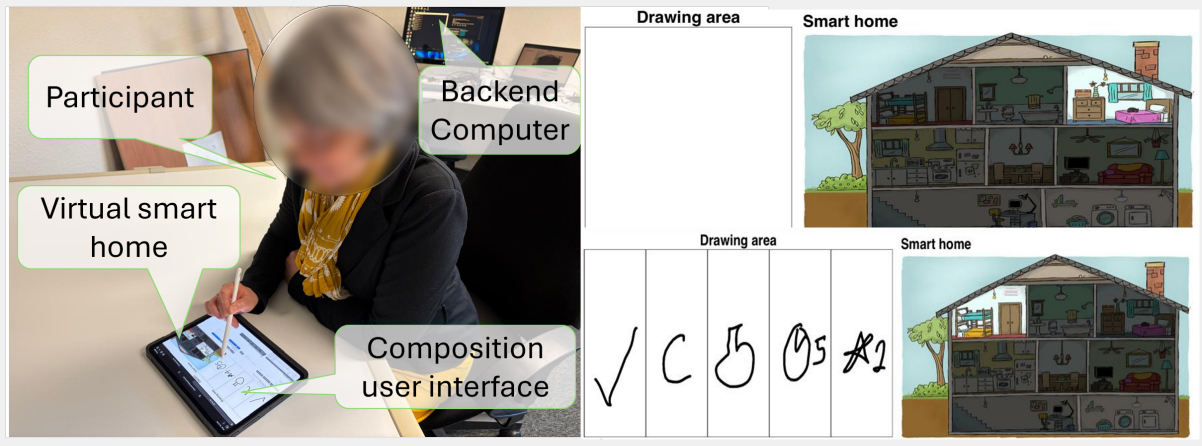

Fig. 15. The experiment setup where a participant interacts with Emeriti on the left, right top shows the interface of the Control condition and right side bottom shows the Composition condition interface.

The sequencing baseline was designed to align with prior work on gesture sequencing in 2D [85] and 3D smart-home interaction [116]. In this baseline, participants issued commands by drawing reusable elementary gestures sequentially on the same canvas. For example, to express "increase TV volume to 5," a participant first selected the device with a TV gesture and then repeatedly performed an "increase volume by 1" gesture until the target value was reached. After completing the sequence, participants pressed the submit button to execute the buffered command.

In the Composition condition, participants used the same elementary command structure but could compose gestures, including parameter values, within a single canvas instance and in any preferred order. This condition leveraged Emeriti's support for congruent, hierarchical, and order-independent gesture composition. After completing the compound gesture, participants pressed the execute button to process the command.

6.3 Participants

Eighteen participants completed the controlled comparative study, with nine participants assigned to each condition. All participants were recruited on a voluntary basis and provided informed consent after an experimenter explained the study procedure and the data collected. The study posed no anticipated risks, all recorded data were anonymized, and participants were informed that they could withdraw at any time without penalty. No compensation was provided. The sample included 10 participants who identified as women and 8 who identified as men. Gender distribution was 4 women and 5 men in the Control condition, and 6 women and 3 men in the Treatment condition. Age was available for 17 participants and ranged from 22 to 63 years (M=31.9, SD=10.4); one participant chose not to report age. Participants came from diverse educational and professional backgrounds, including PhD students (n=3), master students (n=4), undergraduate students (n=2), a researcher (n=1), a teaching assistant (n=1), administrative staff (n=1), a secretary (n=1), an accountant (n=1), a chef (n=1), a software engineer (n=1), an aircraft engineer (n=1), and a professor (n=1). Participants also reported diverse native-language backgrounds: Arabic (n=1), Dutch (n=1), French (n=9), German (n=1), Hindi (n=1), Romanian (n=1), and Sinhala (n=4). These languages span multiple canonical word-order typologies, including SOV (Hindi and Sinhala; n=5), SVO (French and Romanian; n=10), VSO (Arabic; n=1) and verb-second (V2) languages with mixed surface orders across clause types (Dutch and German; n=2). All participants reported regular computer use, but none had prior experience with sequence or compositional gestural inputs for smart-home environments.

6.4 Study Design

We employed a between-subjects design with two conditions. A within-subjects design would have introduced carry-over effects that are difficult to control here: the two conditions require qualitatively different mental models and use partially non-overlapping gesture vocabularies (Figure 16), so exposing the same participant to both would risk contaminating the sequential baseline with composition strategies and inflating memory load. The design isolates the role of parameter encoding (*e.g.,* brightness, volume, time) as the sole manipulated variable. Participants were randomly assigned to one group to evaluate how gesture composition influences task performance, specifically isolating the role of *parameter encoding* (*e.g.,* brightness, volume, time).

6.4.1 Study Conditions.

- **Sequence Condition (Control):** Participants issued commands as a *sequence of discrete gesture events* on a single canvas instance. Each iteration contained one elementary gesture and the canvas automatically reset between iterations (after a short pause), enabling participants to build up a command step-by-step. Participants specified the base command by successively drawing gestures one-after-other for a walk-through example, for the task "Turn on Kitchen Light and Set Brightness to 5", the participant starts by drawing action (*e.g.,* a *check mark* for “Turn On”), wait until two seconds timer goes off, then draws environment (e.g., a *house* for “Kitchen”), wait until the timer goes off, and draws device (e.g., a *bulb* for “Light”). Parameter values were encoded through *incremental repetition*: for instance, expressing “brightness to 5” required iterating an “increase brightness by 1” gesture five consecutive times, with the canvas resetting after each repetition. After building the whole task by sequentially combining the elementary gestures one-after-other, the participant submits and Emeriti identify and execute the action. In the Sequence condition, participants could draw gesture elements in any order permitted by the grammar; the canvas reset after each submitted gesture. Unlike some sequential baselines [115], we did not require sequences to start with a Device. This kept ordering flexible in both conditions, so any performance differences primarily reflect parameter encoding (incremental repetition vs. direct composition) rather than ordering constraints.
- **Composition Condition (Treatment):** Participants accessed the same elementary gestures but could compose them—including parameter values—into a *single compound gesture* on one canvas iteration and submit all-together in the end. Instead of repeating incremental gestures, parameter values were specified directly as part of the compound command. For example, “brightness to 5” was encoded by composing the brightness parameter with its target value in one submission, eliminating repetition.

Both conditions support the same *base-command structure* (action + environment + device). They differ in how *parameters* are expressed: sequential incremental repetition in the Sequence condition versus direct parameter-value composition in the Composition condition. This manipulation isolates the effect of parameter composition while keeping the remainder of the interaction comparable across conditions.

6.4.2 Tasks. The study comprised fifteen tasks, identical across conditions, grouped into three categories of five tasks each (Table 5). Across categories, the required end state of each task is generally reachable in both conditions; however, for parameterized tasks the *interaction cost* increases substantially in the Sequence condition because target values must be reached through repeated incremental submissions. The three categories form a gradient of parameter-encoding burden for the Sequence condition:

- **Straightforward** tasks require no parameter gestures (e.g., “Turn on the AC”). These tasks serve as a baseline, confirming that allowing parameter composition does not degrade performance on simple commands.
- **Composition-optional** tasks involve moderate parameter values (e.g., volume 5, brightness 5). In the Sequence condition, users can reach the target with a feasible number of repetitions ($\leq$5). These tasks test whether direct parameter composition provides benefits even when sequential repetition remains practical.

- **Composition-beneficial** tasks involve larger and/or multiple parameter values (e.g., brightness 8 and time 5 in one command, or temperature 22). In the Sequence condition, these tasks require long repetition sequences, increasing time and effort and raising the likelihood of stopping early. These tasks test whether direct parameter composition enables more practical completion under higher parameter demands.

The phrasing of the task descriptions reflects this emphasis: "*set* the volume to 5" highlights an explicit parameter-setting step, whereas "*with* brightness 5" emphasizes that the parameter is intrinsic to the desired command state.

Table 5. Task categories and what each requires from the Sequence condition. In the Composition condition, parameterized commands can be expressed by composing parameter values directly, avoiding incremental repetition. In the Sequence condition, the parameter-encoding burden increases across categories.

Category	**Example task**	**Sequence condition requires**	**Parameter repetitions**
Straightforward (5 tasks)	"Turn on bedroom light"	Base command only; no parameter gestures needed.	0
Composition-optional (5 tasks)	"Turn on the radio and *set* the volume to 5"	Base command, then repeat "increase volume by 1" until reaching the target value.	3–5
Composition-beneficial (5 tasks)	"Turn on kitchen light *with* brightness 8 in 5 minutes"	Base command, then repeat brightness-increment 8 times *and* time-increment 5 times.	5–30

6.5 Gesture Set

To ensure a fair and robust evaluation, we selected elementary gestures that were as congruent as possible across conditions. Wherever feasible, the same gestures were used in both study conditions. For task components that could not be expressed in the Sequence condition using the same direct parameter-value composition strategy as the Composition condition, we introduced alternative gestures designed to preserve semantic congruency while maintaining comparable visual complexity (*e.g.,* similar stroke count and corner structure). Table 6 summarizes which gestures were available in each condition, and Fig. 16 illustrates the gesture forms.

For readability, Fig. 16 shows one representative example for paired directional parameter gestures in the Sequence condition (*e.g.,* increase/decrease brightness and next/previous channel). In these cases, the illustrated gesture depicts one direction (*e.g.,* an upward arrow for *increase*); the opposite command uses the same base form with the direction reversed (*e.g.,* a downward arrow for *decrease*).

6.6 Procedure

The study procedure is described in three parts: (1) a common flow shared by both conditions, and (2–3) condition-specific task execution flows.

6.6.1 Common User Flow. One experimenter welcomed the participants in a separate room, briefed them on the study purpose, risks, and data collection, and obtained informed consent. Participants then completed a demographics questionnaire that was formally approved by the Ethics Committee in Psychology of our organization. The experimenter escorted the participant to the lab, where a second experimenter ran the session. After seating the participant, the experimenter explained the setup and condition-specific interaction technique,

Table 6. Gesture inventory used in the study, grouped by availability and semantic role. Visual examples are shown in Fig. 16.

Avail.	Type	Gestures / command elements	See Fig.
Shared in both conditions			
Both	Action	Turn on; Turn off	(a)
Both	Device	Light; Television; Radio; Air conditioner; Computer	(a)
Both	Environment	Kitchen; Children's bedroom; Parents' bedroom; Living room	(a)
Sequence condition (Control) only			
Control	Parameter	Increase/decrease brightness by 1; increase/decrease volume by 1; next/previous channel by 1; increase/decrease time by 1 min; next/previous program by 1	(b)
Composition condition (Treatment) only			
Treatment	Parameter	Set time; set volume; set brightness; set channel; set program	(c)
Treatment	Macro	Numeric value gestures used as parameter values in composition (*e.g.,* digits used in task parameters)	(c)

introduced the available stroke gestures and their meanings, and demonstrated the system behavior. Participants completed one practice task before starting the evaluation. In both conditions, participants could consult the gesture reference sheet at any time.

6.6.2 Control Condition User Flow (Sequence). In the control condition, participants completed each task by drawing the required gestures sequentially. The canvas reset after each gesture. After drawing the full sequence, participants pressed the *recognize* button to submit and execute the intent.

6.6.3 Treatment Condition User Flow (Composition). In the treatment condition, participants drew gestures as a compound on a single canvas instance (rather than sequentially with resets). After finishing the compound gesture, participants pressed the *recognize* button to submit and execute the intent. As in the control condition, execution occurred only after submission.

6.7 Data Collection and Analysis

Emeriti logged all task interactions, including the drawn stroke gestures, task identifiers, user identifiers, condition assignment, timestamps, and retry counts. Participants were free to stop at any point during the evaluation. After completing the tasks, participants completed post-study questionnaires covering workload, usability, and user experience: the NASA-TLX [45, 46] (six subscales: Mental, Physical, Temporal, Performance, Effort, and Frustration), the System Usability Scale [14] (SUS; 10 items rated on a 5-point Likert scale [66], where responses range from 1 to 5 and the minimum/maximum scale values are 1 and 5), and the UEQ-S [91] (8 semantic differential items collected on a 5-point response scale and centered for analysis, with raw responses ranging from 1 to 5 before centering). This questionnaire is more up-to-date than IBM CSUQ [62, 63]. In addition, we collected four custom Likert items on a 1–10 scale (minimum = 1, maximum = 10): participants rated whether the gestures allowed them to effectively complete simple tasks (*e.g.,* single actions such as turning on a light), whether the gestures enabled efficient execution of complex tasks (e.g., multi-device or timed commands), whether the gesture system expanded the range of commands without feeling overwhelming, and whether the gesture approach felt natural and intuitive in tasks requiring multiple steps.

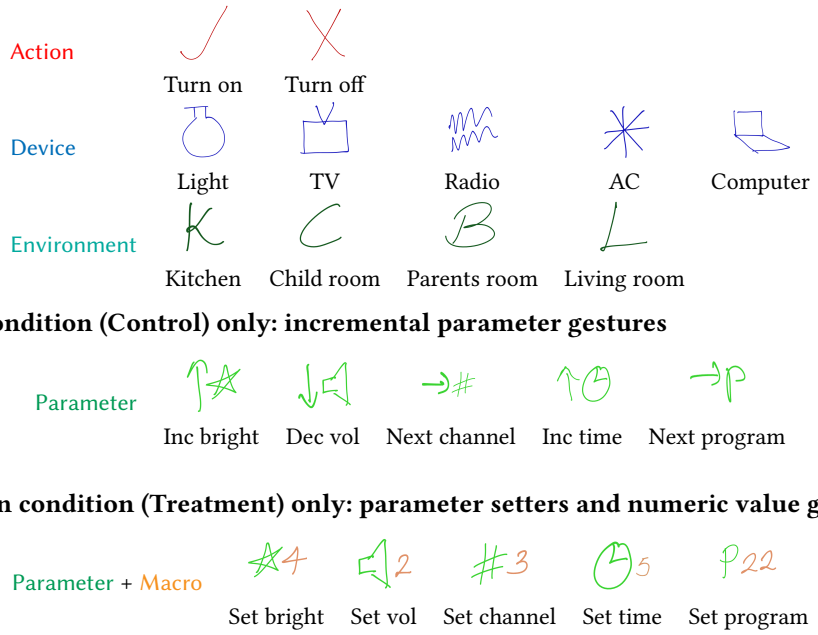

Fig. 16. Gesture set used in the study. (a) Shared gestures available in both conditions, including Action, Device, and Environment gestures. (b) Control-only Parameter gestures for incremental adjustment, repeated to reach target values. The increase brightness gesture is displayed; the decrease brightness gesture is the same but with a downward arrow. For the other gestures, the corresponding opposite gestures follow the same pattern: increase volume uses an upward arrow, and previous channel uses a backward arrow instead of the next channel arrow. (c) Treatment-only parameter setter gestures (Parameter) and numeric value gestures (Macro) used for direct parameter-value composition.

6.7.1 Effectiveness and Efficiency Metrics. We evaluated task performance using two effectiveness metrics and one efficiency metric. First, Task Success Rate measures the proportion of tasks for which the final executed command fully matched the target command. A task was scored as successful only when the participant submitted the required command components without missing, extra, overshot, or undershot components. Second, Goal Completion Ratio captures partial completion. Because participants were allowed to stop before completing a task, this metric measures how much of the required command was completed:

$$\text{Goal Completion Ratio}_u = \frac{\sum_t \min(G_{u,t}^{exec}, G_t^{req})}{\sum_t G_t^{req}},$$

where $G_{u,t}^{exec}$ is the number of correctly executed gesture submissions by participant u for task t, and G_t^{req} is the number of required submissions for that task. The ratio is capped at 1.0, so overshooting does not increase the score. Third, Task Completion Time measures efficiency. For each participant, we used the median task-completion time across the 15 evaluation tasks. Because the Control condition included a 2-second automatic reset after each submitted gesture, we report timeout-corrected task time by subtracting 2 seconds per submitted gesture in the Control condition. No timeout correction was applied to the Treatment condition.

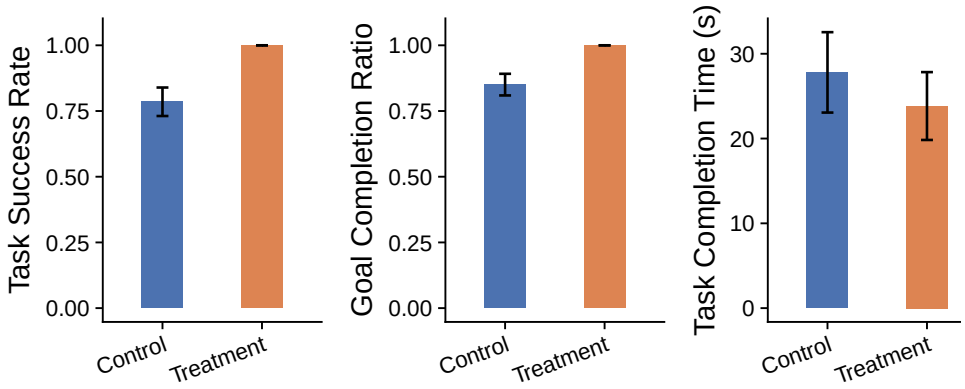

Fig. 17. Participant-level Task Success Rate, Goal Completion Ratio, and timeout-corrected Task Completion Time by condition. Bars show means and error bars indicate standard errors. Higher values are better for Task Success Rate and Goal Completion Ratio; lower values are better for Task Completion Time.

6.7.2 Statistical Analysis. All task-performance metrics were analyzed at the participant level, rather than at the task-row level, because each participant completed multiple tasks. For each participant, we computed Task Success Rate, Goal Completion Ratio, and median timeout-corrected Task Completion Time across the 15 evaluation tasks. Given the small sample size, bounded effectiveness measures, ties, and ceiling effects in the Treatment condition, we used two-sided permutation Mann–Whitney–Wilcoxon tests to compare the Control and Treatment conditions. We report the Mann–Whitney U statistic, Holm-corrected p-values across the three planned task-performance comparisons, and Cliff's δ as a nonparametric effect size. Positive Cliff's δ values indicate higher values in the Treatment condition, whereas negative values indicate lower values in the Treatment condition. For Task Completion Time, lower values indicate better performance.

6.7.3 Task-Specific Subjective Questions. In addition to standardized questionnaires, we collected four task-specific Likert items on a 1–10 response scale (minimum = 1, maximum = 10) to capture perceived effectiveness and interaction quality for different task complexities. Participants rated: (i) whether the gesture set allowed them to effectively complete *simple tasks* (*e.g.,* single actions such as turning on a light), (ii) whether the gestures enabled *efficient execution of complex tasks* (e.g., multi-device commands or commands with time- or value-based parameters), (iii) whether gesture composition *expanded the expressible command range without feeling overwhelming*, and (iv) whether the gesture approach felt *natural and intuitive* when tasks required multiple steps or parameter specification.

6.8 Results

Figure 17 summarizes the participant-level task-performance results for the Control and Treatment conditions.

6.8.1 Task Performance. Figure 17 shows the participant-level Task Success Rate, Goal Completion Ratio, and timeout-corrected Task Completion Time for the Control and Treatment conditions. All reported p-values are Holm-corrected across the three planned task-performance comparisons.

For effectiveness, the Treatment condition achieved a higher Task Success Rate than the Control condition. Participants in the Treatment condition completed all tasks successfully (M=1.000, SE=0.000), whereas participants in the Control condition showed lower task success (M=0.785, SE=0.054). This difference was statistically significant, U=76.5, p_{Holm}=0.001**, with a large effect size, Cliff's δ=0.889.

The same pattern was observed for Goal Completion Ratio. Participants in the Treatment condition reached full goal completion (M=1.000, SE=0.000), whereas participants in the Control condition completed a smaller proportion of the required command components (M=0.850, SE=0.041). This difference was also statistically significant, U=76.5, p_{Holm}=0.001**, with a large effect size, Cliff's δ=0.889.

For efficiency, timeout-corrected Task Completion Time was numerically lower in the Treatment condition (M=23.84 s, SE=4.00) than in the Control condition (M=27.80 s, SE=4.74). However, this difference was not statistically significant after Holm correction, U=31.0, p_{Holm}=0.436, Cliff's δ= − 0.235. Thus, the expanded dataset provides strong evidence that gesture composition improved task success and goal completion, while the observed time difference should be interpreted descriptively.

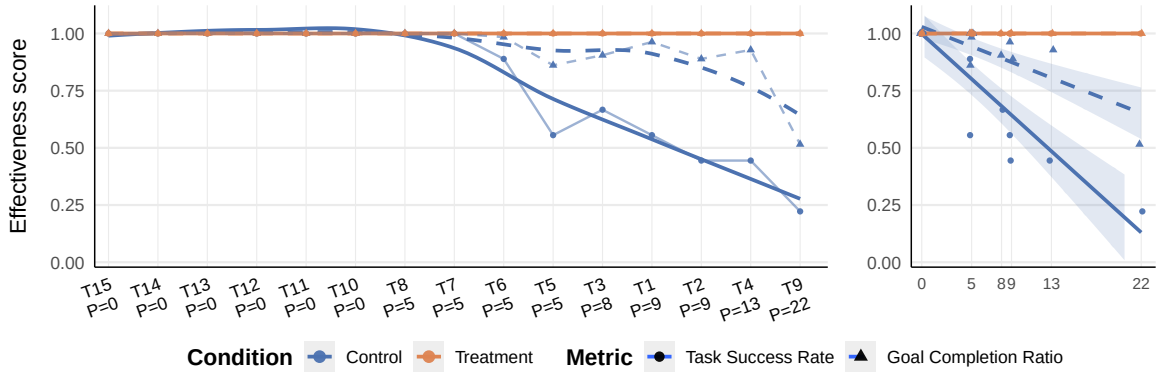

Fig. 18. Effectiveness scores for the Control and Treatment conditions, measured using Task Success Rate and Goal Completion Ratio. The left plot shows task-level effectiveness, with tasks ordered by required number of parameters. The right plot shows the relationship between the required number of parameters and effectiveness, with fitted trend lines. Overall, effectiveness remains stable in the Treatment condition, whereas the Control condition decreases as the required number of parameters increases.

As shown in Table 5, the fifteen tasks were designed to increase in expressivity demand. This progression allows us to examine whether task effectiveness changes differently across the Control and Treatment conditions as tasks become more complex. To analyze this relationship, we plotted Task Success Rate and Goal Completion Ratio for both conditions across the ordered task set. Figure 18 illustrates how effectiveness varies as the required expressivity of the task increases.

For tasks T15, T14, T13, T12, T11, and T10, expressivity demand is low, because these tasks correspond to simple base commands, such as *e.g.,* "Turn off Light", and require no parameter repetitions in the Control condition. Accordingly, both conditions show similarly high effectiveness for these tasks. As expressivity demand increases, however, the Control condition shows a marked decline in both Task Success Rate and Goal Completion Ratio. This trend suggests that when additional expressivity must be achieved through sequential repetition, participants become less likely to complete tasks accurately and completely.

In contrast, the Treatment condition remains stable across the task sequence, maintaining high effectiveness even for tasks with greater expressivity demands. This result highlights the advantage of Emeriti: direct gesture composition supports more expressive commands without imposing the repeated incremental input required in the Control condition.

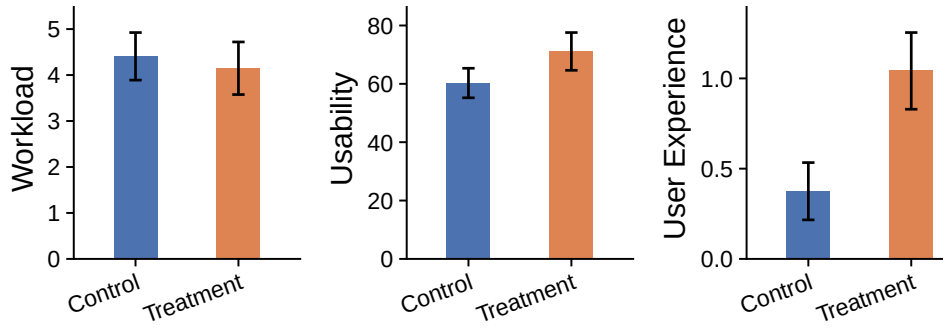

Fig. 19. Comparison of subjective measures between the Control and Treatment conditions. From left to right, the plots report perceived workload using NASA-TLX, usability using SUS, and user experience using UEQ-S. Lower NASA-TLX scores indicate better performance because they reflect lower workload, whereas higher SUS and UEQ-S scores indicate better usability and user experience. Error bars show standard error.

6.8.2 Subjective Measures. Figure 19 summarizes participant-level subjective outcomes for the Control and Treatment conditions, including perceived workload (Workload; NASA-TLX, lower is better), perceived usability (Usability; SUS, higher is better), and overall user experience (User Experience; UEQ-S overall, higher is better). All reported p-values are Holm-corrected across the three subjective comparisons.

For workload, ratings were comparable across conditions. The Treatment condition showed slightly lower workload (M=4.148, SE=0.572) than the Control condition (M=4.407, SE=0.518), but this difference was not statistically significant after Holm correction (U=30.0, p_{Holm}=0.371, Cliff's δ= − 0.259 [−0.753, 0.284]).

For usability, SUS scores were descriptively higher in the Treatment condition (M=71.111, SE=6.470) than in the Control condition (M=60.278, SE=5.059); however, this difference was not statistically significant after Holm correction (U=60.0, p_{Holm}=0.174, Cliff's δ=0.481 [−0.062, 0.951]). The same directional pattern was observed for user experience. Participants in the Treatment condition reported higher UEQ-S overall scores (M=1.042, SE=0.212) than those in the Control condition (M=0.375, SE=0.159). This difference did not reach significance after Holm correction (U = 63.5, p_{Holm}=0.126, Cliff's δ=0.568 [0.086, 0.914]).

Overall, subjective ratings did not differ significantly between conditions after correcting for multiple comparisons, but the direction of effects consistently favored the Treatment condition for usability and user experience without increasing perceived workload.

6.8.3 Task-Specific Subjective Ratings. We also included four task-specific subjective questions on a 1–10 scale (subsubsection 6.7.3) to capture participants' perceptions of the gesture approach beyond standardized questionnaires. Overall, we did not observe statistically significant differences between conditions for these items; however, the response patterns showed consistent trends favoring the Treatment (Composition) condition for multi-step and parameterized tasks. Participants in both conditions reported high agreement that the gesture set supported *simple tasks* (*e.g.,* single actions such as turning a light on), suggesting that either interaction style was perceived as adequate for straightforward commands.

For *complex tasks* (*e.g.,* multi-device or timed commands), participants tended to rate the Treatment condition higher, indicating that composition was perceived as more efficient when tasks required multiple steps or parameter specification. A similar trend was observed for perceived *command-range expansion without feeling overwhelming*: participants generally indicated that composition expanded what they could express while remaining manageable.

Finally, for *naturalness and intuitiveness in multi-step tasks*, responses again tended to favor the Treatment condition, suggesting that composing gestures on a single canvas was perceived as a more natural interaction strategy than drawing a reset sequence.

7 Discussion and Limitations

7.1 Summary of Findings Relative to Research Questions

The technical demonstration and the controlled study provide convergent evidence for the three research questions in Table 1.

RQ_1—Expanding expressivity without proportionally expanding vocabulary. The results provide positive evidence that Emeriti can expand the expressivity of gesture input without requiring a proportional increase in elementary gesture vocabulary. At the technical level, the EBNF grammar represents commands as reusable elements—Action, Device, Parameter, Environment, and Macro—and the ANTLR-based parser accepts multiple valid permutations of these elements. This allows a limited set of elementary gestures to be recombined into a larger command space, rather than requiring a separate gesture for every complete command or parameterized command variant.

In the controlled study, this advantage was most visible for parameterized tasks. The two conditions shared the same Action, Device, and Environment gestures, but differed in how parameters were entered. In the Sequence condition, participants used incremental parameter gestures, which had to be repeated until the target value was reached. In the Composition condition, participants used parameter setters and numeric value gestures, allowing the target parameter value to be expressed directly within a compound gesture. Thus, the study evaluates a specific and practically important form of expressivity expansion: direct parameter-value composition versus sequential incremental entry.

Figure 18 shows that both conditions achieved similarly high effectiveness on low-expressivity tasks that required no parameter repetitions. However, as the required parameter demand increased, effectiveness in the Sequence condition decreased, whereas effectiveness in the Composition condition remained stable. At the participant level, the Composition condition achieved perfect Task Success Rate and Goal Completion Ratio, while the Sequence condition showed lower and more variable performance. These differences were statistically significant for both Task Success Rate and Goal Completion Ratio, indicating that composition improved not only whether participants completed the intended command, but also how completely they progressed through parameter-heavy tasks.

The efficiency evidence should be interpreted more cautiously. Timeout-corrected Task Completion Time was numerically lower in the Composition condition than in the Sequence condition, but this difference was not statistically significant after correction for multiple comparisons. Therefore, RQ_1 is supported more strongly by the command-space analysis and by the effectiveness and goal-completion results than by completion-time results. One plausible interpretation is that some participants in the Sequence condition stopped before completing long repetition sequences; this reduced their completion scores, while also limiting the amount of time that would otherwise have been spent on repeated incremental input.

The subjective results are consistent with this interpretation, although they should be treated as preliminary. Standardized workload, usability, and user-experience measures did not show statistically significant between-condition differences. Nevertheless, workload was slightly lower in the Composition condition, and task-specific ratings tended to favor composition for complex and parameterized tasks, as well as for expanding the range of expressible commands without feeling overwhelming. Taken together, the evidence suggests that Emeriti answers RQ_1 within the scope of the evaluated smart-home tasks: gesture composition can preserve a compact elementary vocabulary while supporting more expressive, parameterized commands and maintaining high task effectiveness. However, this result should not be generalized to all possible sequential baselines or to long-term vocabulary learning, which remain subjects for future work.

RQ_2—Interpreting variable-order compositions into intended commands. RQ_2 concerns whether variable-order gesture compositions can be interpreted as the intended task-level commands. At the technical level, the EBNF

grammar and ANTLR-based parser support this by accepting multiple valid permutations of command components, such as Action, Device, Parameter, Environment, and value elements. Thus, users are not forced to follow a single fixed ordering when composing a command.

Empirically, the Composition condition provides end-to-end evidence that this interpretation process worked for the evaluated smart-home tasks. Participants achieved perfect Task Success Rate and Goal Completion Ratio, indicating that their compound gestures were not only syntactically accepted but also mapped to the intended commands. In contrast, the Sequence condition showed lower and more variable completion, especially for parameterized tasks. This comparison suggests that direct composition helped participants express complete commands more reliably than sequential incremental entry.

As we have participants from many languages and cultural differences, we observed that they executed the gestures in different orders, as their language influence. Many times Sinhala, Hindi speaking participants executed the gestures in order Environment, Device and then action, even though the tasks were displayed in English.

RQ_3—Impact on subjective workload, usability, and user experience. Post-study questionnaires (NASA-TLX, SUS, UEQ-S) did not show statistically significant between-condition differences after Holm correction. Nevertheless, the direction of effects generally favored composition for usability and user experience, without increasing perceived workload. Given the sample size, these subjective trends should be treated as preliminary.

A further consideration is that task completion was voluntary: participants could stop at any time and were not required to complete every task. Several participants chose to stop gesture execution early in the Control (Sequencing) condition. This reduced their exposure to the full repetition burden of sequential parameter entry and may have attenuated between-condition differences in perceived workload and usability. Accordingly, the subjective ratings should be interpreted in light of this differential exposure across conditions.

The open-ended responses provide additional context for these trends, particularly around parameter specification. In the Control (Sequencing) condition, multiple participants explicitly described the burden of repeating the same gesture to reach a target value and suggested mechanisms for embedding or compressing parameter magnitude (*e.g.,* a multiplier). Representative comments included: “I need to repeat the same gesture many times” and “I wanted to use symbols to multiply the number of volume increases.” Others similarly emphasized that setting larger values felt tedious or awkward due to repetition. In contrast, participants in the Treatment (Composition) condition emphasized that composing parameter values directly enabled more precise expression of intent (*e.g.,* “Allow to more precise in the action that needed to be done”).

Taken together, these qualitative responses suggest that composition reduces a key friction point of sequential approaches for parameter-heavy commands: the repetitive effort required to encode magnitude. This aligns with the objective task-performance results, and helps explain why composition was perceived as better suited for multi-step and parameterized tasks even when standardized subjective measures did not reach statistical significance after correction.

7.2 Technical Feasibility of Gesture Composition

Prior work suggests that gesture composition can improve usability and memorability [38, 88, 116], but often focuses on limited compositions (e.g., Action+Device). Extending composition to parameters and environments requires reliable disambiguation and a command representation that can be parsed under flexible ordering. Our contribution primarily addresses this feasibility: elementary gestures can be composed into compound gestures and mapped to Basic, Common, Macro, and Composed instructions, including order-interchanged compositions. However, technical feasibility does not establish human feasibility at scale; longitudinal studies are needed to assess learnability, memorability, and sustained use as the command space grows.

7.3 Limitations on Application

Our evaluation is scoped to a simulated smart-home environment. While this enabled controlled testing across multi-step and parameterized tasks, it does not fully capture real-world conditions (*e.g.,* environmental variability, device heterogeneity, interruptions, and long-term use). In addition, the controlled comparison contrasts two complete interaction designs. Although both conditions preserve order flexibility, they differ in parameter-value entry and interface flow (including per-gesture resets in the Sequence condition). Future work should replicate these findings in ecologically valid deployments, explore alternative baselines for sequential parameter entry, and extend evaluation to additional domains (including 3D use cases such as the coffee-shop scenario).

Also our current Emeriti prototype does not implement fully online segmentation (*i.e.,* automatically detecting gesture boundaries while the user is still drawing). Instead, gestures are recognized only after the participant explicitly submits the stroke input. As a consequence, in the Sequence condition we introduced a 2 s delay between consecutive gesture submissions to ensure reliable boundary separation for multi-stroke gestures and to prevent unintended merging of strokes across successive elements. This interface-imposed waiting time may have increased interaction cost in the Sequence condition beyond what a production-ready sequential system would require. In the Composition condition, the prototype similarly relies on explicit spatial separation: participants drew each gesture element within a dedicated square region so that the recognizer could classify each region independently. This constrained drawing space may have influenced input comfort and could affect generalizability to unconstrained free-form composition. Overall, these limitations reflect the prototype's reliance on manual temporal and spatial segmentation rather than robust automatic segmentation. Future work should integrate online gesture segmentation and region-free element separation to support both sequencing and composition without imposed delays or fixed drawing regions.

7.4 Limitations on the Recognizers

Emeriti currently integrates template-based recognizers for 2D and 3D gesture input. This does not constitute exhaustive coverage of recognizer families, and comparing recognizers across diverse datasets remains methodologically and computationally demanding. Future work could integrate and compare alternative recognizers from pattern-matching and learning-based families to improve robustness, especially under user-defined vocabularies and limited examples.

7.5 Limitations of the Dataset

In our prototype setup, gesture classes were initialized using experimenter-provided templates, and participants did not contribute their own training samples. This supports reproducibility but may not match how end users naturally articulate gestures in deployment. Future work should evaluate user-defined vocabularies, personalization (e.g., few-shot template capture), and adaptation strategies to better support variability in gesture production.

8 Conclusion and Future Work

Increasing the number of elementary gestures, each associated with a single command in a vocabulary, is not a tenable approach for end users who prefer to interact with only a limited number of gestures. Because users cannot easily retain and use a very large set of elementary gestures, composing a smaller set of elementary gestures into compound gestures offers a practical way to support both simple tasks and more complex tasks without proportionally expanding the gesture vocabulary.

To this end, we rigorously defined a set of new concepts, *i.e.,* a basic instruction, a common instruction, a macro instruction, a composed instruction, an element, a composed element, to create an elementary gesture and to compose them into a compound gesture. To express such gestures corresponding to such instructions, we created an EBNF grammar enabling the parsing of instructions in any order. Beyond the number of gestures, the

number of associated commands, and the number of objects, our grammar also introduced the notion of device, environment, parameter with values, which is not achieved by other approaches.

We validated the grammar with an ANTLR-based parser generator and implemented these concepts in Emeriti, a framework for recognizing elementary and compound gestures in both 2D (uni- and multistroke gestures on an interactive surface) and 3D (point trajectories in space). In our smart-home application, this enables compound commands with parameters to be issued through gesture composition, substantially increasing the range of executable commands without requiring a large increase in memorized gesture primitives.

Importantly, the user study provides initial empirical evidence that this design choice yields practical benefits. In a between-subjects task-based evaluation, the Composition condition achieved significantly higher participant-level task effectiveness (strict completion) and willingness (coverage) than the Sequence condition, while also yielding significantly faster timeout-corrected task completion times. All participants in the Composition condition reached perfect aggregated task effectiveness and willingness scores, whereas the Sequence condition showed lower and more variable performance. These results suggest that direct parameter-value composition improves not only whether participants complete tasks, but also how willing they are to persist on parameter-heavy tasks that would otherwise require repetitive sequential input.

Subjective measures (NASA-TLX, SUS, UEQ-S, and custom items) did not show statistically significant between-condition differences in this sample, although the effect directions generally favored the Composition condition, particularly for pragmatic user experience and perceived efficiency on complex tasks. We therefore interpret the subjective findings as encouraging but preliminary, and we explicitly acknowledge the limited statistical power of the current sample size.

Regarding future work, we plan to extend Emeriti with a multimodal component that accepts the same compound command across modalities: natural language (recognized by a Speech-to-Text module and transformed into a grammar-compliant command), graphical user interfaces (with dynamically generated widgets for parameters), and gesture interaction as demonstrated in this paper. This multimodal alignment would preserve a common command representation while improving discoverability and correction support through shared grammar-based structures.

We also plan to further evaluate Emeriti in larger and more ecologically realistic deployments, with broader task sets and stronger longitudinal evidence on memorability, learnability, and sustained use. In parallel, future work should continue improving recognizer robustness and dataset support, especially for dynamically created gesture vocabularies with limited per-user templates.

Open Science. We share a Emeriti [1] repository with the software developed for this paper, all recognizers written in JavaScript (tested under Node.js v18 LTS; requires Node.js ≥ 18) and associated datasets to foster reproducible research around gesture recognition and its incorporation into mainstream interactive applications. Since our Emeriti is publicly available, it can be reused to perform other use cases for reproducibility. Beyond that, it can serve as a metric benchmarking based on measures to compare various vocabularies and their composition.

Acknowledgments

The authors would like to thank the EICS '26 reviewers and coordinator for their constructive feedback on earlier versions of this manuscript. Nuwan Attygalle and Jean Vanderdonckt are supported by the EU EIC Pathfinder-Awareness Inside challenge Symbiotik project (October 1st, 2022-September 30th, 2026) under Grant no. 101071147. All authors acknowledge the support of the project "Cross-Cultural Gesture in Prosocial Virtual Reality Systems" (Respire: May 1st, 2025 - January 31st, 2027) funded by the MIT International Science and Technology Initiative (MISTI) between MIT and Belgium, Global Seed Funds (GSF) program, under Grant no. P093800.

[1] https://anonymous.4open.science/r/EMERITI-CE4C

References

[1] Roland Aigner, Daniel Wigdor, Hrvoje Benko, Michael Haller, David Lindbauer, Alexandra Ion, Shengdong Zhao, and Jeffrey Tzu Kwan Valino Koh. 2012. *Understanding Mid-Air Hand Gestures: A Study of Human Preferences in Usage of Gesture Types for HCI.* Technical Report MSR-TR-2012-111. Microsoft Research. 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/

[2] Ahsan Jamal Akbar, Zhiyao Sheng, Qian Zhang, and Dong Wang. 2023. Cross-Domain Gesture Sequence Recognition for Two-Player Exergames using COTS mmWave Radar. *Proc. ACM Hum.-Comput. Interact.* 7, ISS, Article 441 (Nov. 2023), 30 pages. https://doi.org/10.1145/3626477

[3] Lisa Anthony and Jacob O. Wobbrock. 2010. A Lightweight Multistroke Recognizer for User Interface Prototypes. In *Proc. of Graphics Interface 2010* (Ottawa, Ontario, Canada) *(GI '10)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 245–252. http://dl.acm.org/citation.cfm?id=1839214.1839258

[4] Lisa Anthony and Jacob O. Wobbrock. 2012. $N-Protractor: A Fast and Accurate Multistroke Recognizer. In *Proc. of Graphics Interface 2012* (Toronto, Ontario, Canada) *(GI '12)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 117–120. http://dl.acm.org/citation.cfm?id=2305276.2305296

[5] Caroline Appert and Shumin Zhai. 2009. Using strokes as command shortcuts: cognitive benefits and toolkit support. In *Proc. of the 27th Int. Conf. on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*, Dan R. Olsen Jr., Richard B. Arthur, Ken Hinckley, Meredith Ringel Morris, Scott E. Hudson, and Saul Greenberg (Eds.). Association for Computing Machinery, New York, NY, USA, 2289–2298. https://doi.org/10.1145/1518701.1519052

[6] Nathalie Aquino, Jean Vanderdonckt, and Oscar Pastor. 2010. Transformation templates: adding flexibility to model-driven engineering of user interfaces. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (Sierre, Switzerland) *(SAC '10)*. Association for Computing Machinery, New York, NY, USA, 1195–1202. https://doi.org/10.1145/1774088.1774340

[7] Nuwan T. Attygalle, Matjaž Kljun, and Klen Čopič Pucihar. 2026. *Digital Signal Processing Tools for Radar-Based Human-Computer Interaction.* Springer Nature Switzerland, Cham, 31–46. https://doi.org/10.1007/978-3-032-04327-6_2

[8] Nuwan T. Attygalle, Matjaž Kljun, Arthur Sluÿters, and Klen Čopič Pucihar. 2026. *Radar-Based Human-Computer Interaction When Sensing Through Materials.* Springer Nature Switzerland, Cham, 65–84. https://doi.org/10.1007/978-3-032-04327-6_4

[9] Nuwan T Attygalle, Matjaž Kljun, and Klen Čopič Pucihar. 2025. Gesture Recognition on Deformable Objects Using Millimeter-Wave Radar. In *Companion Proceedings of the 17th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '25 Companion)*. Association for Computing Machinery, New York, NY, USA, 50–58. https://doi.org/10.1145/3731406.3734979

[10] Nuwan T. Attygalle, Matjaž Kljun, Una Vuletić, and Klen Čopič Pucihar. 2026. *Comparative Testing of Radar Signal Representations When Sensing Through Materials.* Springer Nature Switzerland, Cham, 47–61. https://doi.org/10.1007/978-3-032-04327-6_3

[11] Nuwan T. Attygalle, Luis A. Leiva, Matjaž Kljun, Christian Sandor, Alexander Plopski, Hirokazu Kato, and Klen Čopič Pucihar. 2021. No Interface, No Problem: Gesture Recognition on Physical Objects Using Radar Sensing. *Sensors* 21, 17 (2021). https://doi.org/10.3390/s21175771

[12] Nuwan T Attygalle, Una Vuletic, Matjaž Kljun, and Klen Čopič Pucihar. 2024. Towards Hand Gesture Recognition Prototype Using the IWR6843ISK Radar Sensor and Leap Motion. In *Proceedings of the 8th Human-Computer Interaction Slovenia (HCI SI) Conference 2023*. 78–88.

[13] Suzanne Aussems, Mingyuan Chu, Sotaro Kita, and Menno van Zaanen. 2015. Applying Pattern-based Classification to Sequences of Gestures. In *Proceedings of the 37th Annual Meeting of the Cognitive Science Society* (Pasadena, California, USA) *(CogSci 2015)*, David C. Noelle, Rick Dale, Anne S. Warlaumont, Jeff Yoshimi, Teenie Matlock, Carolyn D. Jennings, and Paul P. Maglio (Eds.). cognitivesciencesociety.org, Pasadena, CA, USA, 124–129. https://mindmodeling.org/cogsci2015/papers/0032/index.html

[14] Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. An Empirical Evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction* 24, 6 (2008), 574–594. https://doi.org/10.1080/10447310802205776 arXiv:https://doi.org/10.1080/10447310802205776

[15] Victor R. Basili. 1992. *Software modeling and measurement: the Goal/Question/Metric paradigm.* Technical Report. University of Maryland at College Park, USA.

[16] Joey Benedek and Trish Miner. 2002. Measuring Desirability: New methods for evaluating desirability in a usability lab setting. In *Proceedings of UPA 2002 Conference* (Orlando, FL, USA). BCS Learning & Development Ltd, BISL, P. O. Box 1454, Station RoadSwindonUnited Kingdom.

[17] Richard E. Berry. 1988. Common User Access - A Consistent and Usable Human-Computer Interface for the SAA Environments. *IBM Syst. J.* 27, 3 (1988), 281–300. https://doi.org/10.1147/SJ.273.0281

[18] François Beuvens and Jean Vanderdonckt. 2012. UsiGesture: An environment for integrating pen-based interaction in user interface development. In *Proc. of 6th International Conference on Research Challenges in Information Science* (Valencia, Spain) *(RCIS 2012)*, Colette Rolland, Jaelson Castro, and Oscar Pastor (Eds.). IEEE, Piscataway, NJ, USA, 1–12. https://doi.org/10.1109/RCIS.2012.6240449

[19] Ronan Billon, Alexis Nédélec, and Jacques Tisseau. 2010. Recognition of Gesture Sequences in Real-Time Flow, Context of Virtual Theater. In *Gesture in Embodied Communication and Human-Computer Interaction*, Stefan Kopp and Ipke Wachsmuth (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 98–109.

[20] Aaron F. Bobick and Andrew D. Wilson. 1997. A State-Based Approach to the Representation and Recognition of Gesture. *IEEE Trans. Pattern Anal. Mach. Intell.* 19, 12 (Dec. 1997), 1325–1337. https://doi.org/10.1109/34.643892

[21] Jean Bovet and Terence Parr. 2008. ANTLRWorks: an ANTLR grammar development environment. *Softw. Pract. Exp.* 38, 12 (2008), 1305–1332. https://doi.org/10.1002/SPE.872

[22] Roberto Bufano, Gennaro Costagliola, Mattia De Rosa, and Vittorio Fuccella. 2022. PolyRec Gesture Design Tool: A tool for fast prototyping of gesture-based mobile applications. *Software: Practice and Experience* 52, 2 (2022), 594–618. https://doi.org/10.1002/spe.3024 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3024

[23] Alexandre Calado, Paolo Roselli, Vito Errico, Nathan Magrofuoco, Jean Vanderdonckt, and Giovanni Saggio. 2022. A Geometric Model-Based Approach to Hand Gesture Recognition. *IEEE Trans. Syst. Man Cybern. Syst.* 52, 10 (2022), 6151–6161. https://doi.org/10.1109/TSMC.2021.3138589

[24] Francisco M. Calisto, Alfredo Ferreira, Jacinto C. Nascimento, and Daniel Gonçalves. 2017. Towards Touch-Based Medical Image Diagnosis Annotation. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 390–395. https://doi.org/10.1145/3132272.3134111

[25] Xiang Cao and Shumin Zhai. 2007. Modeling human performance of pen stroke gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 1495–1504. https://doi.org/10.1145/1240624.1240850

[26] F. M. Caputo, P. Prebianca, A. Carcangiu, L. D. Spano, and A. Giachetti. 2017. 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* (Catania, Italy) *(STAG '17)*. Eurographics Association, Goslar, DEU, 9–15. https://doi.org/10.2312/stag.20171221

[27] Alessandro Carcangiu and Lucio Davide Spano. 2018. G-Gene: A Gene Alignment Method for Online Partial Stroke Gestures Recognition. *Proc. ACM Hum.-Comput. Interact.* 2, EICS, Article 13 (June 2018), 17 pages. https://doi.org/10.1145/3229095

[28] Alessandro Carcangiu, Lucio Davide Spano, Giorgio Fumera, and Fabio Roli. 2019. DEICTIC: A compositional and declarative gesture description based on hidden markov models. *International Journal of Human-Computer Studies* 122 (2019), 113–132. https://doi.org/10.1016/j.ijhcs.2018.09.001

[29] John M. Carroll. 1982. Learning, using and designing filenames and command paradigms. *Behaviour & Information Technology* 1, 4 (1982), 327–346. https://doi.org/10.1080/01449298208914457

[30] Philip Cash and Anja Maier. 2016. Prototyping with your hands: the many roles of gesture in the communication of design concepts. *Journal of Engineering Design* 27, 1-3 (2016), 118–145. https://doi.org/10.1080/09544828.2015.1126702 arXiv:https://doi.org/10.1080/09544828.2015.1126702

[31] Hong Cheng, Lu Yang, and Zicheng Liu. 2016. Survey on 3D Hand Gesture Recognition. *IEEE Transactions on Circuits and Systems for Video Technology* 26, 9 (2016), 1659–1673. https://doi.org/10.1109/TCSVT.2015.2469551

[32] Mauricio Cirelli and Ricardo Nakamura. 2014. A Survey on Multi-touch Gesture Recognition and Multi-touch Frameworks. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (Dresden, Germany) *(ITS '14)*. Association for Computing Machinery, New York, NY, USA, 35–44. https://doi.org/10.1145/2669485.2669509

[33] Klen Čopič Pucihar, Matjaž Kljun, and Nuwan T. Attygalle. 2026. *Radar-Based Gesture Recognition on Deformable Objects.* Springer Nature Switzerland, Cham, 107–122. https://doi.org/10.1007/978-3-032-04327-6_6

[34] Klen Čopič Pucihar, Dariush Salami, and Nuwan T. Attygalle. 2026. *Current State and Future Research Directions of Radar-Based Human-Computer Interaction.* Springer Nature Switzerland, Cham, 235–267. https://doi.org/10.1007/978-3-032-04327-6_11

[35] Gennaro Costagliola, Mattia De Rosa, and Vittorio Fuccella. 2023. *Gesture-Based Computing.* John Wiley & Sons, Ltd, New York, NY, USA, Chapter 32, 397–408. https://doi.org/10.1002/9781119863663.ch32 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119863663.ch32

[36] Joëlle Coutaz, James L. Crowley, Simon Dobson, and David Garlan. 2005. Context is key. *Commun. ACM* 48, 3 (2005), 49–53. https://doi.org/10.1145/1047671.1047703

[37] Adrien Coyette, Sascha Schimke, Jean Vanderdonckt, and Claus Vielhauer. 2007. Trainable Sketch Recognizer for Graphical User Interface Design. In *Proc. of 11th IFIP TC 13 International Conference on Human-Computer Interaction, INTERACT '07* (Rio de Janeiro, Brazil) *(Lecture Notes in Computer Science, Vol. 4662)*, Maria Cecília Calani Baranauskas, Philippe A. Palanque, Julio Abascal, and Simone Diniz Junqueira Barbosa (Eds.). Springer, Berlin Heidelberg, 124–135. https://doi.org/10.1007/978-3-540-74796-3_14

[38] William Delamare, Chaklam Silpasuwanchai, Sayan Sarcar, Toshiaki Shiraki, and Xiangshi Ren. 2019. On Gesture Combination: An Exploration of a Solution to Augment Gesture Interaction. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces* (Daejeon, Republic of Korea) *(ISS '19)*. Association for Computing Machinery, New York, NY, USA, 135–146. https://doi.org/10.1145/3343055.3359706

[39] Stefano Dessì and Lucio Davide Spano. 2020. DG3: Exploiting Gesture Declarative Models for Sample Generation and Online Recognition. *Proc. ACM Hum.-Comput. Interact.* 4, EICS, Article 82 (June 2020), 21 pages. https://doi.org/10.1145/3397870

[40] Anind K. Dey. 2001. Understanding and Using Context. *Pers. Ubiquitous Comput.* 5, 1 (2001), 4–7. https://doi.org/10.1007/s007790170019

[41] Tilman Dingler, Rufat Rzayev, Alireza Sahami Shirazi, and Niels Henze. 2018. Designing Consistent Gestures Across Device Types: Eliciting RSVP Controls for Phone, Watch, and Glasses. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173993

[42] Nicholas Edward Gillian and Joseph A. Paradiso. 2014. The gesture recognition toolkit. *The Journal of Machine Learning Research* 15, 1 (2014), 3483–3487. https://doi.org/10.5555/2627435.2697076

[43] Alix Goguey and Michael Ortega. 2023. Studies and guidelines for two concurrent stroke gestures. *International Journal of Human-Computer Studies* 170 (2023), 102942. https://doi.org/10.1016/j.ijhcs.2022.102942

[44] Aakar Gupta, Thomas Pietrzak, Cleon Yau, Nicolas Roussel, and Ravin Balakrishnan. 2017. Summon and Select: Rapid Interaction with Interface Controls in Mid-air. In *Proceedings of the ACM International Conference on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 52–61. https://doi.org/10.1145/3132272.3134120

[45] Sandra G. Hart. 2006. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50, 9 (2006), 904–908. https://doi.org/10.1177/154193120605000909 arXiv:https://doi.org/10.1177/154193120605000909

[46] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, US, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[47] James Herold and Thomas F. Stahovich. 2012. The One Cent Recognizer: A Fast, Accurate, and Easy-to-Implement Handwritten Gesture Recognition Technique. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, Karan Singh and Levent Burak Kara (Eds.). The Eurographics Association, Cagliary, Italy, 39–46. https://doi.org/10.2312/SBM/SBM12/039-046

[48] Uta Hinrichs and Sheelagh Carpendale. 2011. Gestures in the wild: studying multi-touch gesture sequences on interactive tabletop exhibits. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 3023–3032. https://doi.org/10.1145/1978942.1979391

[49] Masoumehsadat Hosseini, Tjado Ihmels, Ziqian Chen, Marion Koelle, Heiko Müller, and Susanne Boll. 2023. Towards a Consensus Gesture Set: A Survey of Mid-Air Gestures in HCI for Maximized Agreement Across Domains. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 311, 24 pages. https://doi.org/10.1145/3544548.3581420

[50] Lode Hoste and Beat Signer. 2014. Criteria, Challenges and Opportunities for Gesture Programming Languages. In *Proceedings of the Workshop on Engineering Gestures for Multimodal Interfaces Co-located with the 6th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Rome, Italy) *(EGMI@EICS 2014, Vol. 1190)*, Florian Echtler, Lode Hoste, Dietrich Kammer, Beat Signer, and Davy Vanacken (Eds.). CEUR Workshop Proceedings, RWTH Aachen University, Aachen, Germany, 22–29. https://ceur-ws.org/Vol-1190/paper4.pdf

[51] ECMA International. 2017. Standard ECMA-404 – The JSON Data Interchange Syntax. https://www.ecma-international.org/publications/standards/Ecma-404.htm

[52] International Organization for Standardization. 2018. ISO 9241-11:2018 Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts. ISO Standard 9241-11:2018. https://www.iso.org/standard/63500.html

[53] Jean-François Jégo, Alexis Paljic, and Philippe Fuchs. 2013. User-defined gestural interaction: A study on gesture memorization. In *Proceedings of the IEEE Symposium on 3D User Interfaces* (Orlando, FL, USA) *(3DUI '13)*. IEEE, Orlando, FL, USA, 7–10. https://doi.org/10.1109/3DUI.2013.6550189

[54] Sven Kratz and Michael Rohs. 2010. A $3 gesture recognizer: simple gesture recognition for devices equipped with 3D acceleration sensors. In *Proceedings of the 15th International Conference on Intelligent User Interfaces* (Hong Kong, China) *(IUI '10)*. Association for Computing Machinery, New York, NY, USA, 341–344. https://doi.org/10.1145/1719970.1720026

[55] Sven Kratz and Michael Rohs. 2011. Protractor3D: A Closed-Form Solution to Rotation-Invariant 3D Gestures. In *Proceedings of the 16th International Conference on Intelligent User Interfaces* (Palo Alto, CA, USA) *(IUI '11)*. Association for Computing Machinery, New York, NY, USA, 371–374. https://doi.org/10.1145/1943403.1943468

[56] Arun Kulshreshth and Joseph J. LaViola. 2014. Exploring the usefulness of finger-based 3D gesture menu selection. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1093–1102. https://doi.org/10.1145/2556288.2557122

[57] Joseph J. LaViola and Daniel F. Keefe. 2011. 3D spatial interaction: applications for art, design, and science. In *ACM SIGGRAPH 2011 Courses* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 1, 75 pages. https://doi.org/10.1145/2037636.2037637

[58] Joseph J. LaViola Jr. 2013. 3D Gestural Interaction: The State of the Field. *International Scholarly Research Notices* 2013, 1 (2013), 514641. https://doi.org/10.1155/2013/514641 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1155/2013/514641

[59] Piotr Lech. 2016. WiFi Multi Access Point Smart Home IoT Architecture. In *Automation Control Theory Perspectives in Intelligent Systems - Proceedings of the 5th Computer Science On-line Conference 2016 (CSOC2016), Vol 3 (Advances in Intelligent Systems and Computing, Vol. 466)*, Radek Silhavy, Roman Senkerik, Zuzana Komínková Oplatková, Petr Silhavy, and Zdenka Prokopova (Eds.). Springer, Zlin, Czech Republic, 247–254. https://doi.org/10.1007/978-3-319-33389-2_24

[60] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the ACM Int. Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–17. https://doi.org/10.1145/3173574.3173610

[61] Luis A. Leiva, Daniel Martín-Albo, and Réjean Plamondon. 2016. Gestures à Go Go: Authoring Synthetic Human-Like Stroke Gestures Using the Kinematic Theory of Rapid Movements. *ACM Trans. Intell. Syst. Technol.* 7, 2 (2016), 15:1–15:29. https://doi.org/10.1145/2799648

[62] James R. Lewis. 1995. IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human–Computer Interaction* 7, 1 (1995), 57–78. https://doi.org/10.1080/10447319509526110 arXiv:https://doi.org/10.1080/10447319509526110

[63] James R. Lewis. 2006. Sample Sizes for Usability Tests: Mostly Math, Not Magic. *interactions* 13, 6 (November 2006), 29–33. https://doi.org/10.1145/1167948.1167973

[64] Yang Li. 2010. Protractor: A Fast and Accurate Gesture Recognizer. In *Proc. of the ACM Int. Conf. on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 2169–2172. https://doi.org/10.1145/1753326.1753654

[65] Katja Liebal, Josep Call, and Michael Tomasello. 2004. Use of gesture sequences in chimpanzees. *American Journal of Primatology* 64, 4 (2004), 377–396. https://doi.org/10.1002/ajp.20087 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ajp.20087

[66] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology* 22, 140 (1932), 55–. http://psycnet.apa.org/record/1933-01885-001

[67] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. UWave: Accelerometer-Based Personalized Gesture Recognition and Its Applications. *Pervasive Mob. Comput.* 5, 6 (Dec. 2009), 657–675. https://doi.org/10.1016/j.pmcj.2009.07.007

[68] Frieder Loch. 2012. Hierarchical gestures: gestural shortcuts for touchscreen devices. http://essay.utwente.nl/61931/

[69] Hao Lü, James A. Fogarty, and Yang Li. 2014. Gesture script: recognizing gestures and their structure using rendering scripts and interactively trained parts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1685–1694. https://doi.org/10.1145/2556288.2557263

[70] Tahir Mustafa Madani, Muhammad Tahir, Sheikh Ziauddin, Syed Raza, and Mirza Ahmed. 2015. An accelerometer-based approach to evaluate 3D unistroke gestures. *The International Arab Journal of Information Technology* 12, 4 (2015), 389–394. http://iajit.org/index.php?option=com_content&task=blogcategory&id=99&Itemid=376

[71] Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. 2022. Two-dimensional Stroke Gesture Recognition: a Survey. *Comput. Surveys* 54, 7 (2022), 155:1–155:36. https://doi.org/10.1145/3465400

[72] Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. 2022. μV: An Articulation, Rotation, Scaling, and Translation Invariant (ARST) Multi-stroke Gesture Recognizer. *Proc. ACM Hum.-Comput. Interact.* 6, EICS, Article 150 (June 2022), 25 pages. https://doi.org/10.1145/3532200

[73] Manavender R. Malgireddy, Ifeoma Nwogu, Subarna Ghosh, and Venu Govindaraju. 2011. A Shared Parameter Model for Gesture and Sub-gesture Analysis. In *Combinatorial Image Analysis*, Jake K. Aggarwal, Reneta P. Barneva, Valentin E. Brimkov, Kostadin N. Koroutchev, and Elka R. Korutcheva (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 483–493.

[74] Daniel Martín-Albo and Luis A. Leiva. 2016. G3: bootstrapping stroke gestures design with synthetic samples and built-in recognizers. In *Proc. of the 18th Int. Conf. on Human-Computer Interaction with Mobile Devices and Services Adjunct* (Florence, Italy) *(MobileHCI '16)*, Fabio Paternò, Kaisa Väänänen, Karen Church, Jonna Häkkilä, Antonio Krüger, and Marcos Serrano (Eds.). Association for Computing Machinery, New York, NY, USA, 633–637. https://doi.org/10.1145/2957265.2961833

[75] P. Morrel-Samuels. 1990. Clarifying the Distinction between Lexical and Gestural Commands. *Int. J. Man-Mach. Stud.* 32, 5 (may 1990), 581–590. https://doi.org/10.1016/S0020-7373(05)80034-3

[76] Meredith Ringel Morris, Anqi Huang, Andreas Paepcke, and Terry Winograd. 2006. Cooperative gestures: multi-user gestural interactions for co-located groupware. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Montréal, Québec, Canada) *(CHI '06)*. Association for Computing Machinery, New York, NY, USA, 1201–1210. https://doi.org/10.1145/1124772.1124952

[77] Tao Ni, Doug Bowman, and Chris North. 2011. AirStroke: bringing unistroke text entry to freehand gesture interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2473–2476. https://doi.org/10.1145/1978942.1979303

[78] Donald A. Norman. 2010. Natural user interfaces are not natural. *Interactions* 17, 3 (May 2010), 6–10. https://doi.org/10.1145/1744161.1744163

[79] International Standard Organization. 1996. ISO - ISO/IEC14977:1996(E) - Information technology — Syntactic metalanguage — Extended BNF. https://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf

[80] Mehdi Ousmer, Arthur Sluÿters, Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. 2020. Recognizing 3D Trajectories as 2D Multi-stroke Gestures. *Proc. ACM Hum.-Comput. Interact.* 4, ISS, Article 198 (Nov. 2020), 21 pages. https://doi.org/10.1145/3427326

[81] Mehdi Ousmer, Arthur Sluÿters, Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. 2022. A Systematic Procedure for Comparing Template-Based Gesture Recognizers. In *Proceedings of 24th International Conference on Human-Computer Interaction, HCII 2022, June 26 - July 1, 2022 (Lecture Notes in Computer Science, Vol. 13519)*, Masaaki Kurosu, Sakae Yamamoto, Hirohiko Mori, Dylan D. Schmorrow, Cali M. Fidopiastis, Norbert A. Streitz, and Shin'ichi Konomi (Eds.). Springer, Heidelberg, Germany, 160–179. https://doi.org/10.1007/978-3-031-17618-0_13

[82] Terence Parr and Kathleen Fisher. 2011. LL(*): the foundation of the ANTLR parser generator. *SIGPLAN Not.* 46, 6 (June 2011), 425–436. https://doi.org/10.1145/1993316.1993548

[83] Vik Parthiban, Pattie Maes, Quentin Sellier, Arthur Sluÿters, and Jean Vanderdonckt. 2022. Gestural-Vocal Coordinated Interaction on Large Displays. In *Companion of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Sophia Antipolis, France) *(EICS '22 Companion)*. Association for Computing Machinery, New York, NY, USA, 26–32. https://doi.org/10.1145/3531706.3536457

[84] Francesca Pulina and Fabio Paternò. 2019. Supporting cross-device interactions with gestures between personal and public devices. In *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia* (Pisa, Italy) *(MUM '19)*. Association for Computing Machinery, New York, NY, USA, Article 50, 5 pages. https://doi.org/10.1145/3365610.3368419

[85] Yosra Rekik, Laurent Grisoni, and Nicolas Roussel. 2013. Towards Many Gestures to One Command: A User Study for Tabletops. In *Human-Computer Interaction – INTERACT 2013*, Paula Kotzé, Gary Marsden, Gitte Lindgaard, Janet Wesson, and Marco Winckler (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 246–263.

[86] Yosra Rekik, Radu-Daniel Vatavu, and Laurent Grisoni. 2014. Match-up & Conquer: A Two-Step Technique for Recognizing Unconstrained Bimanual and Multi-Finger Touch Input. In *Proc. of the ACM Int. Working Conference on Advanced Visual Interfaces* (Como, Italy) *(AVI '14)*. Association for Computing Machinery, New York, NY, USA, 201–208. https://doi.org/10.1145/2598153.2598167

[87] J.R. Rhyne and C.G. Wolf. 1986. *Gestural Interfaces for Information Processing Applications*. International Business Machines Incorporated, Thomas J. Watson Research Center, Yorktown Heights, New York, United States. https://books.google.be/books?id=NzVsGwAACAAJ

[88] Quentin Roy, Sylvain Malacria, Yves Guiard, Eric Lecolinet, and James Eagan. 2013. Augmented Letters: Mnemonic Gesture-Based Shortcuts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2325–2328. https://doi.org/10.1145/2470654.2481321

[89] Priscila T. M. Saito, Rodrigo Y. M. Nakamura, Willian P. Amorim, João P. Papa, Pedro J. de Rezende, and Alexandre X. Falcão. 2015. Choosing the Most Effective Pattern Classification Model under Learning-Time Constraint. *PLOS ONE* 10, 6 (06 2015), 1–23. https://doi.org/10.1371/journal.pone.0129947

[90] Ovidiu-Andrei Schipor, Radu-Daniel Vatavu, and Jean Vanderdonckt. 2019. Euphoria: A Scalable, event-driven architecture for designing interactions across heterogeneous devices in smart environments. *Inf. Softw. Technol.* 109 (2019), 43–59. https://doi.org/10.1016/J.INFSOF.2019.01.006

[91] Martin Schrepp, Andreas Hinderks, and Thomaschewski Jörg. 2017. Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence* 4, 6 (Dec. 2017), 103–108. https://doi.org/10.9781/ijimai.2017.09.001

[92] Vaidyanath Areyur Shanthakumar, Chao Peng, Jeffrey T. Hansberger, Lizhou Cao, Sarah C. Meacham, and Victoria R. Blakely. 2020. Design and evaluation of a hand gesture recognition approach for real-time interactions. *Multim. Tools Appl.* 79, 25-26 (2020), 17707–17730. https://doi.org/10.1007/s11042-019-08520-1

[93] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. 2016. *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (6 ed.). Pearson, London, United Kingdom. https://www.pearson.com/en-us/subject-catalog/p/designing-the-user-interface-strategies-for-effective-human-computer-interaction/P200000003485/9780137503889

[94] A. Siewierska. 2001. Word Order. In *International Encyclopedia of the Social & Behavioral Sciences*, Neil J. Smelser and Paul B. Baltes (Eds.). Pergamon, Oxford, 16552–16555. https://doi.org/10.1016/B0-08-043076-7/02968-5

[95] Beat Signer, U. Kurmann, and Moira C. Norrie. 2007. iGesture: A General Gesture Recognition Framework. In *Proceedings of 9th International Conference on Document Analysis and Recognition* (Curitiba, Paraná, Brazil) *(CDAR 2007)*. IEEE Computer Society, Piscataway, NJ, USA, 954–958. https://doi.org/10.1109/ICDAR.2007.4377056

[96] Arthur Sluÿters, Sébastien Lambot, Jean Vanderdonckt, and Radu-Daniel Vatavu. 2023. RadarSense: Accurate Recognition of Mid-air Hand Gestures with Radar Sensing and Few Training Examples. *ACM Trans. Interact. Intell. Syst.* 13, 3, Article 16 (Sept. 2023), 45 pages. https://doi.org/10.1145/3589645

[97] Arthur Sluÿters, Mehdi Ousmer, Paolo Roselli, and Jean Vanderdonckt. 2022. QuantumLeap, a Framework for Engineering Gestural User Interfaces based on the Leap Motion Controller. *Proc. ACM Hum. Comput. Interact.* 6, EICS (2022), 161:1–161:47. https://doi.org/10.1145/3532211

[98] Arthur Sluÿters, Jean Vanderdonckt, Paolo Roselli, and Radu-Daniel Vatavu. 2025. Congruent and Hierarchical Gesture Set Design. In *Companion Publication of the 2025 ACM Designing Interactive Systems Conference* (Funchal, Madeira, Portugal) *(DIS 2025)*, Nuno Jardim

Nunes, Valentina Nisi, Ian Oakley, Clement Zheng, and Qian Yang (Eds.). Association for Computing Machinery, New York, NY, USA, 419–426. https://doi.org/10.1145/3715668.3736383

[99] Arthur Sluÿters, Quentin Sellier, Jean Vanderdonckt, Vik Parthiban, and Pattie Maes. 2022. Consistent, Continuous, and Customizable Mid-Air Gesture Interaction for Browsing Multimedia Objects on Large Displays. *International Journal of Human–Computer Interaction* 0, 0 (2022), 1–32. https://doi.org/10.1080/10447318.2022.2078464 arXiv:https://doi.org/10.1080/10447318.2022.2078464

[100] Lucio Davide Spano, Antonio Cisternino, and Fabio Paternò. 2012. A Compositional Model for Gesture Definition. In *Proceedings of 4th International Conference on Human-Centered Software Engineering, HCSE '12* (Toulouse, France) *(Lecture Notes in Computer Science, Vol. 7623)*, Marco Winckler, Peter Forbrig, and Regina Bernhaupt (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 34–52. https://doi.org/10.1007/978-3-642-34347-6_3

[101] V. Sylaidis, I. Nanakis, and V. Kopanas. 1997. *Introducing the Goal-Question-Metric approach to telecommunications software development: the PITA experiment.* Springer US, Boston, MA, 215–230. https://doi.org/10.1007/978-0-387-35097-4_17

[102] Eugene M. Taranta, II and Joseph J. LaViola, Jr. 2015. Penny Pincher: A Blazing Fast, Highly Accurate $-family Recognizer. In *Proc. of the 41st Graphics Interface Conference* (Halifax, Nova Scotia, Canada) *(GI '15)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 195–202. http://dl.acm.org/citation.cfm?id=2788890.2788925

[103] Eugene M. Taranta II, Amirreza Samiei, Mehran Maghoumi, Pooya Khaloo, Corey R. Pittman, and Joseph J. LaViola Jr. 2017. Jackknife: A Reliable Recognizer with Few Samples and Many Modalities. In *Proc. of the ACM Int. Conf. on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5850–5861. https://doi.org/10.1145/3025453.3026002

[104] Junyun Tay and Manuela Veloso. 2012. Modeling and composing gestures for human-robot interaction. In *Proceedings of the 21st IEEE International Symposium on Robot and Human Interactive Communication* (Paris, France) *(RO-MAN '12)*. IEEE Press, Piscataway, NJ, USA, 107–112. https://doi.org/10.1109/ROMAN.2012.6343739

[105] Petr Vanc, Jan Kristof Behrens, Karla Stepanova, and Vaclav Hlavac. 2023. Communicating human intent to a robotic companion by multi-type gesture sentences. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (Detroit, MI, USA) *(IROS '23')*. IEEE Press, Piscataway, NJ, USA, 9839–9845. https://doi.org/10.1109/IROS55552.2023.10341944

[106] Jean Vanderdonckt, Paolo Roselli, and Jorge Luis Pérez-Medina. 2018. !FTL, an Articulation-Invariant Stroke Gesture Recognizer with Controllable Position, Scale, and Rotation Invariances. In *Proc. of the 20th ACM Int. Conf. on Multimodal Interaction* (Boulder, CO, USA) *(ICMI '18)*. Association for Computing Machinery, New York, NY, USA, 125—-134. https://doi.org/10.1145/3242969.3243032

[107] Radu-Daniel Vatavu. 2012. 1F: One Accessory Feature Design for Gesture Recognizers. In *Proc. of the ACM Int. Conf. on Intelligent User Interfaces* (Lisbon, Portugal) *(IUI '12)*. Association for Computing Machinery, New York, NY, USA, 297–300. https://doi.org/10.1145/2166966.2167022

[108] Radu-Daniel Vatavu. 2017. Improving Gesture Recognition Accuracy on Touch Screens for Users with Low Vision. In *Proc. of the ACM Int. Conf. on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4667–4679. https://doi.org/10.1145/3025453.3025941

[109] Radu-Daniel Vatavu. 2024. Gesture-Based Interaction. In *Handbook of Human-Computer Interaction*, Jean Vanderdonckt, Philippe Palanque, and Marco Winckler (Eds.). Springer International Publishing, Cham, 1–47. https://doi.org/10.1007/978-3-319-27648-9_20-1

[110] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes. In *Proc. of the 14th ACM Int. Confe. on Multimodal Interaction* (Santa Monica, California, USA) *(ICMI '12)*. Association for Computing Machinery, New York, NY, USA, 273–280. https://doi.org/10.1145/2388676.2388732

[111] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2018. $Q: A Super-quick, Articulation-invariant Stroke-gesture Recognizer for Low-resource Devices. In *Proc. of the 20th Int. Conf. on Human-Computer Interaction with Mobile Devices and Services* (Barcelona, Spain) *(MobileHCI '18)*. Association for Computing Machinery, New York, NY, USA, Article 23, 12 pages. https://doi.org/10.1145/3229434.3229465

[112] Klen Čopič Pucihar, Nuwan T. Attygalle, Matjaz Kljun, Christian Sandor, and Luis A. Leiva. 2022. Solids on Soli: Millimetre-Wave Radar Sensing through Materials. *Proc. ACM Hum.-Comput. Interact.* 6, EICS, Article 156 (June 2022), 19 pages. https://doi.org/10.1145/3532212

[113] Santiago Villarreal-Narvaez, Arthur Sluÿters, Jean Vanderdonckt, and Radu-Daniel Vatavu. 2024. Brave New GES World: A Systematic Literature Review of Gestures and Referents in Gesture Elicitation Studies. *ACM Comput. Surv.* 56, 5, Article 128 (Jan. 2024), 55 pages. https://doi.org/10.1145/3636458

[114] Santiago Villarreal-Narvaez, Jean Vanderdonckt, Radu-Daniel Vatavu, and Jacob O. Wobbrock. 2020. A Systematic Review of Gesture Elicitation Studies: What Can We Learn from 216 Studies?. In *Proceedings of the ACM Designing Interactive Systems Conference* (Eindhoven, Netherlands) *(DIS '20)*. Association for Computing Machinery, New York, NY, USA, 855–872. https://doi.org/10.1145/3357236.3395511

[115] Panagiotis Vogiatzidakis and Panayiotis Koutsabasis. 2021. Mid-air gestures for manipulation of multiple targets in the physical space: comparing the usability of two interaction models. In *CHI Greece 2021: 1st International Conference of the ACM Greek SIGCHI Chapter* (Online (Athens, Greece), Greece) *(CHI Greece 2021)*. Association for Computing Machinery, New York, NY, USA, Article 15, 9 pages. https://doi.org/10.1145/3489410.3489425

[116] Panagiotis Vogiatzidakis and Panayiotis Koutsabasis. 2022. 'Address and command': Two-handed mid-air interactions with multiple home devices. *International Journal of Human-Computer Studies* 159 (2022), 102755. https://doi.org/10.1016/j.ijhcs.2021.102755

[117] Sy Bor Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell. 2006. Hidden Conditional Random Fields for Gesture Recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, NY, USA, 1521–1527. https://doi.org/10.1109/CVPR.2006.132

[118] Edmond H. Weiss. 1990. Visualizing a Procedure with Nassi-Schneiderman Charts. *Journal of Technical Writing and Communication* 20, 3 (1990), 237–254. https://doi.org/10.2190/0UVT-TWMK-LN59-UKN8 arXiv:https://doi.org/10.2190/0UVT-TWMK-LN59-UKN8

[119] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-Defined Gestures for Surface Computing. In *Proc. of the ACM Int. Conf. on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1083—-1092. https://doi.org/10.1145/1518701.1518866

[120] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proc. of the 20th Annual ACM Symposium on User Interface Software and Technology* (Newport, Rhode Island, USA) *(UIST '07)*. Association for Computing Machinery, New York, NY, USA, 159–168. https://doi.org/10.1145/1294211.1294238

[121] Haijun Xia, Michael Glueck, Michelle Annett, Michael Wang, and Daniel Wigdor. 2022. Iteratively Designing Gesture Vocabularies: A Survey and Analysis of Best Practices in the HCI Literature. *ACM Trans. Comput. Hum. Interact.* 29, 4 (2022), 37:1–37:54. https://doi.org/10.1145/3503537

[122] Shan Xu, Sarah Sykes, Parastoo Abtahi, Tovi Grossman, Daylon Walden, Michael Glueck, and Carine Rognon. 2024. Designing Haptic Feedback for Sequential Gestural Inputs. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 711, 17 pages. https://doi.org/10.1145/3613904.3642735

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

[124] Shumin Zhai, Per Ola Kristensson, Caroline Appert, Tue Haste Anderson, and Xiang Cao. 2012. Foundational Issues in Touch-Surface Stroke Gesture Design — An Integrative Review. *Foundations and Trends in Human–Computer Interaction* 5, 2 (2012), 97–205. https://doi.org/10.1561/1100000012

[125] Yougen Zhang, Wei Deng, Hanchen Song, and Lingda Wu. 2013. A Fast Pen Gesture Matching Method Based on Nonlinear Embedding. In *Advances in Image and Graphics Technologies*, Tieniu Tan, Qiuqi Ruan, Xilin Chen, Huimin Ma, and Liang Wang (Eds.). Springer, Berlin, Heidelberg, 223–231.

[126] Klen Čopič Pucihar, Radu-Daniel Vatavu, Dariush Salami, and Nuwan T. Attygalle (Eds.). 2026. *Radar-Based Human-Computer Interaction* (1 ed.). Springer Nature Switzerland, Cham. XIX, 270 pages. https://doi.org/10.1007/978-3-032-04327-6