Recognizing 3D Trajectories as 2D Multi-stroke Gestures

MEHDI OUSMER, Université catholique de Louvain, LouRIM, Belgium
ARTHUR SLUŸTERS, Université catholique de Louvain, LouRIM, Belgium
NATHAN MAGROFUOCO, Université catholique de Louvain, LouRIM, Belgium
PAOLO ROSELLI, Università degli Studi di Roma, Italy and Univ. cath. de Louvain, Belgium
JEAN VANDERDONCKT, Université catholique de Louvain, LouRIM, Belgium

While end users can acquire full 3D gestures with many input devices, they often capture only 3D trajectories, which are 3D uni-path, uni-stroke single-point gestures performed in thin air. Such trajectories with their (x, y, z) coordinates could be interpreted as three 2D stroke gestures projected on three planes,*i.e.*, XY, YZ, and ZX, thus making them admissible for established 2D stroke gesture recognizers. To investigate whether 3D trajectories could be effectively and efficiently recognized, four 2D stroke gesture recognizers, *i.e.*, $P, $P+, $Q, and Rubine, are extended to the third dimension: $\$P^3$, $\$P+^3$, $\$Q^3$, and Rubine-Sheng, an extension of Rubine for 3D with more features. Two new variations are also introduced: $\$F$ for flexible cloud matching and FreeHandUni for uni-path recognition. Rubine3D, another extension of Rubine for 3D which projects the 3D gesture on three orthogonal planes, is also included. These seven recognizers are compared against three challenging datasets containing 3D trajectories, *i.e.*, SHREC2019 and 3DTCGS, in a user-independent scenario, and 3DMadLabSD with its four domains, in both user-dependent and user-independent scenarios, with varying number of templates and sampling. Individual recognition rates and execution times per dataset and aggregated ones on all datasets show a highly significant difference of $\$P+^3$ over its competitors. The potential effects of the dataset, the number of templates, and the sampling are also studied.

CCS Concepts: • **Human-centered computing → Gestural input**; **Graphical user interfaces**;

Keywords: Large display interfaces and multi-display environments; Gesture-based interfaces; Gesture recognition; Mid-air gestural interaction; Stroke gestures; Surface computing; 3D trajectory.

ACM Reference Format:
Mehdi Ousmer, Arthur Sluÿters, Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. 2020. Recognizing 3D Trajectories as 2D Multi-stroke Gestures. In *Proceedings of the ACM on Human-Computer Interaction*, Vol. 4, ISS, Article 198 (November 2020). ACM, New York, NY. 21 pages. https://doi.org/10.1145/3427326

1 INTRODUCTION AND MOTIVATIONS

Nowadays, more and more 3D input devices enable end users to acquire gestures in space and many environments are developed for this purpose [37]. Yet, an important portion of these gestures

Authors' addresses: Mehdi Ousmer, Université catholique de Louvain, LouRIM, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium, mehdi.ousmer@uclouvain.be; Arthur Sluÿters, Université catholique de Louvain, LouRIM, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium, arthur.sluyters@uclouvain.be; Nathan Magrofuoco, Université catholique de Louvain, LouRIM, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium, nathan.magrofuoco@uclouvain.be; Paolo Roselli, Università degli Studi di Roma, Piazzale Aldo Moro, 5, Roma, 00185, Italy, Univ. cath. de Louvain, Place des Sciences, 2, Louvain-la-Neuve, 1348, Belgium, roselli@mat.uniroma2.it,paolo.roselli@uclouvain.be; Jean Vanderdonckt, jean.vanderdonckt@uclouvain.be, Université catholique de Louvain, LouRIM, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium, jean.vanderdonckt@uclouvain.be.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.
2573-0142/2020/11-ART198 $15.00
https://doi.org/10.1145/3427326

Fig. 1. Two 3D trajectories: (1) a "V" gesture for controlling the volume, (2) a helicoidal gesture for controlling the level of details.

covers 3D variations of existing 2D gestures [10, 28]. We hereby refer to these gestures as *3D trajectories*, *i.e.*, 3D uni-path, uni-stroke single-point gestures performed in thin air. These gestures can be classified according to the number of spatial dimensions in which gestures are acquired: 0D for point-based gestures (*e.g.*, a simple tap), 1D for linear gestures (*e.g.*, a "V" ring gesture increases or decreases the volume of a smartphone while talking – see Fig. 1-1), 2D for planar gestures (*e.g.*, surface stroke gestures [57]), and 3D for spatial gestures (*e.g.*, a helicoidal gesture increases or decreases the level of details of a virtual reality scene while navigating – see Fig. 1-2).

The importance of 3D trajectories partially relies in our kinesthetic sensory-motor system, which controls 3D spatial gestures, their amplitude, and target reaching. Different parts of our body are used as external and internal stimuli to perceive our environment. *Proprioception* represents our body position relatively to any other part or to any object of the environment, primarily by our hands and fingers. Our human vision is almost planar in reception, thus inducing a lack of feedback in gesturing to be compensated by *extroception*, our capacity to position in space.

This phenomenon has several manifestations. A significant part of gesture taxonomies [1] covers symbolic gestures, which reproduce in 3D various familiar 2D gestures, such as letters, digits, symbols, drawings [36], probably because of the legacy bias [31]. Cheng's taxonomy [11] classifies 3D gestures into three categories: static gestures, trajectories consisting of sequential data (temporal for example) of one point of the gesture, and continuous for general limb trajectory. In gesture elicitation studies [55], several gestures proposed by participants are actually 3D trajectories [29, 53]. Gheran *et al.* [15] report that participants proposed 80 gestures representing hand poses or combinations between hand poses and trajectories in mid-air with one ring device and 89 with two rings. Even when a task is spatial, like in 3D navigation [33], end users prefer 3D trajectories.

Since the body of 3D trajectories becomes more important, their recognition is gaining more interest. While many efficient 3D gesture recognizers exist [5, 6, 17, 18, 27, 28, 30] in general and in particular for mobile devices [38, 44], they are not specifically tailored to 3D trajectories and turn out to be more complex than necessary for recognizing them. This raises the question whether 3D trajectories can be recognized by state-of-the-art 2D stroke gesture recognizers [48]. To address this question, this paper proposes the following contributions:

(1) We conduct a targeted literature review of 2D stroke recognizers (Section 2) resulting into a selection of four candidates (Section 3.1) to create a three-dimensionalized version, *i.e.*, Rubine [35], a pioneer in stroke gesture recognition (Section 3.2), $*P* [50] for its accuracy (Section 3.4), $*P*+ [49] for its optimization for low-vision users (Section 3.7), and $*Q* [52] for its speed namely for low-end devices (Section 3.8).

(2) We develop $*F* (Section 3.5) and *FreeHandUni* (Section 3.6), two new $P variants for controlling the sampling and the point matching. We integrate these six recognizers and Rubine-Sheng (Section 3.3), a 3D version of Rubine recognizer, into a testing framework (Section 3.9).
(3) We evaluate these seven recognizers on challenging gesture sets (Section 4): two in user-independent scenario and one in both user-dependent and user-independent scenarios to collect their recognition rates and execution times. Some recognizers achieve a recognition rate and an execution time comparable to existing 3D recognizers for these gestures.
(4) We discuss the implications for choosing a particular recognizer depending on the conditions imposed on the context of use, such as sampling and amount of points (Section 6).

2 RELATED WORK

We conducted a Targeted Literature Review (TLR), which is a non-systematic, in-depth and informative literature review, that keeps only the references maximizing rigorousness and relevance while minimizing selection bias [24].

2.1 2D Stroke Gesture Recognizers

Several algorithms recognize 2D stroke gestures [57] by *template matching* [7]: they compare a *candidate gesture* against pre-recorded *template gestures* gathered by *gesture classes* in a *training set* that was used to train the recognizer beforehand. This assumes calculating a (dis-)similarity distance between the candidate gesture and template classes.

GrandMa, Rubine's recognizer [35], creates a vector of 13 geometric features for each template and candidate, which is classified with respect to gesture classes by a linear evaluation. This offers a high recognition accuracy for a low computational cost, manual opportunistic programming is no longer required, and its understanding is simple and facilitated by a geometric interpretation. These three advantages form a common motto for several other recognizers.

The $1 recognizer [55] recognizes 2D uni-stroke gestures into four steps: resampling, rotation, scaling, and moving. The candidate gesture is resampled into a fixed number of points that are evenly spaced along the path. Then the path is rotated so that the line from the centroid of the path to the first point of the path is parallel to the x-axis. Non-uniform scaling is used to fit the path in a reference square. Finally, the path moves its centroid to the origin $(0, 0)$. The mean value of the Euclidean distance between the corresponding points is computed as a similarity distance.

Instead of the Euclidean distance, Protractor [26] minimizes the cosine distance between two gestures by calculating the best rotation angle, but no scaling is performed. LVS [12] computes the Levenshtein distance between two gestures decomposed into characters representing directions.

The $*N* recognizer [3] generalizes $1 to 2D multi-stroke gestures. Each gesture sample is considered as a uni-stroke where part of the gesture is made away from the sensing surface (*i.e.*, by following the user's hand through the air), and the algorithm automatically generates all possible uni-stroke permutations of the gesture.

The $*P* recognizer [50] overcomes the combinatorial explosion of $*N* by representing gestures as unordered point-clouds, therefore making them invariant to order, number, and direction. The $*P*+ algorithm [49] increases the $P flexibility such that any point from the first cloud can be matched with any point from the second cloud. Although this recognizer was tested for low-vision users, it turns out to be the most accurate recognizer of the $-family.

The $*Q* recognizer [52] improves $*P* with several algorithmic revisions: the classification of multi-stroke gestures is faster when the size of the training set is large. Aligned with the accuracy of $*P*, $*Q* is announced as the fastest recognizer of the $-family. While formerly aimed at user interface prototyping, the $-family initiated several applications and recognizers [9, 16, 22, 23, 34, 47].

Penny Pincher [42] converts resampled gesture points into a series of between-point vectors and calculates the sum of the angles between their corresponding vectors. Penny Pincher, due to this simplicity, is reported as a super-fast algorithm with a high accuracy (>90%). Jackknife [43] is a multitool for recognizing gestures sampled from many modalities such as 2D touch, 3D hand, and full-body gestures. Gestures are represented as a set of unit-length direction vectors. The resampled gesture points are converted into a series of unit-length direction vectors. Dynamic Time Warping (DTW) then matches a candidate and a template direction vectors. Correction factors are computed to inflate the score of dissimilar gestures. The !FTL recognizer [46] converts resampled gesture points into a series of pairs of vectors called "shapes" and computes the Local Shape Distance (LSD) between each pair of shapes in sequence to assess how two gestures are dissimilar.

2.2 3D Gesture Recognizers

2.2.1 By Template Matching. The $3 recognizer [20] is probably the first to extend an existing 2D stroke recognizer to 3D: a $1 variant records 3D mid-air gestures using the accelerometer data of a single point issued by a Nintendo Wii Remote Controller device. Protractor3D [21] adds rotation invariance to $3 by finding the rotation that minimises the sum of square errors (Euclidean distance) between a candidate and a template. The uWave recognizer [27] also exploits the data from a three-axis accelerometer, which is particularly appropriate for wearable devices such as smartwatches and smartphones. Raw data are compressed by an averaging window, are quantized non-linearly, and matched with another time series by Dynamic Time Warping (DTW). More recently, the 3¢ algorithm [10] extends the 1¢ [16] with the same principles as in $3 and Protractor3D: comparing sampled 3D trajectories, but with different and simpler processing options. This translates into a better accuracy and a higher performance. The algorithm performs scaling with respect to the length of the gesture (with a normalised length of 1) instead of scaling a fixed size bounding box [9].

2.2.2 By Other techniques. 3D gesture recognition is a huge field of research and development in view the proliferation of devices, wearable or not, human limbs considered for gesturing, and techniques used for recognition. Many techniques are used [11, 56]: general purpose toolkits [54], Artificial Neural Networks (ANN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), *k*-Nearest Neighbor [14, 51], Support Vector Machine (SVM), Hidden Markov Model (HMM), Dynamic Time Warping (DTW) [43]), Naive Bayes classifier, Principal Component Analysis (PCA), Deep Learning [28]). A common pattern identified in this portion of the literature is the overfitting in the data sets for artificial intelligence and machine learning algorithms: they exhibit an accuracy superior to 90% and a performance inferior to 100 msec., a threshold for system response time [32]), but they are entirely optimized for a given dataset, thus leaving little or no space for keeping the same performance if the dataset is changed. This occurs when samples and/or classes need to be modified, thus leading to a re-training of the algorithm, although this is now achievable in real-time.

3 THREE-DIMENSIONALIZATION OF TWO-DIMENSIONAL RECOGNIZERS

2D recognizers selected from Section 2 are now extended and implemented into 3D.

3.1 Selection of 2D Recognizers and their Three-Dimensionalization

Based on the TLR in Section 2, four recognizers were selected for comparative testing: (1) *Rubine*: this pioneer in 2D stroke recognition performs a statistical matching on weighted feature vectors which are computed from training examples, an easily repeatable process for any plane; (2) $$P$ introduced the cloud matching principle that preserves scaling and translation-invariances, as well as order, number, and direction invariances; (3) $$P$+ displays the best accuracy to our knowledge; (4) $$Q$ benefits from the fastest execution time to our knowledge.

$1 is not included as several aforementioned recognizers outperforms it [42, 46]. Despite two speed optimizations, $$N$ is not included neither because it faces a combinatory explosion in both memory and execution time. $$P$ remedies this limitation by considering gestures as unordered sets of points. Similarly, the other extensions and local optimizations discussed in Section 2 are not included. $3 and similar algorithms are admissible candidates, but are finally not included as they require additional data coming from a wearable device, such as an accelerometer, a constraint that we do not want to impose. Vector-based recognizers [42, 46] are not retained since they potentially require to re-compute all vectors between points or vectors between vectors for each plane, which could increase their computational cost. Other 3D recognizers, which belong to classes of classifiers with different computational complexities to cover a wider range of 3D gestures, were not included.

3.2 Rubine3D, a 3D Extension of Rubine

Rubine3D (*R3D*) extends the feature-Based 2D gesture recognizer Rubine [35], since it takes charge of 3D gestures. Inspired by the iGesture framework [40], our Rubine3D recognizer combines a set of three individual 2D Rubine recognizers, one for each plane XY, YZ, and ZX. The projection on each plane is done by transforming each point of the 3D trajectory to the three coordinate system planes (XY, YZ, ZX). The general equation of a plane in an orthogonal coordinate system is represented by the linear equation $Ax + By + Cz + D = 0$. The coordinates of the normal vector $n(A, B, C)$ to a plane are the coefficients in the general equation above. If the plane passes through the origin, the equation has constant term D=0. Hence, the equations of respectively the coordinate system XY, YZ, ZX planes that pass through the origin are z=0, x=0, and y=0. Let us consider $M(i, j, h)$ as a point belonging to a 3D trajectory. P, Q, R are the three points where the three coordinates of the point M pass through the coordinate system plan (XY, YZ, ZX). These 2D points are the orthogonal projections of M on each plane with vectors parallels to X, Y, and Z axes. Therefore, $P(i, j)$, $Q(j, h)$, and $R(h, i)$ are the 2D points coordinates of the projection on (XY, YZ, ZX).

First, the raw data are pre-processed by scaling and filtering points. Thus, a point is discarded if the distance from the previous point is under a threshold of d=.005. Next, for each training gesture projected on each plane, the thirteen original features $(f_1, .., f_{13})$ [35] are calculated.

After that, every class mean feature vector and covariance matrix are computed for all planes. We then calculate the common covariance matrix and the inverse matrix, of which the gesture classes weights are estimated. Thus, there are several actions to determine the possible class of the candidate gesture. To begin, the gesture is projected on each plane.

Next, the feature vectors are extracted from the three gestures projections. After that, the gestures projections are defined as one of the possible gesture classes with the aid of a linear function. This function evaluates which class maximizes the evaluation function result.

In the end, a heuristic inspired by iGesture, is used to determine the result class of the 3D gesture. If the resulting classes of the three projections are the same, it is designated as the final result. Otherwise, if there are two or three different classes, we calculate the evaluations of these classes for the three planes. Then, each evaluation result is multiplied by a weight factor in relation to planes (W_{XY}=0.4, W_{YZ}=0.3, W_{ZX}=0.3). These values are determined according to the easiness of producing gestures by our human body in the saggital plane, then in the transversal plane, and finally in the frontal plane, which is observed particularly for head and shoulders gestures [45]. In addition, we tried to perform the same algorithms with two planes only, but we observed a decrease in the accuracy (see Appendix B.1 for this test).

The greatest sum is decided as the final gesture class. However, the gesture rejection part coming from Rubine was not implemented to keep the extension in line with other recognizers in terms of computational complexity and since we only focus on the classification problem.

3.3 Rubine-Sheng, Another 3D Extension of Rubine

Rubine-Sheng (*RS*) uses a vector of sixteen features computed from a 3D gesture $G = \{p_t = (x_t, y_t, z_t) | \forall t = 1, ..., n\}$. These features extend the Rubine's original feature vector to 3D by adding three features, they are proposed in the AdaBoost recognizer and applied to 3D gestures [39]. As for its 2D counterpart, the Rubine-Sheng recognizer classifies a candidate gesture by finding its corresponding gesture class. First, with training samples feature vectors, the estimation of mean feature vectors is computed as well as the covariance matrix of each class. After that, the common covariance matrix is calculated and its inverse from which the weights of gesture classes are calculated. Finally, the class of a candidate gesture is the class that maximizes the result of the discrimination function.

3.4 $\$P^3$, a 3D Extension of $\$P$

The $\$P$ algorithm matches the point cloud of a candidate gesture C against the point cloud of each template T in the training set via a function M that associates each point $C_i \in C$ with exactly one point $T_j \in T$, $T_j = M(C_i)$. The classification result determines the closest template T to the candidate C through matching distance calculation: $C \in$ class of T where $T = argmin_T\{\$P(C, T)\}$. This principle is generalized to 3D by defining a set of points with three-dimensional coordinates: $C = \{p_i = (x_i, y_i, z_i) | \forall i = 1, ..., n\}$. $\$P^3$, our 3D extension of $\$P$, performs three pre-processing steps to minimize the gesture variations: (1) gestures point clouds are resampled to get the same number of points n for matching them; (2) gesture points are scaled to a non-uniform reference box; and (3) the gesture centroid (Gx, Gy, Gz) is translated to the origin $O = (0, 0, 0)$. The matching score is then defined as the sum of Euclidean distances for all pairs of points from M. This is generalized into 3D by adding the third coordinates z in the calculation. Let us assume that the point C_i from the candidate gesture C is matched to the point T_j from the template T. Hence, the score is given by:

$$\sum_{i=1}^{n} \parallel C_i - T_j \parallel = \sum_{i=1}^{n} \sqrt{(C_i.x - T_j.x)^2 + (C_i.y - T_j.y)^2 + (C_i.z - T_j.z)^2} \quad (1)$$

In order to compute the dissimilarity score between two clouds of points, we performed a one-to-one time-free alignment between points, inspired by Vatavu's matching heuristic named Greedy-5 because it gave the best results among the tested methods [50]. The heuristic matches for each point from the first cloud one point in the second cloud with the condition that it has not been matched before. Next, it matches points from the second cloud that have not been matched yet can be matched to one point in the first set resulting in a complexity of $O(n^2)$. The algorithm runs several times with different starting points considered circularly through all points, and returns the minimum matching of all runs. The returned matching sum is multiplied by a weight representing a confidence degree on the matching.

$$w = \sum_{i} wi \cdot \parallel C_i - T_j \parallel \quad (2)$$

The ϵ threshold controls the number of runs and affects the complexity of the heuristic to $O(n^{2+\epsilon})$. We take into account that the direction of matching impacts the result of the heuristic.

3.5 $\$F$, a Flexible Variant of $\$P^3$

$\$F$ extends the aforementioned $\$P^3$ with $\$P$+'s flexible cloud matching [49]. As usually, the candidate and the templates points are resampled to equidistantly-spaced points, scaled within a unit box, and translated so their centroid is at the origin $(0, 0, 0)$. The template having to the lowest dissimilarity score is considered as the best matching template for the candidate. The $\$P$+'s cloud matching process consists of matching the points from the first cloud with their closest point from the second

cloud, then of matching the points from the second cloud that have not been matched yet with their closest point in the first cloud. To investigate whether a more flexible matching could be obtained, $\$F$ enables the matching of two points that have already been matched, hence its name $\$F$.

3.6 FreeHandUni, a 3D Extension of $P

The FreeHandUni (*FH*) recognizer is derived from the FreeHand recognizer pseudocode which extends the 2D gesture recognizer $P++ [13]. Moreover, it uses full hand information for free-hand gesture recognition. FreeHandUni is adapted to meet our need for recognizing the unipath character of 3D trajectories by replacing the hand pose structure with a 3D point structure (x, y, z). With this modification, FreehandUni corresponds to an improvement of $\$P^3$, using a flexible cloud matching based on a one-to-many alignment between points [49]. The pre-processing remains exactly the same, the Euclidean distance of Eq. 1 is used in the matching process with more flexibility, each point of the template cloud being matched with the closest point from the candidate cloud, then matching each remaining point from the candidate cloud with the closest point from the template cloud. The gesture class is identified as the class of the template cloud with the lowest dissimilarity score between it and the candidate cloud, and conversely. FreeHandUni is different from $\$F$ in that the early abandoning is not implemented, to align the computational complexity to $\$P^3$.

3.7 $P+^3, a 3D Extension of $P+

One of the $\$P+$ major improvements with respect to $\$P$ lies in expanding the matching from one-to-one to one-to-many points, which clears the weights of the starting point used in $\$P$. Based on the 3D points, we compute the turning angle at each point with points coordinates gesture:

$$C_{i.\theta} = \frac{1}{\pi} \arccos\left(\frac{(C_{i+1}.x - C_i.x)\cdot(C_{i+1}.x - C_i.x) + (C_{i+1}.y - C_i.y)\cdot(C_{i+1}.y - C_i.y) + (C_{i+1}.z - C_i.z)\cdot(C_{i+1}.z - C_i.z)}{||C_{i+1} - C_i|| \cdot ||C_i - C_{i-1}||}\right) \quad (3)$$

To optimize the execution time, the early abandoning used in $\$Q$ is added, as well as the third coordinate in the computation of the point distance beside the angle as in $\$P+$:

$$D(C_i, T_j) = \sqrt{(T_j.x - C_i.x)^2 + (T_j.y - C_i.y)^2 + (T_j.z - C_i.z)^2 + (T_j.a - C_i.a)^2} \quad (4)$$

3.8 $Q^3, a 3D Extension of $Q

$\$Q^3$, our 3D extension of $Q [52], performs the same pre-processing as in $\$P+^3$, apart from calculating a 3D Look-Up-Table (LUT) offline for every template and storing it with the cloud points. The look-up point-matching technique is generalized to three dimensions via a 16×16×16 3D grid of equidistant points. Each cloud point C is nested inside this grid. Next, the index of the row, of the column, and of the layer of the closest point in the grid are stored for each point, which has a computation complexity of $O(n \cdot m^3)$. To recognize a candidate, the closest point from a template to C_i is identified by iterating through cloud points and summing the Euclidean distances of Eq. 1 between each pair of points. This computation stops when the sum exceeds the minimum dissimilarity score.

3.9 Testing Framework

In order to perform a comparative testing on the same, consistent basis, the seven aforementioned recognizers, *i.e.*, Rubine3D, Rubine-Sheng, $\$F$, FreeHandUni, $\$P^3$, $\$P+^3$, and $\$Q^3$, have been all implemented in JavaScript (ECMA Script 2019 version), a cross-platform language which satisfies testing requirements. We also developed in JS a framework for testing the recognizers, built in a modular way to accommodate variations of sensors, gesture sets, and recognizers. The workflow consists of a pipeline architecture (Fig. 2): the recognizer is trained with the data provided by the

dataset-loader; the sensor sends data frame by frame in a pose-classifier. If a pose (a static gesture) is recognized, it is sent directly to the application. Otherwise the data is either sent directly to the recognizer or to a segmenter [19] if the recognizer does not manage this step. Once a gesture is recognized, a message is sent to the application and the corresponding action is performed.

4 EVALUATION

We evaluated the seven recognizers in the traditional user-dependent and user-independent procedures [3, 4, 49, 50] with 3D trajectories from six datasets in order to show the efficiency of these recognizers depending on the conditions. We evaluated the classification task only, not the pre-processing since this task can be achieved off-line.

4.1 Experiment

4.1.1 Design. Our study was within-factors with four independent variables:

(1) Recognizer: nominal variable with 7 conditions, representing the various recognizers implemented for recognizing 3D trajectories: $\$P^3$, $\$P+^3$, $\$Q^3$, $\$F$, *FH* (FreeHandUni), *R3D* (Rubine3D), and *RS* (Rubine-Sheng).
(2) Dataset: nominal variable with 6 conditions (see Table 1 and Appendix A), representing two datasets considered as a whole, *i.e.*, SHREC2019 [8] and 3DTCGS [9], and four domains of 3DMadLabSD [18]-D1, 2, 3, and 4.
(3) Number of Templates: numerical variable with 5 conditions, representing the number of templates per gesture class used to train the recognizer: T={1, 2, 4, 8, 16}.
(4) Sampling: numerical variable with 5 values representing the number of points per gesture: N={4, 8, 16, 32, 64}.

Name	Sensor	Subjects	Classes	Instances	Ground truth information
SHREC2019 [8]	Leap Motion	13	5	195	T, L, J
3DTCGS [9]	Leap Motion	13	26	347	T, L
3DMadLabSD [18]	SoftKinetic DepthSense DS325	10	40 (10 x 4 Domains)	4000	T, L

Table 1. Description of selected datasets. Notation for the Ground truth information: Timestamp (T), Label (L), Hand joints (J).

4.1.2 Apparatus. We used a sexa-core Intel Core i7 2.20 GHz CPU and running a Windows 10 Home Edition operating system. The RAM was 16 GB DDR4 memory with 2400 MHz. We ran the framework described in Section 3.9 with the seven recognizers on the six individual datasets.

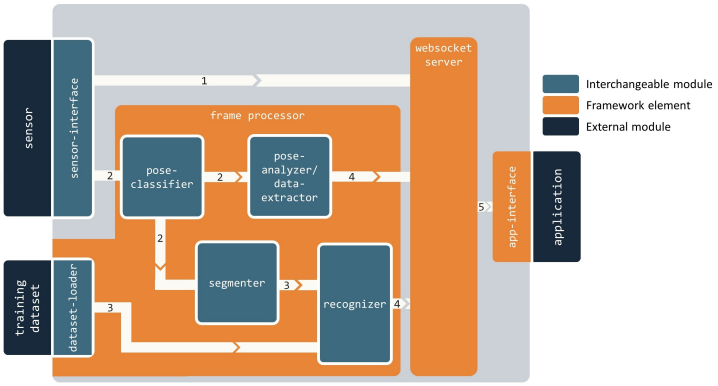

Fig. 2. Framework software architecture.

The **SHREC2019** dataset [8] served as a benchmarking for the Eurographics 2019 SHape Retrieval Contest (SHREC) track on online gesture recognition to detect command gestures from hands' movements in a virtual reality context. The proposed dataset is topical due to its recency. This dataset consists of 195 3D trajectories performed by 13 participants with the whole hand. Each trajectory contains a sequence of 3D points and quaternions designating one of five iconic gestures: "Cross" (X), "Circle" (O), "V-mark" (V), "Caret" (/\), and "Square" ([])). Since this dataset contains unsegmented gestures, the training set and the testing set were merged to create a unique dataset in which, for every trajectory, unnecessary hand movements were filtered. We clipped the sequences which predate and postdate the main section [19].

The **3DTCGS** dataset [9] is employed to test the classification performance of the 3¢ [9] recognizer on interface command gestures. Data were provided in a segmented form which makes them easier to exploit during experimentation for the classification task. This dataset contains gestures with a varying complexity, ranging from the simplest such as a "3D swipe" to the most complex one such as a "3D spiral". Similar trajectories also include a different direction (*e.g.*, "left-swipe" vs. "right-swipe", "arc3Dleft" vs. "arc3Dright"), to test the direction-invariance property. Each subject produces one sample per gesture which makes the user-independent evaluation particularly challenging. Participants were asked to issue short iconic gestures with the dominant hand forefinger. The gathered data consist of 347 sequences of 3D coordinates. Combined with a timestamp, they describe 26 gesture classes performed by 14 subjects and recorded with a Leap Motion (see Appendix A).

The **MadLabSD** (MadLab Sketch Dataset) dataset [18] is publicly available online since 2018 and was used to assess a new gesture-based system [18]. This large dataset comprises mid-air single stroke gestures which are palm-centred dynamic motion trajectories. A variety of gestures from the most common symbols (*i.e.*, alphabets and numerals) to the most unusual ones (*e.g.*, CAD primitives) makes this dataset particular attractive. A group of 10 users recorded 40 segmented symbols and sketches split into four domains (see Fig. 6) for a total of 4,000 samples. Each user recorded 10 samples per gesture which consist of 3D coordinates of the centre of his palm in 3D with a timestamp.

Moreover, they extracted the centre palm from the depth greyscale images which were recorded with a SoftKinetic DepthSense DS325. During the hand tracking, the static hand pose was used to record the gesture. The user can switch between active (hand open) and inactive (closed hand) states. Raw data from the depth camera are not pre-processed, thus potentially containing spurious data, such as noisy points at the start of the sketch, due to the user latency after mode switching[18]. The dataset contains four domains: (D1) Arabic numerals: 0-9, (D2) English alphabets (Handwritten): a-j, (D3) Simulation symbols: Spring-Mass-Wheel-Pulley-Hinge-Fast Forward-Rewind-Play-Pause-Delete, and (D4) CAD primitives: Cuboid-Cylinder-Sphere-Rectangular Pipe-Hemisphere-Cylindrical Pipe-Pyramid-Tetrahedron-Cone-Toroid.

4.2 Procedures and Quantitative Measures

We compute the *recognition rate* (computed as the ratio of positive recognitions divided by the total number of trials) and the *execution time* (computed in milliseconds as the time for pre-processing a candidate gesture and recognizing its result class) for the 7 (Recognizer) × 6 (Dataset) = 42 basic configurations following the typical method used in the literature to evaluate gesture recognizers [2–4, 50, 52, 55]: the *user-independent scenario* evaluates the recognition on gestures produced by users who are different from those used for training the recognizer; the *user-dependent scenario* evaluates the recognition on gestures produced by users who are the same who trained the recognizer.

4.2.1 User-Independent Scenario. One template is randomly selected for each gesture class from all participants and saved for the testing. Then, a training set is obtained by randomly choosing T

templates for each gesture class for all users. They should be different from the templates previously selected for the testing. Then, the recognizer is trained on the resulting training set. Each time a candidate is positively recognized, an internal counter is incremented. This operation is repeated 100 times for each T: T={1, 2, 4, 8, 16} for the SHREC2019 and MadLabSD, T={1, 2, 4, 8, 11} for the 3DTCGS. While T={2, 3, 4, 8, 16} the SHREC2019 and MadLabSD, and T={2, 3, 4, 8, 11} for the 3DTCGS, for the Rubine's recognizer 3D extensions (*R3D* and *RS*). In the end, the number of recognized gestures is averaged to get the recognition rate and formatted as a percentage. Overall, we performed 6 (Dataset) × 5 (Sampling) × 5 (Number of Templates) × 100 (repetitions) × 7 (Recognizer) = 105,000 recognition trials.

4.2.2 User-Dependent Scenario. One template is randomly selected from each gesture class for each user. Next, for each user, a set of randomly selected T templates is created for every gesture class different from the templates selected for the testing previously from the same user. Then, the recognizer is trained on the resulting training dataset. Each time a candidate is positively recognized, an internal counter is incremented. This operation is repeated 100 times for each user and for each T depending on the recognizer: T={2, 3, 4, 8} for *R3D* and *RS*, T={1, 2, 4, 8} for the others. In the end, the number of recognized gestures is averaged to get the recognition rate and show it in a percentage format for each N and for each recognizer. Only the MadLabSD dataset was used because SHREC2019 does not identify the gesture's emitter and 3DTCGS only includes one template per user for each gesture class. Overall, we performed 4 (Dataset) × 10 (users) × 5 (Sampling) × 4 (Number of Templates) × 100 (repetitions) × 7 (Recognizer) = 560,000 recognition trials. If we sum up all trials, a grand total of 665,000 trials is obtained.

5 RESULTS

We used IBM SPSS V27 to perform a series of one-way ANOVAs for evaluating the effect of the recognizer, the number of templates T, and the sampling N on the recognition rate and the execution time. We used Tukey's HSD post-hoc analysis when Levene's test [25] did not indicate unequal variances, and Games-Howell when it did. Data were submitted to a Bonferroni Type I correction before handling. Table 2 summarizes these results for all datasets and per individual dataset.

5.1 Recognition Rate

5.1.1 User-independent Scenario. The recognizers are sorted in decreasing order of their recognition rate averaged on *all* datasets as follows (see overall results in Fig. 14 and individual results per dataset in Fig. 9 to 13 in Appendix B): \$$P$+^{3} is superior to \$$F$, then FH, \$$Q^3$, \$$P^3$, ended by the two Rubine conditions *R3D* and *RS*. \$$P$+^{3} is 9.98% more accurate than its successor \$$F$, which is roughly in the same interval as FH, \$$Q^3$, and \$$P^3$, which is in turn 15.29% more accurate than *R3D*. The overall difference between the recognizers was statistically very highly significant ($F_{6,10459}$ = 352.89, $^{***}p$<.001, η^2=.019 (–)). Table 6 details the results of ANOVAs for all datasets (column "Overall") and per dataset. For example, \$$P$+^{3} is more accurate than \$$F$ with a very high significant difference (q=21.18) with a small effect size, more accurate than FH with a very high significant difference but without effect size, more accurate than \$$Q^3$ without any effect size, and so forth. In short, \$$P$+^{3} is more accurate than all other recognizers with a very highly significant difference, the effect size ranging from none to medium. The difference between \$$F$ and FH, \$$Q^3$ is not significant, but becomes significant over \$$P^3$ with a medium effect size, over *R3D* and *RS* with a large effect size.

5.1.2 User-dependent Scenario. The recognizers are sorted in decreasing order of their recognition rate averaged on *all* datasets as follows (see right part of Fig. 10 to Fig. 13 in Appendix B): \$$P$+^{3} is superior to FH, then \$$F$, \$$Q^3$, \$$P^3$, ended by the two Rubine conditions *R3D* and *RS*. \$$P$+^{3} is only 2.04% more accurate than its successor FH, which is roughly in the same interval as \$$F$,

Q^3, and P^3, which is in turn 25.66% more accurate than *R3D*. The overall difference between the recognizers was again statistically very highly significant. Table 7 details the results of ANOVAs for the four 3DMadLab domains (column "Overall") and per domain. For example, $P+^3$ is more accurate than $F ($q$=14.38), *FH* ($q$=13.62), Q^3 (q=16.13), and P^3 (q=19.50), all with a very high significant difference and a small effect size. In short, $P+^3$ is more accurate than all other recognizers with a very high significant difference, the effect size ranging from small to large. Similarly, the difference between $F and *FH*, Q^3 is not significant, but becomes significant over P^3 without any effect size, over *R3D* and *RS* with a small and large effect size, and so forth.

5.1.3 User-dependent vs. User-independent Scenario. The $P+^3$ recognizer remains the most accurate for the first domain (*i.e.*, the ten digits) in both user-dependent (M=98.45%) and user-independent scenarios (M=88.08%). For all values of N, a high recognition rate is obtained as soon as with one template and the best recognition rate for more templates. With two or more templates ($T>2$) used for training, *R3D*'s curve surpasses all the other curves except the $P+^3$ in the user-independent scenario and is competitive with other $-like recognizers in the user-dependent scenario. With less than three templates, *R3D* is inaccurate, which explains why its overall rate is inferior to the others, except for *RS*. In the user-independent scenario, $-like recognizers are belong to an envelope that progressively grows when N grows. In the user-dependent scenario, their curves are confounded most of the time. The recognition rates of the second domain, *i.e.*, the lowercase letters a-j, are quite similar to the first domain with a margin of about 1-2%. For the third domain, *i.e.*, the simulation symbols, $-like recognizers are close to perfection in the user-dependent

Recognizer		Datasets											
		Overall		SHREC	3DTCGS	Domain 1		Domain 2		Domain 3		Domain 4	
		UD	UI	UI	UI	UD	UI	UD	UI	UD	UI	UD	UI
$P+^3$	rate (M)	74.40	87.48	86.94	84.28	98.45	88.08	98.81	87.87	99.26	94.45	93.86	66.36
	rate (SD)	1.73	13.68	0.75	2.22	4.13	2.34	3.67	1.27	2.78	0.76	0.92	1.65
	time (M)	0.53	0.77	0.36	1.24	0.54	0.76	0.53	0.77	0.49	0.69	0.54	0.81
	time (SD)	1.43	0.85	0.64	2.00	0.87	1.37	0.87	1.39	0.78	1.24	0.88	1.46
$F	rate (M)	68.13	79.54	79.36	78.83	96.22	76.38	96.78	75.45	97.67	85.66	91.76	57.94
	rate (SD)	1.89	16.08	3.55	2.76	6.98	1.63	6.70	3.28	5.26	1.28	1.07	1.68
	time (M)	0.42	0.59	0.29	0.91	0.41	0.57	0.42	0.58	0.38	0.53	0.45	0.67
	time (SD)	1.04	0.63	0.48	1.40	0.63	0.96	0.64	1.00	0.57	0.91	0.69	1.15
FH	rate (M)	68.46	79.49	79.28	78.84	96.21	76.05	96.69	75.78	97.70	85.55	91.96	57.75
	rate (SD)	1.89	16.28	0.93	2.80	6.97	3.20	6.79	3.24	5.28	1.26	1.04	1.70
	time (M)	0.94	1.23	0.49	2.41	0.92	1.07	0.92	1.08	0.93	1.14	0.97	1.19
	time (SD)	2.55	1.65	0.94	4.22	1.62	2.04	1.62	2.07	1.66	2.13	1.70	2.30
Q^3	rate (M)	67.36	79.09	79.36	78.45	95.79	75.20	96.28	74.55	97.51	84.88	91.31	57.02
	rate (SD)	1.91	16.54	0.93	2.94	7.22	3.21	7.05	3.22	5.47	1.31	1.09	1.72
	time (M)	3.12	3.76	3.01	4.81	3.14	3.55	2.97	3.53	3.13	3.42	3.25	4.27
	time (SD)	4.16	3.14	2.94	5.39	3.11	3.70	2.89	3.67	3.19	3.51	3.36	4.92
P^3	rate (M)	65.90	77.50	78.10	77.43	95.28	72.57	95.70	71.55	97.18	82.92	90.68	54.98
	rate (SD)	1.93	16.88	0.96	2.99	7.70	3.19	7.54	3.28	5.93	1.40	1.12	1.73
	time (M)	8.71	10.94	4.80	20.79	8.59	9.83	8.65	10.19	8.72	9.87	8.88	10.16
	time (SD)	24.45	16.76	10.02	39.79	16.49	20.69	16.62	21.38	16.76	20.87	17.15	21.12
R3D	rate (M)	60.80	67.22	52.51	74.48	75.60	72.57	75.94	70.33	76.93	76.52	72.99	54.40
	rate (SD)	2.98	24.93	1.61	2.29	38.46	6.49	38.48	6.45	38.83	3.43	3.74	2.69
	time (M)	0.08	0.08	0.05	0.12	0.06	0.07	0.06	0.06	0.07	0.08	0.10	0.10
	time (SD)	0.027	0.016	0.013	0.018	0.004	0.009	0.005	0.005	0.005	0.009	0.002	0.007
RS	rate (M)	50.04	57.45	40.82	68.82	69.07	58.37	70.76	58.10	73.36	67.98	65.04	43.49
	rate (SD)	2.63	23.56	1.51	2.50	36.18	5.77	36.86	5.66	38.01	3.24	3.48	2.35
	time (M)	0.03	0.03	0.02	0.05	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.04
	time (SD)	0.012	0.007	0.006	0.011	0.002	0.002	0.005	0.005	0.003	0.003	0.004	0.002

Table 2. Summary of recognition rates and execution times for all datasets and per individual dataset. Recognizers are sorted in decreasing order of their average recognition rate for all datasets.

scenario with $\$P+^3$ being the champion in both cases (M=99.26% and M=94.45%, respectively). The results for the user-independent scenario are slightly superior to those obtained for the previous domains. However, the rates are very low for the fourth domain (*i.e.*, the CAD symbols) in the user-independent scenario (they are all below 66%), but still very good for the user-dependent scenario (between M=93.86% for $\$P+^3$ and M=90.68% for $\$P^3$).

For all four domains, the recognition rate significantly decreases when we are switching from the user-dependent scenario to the user-independent scenario, which was expected. Recognizing a 3D trajectory that is different from the templates used for training remains more demanding than re-identifying an already existing one. Fig. 3 depicts the loss of recognition rate in terms of a difference of percentage from user-dependent to user-independent. *R3D* wins the lowest averaged difference of percentage (M=11, 75%), but its global recognition rate is below the $-like recognizers. The second place is occupied by $\$P+^3$, which undergoes the next lowest averaged difference of percentage (M=17.50% on four domains), followed by $\$F$, *FH*, $\$P+^3$, $\$P^3$, and *RS*.

In conclusion, $\$P+^3$ reasonably resists to user-independence while keeping the best recognition rate. The two first domains, *i.e.*, digits and letters, remain constant in terms of recognition loss for each recognizer, again with a loss of 12% for $\$P+^3$. Overall, the third domain benefits from the minimum loss, probably because of straightforward symbols, and the fourth domain suffers from the maximum loss, probably because of the most uncommon symbols.

Recognizer		**Scenario**	
G1	G2	UI	UD
$\$P+^3$	$\$F$	21.18,***, S	14.38,***, S
	FH	21.30,***, –	13.62,***, S
	$\$Q^3$	22.89,***, –	16.13,***, S
	$\$P^3$	27.50,***, S	19.50,***, S
	R3D	37.75,***, S	31.19,***, M
	RS	61.51,***, M	21.18,***, S
$\$F$	*FH*	0.11, *n.s.*	0.76, *n.s.*
	$\$Q^3$	1.70, *n.s.*	1.75, *n.s.*
	$\$P^3$	6.31,***, M	5.11,**, –
	R3D	16.56,***, L	16.81,***, S
	RS	40.32,***, L	41.48,***, L
FH	$\$Q^3$	1.58, *n.s.*	2.51, *n.s.*
	$\$P^3$	6.19,***, M	5.87,***, –
	R3D	16.44,***, L	17.57,***, S
	RS	40.22,***, L	42.25,***, L
$\$Q^3$	$\$P^3$	4.61,***, S	3.36, *n.s.*
	R3D	14.86,***, L	15.06,***, S
	RS	36.61,***, L	39.73,***, L
$\$P^3$	*R3D*	10.24,***, L	11.69,***, S
	RS	34.01,***, L	36.37,***, M
R3D	*RS*	23.75,***, L	24.67,***, S

Table 3. ANOVAs computed for the 7 Recognizers in the user-independent (UI) and user-dependent (UD) scenarios: G1=group 1, G2=group 2. For each dataset, three data are provided: the q value resulting from the ANOVA, the significance of the p value if any (***$p \leq .001$), and Cohen's d coefficient for effect size ((S)mall when $d \geq .02$, (M)edium when $d \geq .05$, (L)arge when $d \geq .08$, and (–) when no significant effect size ($d < .02$)).

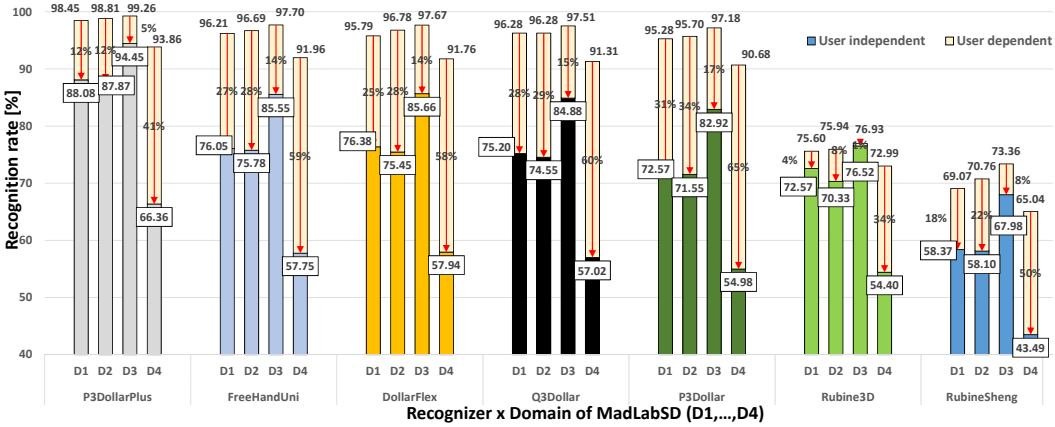

Fig. 3. Difference of percentage in recognition rate loss for all recognizers on the MadLabSD dataset [18], all domains, when restricting the scenario from user-dependent to user-independent. Recognizers are sorted in decreasing order of their global recognition rate.

5.2 Execution Time

Appendix D plots the execution time for the six datasets based on tables for the user-independent scenario and the user-dependent scenario in Appendix D.2, resp. in Appendix D.3. For each table, a colour key scheme is determined based on the median values of the execution times obtained in these configurations. Grey lines of Table 2 report times averaged for all datasets.

5.2.1 User-independent Scenario. Fig. 15 plots the execution time in msec. for all recognizers in all conditions for the 3DTCGS and SHREC2019 datasets. Overall, *R3D* (M=.047, SD=.013) and *RS* (M=.017, SD=.006) have the lowest and the most constant times in all configurations. On the contrary, the $-like recognizers are the most time-consuming: 1F (M=.289, SD=.490) is the fastest recognizer among them, followed by 1P+^{3} (M=.363, SD=.649), *FH* (M=.492, SD=.956), 1Q^3 (M=3.005, SD=3.005) and 1P^3 (M=4.798, SD=10.224). This difference suggests that feature-based recognizers, like *R3D* and *RS*, are not influenced by the same parameters as template-based ones (*i.e.*, 1P^3, 1P+^{3}, 1Q^3, 1F, and *FH*). *RS* is the fastest recognizer in most N and T conditions, except when N=4 and $T \leq 4$. The times for all $-like recognizers increase while N and T grow. 1Q^3 is the slowest recognizer for N=4, 8. Other curves are under the 1Q^3 one but from N=16, 1P^3 execution times increase quickly and exceed the execution time of 1Q^3 for T=16 and it occurs even quicker for N=32 and N=64, where 1P^3 exceeds 1Q^3 immediately after T=4, resp., T=2. *RS* is three-time faster than *R3D*: the ratio of their averaged execution times is $\frac{t_{RS}}{t_{R3D}}$=2.82. This ratio makes sense considering that *R3D* computes 39 features of the candidate (3 times 13 features for the three planes *XY*, *YZ*, and *ZX*) while *RS* computes 16 features once. For these feature-oriented recognizers, the candidate is pre-processed, features are extracted and multiplied by pre-computed weights of each class, then summed. The candidate's class is designated by the larger sum resulting from this calculation, which confirms a constant time influenced by the number of classes. The 1F, 1P+^{3}, and *FH* have short execution times compared to other $-like recognizers. 1F and 1P+^{3} are better than *FH* ones, this variation is caused by the early abandoning of the two recognizers. The execution times for all the conditions for this dataset are below 100 msec, the limit for a user to feel that a system is operating in real-time [32].

The overall picture for 3DTCGS remains the same as for the SHREC2019: the execution times increase for all the recognizers and conditions, which pushes the 1P^3 over the limit of 100 msec. This is due to the increase of the number of classes and samples per class in the dataset (26 classes

compared with 5 gestures for SHREC2019). We observe a rapid increase of the $\$P^3$ curve which surpasses the $\$Q^3$ just after T=4 for N=8.

The four 3DMadLab domains are depicted in Fig. 16 and Fig 17. Their results are analogous to the first two datasets. Rubine-Sheng remains the fastest recognizer (M_{D1}=.26 ms, M_{D2}=.28 ms,M_{D3}=.31 ms, and M_{D4}=.41ms), whereas the $\$P^3$ stays the slowest one (M_{D1}=9.83 ms, M_{D2}=10.19 ms, M_{D3}=9.87 ms, M_{D4}=10.16 ms). For the fourth domain containing the CAD symbols, the execution times are slightly above the other domains, probably because their shape complexity with more crossing points and angle variations require more comparisons.

5.2.2 User-dependent Scenario. The execution time curves of the four domains are depicted in Fig. 18 and Fig. 19, resp. In contrast to the user-independent scenario, *RS* remains the fastest recognizer in all conditions (M_{D1}=.030 ms, M_{D2}=.027 ms, M_{D3}=.030 ms, and M_{D1}=.042ms). The $\$P^3$ curve is located under the $\$Q^3$ for N=4 ($M_{\$P^3}$=.058 ms and for $M_{\$Q^3}$=.447 ms) and N=8 ($M_{\$P^3}$=.444 ms and $M_{\$Q^3}$=.853 ms). The $\$P^3$ execution time curve progressively increases as N increases and it intersects with the $\$Q^3$ at different points, the intersection point is determined by T. The execution time for this scenario is larger than for the user-independent scenario because of the method used in JavaScript to return the timestamp during the recognition. For the user-independent scenario, we used performance.now(), which is a high performance method returning a value representing the time spent since the beginning of the program with a precision of up to one microsecond, whereas in the user-dependent scenario, we used date.now() which is a method returning the number of milliseconds elapsed since UNIX epoch, which limits the precision by rounding it to 1 millisecond.

6 DISCUSSION AND LIMITATIONS

The evaluation of the seven recognizers on the six datasets suggests the following considerations:

- $\$P+^3$ stands out as the first-class recognizer for the 3D trajectories tested with the best recognition rate in almost all conditions with a very high statistical significance, apart from a few exceptions, such as when the number of templates is very reduced. It is the most robust to user-independence in terms of rate loss and benefits of an excellent execution time in almost all conditions, thus making it the first choice tenable for recognizing efficiently 3D trajectories, especially in contrast to other recognizers.
- In contrast, *RS* is always the worst recognizer for the 3D trajectories tested. As it suffers from low to very low recognition rates, it should be simply avoided.
- After the $\$P+^3$ recognizer, there is a recurrent pattern in the ordering of subsequent recognizers both in terms of recognition rate and execution time: a first batch of other $-like recognizers appears with $\$F$, FH, $\$Q^3$, and $\$P^3$ and a second batch containing the two feature-oriented algorithms, *i.e.*, *R3D* and *RS*.
- The preference for recognizers belonging to the first batch, by contrasting and weighting their recognition rate more than their execution time, is: $\$P+^3$, $\$F$, FH, $\$Q^3$, and $\$P^3$.
- Our two $\$F$ and FH variations have some more flexibility, but significantly, improve the recognition rate with a small to medium effect size.
- The preference for recognizers belonging to the second batch is: *R3D* is always before *RS*.
- $\$P+^3$, our tri-dimensionalization of $\$P+$, gives superior results, as well as $\$Q^3$ for $\$Q$, and $\$P^3$ for $\$P$. *R3D* also gives superior results to *RS* in all conditions and *RS* should just be forgotten.

All reported results are averaged on all datasets, ranging from the simplest one to the most complex one (*e.g.*, Domain 4). In order to determine any effect of the datasets (*i.e.*, SHREC2019, 3DTCGS, Domains 1–4 in the user-independent scenario and Domain 1–4 in the user-dependent scenario) on the recognition rate and the execution time, we computed another series of one-way ANOVAs (Table 4). The analyses showed a significant effect of the datasets on the recognition

Recognizer	Effect	User independent			User dependent		
		$F_{5,144}$	p	η^2	$F_{3,76}$	p	η^2
$\$P^3$	Datasets × Rates	21.75	***	.43 (L)	8.66	***	.25 (L)
	Datasets × Time	1.15	*n.s.*	.04 (–)	0	*n.s.*	0 (–)
$\$Q^3$	Datasets × Rates	21.87	***	.43 (L)	8.86	***	.26 (L)
	Datasets × Time	0.61	*n.s.*	.02 (–)	0	*n.s.*	0 (–)
$\$P+^3$	Datasets × Rates	41.48	***	.59 (L)	20.9	***	.45 (L)
	Datasets × Time	0.95	*n.s.*	.03 (–)	0	*n.s.*	0 (–)
$\$F$	Datasets × Rates	22.1	***	.43 (L)	8.45	***	.25 (L)
	Datasets × Time	0.93	*n.s.*	.03 (–)	0	*n.s.*	0 (–)
FH	Datasets × Rates	22.48	***	.44 (L)	8.28	***	.25 (L)
	Datasets × Time	1.55	*n.s.*	.05 (–)	0	*n.s.*	0 (–)
R3D	Datasets × Rates	3.64	**	.11 (M)	0	*n.s.*	0 (–)
	Datasets × Time	147.18	***	.84 (L)	357.48	**	.93 (L)
RS	Datasets × Rates	6.07	***	.17 (L)	.19 0	*n.s.*	.01 (–)
	Datasets × Time	89.39	***	.76 (L)	65.55	***	.72 (L)

Table 4. The effect of dataset on the recognition rate and the execution time with the effect size ((L)arge when $\eta^2 \geq .14$, (M)edium when $\eta^2 \geq .06$, (S)mall when $\eta^2 \geq .01$, and (—) when there is a null effect size).

rate with a large effect size, but no significant effect on the execution time in both scenarios. A medium effect size in the user-independent scenario was observed on the *R3D* recognition rate, but no significant effect on it and on *RS* recognition rate in the user-dependent scenario. The Tukey's HSD and Games-Howell post-hoc analyses revealed that the recognizers got the lowest rate for Domain 4 in the two scenarios. For instance, there were significant differences for the $\$P+^3$ recognition rate in user-independent scenario between the different datasets, Domain 4 *vs.* SHREC2019 ($M_{\text{Diff}}=-20.588$,$^{***}p<.001$), Domain 4 *vs.* 3DTCGS ($M_{\text{Diff}}=-17.923$,$^{***}p<.001$), Domain 4 *vs.* Domain 1 ($M_{\text{Diff}}=-21.720$,$^{***}p<.001$), Domain 4 *vs.* Domain 2 ($M_{\text{Diff}}=-21.512$,$^{***}p<.001$), Domain 4 *vs.* Domain 3 ($M_{\text{Diff}}=-28.092$,$^{***}p<.001$), and Domain 4 *vs.* 3DMadLab ($M_{\text{Diff}}=-20.588$,$^{***}p<.001$). There was also a significant effect of the datasets on *R3D* and *RS* execution time. The *R3D* execution times are significantly different between each pair of datasets. Both T and N have a significant impact on the recognition rate and execution time of most recognizers (Table 5):

- Regarding the $\$P^3$, $\$Q^3$, $\$P+^3$, $\$F$, and *FH*, we observe a significant effect of both factors, with a large effect size in the two scenarios. The Games-Howell analyses revealed that the execution time is significantly slower as the number of points increases.
- The $\$Q^3$ execution time is not significantly impacted by the number of templates in both scenarios. The effect of the number of templates on the execution time of $\$F$ is significant with a medium effect size, the Game-Howell post-hoc analysis revealed no significant difference between the execution times of each T in the user-dependent scenario.
- As per *R3D* and *RS*, only T had a significant effect on the recognition rates with a large effect size. The Game-Howell confirms that the recognizers are more accurate with more templates.

We now discuss some limitations of our experiment. Other recognizers, such as $3 [20] or vector-based recognizers [42], were not considered, mainly due to their computational complexity. Comparing them against the winner of this test, *i.e.*, $\$P^3+$, in the same conditions might be interesting. Although we tested six different datasets, covering a wide range of 3D trajectories in terms of variety, complexity, and familiarity, their gestures mostly represent semaphoric gestures [1] associated to a particular meaning. The effect of depth on performing and recognizing such gestures was not studied in this experiment, and represent an opportunity to test robustness to depth instability. The 3D trajectories were tested within our framework (Fig. 2) running on a personal

Recognizer	Effect	User independent			User dependent		
		$F_{4,145}$	p	η^2	$F_{3,76}$	p	η^2
\$$P^3$	Templates × Rates	16.31	***	.31 (L)	16.80	***	.40 (L)
	Points × Rates	9.40	***	.21 (L)	5.98	***	.24 (L)
	Templates × Time	5.69	***	.14 (L)	4.04	***	.14 (L)
	Points × Time	25.27	***	.41 (L)	24.87	***	.57 (L)
\$$Q^3$	Templates × Rates	15.31	***	.30 (L)	15.44	***	.38 (L)
	Points × Rates	9.90	***	.21 (L)	6.22	***	.25 (L)
	Templates × Time	0.64	*n.s.*	.02 (–)	0.09	*n.s.*	0 (–)
	Points × Time	311.36	***	.90 (L)	1587.00	***	.99 (L)
\$$P$+^{3}	Templates × Rates	11.46	***	.24 (L)	7.99	***	.24 (L)
	Points × Rates	2.65	*	.07 (M)	2.16	*n.s.*	.10 (-)
	Templates × Time	6.26	***	.15 (L)	3.51	*	.12 (M)
	Points × Time	33.96	***	.48 (L)	36.02	***	.66 (L)
\$$F$	Templates × Rates	16.77	***	.32 (L)	13.89	***	.36 (L)
	Points × Rates	8.66	***	.19 (L)	6.88	***	.27 (L)
	Templates × Time	6.70	***	.16 (L)	3.52	*	.12 (M)
	Points × Time	34.95	***	.49 (L)	39.12	***	.68 (L)
FH	Templates × Rates	16.25	***	.31 (L)	13.83	***	.35 (L)
	Points × Rates	8.79	***	.20 (L)	7.26	***	.28 (L)
	Templates × Time	6.62	***	.15 (L)	4.76	**	.16 (L)
	Points × Time	22.65	***	.39 (L)	23.83	***	.56 (L)
R3D	Templates × Rates	89.56	***	.71 (L)	12047.47	***	1 (L)
	Points × Rates	0	*n.s.*	0 (–)	0	*n.s.*	0 (–)
	Templates × Time	0.42	*n.s.*	.01 (–)	0.13	*n.s.*	.01 (–)
	Points × Time	0.14	*n.s.*	0 (–)	0.1	*n.s.*	.01 (–)
RS	Templates × Rates	70.35	***	.66 (L)	2258.11	***	.99 (L)
	Points × Rates	0	*n.s.*	0 (–)	0.09	*n.s.*	0 (–)
	Templates × Time	1.21	*n.s.*	.03 (–)	0.37	*n.s.*	.01 (–)
	Points × Time	0.19	*n.s.*	.01 (–)	0.10	*n.s.*	.01 (–)

Table 5. The effect of number of templates and points on the recognition rate and the execution time with the effect size ((L)arge when $\eta^2 \geq .14$,(M)edium $\eta^2 \geq .06$, (S)mall $\eta^2 \geq .01$, and (–) when null effect size).

computer, thus suggesting to reproduce the experiment in a real-world application to investigate other user-oriented factors, like user experience, memorability, attractiveness, and system-oriented factors, such as computational power and memory usage. While the effect of number of templates and sampling is investigated, we did not explore the potential impact of the number of gesture classes on recognition rates and execution times (5 for SHREC2019, 26 for 3DTCGS, and 10 for each 3DMadLabSD). It might be interesting to repeat the evaluation for the complete 3DMadLabSD at once with 40 classes. We do not know however what is the maximal amount of gesture classes that an end-user may remember.

We did not test some other recognizers, such as \$3 [20] or vector-based recognizers [42, 46], mainly because they apparently require more computational steps than our three-dimensionalized versions. This remains to be proven. The 3¢ algorithm [10] was tested on SHREC2019, but not on the other ones. Therefore, it might be interesting to test them in the same conditions against the winner of this test, *i.e.*, \$$P^3$+.

Although we tested six different datasets, covering a wide range of 3D trajectories in terms of variety, complexity, and familiarity, these gestures mostly represent semaphoric gestures [1] associated to a particular meaning, not particularly to a movement. The 3D trajectories tested represent more symbols drawn in thin air than genuine movement gestures. For example, a spiral can be issued in many different ways, ranging from a flat drawing remaining mostly coplanar to a

depth-varying, representing a helicoidal spiral (Fig. 1-2). The effect of depth on performing and recognizing such gestures was not studied in this experiment, and represent an opportunity to test robustness to depth instability. Finally, the 3D trajectories were only tested for recognition inside the framework, and not in a real-world application. This suggests repeating the same experiment by deploying the framework for a particular application using the same dataset to test other factors, like user experience, memorability, attractiveness, etc.

7 CONCLUSION

This paper is related to 3D trajectories, which are referred to as uni-stroke single-point gestures performed in thin air. Such trajectories with their (x, y, z) coordinates could be interpreted as three 2D stroke gestures projected on three planes,*i.e.*, XY, YZ, and ZX, thus making them admissible for established 2D stroke gesture recognizers. In order to investigate whether these 3D trajectories could be effectively and efficiently recognized via these recognizers, four 2D stroke gesture recognizers selected from a target literature review, *i.e.*, $P, $P+, $Q, and Rubine, are extended to the third dimension: $\$P^3$, $\$P+^3$, $\$Q^3$, and Rubine-Sheng, an extension of Rubine for 3D with more features.

Two new variations are also introduced: $\$F$ for flexible cloud matching and FreeHandUni for uni-path recognition. Rubine3D, another extension of Rubine for 3D which projects the 3D gesture on three orthogonal planes, is also included. These seven recognizers are implemented together in JavaScript in a testing framework to compare them against three challenging datasets containing 3D trajectories, *i.e.*, SHREC2019 [8] and 3DTCGS [9] in a user-independent scenario, and 3DMadLabSD [18] with its four domains, in both user-dependent and user-independent scenarios. Our study was within-factors with four independent variables: the seven recognizers, the six individual datasets, the varying number of templates and the sampling, thus totalling 665,000 trials. Individual recognition rates and execution times per dataset and aggregated ones on all datasets show a highly significant difference between recognizers in most configurations, despite its original goal for user interface fast prototyping: $\$P+^3$, $\$F$, FreeHandUni, $\$Q^3$, $\$P^3$, Rubine3D, and Rubine-Sheng.

The potential effects of the dataset, the number of templates, and the sampling are also studied. Since 3DMadLabSD is evaluated in both scenarios, the recognition loss when switching from user-dependent to user-independent, the most constraining scenario, is also reported: $\$P+^3$ remains the most effective and efficient recognizer in terms of recognition rate, loss, and execution time.

We also release the testing framework developed in JavaScript for the purpose of the evaluation as it is publicly available at https://github.com/Mehous/TestingFramework. Sub-parts contain the seven recognizers, which can be reused in other contexts of use or experiments, for implementing new versions of these recognizers or another version in other markup or programming languages, and for future work contributing to research reproducibility, as recommended by ACM[1].

ACKNOWLEDGEMENTS

The authors of this paper are grateful to the anonymous ACM ISS reviewers for their detailed and constructive comments on earlier versions of this manuscript. They also thank Radu-Daniel Vatavu from University of Suceava for suggesting the research question addressed in this paper based on [48] and for providing us with in-depth guidance for this evaluation.

[1]See https://www.acm.org/publications/policies/artifact-review-and-badging-current

REFERENCES

[1] Roland Aigner, Daniel Wigdor, Hrvoje Benko, Michael Haller, David Lindbauer, Alexandra Ion, Shengdong Zhao, and Jeffrey Tzu Kwan Valino Koh. 2012. *Understanding Mid-Air Hand Gestures: A Study of Human Preferences in Usage of Gesture Types for HCI.* Technical Report MSR-TR-2012-111. https://www.microsoft.com/en-us/research/publication/understanding-mid-air-hand-gestures-a-study-of-human-preferences-in-usage-of-gesture-types-for-hci/

[2] Ahmad Akl and Shahrokh Valaee. 2010. Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2270 – 2273. https://doi.org/10.1109/ICASSP.2010.5495895

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

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

[5] F. Argelaguet, M. Ducoffe, A. Lécuyer, and R. Gribonval. 2017. Spatial and rotation invariant 3D gesture recognition based on sparse representation. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. 158–167.

[6] Said Yacine Boulahia, Eric Anquetil, Richard Kulpa, and Franck Multon. 2017. 3D Multistroke Mapping (3DMM): Transfer of Hand-Drawn Pattern Representation for Skeleton-Based Gesture Recognition. In *Proceedings of the 12th IEEE International Conference on Automatic Face Gesture Recognition* (Washington, DC, USA) *(FG '17)*. IEEE, 462–467. https://doi.org/10.1109/FG.2017.63

[7] R. Brunelli. 2009. *Template Matching Techniques in Computer Vision: Theory and Practice.* John Wiley & Sons, New York.

[8] F. M. Caputo, S. Burato, G. Pavan, T. Voillemin, H. Wannous, J. P. Vandeborre, M. Maghoumi, E. M. Taranta II, A. Razmjoo, J. J. LaViola Jr., F. Manganaro, S. Pini, G. Borghi, R. Vezzani, R. Cucchiara, H. Nguyen, M. T. Tran, and A. Giachetti. 2019. Online Gesture Recognition. In *Eurographics Workshop on 3D Object Retrieval*, Silvia Biasotti, Guillaume Lavoué, and Remco Veltkamp (Eds.). The Eurographics Association. https://doi.org/10.2312/3dor.20191067

[9] Fabio Marco Caputo, Pietro Prebianca, Alessandro Carcangiu, Lucio D. Spano, and Andrea Giachetti. 2017. A 3 Cent Recognizer: Simple and Effective Retrieval and Classification of Mid-air Gestures from Single 3D Traces. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*, Andrea Giachetti, Paolo Pingi, and Filippo Stanco (Eds.). The Eurographics Association. https://doi.org/10.2312/stag.20171221

[10] Fabio M. Caputo, Pietro Prebianca, Alessandro Carcangiu, Lucio D. Spano, and Andrea Giachetti. 2018. Comparing 3D trajectories for simple mid-air gesture recognition. *Computers & Graphics* 73 (2018), 17 – 25. https://doi.org/10.1016/j.cag.2018.02.009

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

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

[13] Elena-Gina Craciun, Ionela Rusu, and Radu-Daniel Vatavu. 2016. *Free-Hand Gesture Recognizer Pseudocode.* http://www.eed.usv.ro/mintviz/projects/GIVISIMP/data/Pseudocode2.pdf

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

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

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

[17] M. Hoffman, P. Varcholik, and J. J. LaViola. 2010. Breaking the status quo: Improving 3D gesture recognition with spatially convenient input devices. In *2010 IEEE Virtual Reality Conference (VR)*. 59–66.

[18] Jinmiao Huang, Prakhar Jaiswal, and Rahul Rai. 2019. Gesture-based system for next generation natural and intuitive interfaces. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 33, 1 (2019), 54–68. https://doi.org/10.1017/S0890060418000045

[19] Sven Kratz and Maribeth Back. 2015. Towards Accurate Automatic Segmentation of IMU-Tracked Motion Gestures. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI EA '15)*. Association for Computing Machinery, New York, NY, USA, 1337–1342. https://doi.org/10.1145/2702613.2732922

[20] 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

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

[22] Per Ola Kristensson and Leif C. Denby. 2011. Continuous Recognition and Visualization of Pen Strokes and Touch-Screen Gestures. In *Sketch Based Interfaces and Modeling, Vancouver, BC, Canada, 5-7 August 2011. Proceedings*, Tracy Hammond and Andrew Nealen (Eds.). Eurographics Association, 95–102. https://doi.org/10.2312/SBM/SBM11/095-102

[23] 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

[24] Lynn Kysh. 2013. Difference between a systematic review and a literature review. (8 2013). https://doi.org/10.6084/m9.figshare.766364.v1

[25] Howard Levene. 1960. Robust tests for equality of variances. In *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, Ingram Olkin and Harold Hotelling et al. (Eds.). Stanford University Press, Palo Alto, CA, USA, 278–292.

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

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

[28] Mehran Maghoumi and Joseph J. LaViola. 2019. DeepGRU: Deep Gesture Recognition Utility. In *Advances in Visual Computing*, George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Daniela Ushizima, Sek Chai, Shinjiro Sueda, Xin Lin, Aidong Lu, Daniel Thalmann, Chaoli Wang, and Panpan Xu (Eds.). Springer International Publishing, Cham, 16–31.

[29] Nathan Magrofuoco, Jorge Luis Pérez-Medina, Paolo Roselli, Jean Vanderdonckt, and Santiago Villarreal. 2019. Eliciting Contact-Based and Contactless Gestures With Radar-Based Sensors. *IEEE Access* 7 (2019), 176982–176997. https://doi.org/10.1109/ACCESS.2019.2951349

[30] Antigoni Mezari and Ilias Maglogiannis. 2017. Gesture Recognition Using Symbolic Aggregate Approximation and Dynamic Time Warping on Motion Data. In *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare* (Barcelona, Spain) *(PervasiveHealth '17)*. Association for Computing Machinery, New York, NY, USA, 342–347. https://doi.org/10.1145/3154862.3154927

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

[32] J. Nielsen. 1994. *Usability Engineering*. Elsevier Science. https://books.google.be/books?id=95As2OF67f0C

[33] F. R. Ortega, A. Galvan, K. Tarre, A. Barreto, N. Rishe, J. Bernal, R. Balcazar, and J. Thomas. 2017. Gesture elicitation for 3D travel via multi-touch and mid-Air systems for procedurally generated pseudo-universe. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. 144–153.

[34] J. Reaver, T. F. Stahovich, and J. Herold. 2011. How to Make a Quick$: Using Hierarchical Clustering to Improve the Efficiency of the Dollar Recognizer. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (Vancouver, British Columbia, Canada) *(SBIM '11)*. ACM, New York, NY, USA, 103–108. https://doi.org/10.1145/2021164.2021183

[35] 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

[36] Ugo Braga Sangiorgi, François Beuvens, and Jean Vanderdonckt. 2012. User Interface Design by Collaborative Sketching. In *Proceedings of the Designing Interactive Systems Conference* (Newcastle Upon Tyne, United Kingdom) *(DIS '12)*. Association for Computing Machinery, New York, NY, USA, 378–387. https://doi.org/10.1145/2317956.2318013

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

[38] V. M. Sethu Janaki, S. Babu, and S. S. Sreekanth. 2013. Real time recognition of 3D gestures in mobile devices. In *2013 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*. 149–152.

[39] Jia Sheng. 2004. *A Study of AdaBoost in 3D Gesture Recognition.* technical report CSC2515. Department of Computer Science, University of Toronto. http://www.dgp.toronto.edu/~jsheng/doc/CSC2515/Report.pdf

[40] Beat Signer, U. Kurmann, and Moira C. Norrie. 2007. iGesture: A General Gesture Recognition Framework. In *9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23-26 September, Curitiba, Paraná, Brazil*. IEEE Computer Society, 954–958. https://doi.org/10.1109/ICDAR.2007.4377056

[41] 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* (Halifax, Nova Scotia, Canada) *(GI '15)*. Canadian Information Processing Society, Toronto, Ontario, Canada, Canada, 195–202. http://dl.acm.org/citation.cfm?id=2788890.2788925

[42] 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* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 5850–5861. https://doi.org/10.1145/3025453.3026002

[43] Nicanor Valdez, Ronnie Besas, China Yu, Donna Dumalaon, and Rowel Atienza. 2014. 3D gestures on 2D screens for mobile games. In *Proceedings of the IEEE Asia Pacific Conference on Wireless and Mobile* (Bali, Indonesia) *(APWiMob '14)*. IEEE, 232–237. https://doi.org/10.1109/APWiMob.2014.6920274

[44] Jean Vanderdonckt, Nathan Magrofuoco, Suzanne Kieffer, Jorge Pérez, Ysabelle Rase, Paolo Roselli, and Santiago Villarreal. 2019. Head and Shoulders Gestures: Exploring User-Defined Gestures with Upper Body. In *Design, User Experience, and Usability. User Experience in Advanced Technological Environments*, Aaron Marcus and Wentao Wang (Eds.). Springer International Publishing, Cham, 192–213.

[45] Jean Vanderdonckt, Paolo Roselli, and Jorge Luis Pérez-Medina. 2018. !!FTL, an Articulation-Invariant Stroke Gesture Recognizer with Controllable Position, Scale, and Rotation Invariances. In *Proceedings of the 20th ACM International Conference 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

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

[47] Radu-Daniel Vatavu. 2013. The impact of motion dimensionality and bit cardinality on the design of 3D gesture recognizers. *International Journal of Human-Computer Studies* 71, 4 (2013), 387 – 409. https://doi.org/10.1016/j.ijhcs.2012.11.005

[48] 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, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4667–4679. https://doi.org/10.1145/3025453.3025941

[49] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures As Point Clouds: A $P Recognizer for User Interface Prototypes. In *Proceedings of the 14th ACM International Conference 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

[50] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2014. Gesture Heatmaps: Understanding Gesture Performance with Colorful Visualizations. In *Proceedings of the 16th International Conference on Multimodal Interaction* (Istanbul, Turkey) *(ICMI '14)*. Association for Computing Machinery, New York, NY, USA, 172–179. https://doi.org/10.1145/2663204.2663256

[51] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. 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* (Barcelona, Spain) *(MobileHCI '18)*. ACM, New York, NY, USA, Article 23, 12 pages. https://doi.org/10.1145/3229434.3229465

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

[53] Tracy Westeyn, Helene Brashear, Amin Atrash, and Thad Starner. 2003. Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition. In *Proceedings of the 5th International Conference on Multimodal Interfaces* (Vancouver, British Columbia, Canada) *(ICMI '03)*. Association for Computing Machinery, New York, NY, USA, 85–92. https://doi.org/10.1145/958432.958452

[54] 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* (Newport, Rhode Island, USA) *(UIST '07)*. ACM, New York, NY, USA, 159–168. https://doi.org/10.1145/1294211.1294238

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

[56] 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

Received July 2020; revised August 2020; accepted September 2020