Comparing Some Distances in Template-based 2D Gesture Recognition

Jean Vanderdonckt
Université catholique de Louvain
Louvain School of Management (LSM)
Louvain Interaction Laboratory (LiLab)
Place des Doyens, 1
B-1348 Louvain-la-Neuve, Belgium
jean.vanderdonckt@uclouvain.be

Bruno Dumas
Université de Namur
Faculté d'Informatique
rue Grandgagnage, 21
B-5000 Namur, Belgium
bruno.dumas@unamur.be

Mauro Cherubini
Université de Lausanne
Faculté des hautes études commerciales
Bâtiment Internef
CH-1015 Lausanne, Switzerland
mauro.cherubini@unil.ch

Permission to make digital or hard copies of part or all 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 third-party components of this work must be honored. For all other uses, contact the owner/author(s).
CHI'18 Extended Abstracts, April 21–26, 2018, Montréal, QC, Canada.
Copyright is held by the author/owner(s).
ACM ISBN 978-1-4503-5621-3/18/04.
https://doi.org/10.1145/3170427.3188452

Abstract
Euclidean distance is traditionally used to compare a gesture candidate against gesture templates in two-dimensional gesture recognizers. This paper compares two distances borrowed from other domains of computer science used in template-based two-dimensional gesture recognizers: the Mahalanobis distance, typically used in computer vision and statistics, and the Jaro-Winckler distance, typically used in information retrieval and pattern recognition. Although the geometric interpretation of these distances is less straightforward for designers, there is a significant impact of the Mahalanobis distance on recognition rate, but not for the Jaro-Winkler one.

Author Keywords
Gesture recognition, gesture recognizer, template-based recognition.

ACM Classification Keywords
H.5. Information Interfaces and Presentation.

Introduction
Two-dimensional template-based gesture recognizers, also called stroke recognizers, compare a gesture candidate against gesture templates collected in a gesture set in terms of classes of samples. For each class, say a triangle, a letter, or a symbol, several samples are captured and stored for pattern matching. After (pre-) processing, gesture recognizers compute a measure between the candidate and the samples to determine whether they are (dis)similar enough to deduce a gesture matching. The

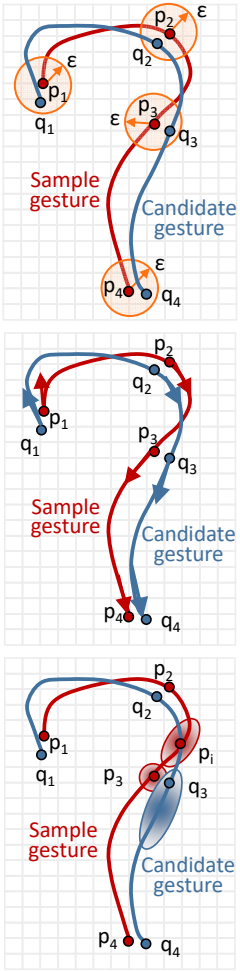

Figure 1: (a) Euclidean distance between points in $P [16], (b) between vectors in Penny Pincher [15], and (c) Mahalanobis distance between points.

members of the $-family 2D gesture recognizers [14] compute the Euclidean distance between the candidate gesture and the sample gestures. Most of them end up with computing such a measure, sometimes referred to as a distance by language abuse. For instance, edit distances are not mathematical distances since they do not satisfy the triangle inequality. Although no systematic study exists today that benchmarks these measures, there is some interest to investigate whether other distances from other domains of computer science may be used favorably independently of these algorithms. This paper investigates two such measures that have never been considered before for 2D gesture recognition.

Related Work

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 [17] and has been extensively used in many recognizers, such as Rubine [15], $1 [18], $N [1], $P [17], and several extensions, such as Penny Pincher [16], 1¢ [7], SHARK2 [10]. The Euclidean distance can be used in both 2D and 3D [17]. A point q_i belonging to a candidate gesture is considered close to a sample one p_i in $P [17] as soon as it belongs to a bowl of center p_i and radius ε (Fig. 1a). Two unitary vectors from p_i and q_i are considered close to each other by PennyPincher [16] (Fig. 1b) when their Euclidean distance is below a certain threshold. Other measures have been investigated in the past.

The *Angular Cosine* computes the similarity between two gestures by calculating the angle between the *n*-dimensional vectors represented by their points [7].

Dynamic Time Warping (DTW) generalizes point-to-point computation of the Euclidean distance while minimizing cost alignment between two gestures [14].

Edit distances compute the similarity between two gestures represented as strings of characters. Each character typically represents one of the four cardinal directions (i.e., (L=left,R=right, U=up, D=down) or one of the eight compass directions (i.e., 0=South, 1=South-East, 2=East, 3=North-East, 4=North, 5=North-West, 6=West, 7=South-West: Fig. 2).

The *Levenshtein distance* expresses the similarity between two strings by computing how many character insertions, deletions, and substitutions are required to transform the candidate string into the sample string. Coyette *et al*. [4] compute the Levenshtein distance between two strings representing the candidate gesture and the sample gestures. As such, it is a particular edit distance and does not benefit from any geometrical interpretation. To recognize the same gesture in any direction, the gesture string should be transformed for all configurations or tested in a more generic way (e.g., a vertical symmetry replaces 4= North by 0=South, 3= North-East by 1=South-East).

The *Stochastic Levenshtein distance* extends the previous one by considering a probabilistic model of the modifications. Only a marginal win was observed in some specific cases for UsiGesture [3].

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. Some derivatives of this distance have been also explored, such as the *Modified Haussdorff distance* [6].

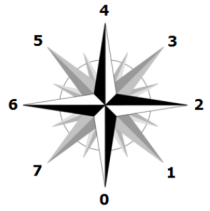

Figure 2: Compass directions for a gesture.

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T \Sigma^{-1} (\vec{x}-\vec{y})}$$
$$\Sigma = \begin{pmatrix} var(X) & cov(X,Y) \\ cov(Y,X) & var(Y) \end{pmatrix}$$

Figure 3: Mahalanobis distance between $\vec{X}$ and $\vec{Y}$.

$$jDistance(A,B) = 1 - jCompare(A,B)$$
$$jCompare(A,B) = \frac{1}{3}\frac{match(A,B)}{length(A)} + \frac{1}{3}\frac{match(A,B)}{length(B)} + \frac{1}{3}\frac{match(A,B) - transpose(A,B)}{match(A,B)}$$

Figure 4: Jaro-Winkler distance between A and B.

Figure 5: Classes of the gesture set: letters and digits.

Two Distances for 2D Gesture Recognition

The general research question to be investigated is whether a gesture recognizer would be impacted by only replacing its initial distance by another measure.

The *Mahalanobis distance*, or generalized squared interpoint distance for its squared value, is defined as a dissimilarity measure between two random vectors $\vec{X}$ and $\vec{Y}$ of the same distribution with the covariance matrix Σ (Fig. 3). If the covariance matrix equals I, the identity matrix, the Mahalanobis distance is reduced to the Euclidean distance. If the covariance matrix is diagonal, then it is reduced to a normalized Euclidean distance. Hence, this distance is considered more general and expressive than the traditional Euclidean distance [13]. If we compute the Euclidean distance in Fig. 1c, q_3 would belong to the class determined by the p_3 bowl since it is closest to the center of this distribution. Instead, according to the Mahalanobis distance, q_3 belongs to the class p_i since it takes the sample distribution into account. For all these reasons, the Mahalanobis distance is widely used in cluster analysis and classification techniques, especially in object and scene recognition for computer vision. In pattern recognition [13], it has been successfully used for accelerometer-based hand gesture recognition [12] and facial recognition [9].

The *Jaro–Winkler distance* is a string metric for measuring the edit distance between two strings, which is the minimum number of single-character transpositions required to change one word into the other [8]. The Jaro–Winkler distance uses a prefix scale p which gives more favorable ratings to strings that match from the beginning for a set prefix length. The lower the Jaro–Winkler distance for two strings is, the closer the strings are. The score is normalized such that 0 indicates no similarity and 1 represents an exact match. The Winkler modification to the Jaro distance [5] acts as a boost on the compare score based on the comparison of the string prefixes. If the Jaro distance is above a given threshold, say 0.7, the score is boosted accordingly with the number of matches between the string prefixes. The prefix size is another variable and is usually set at 4. The prefix match score is the number of exact matches in the prefix of the two strings. The Jaro–Winkler similarity is given by 1−Jaro–Winkler distance in web semantic, information retrieval [5].

User study

To determine the potential impact of these two measures on 2D gesture recognition, a user study is conducted comparing them with respect to a baseline: Jaro-Winkler with respect to the Levenshtein distance as computed by the LVS algorithm [4] (since both are edit distances) and Mahalanobis with respect to Euclidean distance as computed by the $1 algorithm [18].

Methodology

To conduct the study, the UsiGesture environment [2,3] was used since it already incorporates the LVS and the $1 algorithm. Two new recognizers have been implemented in this environment: the LVS algorithm using the Jaro-Winkler distance and the $1 algorithm using the Mahalanobis distance. UsiGesture has been trained beforehand to create a gesture set containing 2x26 classes corresponding to the letters of the Latin alphabet (lowercase and uppercase), the 10 digits from 0 to 9 (Fig. 5), 16 action commands representing 8 flicks and 8 marks (arrows), and 8 geometrical shapes (Fig. 6). The training gesture set therefore contains 86 classes of samples repeated by 30 participants each, thus giving 86x30=2,580 samples. All details are accessible at https://sites.google.com/site/comparativeevaluation/data-collection. For the purpose of the user study, all the

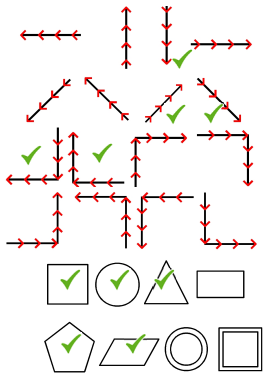

Figure 6: Classes of the gesture set: action commands and shapes.

letters were kept for the candidate set along with a selection of 5 actions and 5 shapes (checked in Fig. 6), thus giving a set of 26+26+5+5= 62 gestures to produce. Since only unistroke recognizers were tested, multi-stroke symbols, like the double circle or square, were left out. The rectangle was considered close to the square. The original $1 recognizer does not recognize directional arrows [11], so different directions were selected in Fig. 6

Hypotheses

The specific research question that this study aims to explore is: is Jaro-Winkler better than Levenshtein, is Mahalanobis better than Euclidean and is there any significant difference between? In order to answer this research question, we present the following hypotheses:

H_1=Jaro-Winkler produces a better recognition rate than Levenshtein for a same gesture set.
H_2=Mahalanobis produces a better recognition rate than Euclidean for a same gesture set.

Task

Participants were instructed to perform a sequence of 62 randomly-selected gestures (Figs. 5, 6) with one sample per gesture and to avoid making corrections. If necessary, they were asked to erase the complete gesture and to start again. The position and scale of the gestures on the screen were free. Participants were told to act naturally without any time or goal constraint.

Apparatus

The physical setup was similar for all participants: a Dell Inspiron computer built with an Intel 1.6GHz CPU, 2Gb DDR of RAM and a 15 inches LCD screen with a resolution of 1400x1050 and equipped with a A4Wacom Intuos3 Pen tablet (9x12 inches) with a precision of 200 lines per millimeter.

Participants

Sixteen subjects (male and female) participated, aged between 23 and 55, with different backgrounds and system experiences. The scenario was a *user-independent*: the system has been trained by other subjects than those involved in the experiment. All participants were regular computer users and they were recruited in other departments of our organization through a mailing list of volunteers. The design setup was therefore as follows: 16 participants x 62 gestures = 992 samples, compared to the training gesture set. Each gesture was then submitted to the 4 recognizers, thus producing 4 conditions: Direction with Levenshtein (original LVS algorithm with Levenshtein distance), Direction with Jaro-Winkler distance, $1 with original Euclidean distance, and $1 modified with Mahalanobis distance.

Results and Discussion

Fig. 7 graphically depicts the results for the four categories of symbols for the four conditions with a confidence interval of α=0.05. Left part of Fig. 7 concerns the 26 lowercase letters: each bar represents the average recognition rate with a confidence interval of α=0.05 obtained for all the 26 letters x 16 participants = 416 samples. Student's t-Test (Paired Two Sample for Means) was systematically computed for comparing two conditions at a time. Jaro-Winkler produced a significantly worse recognition than Levenshtein ($t(31)=2.73$, $p^{**}\leq 0.01$), Mahalanobis performed significantly better than Euclidean, but not so strongly ($t(31)=1.90$, $p^{*}\leq 0.05$). A large effect size was observed based on Cohen's d=1.78 and Gates's Δ=4. Euclidean ($t(31)=6.37$, $p^{***}\leq 0.001$ and $t(31)=8.69$, $p^{***}\leq 0.001$) and Mahalanobis produced a significantly better recognition rate than both Levenshtein and Jaro-Winkler (not represented to keep the figure legible): edit distances do not

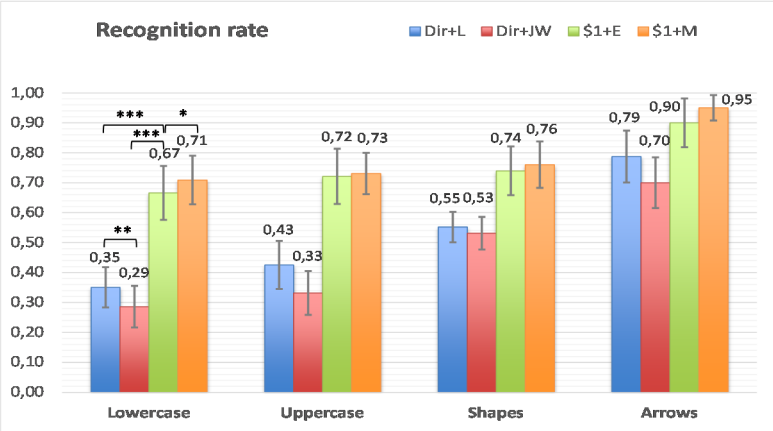

Figure 7: Recognition rate for the user study (*: $p \leq 0.05$, **: $p \leq 0.01$, ***: $p \leq 0.001$).

easily distinguish fine-grained changes of direction as encountered in lowercase letters, which are probably the most complex symbols to reproduce in this controlled experiment. This may explain why their recognition rates are low comparatively to $1 in both versions. The results for the uppercase letters are quite similar: $1+M was found slightly better than $1+E, which in turn was proved significantly better than Levenshtein, itself better than Jaro-Winkler. Only a small effect size was indicated by Cohen's d=0.07 and Gates's Δ=1. The $1 algorithm has been tested with similar letters in the literature: its average recognition rate is comparable to the one obtained in user-independent scenarios reported insofar [1].

Regarding the shapes, Jaro-Winkler was not found performing better than Levenshtein: $t(31)=0.80$, $p>0.05$. $1+M did not perform better than $1+E neither ($t(31)=0.49$, $p>0.05$), but remains still superior to Levenshtein ($t(31)=5.58$, $p^{***} \leq 0.001$). The five tested shapes were relatively standard and easy shapes for which there was probably no need to have an extended version of the Euclidean distance, which was already proved sufficient for these kinds of symbols.

The conclusion regarding the arrows is similar and even stronger: the symbols are so straightforward to recognize that relying on a more sophisticated distance is not appropriate. The Levenshtein was already enough with respect to Jaro-Winkler ($t(31)= 1.96$, $p^* \leq 0.05$) and was not proved significantly inferior to $1+E ($t(31)= 1.64$, $p>0.05$), which is itself not significantly worse than $1+M ($t(31)=1.29$, $p>0.05$).

$1+M benefits from a slightly higher average recognition rate than $1+E for all four categories (e.g., μ=0.73 and σ=0.03 with respect to μ=0.72 and σ=0.08 for lowercase letters - μ=0.95 and σ=0.01 with respect to μ=0.90 and σ=0.02 for arrows). Both versions of $1, i.e. the original $1+E and the modified $1+M, were discovered significantly better than Levenshtein for all categories, which is itself better than Jaro-Winkler. So, in conclusion, H_1 is not supported at all while H_2 is partially supported.

The average recognition rate for all participants involved in this user study was μ=0.63 with a standard deviation of σ=0.07 for a median M=0.62. Overall, participants ended up with a very similar range of recognized gestures, even if the distribution of recognized gestures is varying from one recognizer to another. Only two participants produced a recognition performance that was higher than others: P16 with an exceptional rate of 0.86 and P2 with a rate of 0.72. Conversely, the P5 participant suffered from a low recognition rate of 0.50.

Conclusion and Future Work

This paper compared two distances to investigate whether they could impact the recognition rate of some 2D gesture recognizers. The Mahalanobis geometric distance was found performing better than the Euclidean distance for the same $1 algorithm, more significantly for lowercase and uppercase letters than for shapes and arrows with a different effect size. For simple gestures, the Mahalanobis distance was not found significantly superior to the Euclidean. For complex gestures that are

hard to reproduce by human being, the Mahalanobis distance offers a larger tolerance in its geometrical surface. The Jaro-Winkler edit distance was not found better than the Levenshtein distance for the same LVS algorithm based on gesture string representation [4]. The Mahalanobis distance was applied to the $1 algorithm [17] only, which is a single-stroke recognizer. This preliminary result should investigated with other recognizers such as $N [1] and $P [17], the most recent and significant member of the $-family of recognizers. $P is a good candidate because it is a between-points multi-stroke recognizer. Any other 2D/3D recognizer may also benefit from this extension.

References

1. L. Anthony and J.O. Wobbrock. 2010. A Lightweight Multistroke Recognizer for User Interface Prototypes. In *Proc. of GI '10*. Canada, 245–252.
2. F. Beuvens and J. Vanderdonckt. 2012a. Designing Graphical User Interfaces Integrating Gestures. In *Proc. of SIGDOC '12*. ACM, NY, USA, 313–322.
3. F. Beuvens and J. Vanderdonckt. 2012b. UsiGesture: An environment for integrating pen-based interaction in user interface development. In *Proc. of RCIS' 12*. IEEE Press, 1–12.
4. A. Coyette, S. Schimke, J. Vanderdonckt, and Cl. Vielhauer. 2007. Trainable Sketch Recognizer for Graphical User Interface Design. In *Proc. of IFIP Interact'2007*. Springer, Berlin, 124–135.
5. E. Deza, M.-M. Deza. 2016. *Encyclopedia of Distances*, 4th ed., Springer, Berlin.
6. M.-P. Dubuisson and A. Jain. 1994. A Modified Hausdorff distance for object matching. In *Proc. of IAPR '94*. IEEE Press, 566–568.
7. J. Herold and T. F. Stahovich. 2012. The 1¢ Recognizer: A Fast, Accurate, and Easy-to-implement Handwritten Gesture Recognition Technique. In *Proc. of SBIM' 12*. Eurographics, Germany, 39–46.
8. M.A. Jaro. 1995. Probabilistic linkage of large public health data file. *Stat. in Med. 14*, 5–7, 491–498.
9. S. Kapoor, S. Khanna, and R. Bhatia. 2010. Facial gesture recognition using Correlation and Mahalanobis Distance. *Int. J. of Computer Science and Information Security 7*, 2, 267–272.
10. P.-O. Kristensson and S. Zhai. 2004. SHARK2: A Large Vocabulary Short hand Writing System for Pen-based Computers. In *Proc. of UIST '04*, 43–52.
11. L.A. Leiva, V. Alabau, V. Romero, A.H. Toselli, and E. Vidal. 2015. Context-Aware Gestures for Mixed-Initiative Text Editing UIs. *Interacting with Computers 27*, 6, 1 November 2015, 675–696.
12. T. Marasović and V. Papić. 2012. Accelerometer based gesture recognition system using distance metric learning for nearest neighbour classification. In *Proc. of IEEE MLSP '12*. IEEE Press.
13. G. McLachlan. 2004. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons.
14. C. Pittman, E.M. Taranta II, and J.J. LaViola, Jr. 2016. A $-Family Friendly Approach to Prototype Selection. In *Proc. of IUI' 16*. ACM, USA, 370–374.
15. D. Rubine. 1991. Specifying Gestures by Example. In *Proc. of SIGGRAPH'1991*. ACM, NY, 329–337.
16. E.M. Taranta, II and J.J. LaViola, Jr. 2015. Penny Pincher: A Blazing Fast, Highly Accurate $-family Recognizer. In *Proc. of GI '15*, 195–202.
17. R.-D. Vatavu, L. Anthony, and J.O. Wobbrock. 2012. Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes. In *Proc. of ICMI' 12*.
18. J.O. Wobbrock, A.D. Wilson, and Y. Li. 2007. Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proc. of UIST' 07*. ACM, New York, NY, USA, 159–168.