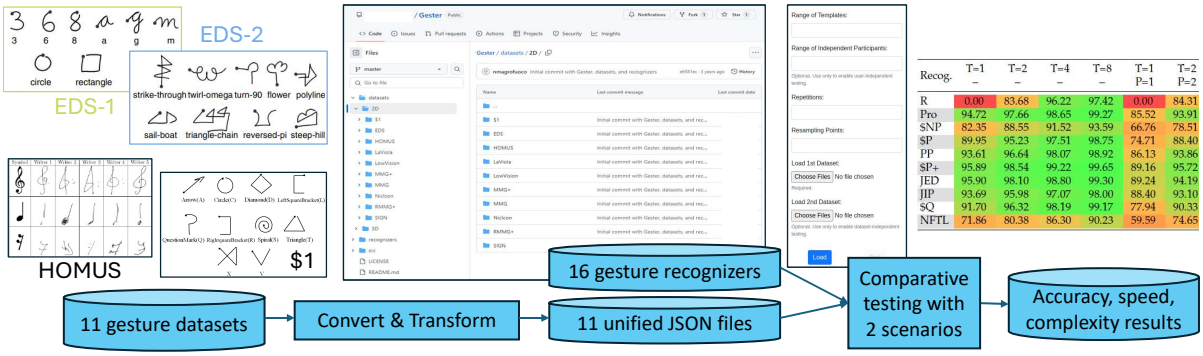
Comparative Testing of 2D Stroke Gesture Recognizers in Multiple Contexts of Use

NATHAN MAGROFUOCO and ARTHUR SLUŸTERS, Université catholique de Louvain, Belgium
PAOLO ROSELLI, Università degli Studi di Roma "Tor Vergata", Italy
JEAN VANDERDONCKT, Université catholique de Louvain, Belgium



Recog.	T=1 –	T=2 –	T=4 –	T=8 –	T=1 P=1	T=2 P=2
R	0.00	83.68	96.22	97.42	0.00	84.31
Pro	94.72	97.66	98.65	99.27	85.52	93.91
$NP	82.35	88.55	91.52	93.59	66.76	78.51
$P	89.95	95.23	97.51	98.75	74.71	88.40
PP	93.61	96.64	98.07	98.92	86.13	93.86
$P+	95.89	98.54	99.22	99.65	89.16	95.72
JED	95.90	98.10	98.80	99.30	89.24	94.19
JIP	93.69	95.98	97.07	98.00	88.40	93.10
$Q	91.70	96.32	98.19	99.17	77.94	90.33
NFTL	71.86	80.38	86.30	90.23	59.59	74.65

Fig. 1. Procedure of our comparative testing: 11 gesture datasets have been converted and transformed into a unified JSON format, and compared in 2 scenarios by 16 recognizers against their accuracy, speed, and algorithmic complexity.

The wide range of algorithms for recognizing two-dimensional stroke gestures raises a crucial question: which recognizer should be selected depending on the gesture set and its context of use? Since the context of use expresses a category of users performing interactive tasks on a device in a physical environment, many contextual conditions are feasible. Recognizers have been tested in only a few of them, making them difficult to compare and select. A recognizer suitable for one context of use does not necessarily translate to another. Recognizer variables, such as the number of resampling points, the number of templates, and the number of users, vary from one recognizer to another, making them challenging to specify for practitioners and heterogeneous to compare for researchers. To address these issues, this article performs comparative tests of 16 recognizers (15 state-of-the-art recognizers and one for rotation invariance) in 11 gesture sets (9 reference sets and 2 for multi-device rotation invariance) against their accuracy, speed, and algorithmic complexity, in user-(in)dependent scenarios based on the variables mentioned above. In particular, the impact of two contextual dimensions, *i.e.,* the visual impairments of end users and input devices, on accuracy is further studied. The comparative testing results in an empirically validated decision table for the researcher to select a recognizer depending on contextual conditions and a flow chart for the practitioner to be guided.

Authors' Contact Information: Nathan Magrofuoco, nathan.magrofuoco@uclouvain.be; Arthur Sluÿters, arthur.sluyters@uclouvain.be, Université catholique de Louvain, Louvain Research Institute in Management and Organizations (LouRIM), Louvain-la-Neuve, Belgium; Paolo Roselli, roselli@mat.uniroma2.it, Università degli Studi di Roma "Tor Vergata", Roma, Italy; Jean Vanderdonckt, jean.vanderdonckt@uclouvain.be, Université catholique de Louvain, Louvain Research Institute in Management and Organizations (LouRIM), Louvain-la-Neuve, Belgium.

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 the author(s) 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.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 2573-0142/2025/6-ARTEICS005
https://doi.org/10.1145/3733051

We break down our design implications into two categories: gesture acquisition, recognition, and gesture set design. We also publicly release Gester, a JavaScript software for conducting comparative testing of recognizers under various contextual conditions, allowing the practitioner to test a recognizer in a project and the researcher to benchmark recognizers.

CCS Concepts: • **Human-centered computing** → **Graphical user interfaces**; *Web-based interaction*; Touch screens; **Gestural input**; User interface management systems; User interface programming; • **Software and its engineering** → *Software prototyping*; • **Theory of computation** → *Pattern matching*; • **Computing methodologies** → Cross-validation.

Additional Key Words and Phrases: Benchmarking, Comparative testing, Gesture datasets, Gesture recognition, Gesture user interfaces, Invariance properties, Stroke gestures.

ACM Reference Format:
Nathan Magrofuoco, Arthur Sluÿters, Paolo Roselli, and Jean Vanderdonckt. 2025. Comparative Testing of 2D Stroke Gesture Recognizers in Multiple Contexts of Use. *Proc. ACM Hum.-Comput. Interact.* 9, 4, Article EICS005 (June 2025), 35 pages. https://doi.org/10.1145/3733051

1 Introduction

Gesture-based interactive applications [84] and computing [21] today provide new opportunities for natural and intuitive interaction as they interpret human movements into computer commands using gesture recognition software [42]. For example, a person searches for a store in a shopping mall by navigating an interactive map through gestures, a customer orders beverages in a restaurant by gestural selection. The quality of this gestural interaction depends a lot on the set of gestures, its mapping to commands, and the recognizer used for gesture recognition [100]. The wide offer of algorithms [5, 6, 10, 23, 33, 34, 42, 46, 62, 65, 75, 78–80, 82, 83, 85, 86, 93, 101] and their recent variants (*e.g.,* $B for biosignal gestures [97]) for recognizing two-dimensional (2D) stroke gestures [100] available to practitioners and researchers (see [49] for a general survey and [69] for a survey specific to children) raises a crucial research question: *RQ = Which recognizer to select and how to specify the appropriate values for its parameters depending on the gesture set and its intended context(s) of use?*

Today, answering these questions remains an open and complex problem for several reasons for both *practitioners* (*e.g.,* designers and developers, representative end users, and other stakeholders) who would like to contribute to designing a gestural interface and *researchers* (*e.g.,* experimenters, engineers, and scientists) who are interested in conducting research and development in this area. Each time, we discuss the introductory point of view of the practitioner and the expert point of view of the researcher, since their concerns vary even if the subject remains the same.

Variability of contexts of use. The researchers observe that the recognizers were developed and tested mainly in designated *contexts of use* [26], for which contextual dimensions are sometimes partial, missing, or too abstract to be reused for another context. These key dimensions [22] cover the different end users (*e.g.,* able-bodied people, people with disabilities) who perform their tasks (*e.g.,* navigating an interactive map, changing levels, and zooming in/out), the devices used (*e.g.,* a touchable public display or tabletop), and the physical environments in which this interaction takes place [55]. Therefore, a recognizer suitable for one context of use does not necessarily translate to another context. Practitioners are interested in gesture-based user interfaces (UIs) tailored toward some well-defined contexts of use. Consequently, selecting an appropriate recognizer and configuring it with the adequate values of its variables becomes critical, as not every recognizer fits any context of use. In sum, practitioners would like to identify a recognizer suitable to their project whereas researchers would like to cover as many contexts of use as possible.

Variability of recognizers support. The recognizers support different types of gestures (*e.g.,* uni-[93] *vs.* multistroke [5], different devices (*e.g.,* workstations [65], desktops [83], tabletops [92], smartphones and tablets [86], and smartwatches [87]), different users, (*e.g.,* able-bodied users *vs.* low-vision users [87]), and different environments (*e.g.,* mobile [87] *vs.* stationary). Therefore, determining the conditions for selecting a

suitable recognizer is critical to everyone. For example, a certain recognizer could work for a wide spectrum of devices (*e.g.,* Penny Pincher [75] is fast for many devices) while a particular one could be optimized for this target (*e.g.,* $Q [86] optimizes the execution time for low-end devices and JackNife [76] works for many modalities). These conditions do not necessarily match those required in the context of use intended by the stakeholder. In sum, practitioners would like to select a recognizer that is effectively and efficiently supported for their project, whereas researchers would like to locate areas of improvement to ensure or improve software support in less covered contexts of use.

Questionable transition from rapid prototyping to real deployment. Although several recognizers were initially introduced for rapid prototyping of gesture-based UIs, they are also deployed in real applications and projects due to their performance. For example, the $-family [5, 83, 85, 86, 93] is motivated by the fast prototyping of gesture-based UIs. These algorithms differ from those exploiting Machine Learning (ML) and Deep Learning (DL) by their simplicity of understanding, training, and usage. Several recognizers support a geometric interpretation of their results (*e.g.,* by comparing the similarity between two gestures based on their Euclidean or angular distances), require a limited number of templates (*e.g.,* only 4-7 templates), achieve high performance, and hold in a reasonable number of lines of code (*e.g.,* 100-200 LOC). Contrarily, ML recognizers are sometimes difficult to scrutinize (their statistical approach challenges any explanation), require many templates to reach high recognition rates, consume a long execution time, and involve complex packages. For these reasons, the aforementioned recognizers remain acceptable substitutes for ML/DL [69] for prototyping and real deployment of applications and projects. For example, $1 [5] recognizes gestures in flower menus [9] and $N [93] is used in Microsoft 'PocketTouch' and its demonstration. In sum, substituting a recognizer for real deployment (such as ML/DL recognizers [43]) with a recognizer for prototyping (such as template-based recognizers [29, 66]) is welcome for moderately experienced practitioners who wish to quickly prototype gesture-based UIs. Experienced practitioners and researchers prefer investigating more advanced recognizers proven for real-world applications.

Need to explicitly consider recognition rate and execution time. These recognizers, when they are kept in real-world gesture-based UIs, significantly impact the User Experience (UX) of their applications in terms of recognition rate (*accuracy*) and execution time (*speed*) [28]. A recognition rate $\tau \geq 90\%$ is the minimum rate condition for a recognition system to operate and to be accepted by end-users [52, 67, 68, 90]. In particular, Read et al. [61] report a recognition rate of $\tau = 91\%$ as an acceptable threshold for children producing handwriting, while Wolf [94] and LaLomia [41] found a recognition rate of $\tau = 93.2\%$, resp. $\tau = 97\%$ to be acceptable, but modulated by the contextual conditions. Chen [19] even locates the best compromise around 500 ms with $\tau \geq 90\%$. Since end-users perceive latency at the 50-ms level for inking tasks on a touch-sensitive surface [3], recognizers should also satisfy a recognition time constraint of $t \leq 50$ ms. In sum, practitioners would like to ensure that the selected recognizer achieves a real UX while researchers would like to test the real recognition rate and execution times in various conditions.

Need for benchmarking gesture recognition. Since the algorithms were tested under heterogeneous experimental conditions, they cannot be compared with each other on the same basis. This makes it impossible to establish any benchmarking process where any existing or new recognizer could be compared to others to establish a reference point. Moreover, a significant number of different gesture classes and sets have been used for comparative tests, but not under the same conditions [69]. Different recognizers can be fairly compared only if tested under the same experimental conditions, such as on one common gesture set according to an identical test protocol. These conditions include the properties of the gesture set (*e.g.,* its difficulty [88], its similarity with others [36], its overall performance [45], its consistency across modalities [4]), the desired recognition properties (*e.g.,* invariance properties [33, 86]), but also the test parameters and scenarios (*e.g.,* which sampling rate [81], how many templates). Recognition accuracy becomes a control condition for such a comparison, but not speed, which depends on other contextual variables like the device. No single recognizer fits all contexts of use and gesture sets. However, it may perform well in some unexpected gestures and contexts.

Therefore, testing any recognizer for several gesture sets and contexts of use to identify its strengths and weaknesses, either for a particular gesture set or for its overall behavior. Due to the lack of a standardized test procedure, published recognizers are tested within their experimental conditions and do not always provide comprehensive comparative testing. In sum, practitioners would like to ensure that their selected recognizer is the best one for the gesture set. Researchers would like to perform comparative testing (also called *competitive testing*) to evaluate the strengths and weaknesses of two or more recognizers based on users' actual experiences and to generate ideas for new recognizers or benchmarking to establish a baseline for recognizers, setting the bar against which to compare and measure the improvements in future releases of recognizers.

Lack of methodological guidance. With the increasing number of recognizers and the above variations, practitioners are left out without any methodological guidance on how to select an appropriate recognizer for a given gesture set and a given context of use and how to specify the values of their parameters to make them work well. Such guidance should not necessarily be based on theoretical considerations but also rely on well-defined programming criteria [32] and empirical results [72]. Researchers are also interested in studying the overall behavior of recognizers to determine the variables making the recognition more accurate and fast, while mitigating complexity.

For this purpose, our comparative testing aims at evaluating the benefits and shortcomings of various recognizers based on a predefined set of criteria to gauge how well a particular recognizer measures up to the others, to select any recognizer depending on contextual variables, to specify adequate values for its variables, and to focus on state-of-the-art recognizers to generate ideas for new avenues. Exploring to what extent recognizer variables may influence the recognition rate and execution time is important for practitioners to fine-tune gesture-based UIs for a context of use and for researchers to get new insights on recognizers' behavior.

The remainder of this paper is structured as follows. After recalling a definition of 2D stroke uni/multi-stroke gestures, the focus of this paper, section 2 and section 3 review state-of-the-art recognizers and existing gesture datasets, respectively. Then, section 4 justifies the consideration of 16 recognizers (15 state-of-the-art recognizers and one for rotation invariance) on 11 gesture sets (9 reference sets and 2 for multidevice rotation invariance), and describes the design of our experiment, whose procedure is summarized in Figure 1. We report and analyze the results of this experiment in section 5 in terms of the recognition rate to assess accuracy, the execution time to assess speed, and the algorithmic complexity to assess their sensitiveness when scaling up. The impact of input devices and visual impairments, two particular contextual variables, is further studied. Based on these results, section 6 suggests some design implications for choosing a recognizer depending on contextual variables. Finally, section 7 concludes the paper by underlining the benefits of the experiment beyond the comparative testing and suggests some future avenues for this research.

2 Related Work

Two-dimensional (2D) stroke gestures are those gestures performed by an end-user on a pen-[12], mouse- [101], or finger-sensitive planar surface [64] to trigger functions, commands [100], their shortcuts [7], and to manipulate information. Gesture recognition consists of classifying a *candidate* gesture based on a training set, *i.e.,* a series of samples called *templates*. The recognizers determine the template most similar to a metric and return its *gesture class*. Researchers distinguish *uni-* from *multistroke*, *i.e.,* when gestures are composed of multiple strokes, and *uni-* from *multi-touch*, *i.e.,* when gestures consist of multiple strokes performed at the same time (*e.g.,* by combining index and thumb).

In the following, we focus on uni-touch, uni/multistroke gestures. Multi-touch gestures are supported by another growing set of recognizers, some of them being extensions of the algorithms discussed hereafter, such as Match-up and Conquer [62] and CircGR [10]. We refer to the literature on multi-touch gestures, surveyed in Cirelli and Nakamura [20] and studied in Bagawade et al. [8] and Rekik et al. [63] in regard to their difficulty. Some

Table 1. Properties of the sixteen 2D stroke gesture recognizers compared. ○=unsupported feature, ●=supported feature, ●=supported feature with limitations.

Recognizer [Ref.]	Multistroke	Articulation invariant	Sampling invariant	Translation invariant	Scaling invariant	Rotation invariant	Combinatorial explosion	Metric
R [65]	○	○	●	●	○	●	○	Linear evaluation of a feature vector
$1 [5]	○	○	●	●	●	●	○	Euclidean distance
Pro [46]	○	○	●	●	●	●	○	Angular distance
$N [5]	●	●	●	●	●	●	●	Euclidean distance
$NR [6]	●	●	●	●	●	●	●	Euclidean distance
$NP [6]	●	●	●	●	●	●	●	Euclidean distance
$NPR [6]	●	●	●	●	●	●	●	Euclidean distance
$P [85]	●	●	●	●	●	○	○	Euclidean distance
$P+ [83]	●	●	●	●	●	○	○	Euclidean distance
$Q [86]	●	●	●	●	●	○	○	Euclidean distance
PP [78]	○	○	●	●	●	○	○	Angular distance
JED [79]	○	○	●	●	●	○	○	Euclidean distance
JIP [79]	○	○	●	●	●	○	○	Angular distance
!FTL [80]	○	○	●	●	●	●	○	Local shape distance
!NFTL [80]	○	○	●	●	●	●	○	Normalized local shape distance
μV [50]	●	●	●	●	●	●	○	Local shape distance

3D stroke gestures can be recognized by recognizers which are themselves a three-dimensional generalization of existing 2D recognizers [58]. For example, $3 [39] is a 3D generalization of $1 [93], μC_1 and μC_2 [13] are a 3D generalization of μV [50]. Other generalizations of 2D stroke gesture recognizers to 3D gestures are considered in Ousmer et al. [57], such as $\$P^3$, $\$P+^3$, $\$Q^3$, and Rubine3D, another extension of Rubine [65] for 3D which projects the 3D unistroke gesture on three orthogonal 2D planes, thus leading to three 2D unistroke gestures.

The two next sub-sections will review a representative subset of existing recognizers for 2D stroke gestures without being extensive and will define the particular problem of recognizing such gestures in the context of our research question to select the best recognizer for this purpose.

2.1 Existing 2D Stroke Gesture Recognizers

Rubine (R) [65] implements a feature-based classifier for recognizing uni-stroke gestures by converting gesture points into a 13-feature vector and computing a vector distance (Figure 2-d) to classify templates independently of their position (*translation invariance*). This method is highly accurate, requires an affordable computational cost, automates the training of new recognizers, and avoids manual opportunistic programming. Its performance initiated Rubine3D, an extension for 3D gestures as part of the iGesture workbench [70].

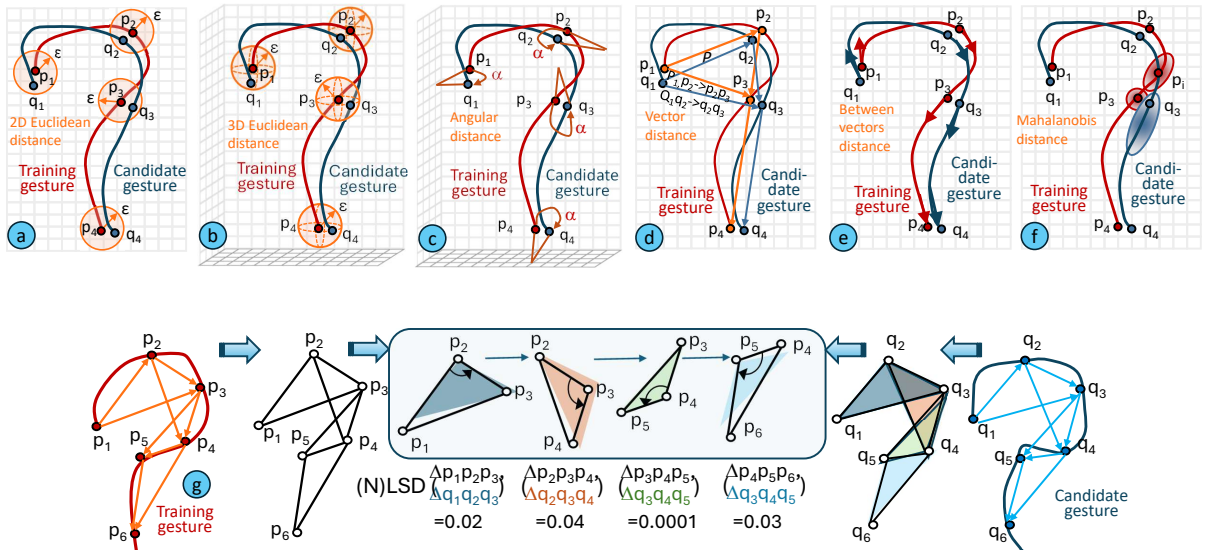


Fig. 2. Metrics for comparing a candidate gesture against a template [80]: 2D Euclidean distance (a), 3D Euclidean distance (b), angular distance (c), vector distance (d), between vectors distance (e), Mahalanobis distance (f), local shape distance (g).

The **$-family** employs template matching to classify a candidate after pre-processing all templates for satisfying *sampling*, *scaling*, and translation invariances. **$1** [5] and **$N** [93] use the golden section search [60] to compare gestures at different angular alignments to ensure *rotation invariance*. $N also recognizes multistroke gestures with *articulation invariance*, *i.e.,* gestures are recognized regardless of variations in the number, order, and direction of their strokes [33]. This technique faces a combinatorial explosion in storage and computational time if the templates are composed of many strokes (*e.g.,* a simple square can be performed in 442 possible ways when considering any order, number, and direction of its strokes [86]). The **$P** [85], **$P+** [83], and **$Q** [86] algorithms mitigate this issue by employing cloud matching to compare any candidate with each template, *i.e.,* the gestures are considered as unordered sets of points. Consequently, the rotation invariance is dropped as the golden section search does not mesh well with this technique. $Q brings several algorithmic improvements to $P to reduce its computational time in low-resource devices. The $-family computes the Euclidean distance (Figure 2-a) between pairs of corresponding points to assess gesture similarity.

Protractor (Pro) [46] and **$N-Protractor** ($NP) [6] extend $1 and $N, respectively, by introducing a closed-form solution to compute the minimal angular distance between two gestures, thus outperforming their predecessors in accuracy, speed, and storage cost. Four variants extended Protractor's closed-form solution and/or bounded rotation invariance: $N, $N-Protractor ($NP), $N-Rotation ($NR), and $N-Protractor-Rotation ($NPR),

Penny Pincher (PP) [78] and **Jackknife** [79] are two uni-stroke recognizers that reduce the pre-processing of the $-family to resampling and vectorizing of the gesture points. They satisfy sampling, translation, and scaling invariances. PP calculates the sum of the angles between pairs of corresponding vectors (Figure 2-c) to assess the similarity between two gestures, whereas Jackknife promotes dynamic time warping using the inner product (JIP) or the Euclidean distance (JED) as the local cost function in addition to correction factors and rejection criteria.

!FTL [80] employs the local shape distance (Figure 2-d) to assess the dissimilarity between two gestures since the local shape distances of two similar gestures converge towards each other [47]. This distance mathematically satisfies the rotation, scaling, and translation invariances after resampling and converting the gesture point into

shapes, *i.e.,* a pair of two vectors connecting three consecutive points. To restore direction invariance that is not satisfied by !FTL, **!NFTL** doubles the calculation, one !FTL for each direction.

Table 1 summarizes the properties satisfied by the recognizers. Researchers distinguish unistroke from multistroke recognizers, considering the latter requires articulation invariance to classify all possible articulations of a multistroke in the number, order, and direction of its strokes. All recognizers are sampling, scaling, and translation invariant, except Rubine being sensitive to the scale of the gestures. The unistroke recognizers from the $-family are rotation invariant thanks to the golden section search finding the best angular alignment between two gestures, but left this property after $N, which faces a combinatorial explosion to satisfy both articulation and rotation invariances. !FTL and !NFTL are also rotation invariant due to the local shape distance properties but do not satisfy the articulation invariance for multistroke gestures. μV is articulation, rotation, scaling, and translation invariant [50].

While the aforementioned recognizers vary in terms of their preprocessing and classification methods, a consensus emerges on the procedure for evaluating their performance [58, 72, 81]: a set of T templates per each of the C gesture classes is randomly selected for training, then one remaining template per gesture class is selected for testing. The process is repeated R times to average the results and observe their recognition rates. Following the method, the recognizers are initially tested in the *user-dependent scenario*, *i.e.,* when the same user produces templates used for training and testing the recognizers, and the most recent recognizers are tested in both user-dependent and *user-independent scenarios*, *i.e.,* when templates used for training are produced by a set of P users who are different from those producing candidates for testing [12, 81, 88].

2.2 Definition of the 2D Stroke Gesture Recognition Problem

We define a 2D stroke gesture when captured by an input device as a set of timestamped points with their optional features, sometimes called *traces* [39]: $g = \{p_i = (x_i, y_i, f_i, t_i)\}, \forall i \in \overline{n}=\{1, ..., n\}$ where x_i, y_i are the 2D Cartesian coordinates of each gesture point, t_i is the time stamp (*e.g.,* a sequence of integers, the date and time expressed in ISO 8601 as year, month, day, hour, minutes, seconds, and milliseconds), and f_i are optional features characterizing the gesture g (*e.g.,* the pressure, the local acceleration, the user category, the input device). Let C_l be a class of all gestures of the same type with its label l: $C_l = \{g_{lj}\}, \forall j \in \overline{m}=\{1, ..., m\}$. Typical labels l are the name of the symbol, letter, or figure.

We therefore define a *gesture set* (also called *library* L [39]) as a set $\mathcal{G}$ of predefined, finite $k \in \mathbf{N}$ gesture classes: $\mathcal{G} = \{C_{kl}\} \forall k$. We also define a 2D stroke gesture recognizer R as a function $R: c \in \mathcal{G} \rightarrow$ score $s \in [0, ..., 1]$ that produces a score s comparing the similarity of a *candidate gesture* c to all gestures of all classes of $\mathcal{G}$. This score is null when nothing is recognized ($R(c) = 0$) and is unitary ($R(c) = 1$) when there is a perfect match (there is a single class of $\mathcal{G}$ for which $s(c, g) = 1$. The R function is typically calculated via a metric or a distance (not all metrics are a mathematical distance [80]) such as those listed in the rightmost column of Table 1).

Let $\mathcal{R}$ be the set of all possible recognizers R: $\mathcal{R} = \{R_1, R_2, R_3, ...\}$. Solving our research question RQ for a given gesture set $\mathcal{G}$ means selecting a recognizer $R_b \in \mathcal{R}$ solving the following objective function:

$$\max_{R_x \in \mathcal{R}} \frac{\sum_{c \in \mathcal{G}} R_x(c)}{\#(\mathcal{G})} \tag{1}$$

For the purpose of recognition, the gestures of the set $\mathcal{G}$ need to be resampled to have a number N of points equal to that of the template gestures that serve for training. It is also applicable to the candidate gesture. This is because the gesture input duration and movement speeds can vary between users, devices, sensors, and operating systems, even for the same intended gesture [39]. Resampling ensures that the gesture points are re-distributed to be at equal distances from each other. The following variables are defined to process the performance testing:

- Number of resampling points (N): integer representing the number of points to which all gestures are resampled before recognition. We observe that the most frequent values of N are powers of 2: 2 [65], 4 [57], 8 [57, 85], 16 [77, 78], 32 [48, 88], 64 [57], 128 [81, 83], and 256 [81, 88], but not always [15]. We can cluster these values in several intervals, such as *very low*= [6..8], *low*=]8..32], *medium*=]32..64], and *high*=]64,...[. We might expect a higher value of N to lead to a better recognition rate, but this is not necessarily the case: increasing the value of N positively impacts the recognition rate up to a certain threshold, which operates like a plateau effect. Beyond this threshold, the recognition rate no longer increases, but may even decrease. For example, Vatavu [81] shows that N=6 is enough for the Euclidean distance and that N=8 reaches the maximum. However, Cao and Zhai [17] use $N = 25$ for the Euclidean distance while Kratz and Rohs [39] use N=150 for the $3 recognizer. $N [5], $N-Protractor [6], and $1 [93] resample up to N=96 points. Maslych et al. [54] propose several techniques for resampling points, particularly for augmenting the gesture set with new templates. *Isometricity* is the property of a gesture set to hold a set of N-equally distanced points while *isochronicity* is the property of a gesture set to hold a set of N-equally timestamped points [48]. Some recognizers, however, require an intrinsic, predefined value: N=64 for $Q [83], between N=4 and N=40 for a portable low-end radar [71].
- Number of templates (T): integer representing the number of templates per gesture class used for training. Typically, $T \in \{1, 2, 4, 8, 16\}$ [57], but could be assigned to virtually any number depending on the recognizer. Zero-shot learning, one-shot learning, and few-shot learning could go up to 0, 1, and 4, respectively.
- Number of participants (P): integer representing the number of participants involved in each template gesture. Again, the value of this number can largely vary depending on the gesture set created and how many participants are considered in an experiment. For example, Erazo and Pérez Medina [27] considered P=19, P=20, and P=32 participants in their comparative analysis of recognizers. [88] did their analysis by varying the number of participants P=1..14. Calisto et al. [14] used P=31. The value of P is always less or equal to the total number of participants who collected gestures in the gesture set.
- Number of repetitions (R): integer representing the number of times a recognizer is tested on the given gesture set. While there is no optimal value of R, a consensus emerges to have at least R=100 repetitions for performance testing [85, 88]. This value also depends on the testing procedure. Cross-validation is a widely used technique for evaluating classifier performance, but its implementation can significantly impact results. Studies suggest that increasing the folds in k-fold cross-validation enhances average performance and reduces variability [35]. Repeatedly performing k-fold cross-validation may not be necessary, as accuracy estimates from different replications are often correlated, especially with a larger number of folds [95].

The Number of resampling points N should not be confused with the Resampling Rate, which defines the number of points generated per second and the time interval between two consecutive points [25], *e.g.,* 200 Hz.

3 Existing 2D Stroke Gesture Datasets

Table 2 compares existing 2D stroke gesture datasets with their reference and URL, if any, description, device/platform properties, the composition of the dataset, and annotation types. Published recognizers are tested on one or many datasets, usually a few of them and not the same ones, which involve their own input devices, gestures, and participants, which makes their testing impossible to compare under equivalent conditions.

$1 [93] is a unistroke dataset acquired for evaluating $1, and later Protractor, $N, $P, PP, and Jackknife (Figure 3, left). This dataset contains 4,800 samples of 16 uni-stroke gestures performed by 10 participants at 3 different speeds, *i.e.,* slow, medium, and fast, with a pen on a Pocket PC. The Synchromedia-Imadoc Gesture New On-Line Database (**SIGN**) [2] (Figure 3, center) contains data corresponding to on-line handwritten gestures for 17 classes associating specific gestures to edit functions (*e.g.,* copy digital ink elements), acquired on Tablet PCs and whiteboards.

Table 2. Comparison of existing 2D stroke gesture datasets.

Dataset	Description	Properties	Statistics	Annotations
$1 [93] (2007)	Unistroke pen-based gestures collected in three writing speeds to compare the performance of the $1 recognizer to other recognizers	HP iPAQ h4355 Pocket PC with pen, unistroke in XML format (X,Y)	10 subjects, 4800 gestures, 16 classes, 3 speeds	Each gesture is annotated with gesture class, user id, and writing speed
SIGN [2] (2010)	Handwritten gestures for testing gesture recognizers used in applications associating specific gestures to edit functions. Random order	Pen input on Tablet PC unistroke, in HBF49 format [24] (X,Y,T)	20 subjects, 33150 gestures, 17 classes	Each gesture is annotated with gesture class and user id
EDS-1 [88] (2011)	Familiar unistroke gestures collected for a gesture execution difficulty study	Wacom DTU-710, pen input, unistroke in XML format (X,Y,T)	14 subjects, 9 classes, 2520 gestures	Each gesture is annotated with gesture class and user id
EDS-2 [88] (2011)	Unfamiliar unistroke gestures collected for a gesture execution difficulty study	Wacom DTU-710, pen input, unistroke in XML format (X,Y,T)	14 subjects, 9 classes, 2520 gestures	Each gesture is annotated with gesture class and user id
MMG [6] (2012)	Multistroke pen gesture dataset collected in three writing speeds containing a set of symbols representative of those used in multistroke gesture input applications	Windows Tablet PC, Finger and Pen, Uni/Multistroke in XML format (X,Y)	20 subjects, 16 classes, 9600 gestures	Each gesture is annotated with gesture class, user id, type (finger, pen), and writing speed
MMG+ [50] (2022b)	Same gestures as in MMG, but equally distributed on a smartphone and a tablet	Huawei P20 Lite and Samsung Galaxy Tab A6. Uni/multistroke in JSON format (X,Y,T)	24 subjects, 16 classes, 5 samples for each class, 1920 gestures for each device, 3840 gestures	Each gesture is annotated with gesture class, user id, and device
RMMG+ [50] (2022b)	Same gestures as in MMG+, but synthetically generated by rotation	Uni/multistroke in JSON format (X,Y,T)	16 classes, 30 samples for each class, 4 rotations, 2 devices, 3840 gestures	Each gesture is annotated with gesture class, user id, and device
LaViola [42] (2007)	Multistroke dataset acquired to evaluate the recognition of symbols in a user-independent procedure		10 subjects, 10 classes, 20 samples per class, 2000 gestures	Each gesture is annotated with gesture class and user id
Low Vision [86] (2018a)	Multi-stroke dataset used to measure the impact of users with visual impairments on touch gesture articulation	Tablet PC TC 1100 with Pen, Uni/Multistroke in DAT format (X,Y)	34 subjects, 48 classes, 11602 gestures	Each gesture is annotated with gesture class and user id
HOMUS [16] (2014)	Handwritten Online Musical Symbols where each musician has their writing style. Collected at the musicians respective schools and conservatories	Samsung Galaxy Note 10.1 with S-Pen, multistroke in TXT format (X,Y)	100 subjects, 38 classes, 15200 gestures	Each gesture is annotated with gesture class and user id
NicIcon [91] (2009)	Iconic multi-stroke gestural commands	Wacom Intuos2 A4 oversize tablet. Pen, Multi-strokes in TXT format (X,Y,T,P)	34 subjects, 14 classes, 26163 gestures	Each gesture is annotated with gesture class and user id

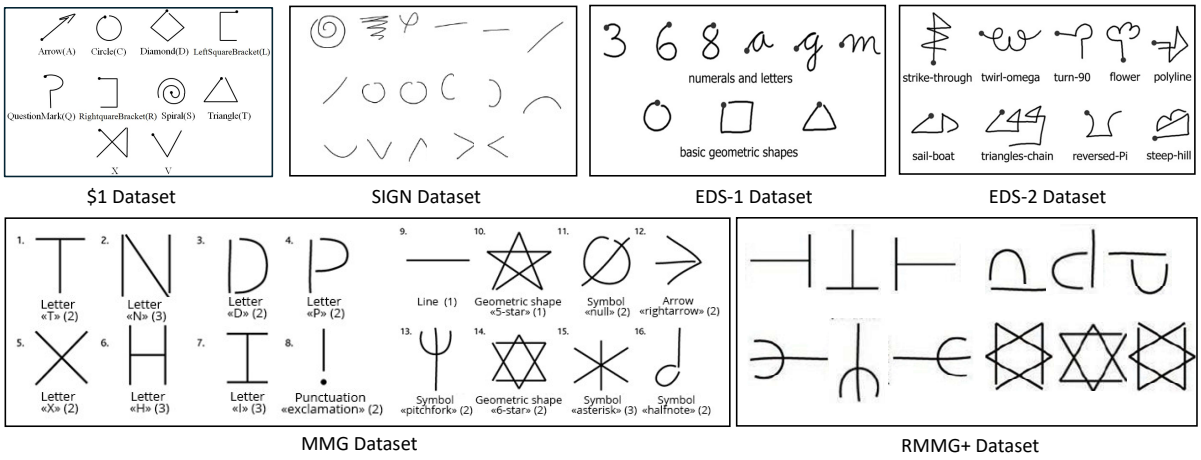


Fig. 3. Existing gesture datasets (1/2): $1 [93], SIGN [2], EDS-1/2 [88], MMG [6], and RMMG+ [50].

EDS-1 and **EDS-2** [88] are two unistroke datasets acquired to evaluate the execution difficulty of stroke gestures (Figure 3, right), and later to evaluate PP. The Mixed Multistroke Gestures (**MMG**) [6] is a multistroke dataset acquired to evaluate the performance of $N-Protractor [46], and later PP and $Q.

MMG+ and **RMMG+** are two multistroke datasets acquired to evaluate the performance of rotation-invariant recognizers [50]. These datasets contain the same gestures as in MMG, but are equally distributed on a smartphone and on a tablet. Inspired by synthetic gesture generation [44, 53, 75], RMMG+ synthesizes the templates from MMG+ by randomly rotating them between -360° and +360°. The 24 participants (6 Female and 18 Male) aged between 20 and 29 years old (M=21, SD=2.1) issued 5 samples for each gesture class on a smartphone (Huawei P20 Lite with a 5.84-inch screen) and 5 samples on a tablet (Samsung Galaxy Tab A6 with a 10.1-inch screen). Half of the participants performed gestures on the smartphone during their first session, and on the tablet during their second session, and vice versa. Similarly to implementing all recognizers in Javascript, we converted the datasets into a unified JSON format to process them uniformly in Javascript. **LaViola** [42] is a multi-stroke dataset acquired to evaluate the recognition of symbols in a user-independent procedure. Ten subjects produced 20 templates per symbol, ten per session.

Low vision [87] is a multistroke dataset assessing the impact of users with visual impairments on gesture articulation, and for evaluating $P+. Twelve gesture classes were used in the original paper, but the dataset includes four additional swipes, *i.e.,* up, down, right, and left, that require using recognizers sensitive to the rotation: a "swipe up" gesture is a "swipe down" rotated by 180°, for example. Half the participants involved in the acquisition of this dataset had visual impairments. Low Vision must not be confused with the $P+ dataset, which also involves low-vision users, but is smaller in number of participants, classes, and samples per class.

HOMUS [16] is a multi-stroke dataset acquired to evaluate the recognition of pen-based music notation. The templates were performed by musicians, *i.e.,* music teachers and advanced students, who were encouraged to draw the music symbols in their style. Therefore, we expect recognizers to achieve significantly lower accuracy between the user-dependent and user-independent scenarios due to within-user variations in gesture articulation.

Table 3 compares the devices, the modalities, the speed, and the participant distinction considered in the datasets mentioned above. Although most of the data sets contained gestures captured for a tablet, only a few included gestures for modern smartphones, which is the main reason for considering MMG+ and RMMG+. This shows the real need for datasets that contain a wide variety of gestures acquired on smartphones and no longer

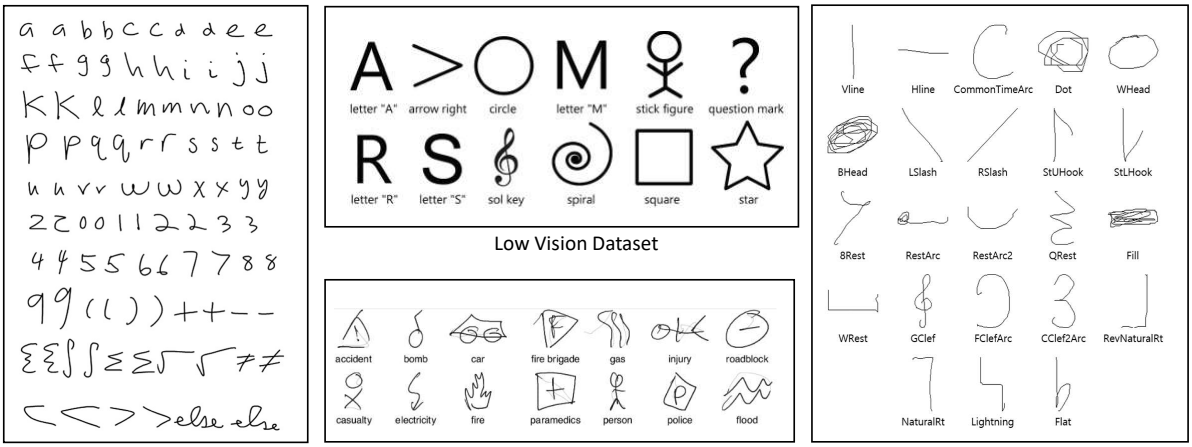


Fig. 4. Existing gesture datasets (2/2): LaViola [42], Low vision [87], HOMUS [16], and NicIcon [91].

on outdated devices. A similar observation can be made when it comes to modalities: while the pen has almost completely disappeared from common use today, fingers are still predominant. Only one dataset distinguishes gestures for visually impaired people.

In sum, several publicly available datasets contain stroke gestures, such as SIGN [2] for uni-strokes, and HOMUS [16], LaViola [42] and NicIcon [91] for multistroke respectively, in the contexts of music notations, symbolic gestures, and iconic gestures for crisis management (*e.g.,* fire, gas, or flood), respectively.

Table 3. Contextual information on the 11 datasets tested in the experiment. ○=absent feature, ●=present feature.

		Devices			Modalities		Speed		Participants	
Dataset	Year	Pocket PC	Smartphone	Tablet	Pen	Finger	Imposed	Free	No distinction	Low-vision
$1	2007	●	○	○	●	●	●	○	●	○
SIGN	2010	○	○	●	●	○	○	●	●	○
EDS-1	2011	○	○	●	●	○	○	●	●	○
EDS-2	2011	○	○	●	●	○	○	●	●	○
MMG	2010	○	○	●	●	●	●	○	●	○
MMG+	2020	○	●	●	○	●	○	●	●	○
RMMG+	2020	○	●	●	○	●	○	●	●	○
LaViola	2007	○	○	●	●	○	○	●	●	○
Low Vision	2018	○	○	●	○	●	○	●	○	●
HOMUS	2014	○	○	●	●	○	○	●	●	○
NicIcon	2009	○	○	●	●	○	○	●	●	○

4 Experiment

We perform comparative testing of the recognizers described in section 2 as well as a few of their variations listed hereafter on a series of datasets, according to the user-(in)dependent scenarios to obtain measures to evaluate the effectiveness of the proposed algorithms in one or many contexts of use.

4.1 Recognizers

Our evaluation requires the use of comparable recognizers written in the same programming language to guarantee the fairness of the measurements and the consistency of the experimental conditions. Javascript versions of the $P+, $Q,!FTL, and Jackknife recognizers being publicly available, we adopted Javascript as the common programming language for all recognizers. Hence, we translated the Penny Pincher algorithm [77] from Swift to Javascript, being inspired by the work of fe9lix. We translated Rubine [65] into Javascript as well, being inspired by the work of Wobbrock et al. [93]. Four modifications were applied to all recognizers to ensure a consistent treatment:

(1) Training instructions were removed (*e.g.,* the $-family pre-defines a few templates in the code).
(2) The Number of resampling points N is no longer a global variable but a part of each recognizer's constructor at run-time.
(3) Variables were prefixed with their recognizer's name to avoid any ambiguity because Javascript uses a shared scope across files (*e.g.,* two methods named resampling can be confused even though they belong to different files, however, PennyPincher_resampling and $1_resampling will never be confused).
(4) Four measurements of time, expressed in milliseconds (ms), are now computed:
(1) Training time, *i.e.,* the time required to pre-process all templates or build the classification model, (2) Pre-processing time, *i.e.,* the time required to preprocess one candidate, (3) Classification time, *i.e.,* the time required to classify one candidate, and (4) Recognition time, *i.e.,* the overall time required to recognize one candidate or the sum of preprocessing and classification times.

In sum, our experiment includes the following recognizers and their variations: Rubine (R), Protractor (Pro), the $-family represented by $1, $N, $N-Protractor ($NP), $P, $P+ and $Q, Penny Pincher (PP), two versions of Jackknife with the Euclidean distance (JED) and with the inner product (JIP) used as the local cost function, !FTL with the local shape distance and !NFTL with the normalized local shape distance. Compared to Erazo and Pérez Medina [27] who tested $1, $P, $Q, !FTL, and Penny Pincher on three gesture sets, *i.e.,*, NicIcon, HHReco, and Utopiano Alphabet, only in a user-independent scenario, our experiment is unique and about 25 larger in scope and in the testing effort.

To satisfy both the rotation and articulation invariances beyond other properties while avoiding the combinatorial explosion of $N, we included μV, a recognizer that combines $P+ and !FTL [50]. The dissimilarity between two gestures, represented by two sets of shapes, is computed as the sum of the Euclidean distances that separate each pair of corresponding shapes, *i.e.,* their local shape distance. This μV recognizer supports articulation invariance by using $P+'s cloud matching to match the corresponding shapes together, and translation, scaling, and rotation invariances by converting resampled points to !FTL shapes. Its name (μV) stands for multi-vector as a shape is a set of two vectors connecting three consecutive points.

4.2 Considered Datasets

After examining the related work in section 3, we collected 11 publicly available datasets containing 2D stroke gestures and converted them into a unified JSON form (Figure 1). The datasets consist of samples acquired in different contexts of use with different participants, on different input devices (*e.g.,* Pocket PCs, tablets, and smartphones), at different speeds (*i.e.,* low, medium, and fast), and in various modalities (*i.e.,* pen and finger). Table 3 summarizes these contextual conditions. All datasets used a tablet to acquire gesture samples, except $1

Table 4. Gesture classes of the 11 datasets tested in the experiment.

	Gesture types								Number of...			
	Swipes	Arrows	Digits	Letters	Punctuation	Shapes	Symbols	Patterns	Classes	Samples p. class	Participants	Samples
$1	-	1	-	2	5	4	4	-	16	30	11	5,280
SIGN	4	-	-	-	2	2	9	-	17	97-100	20	33,154
EDS-1	-	-	3	3	-	3	2	7	18	20	14	5,040
EDS-2	-	2	-	1	1	3	8	5	20	20	11	4,400
MMG	1	1	-	7	1	2	4	-	16	10	20	9,128
MMG+	1	1	-	7	1	2	4	-	16	10	24	3,840
RMMG+	1	1	-	7	1	2	4	-	16	10	24	3,840
LaViola	-	-	10	26	2	-	9	1	48	20	11	10,056
Low Vision	4	1	-	4	1	3	3	-	16	8-10	54	8,539
HOMUS	-	-	-	-	-	-	38	-	38	4	100	15,200
NicIcon	-	-	-	-	-	-	8	6	14	53	32	23,744

for which a Pocket PC was used, and MMG+ and RMMG+ for which both a smartphone and a tablet were used. $1 and MMG present two modalities as the participants performed samples distributed equally with a digital pen (stylus) and a finger. The other datasets consist of gestures performed with a pen or with a finger.

Finally, half of the participants involved in the low-vision dataset were selected because they had visual impairments to examine differences between low-vision and full-vision users. Due to all these differences, our comparative testing reports results when gestures are performed on multiple devices, with different modalities, at different speeds, by users with or without visual impairments, thus leading to multiple contexts of use.

The gesture types also vary from one dataset to another, except for MMG, MMG+, and RMMG+, which share the gesture set originally defined in MMG. Table 4 describes the composition of each gesture set by classifying their gesture classes into 8 categories: (1) swipes, (2) arrows (*i.e.,* swipes with a sharp end), (3) digits from 0 to 9, (4) letters from any alphabet, (5) punctuation marks, (6) geometric shapes, (7) symbolic gestures (*i.e.,* whose shapes refer to a common symbol), and (8) undefined patterns that do not fall into any other category. EDS-2, MMG, MMG+, and RMMG+ are interesting to evaluate recognizers, since they cover most categories (6/8). HOMUS consists of only musical symbols, which, conversely, narrows the variety of classes and makes it very specific.

The datasets considered in this work are also characterized by different gesture classes, *i.e.,* they contain a wide variety of gesture classes whose paths can be split into seven categories, as demonstrated by Table 2. For example, NicIcon consists exclusively of classes that belong to one category, *i.e.,* symbols and patterns, respectively. Diversified gesture sets include classes from 6 categories (6/7), as follows:

(1) **Taps.** Tapping once, twice, or multiple times in a short time interval is one of the most obvious ways to interact with a touch-sensitive surface. Nevertheless, only two datasets (2/11) consider a single tap or a double tap in their gesture set.

(2) **Swipes**. Swiping, *i.e.,* flicking from one side of the screen to the other, to navigate content is undoubtedly the most used gesture on smartphones and tablets. Surprisingly, only six datasets (6/11) consider swipes.
(3) **Letters and punctuation marks.** Most datasets (9/11) consider letters and punctuation marks. Some consider entire alphabets such as "Letters" which consists of all lowercase letters from the Latin alphabet, and "UJI" which contains multiple variations of each letter (*e.g.,* lowercase and uppercase). Some consider a combination of letters and punctuation marks, such as "xstroke" which consists of all relevant ASCII characters to act as a full keyboard replacement.
(4) **Digits.** Digits from 0 to 9 are part of a few gesture sets (2/11) among which only EDS-1 considers three digits instead of the existing ten.
(5) **Geometric shapes.** Most datasets (9/11) consider geometric shapes (*e.g.,* squares, rectangles, stars) that are well-known by end-users, easy to perform, and easy to remember, therefore perfectly suited for gesture-based interfaces.
(6) **Symbols.** All datasets (11/11) consider symbols, *i.e.,* marks that indicate, signify, or are understood as representing an idea, an object, or a relationship, for the same reasons.
(7) **Patterns or undefined paths**. We consider the gesture classes that do not fit any of the previous categories as "patterns", or "undefined paths", such as the ones defined in SHARK and SHARK^2 (a series of paths that connect letters on a virtual keyboard to form a word).

4.3 Measures

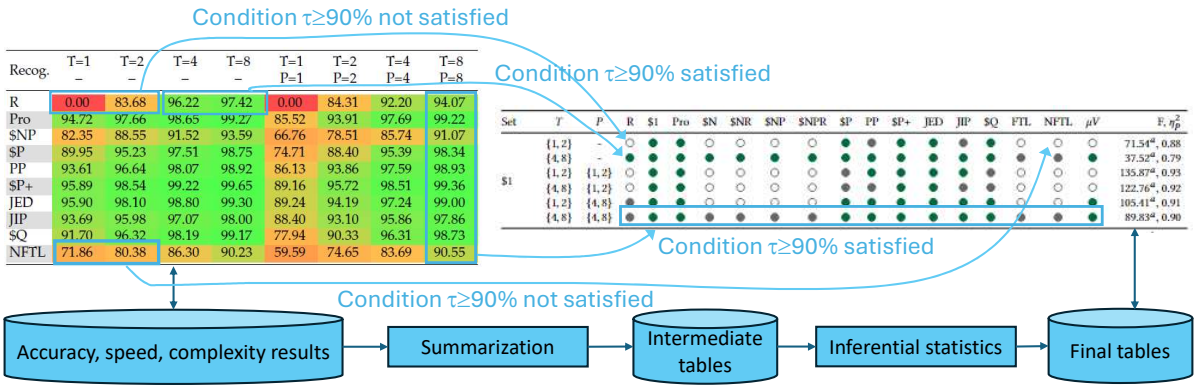
We first measure the accuracy of the recognition of all recognizers, computed by the Recognition rate [%], *i.e.,* the ratio of correctly recognized candidates to the total number of templates. The Recognition time [milliseconds, ms] is also measured since it represents the time required to provide a response to the end-user. In the following, we analyze four measurements of time. The training time captures the amount of ms required to preprocess all templates and train the recognizer. The recognition time captures the amount of ms required to recognize a candidate, which includes its preprocessing, *i.e.,* the preprocessing time plus its comparison with all stored templates, *i.e.,* the classification time.

To concisely report the individual recognition rates and times for each configuration (detailed figures are publicly available in our repository), we adopt a rate- and time-constrained classification approach [66, 98] by considering two conditions: a recognition rate $\tau \geq 90\%$ is considered as the minimum rate condition for a recognition system to gracefully operate [52, 67, 68, 90] and a recognition time of $t \leq 50$ ms is considered as the minimum system response time [19]. This condition is aligned with the admitted response time limits [56]: 0.1 sec when the end-user should feel that the system is reacting instantaneously, such as in direct manipulation, 1.0 sec when the end user's flow should stay uninterrupted, even with some delay, and 10 sec when the end user's attention should be kept focused, even for longer tasks.

For example, Figure 5 shows a partial table with the recognition rate obtained in various conditions for the $1 gesture set and how these results have been summarized in Table 5 by applying the conditions and inferential analysis to determine whether there is a significant difference for a solution.

4.4 Scenarios

Stroke recognizers are traditionally evaluated in two scenarios [6, 86]: *user-dependent*, when gestures used for training and testing are performed by the same user, and *user-independent*, when gestures used for training are performed by a set of users and for testing by a different user. The evaluation procedure is common to both scenarios: (1) a set of T templates per gesture class is randomly selected to train the recognizer, (2) one remaining template per gesture class is selected as a candidate for testing, (3) the procedure is repeated R times, and (4) the results are averaged for all participants and for all gesture classes.



Recog.	T=1 –	T=2 –	T=4 –	T=8 –	T=1 P=1	T=2 P=2	T=4 P=4	T=8 P=8
R	0.00	83.68	96.22	97.42	0.00	84.31	92.20	94.07
Pro	94.72	97.66	98.65	99.27	85.52	93.91	97.69	99.22
$NP	82.35	88.55	91.52	93.59	66.76	78.51	85.74	91.07
$P	89.95	95.23	97.51	98.75	74.71	88.40	95.39	98.34
PP	93.61	96.64	98.07	98.92	86.13	93.86	97.59	98.93
$P+	95.89	98.54	99.22	99.65	89.16	95.72	98.51	99.36
JED	95.90	98.10	98.80	99.30	89.24	94.19	97.24	99.00
JIP	93.69	95.98	97.07	98.00	88.40	93.10	95.86	97.86
$Q	91.70	96.32	98.19	99.17	77.94	90.33	96.31	98.73
NFTL	71.86	80.38	86.30	90.23	59.59	74.65	83.69	90.55

Fig. 5. Acquisition of measures and their treatment: the case of $1 gesture set.

The templates selected for training and the candidates for testing follow the rules of each scenario (*e.g.*, in the user-independent scenario with T=2 and P=2, a set of T templates is randomly chosen among the samples performed by the P participants for each of these participants, and one candidate per gesture class is chosen among the samples performed by a different participant). This testing method is a Leave-one-out cross-validation (LOOCV), a modified form of the k-fold cross-validation [74] popular in ML where k=1. In this way, the gesture sets are not divided into different training and testing sets, such that a gesture sample can be used for training and for testing in two consecutive repetitions, but not within the same repetition.

4.5 Apparatus

The evaluation is performed with Gester [51], a web application developed for this purpose running within the Google Chrome browser on a Dell Precision M6800 laptop equipped with an Intel Core i7-4810MQ 2.8GHz CPU and 16 Gb of RAM. This application, written in HTML5, CSS3, and Javascript, automates the testing of several recognizers on one or several datasets based on a configuration file capturing the testing parameters, *i.e.*, the number of resampling points N, of templates T, of participants P, of repetitions R, and which scenarios to simulate among user-dependent and user-independent. The results are exported in csv files that are easy to import into mainstream statistical programs.

4.6 Design

We evaluate the 16 recognizers on the unistroke datasets with T={1,2,4,8} templates in the user-dependent scenario, and T={1,2,4,8} templates per P={1,2,4,8} participants in the user-independent scenario, with R=100 repetitions in each condition. The recognizers were set up to resample raw gestures to N=8 points as Vatavu [81] suggested that N≥6 points are sufficient to reach a high recognition rate (τ≥90%) with Euclidean and angular recognizers. This design is reproduced for the multi-stroke datasets. Our design is therefore a within-factor study with five independent variables:

(1) Recognizer: nominal variable with 16 conditions, representing the 2D stroke gesture recognizers: $\mathcal{R}$ = {R, $1, Pro, $N, $NR, $NP, $NPR, $P, $P+,$Q, PP, JED, JIP, !FTL, !NFTL, μV }.
(2) Dataset: nominal variable with 11 conditions, representing the datasets: $\mathcal{G}$ = { $1, SIGN, EDS-1, EDS-2, MMG, MMG+, RMMG+, LaViola, Low Vision, HOMUS, NicIcon }.
(3) Number of resampling points (N): numerical variable with 2 conditions, representing the number of points to which all gestures are resampled before classification: N={8, 16}.

(4) Number of Templates (T): numerical variable with 4 conditions representing the number of templates per gesture class used for training: T={1, 2, 4, 8}.
(5) Number of participants (P): numerical variable with 4 values representing the number of participants: P={1, 2, 4, 8}.

5 Results and Discussion

With our apparatus, the evaluation took about 51.57 days of full-time computation with a maximum of 6.55 days for the HOMUS dataset with N=16 resampling points.

5.1 Recognition Rate on Unistroke Datasets

A series of Repeated Measures ANalysis Of VAriance (RM-ANOVAs) is conducted on the uni-stroke evaluation to compare the effect of the recognizers on recognition rate in both user-dependent and user-independent scenarios, respectively with a varying number of templates (T), and a varying number of templates per participant (T×P) provided for training. A series of Friedman tests is conducted to compare this effect when the data do not meet the normality tests. In all conditions, the recognizers have a significant effect on the recognition rate, *i.e.,* at least one recognizer achieves a significantly different recognition rate.

Table 5 reports the results obtained by the post-hoc analysis, *i.e.,* a series of two-stage linear step-up procedures [11]. The last column of Table 5 reports the results of the RM-ANOVAs, indicating that recognition rate is significantly different for at least one recognizer in all conditions and both scenarios (the F computed is provided in the legend, $p \leq .01^{**}$, and the effect size η_p^2 is added).

JIP and PP are significantly more accurate than their competitors on all uni-stroke datasets. The SIGN dataset is the most challenging unistroke dataset since all recognizers, except PP, JED, and JIP, failed to classify gestures accurately (τ<90%). Indeed, SIGN involves direction variant and rotation variant gestures (*e.g.,* clockwise *vs.* counter-clockwise circle, and swipe left *vs.* swipe right), which are difficult to distinguish for rotation invariant recognizers, *i.e.,* unistroke recognizers from the $-family, !FTL, and !NFTL, and for articulation invariant recognizers, *i.e.,* the multistroke recognizers from the $-family and μV. This underlines the importance of choosing the right recognizers, whose properties fit the gesture set and the contextual conditions. Conversely, the EDS-1 and EDS-2 datasets are the least challenging uni-stroke datasets since almost all recognizers managed to achieve more than 90% accuracy in all conditions.

$1. All recognizers, except R, $NP, $P, and NFTL, satisfy the rate condition $\tau \leq 90\%$ in the smallest conditions (T=2, and T=2, P=2). This condition becomes satisfied by all recognizers in the largest conditions (T=8, and T=8, P=8). Contrarily to other recognizers, $NP and NFTL do not reach 90% in the intermediate user-independent condition (T=4, P=4). Post-hoc analyses confirm that R, $NP, and NFTL achieve significantly lower recognition rates than the other recognizers.

EDS-1. All recognizers, except R and NFTL, satisfy the rate condition in the smallest user-dependent condition (T=2). Contrarily to other recognizers, R, $P, and NFTL do not satisfy this condition in the smallest user-independent condition (T=2, P=2). This condition becomes satisfied by all recognizers in the remaining conditions (T=8, and T=4, P=4 and T=8, P=8). Post-hoc analyses confirm that R, $P, $Q, and NFTL achieve significantly lower recognition rates than the other recognizers in all conditions. PP is also significantly less accurate than other recognizers in the small user-dependent condition (T=2).

EDS-2. All recognizers satisfy the rate condition. However, post-hoc analyses reveal that R, $NP, $P, and NFTL achieve significantly lower recognition rates than other recognizers in all conditions. $Q is also significantly less accurate than other recognizers in the small and large user-independent condition (T=2, P=2, and T=4, P=4), but not in the very large user-independent condition (T=8, P=8).

Table 5. Evaluation of the 16 recognizers on the unistroke datasets with N=8, and varying conditions of T templates in the user-dependent scenario, and T templates per P participants in the user-independent scenario. The recognizers failing the rate condition $\tau \geq 90\%$ **and/or** the time condition $t \leq 50$ ms are depicted by the ○ symbol. The recognizers achieving $\tau \geq 90\%$ **and** $t \leq 50$ ms are depicted by the ● symbol. The significantly most accurate recognizers according to the post-hoc analysis (a=$F_{15,150}$, b=$F_{15,300}$, c=$F_{15,195}$), *i.e.,* achieving significantly better recognition rate than others, and not significantly different from others, are depicted by ● if they achieve $\tau \geq 90\%$ and $t \leq 50$ ms, otherwise their significance is not shown.

Set	T	P	R	$1	Pro	$N	$NR	$NP	$NPR	$P	PP	$P+	JED	JIP	$Q	FTL	NFTL	μV	F, η_p^2
$1	{1, 2}	-	○	●	●	○	○	○	○	●	◐	●	●	◐	●	○	○	○	71.54[a], 0.88
	{4, 8}	-	●	●	●	●	●	●	●	●	●	●	●	●	●	◐	◐	●	37.52[a], 0.79
	{1, 2}	{1, 2}	○	●	●	○	○	○	○	◐	●	●	●	●	◐	○	○	○	135.87[a], 0.93
	{4, 8}	{1, 2}	○	●	●	○	○	○	○	◐	◐	●	●	◐	◐	○	○	○	122.76[a], 0.92
	{1, 2}	{4, 8}	◐	●	●	○	○	○	○	●	●	●	●	●	●	○	○	●	105.41[a], 0.91
	{4, 8}	{4, 8}	◐	●	●	◐	◐	◐	◐	●	●	●	●	●	●	◐	◐	●	89.83[a], 0.90
SIGN	{1, 2}	-	○	○	○	○	○	○	○	○	◐	○	○	●	○	○	○	○	435.27[b], 0.96
	{4, 8}	-	◐	○	○	○	○	○	○	○	◐	○	◐	●	○	○	○	○	379.07[b], 0.95
	{1, 2}	{1, 2}	○	○	○	○	○	○	○	○	◐	○	○	●	○	○	○	○	1789.54[b], 0.99
	{4, 8}	{1, 2}	○	○	○	○	○	○	○	○	●	○	◐	●	○	○	○	○	1370.96[b], 0.99
	{1, 2}	{4, 8}	○	○	○	○	○	○	○	○	●	○	◐	●	○	○	○	○	994.67[b], 0.98
	{4, 8}	{4, 8}	○	○	○	○	○	○	○	○	●	○	◐	●	○	○	○	○	951.30[b], 0.98
EDS-1	{1, 2}	-	○	●	●	●	●	●	●	◐	◐	●	●	●	◐	○	○	○	139.57[c], 0.91
	{4, 8}	-	◐	●	●	●	●	●	●	●	●	●	●	●	●	◐	◐	◐	42.48[c], 0.76
	{1, 2}	{1, 2}	○	●	●	●	●	●	●	◐	◐	●	●	●	◐	○	○	○	251.56[c], 0.95
	{4, 8}	{1, 2}	◐	●	●	●	●	●	●	◐	●	●	●	●	◐	○	◐	○	122.76[c], 0.92
	{1, 2}	{4, 8}	◐	●	●	●	●	●	●	◐	●	●	●	●	◐	◐	◐	◐	85.00[c], 0.87
	{4, 8}	{4, 8}	◐	●	●	●	●	●	●	●	●	●	●	●	●	◐	◐	◐	36.54[c], 0.74
EDS-2	{1, 2}	-	◐	●	●	◐	◐	◐	◐	◐	●	●	●	●	●	◐	◐	◐	25.33[a], 0.71
	{4, 8}	-	●	●	●	●	●	●	●	●	●	●	●	●	●	●	◐	●	11.20[a], 0.53
	{1, 2}	{1, 2}	◐	◐	◐	◐	◐	◐	◐	◐	●	◐	●	●	◐	◐	◐	◐	47.58[a], 0.83
	{4, 8}	{1, 2}	●	●	●	◐	◐	◐	◐	●	●	●	●	●	●	◐	◐	◐	23.21[a], 0.70
	{1, 2}	{4, 8}	●	●	●	●	●	●	●	●	●	●	●	●	●	◐	◐	◐	15.49[a], 0.61
	{4, 8}	{4, 8}	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	◐	9.22[a], 0.48

SIGN. PP and JIP are the only recognizers satisfying the rate condition in all conditions, whereas JED does it in the large user-dependent condition (T=8), and the large and very large user-independent conditions (T=4, P=4, and T=8, P=8). Post-hoc analyses confirm that PP and JIP are significantly more accurate than other recognizers in all conditions, and JED is significantly more accurate than all recognizers, except PP and JIP, in the very large user-independent condition (T=8, P=8).

5.2 Recognition Rate on Multistroke Datasets

A second series of RM-ANOVAs is conducted on the multistroke datasets to compare the effect of the recognizers on recognition rates in the user-dependent and user-independent scenarios with the same conditions of T and P. A series of Friedman tests is also conducted to compare this effect when the data do not follow a normal distribution. In all conditions, the recognizers have a significant effect on recognition rates, *i.e.,* at least one recognizer achieves a significantly different recognition rate.

Table 6 reports the results obtained by the post-hoc analysis, *i.e.,* a series of two-stage linear step-up procedure of Benjamini et al. [11]. Overall, the recognizers achieved a recognition rate on the multistroke datasets lower than on the unistroke datasets. Most recognizers satisfy our two conditions on the MMG and MMG+ datasets,

but achieve a rate less than 90% on any other multistroke dataset. First, because LaViola and HOMUS involve much larger gesture sets (resp., 48 and 38 gesture classes compared to all datasets: M=16, SD=11). Second, the subjects involved in the HOMUS dataset were encouraged to perform gestures in their own styles, increasing between-user differences in drawing. Third, they performed only four samples per gesture class, which limits training the recognizers to T=3 templates.

The NicIcon dataset requires rotation invariance to differentiate a lying body symbol from a standing one and allows two fuzzy patterns, *i.e.,* wind and gas, that are known to be difficult to recognize for Euclidean and angular recognizers because of the randomness involved in their gesture paths. Finally, Low Vision includes four directional swipes requiring direction invariant recognition, whereas multistroke recognizers are strictly articulation invariant, *i.e.,* they are insensitive to variations in the stroke direction. As expected, only the rotation invariant multistroke recognizers, *i.e.,* $N, $NP, and μV, achieve high recognition rates on RMMG+, the rotation invariant dataset. In terms of recognition time, $N and related variants faced a combinatorial explosion on the Low Vision, HOMUS, and NicIcon datasets, as they involve gestures with more than four strokes. This explosion resulted in exceeding the $t \leq 50$ ms time condition for classifying any candidate from those datasets.

MMG. $NP and its variants satisfy the rate condition in all cases. All the other recognizers, except R, Pro, and NFTL, go beyond 90% in the user-dependent conditions (T=2 and T=4) and in the large and very large user-independent conditions (T=4, P=4, and T=8, P=8). However, PP and JIP do not reach 90% in the large user-independent conditions (T=8, P=8). Post-hoc analyses suggest that $NP and $P+ are significantly more accurate than all other recognizers in all conditions, and that $NP, $P+, and $Q are significantly more accurate than all other recognizers in the very large user-independent condition (T=8, P=8).

LaViola. $P+ and JED satisfy the rate condition in the large user-dependent condition (T=8) and the very large user-independent condition (T=8, P=8). In all other conditions, no recognizer does satisfy the rate condition. Post-hoc analyses confirm that $P+ and JED are significantly more accurate than all other recognizers in all conditions.

NicIcon. R, Pro, $P+, JED, and JIP satisfy the rate condition in the large user-dependent condition (T=8). $NP and its variants face a combinatorial explosion because the dataset contains a few samples performed with four or more strokes, which leads the algorithms to exceed the 1GB memory limit of Google Chrome when generating all multistroke permutations. Post-hoc analyses confirm that R, Pro, $P+, JED, and JIP achieve significantly better accuracy in the very large user-dependent condition (T=8).

HOMUS. No recognizer satisfies the rate condition for two potential reasons: (1) participants were encouraged to draw gestures in their style, increasing variations in gesture articulation, and (2) only four samples are available for each class, constraining the number of training templates. Similarly to the NicIcon dataset, $NP faces a combinatorial explosion when manipulating multistroke samples from HOMUS.

Low Vision. PP and JIP satisfy the rate condition in both user-dependent conditions (T=2 and T=4). JED achieves a rate of more than 90% in the large user-dependent condition (T=8) and the very large user-independent condition (T=8, P=8). Similarly to NicIcon and HOMUS, $NP faces a combinatorial explosion. Post hoc analyses confirm that PP and JIP are significantly more accurate than all other recognizers in all conditions.

In conclusion, the accuracy observed on multistroke datasets is far below the accuracy observed on unistroke datasets. Depending on the dataset, $NP, R, Pro, $P+, PP, JED, and JIP are significantly more accurate than the other recognizers. Nevertheless, $NP and its variants face a combinatorial explosion on NicIcon, HOMUS, and Low Vision datasets because they include samples containing four or more strokes. Since many recognizers do not satisfy our rate condition on multistroke datasets with N=8, we decided to increase the number of resampling points to N=16 in a second evaluation. We conduct the same analysis, reporting a significant effect of the recognizers on recognition rate in all conditions, *i.e.,* at least one recognizer achieves significantly different recognition rates.

Table 6. Evaluation of the recognizers on the multistroke datasets with N=8 resampling points, and varying conditions of T templates in the user-dependent scenario and T templates per P participants in the user-independent scenario. Refer to Table 5 for the meaning of the ○, ●, and ● symbols. The significantly most accurate recognizers according to the post-hoc analysis (a=$F_{15,150}$, b=$F_{15,300}$, c=$F_{15,195}$, d=$F_{15,270}$, e=$F_{15,345}$, f=$F_{11,572}$, g=$F_{11,1089}$, h=$F_{11,352}$), *i.e.,* achieving significantly better recognition rates than others, and not significantly differing from others, are depicted by the ● symbol if they achieve $\tau \geq 90\%$ and $t \leq 50$ ms, otherwise their significance is not shown. The ● symbol depicts the recognizers which either did not complete their execution correctly (*e.g.,* $N and $N-Protractor with a combinatorial explosion), which exceeded the time limit of 50 ms (in fact, none), or which did not satisfy the $\tau \geq 90\%$ recognition rate (many recognizers on multistroke datasets).

Set	T	P	R	$1	Pro	$N	$NR	$NP	$NPR	$P	PP	$P+	JED	JIP	$Q	FTL	NFTL	μV	F, η_p^2
MMG	{1, 2}	-	○	○	○	●	●	●	●	●	○	●	●	●	●	○	○	○	218.22[d], 0.92
	{4, 8}	-	●	●	●	●	●	●	●	●	●	●	●	●	●	○	○	●	127.80[d], 0.88
	{1, 2}	{1, 2}	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	○	571.08[d], 0.97
	{4, 8}	{1, 2}	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	○	391.89[d], 0.96
	{1, 2}	{4, 8}	○	○	○	●	●	●	●	●	●	●	●	●	●	○	○	●	189.12[d], 0.91
	{4, 8}	{4, 8}	○	●	●	●	●	●	●	●	●	●	●	●	●	○	○	●	137.70[d], 0.88
MMG+	{1, 2}	-	○	○	○	●	●	●	●	●	○	●	●	●	●	○	○	●	146.81[e], 0.86
	{4, 8}	-	●	●	●	●	●	●	●	●	●	●	●	●	●	○	○	●	104.88[e], 0.82
	{1, 2}	{1, 2}	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	○	270.47[e], 0.92
	{4, 8}	{1, 2}	○	○	○	●	●	●	●	●	○	○	○	○	●	○	○	○	176.96[e], 0.88
	{1, 2}	{4, 8}	○	○	○	●	●	●	●	●	○	●	○	○	●	●	●	●	114.09[e], 0.83
	{4, 8}	{4, 8}	○	●	○	●	●	●	●	●	●	●	○	○	●	●	●	●	76.77[e], 0.77
RMMG+	{1, 2}	-	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	●	1500.35[e], 0.98
	{4, 8}	-	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	●	590.08[e], 0.96
	{1, 2}	{1, 2}	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	○	1333.66[e], 0.98
	{4, 8}	{1, 2}	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	○	655.64[e], 0.97
	{1, 2}	{4, 8}	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	●	377.08[e], 0.94
	{4, 8}	{4, 8}	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	●	302.54[e], 0.93
LaViola	{1, 2}	-	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	234.65[a], 0.96
	{4, 8}	-	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	159.16[a], 0.94
	{1, 2}	{1, 2}	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	368.90[a], 0.97
	{4, 8}	{1, 2}	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	193.94[a], 0.95
	{1, 2}	{4, 8}	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	373.55[a], 0.97
	{4, 8}	{4, 8}	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	171.80[a], 0.94
Low Vis.	{1, 2}	-	○	○	○	○	○	○	○	○	●	○	●	●	○	○	○	○	553.33[f], 0.91
	{4}	-	●	○	○	○	○	○	○	○	●	●	●	●	○	○	○	○	552.73[f], 0.91
	{1, 2}	{1, 2}	●	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	545.69[f], 0.91
	{4}	{1, 2}	●	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	487.10[f], 0.90
	{1, 2}	{4, 8}	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	488.53[f], 0.90
	{4}	{4, 8}	●	○	○	○	○	○	○	○	●	○	●	●	○	○	○	○	433.11[f], 0.89
HOMUS	{1, 2}	-	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	654.11[g], 0.87
	{1, 2}	{1, 2}	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	1407.91, 0.93[g]
	{1, 2}	{4, 8}	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	1186.20[g], 0.92
NicIcon	{1, 2}	-	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	395.84[h], 0.92
	{4, 8}	-	●	●	●	○	○	○	○	○	○	●	●	●	○	○	○	○	250.57[h], 0.88
	{1, 2}	{1, 2}	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	179.43[h], 0.85
	{4, 8}	{1, 2}	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	96.55[h], 0.75
	{1, 2}	{4, 8}	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	○	285.23[h], 0.90
	{4, 8}	{4, 8}	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	○	239.25[h], 0.88

Table 7. Evaluation of the recognizers on four multistroke datasets with N=16 resampling points with conditions of T and P. Refer to Table 5 and Table 6 for the meaning of the ○, ●, ●, and ● symbols. (a= $F_{15,150}$, b=$F_{11,572}$, c=$F_{11,1089}$, d=$F_{11,352}$.

Set	T	P	R	$1	Pro	$N	$NR	$NP	$NPR	$P	PP	$P+	JED	JIP	$Q	FTL	NFTL	μV	F, η_p^2
LaViola	{1, 2}	-	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	333.35[a], 0.97
	{4, 8}	-	○	○	○	○	○	○	○	○	●	●	●	●	○	○	○	○	320.58[a], 0.97
	{1, 2}	{1, 2}	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	438.45[a], 0.98
	{4, 8}	{1, 2}	○	○	○	○	○	○	○	○	●	●	●	●	○	○	○	○	417.45[a], 0.98
	{1, 2}	{4, 8}	○	○	○	○	○	○	○	○	●	●	●	●	○	○	○	○	466.67[a], 0.98
	{4, 8}	{4, 8}	○	○	○	○	○	○	○	○	●	●	●	●	○	○	○	○	344.20[a], 0.97
Low Vis.	{1, 2}	-	○	○	○	○	○	○	○	○	●	●	●	●	○	○	○	○	733.52[b], 0.93
	{4}	-	●	○	○	○	○	○	○	○	●	●	●	●	○	○	○	○	627.91[b], 0.92
	{1, 2}	{1, 2}	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	858.70[b], 0.94
	{4}	{1, 2}	○	○	○	○	○	○	○	○	○	●	○	●	○	○	○	○	582.45[b], 0.92
	{1, 2}	{4, 8}	○	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	532.38[b], 0.91
	{4}	{4, 8}	●	○	○	○	○	○	○	○	●	○	●	●	○	○	○	○	475.21[b], 0.90
HOMUS	{1, 2}	-	○	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	1004.06[c], 0.91
	{1, 2}	{1, 2}	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	2544.58[c], 0.96
	{1, 2}	{4, 8}	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	2662.06[c], 0.96
NicIcon	{1, 2}	-	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	584.58[d], 0.95
	{4, 8}	-	●	●	●	○	○	○	○	●	●	●	●	●	●	○	○	○	283.29[d], 0.90
	{1, 2}	{1, 2}	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	286.44[d], 0.90
	{4, 8}	{1, 2}	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	239.80[d], 0.88
	{1, 2}	{4, 8}	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	540.26[d], 0.94
	{4, 8}	{4, 8}	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	459.53[d], 0.93

Table 7 reports the results obtained by the post-hoc tests. All recognizers, except !FTL, increase their average recognition rates in all conditions with a slight increase in the user-dependent conditions (M=+2.56%, SD=3.36%) and a large increase in the user-independent conditions (M=+10.52%, SD=6.02%). Note that !FTL reduces its recognition rates in all conditions (M=-8.78%, SD=4.70%). JIP and JED fulfill both requirements on the Low Vision dataset, except with T=2 and P=2 in the user-independent scenario. Nevertheless, most recognizers fail to achieve more than 90% recognition accuracy despite the increase in resampling points, highlighting the inaccuracy of gesture recognizers designed for fast prototyping in classifying complex gesture sets with few templates ($T \times P \leq 4 \times 4$) and few resampling points ($N \leq 16$).

5.3 Impact of Input Device on Recognition Rate

The MMG+ dataset comprises gestures performed on multiple input devices as participants performed half of their samples with a smartphone and the other half with a tablet. Consequently, the dataset offers perspectives to evaluate both the within-subject impact of the input device and the impact of the recognizers on recognition rates. We verify two hypotheses in the following: (1) the recognition rates are not significantly different when the recognizers are trained and tested with gestures performed only on a smartphone or only on a tablet, and (2) the recognition rates are not significantly different when the recognizers are trained with gestures performed on a smartphone (respectively, a tablet) and tested with gestures performed on both devices. In other words, we try to demonstrate that all recognizers are device-independent, *i.e.,* they can achieve high recognition rates (>90%) even if the gestures are performed on different input devices.

First, a series of two-way ANOVAs are performed to compare the effect of recognizers and input devices on recognition rate in user-dependent and user-independent scenarios with varying conditions of T and P when recognizers are trained and tested with gestures performed only on a smartphone (resp., on a tablet). As observed in the previous sections, recognizers have a significant effect on recognition rates in all conditions. The first

Table 8. Two-way ANOVAs to study the effect of the input device on recognition rates. The average recognition rates [%] for all recognizers on each device are reported along with their standard deviations (*SD*).

Effect of the input device on recognition rate			
Conditions	Two-way ANOVA	Smartphone: *M* (*SD*) [%]	Tablet *M* (*SD*) [%]
T=2	$F_{1,644}$=1.18, p=.28	89.44% (3.40)	89.23% (3.65)
T=4	$F_{1,644}$=0.51, p=.47	93.54% (2.27)	93.75% (2.71)
T=2, P=2	$F_{1,644}$=21.61, p<.001***	70.15% (4.61)	72.39% (5.58)
T=4, P=4	$F_{1,644}$=16.49, p<.001***	82.29% (4.94)	84.31% (5.74)
Effect of the training input device on recognition rate			
Conditions	Two-way ANOVA	Smartphone *M* (*SD*) [%]	Tablet *M* (*SD*) [%]
T=2	$F_{1,644}$=35.88, p<.001***	86.40% (3.63)	88.56% (2.98)
T=4	$F_{1,644}$=76.69, p<.001***	90.64% (2.95)	93.18% (2.09)
T=2, P=2	$F_{1,644}$=5.37, p=.02	70.17% (5.24)	71.28% (5.12)
T=4, P=4	$F_{1,644}$=0.59, p=.44	88.19% (4.20)	88.53% (4.89)

part of Table 8 reports the results obtained to measure the effect of the input device on recognition rates. In the user-dependent conditions (T=2 and T=4), the input devices have no significant effect on recognition rates. In user-independent conditions (T=2, P=2, and T=4, P=4), the input devices have a significant effect on recognition rates. Post-hoc analysis, consisting of a series of two-stage linear step-up procedures of Benjamini et al. [11], reveals that !NFTL achieved significantly smaller recognition rates with gestures performed on a smartphone than with gestures performed on a tablet (*e.g.,* M=53.08%, SD=5.22% *vs.* M=58.54%, SD=6.00% in the T=2, P=2 condition). However, the other recognizers have no significant difference in recognition rates.

Second, a series of two-way ANOVAs are conducted to compare the effect of the recognizers and the training input devices on recognition rates in the user-dependent and user-independent scenarios with varying conditions of T and P when the recognizers are trained with gestures performed on a smartphone (resp., a tablet), and tested with gestures performed on both devices. As expected, the recognizers have a significant effect on recognition rates in all conditions. The second part of Table 8 reports the results obtained to measure the effect of the training input device on recognition rates. In the user-dependent conditions (T=2 and T=4), the training input devices have a significant effect on recognition rates. Post-hoc analysis reveals that R and !FTL (resp., R, $1, Protractor, !FTL and !NFTL) achieve significantly smaller recognition rates when they are trained with gestures performed on a smartphone than with gestures performed on a tablet in the T=2 (resp., T=4) condition. However, there is no significant difference in recognition rates in the user-independent conditions (T=2, P=2, and T=4, P=4). Thus, some recognizers are better trained with few samples (T<4) performed on a tablet than with few samples performed on a smartphone in the user-dependent scenario.

5.4 Impact of Visual impairments on Recognition Rate

The Low Vision dataset includes gestures performed by users with and without visual impairments. Consequently, the dataset offers perspectives to evaluate both the between-subject impact of visual impairment and the impact of the recognizer on the recognition rate. We verify two research questions: (1) the recognition rate is significantly different when the recognizers are trained and tested with gestures performed only by users with visual impairment, respectively without, which should confirm Vatavu's observation [83] with $P+, and (2) the

Table 9. Two-way ANOVAs to study the effect of visual impairments on recognition rates. The average recognition rates [%] for all recognizers with each type of user are reported along with their standard deviations (*SD*).

Effect of visual impairments on recognition rates			
Conditions	Two-way ANOVA	Full-vision M (SD) [%]	Low-vision M (SD) [%]
T=2	$F_{1,600}$=79.92, p<.001***	81.80% (0.88)	77.68% (5.34)
T=4	$F_{1,600}$=91.44, p<.001***	86.80% (0.85)	83.07% (4.38)
T=2, P=2	$F_{1,600}$=347.5, p<.001***	71.67% (1.29)	62.49% (5.34)
T=4, P=4	$F_{1,600}$=279.8, p<.001***	80.51% (1.10)	72.60% (5.48)
Effect of training with or without visual impairments on recognition rates			
Conditions	Two-way ANOVA	Full-vision M (SD) [%]	Low-vision M (SD)[%]
T=2, P=2	$F_{1,600}$=324.9, p<.001***	71.42% (3.72)	64.43% (3.92)
T=4, P=4	$F_{1,600}$=288.7, p<.001***	84.66% (2.88)	78.99% (3.31)

recognition rate is significantly different when the recognizers are trained with gestures performed by users with their visual impairments, respectively without, and tested with gestures performed by both types of users.

A series of two-way ANOVAs are performed to compare the effect of recognizers and visual impairments on the recognition rate in user-dependent and user-independent scenarios with varying conditions of T and P when recognizers are trained and tested with gestures performed by users with visual impairments, respectively without. As expected in all conditions, the recognizers have a significant effect on the recognition rates. The first part of Table 9 reports the results measuring the effect of the visual impairment on the recognition rate. In all conditions, the visual impairment has a significant effect on the recognition rate. Post-hoc analyses, *i.e.*,, a series of two-stage linear step-up procedures of Benjamini et al. [11], suggest that all recognizers, except R, PP, JED, JIP, and μV, achieve significantly smaller recognition rates with gestures performed by low-vision users than with gestures performed by full-vision users in the user-dependent conditions (M=80.38%, SD=4.86% *vs.* M=84.30%, SD=0.87%). Moreover, all recognizers achieved significantly smaller recognition rates with gestures performed by low-vision users in the user-independent conditions (M=69.63%, SD=4.51% *vs.* M=77.07%, SD=2.24%). This suggests that gestures performed by users without visual impairment are easier to recognize than gestures performed by users with visual impairment, as already observed [83].

A suite of two-way ANOVAs is performed to compare the effect of recognizers and visual impairments on the recognition rate in user-dependent and user-independent scenarios with the same conditions of T and P when the recognizers are trained with gestures performed by low-vision and full-vision users, respectively, and tested with gestures performed by both types of users. As expected in all conditions, the recognizer has a significant effect on the recognition rate. The second part of Table 9 reports the results obtained to measure the effect of training with gestures performed by users with or without visual impairment on the recognition rate. In all conditions, visual impairment has a significant effect on the recognition rate. Post-hoc analysis reveals that all recognizers achieve a smaller recognition rate when they are trained by gestures performed by low-vision users than with gestures performed by full-vision users. This suggests that recognizers are better trained on gestures performed by users without visual impairments to classify candidates performed by both types of users.

Table 10. Algorithmic complexity of the 14 recognizers with C=number of gesture classes, W=number of raw points, N=number of resampled points, T=number of templates, U=number of unistroke permutations, and S=number of strokes.

Recognizer	Complexity $O(...)$	Growth
R	$W + N + C$	Linear
$1	$W + N + T(N + N^2)$	Quadratic
Pro	$W + TN$	Linear
$N	$SW + N + TU(N + N^2)$	Quadratic
$NR	$SW + N + TU(N + N^2)$	Quadratic
$NP	$SW + TUN$	Linear
$NPR	$SW + TUN$	Linear
$P	$W + N + T\sqrt{N}(N + N^2)$	Quadratic
PP	$W + TN$	Linear
$P+	$W + N + T(N + N^2)$	Quadratic
JED	$W + T + N(\log(N) + T + TN)$	Linearithmic
JIP	$W + T + N(\log(N) + T + TN)$	Linearithmic
$Q	$W + N + TN + \sqrt{N}(N + N^2)$	Quadratic
!FTL	$W + TN$	Linear
!NFTL	$W + TN$	Linear
μV	$W + N + T(N + N^2)$	Quadratic

5.5 Recognition Time and Algorithmic Complexity

The $-family, Pro, PP, !(N)FTL, and μV are template matchers using an Euclidean, an angular, or a local shape distance, respectively, to compute the (dis)similarity between a set of templates and any candidate. The templates are preprocessed only once before classifying any candidate. Adding a template to the training set requires only the preprocessing of this new template, which takes less than 0.1 ms for all recognizers except for the $Q algorithm during our experiment. $Q requires calculating a look-up table for each template to reduce the time for comparing a candidate to a template, an operation requiring from 0.30 to 0.40 ms in our setup. These template-matching recognizers are suitable for the real-time addition of new gesture classes and new templates to these classes.

However, R, JED, and JID require the training of their classifier prior to the recognition of any candidate. Consequently, adding a template to the training set implies the retraining of the entire algorithm. Our measurements show that this training time increases as the number of templates and the number of gesture classes increase, *i.e.,* the more templates are provided the longer it takes to train a new classifier. With LaViola's 48 gesture classes, Rubine takes less than 10 ms with less than 32 templates per gesture class and less than 25 ms with 32 or 64 templates. But JED and JIP take less than 500 ms and 300 ms, respectively, to train a new classifier with a single template per gesture class and up to 24 s (2400 ms) and 22 s (2200 ms), respectively, with 64 templates. If Rubine is also suitable for the real-time addition of new gesture classes and new templates, practitioners should keep in mind that Jackknife in those conditions needs more than the 50-ms time condition with lots of templates and gesture classes.

Table 10 compares the recognizers according to their algorithmic complexity expressed in terms of the Bachman-Landau asymptotic notation [38]. While six recognizers share a linear complexity, two of them are quasi-linear (or linearithmic), which are still below the quadratic complexity of the six other recognizers. As the resampling rate N considered in the experiment is bounded by $N \leq 16$ and is not meant to tend to infinity in practice, practitioners

Table 11. Theoretical decision table for selecting a recognizer. The recognizers are highlighted in red when they require training with user-dependent samples, in green if they proved accurate when trained with user-independent samples, and in blue when a formal definition of all gesture classes should be provided instead of templates [49]. References are: xstroke [96], SED-NE (Squared Euclidean Distance Nonlinear Embedding) [101], G-Gene [18], SHARK^2 [40], Gestimator [99].

Strokes	Invariant to...	Modalities	Recognizers
Uni-	Sampling, Translation	Mouse	Rubine
	Sampling, Scaling, Translation	Pen	xstroke, SHARK^2 SED-NE, Gestimator, Jackknife
		Pen, Finger	Penny Pincher
	Sampling, Scaling, Translation, Rotation	Pen	$1, Protractor, !FTL
Multi-	Sampling, Scaling, Translation, Articulation	Pen	$P+, G-Gene
		Pen, Finger	$P, $Q
	Sampling, Scaling, Translation, Rotation, Articulation	Pen	$N, μV
3D Uni-	Sampling, Scaling, Translation	Hand	Gestimator
		Hand, Full-body	Jackknife

and designers have room for increasing further the N value with linear and linearithmic recognizers but they must be aware that quadratic recognizers will suffer swiftly from such increase. $NP is linear, exactly like Pro, !FTL, and !NFTL, but depends on the number of unistroke permutations U and the number of strokes S, which could become two important multiplicative factors as observed with three multistroke datasets that faced a combinatorial explosion of the unistroke permutations (U) due to the number of strokes (S) contained in some of their templates.

6 Design Implications

We now suggest some design implications for both practitioners and researchers willing to use recognizers for gesture-based UIs. For this purpose, we first consolidate our findings into an empirically based decision table to select a recognizer with a theoretical table. Then, we break our design implications into two categories: gesture acquisition/recognition and gesture set design. We end up with implications for further research and a discussion of some limitations.

6.1 General Implications

Table 11 suggests a theoretical decision table for practitioners and researchers to select a recognizer according to their invariance properties and modalities for gesture input [49]. Prior to using the table, the following questions should be answered:

- Does the gesture set include uni-stroke or multistroke gestures?
- What invariance properties should differentiate gesture classes?
- Are the end-users interacting with a mouse, a pen, and/or a finger?
- Are end-users available to acquire templates?

Table 12. Empirical decision table to select a recognizer suitable for criteria given the theoretical and empirical knowledge we have on state-of-art recognizers. [On-demand properties] – Significantly more accurate than tested recognizers – More accurate than other recognizers but not significantly – "L" stands for well-suited to Low-end devices – "E" stands for facing combinatorial Explosion – "S" stands for too Slow for real-time addition of new classes and templates.

Invariant to	Sampling	Sampling Translation	Sampling Translation Scaling [Rotation]	Sampling Translation Scaling Rotation	Sampling Translation Scaling – Articulation	Sampling Translation Scaling [Rotation] Articulation
Unistrokes	Rubine^{L}	Jackknife^{S} Penny Pincher^{L}	Protractor^{L}	$P $1^{L} !FTL^{L}	$P+ $Q	$N^{E} $N-Protractor^{LE}
Multistrokes		Jackknife^{S}			$P $P+ $Q	$N^{E} $N-Protractor^{LE}

If end-users are available for the target gesture UI, they could themselves acquire the templates so that user-dependent recognizers are appropriate. If end-users are not available, user-independent recognizers should be used instead. If end-users interact with low-resource devices, we favor Penny Pincher and $Q, which perform well with limited processing power and computational time. Table 11 classifies recognizers based on the conditions claimed and tested in their respective references, thereby providing a theoretical table based on properties only. This table is not based on empirical results and should therefore be complemented with a corresponding empirical decision table.

In this paper, we empirically evaluated the accuracy and speed of state-of-art recognizers by measuring their recognition rates and their average training, preprocessing, classification, and recognition times in both user-dependent and user-independent procedures. Table 12 suggests an empirical decision table based on this comparative testing so that practitioners can select a suitable recognizer according to a combined set of theoretical and empirical criteria. Each available recognizer is mapped to a series of properties and a stroke cardinality. Recognizers suitable for low-end devices are marked with the *L* letter. Significantly more accurate recognizers are highlighted in green and not significantly more accurate in red. Even though Jackknife does not satisfy articulation invariance, it proved highly accurate in classifying multistroke gestures, whereas the other unistroke recognizers performed poorly on multistroke datasets. On the other way around, multistroke recognizers, however, proved highly accurate on unistroke datasets.

6.2 Implications for Gesture Acquisition and Recognition

If the gesture set contains only unistroke gestures, we suggest using Jackknife with the inner product (JIP) and Penny Pincher (PP) with no particular recommendation regarding the number of templates (T), the number of independent participants (P) and the number of resampling points (N) since they all proved efficient in all scenarios. Indeed, JIP and PP are green in Table 5, satisfying both conditions on all the datasets. JIP are significantly

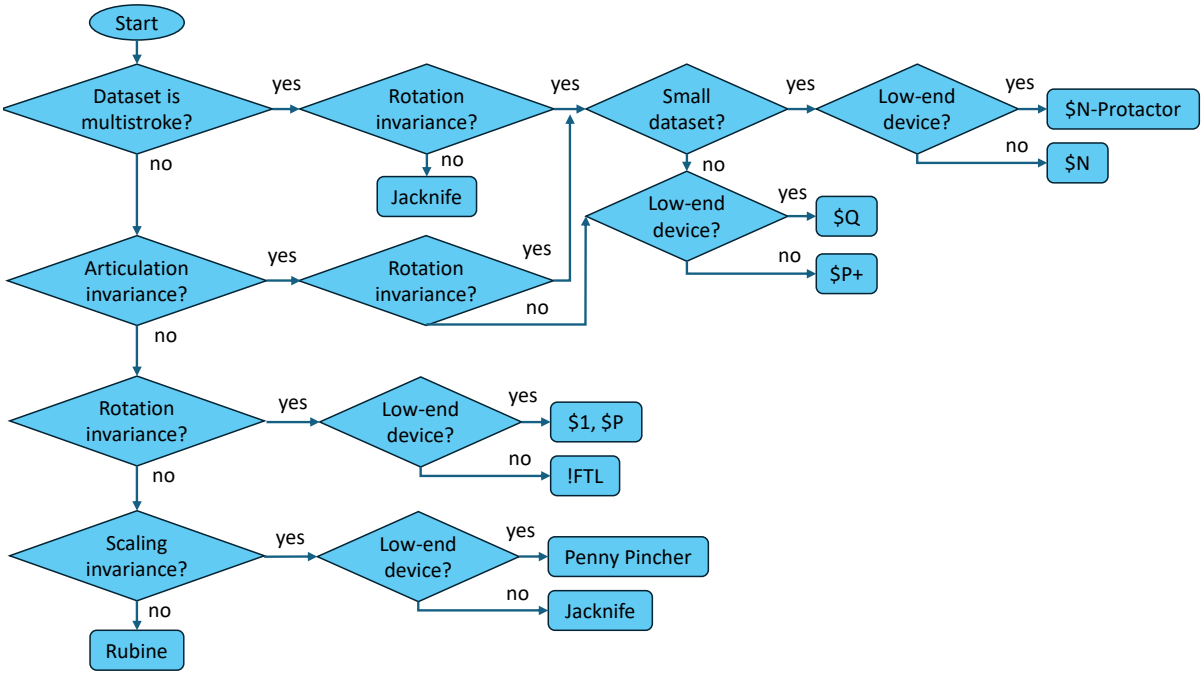


Fig. 6. Flowchart for selecting a recognizer.

more accurate than the other recognizers on all unistroke datasets with as few as N=8 resampling points. On most multistroke datasets, $P+ and JIP/JED are significantly more accurate than the other recognizers with N=16 resampling points, yet they do not satisfy our rate condition ($\tau \geq 90\%$).

If the gesture set contains multistroke gestures, we suggest using $N, $N-Protractor ($NP), $P+, and Jackknife with the inner product (JIP) or the Euclidean distance (JED) with a few recommendations regarding the number of resampling points ($N \geq 16$), templates ($T \geq 4$), and participants ($P \geq 4$) in the user-independent scenario. Indeed, although these recognizers do not satisfy both conditions on all datasets, they exhibit the best behaviors in Tables 6 and 7.

However, if the multistroke gestures are composed of four or more strokes, we suggest avoiding $N and $N-Protractor which are prone to a combinatorial explosion of unistroke permutations and considering μV instead, which does not suffer from this limitation. If the device used for recognition is limited in computational resources, we suggest using linear algorithms, such as Rubine (R), Protractor (Pro), $N-Protractor ($NP), Penny Pincher (PP), !FTL, and !NFTL. Quadratic algorithms should be discarded, especially when a large number of templates ($T \geq 8$), participants ($P \geq 8$), or resampling points ($N > 16$) are involved, even if their recognition rate is acceptable. We suggest using the !(N)FTL algorithms only with a small number of resampling points (N=8).

Figure 6 provides a flowchart for guiding the practitioner in selecting a 2D stroke gesture recognizer based on the topmost important questions. Applying this flowchart to select an appropriate 2D stroke gesture recognizer depending on the gesture set and its context(s) of use (as defined in our initial research question *RQ*) is a stage that can be integrated into a larger Software Development Life Cycle (SDLC) [37] of an interactive application using a gesture-based UI:

- *Vertically*: before this stage, gesture requirements and gesture elicitation studies [89, 92] are needed to create a gesture set with software such as GestureWiz [73] or Crowdlicit [1]. After this stage is completed, the recognizer selected as the appropriate candidate can be incorporated into the rest of the development life cycle with its dataset, within an Integrated Development Environment (IDE) such as iGesture [70], UsiGesture [12], GestUI [59], with a gesture toolkit, such as the Gesture Recognition Toolkit [31], or with an appropriate programming language [32].
- *Horizontally*: at the same level of this stage, there is no comparable software to Gester [51], but there are other initiatives that contribute to this level, such as iGesture [70], Gesture Recognition Software, or Hand Gesture Recognition Software).

6.3 Implications for Gesture Set Design

Table 11 suggests which recognizer to select according to the invariance properties of the gesture set. All are sampling invariant [33], most of them are rotation-, scaling-, and translation-invariant (RST), $N and μV are in addition articulation invariant (ARST). If the gesture set should satisfy rotation invariance, we suggest using $N, $N-Protractor ($NP), and μV, based on our results. If the gesture set should be modified by the end user in real-time, such as when adding, deleting, and replacing a template or a gesture class, we suggest avoiding Jackknife that exceeds the $t \leq 50$ ms condition for training.

If the gestures will be used on smartphones and tablets, we suggest training the recognizers with a gesture set acquired on a tablet with few templates and participants ($T \times P < 16$). If the templates are designed for end users with and without visual impairment, we suggest training the recognizers with gestures performed by users without visual impairment according to Table 9.

6.4 Implications for Future Research

Reproducibility in research and development is a necessary condition to acquire and to consolidate knowledge. Likewise, reproducibility in gesture recognition remains a requirement in three cases, expressed by ACM badges for reproducibility:

(1) *Repeatability* (same team, same experimental setup): the same results on recognition rates and execution times can be obtained with similar precision by the same team using the same procedure, the same software (here, Gester), under the same operating conditions (*e.g.*,, a similar computing platform), even with multiple trials. In our case, we should be able to reliably repeat our computations to obtain the same results.
(2) *Reproducibility* (different team, same experimental setup): the same results on recognition rates and execution times can be obtained with similar precision by another team using the same procedure, the same software, under the same operating conditions. In our case, an independent group can obtain the same results with Gester, our web application which can also be reused by any independent group to check our results. Similarly, we can verify the correctness of recognition rates and execution times of existing and new evaluations by reproducing the evaluation described in the respective papers introducing the recognizers, thus enabling us to confirm of disconfirm its results. In this case, matching the same experimental conditions would be challenging, especially if Gester will be used instead of the various, heterogeneous, respective procedures. For example, the same evaluation can be reproduced on another platform than the one used in our experiment. Therefore, the third case should be considered.
(3) *Replicability* (different team, different experimental setup): the same results on recognition rates and execution times can be obtained with similar precision by another team using another procedure, or software, under the same or different operating conditions. In our case, an independent group can obtain the same result using another procedure and another software than Gester.

Since our software Gester is publicly available [51], it can be reused to perform all the above cases of reproducibility and those specified in the RepliGES framework [30]. Beyond that, it can serve for metric benchmarking based on recognition rate and execution time. When a researcher introduces a new recognizer for a certain context of use (*e.g.,* games, virtual/augmented reality instead of Internet-of-Things), Gester could be used to demonstrate that existing recognizers do not fit well for this context of use and to perform a benchmarking by comparing the new recognizer to other ones in dedicated contextual conditions. In this vein, further research could address more general or specific research questions such as [51]: which are the n best recognizers for a series of contexts of use and, by corollary, which are the m worst ones?, which are the n best recognizers belonging to the same computational category?, which are the datasets that are the best recognized?, and within a gesture set, which gesture classes are the best recognized?

6.5 Limitations

Although our evaluation covered a significant set of state-of-the-art recognizers compared across reference datasets, it cannot be considered a complete evaluation because comparing all recognizers on all possible datasets remains an elusive problem that is computationally inefficient. Thus, we hereby discuss some limitations.

Limitations on datasets. Recognizers were tested on accessible, reference, and challenging datasets, and implications were drawn from these results. Another dataset may lead to another recognizer behavior than expected, particularly if the gesture classes largely differ from those classes studied. One can argue that some of these datasets contain "artificial" gestures that will be of little or no usage in real gesture-based UIs, especially compared to user-defined gestures [92]. Comparing recognizers with user-defined gestures instead of "artificial" gestures represents an interesting investigation. Participants could feel less constrained during gesture elicitation than during gesture acquisition, thus enlarging the scope of gesture articulation. Once a gesture has been decided, another session could take place to acquire the gesture templates more systematically than during elicitation.

Limitations on the Procedure. Recognizers were tested in user-dependent and user-independent scenarios, but only based on participants who acquired their gestures in the dataset. Reproducing this procedure with new participants acquiring a new dataset of the same gestures supports replicability. Ultimately, recognizers should be tested in real-world running applications based on usability testing to confirm or disconfirm the results. The Leave-one-out cross-validation (LOOCV), a modified form of the k-fold cross-validation [74] with k=1, was used for the user-independent scenario. Increasing the value of k should expand the coverage, but the number of combinations grows exponentially, such as 7 tests for k=3 instead of one test for k=1.

7 Conclusion

We performed comparative testing of 16 recognizers on 11 gesture sets satisfying different invariance properties with the goal of selecting recognizers suitable for a given context of use. To satisfy the rotation invariance, we included in the testing μV, a recognizer that does not suffer from a combinatorial explosion like \$N for multistroke gestures. We filtered the results based on constraints imposed on recognition rate ($\tau \geq 90\%$) and on execution time ($t \leq 50$ ms). An empirical decision table emerged from these results that complements a theoretical decision table based only on the invariance properties and modalities used for gesture input. By considering this set of state-of-the-art recognizers and a new one and by comparing them on the same series of gesture sets, we gained new insights into the conditions required for appropriately selecting recognizers for gesture-based UIs. Such comparative testing might pursue a variety of purposes, such as: verify that rate and time conditions are met for a certain template-based recognizer, consider ML/DL recognizers instead when the context of use requires more performance, decide whether a new recognizer under development is on track by benchmarking it, provide standard data for other quality assurance functions, validate suitability for given end-users, provide technical means of comparison of several parameters (such as variables tested or untested),

help answer research/development questions and solve gesture recognition problems with current recognizers, and help identify the potential benefits of a particular recognizer with respect to competitors. Based on our observations, we formulated some design implications to provide practitioners and researchers with some methodological guidance. Finally, we share the code of all mentioned recognizers in Javascript, and the datasets in a unified JSON format, and Gester, a web application used for this testing to be reused for three cases of reproducibility of gesture recognition.

Open Science. We share a Gester repository with the software developed for this paper, all recognizers written in JavaScript and datasets converted into a unified JSON format to foster reproducible research around gesture recognition and its incorporation into mainstream interactive applications. Any performance testing with Gester results in an archive composed of Comma Separated Values (CSV) files divided into multiple subfolders, each subfolder consisting of:

- Confusion matrices. Gester creates one set of confusion matrices per recognizer for all conditions, *i.e.,* all values of T templates per class and P independent participants. The first matrix reports results for all participants, whereas the remaining matrices report results for each participant individually. A row represents an actual class C_k while a column represents the recognition rate of a predicted class.
- Rates per class. The recognition rates obtained for each recognizer for each class are stored in a separate file called rates_class.csv. The rates are reported for all conditions at once, *i.e.,* all values of T and P. A row represents an actual class while a column represents the recognition rate achieved by one specific recognizer.
- Rates per participant. The recognition rates obtained by each recognizer for each of the P participants are stored in a separate file called rates_participant.csv. The rates are reported for all conditions at once, *i.e.,* all values of T and P, when samples from the participant are used for testing.
- Rates. The recognition rates obtained for each recognizer in each condition are stored in a separate file called rates_global.csv. A row represents a recognizer while a column represents the recognition rate for a condition.
- Training times. The training times are reported for each recognizer in a separate file called training_times.csv. Training time corresponds to the time required to train the recognizer (*e.g.,* to pre-process all templates or to generate the classification model from all templates).
- Pre-processing time. The pre-processing times are reported for each recognizer in a separate file called preprocessing_times.csv. Pre-processing time is the time required to pre-process a candidate.
- Classification time. The classification times are reported for each recognizer in a separate file called classification_times.csv. Classification time corresponds to the time required to classify a pre-processed candidate (*e.g.,* to compare it to all pre-processed templates).
- Recognition time. The recognition times are reported for each recognizer in a separate file called recognition_times.csv. The recognition time is the sum of the pre-processing and the classification times for each candidate. A row represents a combination of one testing participant and one recognizer while a column represents the average recognition time.

Acknowledgments

Nathan Magrofuoco is funded by the UCLouvain Fonds Speciaux de Recherche under Grant No. 61273304 and by FRIA under convention No. FC 31773. Arthur Sluÿters is funded by the "Fonds de la Recherche Scientifique - FNRS" under Grants n°40001931 and n°40011629. Jean Vanderdonckt is supported by the EU EIC Pathfinder-Awareness Inside challenge Symbiotik project (1 Oct. 2022-30 Sept. 2026) under Grant no. 101071147. The authors would like to thank Radu-Daniel Vatavu for the fruitful discussion on the topic of this article, Adrien Delaye for his help writing Table 11, and the EICS '25 reviewers for their constructive feedback on earlier versions of this paper.

References

[1] Abdullah X. Ali, Meredith Ringel Morris, and Jacob O. Wobbrock. 2019. Crowdlicit: A System for Conducting Distributed End-User Elicitation and Identification Studies. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300485

[2] Abdullah Almaksour, Eric Anquetil, Solen Quiniou, and Mohamed Cheriet. 2010. Personalizable Pen-Based Interface Using Lifelong Learning. In *Proceedings of the 2010 12th International Conference on Frontiers in Handwriting Recognition (ICFHR '10)*. IEEE Computer Society, USA, 188—-193. https://doi.org/10.1109/ICFHR.2010.37

[3] Michelle Annett, Albert Ng, Paul Dietz, Walter F. Bischof, and Anoop Gupta. 2014. How Low Should We Go? Understanding the Perception of Latency While Inking. In *Proc. of Graphics Interface 2014* (Montreal, Quebec, Canada) *(GI '14)*. Canadian Information Processing Society, Canada, 167––174.

[4] Lisa Anthony, Radu-Daniel Vatavu, and Jacob O. Wobbrock. 2013. Understanding the consistency of users' pen and finger stroke gesture articulation. In *Proc. of Graphics Interface 2013* (Regina, SK, Canada) *(GI '13)*, Faramarz F. Samavati and Kirstie Hawkey (Eds.). Canadian Information Processing Society / ACM, Mississauga, Canada, 87–94. http://dl.acm.org/citation.cfm?id=2532145

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

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

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

[8] Ramdas Pandurang Bagawade, Pournima Akash Chavan, and Kajal Kantilal Jadhav. 2017. Multi touch Gesture Generation and Recognition Techniques: A Study. *International Journal of Advanced Research in Computer and Communication Engineering* 6, 3 (2017), 5–9. https://doi.org/10.17148/IJARCCE.2017.6302

[9] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2008. Flower Menus: A New Type of Marking Menu with Large Menu Breadth, within Groups and Efficient Expert Mode Memorization. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (Napoli, Italy) *(AVI '08)*. Association for Computing Machinery, New York, NY, USA, 15–22. https://doi.org/10.1145/1385569.1385575

[10] Ruben Balcazar, Francisco R. Ortega, Katherine Tarre, Armando Barreto, Mark Weiss, and Naphtali D. Rishe. 2017. CircGR: Interactive Multi-Touch Gesture Recognition Using Circular Measurements. In *Proc. of the ACM Int. Conf. on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 12—-21. https://doi.org/10.1145/3132272.3134139

[11] Yoav Benjamini, Abba M. Krieger, and Daniel Yekutieli. 2006. Adaptive linear step-up procedures that control the false discovery rate. *Biometrika* 93, 3 (09 2006), 491–507. arXiv:https://academic.oup.com/biomet/article-pdf/93/3/491/1080958/933491.pdf https://doi.org/10.1093/biomet/93.3.491

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

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

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

[15] Francisco Maria Calisto, Carlos Santiago, Nuno Nunes, and Jacinto C. Nascimento. 2022. BreastScreening-AI: Evaluating medical intelligent agents for human-AI interactions. *Artificial Intelligence in Medicine* 127 (2022), 102285. https://doi.org/10.1016/j.artmed.2022.102285

[16] J. Calvo-Zaragoza and J. Oncina. 2014. Recognition of Pen-Based Music Notation: The HOMUS Dataset. In *Proc. of the 22nd Int. Conf. on Pattern Recognition* (Stockholm, Sweden) *(ICPR '04)*. IEEE, Piscataway, NJ, USA, 3038–3043. https://doi.org/10.1109/ICPR.2014.524

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

[18] 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

[19] Zhaoxin Chen. 2017. *Recognition and interpretation of multi-touch gesture interaction.* Ph. D. Dissertation. INSA de Rennes, France. https://tel.archives-ouvertes.fr/tel-01578068

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

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

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

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

[24] Adrien Delaye and Eric Anquetil. 2013. HBF49 feature set: A first unified baseline for online symbol recognition. *Pattern Recognition* 46, 1 (2013), 117–130. https://doi.org/10.1016/j.patcog.2012.07.015

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

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

[27] Ana Belén Erazo and Jorge Luis Pérez Medina. 2020. Algorithmic Efficiency of Stroke Gesture Recognizers: a Comparative Analysis. *International Journal on Advanced Science, Engineering and Information Technology* 10, 2 (Mar. 2020), 438–446. https://doi.org/10.18517/ijaseit.10.2.10807

[28] Clive Frankish, Richard Hull, and Pam Morgan. 1995. Recognition Accuracy and User Acceptance of Pen Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '95)*. ACM Press/Addison-Wesley Publishing Co., USA, 503–510. https://doi.org/10.1145/223904.223972

[29] Vittorio Fuccella and Gennaro Costagliola. 2015. Unistroke Gesture Recognition Through Polyline Approximation and Alignment. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3351–3354. https://doi.org/10.1145/2702123.2702505

[30] Bogdan-Florin Gheran, Santiago Villarreal-Narvaez, Radu-Daniel Vatavu, and Jean Vanderdonckt. 2022. RepliGES and GEStory: Visual Tools for Systematizing and Consolidating Knowledge on User-Defined Gestures. In *Proceedings of the ACM International Conference on Advanced Visual Interfaces* (Frascati, Rome, Italy) *(AVI 2022)*, Paolo Bottoni and Emanuele Panizzi (Eds.). ACM, New York, NY, USA, 5:1–5:9. https://doi.org/10.1145/3531073.3531112

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

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

[33] H. Hse and A. R. Newton. 2004. Sketched symbol recognition using Zernike moments. In *Proc. of the 17th Int. Conf. on Pattern Recognition* (Cambridge, UK) *(ICPR '04., Vol. 1)*. IEEE, Piscataway, NJ, USA, 367–370. https://doi.org/10.1109/ICPR.2004.1334128

[34] Heloise Hse, Michael Shilman, and A. Richard Newton. 2004. Robust Sketched Symbol Fragmentation Using Templates. In *Proc. of the 9th Int. Conf. on Intelligent User Interfaces* (Funchal, Madeira, Portugal) *(IUI '04)*. Association for Computing Machinery, New York, NY, USA, 156–160. https://doi.org/10.1145/964442.964472

[35] Yalcin Isler, Ali Narin, and Mahmut Ozer. 2015. Comparison of the Effects of Cross-validation Methods on Determining Performances of Classifiers Used in Diagnosing Congestive Heart Failure. *Measurement Science Review* 15, 4 (2015), 196–201. https://doi.org/10.1515/msr-2015-0027

[36] Allan Christian Long Jr., James A. Landay, Lawrence A. Rowe, and Joseph Michiels. 2000. Visual similarity of pen gestures. In *Proc. of the ACM Int. Conf. on Human factors in computing systems* (The Hague, The Netherlands) *(CHI '00)*, Thea Turner and Gerd Szwillus (Eds.). Association for Computing Machinery, New York, NY, USA, 360–367. https://doi.org/10.1145/332040.332458

[37] Iyad Khaddam, Hanaa Barakat, and Jean Vanderdonckt. 2016. Enactment of User Interface Development Methods in Software Life Cycles. In *13th International Conference on Human Computer Interaction, RoCHI 2016, Iasi, Romania, September 8-9, 2016*, Adrian Iftene and Jean Vanderdonckt (Eds.). Matrix Rom, Bucarest, Romania, 26–35. https://rochi.utcluj.ro/articole/4/RoCHI2016-Khaddam.pdf

[38] Donald E. Knuth. 1976. Big Omicron and Big Omega and Big Theta. *SIGACT News* 8, 2 (April 1976), 18––24. https://doi.org/10.1145/1008328.1008329

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

[40] Per-Ola Kristensson and Shumin Zhai. 2004. SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) *(UIST '04)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/1029632.1029640

[41] Mary LaLomia. 1994. User acceptance of handwritten recognition accuracy. In *Conference Companion on Human Factors in Computing Systems* (Boston, Massachusetts, USA) *(CHI '94)*. Association for Computing Machinery, New York, NY, USA, 107–108. https://doi.org/10.1145/259963.260086

[42] Joseph J. LaViola Jr. and Robert C. Zeleznik. 2007. A Practical Approach for Writer-Dependent Symbol Recognition Using a Writer-Independent Symbol Recognizer. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 11 (Nov. 2007), 1917–1926. https://doi.org/10.1109/TPAMI.2007.1109

[43] Emanuele Ledda and Lucio Davide Spano. 2021. Applying Long-Short Term Memory Recurrent Neural Networks for Real-Time Stroke Recognition. In *Companion of the 2021 ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Virtual Event, Netherlands) *(EICS '21)*. Association for Computing Machinery, New York, NY, USA, 50–55. https://doi.org/10.1145/3459926.3464754

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

[45] Luis A. Leiva, Radu-Daniel Vatavu, Daniel Martín-Albo, and Réjean Plamondon. 2020. Omnis Prædictio: Estimating the full spectrum of human performance with stroke gestures. *International Journal of Human-Computer Studies* 142 (2020), 102466. https://doi.org/10.1016/j.ijhcs.2020.102466

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

[47] Lorenzo Luzzi and Paolo Roselli. 2020. The shape of planar smooth gestures and the convergence of a gesture recognizer. *Aequationes mathematicae* 94 (2020), 219–233. https://doi.org/10.1007/s00010-020-00712-7

[48] Nathan Magrofuoco, Paolo Roselli, Jorge Luis Pérez-Medina, Jean Vanderdonckt, and Santiago Villarreal-Narvaez. 2019. Vector-based, Structure Preserving Stroke Gesture Recognition (P). In *Proceedings of the 25th International DMS Conference on Visualization and Visual Languages* (Hotel Tivoli, Lisbon, Portugal) *(DMSVIVA 2019)*, Joseph J. Pfeiffer Jr. (Ed.). KSI Research Inc. and Knowledge Systems Institute Graduate School, Skokie, Illinois, United States, 58–74. https://doi.org/10.18293/DMSVIVA2019-013

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

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

[51] Nathan Magrofuoco, Jean Vanderdonckt, and Paolo Roselli. 2025. Performance Testing of Stroke Gesture Recognizers with Gester. *Proc. ACM Hum.-Comput. Interact.* 9, EICS, Article 123 (June 2025), 25 pages. https://doi.org/10.1145/3427326

[52] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. 2016. Hand Gesture Recognition with Jointly Calibrated Leap Motion and Depth Sensor. *Multimedia Tools Appl.* 75, 22 (November 2016), 14991–15015. https://doi.org/10.1007/s11042-015-2451-6

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

[54] Mykola Maslych, Eugene Matthew Taranta, Mostafa Aldilati, and Joseph J. Laviola. 2023. Effective 2D Stroke-Based Gesture Augmentation for RNNs. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 282, 13 pages. https://doi.org/10.1145/3544548.3581358

[55] Vivian Genaro Motti and Jean Vanderdonckt. 2013. A computational framework for context-aware adaptation of user interfaces. In *Proc. of IEEE 7th International Conference on Research Challenges in Information Science* (Paris, France) *(RCIS '13)*, Roel J. Wieringa, Selmin Nurcan, Colette Rolland, and Jean-Louis Cavarero (Eds.). IEEE, Piscataway, NJ, USA, 1–12. https://doi.org/10.1109/RCIS.2013.6577709

[56] Jakob Nielsen. 1993. *Usability Engineering* (1 ed.). Morgan Kaufmann, Burlington, Massachusetts, United States. https://www.sciencedirect.com/book/9780125184069/usability-engineering

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

[58] Mehdi Ousmer, Arthur Sluÿters, Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. 2022. A Systematic Procedure for Comparing Template-Based Gesture Recognizers. In *Proceedings of 24th International Conference on Human-Computer Interaction, HCII 2022, June 26 - July 1, 2022 (Lecture Notes in Computer Science, Vol. 13519)*, Masaaki Kurosu, Sakae Yamamoto, Hirohiko Mori,

Dylan D. Schmorrow, Cali M. Fidopiastis, Norbert A. Streitz, and Shin'ichi Konomi (Eds.). Springer, Heidelberg, Germany, 160–179. https://doi.org/10.1007/978-3-031-17618-0_13

[59] Otto Parra, Sergio España, Jose Ignacio Panach, and Oscar Pastor. 2019. An empirical comparative evaluation of gestUI to include gesture-based interaction in user interfaces. *Science of Computer Programming* 172 (2019), 232–263. https://doi.org/10.1016/j.scico.2018.12.001

[60] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C (2nd Ed.): The Art of Scientific Computing*. Cambridge University Press, USA.

[61] Janet C. Read, Stuart MacFarlane, and Chris Casey. 2003. 'Good Enough for What?': Acceptance of Handwriting Recognition Errors by Child Users. In *Proceedings of the ACM Int. Conf. on Interaction Design and Children* (Preston, England) *(IDC '03)*. Association for Computing Machinery, New York, NY, USA, 155. https://doi.org/10.1145/953536.953565

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

[63] Yosra Rekik, Radu-Daniel Vatavu, and Laurent Grisoni. 2014. Understanding Users' Perceived Difficulty of Multi-Touch Gesture Articulation. In *Proceedings of the 16th International Conference on Multimodal Interaction* (Istanbul, Turkey) *(ICMI '14)*. Association for Computing Machinery, New York, NY, USA, 232–239. https://doi.org/10.1145/2663204.2663273

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

[65] Dean Rubine. 1991. Specifying Gestures by Example. In *Proc. of the 18th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. ACM, New York, NY, USA, 329–337. https://doi.org/10.1145/122718.122753

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

[67] Ugo Braga Sangiorgi, Vivian Genaro Motti, François Beuvens, and Jean Vanderdonckt. 2012. Assessing lag perception in electronic sketching. In *Proc. of ACM Int. Nordic Conf. on Human-Computer Interaction* (Copenhagen, Denmark) *(NordiCHI '12)*, Lone Malmborg and Thomas Pederson (Eds.). Association for Computing Machinery, New York, NY, USA, 153–161. https://doi.org/10.1145/2399016.2399040

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

[69] Alex Shaw, Jaime Ruiz, and Lisa Anthony. 2021. A Survey on Applying Automated Recognition of Touchscreen Stroke Gestures to Children's Input. *Interacting with Computers* 32, 5-6 (04 2021), 524–547. https://doi.org/10.1093/iwc/iwab009 arXiv:https://academic.oup.com/iwc/article-pdf/32/5-6/524/38856860/iwab009.pdf

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

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

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

[73] 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* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3173574.3173681

[74] M. Stone. 1974. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)* 36, 2 (1974), 111–133. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x

[75] Eugene M. Taranta, Mehran Maghoumi, Corey R. Pittman, and Joseph J. LaViola. 2016. A Rapid Prototyping Approach to Synthetic Data Generation for Improved 2D Gesture Recognition. In *Proc. of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 873––885. https://doi.org/10.1145/2984511.2984525

[76] Eugene M. Taranta, Corey R. Pittman, Jack P. Oakley, Mykola Maslych, Mehran Maghoumi, and Joseph J. LaViola. 2020. Moving Toward an Ecologically Valid Data Collection Protocol for 2D Gestures in Video Games. In *Proc. of the ACM Int. Conf. on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1––11. https://doi.org/10.1145/3313831.3376417

[77] Eugene M. Taranta, Andrés N. Vargas, and Joseph J. LaViola. 2016. Streamlined and Accurate Gesture Recognition with Penny Pincher. *Comput. Graph.* 55, C (April 2016), 130—-142. https://doi.org/10.1016/j.cag.2015.10.011

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

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

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

[81] Radu-Daniel Vatavu. 2011. The Effect of Sampling Rate on the Performance of Template-Based Gesture Recognizers. In *Proc. of the 13th Int. Conf. on Multimodal Interaction* (Alicante, Spain) *(ICMI '11)*. Association for Computing Machinery, New York, NY, USA, 271—-278. https://doi.org/10.1145/2070481.2070531

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

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

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

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

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

[87] Radu-Daniel Vatavu, Bogdan Gheran, and M.D. Schipor. 2018. The Impact of Low Vision on Touch-Gesture Articulation on Mobile Devices. *IEEE Pervasive Computing* 17, 1 (2018), 27–37. https://doi.org/10.1109/MPRV.2018.011591059

[88] Radu-Daniel Vatavu, Daniel Vogel, Géry Casiez, and Laurent Grisoni. 2011. Estimating the Perceived Difficulty of Pen Gestures. In *Proc. of the 13th IFIP TC 13 Int. Conf. on Human-Computer Interaction, Part II* (Lisbon, Portugal) *(INTERACT '11)*. Springer, Berlin, Heidelberg, 89—-106.

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

[90] Chong Wang, Zhong Liu, and Shing-Chow Chan. 2015. Superpixel-Based Hand Gesture Recognition With Kinect Depth Camera. *IEEE Transactions on Multimedia* 17, 1 (2015), 29–39. https://doi.org/10.1109/TMM.2014.2374357

[91] Don Willems, Ralph Niels, Marcel van Gerven, and Louis Vuurpijl. 2009. Iconic and multi-stroke gesture recognition. *Pattern Recognition* 42, 12 (2009), 3303–3312. https://doi.org/10.1016/j.patcog.2009.01.030 New Frontiers in Handwriting Recognition.

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

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

[94] Catherine G. Wolf. 1990. Understanding Handwriting Recognition from the User's Perspective. *Proceedings of the Human Factors Society Annual Meeting* 34, 4 (1990), 249–253. https://doi.org/10.1177/154193129003400402 arXiv:https://doi.org/10.1177/154193129003400402

[95] Tzu-Tsung Wong and Po-Yang Yeh. 2020. Reliable Accuracy Estimates from k-Fold Cross Validation. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2020), 1586–1594. https://doi.org/10.1109/TKDE.2019.2912815

[96] Carl D. Worth. 2003. xstroke: Full-screen Gesture Recognition for X. In *Proceedings of the FREENIX Track: 2003 USENIX Annual Technical Conference* (San Antonio, Texas, USA). USENIX Association, Berkeley, California, United States, 187–196. http://www.usenix.org/events/usenix03/tech/freenix03/worth.html

[97] Momona Yamagami, Claire L. Mitchell, Alexandra A Portnova-Fahreeva, Junhan Kong, Jennifer Mankoff, and Jacob O. Wobbrock. 2024. Customized Mid-Air Gestures for Accessibility: A $B Recognizer for Multi-Dimensional Biosignal Gestures. *CoRR* abs/2409.08402

(2024), 1–20. https://doi.org/10.48550/ARXIV.2409.08402 arXiv:2409.08402

[98] Ying Yang, Geoff Webb, Kevin Korb, and Kai Ming Ting. 2007. Classifying under computational resource constraints: anytime classification using probabilistic estimators. *Machine Learning* 69 (10 2007), 35–53. https://doi.org/10.1007/s10994-007-5020-z

[99] Yina Ye and Petteri Nurmi. 2015. Gestimator: Shape and Stroke Similarity Based Gesture Recognition. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction* (Seattle, Washington, USA) *(ICMI '15)*. Association for Computing Machinery, New York, NY, USA, 219–226. https://doi.org/10.1145/2818346.2820734

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

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

Received 26 July 2024; revised 25 October 2024; revised 15 December 2024; revised 9 January 2025