!FTL, an Articulation-Invariant Stroke Gesture Recognizer with Controllable Position, Scale, and Rotation Invariances

Jean Vanderdonckt
Université catholique de Louvain
Louvain Research Institute in Management and Organizations
Louvain-la-Neuve, Belgium
jean.vanderdonckt@uclouvain.be

Paolo Roselli
Università degli Studi di Roma Matematica Dipartimento
Roma, Italy
Université catholique de Louvain
Institut de recherche en mathématique et physique
Louvain-la-Neuve, Belgium
roselli@mat.uniroma2.it
paolo.roselli@uclouvain.be

Jorge Luis Pérez Medina
Universidad de las Américas
Intelligent & Interactive Systems Lab
Quito, Ecuador
Université catholique de Louvain
Louvain Research Institute in Management and Organizations
Louvain-la-Neuve, Belgium
jorge.perez.medina@udla.edu.ec
jorge.perezmedina@uclouvain.be

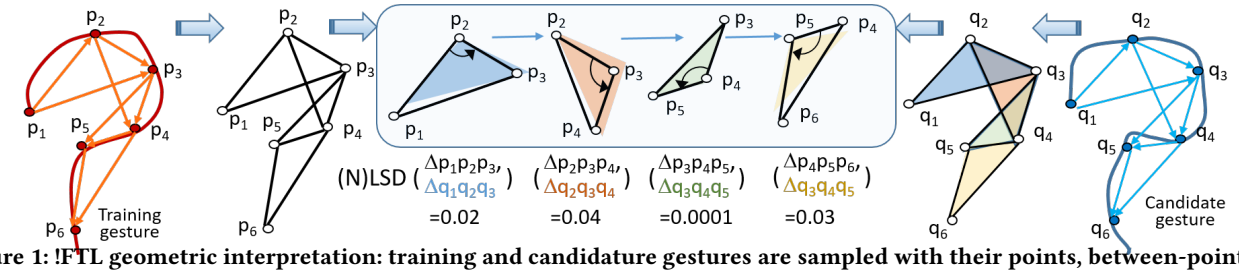


Figure 1: !FTL geometric interpretation: training and candidature gestures are sampled with their points, between-points vectors initiate basic triangles, Local Shape Distance (LSD) computes their respective similarity.

ABSTRACT

Nearest neighbor classifiers recognize stroke gestures by computing a (dis)similarity between a candidate gesture and a training set based on points, which may require normalization, resampling, and rotation to a reference before processing. To eliminate this expensive preprocessing, this paper introduces a vector-between-vectors recognition where a gesture is defined by a vector based on geometric algebra and performs recognition by computing a novel Local Shape Distance (LSD) between vectors. We mathematically prove the LSD position, scale, and rotation invariance, thus eliminating the preprocessing. To demonstrate the viability of this approach, we instantiate LSD for n=2 to compare !FTL, a 2D stroke-gesture recognizer with respect to $1 and $P, two state-of-the-art gesture recognizers, on a gesture set typically used for benchmarking. !FTL benefits from a recognition rate similar to $P, but a significant smaller execution time and a lower algorithmic complexity.

KEYWORDS

Stroke gesture recognition; Articulation invariance; Isometricity; Isochronicity; Isoparameterization; Local Shape Distance.

ICMI '18, October 16–20, 2018, Boulder, CO, USA

ACM ISBN 978-1-4503-5692-3/18/10…$15.00
https://doi.org/10.1145/3242969.3243032

ACM Reference Format:
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 *ICMI '18: 2018 Int'l Conference on Multimodal Interaction, Oct. 16–20, 2018, Boulder, CO, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3242969.3243032

1 INTRODUCTION

Gesture-based user interfaces [10, 14, 53], which are largely supported by operating systems, have become widely used in many domains of interactivity, such as hand recognition [27, 34], diagram sketching [16, 31], user interface prototyping [17], on-line food order [8], handwriting [26], and mobile commands [43]. Operating systems usually support a closed vocabulary of standard gestures, thus raising the need to integrate new, potentially user-defined, gestures to form an expandable vocabulary. Gesture recognizers rely on several techniques, such as machine learning [14], data mining [13, 25], template-based matching [56], and pattern recognition [20, 28]. Many recognizers, but not all, adhere to the Nearest Neighbor Classification (NNC) [20], which recognizes a *candidate gesture* issued by a user among a finite set of *reference gestures*, referred to as the *training set*, by computing a distance between them. The *k*-Nearest Neighbors algorithm (k-NN) is a non-parametric method used for classifying an object with respect to a class of objects among its *k* nearest neighbors [25]. In 1-NN, its simplified version with k=1, the candidate object is simply assigned to the class of that single nearest neighbor. Probably the most prolific NNC manifestation for stroke recognition is the $-family of recognizers [1, 32, 33, 44, 51, 57, 57, 63] thanks to a series of significant advantages demonstrated.

Efficient implementation. NNC algorithms imply only basic mathematical operations and functions, as opposed to complex trigonometric functions, feature extraction [45], filtering techniques, machine learning with training and calibration [14]. These algorithms lead to an affordable implementation in nearly all programming languages, benefit from a small amount of lines of code and a reasonable order of computational complexity.

Low execution time. The average time required to recognize a candidate gesture from a training set is very low, with no observed negative effect on lagging [9], such as for low-end devices.

High recognition rate. The estimated percentage of gestures that are correctly recognized by a gesture recognizer on a training set is high, typically ranging from 90%up to 99% for some datasets in some particular contexts of use [2, 56, 63].

Low memory consumption. The amount of RAM used for recognizing gestures is very limited. NNC processes a simple data structure based on an efficient implementation, thus consuming a limited amount of resources [58, 63].

Possible geometric interpretation. A candidate gesture is recognized as soon as it is near (or close) to a reference gesture in terms of points captured for both gestures. Since no particular machine learning or modeling technique is used, the algorithm is subject to a straightforward geometric interpretation that is manageable and understandable by human being while being easily processable by machine algorithms, although they need less points than human to recognize a gesture [56].

Independence of the algorithm with respect to the training set. When the training set is edited, such as for adding a new gesture, for removing an existing one, or for modifying one, the algorithm remains untouched contrarily to other techniques which need to be trained again on the resulting set, restructured on the underlying model, or updated on an expanded search graph [57].

Gesture adaptation. A gesture-based user interface is *adaptable* by the end user for editing the training set (e.g., when a system-provided gesture is replaced by a user-defined gesture to remember it better) and/or *adaptive* by the system when gesture-based commands could be redefined (e.g., a new function is offered through a new gesture, existing gestures are re-purposed, synthetic gestures [36] enrich the training set, the system suggests a new, sufficiently distinguishable from others, gesture that the end user may accept, reject, or modify) [48].

These requirements are particularly suitable for devices suffering from low computational power, such as ring devices, game controllers, wrist watches, body-mounted devices, and finger, hand, wrist, arm-based wearable devices. For all these advantages, the $-family of gesture recognizers [44] has been proved successful in the community to assist practitioners (e.g., designers, developers, usability engineers, human factors experts) in incorporating gesture-based user interfaces in today's interactive applications.

A gesture is captured by an input device as raw data defined as $G = \{p_i=(x_i, y_i, z_i, t_i)\}, i \in \overline{n}=\{1, ..., n\}$ where x_i, y_i, z_i are the 3D Cartesian coordinates of each gesture point, t_i is the time stamp. When a gesture is captured on an interactive surface, like in pen-based computing [35], z_i is not captured. Sometimes t_i is also optional when the gesture shape matters more than its motion or t_i is fed with a continuous identifier ID_i. When a gesture is captured by an input device with more parameters, like 6DOF for optical 3D trackers, pressure, velocity, acceleration, etc., G is then characterized by a n-D vector: $G = \{p_i=(x_i, y_i, z_i, p_{i1}, p_{i2}, ..., p_{ij}, ..., p_{im}, t_i)\}$ where p_{ij} denotes the j^{th} variable of the i^{th} point. The variables could be either captured (e.g., 3D coordinates and time stamp) or calculated (e.g., the velocity is the first derivative of position with respect to time, the acceleration is the second derivative, and the rate of change of acceleration, also known as jerk, is the third derivative). Gestures can be represented as sets of points or features [16, 27, 45] with comparable recognition rates [63]. NNC may require a combination of the following pre-recognition steps:

- *Normalization*: all points need to fit in the $\mathcal{D}=[0..1]^2 \subset \mathbb{R}^2$ unit square to be properly compared. This normalization is sometimes mandatory [57].
- *Resampling*: all points of any candidate gesture need to be resampled into a set of equally-distanced points according to the reference gesture in the training set. Resampling in the arc length domain is preferred to resampling in time in order to make recognizers invariant to articulation speed [56], but should be captured with the same sampling rate.
- *Rotation to a reference point*: to be effectively compared, a candidate gesture needs to be rotated to a reference point, such as 0°, with respect to the centroid, or to any reference angle according to which the training set has been recorded.

These three pre-processing steps can ensure NNC position, scale, and rotation invariance, but negatively affect its overall performance and its complexity. In order to address these challenges, we contribute NNC for stroke gesture recognition by fundamental and practical contributions:

- Instead of a point-based gesture representation, any gesture is defined by vectors expressed with respect to three properties: isochronicity, isometricity, and iso-parametrization.
- Instead of a point-to-point computation, a Local Shape Distance (LSD) distance measures the dissimilarity of two or more gestures represented as n-D vectors.
- Instead of being position, scale, and rotation variant, the LSD distance is mathematically proved as being position, scale, and rotation invariant after applying respectively a translation, a homogeneous dilatation, and a rotation for any 2D, 3D, or nD gesture, thus eliminating the need for the three pre-processing steps.
- !FTL, an algorithm for 2D stroke gesture recognition based on an instantiation of the general LSD formula for n=2, with its pseudo-code for implementation in any language (Fig. 1).
- A GeoGebra implementation of !FTL for illustrative and pedagogical purposes that visually demonstrates position, scale, and rotation invariance on 8 points.
- A JavaScript implementation of !FTL in an application software for assisting practitioners when organizing gestures into an appropriate gesture set. Both the GeoGebra and the JavaScript implementations are publicly available.
- An experiment comparing !FTL with respect to $1 [63] and $P [56], two state-of-the-art NNC recognizers of the $-family on a benchmarking gesture set. This experiment is not aimed at showing that !FTL is superior to $P or any other recognizer, but that the requirements are similarly satisfied.

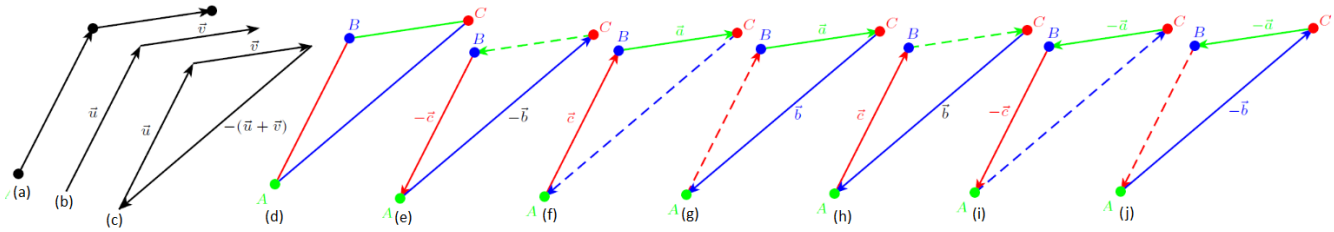


Figure 2: Vector-based representation: (a) a basic gesture, (b) the ordered couple of vectors for a basic gesture, (c) the corresponding triangle, (d) the three points of a triangle, (e-j) the six possible corresponding basic gestures.

2 RELATED WORK

To better characterize algorithm invariance, we define *isometricity* as the property of a gesture set to hold a set of n-equally distanced points: $\forall i \in \overline{n-1}, d(p_i, p_{i+1}) = constant$ e.g., $\|p_{i+1} - p_i\| = \frac{1}{n-1} \sum_{i=1}^{n} \|p_{i+1} - pi\|$. We define *isochronicity* as the property of a gesture set to hold a set of n equally-timestamped points, i.e., $\forall i \in \overline{n-1}, \|t_i - t_{i+1}\| = constant$, e.g., $\|t_{i+1} - t_i\| = \frac{1}{n-1} \sum_{i=1}^{n} \|(t_{i+1} - t_i)\|$. We also define *isopara-metrization* as the property of two or more gestures/sets to contain the same amount of points, i.e. $\forall G = \{p_i\}_{i=1,\ldots n}, H = \{q_j\}_{j=1,\ldots,m} : m{=}n$. Two gesture sets can be isoparameterized whether they are isometric or not, isochronic or not. Several distances have been investigated to compute the similarity between the candidate gesture and a reference one. We hereby summarize them with respect to the distance used.

The Euclidean distance computes the similarity between two gestures as a sum of Euclidean distances between pairs of corresponding points. Since this distance offers a straightforward geometric interpretation, it received a lot of attention and has been extensively researched and demonstrated in many recognizers: Rubine [45], $1 [63], $3 [33], $N [1], Enhanced $N [32], $P [57], $P+ [55], $V [65], Penny Pincher [51], 1F [57], 1 [26], *SHARK*^{2} [35], Rubine 2D and 3D combined in iGesture [47]. Most of them belong to the $-family of recognizers[1]. No assumption is made with respect to the nature of points, which may represent position or something else, provided that isometricity is preserved. The Euclidean distance can be used in both 2D [1, 2, 11, 35, 63] and 3D [33] setups. For example, *Flower Menus* [6] investigated NNC for a gesture-based command selection to reveal a recognition rate of 99% for the first 24 commands (i.e. straight and bent gestures), 96.5% for the first 40 commands (cusped gestures added), and of 93% for all the commands. Since all gestures were differentiated by the direction and shape, only position invariance was desired. A gesture shortcut could be attached to a menu item instead of a keyboard shortcut in order to increase its remembrance [4]. Since each gesture was attached to a pull-down menu of a desktop application, thus remaining located at the same place with the same orientation, only scale invariance was a barrier to recognition. Yamaji [65] introduced scale variance: a small right arrow will be recognized for movie forward and a large right arrow for fast forward. $P [57] was augmented into $V [65], a scale dependent recognizer, by adding calculation considering classes of scales for the same gesture, thus requiring more classifiers.

The Angular Cosine computes the similarity between two gestures by calculating the angle between the n-dimensional vectors represented by the points in the gesture. The distance has been shown to work well for both 2D and 3D gestures respectively in ProTractor [38], $N-ProTractor [2], and ProTractor3D [34].

Dynamic Time Warping (DTW) generalizes point-to-point computation of the Euclidean distance while minimizing cost alignment between two gestures [5, 51, 60, 63]. JackKnife [52] has been proved useful for multimodal recognition with a low sampling.

A String distance: computes the similarity between two gestures represented as strings of characters. The Simple Gesture Recognizer (SIGeR) [50] classified pen-based gestures on a MS Tablet PC by comparing a direction-based representation of a gesture candidate made up of four directions (L=left,R=right, U=up, D=down) to a regular expression such as (NE|E|SE) + (NW|N|NE) + (SW|W|NW) + (SE|S|SW), where letters indicate the four compass directions (i.e., E=East, W=West, N=North, and S=South). This classifier is therefore very sensitive to position, scale, and rotation. G-Gene [12] also relies on a directional representation of gesture to perform partial uni-stroke recognition at run-time.

The Levenshtein distance: computes the similarity between two strings representing a gesture based on directions by computing how many character insertions, deletions, and substitutions are required to transform the candidate gesture into the reference gesture. As such, it is a particular string distance. Coyette *et al.* [17] computed the Levenshtein distance (a distance measuring the character permutations and changes between two strings) between the two strings representing the candidate gesture and reference gestures. Each stroke is then used in CALI [21], a gesture recognizer based on primitives between shapes (e.g., a triangle included in a rectangle). This recognizer is position invariant, but is scale and rotation dependent. To recognize the same gesture in any direction, the gesture string should be transformed (e.g., a vertical symmetry replaces 1=North by 5=South, 2 = North-East by 4=South-East) for all configurations required. UsiGesture [7] relies on the Stochastic Levenshtein distance [8], which extends the previous one by considering a probabilistic model of the modifications, with only a marginal win observed in some specific cases.

The Hausdorff distance computes the similarity between two gestures by calculating the maximum of all the minimum Euclidean distances between each point of the candidate set to all points in the reference set [46]. Some derivatives of this distance have been also explored, such as the Modified Haussdorff and the Haussdorf-Besicovitch, a measure of the local size of a set of numbers [19, 31].

[1]See http://depts.washington.edu/madlab/proj/dollar/impact.html

Other distances [42] have been investigated with less systematic approach and success, such as the Mahalanobis distance (a measure used in computer vision), the Jaro-Winkler distance (a measure used in semantic web) [54]. But different drawbacks, such as sensitivity to outliers, variance to position, scale or rotation, divergence in extreme cases, lack of convergence in simplistic cases, unsatisfying performance have been observed. Already investigated distances and unexplored ones add more to the complexity and the confusion in choosing the right distance. In conclusion, NNC exhibits a high recognition rate with different geometric distances: Euclidean distance [57], Levenshtein distance [8, 17], angular cosine [48], DTW [16, 51], minimum-cost point cloud alignments [60]. Experiments have been conducted for 2D [56] and 3D [59, 64].

3 VECTOR-BASED GESTURE RECOGNITION

3.1 Why Vectors?

A gesture is by definition expressing a motion between an initial point and a final point with several characteristics such as position, scale, direction, curvature, pressure in case of a pressure-sensitive device, tangential acceleration, all of them can be subject to feature extraction and classification [13, 53]. A point-based representation of a gesture has the advantage of significantly simplifying a gesture to a series of points, thus reducing the gesture recognition to a comparison of two series of points. This approach works well particularly for static characters, like symbols, letters, simple commands, where the gesture shape is more important than the gesture motion. A point-based representation looses a lot of information which can be exploited or not depending on constraints imposed on the recognition, such as for position, scale, and rotation variations. For instance, a human signature is known to be easily recognized by a human forger imitating the gesture shape, but would be hardly recognized when motion is considered. A vector is a geometric object that intrinsically holds some motion expression such as a direction and a magnitude [31]. Vectors adequately represent the following quantities and properties that are particularly suitable [18, 29–31]:

- *Position in space*: position vectors define the positions of points by their displacement from any origin O.
- *Direction in space*: vectors indicate orientation of lines (such as strokes) and surfaces normals to planes. Only direction is important, magnitude is ignored.
- *Displacement, velocity, and force*: vectors specify in which direction, over what distance, and at what velocity a gesture can be issued or what force is acting on it. Those vectors are not concerned with the starting point of the position, and are called *free vectors*.

3.2 Basic Definitions

The continuous trace of two consecutive non-trivial translations of a point will be called *basic gesture* (Fig. 2a). A basic gesture in a finite dimensional affine space can then be formalized by a ordered couple $(\vec{u}, \vec{v})$ of two non-zero free vectors $\vec{u}$ and $\vec{v} \in \mathbb{R}^n$ (Fig. 2b).

A basic gesture $(\vec{u}, \vec{v})$ generates a well precise (eventually trivial) oriented triangle, whose third oriented side is the free vector $-(\vec{u} + \vec{v})$ (Fig. 2c). However, a well precise (possibly non trivial) triangle, having points A, B and C as distinct vertices (Fig. 2d) can be generated by six possibly different basic gestures. If we denote $\vec{a}$=C−B, $\vec{b}$=A−C, $\vec{c}$=B−A, then we have six basic gestures (Fig. 2e-j): $(\vec{c}, \vec{a})$, $(\vec{a}, \vec{b})$, $(\vec{b}, \vec{c})$, $(-\vec{a}, -\vec{c})$, $(-\vec{c}, -\vec{b})$, and $(-\vec{b}, -\vec{a})$, respectively. In particular, basic gestures in an affine plane correspond to ordered couples $(\vec{u}, \vec{v})$ of two non-zero free vectors $\vec{u}$=$(u_1, u_2) \in \mathbb{R}^2$ and $\vec{v}$=$(v_1, v_2) \in \mathbb{R}^2$. There is a one-to-one correspondence between vectors in $\mathbb{R}^2$ and complex numbers in $\mathbb{C}$; more precisely, to each free vector $\vec{x} = (x_1, x_2) \in \mathbb{R}^2$ corresponds the complex number $\mathbf{x} = x_1 + ix_2 \in \mathbb{C}$, and vice versa ($i$ is the imaginary unit such that $i^2 = -1$). The shape of an ordered triangle, traced by a basic gesture $(\vec{u}, \vec{v})$, can be encoded by the complex number [37] obtained as the quotient $\frac{\mathbf{u}}{\mathbf{v}} \in \mathbb{C}$ of the complex numbers $\mathbf{u}, \mathbf{v} \in \mathbb{C}$ corresponding to vectors $\vec{u}, \vec{v} \in \mathbb{R}$ of the basic gesture $(\vec{u}, \vec{v})$. Every oriented triangle is characterized up to similarity by a single complex number, called *shape* [37]. Analogously, we define the quotient $\frac{\mathbf{u}}{\mathbf{v}} \in \mathbb{C}$ as the *similarity ratio of the basic gesture* $(\vec{u}, \vec{v})$. Let us recall the properties and correspondences between the metrics in $\mathbb{R}^2$ and $\mathbb{C}$: the norm of a free vector $\vec{x} \in \mathbb{R}^2$ and the modulus of its corresponding complex number $\mathbf{x} \in \mathbb{C}$ coincide:

$$|\vec{x}| = \sqrt{\vec{x} \cdot \vec{x}} = \sqrt{(x_1)^2 + (x_2)^2} = \sqrt{\mathbf{x}\,\mathbf{x}^*} = |\mathbf{x}|_{\mathbb{C}} \; where$$

- $\vec{x} \cdot \vec{y} = (x_1 y_1) + (x_2 y_2)$ is the scalar product between $\vec{x} = (x_1, x_2) \in \mathbb{R}^2$ and $\vec{y} = (y_1, y_2) \in \mathbb{R}^2$,
- $\mathbf{x}^*$=$x_1 - ix_2 \in \mathbb{C}$ is the complex conjugate of $\mathbf{x}$=$x_1 + ix_2 \in \mathbb{C}$.

Thus, also the distance between two vectors $\vec{x}, \vec{y} \in \mathbb{R}^2$ and the distance between the two corresponding complex numbers $\mathbf{x}, \mathbf{y} \in \mathbb{C}$ coincide: $|\vec{x} - \vec{y}| = |x - y|_{\mathbb{C}}$, that is, $\mathbb{R}^2$ and $\mathbb{C}$ are isometric.

3.3 Local Shape Distance between Gestures

Inspired by the Global Shape Distance (GSD) [11, 53], the dissimilarity (not the similarity!) between two basic gestures $(\vec{a}, \vec{b})$ and $(\vec{u}, \vec{v})$, defined as the *Local Shape Distance*, denoted by the symbol $LSD\big((\vec{a}, \vec{b}), (\vec{u}, \vec{v})\big)$, is defined as the Euclidean distance between the similarity ratios of the basic gestures $(\vec{a}, \vec{b})$ and $(\vec{u}, \vec{v})$. Equations and their demonstrations can be found in Appendix B.

3.4 Invariance of the Local Shape Distance

In order to demonstrate the invariance properties of LSD, let us use function *LSD* to compare two basic gestures $(\vec{a}, \vec{b})$ and $(\vec{u}, \vec{v})$ laying on a same affine plane. LSD is said to be articulation-invariant because it satisfies the following properties [28].

Point-number and stroke-number invariance. LSD is computed on a set of vectors that are either continuous (a basic gesture) or a series of continuous ones (a series of basic gestures). If no stroke exists between the ending point of a gesture stroke and the starting point of the next stroke, no vector is created and the LSD remains unaffected. Therefore, both single-stroke and multi-stroke gestures are supported. Although the sampling limit can be pushed to 8 points [57], a 32 point-sampling represented a viable compromise between recognition rate and execution time.

Stroke-order invariance. LSD is computed on vectors created from one or many series of points, provided that isoparameterization is ensured. How the points are considered and thus how vectors are defined for computing the LSD does not affect LSD. Thus, stroke-order invariance is supported. For instance, the house in Fig. 5 is sketched as a multi-stroke gesture with different orders of strokes: the house parts could be drawn in any order.

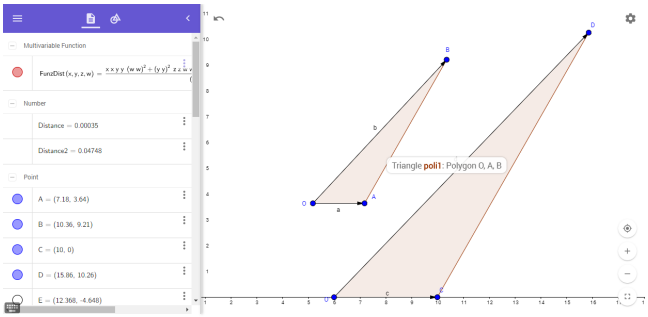


Figure 3: GeoGebra implementation: scale invariance.

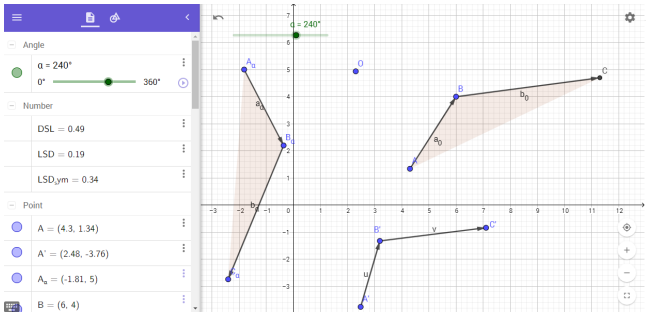


Figure 4: GeoGebra implementation: rotation invariance.

Stroke-direction invariance. LSD is not symmetric: it can happen that $LSD(\vec{a},\vec{b},\vec{c},\vec{d}) \neq LSD(\vec{b},\vec{a},\vec{d},\vec{c})$. An oriented gesture can be transformed into its unoriented version and redirected into a reference vector in one direction. Instead, a simpler and more efficient approach is preferred: since only the denominator differs for both $LSD(\vec{a},\vec{b},\vec{c},\vec{b})$ and $LSD(\vec{b},\vec{a},\vec{d},\vec{c})$, we compute the symmetrized version which is independent of the local direction (See Equation (2) in appendix B). One can also observe that both LSD and LSD_{sym} depends on the lengths of vectors $\vec{a}$, $\vec{b}$, $\vec{c}$, and $\vec{d}$. Such sensitivity to the length of each basic gesture is valuable when the points of the corresponding ordered triangle faithfully represents the isochrone sampling of a real gesture. The sampling offered by a real-world device is rarely isochrone. Equally spaced successive sampled points, i.e. isometric points, do not correspond to equally time-spaced points of the gesture, i.e. to isochrone points. In order to reduce this potential bias, we define a LSD independent on the lengths of the vectors in the basic gestures: $\forall(\vec{a},\vec{b}),(\vec{c},\vec{d})$, the Normalized Local Shape Distance (NLSD) is defined by the Equation (3) in Appendix B. $NLSD$ is symmetric: $NLSD\big((\vec{a},\vec{b}),(\vec{c},\vec{d})\big) = NLSD\big((\vec{b},\vec{a}),(\vec{d},\vec{c})\big)$.

Position Invariance. This is ensured by proving that a translation preserves LSD. The translation performed by a free vector $\vec{t}$ of a basic gesture $(\vec{a},\vec{b})$ (corresponding to a oriented triangle of vertices A, B and C) produces a basic gesture $(\vec{a}',\vec{b}')$ (corresponding to a oriented triangle of vertices A'=$A+\vec{t}$, B'=$B+\vec{t}$ and C'=$C+\vec{t}$). Any translation $\vec{t}$ does not affect the basic gesture, as we have that $\vec{a}'$=$\vec{a}$ and $\vec{b}'$=$\vec{b}$.

Scale Invariance. This is ensured by proving that a homogeneous dilation preserves LSD (See Equation (4) in Appendix A.2).

Rotation Invariance. This is ensured by proving that a rotation preserves LSD. One can note that:

- Every counter-clockwise rotation $\mathcal{R}_\alpha$ in the Euclidean plane $\mathbb{R}^2$ of a radian angle $\alpha\in\mathbb{R}$ corresponds to a counter-clockwise rotation R_α in the isometric complex plane $\mathbb{C}$ of equal angle.
- Recalling the Euler's formulas $e^{i\alpha} = \cos\alpha + i\sin\alpha$, $e^{i(\alpha+\beta)} = e^{i\alpha}e^{i\beta}$, the complex number $R_\alpha(\mathbf{x})$ can be obtained throughout the complex multiplication $R_\alpha(\mathbf{x}) = e^{i\alpha}\mathbf{x}$.
- The ratio of two complex numbers, both rotated by a same angle, is equal to their original ratio (See the Equations (5) and (6) in Appendix B).

One can verify that the general LSD defined in (1) has the same invariance properties in any finite dimensional non-degenerate quadratic space.

4 ALGORITHMS AND IMPLEMENTATIONS

To demonstrate the applicability of (N)LSD for gesture recognition, equation (1) was instantiated to n=2 and implemented into !FTL, a 2D gesture recognizer based on (N)LSD.

GeoGebra Implementation. GeoGebra is a Dynamic Mathematics Software (DMS) for teaching and learning mathematics by bridging some gaps between geometry, algebra and calculus. LSD, NLSD and !FTL were implemented in a GeoGebra geometry application [2] to visually demonstrate the position, scale, and rotation invariance. Fig. 3 reproduces a screen shot where a first gesture is recorded as a reference gesture and a second gesture is acquired as a candidate gesture. The LSD is automatically computed between these two basic gestures: when any edge of the candidate gesture is moved, the corresponding point location is updated as well as its corresponding vectors, and so does the distance value. When the candidature gesture is translated, dilated (Fig. 3) or rotated (Fig. 4) by direct manipulation, the distance remains unaltered. The rotation could be animated step by step, by a scale at any desired angle. The sampling is fixed to 8 points, as recommended in [56].

JavaScript Implementation. The appropriate management of gestures into one or several training sets that can be effectively used later on in corresponding interactive systems is a process that is assisted by a dedicated software developed for helping practitioners in their responsibility to produce the entire material required for a gesture-based user interface, such as a gesture library [22]. This follows the tradition of computer-aided design tools for gesture recognition initiated and continued by representative examples such as *gdt* [39], GART [41], Magic [5], GestureBar [10], UsiGesture [8], GDATK [59], GestureSplit [40], Gesture Recognition Toolkit [23], RATA.gesture [13], Gestimator [66] and most recently by JackKnife [52]. The system has been developed as a cross-device responsive application which can be accessed from any device.

Fig. 5 reproduces a screen shot of the on-line environment for managing gesture samples and gesture sets with four recognizers: !FTL with LSD, FTL with NLSD, $1 [1] and $P [56]. Any gesture set can be created, edited, deleted, and loaded for testing. In this example, a house captured as a multi-stroke gesture is recognized with a full success (total dissimilarity: LSD=0 and $NLSD$ extremely low) both with an execution time t<1 msec. Note that $1 and $P recognize the same gesture with a distance d=0.99, respectively d=1, and an execution time of t=0.89 msec, resp. t=3.73 msec.

[2] www.geogebra.org/geometry

Function	LOC
LSD calculation	12
NLSD calculation	12
Interpolation	14
Scalar product	1
Gesture recognition	93
Total with all functions without comments	132

Table 1: Lines of code for !FTL.

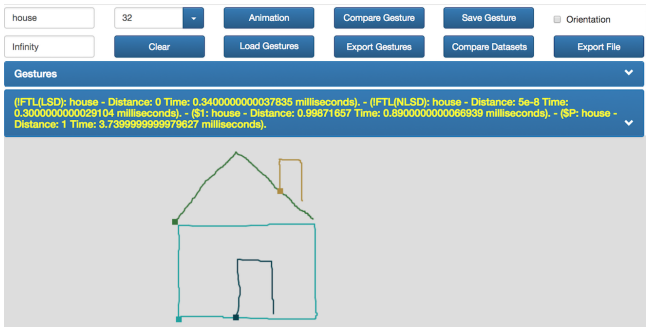


Figure 5: !FTL JavaScript in its environment.

Appendix A provides the !FTL pseudo-code used for implementing the JavaScript and the GeoGebra versions. Table 1 shows how the JavaScript implementation is distributed in terms of lines of code (LOC): 132 LOC basically implements !FTL in a grand total of 219 LOC for the whole recognizer including comments and blank lines. This is comparable with $1, $N and $P which respectively require 100 LOC, 200 LOC, and 70 LOC [57]. Regarding the order of the computational complexity, the simplified form of $P needs $O(n^{2.5})$ time to recognize a gesture where n is the number of sampled points, thus representing a polynomial complexity. In regard, !FTL only needs $O(n)$ since it computes LSD by summing up consecutive basic gestures, where each basic gesture involves three points. !FTL is linear with respect to both the amount of basic gestures and the amount of points, the same complexity as for $1.

5 EXPERIMENT

By following the methodology from the literature [1, 2, 57, 62, 63], we conducted an experiment to show that !FTL performance is aligned with $1 [1] and $P [57] for multi-stroke, two state-of-the-art recognizers belonging to the $-family. At the submission time of this paper, $P was the last member of this family and the most flexible and efficient. $Q [58], an optimized version of $P for low-end devices, appeared in September 2018 and was therefore not included in the comparison. PennyPincher [51] was not included for two reasons: "Penny Pincher is scale invariant, though unlike other recognizers it is not rotation invariant" ([51], p. 201, c1); due to the normalization of vectors to 1, two "L"-shaped vectors, one with a short arm and one with a long arm, cannot be distinguished.

5.1 Apparatus

We employed NicIcon [62], a publicly-available large gesture set consisting of 14,005 uni- and multi-stroke gestures produced by 35 participants for 14 symbol types: accident, bomb, car, casualty, electricity, fire, fire brigade, flood, gas, injury, paramedics, person, police, and roadblock. This gesture set was considered more challenging than other sets because of its variety and complexity and because it contains dynamic gestures, not just static shapes like in HHReco [28] (e.g., arch, hexagon, heart, moon, pentagon, ellipse, square). We employed an Apple MacBook 13" running a Intel Core i5 2.9 GHz processor and running the macOS Sierra V10.12.6 operating system. The RAM was 8 Go DDR3 memory with 1867 MHz. Google Chrome V60.0.3112.101 was used in its 64 bits version.

5.2 Design and Measures

When downloading NicIcon, we noticed that participants #26 and #32 were absent from the announced 35 participants and that sometimes up to 51 gestures were available for a single symbol, although an average of 30 was published. Thus, to preserve fairness, we counted on the 33 participants with complete, aligned data. Beyond these variations in number of samples per class, for instance up to 51 samples for a single class, we decided to keep all gestures on a sampling of 32 points [57]. Our user-dependent scenario was therefore with a design setup as follows: 33 participants × 14 symbols × 30 gesture samples per class (minimum) × 4 recognizers (!FTL+LSD, !FTL+NLSD, $1, and $P) = 55,440 samples. The total sampling is certainly above since some symbols were recorded with [30, ..., 51] samples, which were all considered. For each gesture class, one sample was randomly selected for testing while keeping the 29 other samples for training. This process was repeated for all samples within each class, execution times and recognition rates were computed individually for each participant. Results were averaged into an execution time and a recognition rate per participant, thus resulting into at least 55,440 samples × 30 permutations = 1,663,200 elementary tests. The hypotheses formulated for this experiment were the following:

H_{11}=LSD and NLSD will be faster than $1 and $P for NicIcon. Their execution times, measured in msec., will be smaller than their counterparts for $1 and $P.

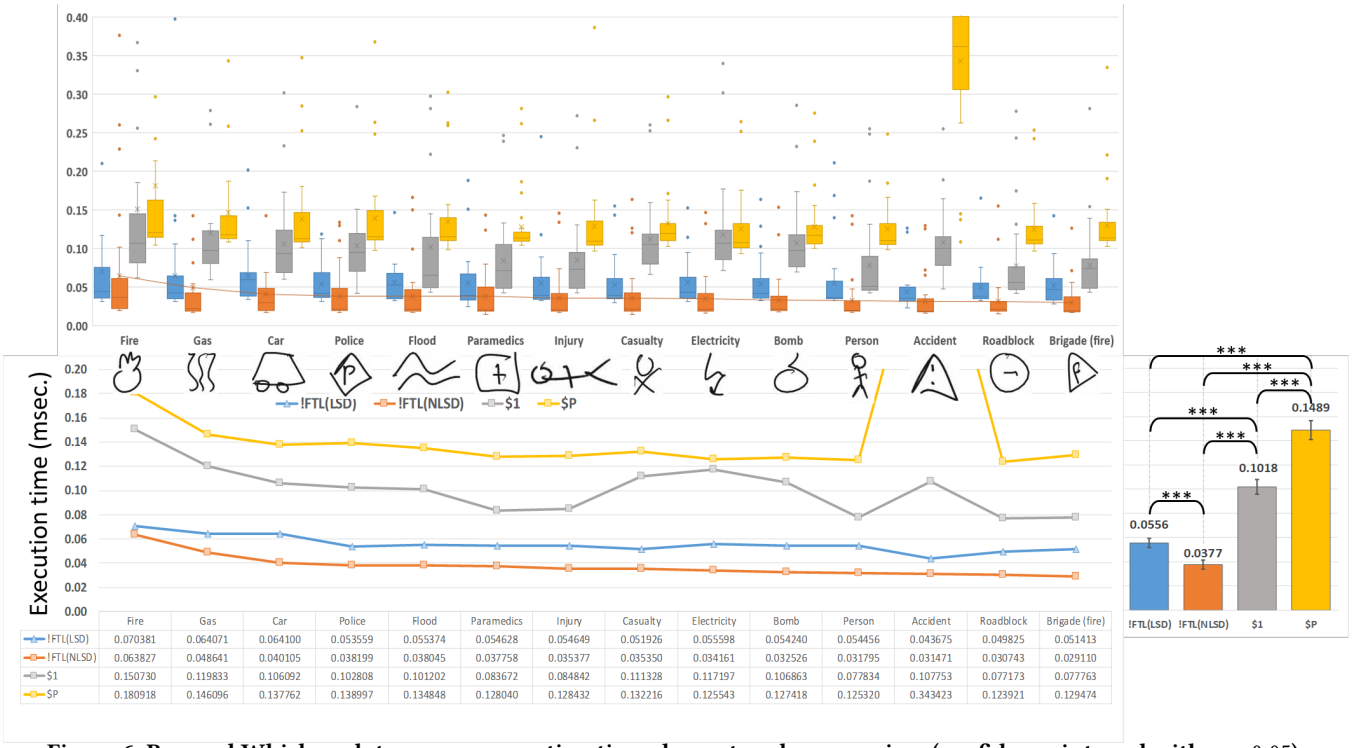
H_{21}=LSD and NLSD will be more accurate than $1 and $P for NicIcon. Their recognition rates, measured in percentage, will be higher than their counterparts for $1 and $P.

H_{31}=NLSD will be faster than LSD for NicIcon.

H_{41}=NLSD will be more accurate than LSD for NicIcon.

The JS implementation was used on the above platform to run the testing, accepting the NicIcon XML files and producing CSV log files structured as follows and imported into MS Excel:

```
Date: accident_4/09/2017 @ 10:45:47 - ALL
Records: 33 (NicIcon)
File name: accident.xml, -, -, -
Iteration: undefined Gesture evaluated=accident
Number of Points:32, Threshold: Infinity, -
Recognizers: LSD, NLSD, $1, $P
Amount of Samples: 30, 30, 30, 30, -, -, -
Distance: 75.89, 7.05, 0, 0.09, -, -, -
Execution time: 1, 0.25, 0, 5, -, -, -
Recognition rate: 10, 26.66, 0, 3.33, -, -, -
```



	Fire	Gas	Car	Police	Flood	Paramedics	Injury	Casualty	Electricity	Bomb	Person	Accident	Roadblock	Brigade (fire)
!FTL(LSD)	0.070381	0.064071	0.064100	0.053559	0.055374	0.054628	0.054649	0.051926	0.055598	0.054240	0.054456	0.043675	0.049825	0.051413
!FTL(NLSD)	0.063827	0.048641	0.040105	0.038199	0.038045	0.037758	0.035377	0.035350	0.034161	0.032526	0.031795	0.031471	0.030743	0.029110
$1	0.150730	0.119833	0.106092	0.102808	0.101202	0.083672	0.084842	0.111328	0.117197	0.106863	0.077834	0.107753	0.077173	0.077763
$P	0.180918	0.146096	0.137762	0.138997	0.134848	0.128040	0.128432	0.132216	0.125543	0.127418	0.125320	0.343423	0.123921	0.129474

Figure 6: Box and Whisker plot, average execution times by gesture by recognizer (confidence interval with $\alpha = 0.05$).

5.3 Results and Discussion

5.3.1 Execution Time. Fig. 6 presents the Box and Whisker plot and the average execution times delivered by the four recognizers for all NicIcon gesture classes in decreasing order of the NLSD execution time, which has the lowest curve, followed by LSD, $1, and $P respectively. A Student's t-Test (independent two-sample) was computed between all six possible pairs of recognizers to compare execution times for (N)LSD and $-conditions.

First of all, there was a very highly significant difference in the execution times for !FTL(LSD) (M=.0556, SD=.0388, Median=.0406) and !FTL(NLSD) (M=.0377, SD=.0421, Median=.0210) conditions; df= 458, t= 9.09, p<.001***, Pearson's ρ=.45. These results suggest that using NLSD instead of LSD does have a positive effect on execution time, which supports H_{31}. Next, there was a very highly significant difference in the execution times for !FTL(LSD) and $1 ($M$=.1017, SD=.0675, Median=.0873) conditions; df=458, t= − 15.56, p<.001***, Pearson's ρ=.39. There was also a significant difference in the execution times for $1 and $P ($M$=.1489, SD=.0848, Median=.1147); df=458, t= − 23.93, p < .001***, Pearson's ρ=0.26. These results suggest that LSD is faster than $1, which is in turn faster than $P for this dataset. Then, starting again from !FTL(NLSD), it is highly significantly faster than $P ($df$=458, t=−27.51, p < .001***, Pearson's ρ=0.20) and $1 ($df$=458, t= − 21.93, p < .001***, Pearson's ρ=0.42). Apart from some cases such as the "Fire" and "Police" classes, the main quartile of LSD is never really overlapping with the NLSD one, and there is little or no overlapping between the intervals of the (N)LSD family and those of the $-family. (N)LSD is faster than $1 and $P, which supports H_{11}.

A one-way (single factor on recognizer) ANOVA also gave for the execution time: p<0.05 with Scheffe = 0.1400 > Tukey's HSD = 0.0105 > Fisher's LSD = 0.0079. For the post-hoc tests, all cells had significant mean differences, e.g. !FTL vs $P: 0.0933, $P vs $1: 0.0472. To estimate the importance of the execution time, we also computed Cohen's d index [15], which defines the effect size as the extent to which the phenomenon is found within the population or, in the context of statistical significance testing, the degree to which the null hypothesis is false. The following values have been obtained for all pairs: $d(LSD, NLSD)$=.44, $d(LSD, \$1)$=.84, $d(LSD, \$P)$ = 1.41, $d(NLSD, \$1)$=1.14, $d(NLSD, \$P)$=1.66, and $d(\$1, \$P)$=.62. According to the general guidelines for interpreting the effect size introduced by Cohen himself, i.e. small (0.2), medium (0.5), and large (0.8), the effect between LSD and NLSD has a small size, but all other comparisons revealed a large size, apart between $1 and $P which is medium. Let us compare the most efficient members of their respective families, i.e. NLSD and $P. With a Cohen's d of 1.66, 95% of the NLSD condition will be above the mean of the $P condition (Cohen's U_3), 42% of the two groups will overlap, and there is a 87% chance that a randomly selected NLSD participant will have a higher score than a $P one (probability of superiority).

Cohen's d assumes [15] that the two samplings share the same equal size and similar variances. Variances are close but not similar. Thus, we computed Glass's Δ index [24] suited for heteroscedastic samplings (with unequal variances) to confirm the initial impression: $\Delta(LSD, NLSD)$=.46, $\Delta(LSD, \$1)$ = 1.19, $\Delta(LSD, \$P)$=2.40, $\Delta(NLSD, \$1)$=1.52, $\Delta(NLSD, \$P)$ = 2.64, and also $\Delta(\$1, \$P)$ = .70. These more strict values confirm the previously established Cohen's ones. Both H_{11} and H_{31} are supported.

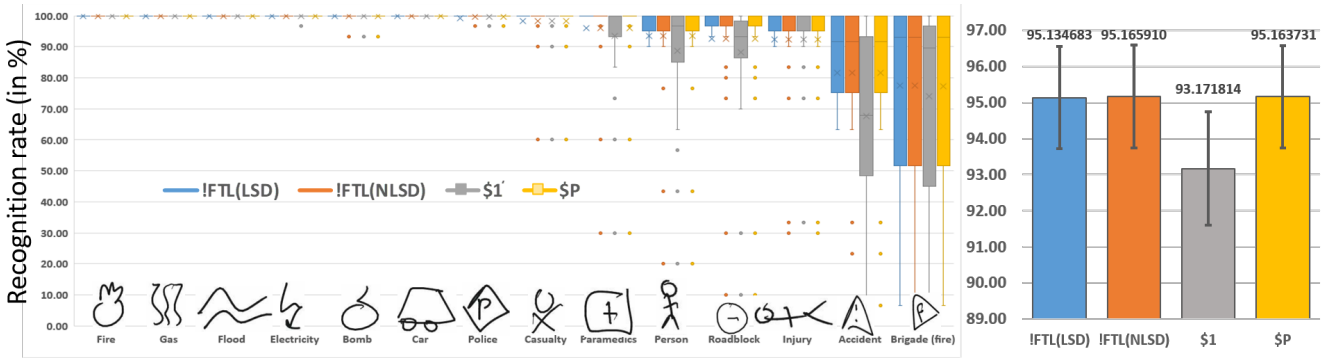


Figure 7: Box and Whisker plot, average recognition rate by gesture by recognizer (confidence interval with $\alpha = 0.05$).

5.3.2 Recognition rate. Fig. 7 presents the Box and Whisker plot and the recognition rates delivered by the four recognizers for all NicIcon gesture classes in decreasing order of the NLSD condition. The !FTL(NLSD) recognizer delivered the best average recognition rate (95.1659%) for the user-dependent scenario followed by !FTL(LSD) with 95.1346%. $P and $1 came next with 95.1637% and 93.1718%, respectively. A Student's t-Test (independent two-sample) was computed between all six possible pairs of recognizers to compare recognition rates for (N)LSD and $-conditions.

There was no significant difference in the recognition rates for !FTL(LSD) (M=95.1346%, SD=15.5366%, Median=100%) and $P ($M$ = 95.1637%, SD=15.5332%, Median=100%) conditions; t= − 1.4, p=0.08, Pearson's ρ=0.99. These results suggest that using LSD instead of $P does does not influence recognition rate. Similarly, there was no significant difference in the recognition rates for !FTL(NLSD) (M=95.1659%, SD=15.5338%, Median=100%) and !FTL(LSD); t= − 1.53, p=0.06, Pearson's ρ=0.97. !FTL(NLSD) is neither more accurate than $P ($t$=1, p=0.15, Pearson's ρ=0.99) nor than $1 ($M$=93.1718%, SD=17.1065%, Median=100%); t=7.00, p=0.08, Pearson's ρ=0.93). Confidence intervals of 95% (α=.05) of respective recognizers largely overlap on each other, but largely with $1, but not totally.

Fig. 7 shows that the four recognizers share almost the same values for the average recognition rate (of the order of 95% with a similar standard deviation of 15%), apart from $1, which is slightly inferior to the other with an average of the order of 93% and a standard deviation of 17%. These results suggest that the global behaviors of LSD, NLSD, and $P are surprisingly similar in terms of recognition: not only they share the same average and standard deviation, but also they see their maximum rate on the same classes (e.g., from "Fire" to "Casualty") and comparable variations on more challenging classes (e.g., "Accident" and "Fire Brigade"). These last gestures are more challenging because of their similarity, but also because the dataset itself contains samples which are not very close to the original gesture. So, when a gesture class contains samples which do not reflect very well the original gestures, it is challenging in the same way for all recognizers and they all behave the same with respect to this complexity.

In conclusion, H_{21} and H_{41} are not supported: (N)LSD are not more accurate than their $-family counterparts, but they are aligned with their performance on the NicIcon dataset.

6 CONCLUSION

This paper presented a novel Local Shape Distance (LSD and NLSD) that computes the dissimilarity between gestures represented as n-dimensional vectors with several properties: point-number and stroke-number invariance, stroke-order and stroke-direction invariance, position, scale, and rotation invariance. An instantiation of LSD (and NLSD) to n=2 gave rise to !FTL, a 2D vector-based gesture recognizer with its pseudo-code and two implementations. This work will benefit practitioners by providing a new gesture recognizer satisfying the requirements in a comparable way (e.g., high recognition rate, small execution time, low resource consumption, and low algorithm complexity with geometric interpretation) while preserving the invariance properties. If for any reason a gesture should become for instance scale variant, the constraint can be imposed on the vectors without any loss of generality and without complexifying the recognizer with additional computations.

!FTL can be subject to several optimizations, such as those of $P [56] and $Q [58]. A stopping criteria can be defined to recognize a gesture as soon as a threshold of vectors are computed similar, thus eliminating the need to consider all vectors. In this way, a gesture can be recognized while being issued, even before it ends, thus improving its execution time. This enables *gesture feedforward* [61] (instead of immediate feedback after recognition, feedforward provides guidance while the gesture is being issued) and *gesture mnemonics* [3] where end users expect an immediate response time ($\simeq$ 1sec) to avoid the lagging effect [9]. A second optimization concerns the combination of LSD and !FTP with DTW [52], which can be effectively and efficiently combined since DTW optimizes the point sampling before processing. This represents a very promising area to investigate. Finally, we will consider Conditional Random Fields (CRF) [49] for inverting the recognition process: as opposed to NNC where a near neighbor should emerge from the classification of objects, CRF proposes a discrete classifier that predicts the near neighbor for a single sample without considering all samples.

ACKNOWLEDGMENTS

The two first authors would like to thank Laetitia Vanderdonckt and Maria Roselli for becoming friends and initiating the meeting of their fathers, which gave rise to this fortuitous collaboration. The authors would also like to warmly thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

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

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

[3] Caroline Appert and Olivier Bau. 2010. Scale Detection for a Priori Gesture Recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 879–882. https://doi.org/10.1145/1753326.1753456

[4] Caroline Appert and Shumin Zhai. 2009. Using Strokes As Command Shortcuts: Cognitive Benefits and Toolkit Support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 2289–2298. https://doi.org/10.1145/1518701.1519052

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

[6] 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 (AVI '08)*. ACM, New York, NY, USA, 15–22. https://doi.org/10.1145/1385569.1385575

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

[8] François Beuvens and Jean Vanderdonckt. 2012. UsiGesture: An environment for integrating pen-based interaction in user interface development. In *Sixth International Conference on Research Challenges in Information Science, RCIS 2012, Valencia, Spain, May 16-18 2012*, Colette Rolland, Jaelson Castro, and Oscar Pastor (Eds.). IEEE, 1–12. https://doi.org/10.1109/RCIS.2012.6240449

[9] Ugo Braga Sangiorgi, Vivian Genaro Motti, François Beuvens, and Jean Vanderdonckt. 2012. Assessing Lag Perception in Electronic Sketching. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design (NordiCHI '12)*. ACM, New York, NY, USA, 153–161. https://doi.org/10.1145/2399016.2399040

[10] Andrew Bragdon, Robert Zeleznik, Brian Williamson, Timothy Miller, and Joseph J. LaViola, Jr. 2009. GestureBar: Improving the Approachability of Gesture-based Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 2269–2278. https://doi.org/10.1145/1518701.1519050

[11] 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 (CHI '07)*. ACM, New York, NY, USA, 1495–1504. https://doi.org/10.1145/1240624.1240850

[12] 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

[13] Samuel hsiao-heng Chang, Rachel Blagojevic, and Beryl Plimmer. 2012. Rata.Gesture: A Gesture Recognizer Developed Using Data Mining. *Artif. Intell. Eng. Des. Anal. Manuf.* 26, 3 (Aug. 2012), 351–366. https://doi.org/10.1017/S0890060412000194

[14] 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 (ITS '14)*. ACM, New York, NY, USA, 35–44. https://doi.org/10.1145/2669485.2669509

[15] J. Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences, 2nd Edition*. Lawrence Erlbaum, Hillsdale.

[16] Paul Corey and Tracy Hammond. 2008. GLADDER: Combining Gesture and Geometric Sketch Recognition. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3 (AAAI'08)*. AAAI Press, 1788–1789. http://dl.acm.org/citation.cfm?id=1620270.1620354

[17] Adrien Coyette, Sascha Schimke, Jean Vanderdonckt, and Claus Vielhauer. 2007. *Trainable Sketch Recognizer for Graphical User Interface Design*. Springer Berlin Heidelberg, Berlin, Heidelberg, 124–135. https://doi.org/10.1007/978-3-540-74796-3_14

[18] Leo Dorst, Daniel Fontijne, and Stephen Mann. 2007. *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[19] M.-P. Dubuisson and A. Jain. 1994. A Modified Hausdorff distance for object matching. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition (IAPR '94)*. IEEE Press, 566–568. http://ieeexplore.ieee.org/document/576361/

[20] Richard O. Duda, Peter E. Hart, and David G. Stork. 2000. *Pattern Classification*. Wiley & Sons, New York.

[21] Manuel J. Fonseca and Joaquim A. Jorge. 2001. Experimental evaluation of an on-line scribble recognizer. *Pattern Recognition Letters* 22, 12 (2001), 1311–1319. https://doi.org/10.1016/S0167-8655(01)00076-9

[22] Bruno Galveia, Tiago Cardoso, Vitor Santor, and Yves Rybarczyk. 2015. Towards the creation of a Gesture Library. *EAI Endorsed Transactions on Creative Technologies* 15, 3 (6 2015). https://doi.org/10.4108/ct.2.3.e3

[23] Nicholas Gillian and Joseph A. Paradiso. 2014. The Gesture Recognition Toolkit. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 3483–3487. http://dl.acm.org/citation.cfm?id=2627435.2697076

[24] G.V. Glass, B. McGaw, and M.L. Smith. 1981. *Meta-Analysis in Social Research*. Sage, Beverly Hills.

[25] Trevor Hastie, Robert Tibshirani, and Jérome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, Berlin.

[26] J. Herold and T. F. Stahovich. 2012. The 1&Cent; Recognizer: A Fast, Accurate, and Easy-to-implement Handwritten Gesture Recognition Technique. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling (SBIM '12)*. Eurographics Association, Goslar Germany, Germany, 39–46. http://dl.acm.org/citation.cfm?id=2331067.2331074

[27] Michael Hoffman, Paul Varcholik, and Joseph J. LaViola. 2010. Breaking the Status Quo: Improving 3D Gesture Recognition with Spatially Convenient Input Devices. In *Proceedings of the 2010 IEEE Virtual Reality Conference (VR '10)*. IEEE Computer Society, Washington, DC, USA, 59–66. https://doi.org/10.1109/VR.2010.5444813

[28] Heloise Hse, Michael Shilman, and A. Richard Newton. 2004. Robust Sketched Symbol Fragmentation Using Templates. In *Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI '04)*. ACM, New York, NY, USA, 156–160. https://doi.org/10.1145/964442.964472 Retrieved September 9, 2017 from https://embedded.eecs.berkeley.edu/research/hhreco/.

[29] M. Murray J. Gilbert. 1991. *Clifford algebras and Dirac operators in harmonic analysis*. Cambridge University Press.

[30] Kenichi Kanatani. 2015. *Understanding Geometric Algebra: Hamilton, Grassmann, and Clifford for Computer Vision and Graphics*. A. K. Peters, Ltd., Natick, MA, USA.

[31] Levent Burak Kara and Thomas F. Stahovich. 2004. Hierarchical Parsing and Recognition of Hand-sketched Diagrams. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST '04)*. ACM, New York, NY, USA, 13–22. https://doi.org/10.1145/1029632.1029636

[32] Kisang Kim and Hyung-Il Choi. 2016. Online Hand Gesture Recognition Using Enhanced $N Recogniser Based on a Depth Camera. *Int. J. Comput. Vision Robot.* 6, 3 (Jan. 2016), 214–222. https://doi.org/10.1504/IJCVR.2016.077352

[33] 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 (IUI '10)*. ACM, New York, NY, USA, 341–344. https://doi.org/10.1145/1719970.1720026

[34] Sven Kratz and Michael Rohs. 2011. Protractor3D: A Closed-form Solution to Rotation-invariant 3D Gestures. In *Proceedings of the 16th International Conference on Intelligent User Interfaces (IUI '11)*. ACM, New York, NY, USA, 371–374. https://doi.org/10.1145/1943403.1943468

[35] 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 (UIST '04)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/1029632.1029640

[36] Luis A. Leiva, Daniel Martín-Albo, and Réjean Plamondon. 2015. Gestures À Go Go: Authoring Synthetic Human-Like Stroke Gestures Using the Kinematic Theory of Rapid Movements. *ACM Trans. Intell. Syst. Technol.* 7, 2, Article 15 (Nov. 2015), 29 pages. https://doi.org/10.1145/2799648

[37] J. A. Lester. 1996. Triangles I: Shapes. *aequationes mathematicae* 52, 1 (01 Feb 1996), 30–54. https://doi.org/10.1007/BF01818325

[38] Yang Li. 2010. Protractor: A Fast and Accurate Gesture Recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 2169–2172. https://doi.org/10.1145/1753326.1753654

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

[40] Hao Lü, James A. Fogarty, and Yang Li. 2014. Gesture Script: Recognizing Gestures and Their Structure Using Rendering Scripts and Interactively Trained Parts. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1685–1694. https://doi.org/10.1145/2556288.2557263

[41] Kent Lyons, Helene Brashear, Tracy Westeyn, Jung Soo Kim, and Thad Starner. 2007. GART: The Gesture and Activity Recognition Toolkit. In *Proceedings of the 12th International Conference on Human-computer Interaction: Intelligent Multimodal Interaction Environments (HCI'07)*. Springer-Verlag, Berlin, Heidelberg, 718–727. http://dl.acm.org/citation.cfm?id=1769590.1769671

[42] Elena Deza Michel Marie Deza. 2016. *Encyclopedia of Distances* (4 ed.). Springer-Verlag, Berlin.

[43] Tao Ni and Patrick Baudisch. 2009. Disappearing Mobile Devices. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 101–110. https://doi.org/10.1145/1622176.1622197

[44] Corey Pittman, Eugene M. Taranta II, and Joseph J. LaViola, Jr. 2016. A $-Family Friendly Approach to Prototype Selection. In *Proceedings of the 21st International Conference on Intelligent User Interfaces (IUI '16)*. ACM, New York, NY, USA, 370–374. https://doi.org/10.1145/2856767.2856808

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

[46] W. J. Rucklidge. 1995. Locating Objects Using the Hausdorff Distance. In *Proceedings of the Fifth International Conference on Computer Vision (ICCV '95)*. IEEE Computer Society, Washington, DC, USA, 457–. http://dl.acm.org/citation.cfm?id=839277.840123

[47] B. Signer, U. Kurmann, and M. Norrie. 2007. iGesture: A General Gesture Recognition Framework. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02 (ICDAR '07)*. IEEE Computer Society, Washington, DC, USA, 954–958. http://dl.acm.org/citation.cfm?id=1304596.1304930

[48] David Bingham Skalak. 1997. *Prototype Selection for Composite Nearest Neighbor Classifiers*. Ph.D. Dissertation. Amherst, MA, USA. UMI Order No. GAX97-37585.

[49] Charles A. Sutton and Andrew McCallum. 2012. An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning* 4, 4 (2012), 267–373. https://doi.org/10.1561/2200000013

[50] Scott Swigart. 2005. *Easily Write Custom Gesture Recognizers for Your Tablet PC Applications*. Retrieved September 9, 2017 from https://msdn.microsoft.com/en-us/library/aa480673.aspx.

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

[52] 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 *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 5850–5861. https://doi.org/10.1145/3025453.3026002

[53] Huawei Tu, Xiangshi Ren, and Shumin Zhai. 2015. Differences and Similarities Between Finger and Pen Stroke Gestures on Stationary and Mobile Devices. *ACM Trans. Comput.-Hum. Interact.* 22, 5, Article 22 (Aug. 2015), 39 pages. https://doi.org/10.1145/2797138

[54] Jean Vanderdonckt, Bruno Dumas, and Mauro Cherubini. 2018. Comparing Some Distances in Template-based 2D Gesture Recognition. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems (CHI EA '18)*. ACM, New York, NY, USA, Article LBW121, 6 pages. https://doi.org/10.1145/3170427.3188452

[55] Radu-Daniel Vatavu. 2017. Improving Gesture Recognition Accuracy on Touch Screens for Users with Low Vision. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017.*, Gloria Mark, Susan R. Fussell, Cliff Lampe, m. c. schraefel, Juan Pablo Hourcade, Caroline Appert, and Daniel Wigdor (Eds.). ACM, 4667–4679. https://doi.org/10.1145/3025453.3025941

[56] Radu-Daniel Vatavu. 2011. The Effect of Sampling Rate on the Performance of Template-based Gesture Recognizers. In *Proceedings of the 13th International Conference on Multimodal Interfaces (ICMI '11)*. ACM, New York, NY, USA, 271–278. https://doi.org/10.1145/2070481.2070531

[57] Radu-Daniel Vatavu. 2012. 1F: One Accessory Feature Design for Gesture Recognizers. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces (IUI '12)*. ACM, New York, NY, USA, 297–300. https://doi.org/10.1145/2166966.2167022

[58] Radu-Daniel Vatavu, Lisa Anthony, and Jacob Wobbrock. 2018. $Q: A Super-Quick, Articulation-Invariant Stroke-Gesture Recognizer for Low-Resource Devices. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '18)*. ACM, New York, NY, USA, 623–635. https://doi.org/10.1145/3229434.3229465

[59] Radu-Daniel Vatavu, Géry Casiez, and Laurent Grisoni. 2013. Small, Medium, or Large?: Estimating the User-perceived Scale of Stroke Gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 277–280. https://doi.org/10.1145/2470654.2470692

[60] Radu-Daniel Vatavu, Laurent Grisoni, and Stefan-Gheorghe Pentiuc. 2009. Gesture-Based Human-Computer Interaction and Simulation. Springer-Verlag, Berlin, Heidelberg, Chapter Gesture Recognition Based on Elastic Deformation Energies, 1–12. https://doi.org/10.1007/978-3-540-92865-2_1

[61] Jo Vermeulen, Kris Luyten, Elise van den Hoven, and Karin Coninx. 2013. Crossing the bridge over Norman's gulf of execution: revealing feedforward's true identity. In *2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '13, Paris, France, April 27 - May 2, 2013*, Wendy E. Mackay, Stephen A. Brewster, and Susanne Bødker (Eds.). ACM, 1931–1940. https://doi.org/10.1145/2470654.2466255

[62] Don Willems, Ralph Niels, Marcel van Gerven, and Louis Vuurpijl. 2009. Iconic and Multi-stroke Gesture Recognition. *Pattern Recogn.* 42, 12 (Dec. 2009), 3303–3312. https://doi.org/10.1016/j.patcog.2009.01.030

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

[64] Jiahui Wu, Gang Pan, Daqing Zhang, Guande Qi, and Shijian Li. 2009. Gesture Recognition with a 3-D Accelerometer. In *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing (UIC '09)*. Springer-Verlag, Berlin, Heidelberg, 25–38. https://doi.org/10.1007/978-3-642-02830-4_4

[65] Daiki Yamaji. 2016. *A Fast and Lightweight Unistroke Recognizer for Large User-defined vocabulary (in Japanese)*. Ph.D. Dissertation. University of Tsukuba, Ibaraki, Japan. Retrieved September 2, 2017 from http://www.iplab.cs.tsukuba.ac.jp/master-e/.

[66] 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 (ICMI '15)*. ACM, New York, NY, USA, 219–226. https://doi.org/10.1145/2818346.2820734