# Graphical Gesture Commands' learning with the help of Customizable Gesture Menus

Peiyu Li, Eric Anquetil

# Graphical Gesture Commands' learning with the help of Customizable Gesture Menus

**PeiYu LI** and **Eric ANQUETIL**
*INSA de Rennes/IRISA*
*Campus de Beaulieu, 263 avenue du Général Leclerc – Bât 12*
*Rennes 35042, France*
{pei-yu.li, eric.anquetil}@irisa.fr.

**Abstract.** The interactive tablets (pen-based or touch screen devices) offer users a new way to run commands by drawing gestures. This induces intuitive interaction but the learning of all gestures for user can be tedious when the set of commands is large. Our objectives are to allow user to draw natural handwritten gestures and to help him learn these gestures quickly. After studying some existing methods, we strongly believe that the personalization capability of gestures is really important for user. In this way, we present in this paper *Customizable Gesture Menus* associated with an evolving gesture recognition engine that can learn incrementally, starting from few data samples. The paper is focused on these menus that allow a simple and compact presentation of personalized gestures. Two comparative tests with 59 users are reported in this paper to evaluate this new concept.

## 1. Introduction

Interactive tablets (interacted with a pen or a finger) can facilitate the interaction between human and computers. Commands can be associated to a gesture (or some gestures) to be run. It is easy to remember all gestures for a simple application with less than 10 commands. Once there are too many possible commands, it becomes difficult to remember them all directly (Kurtenbach, 1993). This is one of the challenges of research on Human Computer Interaction nowadays.

### 1.1. State of the art

Main approaches of the help on gestures' learning are based on *Marking Menus* (Kurtenbach & al., 1994). These approaches helped users to memorize gestures by making them practice drawing (by following the menu). *Marking Menus* propose two ways of utilization: *novice mode* where the user has menus displayed to help him finalize his gesture and *expert mode* where he only needs to draw the gesture and a recognizer will try to understand which command is invoked. First enhancement of the *Marking Menus* is their hierarchical version (Kurtenbach & al., 1993). They allow multiple levels of menus so that commands can be organized by family and more commands' gestures can be presented. Based on *Hierarchical Marking Menus*, more menus appeared like *Zone and Polygon Menus* (Zhao & al., 2006). Evidently, the final form of gestures depends strongly on the menus' ergonomics. For user's familiarization and learning, this is a big constraint.

Another interesting example is *Octopocus* (Bau & al., 2008). It is a one level menu. And it permits predefined natural fluid gestures. In novice mode, it gave a continuous feedback. It highlights the currently recognized command, and weakens gradually the others. This approach helps users' learning and it has a good accuracy. However in case of a large set of gestures, gestures will be superposed and it will be illegible for users, while losing the semantic organization of commands.

### 1.2. Objectives

Our study here is about how to obtain natural cursive gestures and how to design a framework which helps users learn them efficiently. Memorizing more than 15 gestures can become a tedious task for a user. We can identify several ways to improve the learning process of the user to memorize all the gestures. The first point we have identified in our previous contributions, is that it is easier for users to memorize gestures if they have defined the set of gestures themselves (personalization) (Li & al., 2011; Renau-Ferrer & al., 2012). Secondly, a "meaningful" gesture for a command can be memorized more easily than an arbitrary gesture without meaning. A meaningful gesture should have a semantic meaning or an ergonomic meaning. So we are interested in the way to define meaningful gestures, the way to help users memorize these gestures, and also the recognition system behind which can guarantee a good accuracy. This paper focuses principally on the two first points. For the last point, we use the auto-evolving recognition system which has been described in (Almaksour & al., 2010; Almaksour & al., 2011).

In this paper, firstly we presented *Customizable Gesture Menus* which combine the advantages of *marking menus* with a strong capability to be *Customizable* to give user better help, from gestures' definition to utilization. At the moment of gestures' definition, user can choose any mono-stroke gesture as he wants. During utilization, if he finds a new way to draw gesture more easily for him, thanks to the incremental recognizer, he can modify the gesture as he wants and the recognizer continue to learn (next section for more details). Then we

report an experiment and analyze its result. For the experiment, we made an interactive gestural framework where the recognizer system evolves (improve its models) at the same time as the user learns the graphical commands. This process happens while the application itself is in use. Some perspectives for further research are also discussed at the end of the paper.

## 2. Customizable Gesture Menus

### 2.1. Ergonomics of Menus

*Customizable Gesture Menus* approach leave user the freedom to define a gesture for each command (mono-stroke). Since the gestures are personalized, the construction of menus is much more complex. It means the construction should be dynamic, and guarantee a good presentation of all gestures at the same time. These gestures are presented in a circle (similar to *Wave Menus*' form (Bailly, 2007)). Strokes are smoothed by *Canonical Spline* to give a clearer appearance to gestures (Petzold, 2009). The place where each gesture is put on the circle is not random. It depends on the initial direction of user's gesture. For example, a gesture with an initial direction from left to right will be placed on the right part of the circle. The exact position is decided by the initial angle.

Once the application detects that user's stylus is moving very slowly or not moving at all, it will be considered as novice mode. The menus will be displayed. The center of the circle is an inactive zone (empty). Without raising the stylus, user can go through all icons of gestures and the corresponding command's label will be displayed, as well as the gesture at bigger size. We decided to show label just near the circle, unlike other menus which show the label at the end of gesture. Because it makes it easier for the user to see the right command as he does not need to search too far to know if it is the right one, especially in case of long gestures. Once user finds the command that he is looking for, he just needs to follow the gesture's shape (this drawing can be approximate), he can see a continuous feedback presented in menu where by filling by white color the stroke spline during his moving. As he finishes the stroke the command will be validated and executed [Figure 1].

An advantage of all multi-levels menus (hierarchical marking menus) is that they organize commands by family, so final gestures of commands share a semantic meaning among them. As mentioned before, *Customizable Gesture Menus* present commands one by one, so the commands from the same family may not be found side by side. For example, command "next page" and command "previous page" belong to family "page manipulation". And unconsciously, we associate a line from left to right and a line from right to left as gesture for these two commands. Since they do not have the same initial direction, they will not be placed side by side. So when user is looking his gesture for "next page", he will neither think about, nor learn at the same time, his gesture for "previous page". Our solution to conserve the semantic meaning of gestures is when user's stylus goes on one command; the other commands of the family are highlighted too. In this way, even they are not side-by-side; they are still displayed at the same time [Figure 1].

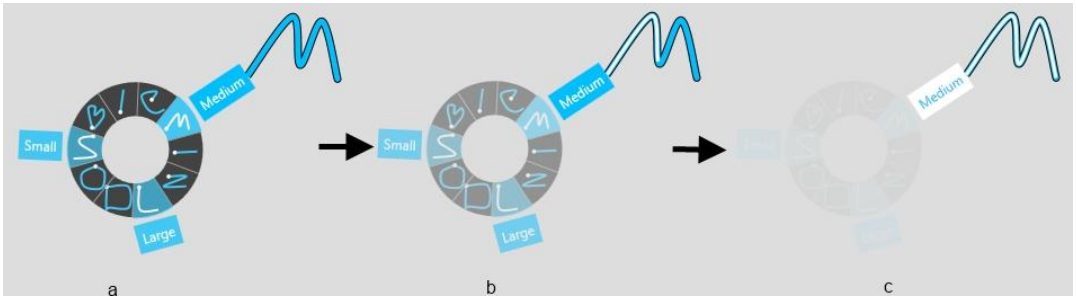



**Figure 1.** Illustration of Continuous Gesture Menus.

### 2.2. Evolving Gesture Recognition Engine

Unlike other menus' research, we also tried to improve our evolving recognizer. Since we allow all kind of strokes from one user, we need a robust recognizer that can learn incrementally, starting from few data samples. Moreover, user change continuously his way of drawing during the utilization: its drawing is more and more accurate going from novice to expert. Our evolving recognition engine is based on first order Takagi-Sugeno (TS) fuzzy inference system. It consists of a set of fuzzy rules, and each rule has an influence zone to which an adaptive linear filter is associated. The incremental learning algorithm of consists of three different tasks: the creation of new rules, the adaptation of the existing rule's influence zones, and the tuning of the linear parameters. These three tasks are done in an online incremental mode and all the needed calculation are completely recursive. More details about Evolve are available in (Almaksour & al., 2010; Almaksour & al., 2011). The recognizer is initialized with two examples of each class and is then continuously trained after each new example. We combine this recognition engine with the universal set of 49 features call "*HBF49"* for the gesture representation (Delaye & al., 2013).

## 3. Evaluation

### 3.1. Experimental Protocol

For this experimentation, 59 persons passed the tests. They are from 22-year-old to 42-year-old. Among these testers, there are 16 women and 43 men. They have all habit to use computers. These people were separated in 2 groups. Group 1 (19 persons) defined all their gestures and the help was presented by a simple table of gestures (cf. Table 1(b)). Group 2 (40 persons) defined also all their gestures but they were helped with *Customizable Gesture Menus* (cf. Figure 1 and Table 1(a)). Everyone passed 2 evaluations of memorization. One was at the beginning of test, i.e. just after the definition of gestures. One was at the end of test, i.e. they already passed a learning phase to practice gestures. So we could obtain a learning curve of all users in the end.

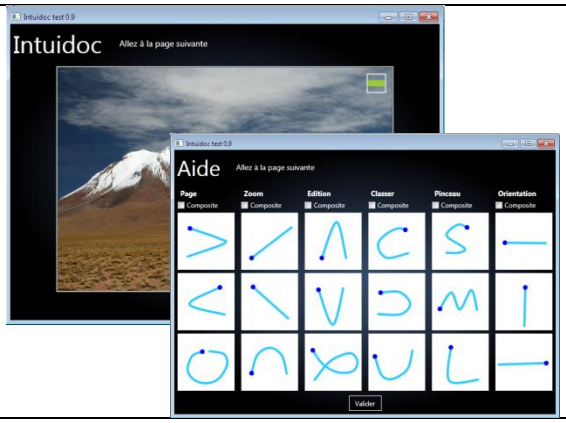
We use 18 commands (separated into 6 families). During learning phase, in each family, there was one command which was asked 3 times; one command which was asked once and another command which was not asked at all. This is to simulate a real application where there are commands used more often while the others not.

The process of tests is the same for the 2 tests. Tests are constructed by 4 phases:

- *Initialization phase*. User could just fill cases of the 18 commands to define gestures. Then for each group, user needed to repeat his gesture once so that the recognizer got another sample to be initialized.
- *First evaluation phase*. User would not have help access (neither "Gesture menu" nor "table of gestures"). But once he made an error, a pop-up message would be displayed. It showed user the correct gesture and the recognized gesture, and it was up to user to tell the recognizer if it was his fault or recognizer's fault. So in the first evaluation, user and recognizer could still continue to learn.
- *Learning phase*. This phase included several sequences of questions. Each question concerned one command. Users of Group1 and Group2 could have help access from a classic table (cf. Table 1 (b)) of all gestures or with our *Customizable Gesture Menus* (cf. Table 1 (a)) respectively. In this phase, in case of error, user would have the table of gestures or Customizable Gesture Menus displayed in a pop-up respectively. We showed him the recognized gesture at the same time. It was up to user to tell us if it was him who drew a wrong gesture or if it was the recognizer who gave the wrong answer. If it was user's fault, the same question would be asked again to practice the user. If it was recognizer's fault, the misunderstood gesture would be used to improve the recognizer. And user moved on to the next question. So in case of error, both recognizer and user could still learn via the pop-up windows.
- *Second evaluation phase*. The same as the first one, but no pop-up would be displayed in case of error because the learning for user and recognizer stopped already.

**Table 1.** Learning phase of two different tests.


(a) Learning phase with help of Customizable Gesture Menus




(b) Learning phase with help of table of gestures. The table displays in front of test interface when user asks for help.

### 3.2. Experimental Results

First of all, we can report the appreciation of menus for users from Group2. In a scale of 10, users gave 7.27 for the ergonomic of the menu. So, users really appreciated the help in form of *Customizable Gesture Menus*.

Secondly, we analyzed users' learning. Few users (28%) succeeded to remember 17 or all gestures. This proves that even if gestures are all personalized, few people are able to learn them all by heart directly. While in the second evaluation phase, up to 51% users learnt 17 or all gestures. The memorization rates of the users for each group are presented at left part of barplots in Figure 2. In the general point of view, both groups increased in their learning. First we can observe that the gradient of learning is quite the same for the two groups (7% of progress for group1 and 8% for group2 (p-value $< 0.001$)). There is no significant difference between the 2

groups (p-value > 0.05). This experiment shows that *Customizable Gesture Menus* does not disturb user's learning of gestures in comparison with classical tabular presentation. But *Customizable Gesture Menus* allow a more compact presentation of all gestures that is a key point to deal with the relative small size of tablet screen (cf. Table 1).

In the right part of barplots in Figure 2, we can see that there is a significant improvement between the two evaluation phases (9% of progress for group1 and 7% for group2 (p-value < 0.001)) of the recognition rates of the evolving recognition engine for both groups. But there is no significant difference of improvement between the two groups (p-value > 0.05). This experiment demonstrates that both user and recognition engine can improve, in the same time, their capabilities of learning respectively: how to draw the gesture and how to recognize the gesture.

This experiment demonstrates that the *Customizable Gesture Menus* offer a good and compact ergonomic that do not disturb the learning of the user and the recognizer engine.

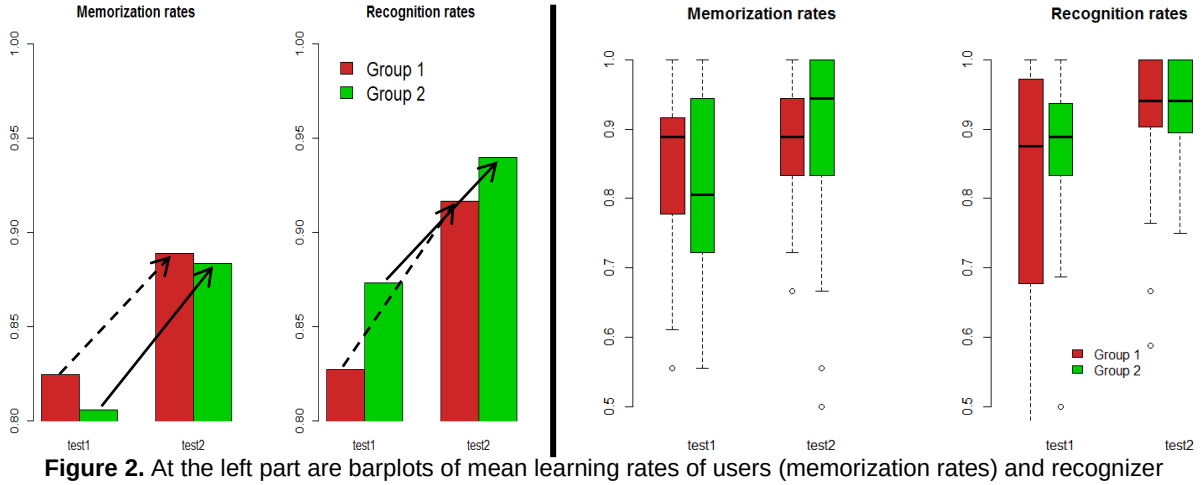



**Figure 2.** At the left part are barplots of mean learning rates of users (memorization rates) and recognizer (recognition rates). At the right part are the corresponding boxplots (sample minimum, lower quartile, median, upper quartile, sample maximum).

## 4. Conclusion

In this paper, we presented *Customizable Gesture Menus*, which are based on a compact version of the Marking Menus, and that allow personalised gestures. The compactness of these menus is well adapted to tablet devices by keeping the same capacity to help the user to learn gestural commands. Moreover we have presented a real cooperative interaction between a user and an evolving recognition engine able to learn from each other for improving their capabilities of respectively, reproduce and recognize, graphical gestures.

## References


Kurtenbach, G., Buxton, W. The limits of expert performance using hierarchic marking menus. *Proc. INTERACT 1993*, ACM Press (1993), 482–487.

Kurtenbach, G., Moran, T.P., Buxton, W., 1994. Contextual animation of gestural commands. Computer Graphics Forum 13 (5), 305–314.

Zhao, S., Agrawala, M. and Hinckley, K. Zone and Polygon Menus: Using Relative Position to Increase the Breadth of Multi-stroke Marking Menus. *Proc. CHI 2006*, ACM Press (2006), 1077-1086.

Bau, O., Mackay, W.E. OctoPocus: a dynamic guide for learning gesture-based command sets. *Proc. UIST 2008*, ACM Press (2008), 37–46.

Delaye, A. and Anquetil, E. "Hbf49 feature set: A first unified baseline for online symbol recognition" Pattern Recognition, vol. 46, no. 1, pp. 117 – 130, January 2013.

Li, P., Delaye, A. and Anquetil, E. Evaluation of Continuous Marking Menus for Learning Cursive Pen-based Commands. *Proc. IGS 2011*, 217-220.

Renau-Ferrer, N., Li, P., Delaye, A. and Anquetil, E. The ILGDB database of realistic pen-based gestural commands. *Proc. ICPR 2012.*

Bailly, G., Lecolinet, E. and Nigay, L. Wave Menus: Improving the Novice Mode of Hierarchical Marking Menus. *Proc. INTERACT 2007.* Springer (2007), 475—488.

Almaksour, A., Anquetil, E., Quiniou, S. and Cheriet, M. Evolving Fuzzy Classifier: Application to Incremental Learning of Handwritten Gesture recognition Systems. *Proc. ICPR 2010.*

Abdullah Almaksour, Eric Anquetil. Improving premise structure in evolving Takagi-Sugeno neuro-fuzzy classifiers. Evolving Systems, 10.1007/s12530-011-9027-0, 2:25-33, 2011.

Petzold, C. "Canonical Splines in WPF and Silverlight." http://www.charlespetzold.com/blog/2009/01/Canonical-Splines-in-WPF-and-Silverlight.html.