Using Confusion Reject to Improve (User and) System (Cross-)Learning of Gesture Commands

Manuel Bouillon, Peiyu Li, Eric Anquetil, Grégoire Richard

Using Confusion Reject to Improve (User and) System (Cross-)Learning of Gesture Commands

Manuel BOUILLON, PeiYu LI, Eric ANQUETIL, Grégoire RICHARD
Université Européenne de Bretagne, France
INSA de Rennes, Avenue des Buttes de Coesmes, F-35043 Rennes
IRISA, CNRS UMR 6074, Campus de Beaulieu, F-35042 Rennes
{manuel.bouillon, pei-yu.li, eric.anquetil, gregoire.richard}@irisa.fr

***Abstract*—This paper presents a new method to help users defining personalized gesture commands (on pen-based devices) that maximize recognition performance from the classifier. The use of gesture commands give rise to a cross-learning situation where the user has to learn and memorize the command gestures and the classifier has to learn and recognize drawn gestures. The classification task associated with the use of customized gesture commands is complex because the classifier only has very few samples per class to start learning from. We thus need an evolving recognition system that can start from scratch or very few data samples and that will learn incrementally to achieve good performance after some using time. Our objective is to make the user aware of the recognizer difficulties during the definition of commands, by detecting confusion among gesture classes, in order to help him define a gesture set that yield good recognition performance from the beginning. To detect confusing classes we apply confusion reject principles to our evolving recognizer, which is based on a first order fuzzy inference system. A realistic experiment has been made on 55 persons to validate our confusion detection technique, and it shows that our method leads to a significant improvement of the classifier recognition performance.**

I. INTRODUCTION

With the increasing use of touch sensitive screens, human-computer interactions are evolving. New interaction methods have been invented to take advantage of the new potential of interaction that those new interfaces offer. Among them, a new concept has recently appeared: to associate commands to gestures. Gesture commands [1][2] enable users to execute various actions simply by drawing symbols. Previous studies [3] have shown that enabling customization is essential to help users' memorization of gestures. To use such gesture commands, a handwritten gesture recognition system is required. Moreover, if gestures are personalized, the classifier has to be flexible and able to learn with few data samples.

As we can't expect users to draw much more than a few gesture samples per class, the recognition engine must be able to learn with very few data. Some template matching classifiers exist, like the $1 classifier [4] for instance, that don't require much training. However, such simple systems have limited performances, and don't evolve with the user writing style. For example, novice users usually draw gestures slowly and carefully, but as they become more and more expert, users draw their gestures more fluidly and rapidly. In that case, we want the classifier to adapt to the user, and not the other way round. More flexibility in a recognizer requires an online system, a system that learns on the runtime data flow.

Evolving classification systems have appeared in the last decade to meet the need for recognizers that work in changing environments. They use incremental learning to adapt to the data flow and cope with class adding (or removal) at runtime. This work uses such an evolving recognizer, *Evolve* [5], which is a first order fuzzy inference system. It can start learning from few data and then learns incrementally – in real time – from the data flow that it tries to recognize, to adapt its model and to improve its performance during its use.

Such gesture commands use give rise to a cross-learning situation where the user has to learn and memorize the gestures and the classifier has to learn and recognize drawn gestures. Enabling customization of the gesture commands is essential for users' memorization. On the other hand, enabling users to choose their own gestures may lead to commands with similar or strange gestures that are hard to recognize by the classifier. In order to facilitate this cross-learning situation, we developed a conflicting class detection mechanism to help users during the definition step. Our goal is to highlight similar classes that the recognition engine will tend to confuse and not recognize well. It allows the user to change his gestures if he has define similar classes (which happens as we have seen during experimentation), or to draw more gesture samples until classes are no longer confused. This conflict detection mechanism uses confusion reject principles [6][7] to detect conflicting classes.

This paper is organized as follows. The Section II will present the architecture of our evolving classifier. We explain in section III how our conflict detection mechanism works. Then, we present a realistic experimentation of gesture commands showing the benefits of our approach in Section IV. Section V concludes and discusses future work.

II. SYSTEM ARCHITECTURE

We focus here on Fuzzy Inference Systems (FIS) [8], with first order conclusion structure [9]. FIS have demonstrated their good performances for incremental classification of changing data flows [10]. Moreover, they can easily be trained online – in real time – and have a good behavior with new classes. In this section, we present the architecture of the evolving FIS *Evolve* [5] that we use to recognize our gesture commands.

Fuzzy Inference Systems consist of a set of fuzzy inference rules like the following rule example.

$$\textbf{Rule}^{(i)}: \ \textbf{IF } \mathbf{x} \text{ is close to } C^{(i)} \ \textbf{THEN } \hat{\mathbf{y}}^{(i)} = (\hat{\mathbf{y}}_1^{(i)}; \ldots; \hat{\mathbf{y}}_c^{(i)})^\top \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^n$ is the feature vector, $C^{(i)}$ the fuzzy prototype associated to the *i*-th rule and $\hat{\mathbf{y}}^{(i)\top} \in \mathbb{R}^c$ the output vector. Rule premises are the fuzzy membership to rule prototypes, which are clusters in the input space. Rule conclusions are fuzzy membership to all classes, that are combined to produce the system output.

A. Premise Structure

Our model uses rotated hyper-elliptical prototypes that are each defined by a center $\boldsymbol{\mu}^{(i)} \in \mathbb{R}^n$ and a covariance matrix $\Sigma^{(i)} \in \mathbb{R}^{n \times n}$ (where n is the number of features).

The activation degree $\alpha^{(i)}(\mathbf{x})$ of each fuzzy prototype is computed using the multivariate normal distribution.

B. Conclusion Structure

In a first order FIS, rule conclusions are linear functions of the input:

$$\hat{\mathbf{y}}^{(i)\top} = (l_1^{(i)}(\mathbf{x}) ; \ldots ; l_c^{(i)}(\mathbf{x})) \tag{2}$$

$$l_k^{(i)}(\mathbf{x}) = \mathbf{x}^\top \cdot \boldsymbol{\theta}_k^{(i)} = \theta_{0,k}^{(i)} + \theta_{1,k}^{(i)} \cdot x_1 + \cdots + \theta_{n,k}^{(i)} \cdot x_n \tag{3}$$

The *i*-th rule conclusion can be reformulated as:

$$\hat{\mathbf{y}}^{(i)\top} = \mathbf{x}^\top \cdot \Theta^{(i)} \tag{4}$$

with $\Theta^{(i)} \in \mathbb{R}^{n \times c}$ the matrix of the linear functions coefficients of the *i*-th rule:

$$\Theta^{(i)} = (\boldsymbol{\theta}_1^{(i)} ; \ldots ; \boldsymbol{\theta}_c^{(i)}) \tag{5}$$

C. Inference Process

The inference process consists of three steps:

1) Activation degree is computed for every rule and then normalized as follows:

$$\alpha^{(i)}(\mathbf{x}) = \frac{\alpha^{(i)}(\mathbf{x})}{\sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x})} \tag{6}$$

where r is the number of rules.

2) Rules outputs are computed using Equation 4 and system output is obtained by sum-product inference:

$$\hat{\mathbf{y}} = \sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x}) \cdot \hat{\mathbf{y}}^{(k)} \tag{7}$$

3) Predicted class is the one corresponding to the highest output:

$$\text{class}(\mathbf{x}) = \arg\max_{k=1}^{c}(\hat{y}_k) \tag{8}$$

D. Incremental Learning Process

Let $\mathbf{x}_i$ $(i = 1..t)$ be the *i*-th data sample, M_i the model at time i, and f the learning algorithm. The incremental learning process can be defined as follows:

$$M_i = f(M_{i-1}, \mathbf{x}_i) \tag{9}$$

whereas a batch learning process would be:

$$M_i = f(\mathbf{x}_1, \ldots, \mathbf{x}_i) \tag{10}$$

In our recognizer *Evolve* [5], both rule premises and conclusions are incrementally adapted:

1) Rule prototypes are statistically updated to model the runtime data:

$$\boldsymbol{\mu}_t^{(i)} = \frac{(t-1) \cdot \boldsymbol{\mu}_{t-1}^{(i)} + \mathbf{x}_t}{t} \tag{11}$$

$$\Sigma_t^{(i)} = \frac{(t-1) \cdot \Sigma_{t-1}^{(i)} + (\mathbf{x}_t - \boldsymbol{\mu}_{t-1}^{(i)})(\mathbf{x}_t - \boldsymbol{\mu}_t^{(i)})}{t} \tag{12}$$

2) Rule conclusions parameters are optimized on the data flow, using the Recursive Least Squares (RLS) algorithm:

$$\Theta_t^{(i)} = \Theta_{t-1}^{(i)} + \alpha^{(i)} \mathrm{C}_t^{(i)} \mathbf{x}_t (\mathbf{y}_t^\top - \mathbf{x}_t^T \Theta_{t-1}^{(i)}) \tag{13}$$

$$\mathrm{C}_t^{(i)} = \mathrm{C}_{t-1}^{(i)} - \frac{\mathrm{C}_{t-1}^{(i)} \mathbf{x}_t \mathbf{x}_t^\top \mathrm{C}_{t-1}^{(i)}}{\frac{1}{\alpha^{(i)}} + \mathbf{x}_t^\top \mathrm{C}_{t-1}^{(i)} \mathbf{x}_t} \tag{14}$$

New rules, with their associated prototypes and conclusions, are created by the incremental clustering method *eClustering* [11] when needed.

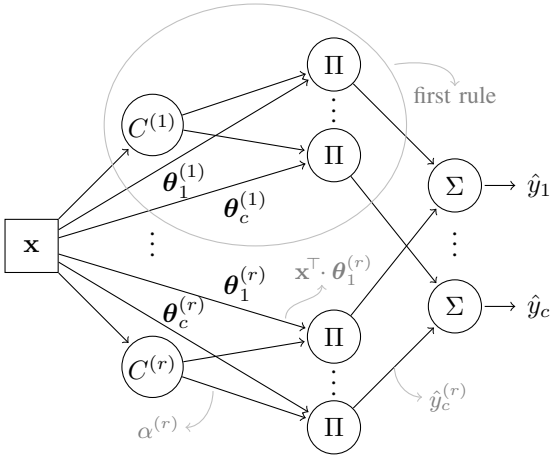

Fig. 1. First order FIS as a radial basis function (RBF) neural network

Figure 1 represents a FIS with first order conclusion structure as a radial basis function (RBF) neural network.

III. Conflict Detection Method

Enabling users to choose their own gestures means that we have very few data samples per class, as users won't be inclined to draw many gestures to initialize the system. Moreover, this may lead to commands with similar gestures that are hard to recognize for the classifier. Our goal is to automatically highlight such classes (see Fig. 2) to allow the user to change his gestures or to draw more samples until every class is well recognized.

To detect conflicting classes, we use confusion reject principles [6]. Normally confusion reject is done using system output [7]. However, we try to detect confusion, to evaluate our model quality, at a very early stage of the learning process. As a result, inference rules conclusions are still unstable and are not suitable for confusion detection. Instead, we detect

confusion on the membership to the different prototypes, which are much more stable at this early stage. Even if every prototype participates in the recognition process of every class, each prototype has been created and is mainly associated with a single class. We use that fact to detect confusion when some gesture activates multiple prototypes.

To detect confusing classes, we define a confidence measure, and an associated confidence threshold. We flag a class as confusing when the confidence of its last gesture is below the threshold (when having learned on all the previous samples).

A. Confidence Measure

We use the Mahalanobis distance to compute the distance of a data sample $\mathbf{x}$ to prototype $C^{(i)}$ (defined by a center $\boldsymbol{\mu}^{(i)}$ and covariance matrix $\Sigma^{(i)}$).

$$distance(C^{(i)}, \mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top (\Sigma^{(i)})^{-1} (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top \quad (15)$$

From this distance, we compute a similarity measure that is smoother than prototype activation.

$$similarity(C^{(i)}, \mathbf{x}) = \frac{1}{1 + distance(C^{(i)}, \mathbf{x})} \quad (16)$$

With this similarity measure, we compute the similarity of a data sample to each prototype and take s_{first} and s_{second} as the first and the second highest similarity value. We compute system confidence as:

$$confidence = \frac{s_{first} - s_{second}}{s_{first}} \quad (17)$$

A data sample is then signaled as confusing when its confidence is below a certain threshold.

B. Threshold Selection

The optimization of the threshold below which we signal gestures as confusing is multi-objective. One wants to maximize both classifier performance and accuracy:

$$Performance = N_{Correct}/N_{Total} \quad (18)$$

$$Accuracy = N_{Correct}/(N_{Correct} + N_{Errors}) \quad (19)$$

where $N_{Correct}$ is the number of correctly classified gestures, N_{Errors} is the number of incorrectly classified gestures, and $N_{Correct}$ is the total number of gestures. As the threshold increases, the number of rejected gestures raises and the number classification errors reduces (see Fig. 3). A high threshold will yield many rejections, which will increase system accuracy, whereas a low threshold will yield only a few rejections, which will increase system performance. There is a tradeoff between the classifier performance and accuracy.

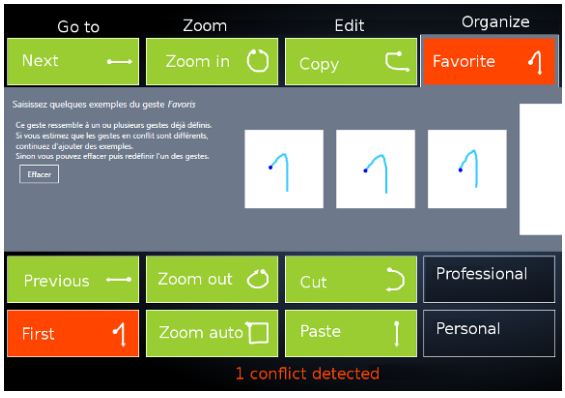

Fig. 2. Gesture commands definition step with conflict detection

To solve this tradeoff, we must define the cost of an error of classification, and the cost of a rejection. During command definition step, a rejection will make the user change the command gesture (or draw more samples). During the utilization of gesture commands, an error of classification will force the user to correct the system. Our goal when detecting conflicting classes is to reduce classification errors during system use. However, we don't want to signal too many classes as conflicting, which would risk discouraging users from using our system.

Finally, without further insights on users' feelings, we chose an identical cost for rejection and misclassification. With those rejection and misclassification costs, the threshold can easily be found experimentally by optimizing the sum of both objectives, weighted by those costs.

IV. Experimentation

This section is dedicated to gesture commands experimentation, and to the evaluation of the impact of conflict detection.

A. Experimental Protocol

We evaluated our system in a real context of gesture commands utilization, so that we obtain realistic results from users as well as from the recognizer. To do so, we designed a testing application: a picture viewer/editor with customizable gesture commands. This application has eighteen basic commands, grouped into six families, to manipulate pictures (like "next", "zoom in", "copy", etc.). We then designed a protocol that simulates a real use of this application to evaluate the impact of confusion detection during definition step.

Our protocol was divided into four phases: an initialization phase, a first evaluation phase, a utilization phase and a second evaluation phase. During each phase (apart from the initialization phase), users were asked to do several commands one at the time. Each time a gesture was drawn, the asked command was executed if the associated gesture was recognized, and the next command was asked. When some gesture was not recognized as the one corresponding to the asked command, a pop-up would ask if it was the user or the recognizer that was mistaken. Except for user mistakes, drawn gestures were used to incrementally train the recognizer to improve its performance.

a) Initialization phase: Users were asked to choose a gesture for each of the eighteen commands of the testing application (see Fig. 2); and to repeat their gestures a few times to provide some initial training samples for the recognizer.

b) First evaluation phase (test1): During this phase, users were asked each of the eighteen commands once in random order.

c) Utilization phase: This phase simulates a potential real use of our testing application. A total of twenty-four commands were asked in random order, some commands were asked thrice, some twice, some once and some were not asked. For this utilization phase, a help menu [12] was available to let users improve their memorization of their gesture commands.

d) Second evaluation phase (test2): During this phase, users were again asked each of the eighteen commands once in random order.

During each test, three rates were measured. First, the recognition rate of the classifier (*reco*). Second, the memorization rate of the user (*memo*), which is the percentage of good answers from the user when he is asked to do a command. Third, the cross-learning rate (*cross*), which is the percentage of good answers from both the user and the classifier at the same time.

We made a first group of user (*Group1*) do this experimentation without detecting confusion during the definition of gestures. Users from this group were not alerted when they used confusing gestures. We made a second group of user (*Group2*) do this experimentation using our conflict detection method during the definition step.

B. Threshold Selection

To choose the confidence threshold for the second group, we used initialization gestures of the first group (without conflict detection) as a tuning dataset. For each user of the first group, we trained our classifier with the two first initialization samples and used the third one as a test sample.

We then computed classifier error rate, rejection rate, performance and reliability for threshold values from 0.01 to 0.25. The results are plotted in Fig. 3.

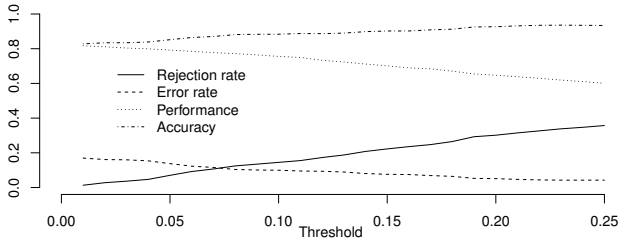

Fig. 3. Rejection rate, error rate, performance and reliability varying with threshold value

With an equal cost of rejection and misclassification, the optimum threshold corresponds to the maximum of the sum of classifier performance and accuracy. This optimum is reached with a threshold of 0.07, which gives a rejection rate of 12.45% and an error rate of 10.40%. With this threshold, system performance is 77.16% and system accuracy is 88.13%.

Furthermore, during the gesture commands definition step, we have access to the true label of each gesture. Instead of only using the confidence measure, we take advantage of gestures true labels to signal misclassified gestures as confusing.

However, to limit the number of conflicting classes, we chose to lower the confidence threshold to 0.05, which gave a rejection rate of 20.79%. On average, one gesture over five is flagged as conflicting/badly recognized.

C. Recognition, Memorization and Cross-Learning Improvements

We had 55 persons, from 21 to 54-year-old, participating in our test and everyone of them was used to manipulate computers in daily life. *Group1* contains 39 users while *Group2* contains 16 users.

We first separated *Group1* into two subgroups by applying the rejection threshold on the third initialization gestures after learning on the two first ones. Users who defined two or less confusing gestures are considered as *Group1+*, and the rest, who made at least three confusing gestures, are considered as *Group1–*. We have 27 users in *Group1+* and 12 users in *Group1–*.

The histograms of the three rates (*cross*, *memo* and *reco*) of the three groups *Group1–*, *Group1+* and *Group2* for the two evaluation phases *test1* and *test2* are shown in Fig. 4. Performances for *test1* are quite limited since the recognizer has only three gestures per class to learn from and since memorizing 18 gestures isn't an easy task for users. However, every rate of each of the three groups increases from *test1* to *test2*, which means that both users and the classifier do effectively learn during the utilization phase.

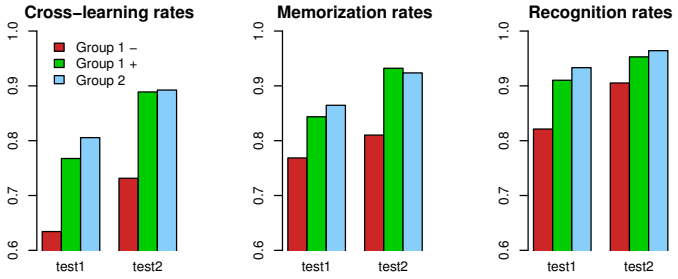

Fig. 4. Cross-learning, memorization and recognition mean rates

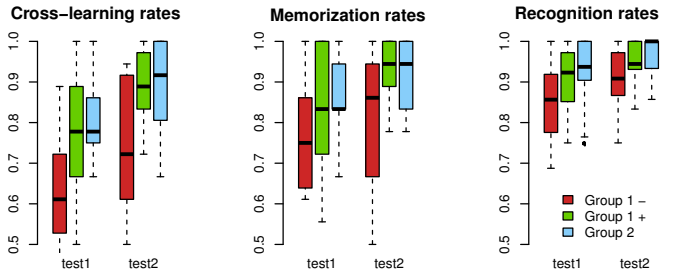

Fig. 5. Cross-learning, memorization and recognition rates distribution

Our hypothesis is that by automatically signaling confusing gestures during the definition step (*Group2*), we should obtain similar performances as users who made few confusing gestures (*Group1+*); and that these two groups should have better performances than the group of users who made many confusing gestures (*Group1–*).

The three histograms in Fig. 4 (*cross*, *memo* and *reco* rates) tend to support our hypothesis. However, it is not enough to claim our hypothesis true! As shown by the boxplots in Fig. 5, we have a lot of variation within each group and we must check that the differences between groups are not just due to chance.

To check our hypothesis, we made some statistical tests to validate the significance of our results.

The results of the pairwise tests comparing *Group1+* to *Group1–* and *Group2* to *Group1–* are shown in Table I and Table II respectively. As we can see, *Group1+* is significantly better than *Group1–* for almost all measures for the two tests, apart from the memorization (*memo*) and recognition (*reco*) rates of *test1* where the variance is too high to make any robust conclusion (the difference of the mean values could be due to chance). Similarly, *Group2* is significantly better than *Group1–* for all measures apart from the memorization (*memo*) rate of *test2* where again there is no statistical difference.

TABLE I. GROUP1+ (G1+) COMPARED TO GROUP1- (G1-)

Rate	Statistical Test Result	Significance
cross - test1	χ^2 = 5.5687, df = 1, p = 0.01828	G1+ > G1- Significant
cross - test2	χ^2 = 7.8593, df = 1, p = 0.00505	G1+ > G1- Very sign.
memo - test1	nothing	Not significant
memo - test2	χ^2 = 5.0743, df = 1, p = 0.02428	G1+ > G1- Significant
reco - test1	F(1, 13.133) = 3.6463, p = 0.07	G1+ > G1- Hardly sign.
reco - test2	χ^2 = 3.8429, df = 1, p = 0.04996	G1+ > G1- Slightly sign.

TABLE II. GROUP2 (G2) COMPARED TO GROUP1- (G1-)

Rate	Statistical Test Result	Significance
cross - test1	χ^2 = 8.4392, df = 1, p = 0.00367	G2 > G1- Very sign.
cross - test2	χ^2 = 6.908, df = 1, p = 0.008581	G2 > G1- Very sign.
memo - test1	χ^2 = 4.0617, df = 1, p = 0.04387	G2 > G1- Slightly sign.
memo - test2	χ^2 = 3.2602, df = 1, p = 0.07098	G2 > G1- Hardly sign.
reco - test1	F(1,15.644) = 5.2093, p = 0.03682	G2 > G1- Significant
reco - test2	χ^2 = 4.8478, df = 1, p = 0.02768	G2 > G1- Significant

From these statistical tests, we can conclude that *Group2* and *Group1+* obtained better performances than *Group1–*, in term of cross-learning and recognition rates. More data would be needed to reduce the variance of the results and possibly show the significance of *memo - test1*. In one word, our hypothesis is verified: users who make many confusing gestures obtain worse performance from the recognizer than users who make few confusing gestures. More importantly, we are able to improve recognizer performance (11.20% of improvement from *Group1–* to *Group2* in *test1*) by helping users not to choose confusing gestures during the definition step.

V. CONCLUSION

We have presented a new method for designing a recognition engine to define personalized gesture commands on pen-based devices. Such gesture commands require an evolving classifier that can learn from very few data samples and learn incrementally. We have explained how our recognizer *Evolve–* a first order fuzzy inference system – is able to detect conflicts among gesture classes by using confusion reject principles. This conflict detection method allows helping users during the definition of gesture commands by making them aware of the recognizer difficulties. Detecting confusing gestures during command definition step is essential to allow the user to personalize gesture commands without deteriorating the recognition performance of the classifier.

We have presented a complete and realistic experimentation of gesture commands and we have studied the impact of our conflict detection method. Gesture commands use give rise to a cross-learning situation where users have to learn and memorize command gestures and the classifier has to learn and recognize drawn gestures. In particular, we have shown that making users aware of the recognition engine allow a statistically significant improvement of the recognition engine performances. This memorization rate seems to increase with the help of conflict detection, by looking at average group results. Nevertheless, we can't make a significant statistical conclusion because variance is very high.

Some differences found during the experimentation were hardly or not significant due to the high variance of the data. It would be interesting to collect more data to see if there really are no differences or if we just have not enough data to make any robust conclusion. Some additional work should be done on users' feelings on the rejection/misclassification tradeoff to improve the rejection threshold. It could also be interesting to use conflict detection during gesture commands use.

REFERENCES

[1] J. O. Wobbrock, M. R. Morris, and A. D. Wilson, "User-defined gestures for surface computing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 1083–1092.

[2] J. Yang, J. Xu, M. Li, D. Zhang, and C. Wang, "A real-time command system based on hand gesture recognition," in *2011 Seventh International Conference on Natural Computation (ICNC)*, vol. 3, 2011, pp. 1588–1592.

[3] P. Y. Li, N. Renau-Ferrer, E. Anquetil, and E. Jamet, "Semi-customizable gestural commands approach and its evaluation," in *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2012, pp. 473–478.

[4] J. O. Wobbrock, A. D. Wilson, and Y. Li, "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*, ser. UIST '07. New York, NY, USA: ACM, 2007, pp. 159–168.

[5] A. Almaksour and E. Anquetil, "Improving premise structure in evolving takagi-sugeno neuro-fuzzy classifiers," *Evolving Systems*, vol. 2, pp. 25–33, 2011.

[6] C. Chow, "On optimum recognition error and reject tradeoff," *IEEE Transactions on Information Theory*, vol. 16, no. 1, pp. 41–46, 1970.

[7] H. Ishibuchi and T. Nakshima, "Fuzzy classification with reject options by fuzzy if-then rules," in *The 1998 IEEE International Conference on Fuzzy Systems Proceedings*, vol. 2, 1998, pp. 1452–1457.

[8] E. Lughofer, *Evolving fuzzy models: incremental learning, interpretability, and stability issues, applications*. VDM Verlag Dr. Mller, 2008.

[9] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man, and Cybernetics, IEEE Transactions on*, vol. 15, no. 1, pp. 116–132, 1985.

[10] P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 6, pp. 1462 –1475, 2008.

[11] P. Angelov and D. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 484–498, 2004.

[12] P. Li, A. Delaye, and E. Anquetil, "Evaluation of continuous marking menus for learning cursive pen-based commands," in *Proceedings of the 15th International Graphonomics Society Conference (IGS)*, 2011, pp. 217–220.