Gesture Elicitation as a Computational Optimization Problem

Theophanis Tsandilas, Pierre Dragicevic

Gesture Elicitation as a Computational Optimization Problem

Theophanis Tsandilas
Université Paris-Saclay, CNRS, Inria, LISN
Orsay, France
theophanis.tsandilas@inria.fr

Pierre Dragicevic
Université de Bordeaux, CNRS, Inria, LaBRI
Bordeaux, France
pierre.dragicevic@inria.fr

ABSTRACT

Gesture elicitation studies are commonly used for designing novel gesture-based interfaces. There is a rich methodology literature on metrics and analysis methods that helps researchers understand and characterize data arising from such studies. However, deriving concrete gesture vocabularies from this data, which is often the ultimate goal, remains largely based on heuristics and ad hoc methods. In this paper, we treat the problem of deriving a gesture vocabulary from gesture elicitation data as a computational optimization problem. We show how to formalize it as an optimal assignment problem and discuss how to express objective functions and custom design constraints through integer programs. In addition, we introduce a set of tools for assessing the uncertainty of optimization outcomes due to random sampling, and for supporting researchers' decisions on when to stop collecting data from a gesture elicitation study. We evaluate our methods on a large number of simulated studies.

CCS CONCEPTS

• **Human-centered computing** → **Interaction design process and methods**; **Gestural input**; *User centered design*; *HCI theory, concepts and models.*

KEYWORDS

Gesture elicitation, design optimization, assignment problems, optimization uncertainty, learning curves, bootstrap

ACM Reference Format:
Theophanis Tsandilas and Pierre Dragicevic. 2022. Gesture Elicitation as a Computational Optimization Problem. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3491102.3501942

1 INTRODUCTION

A fundamental problem of interaction design is how to effectively map computer actions to user input. This problem is especially interesting in the context of gesture-based user interfaces where input can combine multiple modalities and take a variety of different shapes. When designing gestural input, the range of legitimate mappings can be extremely large. A method for dealing with this problem is to look for intuitive mappings that naturally emerge. *Gesture elicitation studies* [45, 46] are commonly used for this purpose. By asking users to perform their own gestures to activate a set of computerized actions (or commands), the method aims to derive mappings of high *guessability* [45], that is, users can guess which gestures trigger each action without prior learning.

Gesture elicitation studies have been extremely popular in HCI research. Villarreal-Narvaez et al. [41] identify 216 gesture elicitation studies published between 2009 and 2019 covering a wide range of interactive applications. However, although Wobbrock et al. [45] present gesture elicitation as a guessability maximization problem, gesture elicitation is most often viewed as an informal design approach that combines a mix of design criteria and ad hoc analysis strategies. Interestingly, the original guessability maximization heuristic of Wobbrock et al. [45] remains intact until today, despite the fact that it is not optimal (as we show in this paper) and cannot deal with custom design constraints, such as allowing selected gestures to be grouped together. An additional limitation of existing research on gesture elicitation is the absence of statistical tools that could help assess how many participants are sufficient for a gesture elicitation study. Villarreal-Narvaez et al. [41] report that the most frequent choice is 20 participants. The authors attribute this choice to the example of the well-cited study by Wobbrock et al. [46]. However, this number is not based on any statistical model or evidence. Would results be the same or at least similar if a new sample of users was used? Unfortunately, statistical tools that determine the sample size of traditional HCI experiments [13] or assess the uncertainty or robustness [11] of their outcomes are not relevant to gesture elicitation studies.

We present solutions to the above problems. We return to the roots of the original guessability maximization approach of Wobbrock et al. [45] but revitalize it with a fresh optimization framework that replaces their heuristic, accommodates custom design constraints, and provides tools for assessing the stability and convergence of optimal solutions as the sample size increases. As with sequential analysis [42], we do not require researchers to fix the sample size in advance. Depending on the evolution of guessability error metrics, the researchers may decide to cease data collection. We base our approach on well-studied methods borrowed from the fields of operations research, machine learning, and statistics, namely combinatorial optimization [27, 28], cross-validation [8], learning curves [25], and bootstrap methods [12]. In summary, we make the following key contributions:[1]

(1) We frame gesture elicitation as a combinatorial assignment problem. We explain how its common formulation, where each computer action (referent) is assigned to a different unique gesture class (sign), can be optimally solved by the Hungarian algorithm [21]. We demonstrate that the heuristic of Wobbrock et al. [45] is not optimal. However, we also show that it generally approximates well the optimal solution.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

[1] Code, case studies, and other materials are available at: https://gelicopt.github.io

(2) We investigate how to support custom design constraints that allow for arbitrary groupings of gestures by expressing the optimization problem as an integer program [27]. We describe an algorithm of constraint aggregation [31] that helps solve practical instances of the optimization problem within a reasonable time.
(3) We formulate gesture elicitation as a machine learning problem, where participants' data are used to train an optimal vocabulary of gestures. We then use cross-validation [8] and a learning-curve sampling method [23] to observe the evolution of guessability error as the sample size increases and further assess whether the number of participants is sufficiently large.
(4) We introduce a bootstrap method [12] for assessing the uncertainty of the optimal mappings produced by a gesture elicitation study. Our method enables researchers to test the robustness of the optimization outcome and communicate its variability. We experimentally demonstrate the good behavior of the method for the Hungarian algorithm.

To evaluate our methods, we develop a computational approach that enables us to generate large amounts of synthetic gesture elicitation data. But in addition, we demonstrate our approach with real data.

2 RELATED WORK

Our work builds upon two main areas of HCI research: gesture elicitation and computational interaction.

Gesture Elicitation. Gesture elicitation studies became extremely popular [41] following the study by Wobbrock et al. [46] on surface gestures. In a gesture elicitation study, participants are shown a set of *referents* (commands or effects of actions) and are asked to propose *gestures* that effect these referents [46]. However, the key concepts of gesture elicitation were introduced four years earlier by Wobbrock et al. [45]. This work presents the method as a guessability maximization problem and defines guessability as:

> *"That quality of symbols which allows a user to access intended referents via those symbols despite a lack of knowledge of those symbols"* [45]

where the term *symbol* (or *sign* [46]) refers to an identifier of a gesture or a groups of gestures. Wobbrock et al. [45] also describe a heuristic solution to the optimization problem that eliminates *conflicts*, i.e., assignments of the same symbol to multiple referents. Although this original version of the method has not changed since, HCI researchers often use their own mix of criteria to resolve conflicts, group gestures together, and produce a final gesture vocabulary. For example, Chan et al. [7] grouped the observed finger micro-gestures into larger categories but then *"looked at each instance of the consensus gesture for each referent."* Rusnák et al. [33] report that they *"took inspiration from the data, while also paying particular attention to the coherence of the proposed interactions"* to deal with conflicts. Dingler et al. [10] describe their own ad hoc strategy to ensure consistency across different devices. Unfortunately, such approaches are difficult to replicate, inflate the risk of research bias [38], and can result in poor generalization due to overfitting. Our goal is to integrate custom design constraints into the optimization problem itself, detaching them from the analysis of the actual data.

Wobbrock et al. [45] set guessability as the target optimization measure. However, guessability scores are rarely (or never) reported in papers, and as Villarreal-Narvaez et al. [41] report, agreement measures [38, 40, 45] have largely dominated the analysis of gesture elicitation data. Although we draw inspiration from Tsandilas' [38] computational modeling approach, we focus instead on the original guessability measure, as we view agreement statistics as a complementary analysis method that does not target optimization. We discuss this issue in more depth in Section 7. The HCI literature has introduced additional usability measures, such as *"endurance"* metrics for characterizing fatigue [18, 32]. For the sake of simplicity, we do not address them here but defer their study to future work.

Computational Interaction. Our work belongs to the broad area of *computational interaction*, whose goal is to use *"algorithms and mathematical models to explain and enhance [human-computer] interaction"* [4], often with the intent of developing tools to assist interaction designers. Computational interaction has a long history, but has recently received increased attention partly due to the complexification of user interfaces and the availability of powerful computational methods [4].

Much of computational interaction involves the use of *combinatorial optimization* methods to help design interfaces that are optimal according to some criteria [27, 28]. Many contributions in this area involve solving layout problems, such as finding optimal keyboard layouts, GUI widget layouts, or menu layouts [2, 5, 28]. These problems can often be formulated as *assignment problems*, which is a common class of optimization problem that often arises in HCI and makes up much work in computational interaction [27, 28].

In this article, we formulate the problem of finding an optimal gesture set in elicitation studies also as an assignment problem, where the goal is to find an optimal mapping between gestures and referents. Despite the wealth of work in both gesture elicitation and computational interaction, we are not aware of any past work that explicitly connects the two areas by formulating the problem of finding optimal gesture sets as an assignment problem.

One of the most related contributions is the work from Sridhar et al. [37], who introduced an algorithm for finding optimal in-air multi-finger gestures according to four usability metrics: performance, anatomical comfort, learnability and mnemonics. Although the broad notion of learnability relates to gesture guessability (our metric of interest), their work differs from ours in several respects. As we optimize for a single metric (guessability), our procedure is simpler [26]. At the same time, our work is more general because it does not focus on a specific class of gestures (in-air multi-finger gestures for Sridhar et al. [37]). In addition, our objective function is strongly empirical (based on data directly provided by users), while the approach by Sridhar et al. [37] is primarily theoretical (based on theories of motor learning). Clearly, the two approaches are complementary and can be combined in future work.

Our work also relates to *decision-theoretic optimization* in HCI, an early body of work in the area of computational interaction whose goal was to *"minimize the expected cost of a user's interactions or (equivalently) to maximize the user's expected utility"* [16]. Since our method maximizes the probability of guessing the correct gesture, it can be seen as using decision-theoretic optimization. However,

we are not aware of any contribution in this stream of work that addresses the problem of finding optimal gesture sets.

Our contribution can be seen as addressing a simpler computational interaction problem in comparison to much of previous work in this area, but the clarifications and solutions it offers can potentially have a large impact due to gesture elicitation studies being extremely numerous in HCI. In addition, our work also tackles difficult related questions, such as how to optimize for a population based on data from a sample, an issue we have not seen discussed in previous work in computational interaction. In distinction to stochastic optimization approaches [34] that aim to find an optimal solution under uncertainty, our goal is to quantify the uncertainty of the optimal solution based on a sample, and further help investigators determine the size of the sample. To this end, we use well-established machine learning and statistical tools that can be easily mastered by novices.

3 BASE TERMINOLOGY

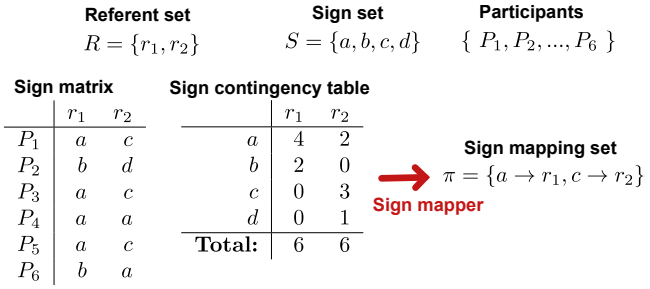
The goal of the gesture elicitation method is to design intuitive, self-discoverable gestural vocabularies for a target *population* of users, e.g., novice users of tablets or young gamers. The participants of a gesture elicitation study are generally assumed to be randomly sampled from the target population, thus they are representative of this population. Wobbrock et al. [46], Tsandilas [38], and Villarreal-Narvaez et al. [41] provide an in-depth analysis of the gesture elicitation methodology and its terminology. The purpose of the following definitions is to make this terminology coherent with our analysis. Figure 1 illustrates the key terms of our analysis for a hypothetical study with six participants and two referents.

Gesture description: *The digital and/or analog recording of a gesture performed by a participant of a gesture elicitation study* (based on Tsandilas [38]).

Gesture dataset: *The dataset that consists of all gesture descriptions collected in a gesture elicitation study, where each gesture description is associated with a participant and a referent.* In some gesture elicitation studies [19, 24], participants are required or encouraged to propose multiple gestures per referent. In this paper, we simplify our analysis by assuming that each participant proposes a single gesture per referent.

Sign: *A label or symbol that identifies a type of gesture. More generally, it can be thought of as a class (or category) of "equal" or "similar" gestures* [38, 46]. Wobbrock et al. [45] and Villarreal-Narvaez et al. [41] refer to signs as *symbols.*

Gesture classification process: *A process that takes as input a gesture description and outputs a sign.* Although it can be implemented as a computer algorithm (e.g., as a regular classifier), for most gesture elicitation studies, it is a manual process [38], where human coders (e.g., the investigators) classify the gesture descriptions of a gesture dataset into signs. Coders often use human-readable instructions in the form of a code book to complete this task. Thus, a gesture classification process may not be deterministic, i.e., for the same input it may not always output the same sign. Nevertheless, for our analysis, we will make the assumption that it is deterministic and can be expressed as a function that associates gesture descriptions to signs.



	r_1	r_2
P_1	a	c
P_2	b	d
P_3	a	c
P_4	a	a
P_5	a	c
P_6	b	a

	r_1	r_2
a	4	2
b	2	0
c	0	3
d	0	1
Total:	6	6

Figure 1: A hypothetical gesture elicitation study, where six participants perform gestures for two referents. In this example, the participants' gesture descriptions have been classified into four distinct signs: *a*, *b*, *c*, and *d*.

Sign set: *The set of unique signs produced when applying a gesture classification process to a gesture dataset.*

Sign matrix: *A matrix containing the sign of each participant's gesture for each referent. It derives by applying a gesture classification process to a gesture dataset.* If n is the number of participants and m is the number of referents, the size of the sign matrix is $n \times m$. See example on Figure 1.

Sign contingency table: *A summary table that captures the frequency distribution of signs across referents.* If q is the number of signs (i.e., the size of the sign set) and m is the number of referents, then the size of the table is $q \times m$.

Sign mapping: *A relationship $\sigma \to r$ that associates a sign σ with a referent r.* We focus on one-to-one and many-to-one (i.e., functional) mappings between signs and referents, but we also discuss the case where a sign can be associated with more than one referents. When multiple signs $\sigma_1, ..., \sigma_\kappa$ are mapped to the same referent r, we will use the more compact notation $\{\sigma_1, ..., \sigma_\kappa\} \to r$.

Sign mapping set: *A set of mappings between signs in a sign set and the referents in a referent set.* It is usually the final outcome of a gesture elicitation study. It is also known as the *user-defined gesture set* [46] or the *consensus gesture set* [41]. This is the entity that we are optimizing in this article.

Sign mapper: *A deterministic algorithm that produces a sign mapping set based on the results of a gesture elicitation study.* In the simplest case, the only input of sign mapper is a sign matrix or a sign contingency table. In the more general case, a sign mapper can take as input additional information, such as constraints on the way signs can be mapped to referents.

4 DESIGN OPTIMIZATION PROBLEM

We express gesture elicitation as a design optimization problem, where our goal is to find a set of optimal sign mappings that satisfy some design constraints. We make the assumption that the gesture classification process is defined before the analysis starts. This assumption is fundamental because it ensures that the classification process is independent of the dataset and will remain the same, irrespective of the given sample. Given this assumption, we can apply the gesture classification process on a gesture elicitation dataset and produce a set of signs. We now have to answer the following question. What is the optimal way of mapping these signs to the referents?

4.1 Formalizing the Problem

The sign mapping set $\pi = \{a \rightarrow r_1, c \rightarrow r_2\}$ in Figure 1 is consistent with 58.3% (7 out of 12) of all proposals. If we constrain our design solution so that no more than one sign is mapped to each referent, then no other solution gives a better score. More generally, the optimization problem consists of: (i) a set of *design constraints* and (ii) an *objective function*. Our goal is to find a *sign mapper* (see Section 3) that optimizes the objective function while respecting all design constraints. We can then apply the sign mapper to the *sign matrix* to produce optimal *sign mappings*.

4.1.1 Notation. Let R denote a referent set, S denote an elicited sign set, and π denote a sign mapping set. If σ_k is a sign in S, then $\pi(\sigma_k)$ is the set of referents to which σ_k is mapped. Also, if r_i is a referent in R, then $\pi^{-1}(r_i)$ is the set of signs mapped to this referent. For the sign mapping set in Figure 1, we have that $\pi(a) = \{r_1\}$, $\pi(b) = \emptyset$, and $\pi^{-1}(r_2) = \{c\}$.

4.1.2 Design Constraints. We express design constrains as constraints on the mappings between signs and referents. A common approach is to allow for no more than one mapping per sign [45, 46], or more formally: $|\pi(\sigma_k)| \leq 1, \forall \sigma_k \in S$. This constraint ensures that a gesture can trigger a unique system command. However, we may instead assume that the final user interface uses additional modalities to infer the right command. So we may prefer to not enforce the above constraint. It is also possible to map multiple signs to the same referent. For example, an interaction designer may decide that a "short press" and a "long press" could both trigger the same action. Since only few sign combinations may make sense for a given design problem, we should be able to set custom constraints on which signs can be grouped together.

4.1.3 Objective Function. An *objective function* is a scalar value that defines the optimization objective. An optimal solution maximizes (or minimizes) this value. In the context of gesture elicitation, the best-known objective function is *guessability*. We rewrite its mathematical definition by Wobbrock et al. [45] as follows:

$$G(\pi) = \frac{1}{m} \sum_{i=1}^{m} \frac{n_{\pi i}}{n_i} \quad (1)$$

where π is a sign mapping set, m is the number of referents in R, n_i is the number of participants who performed a gesture for the i^{th} referent, and $n_{\pi i}$ is the number of participants whose gesture sign for this referent belongs to π. If all participants perform gestures for all the referents, n_i is equal to the total number of participants n. $G(\pi)$ can be viewed as the probability that the sign of a gesture performed by a random user for a random referent r_i belongs to $\pi^{-1}(r_i)$. An optimal sign mapping set π_{opt} is the one that maximizes this probability.

4.1.4 Sign Mapper. We first consider the common case where each referent is mapped to exactly one sign. If the same sign can be mapped to more than one referent, a trivial optimal solution is to take the most frequent sign, i.e., the mode, for each individual referent. We will call this sign mapper the *mode mapper*. It does not deal with conflicts (i.e., the same gesture can trigger multiple commands) but we use it as reference for our analyses.

	Signs	Referents r_1	Referents r_2
	a	10	9
	b	8	2
	c	1	3
	d	0	3
	e	1	3
Total:		20	20

Sign Mappings

Heuristic (Wobbrock et al., 2005)	Optimal
$a \rightarrow r_1$ $c \rightarrow r_2$	$a \rightarrow r_2$ $b \rightarrow r_1$
$G(\pi_h) = \frac{10+3}{40} = 32.5\%$	$G(\pi_{opt}) = \frac{9+8}{40} = 42.5\%$

Figure 2: An example for which the heuristic of Wobbrock et al. [45] fails to deduce the optimal mappings. The sign contingency table summarizes the results of a hypothetical gesture elicitation study where 20 participants perform gestures for two referents. Its values represent sign frequencies for each referent. For the two sign-mapping solutions, we also report their percent guessability scores.

For the more useful scenario where each referent is mapped to a single and unique sign, Wobbrock et al. [45] describe a heuristic that aims to maximize guessability. Wobbrock et al. [46], who use it for the analysis of their gesture dataset, summarize it as follows:

> *"The user-defined gesture set was developed by taking the largest groups of identical gestures for each referent and assigning those groups' gestures to the referent. However, where the same gesture was used to perform different commands, a conflict occurred because one gesture cannot result in different outcomes. To resolve this, the referent with the largest group won the gesture."*

Unfortunately, this heuristic does not guarantee an optimal solution. Figure 2 demonstrates a situation for which it fails to infer optimal mappings. We observe that picking the *"largest groups of identical gestures for each referent"* [46] (i.e., the most frequent sign for each referent) is not necessarily the best strategy.

The problem becomes more challenging if we allow for custom constraints, such as multiple signs per referent. For example, if we allow a and b in Figure 2 to be grouped together, the optimal solution is to map both signs to r_1 such that $\{a, b\} \rightarrow r_1$ and $c \rightarrow r_2$. In this case, the guessability score is $G(\pi_{opt}) = 52.5\%$.

We next present solutions to both instances of the problem.

4.2 Basic Assignment Problem

Mapping signs to referents can been seen as an instance of the *assignment problem* [6], a fundamental combinatorial-optimization problem [27]. The most general form of the assignment problem is as follows: a group of workers needs to be assigned to complete a maximum set of tasks, and for each worker and task, there is a cost for assigning the worker to the task, e.g., the cost can be the wage that the worker demands for the task. The problem is to assign each worker to a task, while minimizing the total cost. We can reduce our sign mapping problem to an assignment problem if we replace workers by signs and tasks by referents. But our goal now is to maximize guessability instead of minimizing a cost.

In the most typical instance of the assignment problem, it is required to assign at most one worker per task and at most one task per worker. A well-known algorithm that solves this problem is the Hungarian algorithm [21]. The time complexity of the latest versions of the algorithm is $O(v^3)$, where v refers to the number of vertices in the assignment bipartite graph.

Table 1: Guessability scores (%) for sign the mappings produced by three sign mappers: the mode mapper that does not deal with conflicts, the heuristic of Wobbrock et al. [45], and the Hungarian algorithm [21].

		Conflict-Free	
Gesture Elicitation Studies	Mode	Heuristic	Hungarian
PaperPhone Gestures (Part 3) [22]	50.0	22.0	22.0
Métamorphe Key Shortcuts [3]	28.3	27.9	27.9
Signle-Hand Micro-Gestures [7]	38.0	17.7	17.9
On-Skin Gestures (Modality) [44]	48.4	12.8	13.0

In the context of gesture elicitation, the Hungarian algorithm addresses the common design constraint where at most one sign per referent and at most one referent per sign can be assigned. To apply the algorithm, we first need to turn frequencies in the elicited contingency table into costs by taking their opposite (negative) values. For example, if we use the algorithm for the study in Figure 2, it will return the optimal solution shown in blue.

Table 1 presents the percent guessability scores for the results of four past gesture elicitation studies. For Métamorphe's key shortcuts [3], we assess guessability for combined signs that describe both the key and the gesture of a performed shortcut. For on-skin gestures [44], we ignore the body part and only consider the signs that characterize the input modality, following the analysis of Tsandilas [38]. The mode mapper results in higher guessability scores since it does not deal with sign conflicts. Interestingly, differences in guessability scores between the heuristic of Wobbrock et al. [45] and the Hungarian algorithm are negligible for these datasets. In Section 6, we assess the outcome of the two mappers with simulated data, where we confirm that substantial differences between the guessability results of the two mappers (as in Figure 2) are statistically rare. Such results are good news for studies that have used the sub-optimal heuristic in the past.

4.3 Adding Custom Design Constraints

If we can group signs together, the assignment problem becomes more difficult since the space of possible solutions becomes larger. We can express such problems as *integer programs* [27, 36] – they can be understood as linear optimization problems with the additional requirement that all variables are integers.

In our case, all variables are binary: a variable $x_{ij} \in \{0, 1\}$ designates whether or not the sign σ_j is mapped to the referent r_i. If m is the number of referents and q is the number of observed signs, then we need a total of $m \times q$ variables to describe the problem. We assume that $m < q$ such that there are enough signs for all referents (if this is not the case, we can easily add dummy signs). Given these variables, we can now define our objective function as follows:

$$maximize \sum_{i=1}^{m} \sum_{j=1}^{q} n_{ij} \cdot x_{ij} \tag{2}$$

where n_{ij} is the number of occurrences of σ_j for the referent r_i. The above expression maximizes guessability (see Equation 1). We then describe the optimization constraints. A first set of constraints ensures that each sign is mapped to at most one referent:

$$\sum_{i=1}^{m} x_{ij} \leq 1, \forall j = 1..q \tag{3}$$

This equation defines a total of q linear relationships.

We also need to describe constraints on the number of signs per referent. We proceed in two steps. First, we ensure that each referent is assigned to at least one sign:

$$\sum_{j=1}^{q} x_{ij} \geq 1, \forall i = 1..m \tag{4}$$

This equation defines a total of m linear relationships.

Second, we describe the sign grouping constraints. Let c_k denote a set of signs that can be grouped together, and let $C = \{c_k \mid k = 1..p\}$ denote the set of all such groups. Likewise, let $\bar{c}_k$ be a set of signs, such that any pair of signs within this set cannot be grouped together, and let $\bar{C} = \{\bar{c}_k \mid k = 1..p'\}$ denote the set of all such groups of illegitimate sign pairings. To better understand this notation, consider our example in Figure 2 with the extra condition that the signs a and b can be grouped together. In this case, $C = \{\{a, b\}\}$, since there is a single ($p = 1$) legitimate grouping. From this, we derive $\bar{C} = \{\{a, c\}, \{a, d\}, \{a, e\}, \{b, c\}, \{b, d\}, \{b, e\}, \{c, d\}, \{c, e\}, \{d, e\}\}$, which describes all ($p' = 9$) illegitimate pairs of signs.

The following set of constraints ensures that for each $\bar{c}_k \in \bar{C}$, at most one sign can be mapped to each referent:

$$\sum_{\sigma_j \in \bar{c}_k} x_{ij} \leq 1, \forall i = 1..m \text{ and } \forall k = 1..p' \tag{5}$$

This equation defines a total of $m \times p'$ linear relationships.

The problem is now fully described by Equations 2 - 5. In summary, it includes an objective function and $q + m \times (p' + 1)$ linear relationships over $m \times q$ binary variables. As long as the number of signs is greater than the number of referents, there is always a solution that satisfies all constraints. We can use an integer programming solver such as Google OR-Tools [1] to find an optimal solution. Our implementation is based on R for Operations Research [35].

4.3.1 Constraint Aggregation. Integer programming is NP-complete. It is a hard problem because many combinations of integer (binary in our case) values must be tested, while the number of combinations can rise exponentially with the size of the problem. The size of the problem, in turn, depends on the number of variables and the number of constraints. The general problem of constraint reduction in optimization models is known as *model aggregation* [31]. Although approximate model aggregation methods that provide close-to-optimal solutions also exist, we are only interested here in exact methods. In the context of integer programming, such methods aim to derive an equivalent but easier-to-solve problem by aggregating the original constraints into one or more *surrogate* constraints [17, 31]. Turning back to our problem, our goal is to derive a minimal number of surrogate constraints for Equation 5 (i.e., minimize p') and ensure that all instances of our problem can be solved in reasonable time.

Before we describe an algorithm for this purpose, we provide some intuition. Consider again the example we presented earlier, where $C = \{\{a, b\}\}$ and $\bar{C} = \{\{a, c\}, \{a, d\}, \{a, e\}, \{b, c\}, \{b, d\}, \{b, e\}, \{c, d\}, \{c, e\}, \{d, e\}\}$. How can we reduce the number of

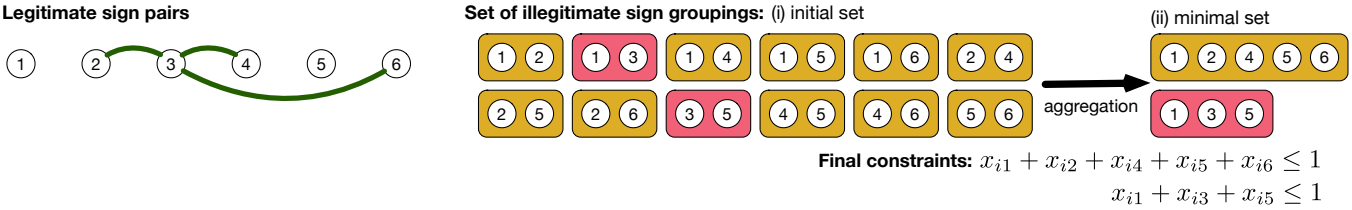


Figure 3: Constraint aggregation for a sign set with six signs and three legitimate pairs (in green). The algorithm starts with an initial set ($\bar{C}$) of 12 illegitimate sign pairs and aggregates them to a set ($\bar{C}_{min}$) of two groups that correspond to two ($p' = 2$) inequality constraints. Notice that a sign may appear in multiple groups.

groupings in $\bar{C}$? A first observation is that some of these pairs can be merged together. For example, we can merge $\{a, c\}$, $\{a, d\}$, and $\{c, d\}$ into $\{a, c, d\}$ and still ensure that no combination of these three signs can be mapped to the same referent. This constraint reduction can be described by the following algebraic expression of logical equivalence:

$$x_a + x_c \leq 1 \text{ and } x_a + x_d \leq 1 \text{ and } x_c + x_d \leq 1 \iff x_a + x_c + x_d \leq 1 \quad (6)$$

where x_a, x_c, and x_d are all binary variables. If we continue merging, we can eventually reduce the original set $\bar{C}$ to its surrogate set $\bar{C}_{min} = \{\{a, c, d, e\}, \{b, c, d, e\}\}$ that contains a minimal number of sign groups.

Algorithm 1 presents a general solution. It takes as input the set C of all legitimate sign pairs. From those, it derives all illegitimate sign pairs $\bar{C}$ and incrementally merges them. Figure 3 graphically describes the algorithm's solution for an additional example. Although Algorithm 1 has a computational cost, it is marginal compared to the cost of solving the optimization problem. In most cases, a smaller number of constraints results in huge time saving.

Algorithm 1: Constraint aggregation algorithm

input : Elicited sign set S and legitimate sign pairs C
output: Minimal set of illegitimate sign groupings $\bar{C}_{min}$

```
C̄ ⟵ pairs(S) − C;   /* Get illegitimate pairs */
C̄_min ⟵ ∅;          /* Initialize output */
for c̄_0 ∈ C̄ do
    C̄ ⟵ C̄ − c̄_0 ;   /* Remove c̄_0 from C̄ */
    for c̄_1 ∈ C̄ do
        I ⟵ c̄_0 ∩ c̄_1 ;   /* Intersect c̄_0 and c̄_1 */
        if pairs(I) ∩ C = ∅ then
            C̄ ⟵ C̄ − c̄_1;   /* If no legitimate pairs
             exist, remove c̄_1 from C̄ */
            c̄_0 ⟵ I;   /* and update c̄_0 */
        end
    end
    C̄_min ⟵ C̄_min ∪ c̄_0;   /* Add c̄_0 to C̄_min */
end
```

4.3.2 Case Study. Weigel et al. [44] conducted a user study to elicit on-skin gestures for 40 referents. We focus here on a subset of 33 referents that represent standard commands and variations and disregard ones that represent emotions. The data classification process of Weigel et al. [44] considered the *on-body location* of gestures (e.g., upper arm, forearm, palm, and fingers) and their *input modality* (e.g., "slide" "tap", and "twist"). Combining these two dimensions to derive an optimal vocabulary can be tricky. If we only focus on modalities, we miss information about participants' preferences of gestures' location in interaction with their modality. If we instead consider all possible combinations of locations and modalities, observed signs can be too variable, leading to mappings of very low guessability.

Alternatively, we can use the full information but relax the way signs are mapped to referents through custom constraints. For example, we can define legitimate pairs of signs that share the same modality but are executed in different yet related parts of the body. We consider three larger groups of body locations to infer legitimate pairs of signs: (i) upper parts of the arm and the shoulder, (ii) the palm and the fingers, and (iii) the wrist and the back of the hand, with or without fingers. This approach produces 7000 constraints for Equation 5 but Algorithm 1 reduces this number to 22.

Another design aspect that we want to capture is the fact that the "slide" modality provides a richer set of possibilities. We focus here on binary sliding directions (upwards vs. downwards) so we allow for up to two referents to share the same "slide" sign. To express this constraint, we change Equation 3 to $\sum_{i=1}^{m} x_{ij} \leq 2$ for all signs j for which the input modality is "slide".

If we run the optimization with the original 7000 constraints of Equation 5, the optimization process does not terminate within any reasonable time. But our R implementation of constraint aggregation solves the optimization problem in 36 seconds on a regular laptop. Table 2 shows the sign mappings for seven representative referents produced by three alternative approaches: (i) the mode mapper, (ii) the Hungarian algorithm, and (iii) our custom optimizer with relaxed constraints. The mode mapper results in high guessability (30.0%) but very low variability. In particular, the "[forearm]slide" sign is mapped to 22 out of the 33 referents. The Hungarian algorithm leads to mappings with very low guessability (11.8%) and little consistency, e.g., Zoom-in and Zoom-out are assigned to very different modalities. The custom optimizer provides a more balanced solution and raises guessability to 15.4%.

5 SAMPLING UNCERTAINTY

The two sets of mappings produced by the two sign mappers (heuristic and optimal) in Figure 2 are based on the gestures of 20 participants. To what extent are such mappings optimal for the target user population? What is a sufficient number of participants? Can we expect that our optimal solution would remain the same if we elicited gestures from a different sample of participants?

Table 2: Sign mappings of three optimization approaches for representative referents in the study by Weigel et al. [44]

	Rotate	Zoom-in	Zoom-out	Shrink	Minimize	Undo	Redo
Mode	[palm]slide	[forearm]slide	[forearm]slide	[forearm]slide	[palm]slide	[forearm]slide	[forearm]slide
Hungarian	[forearm]twist	[forearm]slide	[fingers]push-press	[palm]slide	[forearm]slide+tap	[forearm]slap	[upperarm]sheer
Custom	[forearm]twist	[forearm]slide	[forearm]slide	[palm]slide	[palm|palm+fingers]slide	[backhand|wrist]tap	[palm|palm+fingers]slide-swipe

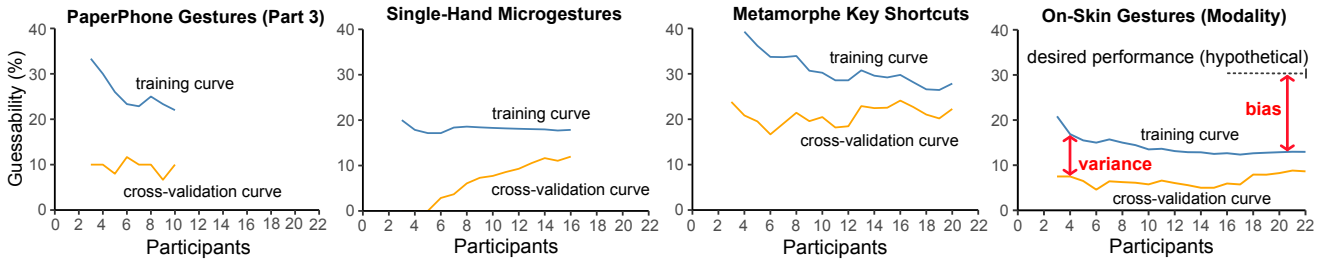


Figure 4: Learning curves for four real gesture elicitation studies. We use the Hungarian algorithm to derive optimal sign mapping sets and the leave-one-out cross-validation (LOOCV) method. The last graph illustrates the variance and bias errors.

5.1 Learning-Curve Sampling Method

We treat our problem as a classical machine learning problem, where the goal is to sufficiently train a classification model in order to optimize a performance score. Certain machine learning approaches maximize classification accuracy, while others minimize error or loss. Our goal, here, is to sufficiently train a sign mapper to maximize guessability.

5.1.1 Problem Formulation. Consider the following fictitious scenario. We recruit the full population of target users to participate in a gesture elicitation study. We use a gesture classification process f_C to classify gestures into signs and apply an optimal sign mapper f_M. The mapper produces a sign mapping set π_{opt} with guessability $G(\pi_{opt})$. Now, suppose we take a random sample of n participants from the above population and apply again the same classification process f_C and sign mapper f_M. The result is a sign mapping set $\widehat{\pi}_{opt}$ that is optimal for this specific sample of participants but not necessarily optimal for the full user population. Thus, $\widehat{\pi}_{opt}$ is generally different than π_{opt}. Nevertheless, for a large enough sample size n, we expect that it will be identical or close to π_{opt}, or at least, it will result in a similar guessability score. Since recruiting participants has a cost, we have to decide about which number of participants is large enough.

Suppose we evaluate $\widehat{\pi}_{opt}$ on the full user population. Its population guessability score will be $G(\widehat{\pi}_{opt}) \leq G(\pi_{opt})$ but we expect it to asymptotically increase as we add more participants. Our goal is to find a large enough n such that the difference from the target guessability score is below an acceptable error threshold ϵ_G:

$$G(\pi_{opt}) - G(\widehat{\pi}_{opt}) \leq \epsilon_G \tag{7}$$

Unfortunately, neither $G(\pi_{opt})$ nor $G(\widehat{\pi}_{opt})$ is known. We can only calculate the sample guessability $\widehat{G}(\widehat{\pi}_{opt})$. The following paragraphs examine the relationship of these three quantities.

5.1.2 Learning Curves. A common challenge for machine learning approaches is how to assess whether training has reached its potential, or whether adding additional training examples can meaningfully improve performance. *Learning curves* are essential tools for this purpose. A learning curve describes performance as a function of the sample size of the training data, where typically, performance approaches some limiting behavior. The idea of a *learning-curve sampling method* [23], from which we get inspiration, is *"to iteratively apply a training algorithm to larger and larger subsets of the data, until the future expected costs outweigh the future expected benefits associated with the training"* [23]. For gesture elicitation studies, the goal is to decide whether the cost of recruiting additional participants outweighs the benefit of expected guessability improvements.

We explain the method for the four real datasets of Table 1. The *training curve* (in light blue) in Figure 4 represents the evolution of the sample guessability $\widehat{G}(\widehat{\pi}_{opt})$ as the number of participants grows. Since we use the same sample of participants both to train and to evaluate the sign mapping set $\widehat{\pi}_{opt}$, $\widehat{G}(\widehat{\pi}_{opt})$ is generally greater than $G(\widehat{\pi}_{opt})$, which is the population guessability of $\widehat{\pi}_{opt}$. Therefore, training curves tend to overestimate guessability and can be thought off as rough upper bounds. The score $G(\widehat{\pi}_{opt})$, in turn, is unknown but we can estimate it with cross-validation methods. The goal of cross-validation is to test the sign mappings on new data that were not used for training.

We use the *leave-one-out cross-validation* (LOOCV) method [8]. Closely related to the jackknife resampling technique [30], it works as follows. The learning algorithm is applied once for each instance of the sample, using all other instances as a training set and the selected instance as a single-item test set. In our case, each instance is the data of a different participant, while the learning algorithm corresponds to the sign mapper. The LOOCV process is especially appropriate when the sample is small, and the need for accuracy outweighs the computational cost of the method. The cross-validation curves in Figure 4 (in orange) were derived with this method.

5.1.3 Overfitting. Figure 4 shows that as more participants are added, the training and cross-validation curves approach each other. In machine learning practice [25], the difference between the two curves is known as *variance error* and reflects *overfitting*, or the sensitivity of a model to small changes in the training set. In our case, we expect that variance error will decrease and will eventually become zero as the number of participants increases. Therefore, a large but decreasing variance error indicates an inadequate sample

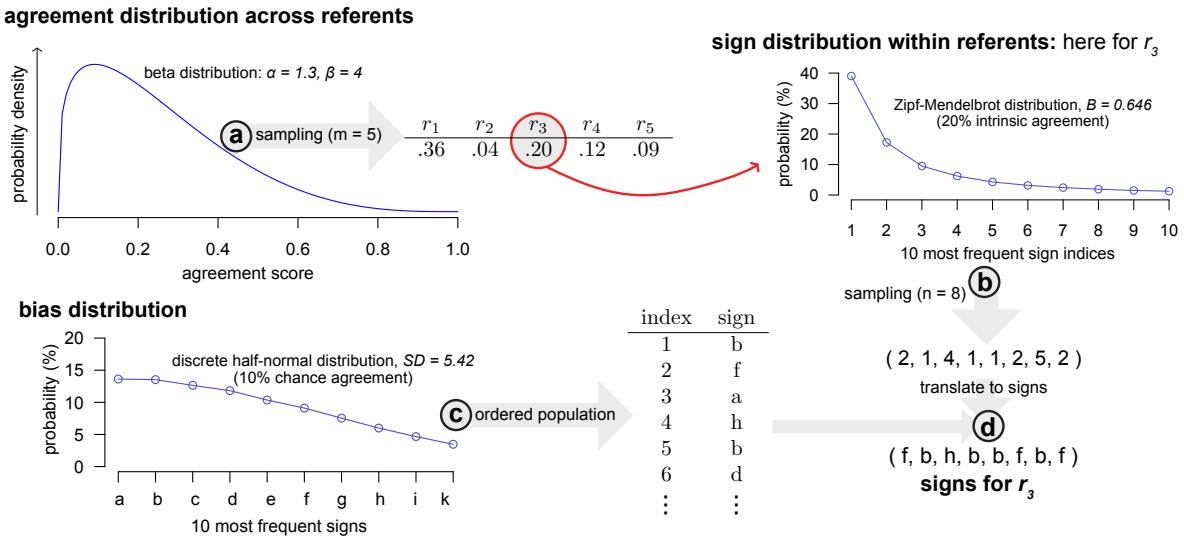


Figure 5: Summary of our sign-generation approach. In this example, we simulate a gesture elicitation study with $n = 8$ participants and $m = 5$ referents. We detail the generation of signs for the third referent r_3.

size. For example, we observe that variance error is large for the first study [22] (PaperPhone gestures), which includes 10 participants only. For the second study [7] (single-hand microgestures), the error is lower but still evolving. Hence, it would seem reasonable to recruit additional participants.

5.1.4 Underfitting. Another quantity of interest is the difference between a hypothetical desired performance and the training score, known as the *bias error* (see Figure 4). The bias error does not improve and can further increase with more data. In machine learning problems, high bias reflects *underfitting*, i.e., the model does not capture well the essential parameters of the dataset. In gesture elicitation studies, high bias translates into low guessability scores. There are many possible reasons for low guessability. In particular, mappings between the gestural modality of interest and the referents under investigation may not naturally emerge. For example, participants may not be able find meaningful associations between on-skin gestures and standard mobile commands [44]. The learning-curve sampling method can help us diagnose such problems early enough and stop wasting participants. Alternatively, the investigators may decide to refine their gesture classification process or relax the design constraints given to the signer mapper to better capture common patterns in a dataset (see Section 4.3). However, increasing the degrees of freedom in the way distinct and perhaps loosely related gestures are associated with each other can lead to overfitting and, therefore, to a higher variance error. More generally, there is a well-known trade-off between bias and variance [8, 25].

5.1.5 Computational Modeling. Learning curves are often noisy. To support their interpretation, we need statistical models that describe their relationship and the uncertainty of the their behavior over time. Creating models with analytical methods is difficult so we follow a computational approach. We rely on Tsandilas' [38] probability distributions for modeling consensus among participants, but we further extend his approach to generate the full sign matrix of simulated studies from user populations with variable *bias* and *intrinsic agreement* characteristics. Figure 5 summarizes our method with an example, where we show that we combine three distinct distributions to generate a sample of sign proposals. The interested reader can find the details of our method in the appendix of our supplementary material: https://osf.io/5w6y2.

By varying the parameters of our distribution models, we generated a dataset (see supplementary material) that reports population parameters and sample statistics (guessability scores, agreement statistics, etc.) for a large number of simulated studies. Figure 6 presents three examples. In addition to the training and cross-validation curves, we show the curves (dashed lines) of the population guessability G computed for both π_{opt} and $\widehat{\pi}_{opt}$. The bottom graphs show their difference, where the grey lines represent the guessability error that we want to minimize (see Equation 7). This error is unknown but is generally roughly half of the difference between the training and cross-validation curves. We can predict it more accurately with models trained on our dataset. The bottom blue curves in Figure 6 are predictions based on a simple linear regression model with two only predictors: the difference between the training and cross-validation scores and n.

Figure 7 presents two additional examples but we now take averages over a 100 sampling iterations. We observe that the cross-validation score is on average very close to $G(\widehat{\pi}_{opt})$ and could serve as its best guess. However, the LOOCV method uses sub-samples of size $n - 1$ for training, which explains why cross-validation scores are slightly biased towards lower values when the sample size is small. For the above examples, we also observe that guessability improves fast for up to 20 or 25 participants, while recruiting more than 30 participants does not result in substantial benefits. Section 6 provides additional results in this regard.

5.2 The Uncertainty of Sign Mapping Sets

In addition to guessability scores, we may also want to assess the uncertainty of the sign mappings themselves. HCI authors have

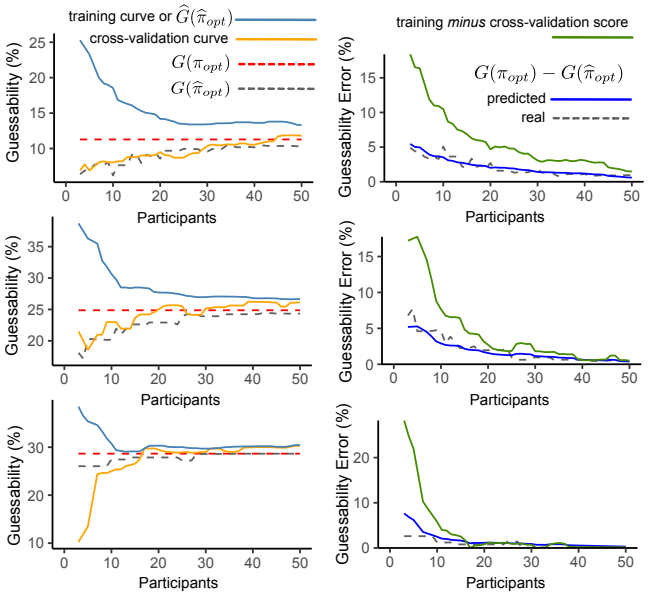


Figure 6: Three examples of simulated studies, where optimization is performed with the Hungarian algorithm. The left graphs show the true but unknown guessability scores $G(\pi_{opt})$ and $G(\widehat{\pi}_{opt})$ (dashed curves), as well as the training and cross-validation curves (solid curves). The right graphs show the difference between the training and cross-validation curves (in green) and the guessability error, both the predicted (in blue) and the real one (dashed line).

used various ad hoc approaches to express the uncertainty of their final mappings. Bailly et al. [3], for example, present more than one *key + gesture* combinations for some referents to account for situations for which there is no clear winner. Other authors like Weigel et al. [44] present a single sign per referent but also report the frequency of the winning signs. Such methods help better assess the quality of a proposed sign mapping set with respect to the observed sign frequencies in the given sample. However, they do not consider the statistical uncertainty of the sample itself and, by extension, how this uncertainty affects the results of the optimization process.

As we discussed in Section 4, solving an assignment problem can be a complex optimization process that is subject to many constraints. Depending on the distribution of signs across different referents as well as the size of the sample, the result of the optimization (i.e., the sign mapping set) can be partly or fully unstable. Our goal is to provide a systematic method that helps evaluate the variability of the optimization result and, if necessary, consider alternative solutions. Table 3 shows an example of how to communicate the variability of optimal sign mappings for Métamorphe's key shortcuts [3]. In this example, we reproduce Table 2 of the original publication [3], but we now use the Hungarian algorithm to find the optimal sign mapping set. Furthermore, we communicate our "confidence" about which signs are optimal given the uncertainty of the sample. For example, one can be confident that for several commands (referents), such as "Align Top", "Align Bottom", "Align Right", and "Rotate", the optimal sign would not likely change even

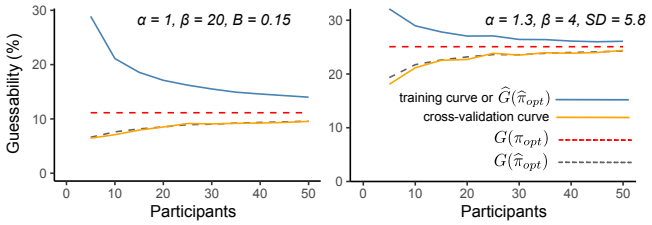


Figure 7: Learning curves for two simulated populations, where $m = 20$ referents. For each graph, we show the distribution parameters we used. We generate sign proposals from a large population of participants and then re-sample from this population. The curves are averages over 100 iterations calculated for intervals of every five participants. Optimization is performed with the Hungarian algorithm.

if the full user population participated in the study. For other commands, however, such as "Align Justify", "Shrink", "Move a Little", and "Play", the optimal sign seems to be extremely variable and thus largely uncertain.

Our approach for assessing uncertainty relies on the bootstrap method [12]. The bootstrap is a versatile statistical inference method that approximates the sampling distribution of a statistic (e.g., the mean or median) by iteratively resampling the original sample *with replacement*. It is commonly used when no analytical methods exist. The main difficulty for our problem is the fact that we do not target the sampling distribution of a statistic but rather the sampling distribution of the discrete solutions of an optimization problem. Still, the bootstrap method can be applied in a similar way but its outcome requires some careful interpretation. In particular, the bootstrap method assumes that the bootstrap distribution converges to the correct limit distribution as the size of the sample increases. This can be hard to prove for arbitrary optimization problems. Nevertheless, as Wasserman [43] explains, although there are many cases where the bootstrap *"is not formally justified"*, the method can be used *"in an informal way to get some intuition of the variability of the procedure"* [43]. As a representative example, Diaconis and Efron [9] have applied the bootstrap method to visually evaluate the variability of rainfall acidity maps. In a different application domain, the method has helped assess the uncertainty in the inference of phylogenetic trees from data on species [20]. As Holmes points out, the method can serve as a measure of robustness: *"Could a small plausible perturbation of the data give a different result?"* [20]. Likewise, our goal is to assess the robustness of the optimal sign mappings and evaluate to what extent we can trust them.

The way we apply the bootstrap method is straightforward. We randomly resample with replacement from the *n* participants of the study and create a new sample of size *n* such that the same participant may appear multiple times in the new sample. We then run optimization on this sample and derive an optimal set of sign mappings. We repeat this process a large number of times, and at the end, we aggregate the optimization results. For each referent, we obtain a distribution of signs (bootstrap distribution) like in Table 3. Our hope is that this distribution approximates the sampling distribution that we would derive if we repeatedly sampled from the full population of participants. Thus, the frequency of a sign

Table 3: Communicating the variability of optimal sign mappings for Métamorphe's key shortcuts [3].

Command	Top Keys + Gestures	Command	Top Keys + Gestures	Command	Top Keys + Gestures
Align Top	**A-Away** (98%)	Save	**S-Top** (100%)	Previous	**P-Left** (59%), <-Top (20%), backspace-Left (13%)
Align Bottom	**A-Towards** (96%)	Save All	**S-Pull** (31%), S-LR (29%), S-FB (15%), S-Left (12%)	Next	**N-Right** (44%), >-Top (26%), >-Right (17%)
Align Left	**A-Left** (100%)	Save As	**S-Right** (58%), S-Pull (20%), S-Towards (12%)	Task Switch	**Tab-Left/Right** (22%), Tab-CW/CCW (20%), W-Top (16%)
Align Middle	**A-FB** (83%), A-Top (13%)	Increase Vol	**V-CW** (52%), +-Top (17%), +-CW (14%)	Accept	**Y-Top** (43%), enter-Top (39%), A-Top (14%)
Align Right	**A-Right** (100%)	Decrease Vol	**V-CCW** (47%), V-Towards (19%), --Top (15%)	Reject	**R-Top** (45%), N-Top (34%)
Align Justify	**A-Pull** (40%), J-Right (14%), J-Pull (13%), J-Top (11%)	Play	spacebar-Top (21%), **P-Top** (18%), Enter-Top (11%)	Help	**H-Top** (49%), H-Pull (23%), ?-Top (14%)
Enlarge	**+-Pull (31%)**, S-CW **(13%)**	Pause	**spacebar-Top** (64%), P-Towards (15%), P-Top (12%)	Menu Access	**menu-Top** (95%)
Shrink	**S-CCW** (14%), --Left (11%)	Cut	**X-Top** (73%), C-Away (24%)	Open	**O-Top** (88%)
Rotate	**R-CW/CCW** (98%)	Copy	**C-Top** (91%)	Close	**esc-Top (25%)**, X-Top **(17%)**, O-Pull **(11%)**
Move a Little	**N-directional** (26%)	Paste	**V-Top** (93%)	Zoom In	**Z-CW** (69%), Z-Away (14%)
Move a Lot	**M-directional** (46%)	Insert	**I-Top** (81%), I-Towards (16%)	Zoom Out	**Z-CCW** (74%), Z-Towards (10%)
Find	**F-Top** (100%)	Duplicate	**D-Top** (40%), D-Pull (18%)	Pan	**L-Left** (< 10%)
Find Prev	**F-Left** (91%)	Undo	**U-Top** (41%), Z-Top (26%), backspace-Left (14%)	Maximize	**M-Pull** (48%), +-CW (32%)
Find Next	**F-Right** (92%)	Delete	**backspace-Top** (84%), D-Top (15%)	Minimize	**--CCW (37%)**, M-Towards **(35%)**, M-Top **(10%)**

Note: Within parentheses, we show the frequency of each sign in the bootstrap distribution. We only include signs for which frequency is equal or greater than 10% and highlight (in **bold**) the best guess based on the available sample. In an informal way, a percentage expresses our "confidence" that the associated sign would be the optimal one if optimization was performed over the full population of users. Here, we use the Hungarian algorithm for optimization. Compare this table with Table 2 of the original publication [3].

shown in Table 3 represents the probability that this sign would be the optimal one if optimization was performed on a random sample of size n drawn from the full user population. By extension, it can be informally used to express our confidence that a sign is optimal.

Is the approximation of the bootstrap method accurate? For a large enough sample n and a large enough number of bootstrapping iterations, we expect that the bootstrap distribution will become close to the unknown sampling distribution. However, proving convergence can be difficult. In Section 6, we show that we can achieve good results with as low as 20 to 30 participants and 200 bootstrap samples (for optimization with the Hungarian algorithm).

6 EXPERIMENTAL RESULTS

We summarize results of a series of experiments that investigate different aspects of our analysis. A detailed report of our methodology and R code can be found in our supplementary material.

Heuristic vs. Optimal Sign Mappings. We assess the magnitude of the error of the heuristic of Wobbrock et al. [45] through a large number of simulated studies. We vary the number of participants $n = 10, 20$, and 30, as well as the number of referents $m = 10, 20$, and 30. For each combination of n and m, we consider 11 distributions (see Figure 5) corresponding to different levels of intrinsic agreement (from 5% to 30%), and for each, we take 100 random samples by varying the bias distribution. With this method, we generate $3 \times 3 \times 11 \times 100 = 9900$ unique populations. From each population, we draw a sample (a unique simulated study) and derive its sign matrix. For each study, we derive the sign mappings by using both the optimal (Hungarian algorithm) and the heuristic approach and calculate their guessability scores $\widehat{G}(\widehat{\pi}_{opt})$ and $\widehat{G}(\widehat{\pi}_h)$. Then, *relative error* $= 1 - \widehat{G}(\widehat{\pi}_h)/\widehat{G}(\widehat{\pi}_{opt})$.

Figure 8a presents the cumulative probability distribution of the relative error, where a cumulative probability $P_c(x)$ represents the proportion of simulated studies for which the error is equal to or lower than x. For the majority ($> 50\%$) of the studies, the heuristic produces optimal mappings (*relative error* $= 0$). Its results improve as the sample size increases: 95% of the studies have an error of less than 5.4% for $n = 10$, less than 4.6% for $n = 20$, and less than 4.0% for $n = 30$. Overall, situations like the one in Figure 2 where the heuristic leads to poor guessability scores are statistically rare.

Guessability Error of Samples. Following the same simulation approach as above, we assess the evolution of the relative guessability error: *relative error* $= 1 - G(\widehat{\pi}_{opt})/G(\pi_{opt})$. This is the true relative guessability error of a sample. For real studies, it is unknown. Our goal is to empirically assess its distribution to help researchers estimate the number of participants required for a study.

We summarize our results in Figure 8b. The mean relative error decreases rapidly until $n = 20$ but is still around 9% for the Hungarian algorithm and continues to drop. The trend is similar for the mode mapper. The error is slightly lower though. Figure 8b also presents the cumulative probability distribution of the relative error for $n = 10, 20$, and 30 (Hungarian algorithm only). We observe that the probability of deriving a suboptimal sign mapping (i.e., with high error compared to an optimal solution for the full user population) can be large for studies with less than 20 participants. To put these results into perspective, we refer to Villarreal-Narvaez et al. [41] who report that for 49% of the 212 studies they reviewed, the number of participants was lower than 20. But even when $n = 20$, for 36% of the studies, the relative error is expected to be higher than 10%.

Time Costs of Optimization. We also assess the computational cost of optimization with custom constraints (see Section 4.3) as the number of constraints increases. We concentrate on studies with $m = 30$ referents and $n = 30$ participants. We fix the ratio of legitimate sign pairs to 5% of the total sign pairs and perform constraint aggregation to minimize the number of constraints. We use a 13-inch MacBook Pro with a 2.3 GHz Quad-Core Intel Core i7 processor and 16 GB RAM. The code is written in R based on Schumacher's library [35].

Figure 8c plots the time needed to optimize the sign mappings of 55 simulated studies in logarithmic scale. Time increases exponentially with the number of observed signs. For studies with 30 signs, optimization lasts around 8 seconds. Yet, as the number of signs

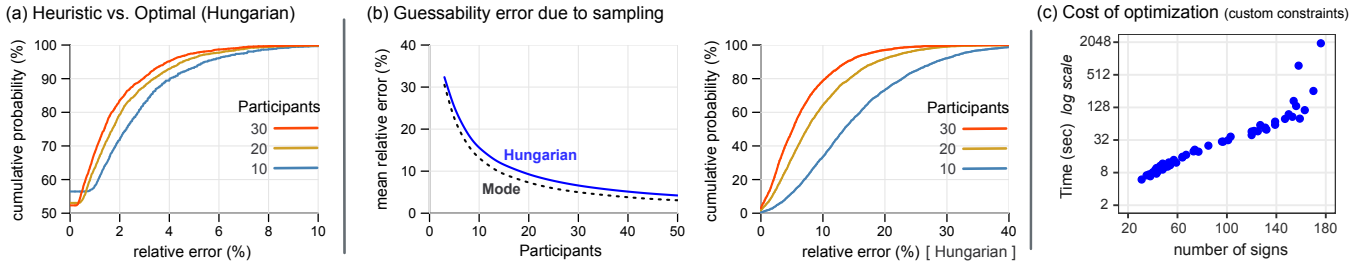


Figure 8: (a) Cumulative probability distributions of the relative guessability error (%) of the heuristic of Wobbrock et al. [45] in comparison with the optimal solution (Hungarian algorithm). (b) Relative error of $G(\widehat{\pi}_{opt})$ over $G(\pi_{opt})$. Left: Evolution of the error for the Hungarian algorithm and the mode mapper. Right: Its cumulative probability distribution for the Hungarian algorithm. (c) Cost of optimization with custom constraints. We show results for 55 simulated studies ($n = 30$ and $m = 30$).

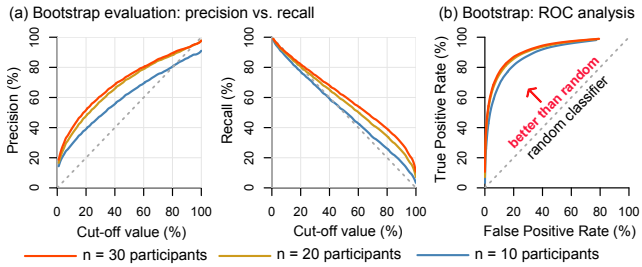


Figure 9: (a) Precision and recall of the bootstrap's binary classification of signs (optimal vs. non-optimal) as a function of the cut-off confidence value used to assign a sign to a referent (200 bootstrap samples). (b) ROC curves [14].

goes beyond 150, optimization can take several minutes (up to 33 minutes in our experiment). Although such costs are not prohibitive, investigators must carefully consider them, in particular if they combine optimization with our cross-validation and bootstrapping methods, since those require a large number of iterations.

The Bootstrap Method. Finally, we evaluate the accuracy of the bootstrap method (see Section 5.2) with the Hungarian algorithm. Specifically, we assess how well a confidence value (e.g., see Table 3) predicts whether a sign is optimal for a given referent. To this end, we set a cut-off value, i.e., a threshold, and keep only the signs with a confidence that is equal to or greater than this value. Then, we assess the proportion of these signs that are in our ground truth π_{opt}. This proportion expresses the *precision* of the method. We also assess the *recall* of the method, that is, the proportion of signs in the ground truth π_{opt} whose confidence is equal to or greater than the cut-off value. Finally, we conduct a ROC (*receiver operating characteristics*) analysis [14] to assess the quality of the binary classification of signs (optimal vs. non-optimal) as we vary the cut-off confidence value. We use 200 bootstrap samples to generate the bootstrap sampling distribution, as we did not observe major improvements with larger samples.

As shown in Figure 9a, precision is globally higher than the reference level of cut-off values (grey line) but deteriorates when $n = 10$. For the larger sample sizes ($n = 20$ and $n = 30$), caution is needed for confidence levels greater 95%, e.g., a confidence value of 100% reflects a true success rate of 97.7%. There is clearly a tradeoff between precision and recall. However, our results on recall must be interpreted with care, as they show average performance. Recall is analogous to the notion of *power* in traditional experiments. It does not only depend on the sample size. It also depends on the magnitude of the effect: the higher the guessability of optimal signs, the higher their confidence levels become, and the further away (upwards) recall moves from the diagonal grey line.

Figure 9b presents the ROC curves that illustrate the trade-off between the *true positive rate* and the *false positive rate*. In summary, generating confidence values with the bootstrap method results in a good balance between true and false positives, which improves as the size of the sample increases.

7 DISCUSSION

We end our analysis by discussing some additional issues.

Overfitting Issues when Classifying Gestures. We made the assumption (see Section 4) that the gesture classification process is fixed prior to the analysis of the gesture dataset. This assumption, however, does not hold for some real studies. Chan et al. [7] (single-hand microgestures), for instance, refined their classification by inspecting individual gesture instances of the actual study. Would the authors have opted for the same classification strategy, had they looked at a subset of the dataset or the gestures from a different sample of participants? This source of overfitting cannot be captured by our analysis because we do not know how the authors would decide to classify gestures for each different subset of participants. For Figure 4, we use the final sign set to calculate guessability for the full evolution of the training and the cross-validation curves, but this approach undervalues the variance error.

In order to best benefit from the tools we described, we encourage HCI researchers to specify as many of the details of their gesture classification process and their design constraints before running their elicitation study or, at least, before analyzing data. Otherwise, cross-validation may underestimate overfitting problems. However, we acknowledge that this may be often hard to achieve in practice. Subtle design constraints may be hard to formally describe or anticipate in advance. As a compromise, our approach allows for testing multiple classification strategies or design constraints at the same time and comparing their bias and variance trade-offs. The investigators may also decide to interrupt a study (or analysis) early enough, e.g., if bias error is high, update their gesture classification or sign grouping strategy and then restart the process with a new sample of participants.

Table 4: Correlation (Pearson's r) between optimal guessability scores and three agreement indices: agreement rate (AR) [40], Fleiss' κ_F [15], and its chance agreement term p_e

m = 10 referents			m = 20 referents			m = 30 referents		
AR	κ_F	p_e	AR	κ_F	p_e	AR	κ_F	p_e
.68	.92	−.14	.38	.75	−.46	.22	.64	−.59

Note: Each result is based on 1100 simulated studies with $n = 30$. The Hungarian algorithm was used for optimization.

Guessability vs. Agreement. A review of a large number of publications by Villarreal-Narvaez et al. [41] shows that agreement measures have largely dominated the analysis of gesture elicitation data. Why do we focus instead on guessability metrics? Agreement, as formalized by the gesture elicitation literature [38, 40, 45], is a global analysis measure that does not characterize a specific sign mapping set. As such, it cannot be used as an objective function. We align with Tsandilas' [38] position that agreement indices must be treated as reliability (rather than optimization) measures. A low agreement score indicates that sign mappings cannot be trusted, either because participants do not agree on which sign to assign to each referent, or simply because their sign assignments are random.

Nonetheless, one would expect that high agreement also results in high optimal guessability scores. Table 4 shows Pearson's correlation r between the optimal guessability G (for samples of size $n = 30$) and three agreement indices: the agreement rate AR of Vatavu and Wobbrock [40], Fleiss' κ_F [15] and its chance agreement term p_e. For a discussion of the last two indices in the context of gesture elicitation, please refer to our previous work [38, 39]. A first observation is that the correlation of the optimal guessability with Fleiss' κ_F is clearly stronger than its correlation with AR. This result supports our argumentation [38, 39] that chance-corrected agreement indices better serve the design goals of gesture elicitation, since they penalize biases that increase potential conflicts among referents. Another observation is that chance agreement negatively correlates with guessability. This correlation becomes stronger as the number of referents increases, since there is more competition for popular signs, and conflicts become more frequent.

Design Complexity. We showed how to optimize guessability for studies with custom design constraints (see Subsection 4.3), but we did not address the more challenging problem of how to derive meaningful design constraints. Gesture taxonomies [46] (or ontologies [29]) can support decisions on how to group similar gestures together. Unfortunately, they are not always available, or they are often too broad to deal with the specificities of real design problems. Solutions will eventually become even more complex if we consider usability metrics beyond guessability, such as performance and mnemonics [37], or fatigue [18].

Interaction design is a creative process, and a designer's experience and intuition can be difficult to describe with formal objective functions and constraints. In our view, computational methods are excellent tools for studying fundamental research questions, such as: What is the risk of overfitting? What is an appropriate sample size? However, their strict formalism and added complexity may be hard to justify in everyday design practice.

8 CONCLUSION

While there is a rich methodology literature to help researchers understand and characterize data arising from gesture elicitation studies, the important task of deriving gesture vocabularies from this data remains largely based on heuristics and ad hoc methods. We contribute to the existing literature by examining how to derive optimal gesture vocabularies.

We presented a computational approach for optimizing gesture vocabularies based on data from gesture elicitation studies. We expressed this optimization task as a combinatorial assignment problem that uses guessability as its objective function. We first showed how to optimally solve the most common formulation of the problem (one sign per referent) with the Hungarian algorithm [21]. We explained that the heuristic of Wobbrock et al. [45] for mapping signs to referents is not optimal, but we found that in practice, it approximates the optimal solution very well. We then extended the problem to account for custom design constraints, in particular when groups of signs are mapped to the same referent. For this instance of the problem, we provided a solution based on integer programming [27, 36], and suggested an algorithm to reduce its computational cost.

In contrast with many optimization problems, user interface optimization based on gesture elicitation uses data from random samples, adding uncertainty to the optimization outcome. To deal with this source of uncertainty, we formulated gesture elicitation as a machine learning problem, where elicited gestures are treated as training data. Based on cross-validation and learning-curve methods [8, 23], we explained how to examine the evolution of the guessability error as the sample size increases, and presented computational tools to help HCI researchers decide when they can stop collecting data. We provided experimental data about the distribution of the guessability error. Finally, we presented a bootstrap method [12] that enables researchers to test the robustness of the sign mappings produced by the optimization process and communicate the uncertainty of their results. We provide extensive supplementary material that can be used by researchers to replicate, verify, and extend our results.

One avenue for future work is to use the proposed tools to develop user interfaces that help HCI researchers and practitioners interactively customize the optimization parameters of their gesture design problem and examine the results. This is consistent with the philosophy of computational interaction whose goal is not to automate HCI design but to *"complement and boost the very human and essential activity of interaction design"* [4]. Future work may also examine how the methods we put forward could be applied to other design problems such as keyboard or GUI layout optimization [2, 5, 28], when the optimal solution is also often based on empirical data from a small numbers of users.

REFERENCES

[1] [n.d.]. Google OR Tools: Integer Optimization. https://developers.google.com/optimization/mip/mip. Accessed: 2020-2-23.

[2] Gilles Bailly and Antti Oulasvirta. 2014. Toward optimal menu design. *Interactions* 21, 4 (2014), 40–45.

[3] Gilles Bailly, Thomas Pietrzak, Jonathan Deber, and Daniel J. Wigdor. 2013. Métamorphe: Augmenting Hotkey Usage with Actuated Keys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. ACM, New York, NY, USA, 563–572. https://doi.org/10.1145/2470654.2470734

[4] Xiaojun Bi, Andrew Howes, Per Ola Kristensson, Antti Oulasvirta, and John Williamson. 2018. Introduction. In *Computational Interaction*, Antti Oulasvirta, Per Ola Kristensson, Xiaojun Bi, and Andrew Howes (Eds.). Oxford University Press, 1–16.

[5] Xiaojun Bi, Brian Smith, Tom Ouyang, and Shumin Zhai. 2018. Soft Keyboard Performance Optimization. In *Computational Interaction*, Antti Oulasvirta, Per Ola Kristensson, Xiaojun Bi, and Andrew Howes (Eds.). Oxford University Press, Chapter 5, 121–152.

[6] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. 2009. *Assignment Problems*. Society for Industrial and Applied Mathematics, USA.

[7] Edwin Chan, Teddy Seyed, Wolfgang Stuerzlinger, Xing-Dong Yang, and Frank Maurer. 2016. User Elicitation on Single-hand Microgestures. In *Conference on Human Factors in Computing Systems (CHI)*. ACM, San Jose, United States. https://doi.org/10.1145/2858036.2858589

[8] Mark de Rooij and Wouter Weeda. 2020. Cross-Validation: A Method Every Psychologist Should Know. *Advances in Methods and Practices in Psychological Science* 3, 2 (2020), 248–263. https://doi.org/10.1177/2515245919898466

[9] Persi Diaconis and Bradley Efron. 1983. Computer-intensive methods in statistics. *Scientific American* (June 1983). https://doi.org/10.1038/scientificamerican0583-116

[10] 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, 419:1–419:12. https://doi.org/10.1145/3173574.3173993

[11] Pierre Dragicevic, Yvonne Jansen, Abhraneel Sarma, Matthew Kay, and Fanny Chevalier. 2019. *Increasing the Transparency of Research Papers with Explorable Multiverse Analyses*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3290605.3300295

[12] Bradley Efron. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Mathematical Statistics* 7, 1 (01 1979), 1–26. https://doi.org/10.1214/aos/1176344552

[13] Alexander Eiselmayer, Chat Wacharamanotham, Michel Beaudouin-Lafon, and Wendy E. Mackay. 2019. Touchstone2: An Interactive Environment for Exploring Trade-Offs in HCI Experiment Design. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3290605.3300447

[14] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010 ROC Analysis in Pattern Recognition.

[15] Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76, 5 (1971), 378–382.

[16] Krzysztof Gajos and Daniel S Weld. 2005. Preference elicitation for interface optimization. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. 173–182.

[17] Fred Glover. 2003. Tutorial on Surrogate Constraint Approaches for Optimization in Graphs. *Journal of Heuristics* 9, 3 (2003), 175–227. https://doi.org/10.1023/A:1023721723676

[18] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. 2014. Consumed Endurance: A Metric to Quantify Arm Fatigue of Mid-Air Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1063–1072. https://doi.org/10.1145/2556288.2557130

[19] 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* (Eindhoven, Netherlands) *(TEI '16)*. Association for Computing Machinery, New York, NY, USA, 86–91. https://doi.org/10.1145/2839462.2839472

[20] Susan Holmes. 2003. Bootstrapping Phylogenetic Trees: Theory and Methods. *Statist. Sci.* 18 (05 2003). https://doi.org/10.1214/ss/1063994979

[21] Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1-2 (1955), 83–97. https://doi.org/10.1002/nav.3800020109

[22] Byron Lahey, Audrey Girouard, Winslow Burleson, and Roel Vertegaal. 2011. PaperPhone: Understanding the Use of Bend Gestures in Mobile Devices with Flexible Electronic Paper Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. ACM, New York, NY, USA, 1303–1312. https://doi.org/10.1145/1978942.1979136

[23] Christopher Meek, Bo Thiesson, and David Heckerman. 2002. The Learning-Curve Sampling Method Applied to Model-Based Clustering. *Journal of Machine Learning Research* 2 (March 2002), 397–418. https://doi.org/10.1162/153244302760200678

[24] Meredith Ringel Morris. 2012. Web on the Wall: Insights from a Multimodal Interaction Elicitation Study. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces* (Cambridge, Massachusetts, USA) *(ITS '12)*. ACM, New York, NY, USA, 95–104. https://doi.org/10.1145/2396636.2396651

[25] Andrew Ng. [n.d.]. Advice for Applying Machine Learning. http://cs229.stanford.edu/materials/ML-advice.pdf. Lecture Notes – Stanford University.

[26] Antti Oulasvirta. 2017. User interface design with combinatorial optimization. *Computer* 50, 1 (2017), 40–47.

[27] Antti Oulasvirta, Niraj Ramesh Dayama, Morteza Shiripour, Maximilian John, and Andreas Karrenbauer. 2020. Combinatorial optimization of graphical user interface designs. *Proc. IEEE* 108, 3 (2020), 434–464.

[28] Antti Oulasvirta and Andreas Karrenbauer. 2018. Combinatorial Optimization for User Interface Design. In *Computational Interaction*, Antti Oulasvirta, Per Ola Kristensson, Xiaojun Bi, and Andrew Howes (Eds.). Oxford University Press, Chapter 4, 97–120.

[29] Mehdi Ousmer, Jean Vanderdonckt, and Sabin Buraga. 2019. An Ontology for Reasoning on Body-Based Gestures. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Valencia, Spain) *(EICS '19)*. Association for Computing Machinery, New York, NY, USA, Article 17, 6 pages. https://doi.org/10.1145/3319499.3328238

[30] Maurice H. Quenouille. 1949. Problems in Plane Sampling. *The Annals of Mathematical Statistics* 20, 3 (09 1949), 355–375. https://doi.org/10.1214/aoms/1177729989

[31] David F. Rogers, Robert D. Plante, Richard T. Wong, and James R. Evans. 1991. Aggregation and Disaggregation Techniques and Methodology in Optimization. *Operations Ressearch* 39, 4 (Aug. 1991), 553–582. https://doi.org/10.1287/opre.39.4.553

[32] 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* (Seoul, Republic of Korea) *(CHI '15)*. ACM, New York, NY, USA, 3347–3350. https://doi.org/10.1145/2702123.2702583

[33] 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, 142:1–142:12. https://doi.org/10.1145/3173574.3173716

[34] Nikolaos V. Sahinidis. 2004. Optimization under uncertainty: State-of-the-art and opportunities. *Computers and Chemical Engineering* 28 (2004), 971–983.

[35] Dirk Schumacher. [n.d.]. R for Operations Research. https://www.r-orms.org/mixed-integer-linear-programming. Accessed: 2020-2-23.

[36] Gerard Sierksma and Yori Zwols. 2015. *Linear and integer optimization : theory and practice* (3rd ed. ed.). Chapman & Hall/CRC.

[37] Srinath Sridhar, Anna Maria Feit, Christian Theobalt, and Antti Oulasvirta. 2015. Investigating the dexterity of multi-finger input for mid-air text entry. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 3643–3652.

[38] Theophanis Tsandilas. 2018. Fallacies of Agreement: A Critical Review of Consensus Assessment Methods for Gesture Elicitation. *ACM Transactions on Computer-Human Interaction* 25, 3 (June 2018), 49. https://doi.org/10.1145/3182168

[39] Theophanis Tsandilas and Pierre Dragicevic. 2016. *Accounting for Chance Agreement in Gesture Elicitation Studies*. Research Report 1584. LRI - CNRS, University Paris-Sud. 5 pages. https://hal.archives-ouvertes.fr/hal-01267288

[40] 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* (Seoul, Republic of Korea) *(CHI '15)*. ACM, New York, NY, USA, 1325–1334. https://doi.org/10.1145/2702123.2702223

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

[42] Abraham Wald. 1945. Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics* 16, 2 (1945), 117 – 186. https://doi.org/10.1214/aoms/1177731118

[43] Larry Wasserman. [n.d.]. The Bootstrap. http://www.stat.cmu.edu/~larry/=sml/Boot.pdf. Lecture Notes – Carnegie Mellon University.

[44] Martin Weigel, Vikram Mehta, and Jürgen Steimle. 2014. More Than Touch: Understanding How People Use Skin As an Input Surface for Mobile Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. ACM, New York, NY, USA, 179–188. https://doi.org/10.1145/2556288.2557239

[45] 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* (Portland, OR, USA) *(CHI EA '05)*. ACM, New York, NY, USA, 1869–1872. https://doi.org/10.1145/1056808.1057043

[46] 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* (Boston, MA, USA) *(CHI '09)*. ACM, New York, NY, USA, 1083–1092. https://doi.org/10.1145/1518701.1518866