Evaluation of an On-line Adaptive Gesture Interface with Command Prediction

Xiang Cao, Ravin Balakrishnan

Department of Computer Science
University of Toronto
caox | ravin @dgp.toronto.edu

Abstract

We present an evaluation of a hybrid gesture interface framework that combines on-line adaptive gesture recognition with a command predictor. Machine learning techniques enable on-line adaptation to differences in users' input patterns when making gestures, and exploit regularities in command sequences to improve recognition performance. A prototype using 2D single-stroke gestures was implemented with a minimally intrusive user interface for on-line re-training. Results of a controlled user experiment show that the hybrid adaptive system significantly improved overall gesture recognition performance, and reduced users' need to practice making the gestures before achieving good results.

Key words: gestures, adaptive interfaces, user studies

1 Introduction

Given that humans use gestures in everyday life to facilitate communication, gestural interaction is likely to be an important component of future multimodal interfaces. Indeed, gestural interaction has been the focus of significant research activity (e.g., [3, 8, 17, 18, 28]), and has begun to appear in commercial products such as pen based PDA's and tablet computers. However, it remains a challenge to develop a robust gesture recognizer that can accommodate different users. A key challenge is that different users will input slightly different stroke patterns for the same intended gesture. As the example in Figure 1 illustrates, the varying input patterns observed when the same gesture is performed by different users (within-class difference) may overrun the varying input patterns observed when different gestures are performed by the same user (between-class difference). The recognizer has to balance between generalizing to different users and distinguishing between different gestures by the same user. In practice, this is a difficult balance to achieve and is a major cause of unsatisfactory gesture recognition.

A popular solution to this problem [17, 24] is to train the recognizer with samples collected from the specific user before the real use of the interface. These systems improve upon predefined recognizers (hard-coded or trained using samples form a few users). The explicit training stage, however, precludes "walk up and use" interaction. Also, if recognition performance is unsatisfactory, or if the user changes his/her input patterns during use, then the user has to explicitly return to the training stage to modify the recognizer, resulting in inefficient and less fluid interaction.

Another solution aimed at alleviating the recognition problem and the requirement for explicit training and re-training phases in gesture interfaces, is to capture and adapt to variations between and within users on-line during use. Such adaptation has shown promising results in applications such as information filtering [7, 9, 15] and handwriting recognition [5, 6, 20].

While there has been substantial work on adaptive recognizers, to the best of our knowledge there has been relatively little published on the impact of an adaptive recognizer combined with a command predictor on recognition performance. We attempt to address this shortfall by conducting a controlled user study that evaluates such a hybrid combination in terms of recognition rate and user performance time. We use standard on-line machine learning techniques to adapt to users' input patterns when entering different gestures in regular use of the system, and to exploit regularities that often exist in command sequences for particular tasks. When a recognition error happens, an on-line interface allows the user to correct the recognition, and the recognizer is re-trained based on this feedback. Thus, the system adapts to the user continually, and results in a user-specific recognizer without requiring an explicit pre-use training phase per user. Although our interface for error correction is new, the main contribution of our work is the systematic evaluation of this hybrid adaptation plus command prediction approach to gesture recognition.

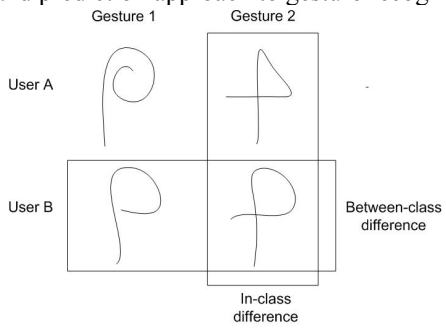


Figure 1. Between-class and within-class differences in gesture entry. Gestures 1 and 2 entered by user B are very similar, while Gesture 2 entered by users A and B are quite different, making overall recognition difficult.

2 Related Work

Gesture interaction has been explored by many researchers. To list a few, Segen et al' [18, 19] GestureVR system uses vision tracking of hand gestures for spatial interaction. Nishi et al. [14] and Wu et al. [28] describe gestural interaction on desks/tabletops. Cao et al. [3] describe a gesture interface using a wand to interact with large displays. Wilson et al. describe the XWand [26] and WorldCursor [25] systems, which control electrical devices with simple gestures using a wand. Rubine [17] describes GRANDMA, a toolkit for building pen-based single-stroke gestural interfaces.

In some systems [17, 24] models are learned in a "watch and learn" framework, requiring the user to demonstrate the gestures in a training phase before the real system use. On-line adaptive interfaces construct the user model during real system use [11], typically using machine learning techniques that take on-line user behavior as training data. Wilson [23] describes additional research beyond [24] that enables the system to capture novel gestures on-the-fly. Pazzani et al. [15], Lang [10], and Boone [1] describe systems for information filtering, aimed at directing the user's attention towards items from a large set that he/she is likely to find interesting or useful. Adaptive frameworks have also been used in handwriting recognition [5, 6, 20]. Researchers have also demonstrated that adaptive interfaces can provide benefits over non-adaptive ones [2, 20, 22], but this has not been done for gesture interfaces in combination with command prediction. Concerns have also been raised [18] that adaptation may make the interface inconsistent and unpredictable over time, thus impairing performance For example, McGrenere et al. [13] showed that if the user has too little control in an adaptive interface, it may be less efficient than an interface manually customized by the user.

3 Hybrid Gesture Interface Framework

In this section we present the framework of the hybrid recognition and command prediction engine behind our prototype system. This framework is designed for general gesture interface systems, independent of the particular application, input dimension or implementation.

The engine takes information from the input device (gesture) as well as from the application program which is using the gesture recognition engine, and outputs the resulting interpreted command to the application.

The engine consists of four relatively independent modules: gesture recognizer, gesture-command interpreter, command predictor, and decision-maker (Figure 2). The first three modules use different input information to provide intermediate results which are then fed to the decision-maker. Both the gesture recognizer and the command predictor continuously adapt to the user.

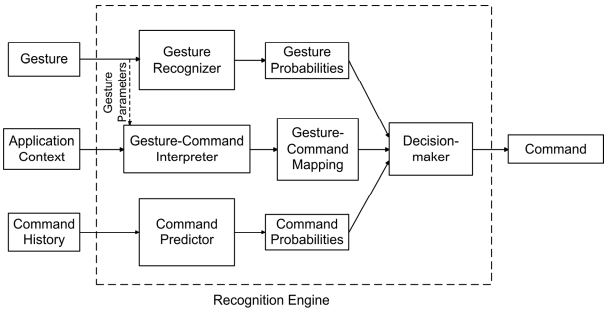


Figure 2. Recognition Engine

3.1 Adaptive Gesture Recognizer

Depending on application demands, the user's gestures could be captured through different input devices. The captured data is fed into the gesture recognizer which calculates the probabilities for each possible gesture that matches the current observation: $P(\mathbf{O}|G_1)$, $P(\mathbf{O}|G_2)$, …, $P(\mathbf{O}|G_n)$, $P(\mathbf{O}|G_0)$ where $\mathbf{O}$ is the observation, $G_1 \ldots G_n$ are all the possible gestures, and G_0 represents "not a gesture" (i.e. the user's unintentional input). These probabilities are then fed into the decision-maker.

Each time a gesture is recognized, there are five different possible outcomes depending on the user's intention and the system's recognition result (Figure 3). Depending on the outcome, the gesture recognizer will take the input data as a new training sample for different gestures, and then re-train itself. Both positive and negative samples are used for re-training. For the 2D single-stroke gesture input in our prototype application (to be discussed later), the gesture recognizer is implemented using Hidden Markov Models (HMM) [16].

Outcome	Description	User's intention	Recognition result	Re-training sample
1	Correct	G_x	G_x	positive sample for G_x
2	Wrong	G_x	G_y	positive sample for G_x negative sample for G_y
3	Confused	G_x	O	positive sample for G_x
4	False alert	O	G_x	negative sample for G_x positive sample for G_0
5	Idle	O	O	None

Figure 3. Re-training rules for gesture recognizer depending on outcome. G_x and G_y refer to different gestures, and G_0 refers to "not a gesture". In the "user's intention" column, O means the user did not intend to enter a gesture but provided input accidentally. In the "recognition result" column, O means the system could not map the input data to any recognizable gesture.

3.2 Gesture-Command Interpreter

The gesture-command interpreter (Figure 4) provides a mapping of all possible gestures into commands for the application, based on the current state and context of

the application program, and (sometimes) the parameters of the gesture (location, scale, duration, etc.). The same gesture may be interpreted as different commands depending on the application context. This gesture-command mapping is fed into the decision-maker. Our system accepts 2D single-stroke input, and uses the location (geometric center) of the gesture and application context to compute the gesture-command mapping.

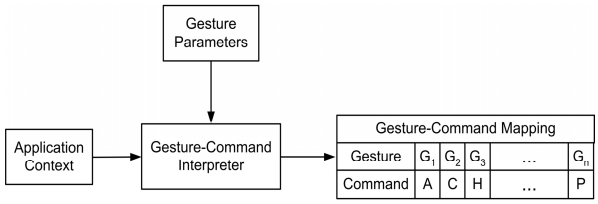


Figure 4. Gesture-command interpreter

3.3 Command Predictor

In addition to the input gestural data and the application context, another valuable information channel is the history of interaction activity. Depending on the application and the particular user, regularities typically exist in the sequence of commands entered to complete tasks. Therefore, the next command the user is likely to execute is partially predictable by the previous command history. For example, the probability of doing "paste" becomes much higher when "copy" or "cut" is executed recently. Including this historical data could improve the reliability of the gesture recognition engine.

Given the current command context T consisting of the previously executed command sequence, the probabilities of all possible commands to be executed next, $P(C_1|T)$, $P(C_2|T)$, …, $P(C_m|T)$ where C_1 … C_m are all possible commands, are calculated and fed into the decision-maker. Each time a command is executed, the prediction model is updated, allowing it to improve over time. Note that the command predictor and gesture recognizer work independently of each other. In our prototype we implement the command predictor based on a popular text compression algorithm called "prediction by partial match (PPM)" [4, 21] which exploits statistics of the characters that follow identical strings.

3.4 Decision-Maker

The decision-maker incorporates information from the gesture recognizer, the command predictor and the gesture-command interpreter, and generates the final recognition result. Given $P(\mathbf{O}|G_1)$, $P(\mathbf{O}|G_2)$, …, $P(\mathbf{O}|G_n)$, $P(\mathbf{O}|G_0)$ from the gesture recognizer, $P(C_1|\mathbf{T})$, $P(C_2|\mathbf{T})$, …, $P(C_m|\mathbf{T})$ from the command predictor, and the mapping from the gesture-command interpreter, we multiply the probabilities of corresponding gestures and commands, and choose the gesture with largest product to be the final recognition result. To enable gesture rejection, the corresponding command possibility for $P(G_0|\mathbf{T})$ is set to a constant, and the other command probabilities normalized. When the product for G_0 becomes the largest among all the products, or all $P(\mathbf{O}|G_1)$, $P(\mathbf{O}|G_2)$, …, $P(\mathbf{O}|G_n)$ give probabilities below a pre-set threshold, G_0 is chosen as the result, and the recognizer concludes that the input is not a gesture.

4 Prototype System

In order to experiment with and evaluate the hybrid gesture interface framework, we developed a test-bed application based on a "picture shopping scenario" that has sufficient functionality to exercise the various parts of the framework, but at the same time is simple enough for users to quickly understand within the constraints of user studies lasting at most a few hours. In this application, the task is to pick pictures from the "shelf" on the left for placement into the "shopping cart" on the right. The user can browse the pictures, zoom in to see details, query detailed information, move pictures into or out of the cart, and so on. All the commands are triggered by making 2D stroke gestures. The location (geometric center) of the gesture determines which object the command is intended for. Figure 5 shows an application screenshot, in which a select gesture has been made atop the middle picture in the bottom row. We defined seven gestures, illustrated in Figure 6. Depending on context, these gestures can be interpreted as eleven different commands.

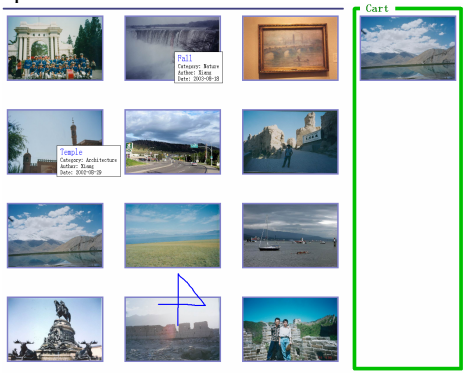


Figure 5. Example "picture shopping" application

4.1 Input

Because of the ubiquity of 2D single-stroke gestures, we chose it as the input modality for our implementation. A pen on a Wacom® tablet is used to input single-stroke 2D gestures to a standard desktop computer. A stroke is recorded from the moment the pen tip touches the tablet to the moment it is lifted. The captured data is fed into the recognition engine. The only data pre-filtering we do is to ignore strokes too short in either time (< 0.2secs) or space (within a 10x10 pixel region).

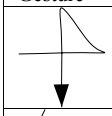
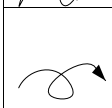
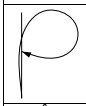
Gesture	Name	Corresponding Commands
	Select	Select Picture: Select a picture to zoom in on it. (on a picture) Select Sort: Select a sort criteria and perform the sorting (on a sort button)
	Query	Query: Display the information sheet of a picture (on a picture) The info sheet includes title, author, date and category.
	Dismiss	Dismiss Picture: Dismiss the zoomed-in picture (on a zoomed-in picture) Dismiss Info: Dismiss the information sheet of a picture (on an info sheet). Dismiss Sort: Dismiss the sort buttons (on a sort button)
	Sort	Sort: Display sort buttons (any context). There are 4 buttons corresponding to 4 sort criteria: title, author, date and category. The date is sorted from earliest to latest, and the other three are sorted in alphabetical order. Selecting one of the buttons, sorts the pictures accordingly.
	Pick	Pick into cart: Place a picture into the cart (on a picture in the shelf) Pick out of cart: Remove a picture from the cart (on a picture in the cart)
	Up	Scroll up: Scroll up the shelf by two lines (any context)
	Down	Scroll down: Scroll down the shelf by two lines (any context)

Figure 6. Gestures and associated commands for test application. Text within brackets explains the context.

4.2 Interface for On-Line Re-Training

The recognition engine adapts to the user's input patterns by taking feedback from the user about the recognition result. We have designed an interface for collecting this user feedback, with the explicit goal of keeping the intrusiveness of the interface to a minimum relative to the task the user is trying to perform. Note that Mankoff et al. [12] describe techniques for ambiguity resolution in recognition systems, with a similar goal of low intrusiveness, but do not deal with gesture recognition per se nor use the pie menu interface we propose.

Our system recognizes the input gesture every time the pen-tip is lifted off the tablet. If the system comes to a recognition result other than "not a gesture", the corresponding command is executed immediately. In this case, three sub-cases can occur:

1. The gesture is recognized correctly. The corresponding command is executed, and the user continues to make the next gesture. The recognizer is re-trained using the current gesture as a positive sample.
2. The gesture is misrecognized as another gesture.
3. The stroke is an unintentional input by the user, but recognized as a gesture.

In cases 2 and 3, the user knows there is an error by seeing a wrong command being executed, or no response at all. The user thus needs to tell the system an error happened. By double-tapping the pen-tip on the tablet, a correction pie-menu is triggered (Figure 7a). This menu includes items for all possible gestures and an "undo" item. The captured gesture stroke is also displayed for the user's reference. In case 2, the user selects the intended gesture by clicking on the relevant menu item. The incorrect response is undone, and the command corresponding to the intended gesture is executed. The recognizer is then re-trained by taking the stroke as a positive sample for the intended gesture, and a negative sample for the misrecognized one. In case 3, the user selects "undo", and the incorrect response is undone. The recognizer is re-trained by taking the stroke as a negative sample for the misrecognized gesture, and a positive one for the "not a gesture" model.

If the system determines the captured stroke to be "not a gesture", i.e. it fails to recognize the gesture, another pie menu automatically pops up to ask for the actually intended gesture. This "fail to recognize" menu (Figure 7b) is almost identical to the correction menu in Figure 7a, except that the "undo" menu item is replaced with a "cancel" item. If the user intends the stroke to be a meaningful gesture, he/she selects the intended gesture item. The corresponding command is executed, and the recognizer is re-trained by taking the stroke as a positive sample for the intended gesture. If the stroke was made unintentionally, i.e. system made the correct decision, the user either selects "cancel" to dismiss the menu, or simply ignores it and makes the next gesture.

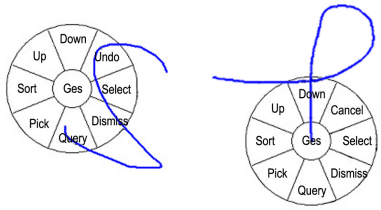


Figure 7. (a) Correction menu (b) "Fail to recognize" menu

5 Experiment

5.1 Goals

We compared the performance of our hybrid adaptive recognition system to a non-adaptive one, for novice users, using the "picture shopping application". Since our engine has two independent adaptive modules – the gesture recognizer and the command predictor – we can test their influence on performance separately.

5.2 Software

The software is implemented in C++. To support the needs of a controlled experiment, the gesture recognizer in the recognition engine can be switched between

adaptive and non-adaptive modes. In adaptive mode, the recognizer works as described previously. In non-adaptive mode, it provides the same interfaces for the users to correct recognition errors, but does not re-train itself. Similarly, the command predictor can be turned on or off. When off, the command context is ignored. The user interface is unaffected by the mode of either the gesture recognizer or command predictor. The basic recognizer was empirically tested to ensure that it had reasonable recognition performance, thus avoiding bias in favor of adaptation due to a poor base-line.

5.3 Participants

12 right-handed university students (9 male, 3 female) volunteered. None had prior experience with pen input.

5.4 Procedure

In each session, a participant was given a different group of 24 pictures with a maximum "shopping cart" size of 4, and asked to use the gestural commands to complete the following five tasks:

Task 1: Place into the cart all the pictures with exactly one person in view. If more than 4 pictures meet the requirement, chose the 4 with the earliest dates.

Task 2: Place into the cart all the pictures with author "X". If there are more than 4 pictures that meet the requirement, choose the 4 with the earliest dates.

Task 3: Place into the cart all the pictures with person "Y" in view and where the title consists of exactly one word. If there are more than 4 pictures that meet the requirements, choose any 4.

Task 4: Place into the cart the 4 pictures with the largest number of people in view. If there are less than 4 pictures with people in view, take all of them.

Task 5: Place your favorite 4 pictures into the cart.

Here "X" refers to a specific name, and "Y" refers to a specific person, depending on the group of pictures. The design goal of these tasks was to oblige the users to actively use all the commands provided, while at the same time exposing their own patterns of command sequences when performing the various tasks.

At the start of the experiment, the application was demonstrated, and all commands explained in detail. Participants read the written task descriptions, and were given one practice session in which they completed the tasks for one group of pictures, using a standard pop-up menu to invoke the commands, instead of the gestures. The goal here was to familiarize the participants with the tasks, independent of the gestures.

After the practice session, the gestures were demonstrated, and a sheet illustrating all gestures provided for reference. Participants were not given time to practice the gestures, since our goal was to study system performance when used by a novice. Participants were told that the correction menu was only for correcting system recognition errors. Conceptual errors caused by the user (i.e. giving a wrong command) should be remedied by giving an opposite command (e.g. "Down" for "Up").

Tasks were performed at the users' own pace, and users rested between sessions. For each gesture entered, we recorded the recognition result, the user-intended gesture, and the time spent.

5.5 Design

A two-factor mixed design with between- and within-participant factors was used. The two independent variables were: gesture recognizer adaptive/non-adaptive (noted as G: G1/G0) and command predictor on/off (noted as P: P1/P0). G was a between-participants factor, and P was a within-participants factor with counterbalancing. In our experiment, the main source of skill transfer would be the user's practicing of the gestures when performing the experiment. Intuitively, this practice effect should not be correlated to the command predictor's performance, given that the gesture recognizer and command predictor are working independently. Therefore we hypothesized that the skill transfer was symmetric for P (this hypothesis was later validated by the experiment data), and set P as the within-subject factor. On the other hand, the user's practice effect would be heavily influenced by the adaptation of the gesture recognizer, since we expected that the adaptation of the gesture recognizer would alleviate the user's burden to practice the gestures. Therefore G is set to be the between-subject factor, so as to avoid the influence of possible asymmetric skill transfer.

Participants were randomly assigned to 4 groups of 3:

Group 1: G1; P0 followed by P1
Group 2: G1; P1 followed by P0
Group 3: G0; P0 followed by P1
Group 4: G0; P1 followed by P0

In each group, participants completed 3 consecutive sessions each for P0 and P1. The order of performance of P0 or P1 is determined by the group, and we note this order of P as P_{order}. During the 3 consecutive sessions of P0 or P1, the adaptation effect was carried forward between sessions. The system was re-initialized when participants switched from P0 to P1 or vice versa. All 6 sessions used different groups of pictures for the same tasks. The pictures were grouped before the experiments such that the task difficulties were the same in each session. In summary, the design was as follows:

4 groups [2 gesture recognizer conditions (G1/G0) x
  2 command predictor orderings (P1->P0/P0->P1)] x
3 participants per group x
2 cmd predictor conditions (P1/P0) per participant x
3 sessions per command predictor condition
= 72 sessions in total

5.6 Results

Data Summary

Participants took an average of 6.6 minutes per session. A total of 5960 gestures were made in the 72 sessions, with an average of 82.8 gestures per session.

Recognition Rate

Percentage recognition rate is defined as the number of correctly recognized gestures divided by the number of all gestures in a session. A gesture is counted in outcomes 1, 2, 3, and 4 (Figure 3), and as correctly recognized in outcome 1. The outcome is determined from the system's recognition result and the user's feedback.

Table 1 summarizes the recognition rate results. The adaptive gesture recognizer showed a 20.2% improvement over the non-adaptive gesture recognizer ($F_{1,8}$ = 26.788, p < .001). The standard error for G0 (4.063%) was higher than that for G1 (1.293%), which indicated that without adaptation, system performance varied a lot, due on different user behavior. With adaptation, system performance is more consistent across users.

	G0	G1	G Overall
P0	63.8	85.6	74.7
P1	70.2	88.8	79.5
P Overall	67.0	87.2	77.1

Table 1: Average Recognition Rates (%)

The overall average recognition rate was 74.7% for P0, and 79.5% for P1. The system with command prediction showed a 4.8% improvement over that without command prediction ($F_{1,16}$ = 10.030, p < .01). The standard error for P0 (2.546%) was also higher than that for P1 (1.511%), suggesting that the command predictor also helped to stabilize system performance. In the experiment design, we counterbalanced P as the within-participants factor, by assigning half the participants to P0 before P1, and half to P1 before P0, with the assumption that any skill transfer between the two techniques would be symmetric and balanced out by the counterbalancing (i.e. the experience of using P1 first affects the performance in subsequently using P0 by roughly the same amount as that of the inverse case). The lack of a significant P_{order} x P interaction ($F_{1,16}$ = 1.014, p = .343) validated this assumption.

To compare our fully adaptive system to the fully non-adaptive system, the average recognition rate was 88.8% for G1 & P1, and 63.8% for G0 & P0. The fully adaptive system showed a 25.0% improvement ($F_{1,10}$ = 23.552, p < .001). Again, the standard error of G0 & P0 (5.071%) is much higher than G1 & P1 (0.906%).

The data did not show a significant G x P interaction ($F_{1,16}$ = 1.089, p = .327). This result suggests that the gesture recognizer and the command predictor were not only working independently, but also affected the user performance independently. It also suggests that user's behavior in performing gestures and executing command sequences are not dependent on each other.

There was also a significant main effect for session ($F_{2,16}$ = 30.400, p < .001), with a steady improvement as the experiment progressed. This is due to both system adaptation and participants' practice. Pairwise comparisons (Tukey test) showed that session 1 (70.8%) was significantly different from session 2 (79.8%) (p < .01) and session 3 (80.8%) (p < 0.05), while session 2 and session 3 were not significantly different from each other (p = 0.472). We did not observe a significant P x session interaction ($F_{2,16}$ = 0.276, p = .762). However, there was a significant G x session interaction (($F_{2,16}$ = 9.463, p < 0.05) as illustrated in Figure 8.

For G1, there was a significant effect for session ($F_{2,8}$ = 23.828, p < .001). Session 1 (79.1%) was significantly different from session 2 (91.1%) (p < .01) and session 3 (91.4%) (p < .01), but sessions 2 and 3 were not significantly different (p = .867). This indicates that system adaptation was probably completed in session 2.

For G0, there was also a significant effect for session ($F_{2,8}$ = 8.562, p < .010). In contrast to G1, all 3 sessions were significantly different from each other. Session 1 (62.4%) was significantly different from session 2 (68.4%) (p < .01) and session 3 (70.3%) (p < .05). And session 2 was significantly different from session 3 (p < .05). This suggests the participants were still improving in session 3. Compared with the result for G1, we conclude that system adaptation significantly reduced user practice requirements.

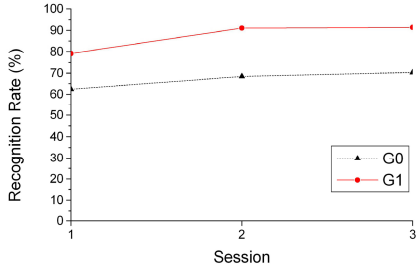


Figure 8. Recognition rates by session and G.

Single Gesture Time

Single gesture time was defined as the time spent from the starting to end points of a single gesture stroke. Although this measure was controlled by the participants rather than the system, we expected that the more reliably the gestures could be recognized, the less time the participants would spend on making each gesture, since they would not have to imitate the pre-defined gestures exactly. We only counted correctly recognized gestures.

Table 2 summarizes the average single gesture times. Overall, participants spent 25.9% less time on a

single gesture with the adaptive system ($F_{1,8} = 53.341$, $p < .001$). This validated our hypothesis that with the adaptive recognizer, participants made the gesture as they saw fit rather than mimicking the system's predefined gesture as in G0, thus enhancing input speed.

	G0	G1	G Overall
P0	1160	853	1006
P1	1130	846	988
P Overall	1145	849	997

Table 2: Average Single Gesture Times (msec)

On the other hand, the command predictor did not have a significant effect on the single gesture time. The averages were 1006 ms for P0, and 988 ms for P1 ($F_{1,16} = 1.170$, $p = .331$). This is interesting compared with the effect of G. Although the command predictor improved recognition performance, the mechanism was invisible to the participants. Therefore, participants did not have an explicit mental model for that, and did not change their behaviors enough to affect gesture time. We did not observe a significant P_{order} x P interaction ($F_{1,16} = 0.047$, $p < .833$), suggesting a symmetric transfer effect. There was also no significant G x P interaction ($F_{1,16} = 0.405$, $p = .542$).

There was a significant main effect for session ($F_{2,16} = 91.641$, $p < .001$), which showed a steady decrease in gesture time through the experiment. Pairwise comparisons showed that the 3 sessions were all significantly different from each other ($p < .001$), with average times of 1096 ms for session 1, 994 ms for session 2, and 901 ms for session 3. This effect incorporates both system adaptation and participants' practice. There was a significant G x session interaction ($F_{2,16} = 7.783$, $p < 0.01$), but no significant P x session interaction ($F_{2,16} = 1.230$, $p = .318$). As Figure 9 illustrates, G1 had a slower rate of decrease, likely because participants were already performing quite fast near the start and thus had less room to improve with practice.

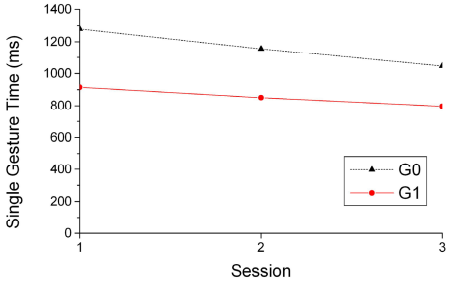


Figure 9. Single gesture time by session and G

Prediction Rate

Although the command predictor is not directly observable by the user, it is still interesting to evaluate how precisely the prediction works. Given the predicted probabilities of gestures $P(G_1|\mathbf{T})$, $P(G_2|\mathbf{T})$, …, $P(G_n|\mathbf{T})$, and the gesture i actually made by the user, the prediction rate is defined as the expectation of $P(G_i|\mathbf{T})$. With a completely uniform prediction (random guess), the prediction rate would be $1/n$. The amount by which the prediction rate is greater than $1/n$ is an indication of how much better the command predictor is working. In our experiment, this expectation was calculated by taking the average of $P(G_i|\mathbf{T})$ for all the gestures made in the session. Obviously the prediction rates were only calculated for P1 conditions. Since we have 7 gestures in the experiment application, the prediction rate for randomly guessing would be 14.3%.

The overall average prediction rate was 30.4%. This was significantly higher than the random guess baseline ($t = 24.346$, $p < .001$). No significant effect for G ($F_{1,10} = 3.827$, $p = .079$), session ($F_{2,20} = 1.095$, $p = .354$), or G x session interaction ($F_{2,16} = 0.993$, $p = .378$) were observed. This indicates that the command predictor was working well, and its performance did not depend on the gesture recognizer or participants' practice time.

6 Discussion

A concern about adaptive interfaces is that the system may become inconsistent and unpredictable due to adaptation, which is detrimental to usability. However, in the design we evaluated, the user has full control of the adaptation process, and thus can better predict the result of the adaptation. The interaction techniques are also consistent over time. The only change the user observes is that the recognition is getting better. Therefore the user does not pay a penalty for the adaptation, beyond providing on-the-fly training data.

Although in the non-adaptive system users had to converge to pre-defined gestures, different behaviours were observed in the adaptive system. Greatest variability was observed in the "pick" and "query" gestures, where not only parameters like tilt or proportion varied, but different topologies were entered (Figure. 10). This variability consolidated our reasoning why an adaptive recognizer would be useful. If the adaptation rate is fast, the user can completely change the definition of a gesture. Since both training and newly collected samples are maintained, the user can also give multiple definitions for the same gesture. This can address the concern that the user may not be self-consistent when making the same gesture each time. In our implementation, the system adaptation rate is such that the recognizer can be trained to accept a completely different definition for one gesture after 2~3 samples are given.

Figure. 10. Variability in input gestures

7 Conclusions and Future Work

We have presented a controlled user experiment to evaluate an on-line gesture interface framework. Results showed significant performance gains, both in recognition rate and in interaction speed. While our experiment provided several revealing results, there are many issues regarding adaptive gesture recognition worthy of further study. It would be interesting to study how different, perhaps higher dof, gestures will perform within an adaptive framework. Also interesting is how the framework generalizes between different users. How well will the command predictor trained by one user work for another user? When the recognizer is retrained by one user, is its ability to recognize other users' gestures improved or impaired? Also, how many gestures can such a system optimally handle? We explored the case where single-stroke gestures are discretely delimited by pen up/down events. It would be interesting to explore multiple-stroke gestures, or the situation where gestures are segmented on-the-fly from continuous input, which occurs with buttonless devices such as laser pointers [27] or passive wands [3].

Acknowledgements

We thank Allan Jepson, Xuming He, DGP lab members, and all who participated in our experiment.

References

1. Boone, G. (1998). Concept features in Re:Agent, an intelligent email agent. *2nd International Conference on Autonomous Agents*. p.141-148.
2. Brusilovsky, P. & Pesin, L. (1998). Adaptive navigation support in educational hypermedia: An evaluation of the ISIS-Tutor. *Journal of Computing and Information Technology*, 6(1). p.27-38.
3. Cao, X. & Balakrishnan, R. (2003). VisionWand: interaction techniques for large displays using a passive wand tracked in 3D. *ACM UIST*. p.173-182.
4. Cleary, J. & Witten, I. (1984). Data compression using adaptive coding and partial string matching. *IEEE Trans. on Comm*, 32(4). p.396-402.
5. Connell, S. & Jain, A. (2002). Writer adaptation for online handwriting recognition. *IEEE Trans. on PAMI*, 24(3). p.329-346.
6. Fu, H., Chang, H., Yeong, Y., & Pao, H. (2000). User adaptve handwriting recognition by self-growing probabilistic decision-based neural networks. *IEEE Trans. on Neural Networks*, 11(6).
7. Hermens, L. & Schlimmer, J. (1993). A machine-learning apprentice for the completion of repetitive forms. *IEEE Conf on AI Applications*. p.164-170.
8. Hinckley, K. (2003). Synchronous gestures for multiple users and computers. *ACM UIST*. p.149-158.
9. Hinkle, D. & Toomey, C. (1994). CLAVIER: Applying case-based reasoning on to composit part fabrication. *6th Innovative App of AI Conf*. p.55-62.
10. Lang, K. (1995). NewsWeeder: Learning to filter news. *Intl. Conf. on Machine Learning*. p.331-339.
11. Langley, P. (1999). User modeling in adaptive interfaces. *Intl. Conf. on User Modeling*. p.357-370.
12. Mankoff, J., Hudson, S., & Abowd, G. (2000). Interaction techniques for ambiguity resolution in recognition based interfaces. *ACM UIST*. p.11-20.
13. McGrenere, J., Baecker, R., & Booth, K. (2002). An evaluation of a multiple interface design solution for bloated software. *ACM CHI*. p.163-170.
14. Nishi, T., Sato, Y., & Koike, H. (2001). SnapLink: Interactive object registration and recognition for augmented desk interface. *Interact*. p.240-246.
15. Pazzani, M., Muramatsu, J., and Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. *Natl. Conference on Artificial Intelligence*. p.54-61.
16. Rabiner, L.R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2). p.257-285.
17. Rubine, D. (1991). Specifying gestures by example. *ACM SIGGRAPH*. p 329-337.
18. Segen, J. & Kumar, S. (1998). Gesture VR: Vision-based 3D hand interface for spatial interaction. *ACM Multimedia*. p.455-464.
19. Segen, J. & Kumar, S. (2000). Look ma, no mouse! *Comm. of the ACM*, 43(7). p.102-109.
20. Vuori, V. (2002). Adaptive Methods for On-Line Recognition of Isolated Handwritten Characters. Ph.D. Thesis. Helsinki University of Technology.
21. Ward, D., Blackwell, A., & MacKay, D. (2000). Dasher - a data entry interface using continuous gestures and language models. *ACM UIST*. p.129 - 137.
22. Weber, G. & Specht, M. (1997). User modeling and adaptive navigation support in WWW-based tutoring systems. *Intl. Conf. User Modeling*. p.289-300.
23. Wilson, A. (2000). Adaptive models for the recognition of human gesture, Ph.D. Thesis. MIT.
24. Wilson, A. & Bobick, A. (2000). Realtime online adaptive gesture recognition. *ICPR'00 International Conference on Pattern Recognition*. p.1270-1275.
25. Wilson, A. & Pham, H. (2003). Pointing in intelligent environments with the World Cursor. *Interact*.
26. Wilson, A. & Shafer, S. (2003). XWand: UI for intelligent spaces. *ACM CHI*. p.545-522.
27. Winograd, T. & Guimbretiere, F. (1999). Visual instruments for an interactive mural. *ACM CHI*. p.234-235.
28. Wu, M. & Balakrishnan, R. (2003). Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. *ACM UIST*. p.193-202.