Gelicit: A Cloud Platform for Distributed Gesture Elicitation Studies

NATHAN MAGROFUOCO, Université catholique de Louvain, Belgium
JEAN VANDERDONCKT, Université catholique de Louvain, Belgium

A gesture elicitation study, as originally defined, consists of gathering a sample of participants in a room, instructing them to produce gestures they would use for a particular set of tasks, materialized through a representation called referent, and asking them to fill in a series of tests, questionnaires, and feedback forms. Until now, this procedure is conducted manually in a single, physical, and synchronous setup. To relax the constraints imposed by this manual procedure and to support stakeholders in defining and conducting such studies in multiple contexts of use, this paper presents Gelicit, a cloud computing platform that supports gesture elicitation studies distributed in time and space structured into six stages:
(1) define a study: a designer defines a set of tasks with their referents for eliciting gestures and specifies an experimental protocol by parameterizing its settings; (2) conduct a study: any participant receiving the invitation to join the study conducts the experiment anywhere, anytime, anyhow, by eliciting gestures and filling forms; (3) classify gestures: an experimenter classifies elicited gestures according to selected criteria and a vocabulary; (4) measure gestures: an experimenter computes gesture measures, like agreement, frequency, to understand their configuration; (5) discuss gestures: a designer discusses resulting gestures with the participants to reach a consensus; (6) export gestures: the consensus set of gestures resulting from the discussion is exported to be used with a gesture recognizer. The paper discusses Gelicit advantages and limitations with respect to three main contributions: as a conceptual model for gesture management, as a method for distributed gesture elicitation based on this model, and as a cloud computing platform supporting this distributed elicitation. We illustrate Gelicit through a study for eliciting 2D gestures executing Internet of Things tasks on a smartphone.

Additional Key Words and Phrases: elicitation technique; gesture elicitation study; gesture interaction; workflow analysis

ACM Reference Format:
Nathan Magrofuoco and Jean Vanderdonckt. 2019. Gelicit: A Cloud Platform for Distributed Gesture Elicitation Studies. *Proc. ACM Hum.-Comput. Interact.* 3, EICS, Article 6 (June 2019), 41 pages. https://doi.org/10145.3331146

1 INTRODUCTION

The affordability of gesture acquisition technologies, such Microsoft Kinect, Thalmic Myo armband, Microchip 3DTouchPad, as well as the availability of supporting software, such as MS Surface gesture collection, Myo armband basic gestures, have launched a new generation of User Interfaces (UIs) promoting gesture interaction that is important to prototype [36]. Consequently, the interest for acquiring [52, 69], managing [60], analyzing [13], studying [68], and recognizing [91] gestures

Authors' addresses: Nathan Magrofuoco, Université catholique de Louvain, LouRIM Institute, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium, nathan.magrofuoco@uclouvain.be; Jean Vanderdonckt, Université catholique de Louvain, LouRIM Institute, Place des Doyens, 1, ICTeam Institute, Avenue Georges Lemaître, 4, Louvain-la-Neuve, 1348, Belgium, jean.vanderdonckt@uclouvain.be.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
2573-0142/2019/6-ART6 $15.00
https://doi.org/10145.3331146

has significantly increased over years. Yet, *system-defined gestures* promoted by software vendors and hardware manufacturers are reported to be difficult to learn [94], remember [65], and reproduce [33]. Hence, UI designers are progressively turning to a *gesture elicitation study* to identify and explore *user-defined gestures* guessed by real-world end users, especially when they recognize that *designer-defined gestures* do not match well with them [63].

A *Gesture Elicitation Study* (GES) prompts participants to elicit gestures for executing tasks in an interactive application. Until now, all GES are conducted manually in a physical and synchronous setup, thus imposing a set of constraints which consume a significant amount of time, efforts, and resources [4]: participants have to be present for a certain period according to a precise time schedule in the physical location where the GES is conducted. And so are the stakeholders involved, such as designers, developers, experimenters, researchers, or investigators.

All GES data are manually captured in paper form and need to be translated into computer files for further data handling and computation. Once participants have left the room, there is little or no chance to involve them again in any future discussion or in another experiment. Reusability of the GES setup and data is compromised by its unique occurrence. If another, perhaps similar, GES should be conducted, the experimental protocol needs to be redefined and re-run.

To release GES from these constraints, to leverage its capabilities to lead it to its full potential, Gelicit (Gesture elicitation platform) aims at providing a cloud computing platform for supporting the full workflow of a GES distributed in time and space, with a unique combination of features: zero-install cloud computing platform with user accounts, persistent management of gesture data, support for distributing stages in time and space, flexible GES parametrization, flexible gesture classification and measurement, computer-aided computation of measures, support for discussion, and continuity with the rest of the development life cycle.

Hence, the contribution of this paper to the field of Engineering Interactive Computing Systems (EICS) is threefold:

(1) Gelicit as a *metamodel* for gesture elicitation: a metamodel covering an array of gestures ranging from 2D (stroke) to 3D (mid-air) gestures is provided that accommodates various GES contexts of use. This metamodel is used to structure and implement the Gelicit database for persistent management of gesture data.
(2) Gelicit as a *guiding method*: based on the metamodel, the originally defined GES method is expanded into a new version where each stage can be distributed in time and space and parameterized depending on the conditions imposed by the context of use, in which stakeholders can play different roles.
(3) Gelicit as a *software* supporting the guiding method by enacting its configuration and automating its stages.

The remainder of this paper is organized as follows: Section 2 demonstrates to what extent GES is important, frequent, recent, widening in scope, and presents a review of the most relevant software tools for supporting partially or totally a GES. Section 3 reports on the process for obtaining Gelicit specifications. Based on these results, Section 4 defines the Gelicit metamodel, its guiding method supporting software. Section 5 exemplifies the workflow with a study for eliciting gestures for Internet of Things (IoT) tasks to be executed on a smartphone. Section 6 compare Gelicit's benefits and current limitations by classifying them according to the roles of designer, developer, experimenter, and end user. Section 7 concludes the paper and presents some avenues for future work based on current limitations.

2 RELATED WORK

This section defines the terminology used throughout the paper, discusses some GES background to give a baseline for reviewing some significant software supporting GES afterwards.

2.1 GES Terminology

We define our terminology based on a reference model structuring any interaction elicitation study, not just for gestures. A *task* describes any activity that has to be carried out to fulfill the goals of end users, real or represented, actual or future. A *task set* gathers a series of users' tasks to initiate an interaction elicitation study. For example, Vatavu and Zaiţi [95] address mid-air gesture input to control a Smart TV by a set of 21 tasks: open/close a TV set, next/previous channel, volume up/down/mute, open/hide menu, help, yes/no answer, go to first/second favorite channel, go to random/#7/#27 channel, open the TV guide, show channels list, and open browser. This task set is reused in several other studies, such as [31, 93]. Each task set is materialized through a series of referents depending on the context of use [15] in which a user (e.g., a TV watcher) will carry out the intended tasks (e.g., select a channel) in a given environment (e.g., a living room).

A *referent* materializes a task depending on its context of use by providing a scene before and after carrying out a task, along with a description of the task. A referent therefore avoids to make any reference to how the task could be carried out (e.g., it does not provide starting gestures for which there should be some elicitation). An *elicited artifact* refers to any symbol elicited by an end user to execute the task based on its referent, e.g., a function key, a command language, a series of actions, or a gesture for which different representations could be preferred [59]. Each elicited artifact could be subject to one or many measures that are defined by a *measurement method*. A measure could be qualitative (e.g., the subjective user satisfaction) or quantitative (e.g., the complexity of the symbol). Any such measurement method should be appropriate for the referent provided.

We decompose the whole GES workflow into six stages:

(1) *Define a study*: covers the expression of all parameters and specifications required to conduct a GES before it is actually conducted. This potentially includes specifying the task set, the associated referents, the context of use with its own parameters, and the various data that need to be collected when running the GES.
(2) *Conduct a study*: covers the running of the previously defined GES, which results in a series of elicited gestures along with their associated data.
(3) *Classify gestures*: covers the classification of previously elicited gestures according to any classification method, such as based on clustering criteria.
(4) *Measure gestures*: aims at computing any measure of interest on previously classified or elicited gestures according to a measurement method.
(5) *Discuss gestures*: covers the interaction among the stakeholders to come up with a consensus set of gestures.
(6) *Export gestures*: covers the conversion of consensus gesture data so as to prepare them to be integrated into one or many gesture recognizer and/or any module for supporting gesture recognition.

GELICIT is not aimed at addressing gesture recognition since extensively covered by a large family of 2/3D gesture recognizers, such as those from the $-family, culminating with its most recent and fastest member $Q [92]. Rather, GELICIT reused some of these established recognizers, such as $1 [99] and $N [5]. Consequently, gesture recognizers were not part of this review. GELICIT is not aimed at supporting any possible interaction elicitation study [4] since they are wide and deep in scope. Rather, GELICIT only focused on elicitation studies for (some) gestures. GELICIT does not pretend to support all possible modalities to elicit gestures, like non-touch gestures [6]. Rather,

it supports some 2D stroke gestures acquired in a HTML5 canvas by any pointing device, 2D1/2 gestures by object tracking, and 3D air+touch [20]. Finally, GELICIT is not aimed at facilitating the development of gesture-based user interfaces since this is also covered by other works.

2.2 Background on Gesture Elicitation Studies

Many GES are conducted along dimensions of the context of use [15]: users and their tasks, platforms and devices, and environments, which are respectively discussed in the next subsections.

2.2.1 With various platforms and devices. Since their inception, GES primarily focused on some particular computing platform or device, such as tabletops in surface computing [64, 98], smartphones in mobile interaction [76], and smart television [25, 95]. Some GES concerned gestures either for multiple devices at once, such as in Multi-Display Environments [81], Multi-windowing [51] or across devices, such as for cross-device interaction [69] or between a combination, e.g., mobile phones, public displays, and tabletops synchronized [42]. As long as technology progressed, so were GES for more modern gestures, like on a hologram [100], with augmented reality [73] for holograms [72], and ring devices [31]. When several devices were exploited simultaneously, the question of maintaining consistency among gestures immediately arised that requires a transfer measure[24].

2.2.2 In different environments. Gestures were typically elicited in a particular physical environment in which devices were determined, such as the steering wheel in a car [26]. The location in which gestures were issued also played some predominant role, like in public displays where social acceptance significantly influenced results [74]. When a user migrated from one environment equipped with a particular device, to another with heterogeneous devices, the question of gesture knowledge transfer also arised [90].

2.2.3 For multiple tasks. The tasks conducted in the aforementioned environments coul also be very different, such as tasks for selecting and buying a good in a store through a public display [30] or opt-in/out for public displays [74]. Space and place had some effect on the engagement of participants in such studies [3].

2.2.4 For multiple users. GES first targeted users populations with a sampling in a *user-independent* fashion (i.e., no particular profile was involved) or *user-dependent* (particular profiles were stratified). For instance, some whole-body gestures were elicited by children in [21]. Different types of users played different roles, with their culture influencing their input [58]. Different parts of the human body were also subject to GES: [12] studied hand gestures, while [33] compared freehand against on-skin gestures. This demonstrated that any particular human ability or physical capability or the deficiency thereof also became the central subject of a GES. Gestures could be also constrained by type, such as hand gestures [12], foot gestures [27], micro-gestures with one hand only [18] or not, like in whole-body gestures [21, 77].

2.2.5 GES analyses. GES were considered **important** for any context of use [96] since they could be conducted for any combination of users, platforms and devices, and environments. If the study was not conducted in the right context of use, the risk for producing biased gestures was real [96]. Without any appropriate contextual cue, eliciting user-defined gestures becomes impractical [21].

GES were also **frequently** conducted since their inception in 2009. A multi-database search with the query "gesture elicitation study" returned 178 peer-reviewed unique papers until October 20^{th}, 2018, all reporting some GES with variation in terms of users, task, devices, platforms, and environments: 107 references from ACM Digital Library, 7 other references from IEEE XPlore, 23

from SpringerLink, 6 from ScienceDirect, and 35 identified by Google Scholar referencing peer-reviewed papers which did not appear before. The total amount of such studies would probably have exceeded 200 if master theses, PhD theses, technical reports were counted.

GES continued to be **recently** conducted an reported. Most of them were published in the last five years, which was probably due to the pioneer guessability method published in 2005 [97], subsequently refined in [98] and finally in [93, 94]. Twenty-three GES have been published in the first half of 2018, the last one being [61], which may be reasonably projected to the forty GES appeared in 2017. GES have been recently published at premier scientific venues like CHI [8, 24, 27, 78] and DIS [31, 72], and taught in text books, like [57].

GES saw their aims and scope being **widened** with time. Not only other areas close to HCI started to launch such studies, such as [86] for interactive television (TVX) or [83] for children interaction design (IDC). But also other disciplines, inside computer science, such as 3D CAD-CAM [43], and outside, such as communication [43] and medicine [56] for accessing patient records. Studies also started to appear with multiple contexts of use, from within-subjects [93] to between-subjects [94] experiments and measures (e.g., comparing female vs male, old vs young) or across devices, like between smartphones, tablets, and smart glasses [24].

Yet, GES were subject to a debate of open questions. Morris et al. [63] observed that participants may be subject to a legacy bias when involved in a GES [8]: participants may be inclined to reproduce what they already know in a previous context of use instead of being creative and proposing really new gestures for the intended context of use. To address these shortcomings, two techniques were tested for reducing the legacy bias revealed by [63]: kinesthetic priming and increased production [34]. Soft constraints are another technique for the same purpose [77]. On the other hand, a GES could benefit from the legacy bias [41]. Agreement scores [97] and rates [94] were two measures for estimating the likelihood that pairs of end users would agree or disagree on their elicited gestures. Other measures have been also computed as agreement measures are subject to some discussion [88]. In conclusion, conducting a GES represented a significant effort for identifying user-defined gestures. Although many GES have been reported, it was likely that future studies will require appropriate support to be conducted in an effective and efficient way, in particular when multiple contexts of use are entailed [90].

2.3 Background on Gesture Software Support

Several works covered some parts of the GES full process. Therefore, we presented a review of only related works on gesture software by discussing their capabilities and their advantages, shortcomings by category.

2.3.1 Gesture Design Tools. They aimed at capturing and manipulating gestures into collections to organize them and test their recognition with one or several recognizers for design purposes only. **Quill** [52] is a Java software helping a single user interface designer to acquire unistroke gestures and to manipulate them by applying a translation, a dilatation, or a rotation. **MAGIC** [7] is a design tool for helping designers in identifying, capturing, and testing motion gestures for games and for other interactive applications. When a motion gesture has been used in production, a video recording its recognition is made available to the designer for annotation and analysis. **CUBOD** [87] is a design tool for body gestures optimizing the real-time recognition. **Myo GestureControl** [1] enables an experimenter to gather surface electromyography gestures acquired by a Thalmic Myo armband and to classify them with an accuracy that is comparable to other similar devices in medical applications. While these tools are more focusing on the design stage, they sometimes consider how to transfer their results to the development stage, thus also touching the category of development environments.

Software	Define	Conduct	Classify	Measure	Discuss	Export
Quill	Pen-based gestures	Samples acquired and managed in the software	Cluster analysis			Gesture file
Magic	Motion gestures	Samples acquired in the software			Retrospection	
CUBOD	Full-body gestures	Samples acquired in the software	Real-time recognition			
Gesture Control	Mid-air gestures	Samples acquired in the software	Pattern recognition			
iGesture	Reference gestures	Test gestures in the software	Test bench and recognition			Database with event manager
Usi-Gesture	Unistroke gestures	Samples acquired in the software	Levenshtein recognizer			Gesture set with packages
Kind-Dama	Stroke gestures	Samples externally acquired	Recognition engine	Agreement rate		Gesture set with recognizer
Aramis	Multimodal gestures	Samples stored in files	Gesture recognition			Gesture set with fusion engine
MDGG	Single-hand mid-air	Samples acquired in the software	Multi-Layer Neu-ral network			
Proton++	Multi-touch gestures	Samples acquired in the software				Proton++ gestures
Gesture Coder	Multi-touch gestures	Demonstrations and declarations in the software				Programming code
GestIt	Multi-touch gestures	Samples acquired in the software				GestIt gestures
Pro Gesture	Full-body gestures	Samples acquired in the software				Gestures prototyped
Gesture Script	Multi-touch gestures	Samples acquired in the software				Rendering Scripts
Gesture Wiz	Freeform gesture templates	Samples by participants and MTurk crowd workers	Nearly real-time recognition		Conflict checking	Gesture set with callback procedure
Gesture Analyzer	Multi-pose gestures	Add poses within the same application	Hierarchical clust., video annotation			Gesture set
Kinect Analysis	Full-body gestures	Samples acquired in the software	Pattern matching			Gesture set
Buzzi	Unistroke single-touch	Samples via Web application				Gestures via web application
AGATe	Any gesture elicited			(co/dis)-agreement		
Gest Analytics	Hand gest. by video	Samples acquired in the software	Video tagging and annotation			
Crowd Sensus	Vocal commands	Commands defined in the application	Comparison interfaces	Agreement score		
Gelicit	Task set, referents, context of use, criteria,...	Elicited gestures by any registered participant, distributed in time and space	Multiple criteria, vocabulary, and labelling	Agreement rate, frequency,...	Rating, ranking, testing	Consensus set with recognizer

Table 1. Comparison of related work against workflow stages.

2.3.2 Gesture Middleware Systems. They aimed at providing an intermediary software layer between the functional core and the gesture user interface of an interactive application. **iGesture** [82] consists of a Java middleware providing an abstraction layer to capture gestures from a pen or a mouse, to recognize them based on a multi-recognizer approach, and organize them in a gesture set. Similarly to the previous family, iGesture is focusing on gesture management: gestures are hard-coded in their own format and cannot be captured and managed by different stakeholders in parallel since it was not the aim of this middleware. **UsiGesture** [10], although incorporated in a workflow with four distinct roles for gesture management, can only be used by a single developer at once and gestures cannot be captured by different users at the same time. More recently, **KIND-DAMA** [60] is an open source modular middleware for developing interactive applications based on gestures. It also offers an abstraction layer and the concurrent capturing of gestures by ensuring a transparent communication between devices and applications. As such, a GES can indeed be supported for its "Conduct" and the "Export" stages as demonstrated in a case study, thus leaving the other stages outside the middleware. **Euphoria** [80] is a scalable, event-driven software architecture for designing flexible, heterogeneous, interactions based on gestures acquired and fused from different sources.

2.3.3 Gesture Development Environments. They aimed at covering as much as possible the development life cycle of a gesture user interface of an interactive application, not just designing gestures as in the first family.

ARAMIS [17] is a development framework that facilitates the development of gesture user interfaces in ubiquitous computing for wearable devices. It provides a layer for synchronizing data streams coming from devices and a multimodal fusion engine to manage concurrent inputs. As such, it is mainly oriented towards the development of gesture-based applications and not for eliciting gestures, although it also offers an abstraction layer and concurrent capture.

MDDGG [28] aimed at rapidly prototyping gesture user interfaces according to a model-based gesture navigation design based on state charts. A declarative modeling is used to design and generate many variants of a gesture-based navigation control with gestures associated to the following tasks: previous, next, closer, farther, select, and confirm.

Proton++ [40] pursues the same aim in specifying gesture user interfaces but offers three different programming paradigms: declarative programming, by example, or combined.

GestureCoder [54] facilitated the development of multi-touch gesture user interface also with these three paradigms. A gesture can be demonstrated in the environment and turned immediately into code for recognition and integration. The declaration extension enables the developer to specify high level events and actions, such as "on recognition of a certain gesture, display concurrently that artefact". **GestureStudio** [55], the successor of GestureCoder, exploited the same paradigms with a video-editing metaphor. A designer demonstrates any touch gesture of interest (e.g., flick, zoom) by a video camera, composes complicated behaviors, tests them straightforwardly in the environment, and exports them as source code to be integrated in a project.

GestIt [84] aimed at modelling gestures for incorporating them in the development by composition of gestures using temporal and composition operators. Any recorded gesture is then attached to a user interface model in which dialog feedback is bound to the leafs and nodes of a modelled decision tree.

Note that GELICIT manages and characterizes gestures for the sole purpose of gesture elicitation, not for development nor for modeling or composition of gestures as GESTIT does. Hence, GELICIT does not address the problem of granularity: Each gesture is simply captured by its reference points and no specification is provided. When exported, gestures are delivered according to a single event

pattern: when gesture is recognized, do [action]. More elaborate events, listeners, or composition of gestures should be addressed outside the elicitation thanks to other environments like Gestit.

GestureScript [53] enhanced gestures by example through rendering scripts (which contains synthesized variations of a same gesture and gesture properties that are directly useful for recognizers) and interactively trained parts.

ProGesture [11] supported rapid prototyping of full-body gestures by incorporating them within a gesture-presentation-dialog design space according to a model-based approach. Gestures are directly captured and assigned to dialog models based on state charts (as in MDDGG) so that they can be tested and evaluated in their very right context of use.

HGRE [39] enables designers to acquire mid-air hand gestures based on surface electromyography by inviting participants to 'play' a game of reproducing gestures exemplified by a camera. These samples are directly acquired in the environment and further exploited for gesture recognition.

GestureWiz [85] was the most recent and advanced manifestation of this family. It consists of an environment for developing gesture user interfaces with rapid prototyping of gestures based on three stages: 1) a designer records a set of template gestures through the Requester user interface; 2) a developer maps the recorded gestures to commands of an interactive application through the library; and 3) a Mechanical Turk crowd worker or a participant tests any gesture for recognition and comparison with respect to the template gestures. When the gesture is recognized, the corresponding command is executed; when it is not, the conflict checker helps resolving ambiguities and assignments.

2.3.4 Gesture Analysis Tools. They aimed at supporting any stakeholder in the "Measure" stage, where various measures should be computed on the gestures collected, such as similarity measure [52], agreement measures [29, 94, 97], and cluster analysis [37].

GestureAnalyzer [37] enabled designers to capture video sessions in which participants elicit gestures and annotate them afterwards to classify gestures. Since GestureAnalyzer was primarily a video annotation software, it partially supports the "Classify" stage, but does not store any gesture in any other form than video, thus preventing them to reuse the results afterwards, like in the "Discuss" and "Export" stages.

KinectAnalysis [67] was a system for recording, analyzing and sharing multimodal interaction elicitation studies by analyzing gestures produced by a Microsoft Kinect. It is associated with Kinect software environment, like XDKinect [69].

Buzzi et al. [14] developed a software for capturing unistroke, single touch gestures from three visually impaired participants each working on their smartphone simultaneously. Their system consisted of three components: a server dashboard for parameterizing, monitoring, and visualizing gestures, a custom web application on a smartphone for capturing the gestures parameterized, and a web server between. The three users were co-located in the same usability laboratory.

AGATe [94] was a C# toolkit running on MS Windows that computes the three scores, i.e., agreement rate, disagreement rate, and co-agreement rate based only on gesture ID (no capture and classification) and counts. Since it is not connected to the rest the workflow, the various stakeholders, such as the UI designers, are responsible for defining and conducting the method beforehand and to consolidate the results afterwards.

GestAnalytics [13] was a software for analyzing hand gestures captured by video by offering the following original features that go beyond a simple video annotation tool: simultaneous video monitoring, video tagging and filtering. But it is focusing on only one type of gesture and one medium, without considering the rest of the study.

CrowdSensus [4] was a crowd-based software that enables experimenters to analyze the GES results more efficiently than manually using the crowd as a reference, subjective human judgment

and automatic clustering. Different types of agreement, namely based on annealing, offer alternative ways to estimate agreement between participants. Like GestureWiz, CrowdSensus exploits the tremendous capabilities of the crowd to reach almost the same conclusion than experts, but faster and even sometimes more consensually.

3 ANALYSIS AND SPECIFICATIONS OF GELICIT

Prior to designing and developing GELICIT, a series of activities was conducted to inform its analysis and deliver its specifications. These activities are reported in the next subsections.

3.1 Comparative Analysis of Related Software

We conducted a comparative analysis of GES-related software to determine their facilities. Table 1 compares the works reviewed in Section 2.3 against the six stages defined in Section 2.1: "Define", "Conduct", "Classify", "Measure", "Discuss", and "Export", while Table 2 compares them against technical parameters. The members of the four aforementioned categories either support some stages of the GES workflow or pursue another goal such as designing, analysis, and visualization of gestures. They are not intended to cover the whole GES workflow per se, either physically or digitally, but they could cover parts of it. Regarding the "Define" and "Conduct" stages, all works directly acquire real gestures of different types, as representative examples.

Therefore, there is no real definition since only the supported gesture types are predefined in the "Define" stage and acquired in the "Conduct" stage. One exception is GestureWiz [85], which immediately defines a set of template gestures to be reproduced and tested in the "Conduct" stage. In contrast, GELICIT covers the "Define" stage by adopting a specification-based approach in which essential GES characteristics are first defined before actually conducting the GES. None of these environments are as parameterizable as GELICIT where a GES can be stored in a configuration file that can be reused at any time. These environments are often dedicated to a particular gesture type (e.g., hand), type of users (e.g., visually impaired persons) or device (e.g., Kinect), as opposed to any context of use in GELICIT. One notable exception is GestureWiz [85] which can be run in multiple contexts of use. GELICIT supports multiple contexts of use, distributed in time and space.

Regarding the "Classify" stage, most environments aimed at optimizing gesture recognition and/or classifying elicited gestures with annotations, tags, comments. In contrast, GELICIT materializes this stage by enabling any stakeholder to definer her own classification criteria (in the "Define" stage) and use them in this stage to precisely classify each gesture against these criteria on-demand. Indeed, authors rely on varying criteria to classify gestures.

Regarding the "Measure" stage, some software provide already appropriate means for computing different measures for assessing the quality of elicited gestures. For instance, AGATe [93] computes three rates for agreement, co-agreement, and disagreement while [88] prefers Fleiss's index. GELICIT aims at incorporating multiple measurement methods.

Regarding the "Discuss" stage, GestureWiz [85] is the only environment supporting this stage by offering a conflict checker, which identifies and solves ambiguities, confusions or multiple choices for gestures elicited. In contrast, GELICIT aims at supporting a discussion between stakeholders and participants via different means: re-discussing together a subset, rating, ranking.

Regarding the "Export" stage, examined environments either limit themselves to merely saving the gesture set in a file, sometime with a package or a library, or go to an entire procedure to seamlessly ensure the transition between the results obtained and the rest of the user interface development life cycle. For instance, GestureWiz [85] extracts the resulting gesture set and provides a library with callback procedure that is ready to run since it was already used during the "Conduct" stage when gestures were assigned to tasks. Similarly, GELICIT exports the consensus set with one recognizer selected among various candidates. Table 2 compares selected works against technical

Software	Devices and modalities	Input format	Output format	Recognizers
Quill	Pen, mouse	2D gestures	Raw data (x, y, t)	Rubine algorithm
Magic	Video-based capture	3D motion gestures	Raw data (x, y, z, t)	Own recognizer
CUBOD	Video-based capture	Full-body gestures	Raw data of body	Own recognizer
Gesture Control	Myo armband	3D mid-air hand gestures	(x, y, z, t) from gyroscope and accelerometer and (x, y, z, w, t) for orientation	Own recognizer
iGesture	Pen, mouse, TUIO devices, Wii remote	2D and 3D gestures	Raw data (x, y, z, t)	Rubine, Siger, Rubine3D
UsiGesture	Pen, mouse, trackpad, and hand tracking	2D multistroke gestures	Raw data (x, y, z, t)	Levenshtein recognizer
Kind-Dama	Stroke gestures	2D multistroke gestures	Raw data (x, y, t)	$N
Aramis	Camera	Multimodal gestures	Raw data (x, y, z, t)	Camera-based recognition
MDGG	Glove and camera	Unistroke single-hand gestures	SCXML specifications	Neural network multi-Layer perceptron
Proton++	Multi-touch device	Multi-touch gestures	Proton++ specifications	Own recognizer
Gesture Coder	Multi-touch gestures	Demonstrations and declarations		
GestIt	Multi-touch device	Multi-touch gestures	GestIT UI model	Own recognizer
ProGesture	Camera	Full-body gestures	State charts	Own recognizer
Gesture Script	Multi-touch device	Multi-touche gestures	Declarations and/or demonstrations	Internal recognizer
GestureWiz	multi-modal, multi-device for any device captured through video	Any stroke drawn in a HTML5 Canvas, any 2D/3D gesture via a video stream	Uni- or multi-stroke gestures from canvas, static images, animated GIF files, video streams	By the crowd or a Wizard of Oz selecting the correct match from the recorded set
Gesture Analyzer	Multi-pose gestures	Add poses within the same application	Hierarchical clustering and video annotation	
Kinect Analysis	Microsoft Kinect	Full-body gestures	Full body raw data	MS Kinect recognition engine
Buzzi	Smartphone	Unistroke single-touch gestures		
AGATe	Any device, any modality	CSV file	Agreement measures	
Gest Analytics	Video	Hand gestures	Hand poses	Own recognizer
Gelicit	2D in a HTML5 canvas by pen, mouse, finger, trackpad, stylus, 2D1/2 by object tracking through webcam, and 3D air+touch gestures	Some types of 2D, 2D1/2, and 3D gestures (Fig. 7)	Raw data captured by device: (x, y, z, w, t)	$1, $P, !FTL,...

Table 2. Comparison of related work against technical parameters.

parameters: which device and modalities are supported, what is the input format of gestures captured, how are they output, and which recognizers are included. Each environment supports its own set of devices and modalities acquired from two types of sources: by *direct acquisition*, when the gesture raw data are acquired directly from the device (which therefore leads to an output format mirroring these raw data) or by *indirect acquisition*, when gestures are captured via a camera (which therefore leads to an output format in animation or video). GestureWiz [85] adopts both techniques by capturing any gesture from any device and modality via stroke capture and/or video stream and transforms it into an appropriate format that can be used for gesture matching. GELICIT adheres to the first technique so that gesture characteristics other than those used for recognition can be also used for further stages.

Many gesture recognizers progressed with more advanced gesture recognition: the pioneer is Rubine recognizer [75] with its 13 features captured to recognize gestures, its 3D version [82], the Siger recognizer[1], the Levenshtein recognizer [22], and the members of the $-family: $1 [99] for unistroke gestures, $N [5] for multistroke gestures, $P [91] for point cloud recognition, and $Q [92] for optimized articulation-invariant gestures on low-end devices. In particular, GELICIT also captures gestures with different systems of coordinates, ranging from scalar 3D data to quaternions, which explains why x, y, z, w, t are stored, thanks to !FTL, a vector-based stroke recognizer [89].

3.2 Workflow Analysis

In order to better understand how a GES could be conducted and consequently a supporting software could be analyzed and designers, a focus group was organized to collect data from GES stakeholders according to the method based on questions/answers [2]. Due to their diversity, a stratified sampling initially identified 4 roles: a *user interface designer* responsible for designing the right gestures for the right tasks for the targeted users, a *developer* responsible for implementing the gesture-based user interface, an *experimenter* setting up and conducting a GES, and a *end user* of the expected gesture-based user interface. These roles will be employed in the remainder of the paper, without imposing any separation of duties. One person may assume different roles alternatively and one role can be ensured by different people concurrently. Roles are interchangeable.

A random selection on a mailing list of volunteers identified members for each role: designers (one designer working in a SME selling gesture-based applications on smartphones and tablets and one designer managing Moodle groups), developers (two independent experienced Web applications developers), experimenter (one psychologist from a full web agency), and end user representative (one member of a bank user group). All members already participated to a GES. A UI researcher was responsible for note taking, observation, and tape recording. A moderator, who is a co-author of this paper, supervised the process. This totals eight members, which is in line with [6, ..., 8], the recommended size of focus groups [44]. The researcher and the moderator were chosen for their experience gained through six past GES.

The moderator's script initially suggested three questions: what task/action do you consider be part of a software for supporting a GES? What is their frequency and their level of importance? What is the relationship between these tasks? The discussion was engaged on defining these tasks with a description and sample related questions. When some tasks/actions emerged with a sufficient level of distinctiveness and precision, the moderator asked the question "What is the task flow?". The focus group produced a scenario expressed as a textual narration of tasks with their flow. The researcher took notes on the following aspects: key points for each task/action and their position in the flow, hints, ideas, and insights suggested by participants for the future software, and follow-up questions. The outcome of this focus group is therefore a narrative scenario expressing

[1]See https://msdn.microsoft.com/en-us/library/aa480673.aspx

the tasks along with different possibilities to be investigated, which can be expressed as a word cloud depending on the most frequent responses issued by participants (Fig. 1).

A workflow analysis method [79] was applied to the final scenario in order to recursively decompose the workflow into stages based on process identification criteria and each process in turn into tasks based on task identification criteria and workflow patterns [79]. Each task was subject to task modelling [62] to provide a first idea of how each task should behave from a user's point of view. The workflow was modeled according to the [79] method in its corresponding environment and stored in YAWL format.

Fig. 1. Word Cloud resulting from the focus group.

3.3 Gelicit Requirements

Based on the material from the previous section and our own experience, the following requirements were stated.

(R1) Zero-install software. All members of the focus group concurred to conclude that a software requiring no install was of utmost importance to enable any stakeholder to contribute to a GES without any technical limitation: no setup of any software, no Java Runtime Environment, no plug-in, no add-on. The services provided by the software should be available over the network and accessed through any standard web browser that promote use by heterogeneous thin or thick client platforms (e.g., smartphone, tablet, laptop, and desktop).

(R2) Collaborative management of gesture sets with persistence. A gesture set is defined as a set of gesture classes, each class being represented by one or many gesture samples. Each gesture sample should be individually and independently managed according to the CRUDS pattern: Create a gesture, Read a gesture, Update a gesture, Delete a gesture, and Search for a gesture. Since multiple stakeholders may play different roles in the workflow, either simultaneously or asynchronously, access rights should be defined independently for collaboration. In addition, computing and storage resources should be pooled to serve multiple stakeholders using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to user's demand. In this way, all GES data should remain persistent, available and accessible for any follow-up, such as further analysis, within-subjects experiments [94], comparison with other setup, or re-design of an experiment. Finally, any stakeholder, especially designers, can unilaterally provision computing and storage capabilities for conducting a larger GES for instance without requiring any manual intervention.

	Same time (synchronous)	**Different times** (asynchronous)
Same place (collocated)	Face-face elicitation	Continuous elicitation
Different places (remote)	Remote elicitation	Distributed elicitation

Fig. 2. The four GELICIT configurations for distributing GES.

(R3) Support for distributed GES stages. Each interactive task identified in the workflow can be carried out according to the four configurations as represented on the time/space matrix in Fig. 2: *face-to-face elicitation* when all stakeholders are conducting the study in the same place and time (which is the case currently in all collocated studies [38]), *continuous elicitation* when stakeholders continue to proceed with the elicitation in the same place, but not with the same time conditions, *remote elicitation* when stakeholders are in different locations (e.g., the end user is not in the same location as the designer), and *distributed elicitation*.

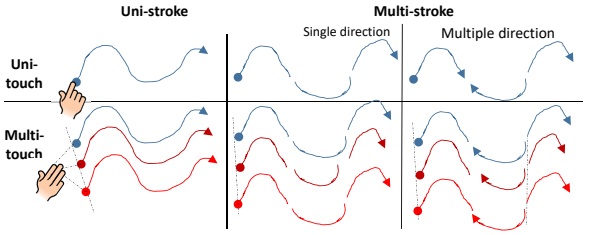

Fig. 3. GELICIT categories of gestures.

(R4) Flexible GES parameterization. To support different elicitation settings, in terms of time and space [3], in terms of contexts of use [15], each elicitation study should be parameterized into a configuration file that can be replayed. This configuration file should cover: the contexts of use, the list of participants, the list of questionnaires before and after the test, the analysis methods, and the list of referents subject to elicitation. In addition, four categories of gestures should be supported (Fig. 3): *unistroke* vs *multistroke* depending on the amount of strokes involved in the gesture, and *single touch or uni-touch* vs *multi-touch* depending on the amount of simultaneous gestures (e.g., three fingers at the same time). A multistroke gesture is issued in any direction.

(R5) Multiple measurement methods for each gesture set. From our review, the agreement rate as computed by AGATE [94] is definitely the most widely and frequently used measurement method. Since this rate evolved from a within-subjects version [93] to a between-subjects version [94], it was expected to accommodate for multiple measurements methods so as to give room for any future method. In addition, different measurement methods could be applied to different levels of granularity: to a single gesture sample (e.g., Rubine's features [75]), to a cluster of gestures (e.g., similarity measure [52]), to a class of gestures (e.g., general characteristics), and to a gesture set.

(R6) Computer-aided computation of measures. Beyond the computation of measurement methods, it is expected that other computations could be performed automatically, like Rubine's features for each gesture and other features required for gesture recognition [91].

(R7) Support for discussion towards consensus set. The agreement rates certainly give a first idea of to what extent an elicited gesture could become a consensus gesture for which a recommendation may be stated. But there are other possibilities to reach a consensus, like through a A/B testing. It is expected to launch a discussion with the participants, particularly when the designer wants to show a small set of elicited gestures to participants and to finally ask them their rating and/or ranking for this subset before issuing a recommendation.

(R8) Continuity with the rest of the user interface development life cycle. Once stakeholders agreed upon a consensus set, it should be exported along with an appropriate gesture recognizer, such as any member of the $-family of recognizers [92], so as to incorporate it in the rest of the UI development life cycle.

Stage	Define	Conduct	Classify	Measure	Discuss	Export
R1. Zero-install	●	●	◑	●	◔	◔
R2. Collaborative management	●	●	◔	◔	◕	◔
R3. Support for distributed GES	◑	●	◔	◔	●	◔
R4. Flexible parameterization	◑	◑	◑	◔	◔	◔
R5. Multiple measurement methods	◔	○	○	◔	◑	○
R6. Computer-aided measurement	○	○	○	◕	◔	○
R7. Support for discussion	○	○	○	○	◑	○
R8. Continuity	◔	○	○	○	○	◔

Table 3. Level of requirement support for stages: ○=none required, ◔=minimal support asked, ◑=partial support asked, ◕=significant support asked, ●=maximal support asked.

Table 3 qualitatively represents to what extent each stage should be supported by which requirement, ranging from no support to maximal support. **(R2)** was reported as the most important requirement from the focus group, with a maximal support for the "Define" and "Condcuct" stages, thus meaning that stakeholders involved in these stages should be enabled to manage data corresponding to a definition of a GES configuration and its conducting as much as possible in a collaborative manner. Every stakeholder should be enabled to play the role assigned in the GES, but could alternatively switch to another role if needed. **(R3)** was considered as the second most important requirement in order to be released form physical constraints for conducting a GES in the same time and the same place.

(R1) was a natural consequence of **(R2)** and **(R3)** since most stakeholders asked for a software that was ready-to-use and straightforward. Otherwise, they reported that the time gained by distributing a GES in time and space could be lost y simply installing, configuring, and calibrating the software. **(R4)** was considered fourth, again as a consequence of the previous ones: if a GES should be defined and conducted collaboratively, this would make sense only if the parameters for regulating a GES could be modified flexibly by authorized and competent persons.

Then came **(R6)** because the storage of gestures elicited in a database should enable the immediate calculation of measures needed to estimate the gestures elicited. **(R7)** came afterwards because it was estimated that computing measures was a preliminary step for establishing a common ground for discussion. **(R5)** was then evoked, but not considered as that important for the moment: people preferred to rely on a short amount of automated measures to leave the space for future measure later in the future.

(R8) came last, not because ensuring the continuity with the development life cycle was not assessed as unimportant, but simply because gestures resulting from the GES was the main goal to achieve. Manually handling the consensus set was reported to be always manageable in any circumstances and was assessed as optional.

4 DESIGN AND IMPLEMENTATION OF GELICIT

To fulfill the eight aforementioned requirements, this section defines a metamodel for characterizing gestures, a guiding method based on this metamodel, a software architecture, and a system walkthrough for illustration.

4.1 GELICIT Metamodel

In order to fulfill **(R2)** and **(R6)**, this section defines a metamodel for characterizing gestures for a GES (Fig. 4: as such this metamodel is not primarily aimed at expressing gestures for developing gesture-based interaction, although some parts are common such as capturing gestures.

The topmost concept of interest in the metamodel (Fig. 4 is a *gesture vocabulary*, hereby referred to as a set of *gesture sets*, which is itself further decomposed into classes of a same gesture. For instance, a gesture vocabulary encompasses all gestures to be shipped with a gesture UI in a certain domain of application, like sign language. A *gesture class* gathers all *gesture samples* captured for this class in any *context of use*, where a context of use may instantiate to a particular user profile performing a set of tasks/actions with a platform/device in a certain environment. A gesture sample is said to be *user-independent*, respectively *user-dependent* when the user model is left undefined, respectively defined. Each gesture set is optionally further recursively structured into clusters, clusters being defined by any particular relationship like similarity analysis [52] or category analysis. For instance, a same gesture set could consist of different alphabets of symbols, each alphabet symbol being grouped into a cluster for separation of concerns (e.g., uppercase letters, lowercase letters, digits, punctuation symbols, command gestures). Any gesture set can be created by a registered user who specifies access rights to any other user, including participants. Gesture dimensionality, modeled by `GestType`, covers the configurations of Fig. 3.

A gesture sample is composed of *strokes* that are binding *points*. Indeed, each gesture sample is captured by an input device as raw data as $G = \{p_i{=}(x_i, y_i, z_i, w_i, t_i)\}, i \in \overline{n}{=}\{1, ..., n\}$ where x_i, y_i, z_i, w_i are the 3/4D coordinates of each gesture point, t_i is the time stamp. When a gesture is captured on an interactive surface, like in pen-based computing, z_i and w_i are all constant equal to 0. 4D gestures, such as for ring devices, are captured by quaternions involving both the z_i and the w_i. Sometimes t_i is optional when the gesture shape matters more than its motion or t_i is fed with a continuous identifier ID_i. When a gesture is captured by an input device with more parameters, like 6DOF (e.g., optical 3D trackers [2]), pressure (e.g., the Anoto device [3]), velocity, acceleration, etc., G is then characterized by a n-dimensional vector of properties: $G = \{p_i{=}(x_i, y_i, z_i, p_{i1}, p_{i2}, ..., p_{ij}, ..., p_{im}, t_i)\}$ where p_{ij} denotes the j^{th} property of the i^{th} point. These properties could be either captured (e.g., 3D coordinates and time stamp) or calculated (e.g., the velocity is the first derivative of position with respect to time, etc.).

Each gesture class can be captured through one or many gesture samples. Each sample could be declared *position-invariant*, respectively *scale-invariant* and *rotation-invariant*, if this sample should be recognized in gesture recognition in any position, respectively after any translation and rotation. For instance, a gesture representing a human body can be captured once and interpreted in the same way in any position, scale, and orientation. If this is not the case, GELICIT offers to the designer the possibility to manually (by clicking on the corresponding icons in the gesture area) or automatically create more samples with variations of position, scale, and rotation.

To better characterize various dimensions of articulation invariance [92], we define *isometricity* as the property of a gesture sample or set to hold a set of n-equally distanced points: $\forall i \in \overline{n-1}, d(p_i, p_{i+1}) = constant$ e.g., $\|p_{i+1} - p_i\| = \frac{1}{n-1}\sum_{i=1}^{n}\|p_{i+1} - pi\|$. We define *isochronicity* as the property of a gesture sample or set to hold a set of n equally-timestamped points, i.e., $\forall i \in \overline{n-1}, \|t_i - t_{i+1}\| = constant$, e.g., $\|t_{i+1} - t_i\| = \frac{1}{n-1}\sum_{i=1}^{n}\|(t_{i+1} - t_i)\|$. We also define *isoparametrization* as the property of two or more gestures samples or sets to contain the same amount of points, i.e. $\forall G{=}\{p_i\}_{i=1,...n}, H{=}\{q_j\}_{j=1,...,m} : m{=}n$. Two gesture sets can be isoparametrized whether they are isometric, isochronic, or not.

[2] http://ps-tech.redblom.com/optical-trackers/PSTracking
[3] http://www.anoto.com

Fig. 4. Metamodel of Gelicit.

To specifically address **(R3-R5)**, the metamodel specifies further concepts: any *participant* involved in an experiment is characterized by an identifier (used for later anonymization and data privacy), by her general demographic data (e.g., gender, age), by a background (e.g., student, executive, employee, independent, retired, unemployed, or other), and by levels of experience on various devices on a 5-point Likert scale [50] (1=very rarely to 7=very frequently). Each participant may issue one or many elicited gestures, a sub-class of `GestureSample`, which represents any suggested gesture for a referent in particular context of use (itself specified by its user profile, platform/device, and environment). For each elicited gesture, GELICIT also records the goodness of fit [33] (i.e., the participant's subjective assessment, expressed as a rating between 1 and 10, of their confidence about how well the proposed gestures fit the referent), the laterality (i.e., left produced, right produced, bilaterally produced), and the thinking time required to decide a gesture.

A *referent* is defined as a materialization of a task or action to be executed by an end user in a certain context of use [97–99]. A referent aims at prompting a participant to elicit a gesture by providing a representation of the context through three resources (e.g., an image, an screen shot, a picture, an animation, a video or any graphical resource stored in a file): a resource representing the state of the context of use before executing the task, an optional resource representing the state of the context of use during execution, and a resource representing the state after this execution.

By providing the participant with these resources, enough contextual information will be conveyed to suggest a gesture. Of course, each referent is only a particular materialization and several materializations could exist, even for the same context of use. Referents are gathered in a referent set and each referent correspond to a task, expressed as a simple sentence in action+object paradigm: the name of an action is to be performed on an object, such as "open a file", "print a document", "format a paragraph of text with justified margins".

Similarly, tasks are gathered in a task set, which can be declared as standard. None of the software reviewed in Section 2.3 supports the definition and usage of referents for tasks according to a specification-based approach like that. Each experiment can be defined along with a series of questionnaires that the participant will be required to fulfill either in pre-test or in post-test conditions. Each questionnaire consists of a series of closed questions with predefined answers. Any elicited gesture can be characterized by one or many classification criteria, which are defined by a name, a description, a series of possible values for a simple or a multiple choice.

The results of each experiment can be subject to one or many measurement methods in order to fulfill requirements **(R5-R6)**. These measures are computed via the corresponding methods, which can be extended. For example, the 13 Rubine features [75] are also automatically computed and stored for each gesture sample in order to fulfill requirement **(R6)**, thus enabling the experimenter and the designer to produce statistics or to later feed a gesture recognizer.

4.2 GELICIT as a Guiding Method

Fig. 5 graphically depicts how the six GES stages (Section 2.1) could be structured to shape a GES. The full worfklow consists in enacting all stages in a linear sequence: once a GES is defined, it can be conducted to elicit gestures that will then be subject to classification, measurement, discussion, and export. Nonetheless, various organizations and their stakeholders do not always apply all these stages sequentially. Any stage, apart from the "Conduct" stage, can be skipped. When the "Define" stage is skipped, any gesture can be captured without any restriction. The subsequent stages can be equally skipped: "Classify", "Measure", "Discuss", and "Export" are optional and repeatable on demand. Similarly, once a GES is defined, any stage can be stopped and resumed. The arrow of each stage in Fig. 5 covers some screenshots manipulating the corresponding classes with the same color as in the metamodel (Fig. 4). The turquoise-colored stage "Define" manipulates data belonging to turquoise classes in the metamodel, such as vocabulary, gesture set, class, experiment.

Fig. 5. Six-stage GES Workflow in Gelicit.

4.3 GELICIT as a Supporting Software

This section motivates the software architecture selected to meet **(R1)-(R4)** requirements and justifies why this decision was made. The first subsection covers the backbone structure of GELICIT by introducing the different frameworks used. The second subsection completes the overview by describing the various third-party libraries involved in specific parts of GELICIT, so as to make it a cloud computing software.

4.3.1 MEAN Stack. After studying 4 potential development frameworks to address the requirement, it was decided that the software architecture is structured according to the "MEAN" (MongoDB-Express-Angular-Node) stack architecture [35], a free and open-source JavaScript software stack for building dynamic web sites. The MEAN software architecture was selected for the following reasons [35]: (1) all MEAN stack components are open-source and free to use, (2) MEAN is straightforward to use for both back- and front-end as opposed to using different languages and technologies for front and back ends; (3) only one language for both server-side and client-side execution is developed; (4) it enables working on the front-end and back-end of the application using common languages (JavaScript and TypeScript, respectively), as well as to stay flexible with the diverse middleware and libraries that we want to use for GES, such as Passport.js, Jwt.io.

4.3.2 Database and Cloud Deployment. MongoDB is a database management system offering a significant flexibility in terms of data structure, accommodating various records, even at run-time. In absence of conversion/mapping of application objects to database objects, this mechanism enables implementing and deploying the database in a cloud-ready format that is compatible with **(R1)**. The middleware *Mongoose* for MongoDB was added since it offers out of the box built-in type casting, validation, query building, and business logic hooks. Coded in Node.js and released in 2011, it has since become the go-to utility library when developing an application connected to a MongoDB database. We also made use of small, yet efficient, libraries such as *JWT.io* for securing user authentication and routing on the platform; *cursometer.js* for capturing speed when drawing a gesture; *tracking.js* for tracking the color used in the webcam-based gesture capture; *chart.js* for producing interactive graphics for GES results; and *farbtastic.js* for the color picker widget used in gesture drawing. We also relied on *Bootstrap*, the dynamic cross-platform CSS library. With its responsive grid system and numerous pre-built and embarked HTML components, it quickly creates the base structure of our web-pages while remaining extremely customizable. Finally, the JQuery cross-platform library was used for its HTML document-traversal, animation.

Once these functionalities were settled out and implemented, we deployed the whole GELICIT application online on Heroku (a cloud application platform, http://www.heroku.com) through their integrated command line interface Heroku CLI. Heroku is one of the oldest, most renowned, monitored and stable frameworks available while offering a free solution for emerging applications, as well as being fully compatible with remote *GitHub* repositories. GELICIT is consequently deployed as a Heroku cloud computing software with zero install satisfied **(R1,R2)**, only an account is needed: http://gestman.herokuapp.com/).

4.3.3 Adding a New Stroke-Gesture Recognizer. GELICIT is today shipped with three stroke-gesture recognizers: $1 [5], $P [91], and FTL [89]. Any new gesture recognizer, such as $Q [92], can be added by modifying only a few lines of codes, in addition to providing the JavaScript source code of the recognizer itself. All gesture recognizers can be triggered in GELICIT via the "Drawing area" page. It is possible to choose in the canvas settings which recognizer(s) are used by default, and then apply it by pressing the "recognize button" from the toolbar. The recognition process works as follows: GELICIT trains the recognizer by submitting all the samples of the gesture set to the recognizer according to a user-dependent fashion, the recognizer attempts to link any gesture

drawn in the canvas with one of the samples of the set based on its own algorithm. The method to add a new gesture recognizer in Gelicit consists of performing the following operations:

(1) Add a JavaScript file containing the source code of the new gesture recognizer in the project, presenting the same data structures than the existing recognizers (Fig. 4).
(2) Add a radio button with the name of the recognizer in the settings of the canvas component.
(3) Adapt the if...else... instructions in the *get_points* function and *recognize functions* in *recognizer.js* file (Listing 1).

Listing 1. If...else... instructions of the Recognize function

```
function recognize(gesture, gesture_classes) {
  if(options.getRecognizer() == "$1") {
    recognizer = new DollarOneRecognizer();}
  else if(options.getRecognizer() == "$P") {
    recognizer = new DollarPRecognizer();}
  else if(options.getRecognizer() == "$Q" || options.getRecognizer() == "$
    recognizer = new DollarQRecognizer();
  }  ...}
```

4.4 Gelicit System Walkthrough

This section illustrates the most significant portions of the six stages supported by Gelicit. The first sub-section starts by illustrating the "Conduct" stage, which is the cornerstone of the whole workflow. The other stages are then illustrated as they are optional according to the general workflow (Fig. 5).

4.4.1 The "Conduct" Stage. This stage consists in creating a new gesture set and populating it by several users: any logged in user fills the various fields for creating the intended gesture set (Fig. 6a): a name, a description, and the access rights. This mechanism can be specified in three ways: *public* when the gesture set can be accessed by any user accessing Gelicit without any registration (this enables any person to access a gesture set without any restriction and to submit modifications that are then monitored, accepted or rejected by the owner), *Application* when the gesture set can be modified only by Gelicit registered users, and *private* when the gesture set can be modified only by Gelicit members who have been invited to join a gesture set by the owner.

Once submitted, the application verifies the data. If everything is compliant, an empty gesture set is created and made available in the main personal page of the user. Then, any user can request joining the gesture set freely or privately with the correct password if the set is protected. Inside a gesture set, any collaborating user is able to select or create a gesture class (samples are assumed to refer to the same gesture type) and/or cluster (samples or classes are related to each other with some relationship), and then draw and save gesture samples (Fig. 6).

Fig. 7 reproduces a typical situation where three categories of gestures are acquired in Gelicit: 2D gestures captured in a HTML5 canvas (a) by a pen, a mouse, by one or many fingers, via a trackpad, a trackball, a stylus, or a penclic, a combined mouse+pen; (b) 2D1/2 gestures by object tracking through a webcam whose settings are customized to the object color to follow; and 3D touch (c) + air (d) gestures captured via a 3DTouchPad.

These two last categories result into a path captured and visualized in 3D (e). The main canvas contains the main menu for managing any sample: clear, recognize by the default recognizer, apply three basic geometric transformation (a translation, a rotation, a scaling down, a scaling up), and setting defining default values for their parameters. For instance, a dilation can be defined by default

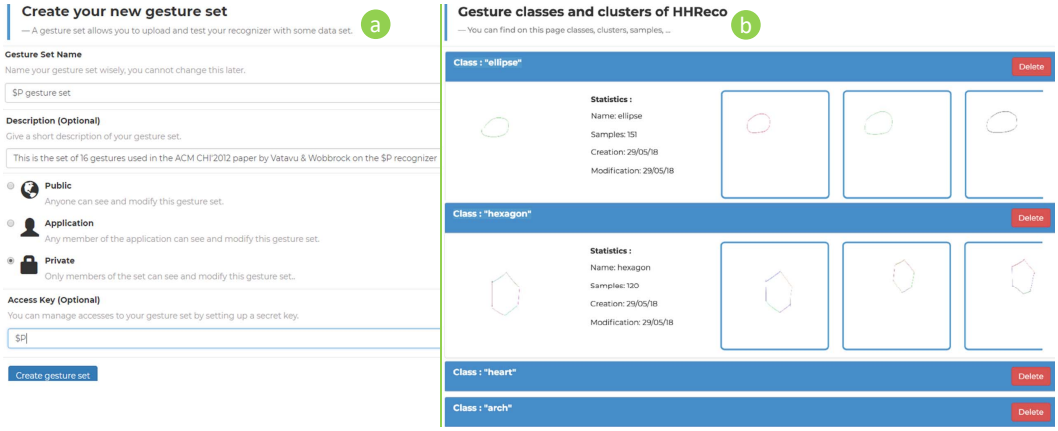

Fig. 6. Creation of a new gesture set: (a) definition, (b) classes and clusters.

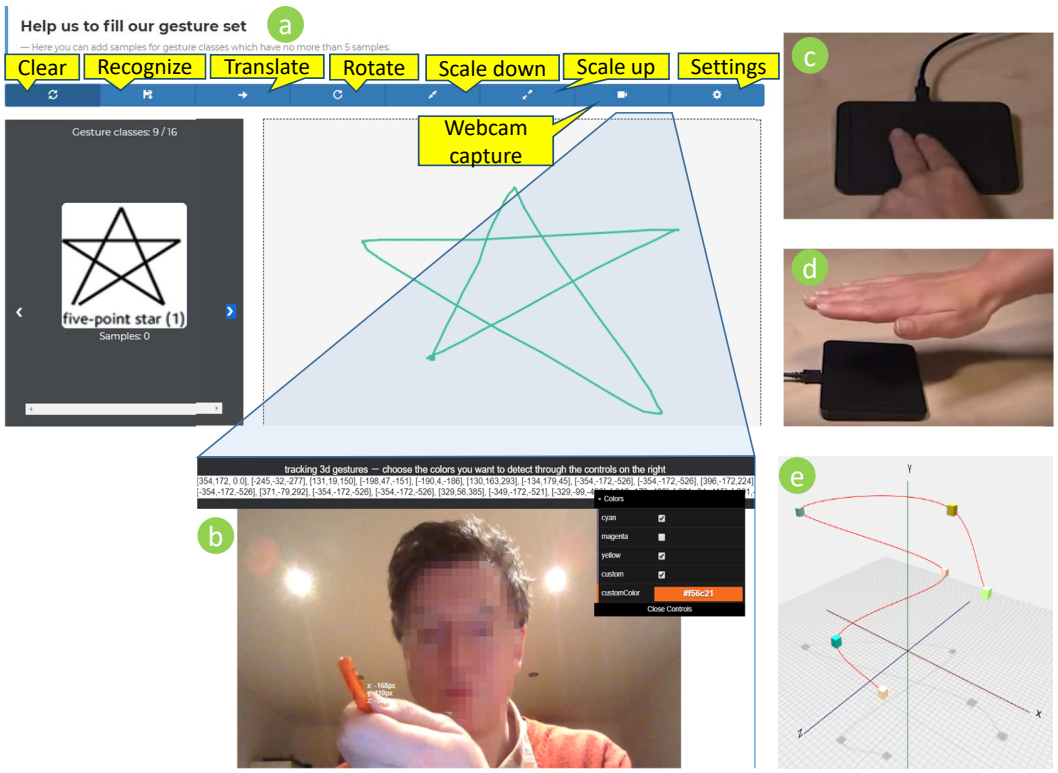

Fig. 7. Gesture acquisition: (a) main area, (b) 2D1/2 object tracking, (c) touch + (d) air gesture, (e) 3D gesture.

via a constant scaling up with the same aspect ratio. A log file keeps all basic activities performed by all its users in a history that can be revisited.

4.4.2 The "Define" Stage. To create a GES, any logged in user accesses the page for defining the main parameters and provide the various information requested by the form (Fig. 8): the pre/post-test forms, the set of tasks, the set of referents, the list of participants, and the classification criteria.

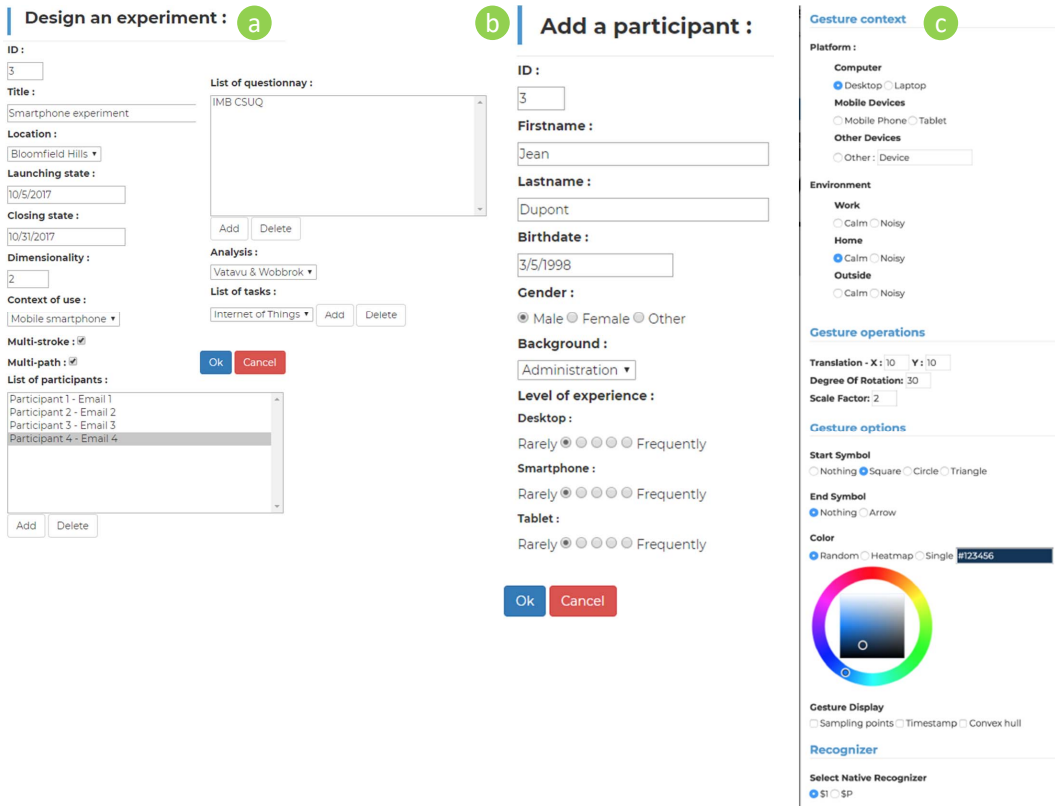

Fig. 8. Gelicit: Defining a new experiment (a), Adding a participant (b), Defining the settings for a context (c).

Once submitted, the application verifies the data conformity. If everything is compliant, a new GES is created and made available in the personal page of the user. Otherwise, an error message requires the user to fix the data. Once displayed in the list, the owner can retrieve, update, edit, search, and delete any experiment. This leads the user to a new page where new task/referent populate the experiment. When the various tasks included in the set are complete, it can be validated and made available to the participants. The experiment is then listed as ”ready” to all the participating users, who can access it through their personal page. A URL is also automatically created that directly points out to the GES which can be copied/pasted to invite other participants via external means, such as distribution lists or social networks.

Fig. 9 depicts how to add a referent corresponding to the task "Indicate a direction", in which a map is presented with a point of interest before and a second point of interest after, representing the target to reach. No resource should be displayed while the action is being carried out. In this activity, a logged in user can select one of its available GES through her personal page. The application then verifies that the starting date of the experiment has passed but that the ending date has not come yet, and also verifies that the user did not already participate to the experiment. If these three conditions are satisfied, the user is then redirected to the elicitation page where she is shown a series of referents related to several tasks, grouped in task sets, and is asked to draw a number of samples. If the user does not want or is unable to provide any gesture sample, she is free to skip the task or even the current task set. Once all the tasks have been reviewed, the user is redirected to its personal page and the form is submitted to the application which saves all the new samples and marks the participant as having participated in the experiment.

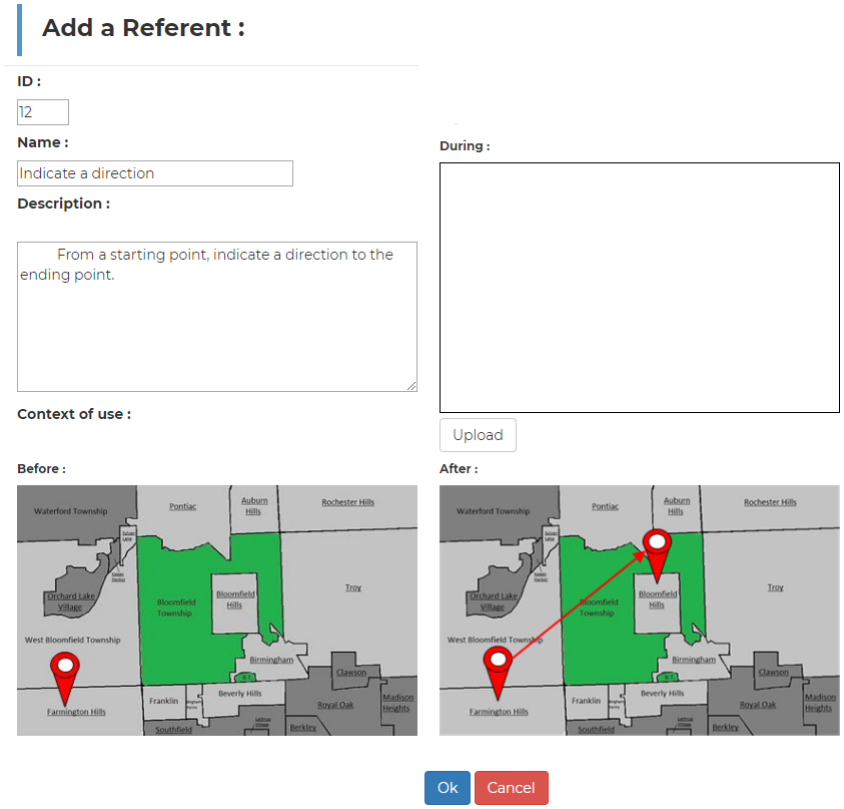

Fig. 9. Adding a new referent in a GES.

4.4.3 The "Classify" Stage. This stage reviews each gesture elicited for each referent by participant (Fig. 10): at any time, a designer or an experimenter browses the collection of gestures elicited by participants, one by one. Then, for each referent, all elicited gestures are displayed and subject to a manual classification. This step is optional for each elicited gesture. For instance, the first gesture elicited for "Turn TV on" (Fig. 10) received a score of 100, thus indicating that this sample is the most appreciated by the participant and will therefore be subject to classification to retain it, whereas the two others will not. This classification subsumes three activities:

(1) *Category assignment*: at any time, a gesture category can be created with an identifier and a name to be assigned to the currently selected gesture. Any already created category is selected by its identifier.
(2) *Descriptive labelling*: each collected gesture is classified and guided by a procedure introduced by Nielsen et al. [71] and used in subsequent GES, such as [81]. Each gesture is described according to a simple sentence expressing their physical motions and actions rather than the semantic meaning.
(3) *Vocabulary classification*: the vocabulary defined through classification criteria is then used to systematically classify the elicited gesture with keywords corresponding to the possible values of each criteria.

4.4.4 The "Measure" Stage. This stage comprises two types of measures: measures related to a gesture sample and those related to a gesture set. Fig. 11 reproduces some individual measures, like production time and Rubine's features [75] for a gesture sample, here a circle. The final results

Fig. 10. Gesture classification.

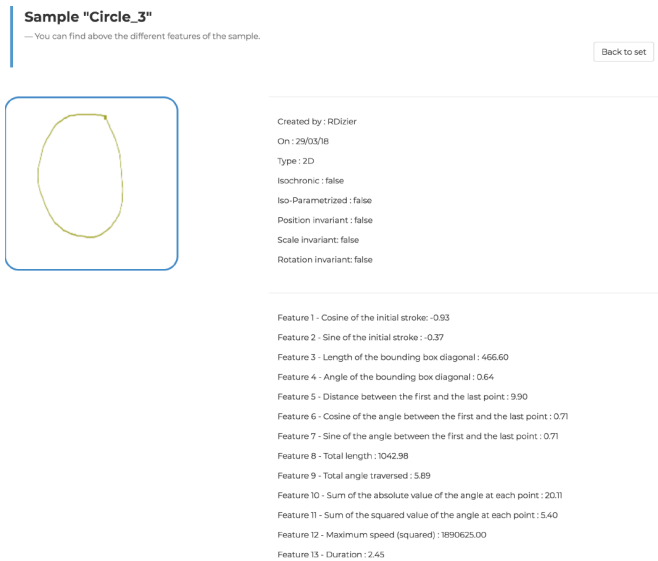

Fig. 11. Features automatically computed for a gesture sample.

belonging to the second type are delivered to the stakeholder according to a graphics depicting the distribution of agreement rates (Fig. 12) implemented in CanvasJs V2.0.2[4], which can be interactively browsed for further details. Similarly to Listing 1, GELICIT may incorporate other functions for computing additional measures. While this method appeared quite efficient at first glance, it later became clear than a more automated way involving direct file import was possible, which is the case currently.

[4]https://canvasjs.com/

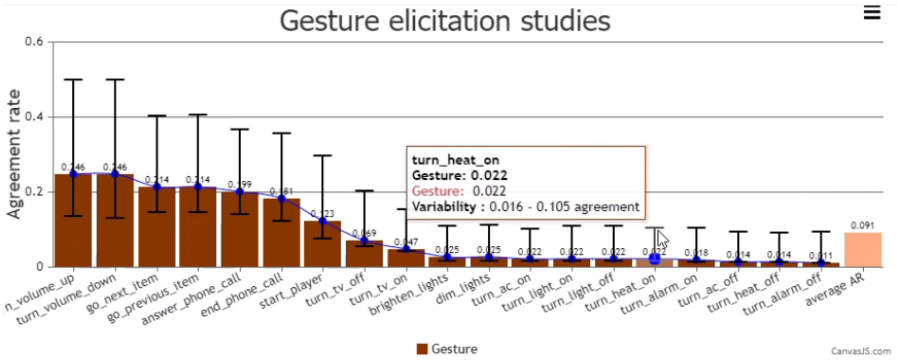

Table of values

Gesture	Agreement rate	CI 95%
turn_volume_up	0.246	[0.134, 0.500]
turn_volume_down	0.246	[0.130, 0.500]
go_next_item	0.214	[0.145, 0.402]
go_previous_item	0.214	[0.145, 0.406]
answer_phone_call	0.199	[0.141, 0.366]
end_phone_call	0.181	[0.123, 0.355]
start_player	0.123	[0.076, 0.297]
turn_tv_off	0.069	[0.054, 0.203]
turn_tv_on	0.047	[0.041, 0.152]
brighten_lights	0.025	[0.016, 0.109]
dim_lights	0.025	[0.016, 0.112]
turn_ac_on	0.022	[0.016, 0.101]
turn_light_on	0.022	[0.016, 0.109]
turn_light_off	0.022	[0.016, 0.109]
turn_heat_on	0.022	[0.016, 0.105]
turn_alarm_on	0.018	[0.013, 0.105]
turn_ac_off	0.014	[0.013, 0.094]
turn_heat_off	0.014	[0.013, 0.091]
turn_alarm_off	0.011	[0.010, 0.094]

---

average AR = ,091

Fig. 12. Results of an elicitation study in GELICIT.

4.4.5 The "Discuss" Stage. After selecting the candidate gestures that will be further discussed, e.g., those benefiting from the highest agreement scores, GELICIT considers three mechanisms:

(1) **Rating**: the subset of elicited gestures feeds a questionnaire asking each participant to review previously elicited gestures (by her or any other participant) and to rate each candidate on a 5-point Likert scale [50]: how would you rate this gesture? (1=very low to 5=very high, with 3=medium as the neutral value).
(2) **Ranking**: the subset of elicited gestures feeds a questionnaire where the candidates are presented simultaneously to the participant who needs to sort them by decreasing order of ranking, from the most to the least preferred.
(3) **A/B testing**: two gesture candidates are randomly selected from the subset and presented together to the participant. By clicking on any gesture, the participant selects the gesture having the higher value for the acceptability in the pair. For every positive selection, one point is given to the global gesture score. If the participant is undecided, the possibility is given to click on the "It's a draw" button (Fig. 17), no point is assigned. Two new gestures are further presented from the subset. To prevent responses biases, the gestures are displayed randomly and each pair of gestures is verified as being unique to avoid any duplicate. This is repeated until all gesture candidates have entered in at least one A/B testing. Note that two variants of this mechanism exists: in the "Keep-the-Best", the preferred gesture would be included

Fig. 13. Gesture discussion.

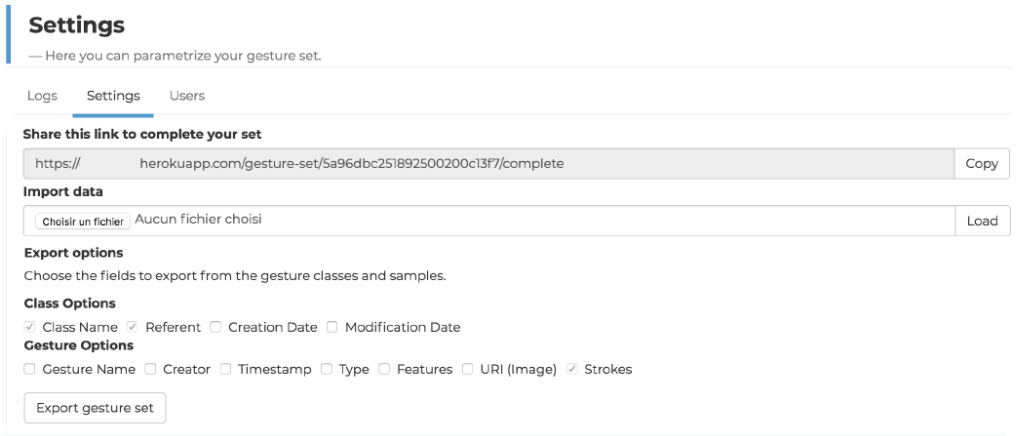

Fig. 14. Settings for exporting a gesture set

.

in the immediate next round and compared against a new one while, in the ”Tournament” mechanism, the selected gesture would be added to a winners pool and would resurface in future comparisons when all other gestures for the same referent had been compared.

4.4.6 The ”Export” Stage. Finally, the owner or the participants of a gesture set are able to export one, several, or all the gesture classes and samples with some or all their attributes by specifying corresponding settings (Fig. 14). These two last steps can be repeated any number of time and at any given moment, independently of the other users of the set. Only the owner of the gesture set can delete it. An archive of an entire gesture set can also be prepared and saved.

5 EXEMPLIFYING GELICIT ON A CASE STUDY

This section exemplifies how GELICIT has been exploited to support a GES distributed in time and space. A GES, as any other experiment, is typically reported according to a Method-Results-Discussion scheme. Since this section aims at illustrating the GELICIT workflow, it is instead structured according to the six stages (Fig. 5).

We opted for the following GES: to explore user-defined gestures for triggering Internet of Things (IoT) tasks on a smartphone. The goal of the study was to investigate to what extent a smartphone, considered as a frequently used and familiar device, could be used as aan alternate device for triggering tasks on a IoT-enabled device, like home appliances (e.g., a smart TV, a heating system, an air-conditioning system). A smartphone is often recognized as a mobile phone shipped with a music player and an Internet browser that can be operated via multi-touch gestures that can be customized, and contains a lot of sophisticated sensors useful for web browsing, GPS navigation, music player, mobile gaming, voice and data communication, and multimedia consumption. Some GES have already addressed the usage of a smartphone as a gesture sensor, namely for motion gestures on smartphones [76], for communicating between smartphones, tabletops, as public displays [42], for investigating cultural differences in smartphone usage [58]. We were not aware of any similar GES to collect users' preferences for using a smartphone for home appliances other than for a smart TV, a device already covered [25, 95].

5.1 The "Define" Stage

This study concerns 2D stroke gestures defined as any movement of one or many fingers of one or two hands on the flat surface delineated by a smartphone screen that causes a detectable change. In this way, the GES restricts the elicitation to contact-based gestures on the surface, not around the device or on the back of the device) or in the air [12]. Gestures are consequently produced on the screen surface and usually mimic interactions common on touchscreen devices, such as taps, pinch gestures, or directional swipes. Here are some admissible examples:

(1) Touch the screen once, twice, or multiple times in a short amount of time.
(2) Swipe on the screen in one, two, or more directions in a bounded amount of time. A single-direction swipe is considered as a flick, while a two-direction swipe is considered as a mark.
(3) Touch the screen with two fingers coming from both hands and converge to a central point.
(4) Draw a symbol with one finger on the screen.
(5) Any combination of the above.

A GES is therefore defined as a within-subject design for 2D stroke gestures. A set of 19 actions representing common tasks to be executed in a IoT environment with a smart phone is defined: (1-2) turn the TV on/off, (3) start player, (4) turn up the volume, (5) turn down the volume, (6) go to the next item in a list, (7) go to the previous item in the list, (8-9) turn Air Conditioning on/off, (10-11) turn lights on/off, (12) brighten lights, (13) dim lights, (14-15) turn heating system on/off, (16-17) turn alarm on/off, (18) answer phone call, and (19) end phone call. For each task, a referent is created or selected from an existing set. Since these 19 tasks are widely used in several GES, such as [31, 95], the corresponding referent set can be declared as "standard" or "reference", which will therefore remain unmodifiable to preserve consistency and reusability. Referents are randomized using a pseudo random number generator and are presented one by one. Note that GELICIT authorizes to enter zero referent (e.g., to leave the participant free from not eliciting anything), one referent (e.g., for a bijection referent-elicited gesture), or multiple referents (e.g., when a participant wishes several suggestions or add another elicited gesture when unhappy with the initial trial). In this case, one and only one gesture can be elicited.

The following elements are also included in the GES definition: a context of use (any user carrying out the 19 tasks on a smartphone in a home environment), a list of classification criteria (i.e., thinking time, goodness of fit, and gesture nature), a measure (i.e., agreement rate [93]), a pre-test questionnaire (a creativity score) and a post-test questionnaire (IBM PSSUQ).

The Thinking-Time measures the time, in seconds, needed by participants to elicit a gesture for a given referent. The Goodness-of-Fit represents participants' subjective assessment, as a rating between 1 and 10, of their confidence about how well the proposed gestures fit the referents. The Gesture nature specifies which nature is underlying the gesture elicited [98]: *physical* when the gesture is produced as if it is physically acting on a real object (e.g., pointing to an object), *symbolic* when the gesture represents a meaningful symbol (e.g., thumbs up/down to indicate acceptation/rejection, drawing a "D" for duplicate), *metaphoric* when the gesture expresses an action based on a metaphor (e.g., approaching the two hands to mimic a compression or a smash), or *abstract* when an arbitrary gesture is produced (e.g., a double tap to deselect an object).

The Creativity score is evaluated using an on-line [5] creativity test returning a score between 0 and 100 (higher scores denote more creativity) computed from answers to a set of questions which cover several factors: *abstraction* (of concepts from ideas), *connection* (between things without an apparent link), *perspective* (shift in terms of space, time, and other people), *curiosity* (to change and improve things accepted as the norm), *boldness* (to push boundaries beyond accepted conventions), *paradox* (the ability to accept and work with contradictory concepts), *complexity* (the ability to operate with a large quantity of information), and *persistence* (to derive stronger solutions).

The IBM PSSUQ (Post-Study System Usability Questionnaire) [46] enables participants to express their level of satisfaction with the usability of the setup and the testing process. This 16-question questionnaire has been empirically validated with a large number of participants on a significant set of stimuli [48], it is widely applicable for any interactive system [47], and it benefits from a proved $\alpha = 0.89$ reliability coefficient between its results and the perceived system usability [46]. Each closed question is measured using a 7-point Likert scale [50] (1=strongly disagree, 2=largely disagree, 3=disagree, 4=neutral, 5=agree, 6=largely agree, 7=strongly agree) and the following measures are computed: system usefulness (SysUse: Items 1-8), quality of the information (InfoQual: Items 9-15), quality of the interaction (InterQual: Items 16-18), and system quality (Overall: Item 19). Twenty-four (24) participants (10 female and 14 male) were randomly selected from a mailing list of volunteers working in different services of an organization: finance, accounting, human resource management, real estate, and trainees. They were aged between 16 and 53 years old ($M = 25.4$, $SD = 6.2$ years). The vast majority of participants ($23/24 = 95.8\%$) was right-handed, only one participant was left-handed and no participant was ambidextrous. All participants owned smartphones and, thus, were accustomed with touch and gesture input.

All this information is maintained in a GES configuration file with an identifier, a representative title, a launching date serving as the initial date for the GES, a closing date representing the deadline for conducting the GES.

5.2 The "Conduct" Stage

Participants received by e-mail an automatically generated URL to launch the previously defined experiment on their own smartphone. Once connected, each participant enters socio-demographic data (e.g., age, gender, handedness) and answers a series of questions about their use of technologies (based on a 7-point Likert scale [50] ranging from 1=strongly disagree to 7=strongly agree). These data are part of the profile stored in the user account. The participants can update or skip this part at any time.

[5] http://www.testmycreativity.com/

Pre-test questionnaires are filled in, if any. Each participant is presented with *referents*, i.e., various functions to control in a IoT environment for which they were asked to think of desirable gesture proposal to execute those referents, i.e., gestures that fit referents well, are acceptable, easy to learn and remember [33]. Participants were given total freedom with respect to the gesture types they elicited provided that is is compliant with the definition. The order of the referents is randomized per participant. Post-test questionnaires are filled in.

5.3 The "Classify" Stage

Once the experiment is closed, the designer reviews and classifies the results by browsing gestures elicited for each referent (Fig. 10 as a browsing example). In total, 24 participants × 19 gestures = 456 elicited gestures, each being formalized by descriptive labeling as recommended by [18], thus resulting into 154 unique gestures, e.g.:

1. Swipe up= TOUCH ON Surface with INDEX, THEN MOVE Up ON Surface with INDEX, THEN RELEASE INDEX
2. Swipe down = TOUCH ON Surface with INDEX, THEN MOVE Down ON Surface with INDEX, THEN RELEASE INDEX
3. Tap = TOUCH ON Surface with FINGER, THEN RELEASE FINGER

19. Swipe left on center = Swipe left ON Surface Center
20. Swipe right on center = Swipe right ON Surface Center
21 Touch and hold bottom right = TOUCH ON Surface Bottom Right with INDEX THEN HOLD FOR 3 Seconds THEN RELEASE INDEX
31. Two fingers drag up = TOUCH ON Surface with INDEX AND MIDDLE FINGER, THEN MOVE Up ON Surface with INDEX AND MIDDLE, THEN RELEASE INDEX AND MIDDLE FINGER
32. Two fingers drag down = TOUCH ON Surface with INDEX AND MIDDLE FINGER, THEN MOVE Down ON Surface with INDEX AND MIDDLE, THEN RELEASE INDEX AND MIDDLE FINGER
34. Rotate reverse clockwise = TOUCH ON Surface with INDEX, THEN DRAW Circle FROM Right TO Left ON Surface with INDEX, THEN RELEASE INDEX
43. Tap on center = Tap ON Surface Center
45. Pinch = TOUCH ON Surface Bottom Left WITH THUMB AND TOUCH ON Surface Top Right WITH INDEX THEN Swipe up from bottom left to top right WITH THUMB AND Swipe down from top right to bottom left WITH INDEX
57. Spread in top left corner = TOUCH Surface WITH INDEX AND THUMB THEN SPREAD ON Surface Top Left WITH INDEX AND THUMB THEN RELEASE INDEX AND THUMB

All 154 unique gestures were classified according to their nature: 101 abstract gestures (e.g., flick, mark, tri-directional gestures), 27 semantic gestures (i.e., which mimic an action on an object), 21 glyph gestures (e.g., an icon, a drawing, a symbol), and 5 alphabetic gestures (e.g., alphabet letters). We now report on the most frequent gestures by referent:

(1) Turn TV on/off: four participants chose to slide their finger up to turn on the TV and down to turn it off. Four people decided to use the same gesture to turn on and off the TV. Three of them use a glyph gesture.
(2) Start player: more than a third decides to use the abstract gesture #43.
(3) Turn up/down the volume. A large majority initially used the volume buttons, but since this was not a gesture on the surface, they turned to drawing a flick top right or left depending the orientation.

(4) Go to next/previous item in a list: most participants use gestures #19 and #20, i.e., to slide their finger to the left to go to the next item and to the right to go to the previous item. Three people decided to do the opposite. We noticed that people who slide their fingers, go to the left to go to the next item while those who click, will click right to go to the next item.
(5) Turn air conditioning on/off: while participants elicited a large spectrum of gestures, one fifth of the sampling preferred gestures #31 and #32.
(6) Turn light on/off: We have many different gestures, with maximum 3 participants eliciting the same gesture. Unlike the previous gesture, a good portion gathers identical gestures or gestures considered similar by category. Participants prefer to have a single gesture that serves as an interrupter. Its action changes according to the state of the lamp.
(7) Brighten/dim lights: they all elicited different gestures. So, a conclusion was hard to reach. Te two most frequent gestures were: three fingers spread and swipe up left.
(8) Turn head on/off: participants mainly used glyph gestures, which is particularly logical because some people elicited a gesture to turn on the heating and others the same to turn it off.
(9) Turn alarm on/off: a third of participants used the same gesture to turn on and turn off their alarm clocks. Few participants used the system-defined gesture as implemented on their smartphone. This may suggest that a user-defined gesture could significantly depart from system-defined gestures imposed by an operating system.
(10) Answer/End phone call: we find mainly 3 pairs of gestures: the sliding down the finger on the screen, the second one detects when you take your mobile phone, the third is unique and consists of touching a screen button.

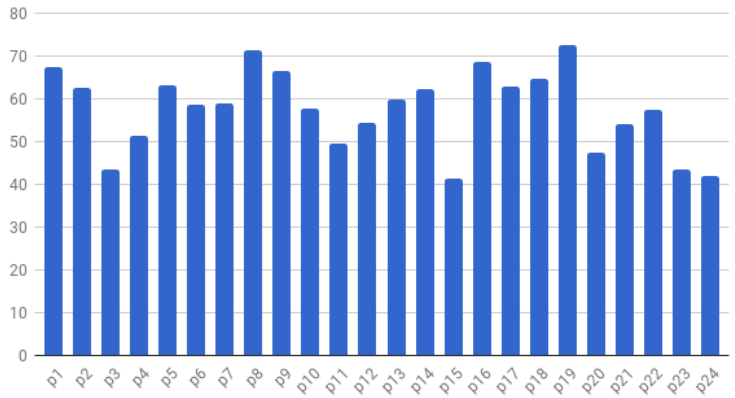

Fig. 15. Results of the creativity test by participant (1-100).

5.4 The ”Measure” Stage

Fig. 16 shows the results by participant by goodness of fit (a) and thinking time (b). Fig. 15 shows the distribution of participants' creativity scores ($Min = 41.43$, $Max = 72.42$, $M = 57.54$, $SD = 9.14$). A creativity score greater or equal than 60 is obtained for a highly creative person. We did not find any statistically significant correlation between the creativity scores and the thinking time. Some participants exhibiting a higher/lower creativity score have a lower/higher average reaction time: p19 benefits of the highest score and one of the lowest average reaction time, and so do p5 and p7. Participants p13 and p14 almost share the same creativity score, but depart from each other with extremely different reaction times. Participant p22 has from far the highest thinking time, but a rather average creativity score.

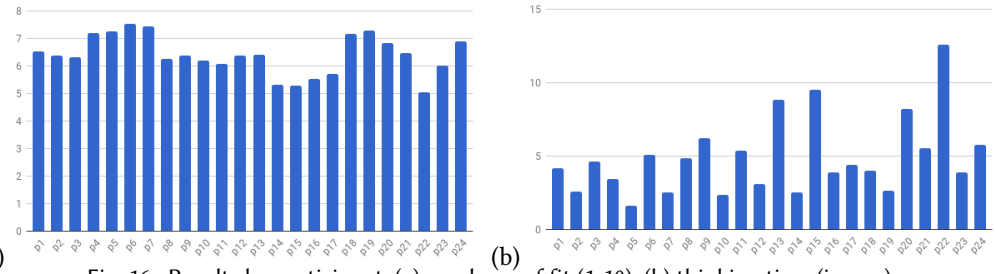

Fig. 16. Results by participant: (a) goodness of fit (1-10), (b) thinking time (in sec.).

GELICIT displays Agreement Scores $A(r)$ [98] and Agreement Rates $AR(r)$ [93] for each REFERENT r condition using their respective formula:

$$A(r) = \sum_{P_i \subseteq P} \left(\frac{|P_i|}{|P|} \right)^2 \geq AR(r) = \frac{|P|}{|P|-1} \sum_{P_i \subseteq P} \left(\frac{|P_i|}{|P|} \right)^2 - \frac{1}{|P|-1} \quad (1)$$

where r denotes the referent for which a gesture will be elicited, $|P|$ denotes the number of elicited gestures, and $|P_i|$ denotes the number of gestures elicited for the i^{th} subgroup of P. Fig. 12 reproduces a screenshot of the agreement rates presented in decreasing order of their value, with the average at the right. Overall, agreement rates are small in magnitude, between 1.1% for "Turn alarm off" and 24.6% for the referents "Turn up/down the volume", for the global sampling ($M = 9.1\%$, $SD = 4.8\%$). 12/19=63% of rates belong to the low range ($\leq$ 10%) and 7/19=37% of rates are medium in the range 10%-30% according to Vatavu and Wobbrock's method [93] to interpret the magnitudes of agreement rates. These results are very similar to the other reported agreement rates in the GES literature ([93], (p. 1332)) that summarizes agreement rates of 18 studies, for which the smallest value (10.8%) was reached by Liang et al. [49] and Seyed et al. [81] for motion+surface and multi-display gestures, respectively. Gheran et al. also display similar magnitudes for ring gestures [31]. According to the recommendations, our results fall inside medium consensus (< .3).

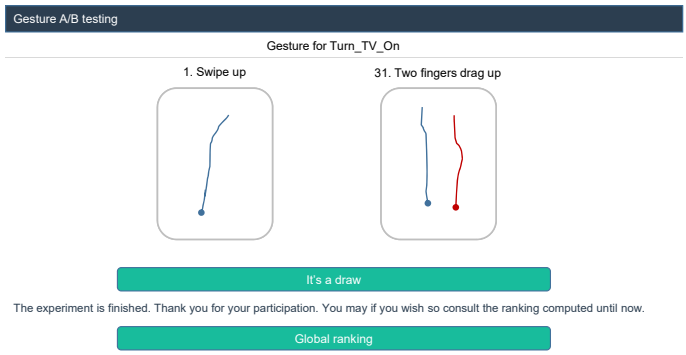

Fig. 17. A/B Testing for the Turn_TV_on referent between gestures #1 and #31.

5.5 The "Discuss" Stage

From the experiment's results, we can see that the average agreement rate is medium, perhaps for several reasons: every participant has a somewhat different way of thinking for the gestures they elicit while using a smartphone, the space of possible gestures is large even on this device, 154 unique gestures have been elicited that could be further classified in more abstract classes (e.g., grouping all swipe gestures into one category that is then decomposed into sub-categories depending on conditions, clustering all gestures in the same category regardless their mount of fingers when possible). A same gesture is sometimes used for a same referent by another participant

or for another referent, which inevitably affects the overall agreement. To better identify the participants' preferences and to test these gesture candidates, the designer can engage into a discussion with participants involved, e.g. through A/B testing (Fig. 17). The takeaway from our experiment are the following:

- People tend to reuse gestures based on context of use as many gestures are biased by legacy and previous experience with similar devices.
- For binary tasks (e.g., on/off), participants tend prefer to use the same gesture (which is used each time to swap the status) or a similar gesture in opposite direction (e.g., swipe up vs down).
- Orientation of tasks influences gesture orientation: left to right gestures are preferred for next item and right to left gestures are preferred for previous items, even if the gesture has a different nature. Orientation to the right or up often means progression or increasing, while orientation to the left or down means regression or decreasing, or even return to the past.
- For physical touches, some specific regions on the surface, such as the center and the four corners, are preferred for different tasks because they are easily identifiable.

5.6 The "Export" Stage

Depending on the results coming from these mechanisms, the designer can keep or discard any elicited gesture from the subset until obtaining a *consensus set*, which is a set of elicited gestures that are considered to benefit from a sufficient agreement among participants. The designer then exports the resulting subset into a JSON file to be incorporated in the rest of the engineering life cycle, along with a gesture recognizer, here $P [91].

6 DISCUSSION OF GELICIT

Through a case study, we have shown that Gelicit can become effective and efficient in conductin a GES in a distributed fashion. Gelicit as a guiding method may induce, stimulate, and facilitate new ways of conducting such elicitation studies. We hereby summarizes the benefits and the limitations.

6.1 Benefits

This subsection revisits the eight requirements initially elicited for Gelicit in the light of the experience gained:

R1. **Zero-install software with user accounts**. This requirement is completely fulfilled since Gelicit is hosted as a cloud computing application on Heroku.

R2. **Collaborative management of gesture sets with persistence**. This requirement is fully satisfied as any gesture set, along with its clusters, classes, and samples, can be managed independently of each other based on the access rights defined by the stakeholders' roles. Full concurrency is ensured by Gelicit.

R3. **Support for distributed GES stages**. This requirement is also satisfied since Gelicit enables ordering the six stages (Fig. 5) flexibly for the four distributions in space and time (Fig. 2), thus opening doors to new types of studies where participants are gathered in one or several locations, which is also particularly welcome for collaborative gestures.

R4. **Flexible GES parameterization**. This requirement is satisfied to the extent of which parameters could cover a GES experiment (Fig. 8). More parameters and more values for them would render this requirement more demanding. Any GES experiment is explicitly stored in a configuration file, thus fostering reusability [66] with the same parameters and participants or not, across multiple contexts of use. The Association for Computing Machinery (ACM) awards badges when a paper's artifacts-software, scripts, or data set, either created by the

authors to be used as part of the study or generated by the experiment itself, are determined to be[6]: *repeatable* by the same team with the same experimental setup, *replicable* by a different team with the same experimental setup, or *reproducible* by a different team with a different experimental setup.

R5. **Multiple measurement methods for each gesture set.** This requirement is satisfied in principle since one or many measurement methods can be attached to any experiment. While GELICIT accommodates several measurement methods, it only considers agreement measures for the moment. Therefore, the consideration and the interest for multiple measurement methods will evolve with the implementation of such methods. The software architecture is flexible enough for this purpose.

R6. **Computer-aided computation of measures.** This requirement is fulfilled depending on whether measures can be automatically computed. Other measures may require a human intervention, either for providing data or for computing unsupported formula,

R7. **Support for discussion towards consensus set.** This requirement is partially satisfied as three mechanisms for discussion are considered. Their results could provide the designer with additional insights on which gestures could be finally kept in the consensus set and the participants with a deeper level of involvement since their opinion can be taken into account more profoundly. User-based vs choice-based elicitation methods have been already investigated [23] and GELICIT would be suitable for these considerations.

R8. **Continuity with the rest of the user interface development life cycle**. This requirement is minimally satisfied by exporting the consensus set to the developer with a gesture recognizer. But there is no traceability ensured afterwards and there is no linking between gestures and application commands.

6.2 Limitations

There are three major limitations to GELICIT as it is currently designed and implemented:

(1) **Limitations due to the gesture capture**: although the GELICIT metamodel is able to afford a wide array of gestures (Fig. 4), its capturing mechanism is limited to three modalities due to HTML5 intrinsic limitations (Fig. 7): pen-based, finger-based, and pointer-based (e.g., using an object tracking system). Further implementation is required to capture more sophisticated categories of gestures through devices equipped with new tracking capabilities. When such devices appear, transforming their raw data into gesture properties is required. GELICIT does not impose any particular constraint in remote gesture capture provided that the device can locally capture them and stream them to the application.

(2) **Limitations related to data management**: GELICIT stores all data coming from participants, whether they are working in a collocated way or remotely, but does not associate them directly. Location, browser, time and system parameters could be captured, but inferring a collaboration type between participants from the data remains hard to achieve. Additional characterization is needed. Moreover, all data manipulated by GELICIT are not subject to statistical analysis apart from those for the measurement methods.

(3) **Limitations intrinsically related to GES**: GELICIT does not support the automatic labeling and classification of elicited gestures. The stakeholder is therefore responsible for completely and consistently ensuring these tasks. Pattern matching and machine learning techniques could be investigated to learn some classification scheme by observing how an experimenter, a rater, a designer or even a participant may want to classify gestures into clusters and categories. This is beyond the scope of this paper.

[6] https://www.acm.org/publications/policies/artifact-review-badging

7 CONCLUSION AND FUTURE WORK

This paper presented Gelicit, probably the first cloud computing collaborative platform for conducting a gesture elicitation study with stakeholders distributed in time and space (Fig. 2). It consists of three contributions: its metamodel (Fig. 4, its six-stage flexible workflow as a guiding method (Fig. 5), and its supporting software (Fig. 7). As such, Gelicit reflects pretty much the original definition of the gesture elicitation study [98] by expanding the ordering of its stages and generalizing them. Based on the experienced gained insofar, we hereby present some avenues to some stages defined in this workflow, which represent some suggestions for the guiding method and its support software.

Definition of task set: Gelicit supports different task sets and the designer can create a new one by importing an existing task set and edit its definition. New task sets could emerge as GES become more proeminent. For example, different standard task sets may be predefined such as the Canonical Task Types [32] and the task types from Lenorovitz [45]. These task sets could then be shared for stimulating further studies on uncovered areas and form a baseline for benchmarking. By relying on the same task set, different studies could compare their respective results.

Definition of referent type: Gelicit focuses on eliciting gestures for graphical user interfaces only. Other type of commands could serve as symbols, such as mouse commands, user interface behaviors, function keys, icons, vocal commands [4], low fidelity gestures [36], or even completely interactive prototypes. This would require behavior specifications that is expressiveness enough to automatically generate corresponding code.

Definition of referent medium: Gelicit supports various referent mediums, such as an image, a picture, a GIF animation, and a video sequence. Eliciting a gesture on top of a user interface wireframe or screenshot was revealed as very expressive because participants elicit gestures in their very right context of use [3]. Superimposing a gesture image on top of the referent image produces an expressive representation. Other medium types could be investigated such as an interactive prototype: instead of drawing all transitions by hand, the participant could be engaged in an interactive scenario where elicitation is an explicit facility of prototyping. The medium used for expressing the referent also impacts the quality with which gestures will be elicited [59].

Selection of referents: referents are typically selected in a random manner to avoid any bias, such as learning effect or fatigue, which is fundamental for between-subjects elicitation studies [94]. This selection is randomized at the price of forgetting potential relationships between referents. For instance, the task "dim the light up" is the symmetric of the task "dim the light down" ; "move up", "move down", "move left", "move right", "move forward", and "move backward" represent all the potential space movements. When their corresponding referents are presented randomly, the participant can only establish their relationship after integrating them all, thus loosing relationships. Other relationships exist: *inclusion* (a referent subsumes another one), *exclusion* (a referent is conflicting with another one), *generalisation/specialisation* (a referent is more general or specific than another one). Participants elicit gestures for these referents without exploiting their relationships.

Selection of participants: participants are recruited via a procedure that is external to Gelicit and are only known by their email and demographic data. This selection is performed under the responsibility of the designer. One can imagine a mechanism for automated selection of participants from a database based on criteria so as to create various samplings, such as Simple Random Sampling (SRS), stratified, cluster Sampling, systematic or multistage sampling. This also touches the field of crowdsourcing [68]: a large amount of participants could be imagined for performing the tasks [70, 85]. The question of tapping the right crowd then occurs and should be supported by a dedicated method [9]: crowdsourcing participants tend to participate more when they enjoy community benefits that increase their utility.

Selection of measurement method: while the agreement measures [93, 94] are widely used in the community, other agreement measures exist [19, 88] and could be compared to each other to detect their correlation, if any. Adding another measurement method is possible since all data required for the computation is already stored in the database. Madapana et al. [56] argue that agreement rates and scores are only based on pairs of occurrences and suggest computing a distance measure between binary vectors representing gesture features.

Gesture classification: for speeding up the classification, a machine learning algorithm, such as deep image analysis associated to feature extraction or random forest search, could be run and provide different options: automated classification of elicited gestures, manual classification when no recognition is accurate but a previously classified gesture can be re-applied through the vocabulary selection (bottom part of Fig. 10), or mixed-initiative classification. In this last category, the algorithm could suggest several candidates for classification and rank them in decreasing order of accuracy. The experimenter could then pick one of them or enter an ew classification as in Detexify[7], which does a similar job for translating a symbol into its LaTeX instruction.

Gesture recognition, querying, and catalog production: GELICIT ends up with a consensus set and an associated recognizer, typically $1 [5], $P [91], and !FTL [89] in its two versions. Any other recognizer could be equally incorporated, such as $Q [92] (Listing 1). When a new gesture is elicited, a recognition process could be launched while producing the gesture to provide immediate feedback and auto-completion such as in G-Gene [16]. Instead of recognizing an elicited gesture from all candidates, the scope could be restricted to those gestures that were previously elicited for the same set of referents or in the same context of use. The scope of recognition could then be relaxed progressively to cover more possibilities. The persistence of the consensus set in the gesture database may also raise a machine learning process to create a classifier that is optimized for this particular consensus set. As soon as the gesture database becomes rich enough, any stakeholder could query this database with requests like "Show me all the gestures that are distinct enough from the user's viewpoint for multi-display environment for this task set" or "Show me all gestures elicited for elderly people on touch screen". These queries are made possible thanks to the context of use associated to each gesture. As another output, the system could create an on-line browsable catalog of gestures, thus serving as a gesture repository.

The above suggestions inevitably touch the elicitation method definition and further fundamental research is expected to address these questions. These new requirements, as soon as they are settled out, could be turned into a new version of GELICIT that reflects this progress. GELICIT therefore offers the premices for letting a GES support evolving over time.

ACKNOWLEDGMENTS

The authors would like to thank the members of the focus group for contributing to the initial textual scenario, these members for providing specifications about the application scenario, Romain Dizier, Zacharie Kerger, Gilles Bajart for implementing GELICIT as described in this paper, and Nicolas Burny for development advise. The authors would like also to thank the anonymous reviewers for their constructive comments on earlier versions of this manuscript.

REFERENCES

[1] Mahmoud Abduo and Matthias Galster. 2015. *Myo Gesture Control Armband for Medical Applications.* Technical Report. University of Canterbury, College of Engineering. http://www.cosc.canterbury.ac.nz/research/reports/HonsReps/abstracts/1502_abs.html

[7] http://detexify.kirelabs.org/classify.html

[2] Anne Adams and Anna L. Cox. 2008. Questionnaires, in-depth interviews and focus groups. In *Research Methods for Human Computer Interaction*, Paul Cairns and Anna L. Cox (Eds.). Cambridge University Press, Cambridge, UK, 17–34. http://oro.open.ac.uk/11909/

[3] Imeh Akpan, Paul Marshall, Jon Bird, and Daniel Harrison. 2013. Exploring the Effects of Space and Place on Engagement with an Interactive Installation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2213–2222. https://doi.org/10.1145/2470654.2481306

[4] Abdullah X. Ali, Meredith Ringel Morris, and Jacob O. Wobbrock. 2018. Crowdsourcing Similarity Judgments for Agreement Analysis in End-User Elicitation Studies. In *The 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. ACM, New York, NY, USA, 177–188. https://doi.org/10.1145/3242587.3242621

[5] Lisa Anthony and Jacob O. Wobbrock. 2010. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of the Graphics Interface 2010 Conference, May 31 - June 02, 2010, Ottawa, Ontario, Canada*, David Mould and Sylvie Noël (Eds.). ACM / Canadian Human-Computer Communications Society, 245–252. http://portal.acm.org/citation.cfm?id=1839258&CFID=37966912&CFTOKEN=40351478

[6] Shaikh Shawon Arefin Shimon, Courtney Lutton, Zichun Xu, Sarah Morrison-Smith, Christina Boucher, and Jaime Ruiz. 2016. Exploring Non-touchscreen Gestures for Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3822–3833. https://doi.org/10.1145/2858036.2858385

[7] Daniel Ashbrook and Thad Starner. 2010. MAGIC: A Motion Gesture Design Tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 2159–2168. https://doi.org/10.1145/1753326.1753653

[8] Ceylan Beşevli, Oğuz Turan Buruk, Merve Erkaya, and Oğuzhan Özcan. 2018. Investigating the Effects of Legacy Bias: User Elicited Gestures from the End Users Perspective. In *Proceedings of the 2018 ACM Conference Companion Publication on Designing Interactive Systems (DIS '18 Companion)*. ACM, New York, NY, USA, 277–281. https://doi.org/10.1145/3197391.3205449

[9] Paul Belleflamme, Thomas Lambert, and Armin Schwienbacher. 2014. Crowdfunding: Tapping the right crowd. *Journal of Business Venturing* 29, 5 (2014), 585 – 609. https://doi.org/10.1016/j.jbusvent.2013.07.003

[10] François Beuvens and Jean Vanderdonckt. 2012. Designing Graphical User Interfaces Integrating Gestures. In *Proceedings of the 30th ACM International Conference on Design of Communication (SIGDOC '12)*. ACM, New York, NY, USA, 313–322. https://doi.org/10.1145/2379057.2379116

[11] Birgit Bomsdorf, Rainer Blum, and Daniel Künkel. 2015. Towards ProGesture, a Tool Supporting Early Prototyping of 3D-Gesture Interaction. *Int. J. People-Oriented Program.* 4, 2 (July 2015), 54–70. https://doi.org/10.4018/IJPOP.2015070103

[12] Idil Bostan, Oğuz Turan Buruk, Mert Canat, Mustafa Ozan Tezcan, Celalettin Yurdakul, Tilbe Göksun, and Oğuzhan Özcan. 2017. Hands As a Controller: User Preferences for Hand Specific On-Skin Gestures. In *Proceedings of the 2017 Conference on Designing Interactive Systems (DIS '17)*. ACM, New York, NY, USA, 1123–1134. https://doi.org/10.1145/3064663.3064766

[13] Oğuz Turan Buruk and Oğuzhan Özcan. 2017. GestAnalytics: Experiment and Analysis Tool for Gesture-Elicitation Studies. In *Proceedings of the 2017 ACM Conference Companion Publication on Designing Interactive Systems (DIS '17 Companion)*. ACM, New York, NY, USA, 34–38. https://doi.org/10.1145/3064857.3079114

[14] Maria Claudia Buzzi, Marina Buzzi, Barbara Leporini, and Amaury Trujillo. 2015. Exploring Visually Impaired People's Gesture Preferences for Smartphones. In *Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter (CHItaly 2015)*. ACM, New York, NY, USA, 94–101. https://doi.org/10.1145/2808435.2808448

[15] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. 2003. A Unifying Reference Framework for multi-target user interfaces. *Interacting with Computers* 15, 3 (2003), 289–308. https://doi.org/10.1016/S0953-5438(03)00010-9

[16] 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

[17] Stefano Carrino, Elena Mugellini, Omar Abou Khaled, and Rolf Ingold. 2011. ARAMIS: Toward a Hybrid Approach for Human- Environment Interaction. In *Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments*, Julie A. Jacko (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 165–174.

[18] Edwin Chan, Teddy Seyed, Wolfgang Stuerzlinger, Xing-Dong Yang, and Frank Maurer. 2016. User Elicitation on Single-hand Microgestures. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3403–3414. https://doi.org/10.1145/2858036.2858589

[19] Alessandro Checco, Kevin Roitero, Eddy Maddalena, Stefano Mizzaro, and Gianluca Demartini. 2017. Let's Agree to Disagree: Fixing Agreement Measures for Crowdsourcing. In *Proceedings of the Fifth AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2017, 23-26 October 2017, Québec City, Québec, Canada.*, Steven Dow and Adam Tauman Kalai (Eds.). AAAI Press, 11–20. https://aaai.org/ocs/index.php/HCOMP/HCOMP17/paper/view/15927

[20] Xiang 'Anthony' Chen, Julia Schwarz, Chris Harrison, Jennifer Mankoff, and Scott E. Hudson. 2014. Air+Touch: Interweaving Touch & In-air Gestures. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 519–525. https://doi.org/10.1145/2642918.2647392

[21] Sabrina Connell, Pei-Yi Kuo, Liu Liu, and Anne Marie Piper. 2013. A Wizard-of-Oz Elicitation Study Examining Child-defined Gestures with a Whole-body Interface. In *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*. ACM, New York, NY, USA, 277–280. https://doi.org/10.1145/2485760.2485823

[22] Adrien Coyette, Sascha Schimke, Jean Vanderdonckt, and Claus Vielhauer. 2007. Trainable Sketch Recognizer for Graphical User Interface Design. In *Human-Computer Interaction – INTERACT 2007*, Cécilia Baranauskas, Philippe Palanque, Julio Abascal, and Simone Diniz Junqueira Barbosa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 124–135.

[23] Nem Khan Dim, Chaklam Silpasuwanchai, Sayan Sarcar, and Xiangshi Ren. 2016. Designing Mid-Air TV Gestures for Blind People Using User- and Choice-Based Elicitation Approaches. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. ACM, New York, NY, USA, 204–214. https://doi.org/10.1145/2901790.2901834

[24] 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 (CHI '18)*. ACM, New York, NY, USA, Article 419, 12 pages. https://doi.org/10.1145/3173574.3173993

[25] Haiwei Dong, Ali Danesh, Nadia Figueroa, and Abdulmotaleb El Saddik. 2015. An Elicitation Study on Gesture Preferences and Memorability Toward a Practical Hand-Gesture Vocabulary for Smart Televisions. *IEEE Access* 3 (2015), 543–555. https://doi.org/10.1109/ACCESS.2015.2432679

[26] Tanja Döring, Dagmar Kern, Paul Marshall, Max Pfeiffer, Johannes Schöning, Volker Gruhn, and Albrecht Schmidt. 2011. Gestural Interaction on the Steering Wheel: Reducing the Visual Demand. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 483–492. https://doi.org/10.1145/1978942.1979010

[27] Yasmin Felberbaum and Joel Lanir. 2018. Better Understanding of Foot Gestures: An Elicitation Study. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 334, 12 pages. https://doi.org/10.1145/3173574.3173908

[28] Sebastian Feuerstack, Mauro Dos Santos Anjo, and Ednaldo B. Pizzolato. 2011. Model-based Design and Generation of a Gesture-based User Interface Navigation Control. In *Proceedings of the 10th Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction (IHC+CLIHC '11)*. Brazilian Computer Society, Porto Alegre, Brazil, Brazil, 227–231. http://dl.acm.org/citation.cfm?id=2254436.2254475

[29] Leah Findlater, Ben Lee, and Jacob Wobbrock. 2012. Beyond QWERTY: Augmenting Touch Screen Keyboards with Multi-touch Gestures for Non-alphanumeric Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2679–2682. https://doi.org/10.1145/2207676.2208660

[30] Ivano Gatto and Fabio Pittarello. 2012. Prototyping a Gestural Interface for Selecting and Buying Goods in a Public Environment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12)*. ACM, New York, NY, USA, 784–785. https://doi.org/10.1145/2254556.2254713

[31] Bogdan-Florin Gheran, Jean Vanderdonckt, and Radu-Daniel Vatavu. 2018. Gestures for Smart Rings: Empirical Results, Insights, and Design Implications. In *Proceedings of the 2018 Designing Interactive Systems Conference (DIS '18)*. ACM, New York, NY, USA, 623–635. https://doi.org/10.1145/3196709.3196741

[32] Juan Manuel Gonzalez-Calleros, Josefina Guerrero-Garcia, Jean Vanderdonckt, and Jaime Munoz-Arteaga. 2009. Towards Canonical Task Types for User Interface Design. In *2009 Latin American Web Congress*. 63–70. https://doi.org/10.1109/LA-WEB.2009.33

[33] Hayati Havlucu, Mehmet Yarkın Ergin, İdil Bostan, Oğuz Turan Buruk, Tilbe Göksun, and Oğuzhan Özcan. 2017. It Made More Sense: Comparison of User-Elicited On-skin Touch and Freehand Gesture Sets. In *Distributed, Ambient and Pervasive Interactions*, Norbert Streitz and Panos Markopoulos (Eds.). Springer International Publishing, Cham, 159–171.

[34] Lynn Hoff, Eva Hornecker, and Sven Bertel. 2016. Modifying Gesture Elicitation: Do Kinaesthetic Priming and Increased Production Reduce Legacy Bias?. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '16)*. ACM, New York, NY, USA, 86–91. https://doi.org/10.1145/2839462.2839472

[35] Simon Holmes. 2015. *Getting MEAN with Mongo, Express,Angular and Node*. Manning Publications, Shelter Island, New York, United States.

[36] Ali Hosseini-Khayat, Teddy Seyed, Chris Burns, and Frank Maurer. 2011. Low-fidelity prototyping of gesture-based applications. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing System, EICS 2011, Pisa, Italy, June 13-16, 2011*, Fabio Paternò, Kris Luyten, and Frank Maurer (Eds.). ACM, 289–294. https://doi.org/10.1145/1996461.1996538

[37] Sujin Jang, Niklas Elmqvist, and Karthik Ramani. 2014. GestureAnalyzer: Visual Analytics for Pattern Analysis of Mid-air Hand Gestures. In *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction (SUI '14)*. ACM, New York, NY, USA, 30–39. https://doi.org/10.1145/2659766.2659772

[38] Tero Jokela, Parisa Pour Rezaei, and Kaisa Väänänen. 2016. Using Elicitation Studies to Generate Collocated Interaction Methods. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '16)*. ACM, New York, NY, USA, 1129–1133. https://doi.org/10.1145/2957265.2962654

[39] Frederic Kerber, Michael Puhl, and Antonio Krüger. 2017. User-independent Real-time Hand Gesture Recognition Based on Surface Electromyography. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17)*. ACM, New York, NY, USA, Article 36, 7 pages. https://doi.org/10.1145/3098279.3098553

[40] Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. 2012. Proton++: A Customizable Declarative Multitouch Framework. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 477–486. https://doi.org/10.1145/2380116.2380176

[41] Anne Köpsel and Nikola Bubalo. 2015. Benefiting from Legacy Bias. *interactions* 22, 5 (Aug. 2015), 44–47. https://doi.org/10.1145/2803169

[42] Christian Kray, Daniel Nesbitt, John Dawson, and Michael Rohs. 2010. User-defined Gestures for Connecting Mobile Phones, Public Displays, and Tabletops. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '10)*. ACM, New York, NY, USA, 239–248. https://doi.org/10.1145/1851600.1851640

[43] Nikhil Krishnaswamy, Pradyumna Narayana, Isaac Wang, Kyeongmin Rim, Rahul Bangar, Dhruva Patil, Gururaj Mulay, Ross Beveridge, Jaime Ruiz, Bruce Draper, and James Pustejovsky. 2017. Communicating and Acting: Understanding Gesture in Simulation Semantics. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*. http://aclweb.org/anthology/W17-6919

[44] Richard Krueger and Mary Anne Casey. 2000. *Focus Groups: A Practical Guide for Applied Research*. Sage Publications, Thousand Oaks, CA.

[45] David R. Lenorovitz, Mark D. Phillips, R.S. Ardrey, and Gregory V. Kloster. 1984. A taxonomic approach to characterizing human-computer interaction. In *Human-Computer Interaction*, Gabriel Salvendy (Ed.). Elsevier Science Publishers, Amsterdam, 111–116.

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

[47] James R. Lewis. 2002. Psychometric Evaluation of the PSSUQ Using Data from Five Years of Usability Studies. *International Journal of HumanâĂŞComputer Interaction* 14, 3-4 (2002), 463–488. https://doi.org/10.1080/10447318.2002.9669130 arXiv:https://doi.org/10.1080/10447318.2002.9669130

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

[49] Hai-Ning Liang, Cary Williams, Myron Semegen, Wolfgang Stuerzlinger, and Pourang Irani. 2012. User-defined Surface+Motion Gestures for 3D Manipulation of Objects at a Distance Through a Mobile Device. In *Proc. of the 10th Asia Pacific Conf. on Computer Human Interaction (APCHI '12)*. ACM, New York, NY, USA, 299–308. https://doi.org/10.1145/2350046.2350098

[50] 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

[51] Lars Lischke, Pascal Knierim, and Hermann Klinke. 2015. Mid-Air Gestures for Window Management on Large Displays. In *Mensch und Computer 2015 âĂŞ Proceedings*, Sarah Diefenbach, Niels Henze, and Martin Pielot (Eds.). De Gruyter Oldenbourg, Berlin, 439–442.

[52] Allan Christian Long, Jr., James A. Landay, and Lawrence A. Rowe. 1999. Implications for a Gesture Design Tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 40–47. https://doi.org/10.1145/302979.302985

[53] 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 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1685–1694. https://doi.org/10.1145/2556288.2557263

[54] Hao Lü and Yang Li. 2012. Gesture Coder: A Tool for Programming Multi-touch Gestures by Demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2875–2884. https://doi.org/10.1145/2207676.2208693

[55] Hao Lü and Yang Li. 2013. Gesture Studio: Authoring Multi-touch Interactions Through Demonstration and Declaration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 257–266. https://doi.org/10.1145/2470654.2470690

[56] Naveen Madapana, Glebys Gonzalez, Richard Rodgers, Lingsong Zhang, and Juan P. Wachs. 2018. Gestures for Picture Archiving and Communication Systems (PACS) operation in the operating room: Is there any standard? *PLOS ONE* 13, 6 (06 2018), 1–13. https://doi.org/10.1371/journal.pone.0198092

[57] Mary Lou Maher and Lina Lee. 2017. Designing for Gesture and Tangible Interaction. *Synthesis Lectures on Human-Centered Informatics* 10, 2 (2017), i–111. https://doi.org/10.2200/S00758ED1V01Y201702HCI036 arXiv:https://doi.org/10.2200/S00758ED1V01Y201702HCI036

[58] Dan Mauney, Jonathan Howarth, Andrew Wirtanen, and Miranda Capra. 2010. Cultural Similarities and Differences in User-defined Gestures for Touchscreen User Interfaces. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems (CHI EA '10)*. ACM, New York, NY, USA, 4015–4020. https://doi.org/10.1145/1753846.1754095

[59] Erin McAweeney, Haihua Zhang, and Michael Nebeling. 2018. User-Driven Design Principles for Gesture Representations. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 547, 13 pages. https://doi.org/10.1145/3173574.3174121

[60] Fabrizio Milazzo, Vito Gentile, Antonio Gentile, and Salvatore Sorce. 2018. KIND-DAMA: A modular middleware for Kinect-like device data management. *Software: Practice and Experience* 48, 1 (2018), 141–160. https://doi.org/10.1002/spe.2521 spe.2521.

[61] Gourav Modanwal and Kishor Sarawadekar. 2018. A Gesture Elicitation Study with Visually Impaired Users. In *HCI International 2018 – Posters' Extended Abstracts*, Constantine Stephanidis (Ed.). Springer International Publishing, Cham, 54–61.

[62] Giulio Mori, Fabio Paternò, and Carmen Santoro. 2002. CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. *IEEE Trans. Software Eng.* 28, 8 (2002), 797–813. https://doi.org/10.1109/TSE.2002.1027801

[63] Meredith Ringel Morris, Andreea Danielescu, Steven Drucker, Danyel Fisher, Bongshin Lee, m. c. schraefel, and Jacob O. Wobbrock. 2014. Reducing Legacy Bias in Gesture Elicitation Studies. *interactions* 21, 3 (May 2014), 40–45. https://doi.org/10.1145/2591689

[64] Meredith Ringel Morris, Jacob O. Wobbrock, and Andrew D. Wilson. 2010. Understanding Users' Preferences for Surface Gestures. In *Proceedings of Graphics Interface 2010 (GI '10)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 261–268. http://dl.acm.org/citation.cfm?id=1839214.1839260

[65] Miguel A. Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. 2013. Memorability of Pre-designed and User-defined Gesture Sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1099–1108. https://doi.org/10.1145/2470654.2466142

[66] Michael Nebeling, Alexander Huber, David Ott, and Moira C. Norrie. 2014. Web on the Wall Reloaded: Implementation, Replication and Refinement of User-Defined Interaction Sets. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces (ITS '14)*. ACM, New York, NY, USA, 15–24. https://doi.org/10.1145/2669485.2669497

[67] Michael Nebeling, David Ott, and Moira C. Norrie. 2015. Kinect Analysis: A System for Recording, Analysing and Sharing Multimodal Interaction Elicitation Studies. In *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '15)*. ACM, New York, NY, USA, 142–151. https://doi.org/10.1145/2774225.2774846

[68] Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. 2013. CrowdStudy: general toolkit for crowdsourced evaluation of web interfaces. In *ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS'13, London, United Kingdom - June 24 - 27, 2013*, Peter Forbrig, Prasun Dewan, Michael Harrison, and Kris Luyten (Eds.). ACM, 255–264. https://doi.org/10.1145/2480296.2480303

[69] Michael Nebeling, Elena Teunissen, Maria Husmann, and Moira C. Norrie. 2014. XDKinect: development framework for cross-device interaction using kinect. In *ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS'14, Rome, Italy, June 17-20, 2014*, Fabio Paternò, Carmen Santoro, and Jürgen Ziegler (Eds.). ACM, 65–74. https://doi.org/10.1145/2607023.2607024

[70] Michael Nebeling, Alexandra To, Anhong Guo, Adrian A. de Freitas, Jaime Teevan, Steven P. Dow, and Jeffrey P. Bigham. 2016. WearWrite: Crowd-Assisted Writing from Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016*, Jofish Kaye, Allison Druin, Cliff Lampe, Dan Morris, and Juan Pablo Hourcade (Eds.). ACM, 3834–3846. https://doi.org/10.1145/2858036.2858169

[71] Michael Nielsen, Moritz Störring, Thomas B. Moeslund, and Erik Granum. 2004. A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for HCI. In *Gesture-Based Communication in Human-Computer Interaction*, Antonio Camurri and Gualtiero Volpe (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 409–420.

[72] Tran Pham, Jo Vermeulen, Anthony Tang, and Lindsay MacDonald Vermeulen. 2018. Scale Impacts Elicited Gestures for Manipulating Holograms: Implications for AR Gesture Design. In *Proceedings of the 2018 Designing Interactive Systems Conference (DIS '18)*. ACM, New York, NY, USA, 227–240. https://doi.org/10.1145/3196709.3196719

[73] Thammathip Piumsomboon, Adrian Clark, Mark Billinghurst, and Andy Cockburn. 2013. User-defined Gestures for Augmented Reality. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems (CHI EA '13)*. ACM, New York, NY, USA, 955–960. https://doi.org/10.1145/2468356.2468527

[74] Isabel Benavente Rodriguez and Nicolai Marquardt. 2017. Gesture Elicitation Study on How to Opt-in & Opt-out from Interactions with Public Displays. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 32–41. https://doi.org/10.1145/3132272.3134118

[75] Dean Rubine. 1991. Specifying gestures by example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1991, Providence, RI, USA, April 27-30, 1991*, James J. Thomas (Ed.). ACM, 329–337. https://doi.org/10.1145/122718.122753

[76] Jaime Ruiz, Yang Li, and Edward Lank. 2011. User-defined Motion Gestures for Mobile Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 197–206. https://doi.org/10.1145/1978942.1978971

[77] Jaime Ruiz and Daniel Vogel. 2015. Soft-Constraints to Reduce Legacy and Performance Bias to Elicit Whole-body Gestures with Low Arm Fatigue. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3347–3350. https://doi.org/10.1145/2702123.2702583

[78] Vít Rusnák, Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. 2018. Designing Coherent Gesture Sets for Multi-scale Navigation on Tabletops. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 142, 12 pages. https://doi.org/10.1145/3173574.3173716

[79] Nick Russell, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. 2016. *Workflow Patterns: The Definitive Guide*. MIT Press.

[80] 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. *Information & Software Technology* 109 (2019), 43–59. https://doi.org/10.1016/j.infsof.2019.01.006

[81] Teddy Seyed, Chris Burns, Mario Costa Sousa, Frank Maurer, and Anthony Tang. 2012. Eliciting Usable Gestures for Multi-display Environments. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces (ITS '12)*. ACM, New York, NY, USA, 41–50. https://doi.org/10.1145/2396636.2396643

[82] Beat Signer, U. Kurmann, and Moira Norrie. 2007. iGesture: A General Gesture Recognition Framework. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. 954–958. https://doi.org/10.1109/ICDAR.2007.4377056

[83] Tiffanie R. Smith and Juan E. Gilbert. 2018. Dancing to Design: A Gesture Elicitation Study. In *Proceedings of the 17th ACM Conference on Interaction Design and Children (IDC '18)*. ACM, New York, NY, USA, 638–643. https://doi.org/10.1145/3202185.3210790

[84] Lucio Davide Spano, Antonio Cisternino, Fabio Paternò, and Gianni Fenu. 2013. GestIT: A Declarative and Compositional Framework for Multiplatform Gesture Definition. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '13)*. ACM, New York, NY, USA, 187–196. https://doi.org/10.1145/2494603.2480307

[85] Maximilian Speicher and Michael Nebeling. 2018. GestureWiz: A Human-Powered Gesture Design Environment for User Interface Prototypes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 107, 11 pages. https://doi.org/10.1145/3173574.3173681

[86] Kashmiri Stec and Lars Bo Larsen. 2018. Gestures for Controlling a Moveable TV. In *Proceedings of the 2018 ACM International Conference on Interactive Experiences for TV and Online Video (TVX '18)*. ACM, New York, NY, USA, 5–14. https://doi.org/10.1145/3210825.3210831

[87] Jeff K. T. Tang and Takeo Igarashi. 2013. CUBOD: a customized body gesture design tool for end users. In *BCS-HCI '13 Proceedings of the 27th International BCS Human Computer Interaction Conference, Brunel University, London, UK, 9-13 September 2013*, Steve Love, Kate S. Hone, and Tom McEwan (Eds.). British Computer Society, 5. http://dl.acm.org/citation.cfm?id=2578056

[88] Theophanis Tsandilas. 2018. Fallacies of Agreement: A Critical Review of Consensus Assessment Methods for Gesture Elicitation. *ACM Trans. Comput.-Hum. Interact.* 25, 3, Article 18 (June 2018), 49 pages. https://doi.org/10.1145/3182168

[89] 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 *Proceedings of the 2018 on International Conference on Multimodal Interaction (ICMI '18)*. ACM, New York, NY, USA, 125–134. https://doi.org/10.1145/3242969.3243032

[90] Radu-Daniel Vatavu. 2017. Characterizing gesture knowledge transfer across multiple contexts of use. *J. Multimodal User Interfaces* 11, 4 (2017), 301–314. https://doi.org/10.1007/s12193-017-0247-x

[91] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures as point clouds: a $P recognizer for user interface prototypes. In *International Conference on Multimodal Interaction, ICMI '12, Santa Monica, CA, USA, October 22-26, 2012*, Louis-Philippe Morency, Dan Bohus, Hamid K. Aghajan, Justine Cassell, Anton Nijholt, and Julien Epps (Eds.). ACM, 273–280. https://doi.org/10.1145/2388676.2388732

[92] Radu-Daniel Vatavu, Lisa Anthony, and Jacob Wobbrock. 2018. $Q: A Super-Quick, Articulation-Invariant Stroke-Gesture Recognizer for Low-Resource Devices. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '18)*. ACM, New York, NY, USA, 623–635. https://doi.org/10.

1145/3229434.3229465

[93] Radu-Daniel Vatavu and Jacob O. Wobbrock. 2015. Formalizing Agreement Analysis for Elicitation Studies: New Measures, Significance Test, and Toolkit. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1325–1334. https://doi.org/10.1145/2702123.2702223

[94] Radu-Daniel Vatavu and Jacob O. Wobbrock. 2016. Between-Subjects Elicitation Studies: Formalization and Tool Support. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3390–3402. https://doi.org/10.1145/2858036.2858228

[95] Radu-Daniel Vatavu and Ionut-Alexandru Zaiti. 2014. Leap Gestures for TV: Insights from an Elicitation Study. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video (TVX '14)*. ACM, New York, NY, USA, 131–138. https://doi.org/10.1145/2602299.2602316

[96] Julie R. Williamson, Stephen Brewster, and Rama Vennelakanti. 2013. Mo!Games: Evaluating Mobile Gestures in the Wild. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction (ICMI '13)*. ACM, New York, NY, USA, 173–180. https://doi.org/10.1145/2522848.2522874

[97] Jacob O. Wobbrock, Htet Htet Aung, Brandon Rothrock, and Brad A. Myers. 2005. Maximizing the Guessability of Symbolic Input. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*. ACM, New York, NY, USA, 1869–1872. https://doi.org/10.1145/1056808.1057043

[98] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined Gestures for Surface Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1083–1092. https://doi.org/10.1145/1518701.1518866

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

[100] Dianna Yim, Garance Nicole Loison, Fatemeh Hendijani Fard, Edwin Chan, Alec McAllister, and Frank Maurer. 2016. Gesture-driven Interactions on a Virtual Hologram in Mixed Reality. In *Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces, ISS Companion '16, Niagara Falls, Ontario, Canada, November 6-9, 2016*, Mark Hancock, Nicolai Marquardt, Johannes Schöning, and Melanie Tory (Eds.). ACM, 55–61. https://doi.org/10.1145/3009939.3009948

Received March 28, 2018; revised July 24th, 2018; revised October 26th, 2018; revised December 17th, 2018; revised January 16th, 2019; accepted February 4th, 2019