# User Adaptation to a Faulty Unistroke-Based Text Entry Technique by Switching to an Alternative Gesture Set

Ahmed Sabbir Arif, Wolfgang Stuerzlinger
York University, Toronto, Ontario, Canada

## Abstract

This article presents results of two user studies to investigate user adaptation to a faulty unistroke gesture recognizer of a text entry technique. The intent was to verify the hypothesis that users gradually adapt to a faulty gesture recognition technique's *misrecognition* errors and that this adaptation rate is dependent on how frequently they occur. Results confirmed that users gradually adapt to *misrecognition* errors by replacing the error prone gestures with alternative ones, as available. Also, users adapt to a particular *misrecognition* error faster if it occurs more frequently than others.

**Keywords**: Adaptation; errors; gesture recognition; learning.

**Index Terms**: H.5.2 User Interfaces: Input devices and strategies (e.g., mouse, touchscreen).

## 1 Introduction

Although the origin of the famous quote, "That's not a bug, it's a feature" is unknown, the computer science community is well aware of its implications [13]. It points towards a phenomenon observed in many domains, including human-computer interaction and programming languages. The quote refers to the possibility that practitioners will adapt to a non-fatal system error, if it remains in the system for long enough. Once users get accustomed to a system error they either actively avoid repeating actions that cause the error or start treating it as a feature. Such behaviour can be indirectly explained through theories of learning. Some of these theories assume that learning is a process of replacement, where incorrect response tendencies are replaced with correct ones [26]. Alternative theories describe learning as a process of accumulation, where incorrect response tendencies remain constant and correct response tendencies increase with practice [25]. Regardless of the explanation, both schools of thought imply that it is important to reduce mistakes to learn correct responses. Human errors are well studied and explained in the field of text entry, error research, and cognitive psychology. But how users deal with a faulty gesture recognizers' *misrecognition* errors is not very well studied. Based on observations from the existing literature, we can hypothesize that users' learning rates for system errors depend on how erroneous a particular system is. Also, users learn to avoid an erroneous action faster if it occurs more frequently than others. Yet, no empirical studies have been conducted to investigate this. Thus, we present two user studies that attempt to verify the above-mentioned hypothesis. We believe that a deeper insight into potential user adaptation to a faulty gesture recognizer will provide designers with guidance for future work on such technologies.

* asarif@outlook.com; wolfgang@cse.yorku.ca

We start with a review of gesture-based text entry techniques and the challenges they face. Based on an informal survey, we discuss the common types of errors that occur in such techniques and also how errors are handled by these systems. The use of alternative gestures is reviewed as well. The review also includes systems that allow users to perform commands with gestures. We present the software used in the user studies and elaborate on how it was designed in accordance with current trends in human and system error handling as well as provisions for alternative method usage in gesture-based techniques. Then, we present the results of two user studies that verify that users gradually adapt to *misrecognition* errors and that this adaptation rate depends on how frequently such errors occur. Finally, we conclude with practical implications and future extensions of this work.

## 2 Related Work

Here, we briefly review related work in gesture recognition and other fields.

### 2.1 Unistroke Gesture Recognition

Gesture-based text entry techniques have been widely explored and aim to increase speed and accuracy over free-form handwriting methods, which support natural handwriting [30]. Gesture-based methods trade naturalness with higher recognition accuracy. Almost all gesture-based techniques limit user behaviours by permitting only a single way of drawing each character to avoid segmentation and other handwriting recognition related problems [4]. Also, many gesture-based techniques use simplified sets of characters that are drawn with a single stroke (unistroke).

One such technique, called *Unistrokes*, use a character set designed to be entered in an eyes-free manner on handheld devices with a stylus [12]. As the name suggests, each character is represented by a single stroke mark. Unistroke gestures are only somewhat similar to their printed counterparts and need to be learned [4]. A similar unistroke technique, called *Graffiti*, attempted to reduce the learning effort with strokes that resemble the equivalent printed letters more closely [14]. A later version, *Graffiti 2*, requires some characters, such as *I*, *K*, *T* and *X*, to be drawn with two strokes [16]. A longitudinal study comparing these two techniques did not find any significant difference between them [6]. *Jot System*, a technique very similar to *Graffiti 2*, includes almost all *Graffiti 2* gestures but adds multiple variants of the gestures to accommodate handwriting-like drawing [21]. *EdgeWrite* simplifies the text entry gestures to strokes between the corners of a square [34, [36]. A study showed that it requires learning effort, similar to *Graffiti* [34]. *Minimal Device-independent Text Input Method* (*MDTIM*) also simplifies the gestures to *up*, *down*, *left*, and *right* strokes [14]. Yet, significant practice is required to achieve fast entry speeds, as its gesture set is quite different from the printed counterparts [21].

A word-based technique, called *SHARK*, assigns unistroke gestures to the most frequent words based on users' finger movement pattern on a keyboard [37]. With this method, users effectively draw gestures for known words and tap on the keys for unfamiliar ones. *Swype* is similar and also permits users to enter words as gestures. It uses shape recognition to identify words, as the resulting stroke usually forms a shape that is unique to the intended word. In case the shape matches multiple words, users can select the desired word from a list. Both of these techniques are somewhat similar to the earlier *Cirrin* technique [23]. There, all characters are arranged inside the perimeter of an annulus. To input each character, users have to move the stylus into and out of the appropriate sector of that annulus. *T-Cube* was developed [33] based on marking menus [17]. It is similar to a two-level pie menu system. The main level contains nine submenus and the second level contains eight pie menus, each representing a specific character. To input a character, users first have to select an entry in the main menu. Then, users have to flick the stylus into the direction where the intended character is situated in the second level menu that appears.

*Unipad* augments *Unistrokes* with word prediction and auto-completion [22]. *UniGest*, in contrast, allows users to input text with pointing devices without a display [7].

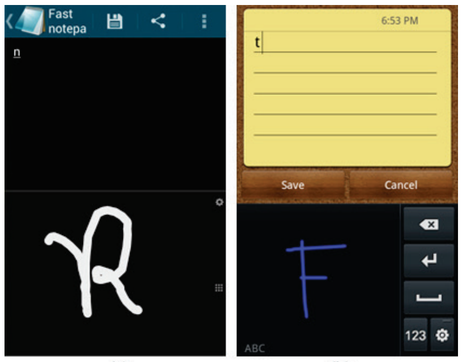
## 2.2 Effort vs. Learning

A theory in psychology research identified the durability of episodic memory as a positive function of the degrees of semantic involvement in processing stimuli [9]. This was verified through empirical studies that showed that deeper encodings take longer to process, but that they also improve performance in tasks such as recall or recognition for words [10]. Similarly, a survey of skill acquisition research argued that manipulations that decrease the speed of acquisition might support long-term learning [28]. Encouraging active information retrieval from memory is a common and effective mechanism for skill acquisition in various domains. Motivated by this, prior work investigated the relationship between user effort and spatial memory in user interfaces. Such explorations revealed that, when interacting with effortful user interfaces, users depend more on memory retrieval than perceptually available information—a characteristic of skilled behaviour [11]. Investigations also showed that interfaces that require greater user effort improve learning for spatial tasks [8], and improve system efficiency and user experience in the long run [27]. In a recent study, Labahn et al. observed that users seemed to adapt to an error-prone recognizer after using it for about half an hour [18]. However, they did not investigate this further.

## 2.3 Errors in Gesture Recognition

Although gesture recognition is technically easier than online handwriting recognition, most gesture recognition techniques still suffer from notable amounts of recognition errors [24, 29]. Moreover, recent developments in gestures sets for pen, finger, and wand user interfaces on different devices have increased the overall gesture ambiguity, which further affects accuracy. A study comparing *Graffiti 2* with a virtual Qwerty keyboard showed that text entry with *Graffiti 2* is substantially slower and more error prone, even when augmented with prefix-based word prediction [16]. Yet, high accuracy is vital for a technique's success. A study showed that users find a gesture-based technique useful only when it is at least 97% accurate [19]. Another user study showed that mobile users usually abandon a gesture-based technique and start using an alternative when accuracy drops below 40% [15].

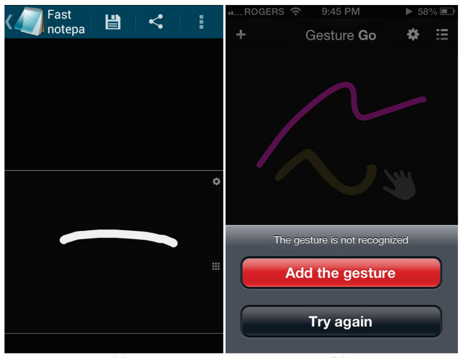
Most gesture recognizers attempt to match a performed gesture to an existing, internal gesture library and return a match score. These libraries contain templates for the supported gestures, often based on the number of strokes, their order, direction, and/or the speed associated with them [30]. When the score is above a predetermined, algorithm-dependent threshold, the system performs the action associated with the gesture that yielded the highest match score. In gesture-based text entry, this action is usually the output of a character. There are two types of errors that occur in most gesture-based techniques: *misrecognitions* and *failures to recognize*.



(a) (b)

Figure 1: Misrecognition in: (a) *Touch-Writer* and (b) *DioPen*. In both cases, the user intended to input one character but the system misrecognized it as another.

A *misrecognition* error occurs when the recognition score is above the predetermined threshold, but the system misinterpreted the performed gesture, and thus, outputted an incorrect letter. A common example is that the user inputs *u*, but the system recognizes and outputs *v*. Or a gesture is too slanted relative to the template. Such errors are usually caused by some internal limitation of the system and are well known to occur in most gesture-based techniques [30]. Research on better gesture recognition algorithms attempts to reduce the potential for these. In an informal survey, we explored the error handling of five popular gesture-based techniques for handheld devices: *Path Input*, *Touch-Writer*, *DioPen*, *Hot Virtual Keyboard*, and *Gesture Go*. The first is similar to *Swype*, the next three are character-based techniques, and the last is an application launcher. Even in a short test, each of these systems misrecognized some performed gestures and output incorrect results. Figure 1 illustrates two such incidents.



The gesture is not recognized

(a) (b)

Figure 2: Error handling in: (a) *Touch-Writer* and (b) *Gesture Go*. In (a), the system displays no output when it fails to find a match for the performed gesture in the library. In (b), it asks the user to include the gesture in the library or to try again.

A *failure to recognize* error occurs when the match score for the user input is below a predetermined threshold or the length of the gesture is too short to be recognized. In our informal survey, human behaviour caused most of such errors. Common examples are the user accidently tapping the screen or aborting a gesture prematurely. Or the user inputs a gesture that is not part of the template library. The surveyed techniques deal with such errors in two different ways. They either do not display output or query the users if they want to include the new gesture in the built-in template library to enrich it. Figure 2 illustrates this.

## 2.4 Alternative Methods/Gestures

Some gesture-based systems permit users to input a given character with several drawing variations. *EdgeWrite* and some commercial products, such as *DioPen* and *Hot Virtual Keyboard*, provide multiple variations for drawing some characters. More relevant for our context, *Jot* permits users to indicate drawing preferences for some characters. In other words, *Jot* enables users to select less intuitive drawing variants as the dominant method for inputting some characters, if the user has problems with the recognition accuracy for those gestures [21]. Here and to distinguish between these variants, we classify the more intuitive ones as "primary" and less intuitive one as "alternative".

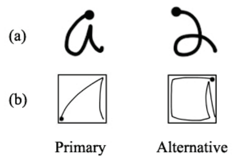


Figure 3: The primary and an alternative method for drawing '*a*' with: (a) *Jot* and (b) *EdgeWrite*.

Alternative gestures are almost always *less intuitive* and *harder to discover* compared to the primary ones. Figure 3 illustrates the primary gesture and one of the alternative ones for inputting the character '*a*' with *Jot* and *EdgeWrite*. Also, one has to either go to the tutorial (with *Jot*, *EdgeWrite*, and *Hot Virtual Keyboard*) or guess (with *DioPen*) to discover such alternative gestures.

## 3 THE EXPERIMENTAL SOFTWARE

Before we describe our user studies, we discuss the custom software we created for them. It was designed according to current trends in human and system error handling and alternative method usage among gesture-based techniques, as discussed above. The software was fine-tuned through several pilots. The design intent was to increase the external validity of this work by making the experiment software reasonably comparable to existing techniques. We also discuss technical design decisions behind the user studies.

### 3.1 Gesture Recognition

The software uses the $1 Unistroke Recognizer [35] to process pen-based gesture input. Similar to geometric template matchers, it recognizes gestures using a nearest-neighbour classifier with a Euclidean scoring function. This recognizer performs well for a limited number of gestures based on very few templates. A study reported 99% accuracy rate for sixteen gestures with three or more templates loaded [35]. Our implementation used fourteen gestures and loaded seven templates for each, which should make the performance of our system equivalent to other recent recognizers in the field.

### 3.2 The Supported Gestures

During the studies, participants inputted seven English letters, specifically *B*, *D*, *O*, *Q*, *R*, *W*, and *Y*. The custom software presented one letter at a time on the screen. Participants then had to input the presented letter with a digital pen on a graphic tablet using either *Graffiti* or *Unistrokes*. The system used *Graffiti* as the *primary* method of inputting the letters, while *Unistrokes* were used as the alternative. That is, users were expected to primarily use *Graffiti* to input the letters, but were permitted to use *Unistrokes* if they felt their use necessary, such as to bypass (injected) *misrecognition* errors. We elaborate on this below.

### 3.3 Unistroke vs. Multistroke Gestures

We used a unistroke gesture system instead of a multistroke one, as the latter systems usually permit different variations for drawing the same letter. This makes it more challenging to recognize a performed gesture. It also makes it difficult to automatically identify human errors due to differences in gesture drawing across users. In addition, due to multiple possible drawing variations for the same letter, users often struggle to identify their mistakes and to discover the right way for drawing a letter with multistroke systems. The $N Recognizer, for example, often fails to correctly recognize a gesture when users use more strokes than the number of strokes used to define said gesture [1]. With adaptive multistroke recognizers, such as Gesture Search [20], it is difficult to isolate the human adaptation rate as the system adapts to human behaviours as well. Unistroke gesture systems usually do not suffer from such problems [30]. A more recent multistroke recognizer, $P Recognizer, resolves these issues [32], but was proposed after the completion of our studies.

### 3.4 Primary vs. Alternative Gestures

*Graffiti* and *Unistrokes* gestures were selected as *primary* and alternative method for drawing the letters for two reasons. First, a previous study did not find a significant difference between these techniques' entry speed, correction rate, and preparation time [6]. Second, *Graffiti* was selected as the *primary* method, as in almost all unistroke-based techniques the primary method is relatively more intuitive and easier to guess than the alternative one, as discussed earlier. In Figure 4, one can see how the primary *Graffiti* gestures look more like their printed counterparts. In addition, participants were encouraged to practice the *primary* gestures before the main studies, to familiarize participants (to a limited degree) with them, as also discussed below. With this experimental design we can assume that any performance effect due to switching the gesture drawing method (from *primary* to alternative and vice versa) mid-study will be attributable *predominantly* to learning.

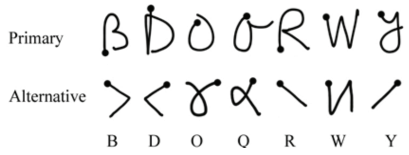


Figure 4: The seven letters and their corresponding gestures. The *primary* gestures are from *Graffiti* and the alternatives are from *Unistrokes*. A dot indicates the start of a stroke.

### 3.5 Discoverability

We discussed earlier how in most gesture-based techniques alternative input methods are relatively harder to discover compared to the *primary* method. In most systems and to discover alternative gestures, one has to either go to an extended tutorial or guess. Based on this, our custom software displayed the *primary* gestures in a panel at all times and presented the to-be-inputted letters in *Graffiti*. To discover the alternative gesture for a particular letter, users had to tap or right-click on the corresponding *primary* gesture in the panel. This displayed the alternative gesture for that letter for two seconds, and then returned to the original state, that is, displaying the *primary* gestures (see Figure 5).

## 3.6 Errors and Error Handling

Similar to other gesture-based systems and based on several pilots, our system reported a *failure to recognize* error when the total number of recorded stroke samples was less than ten in our setup, i.e., when the stroke was much too short for a gesture. This threshold is system specific and needs to be adapted for other setups. Our pilots identified that such short gestures are almost always caused by accidental interactions. Examples include that users tapped on the graphic tablet with the pen, pressed the buttons on the pen by mistake, or stopped drawing prematurely. Similar to many gesture-based techniques, the custom software provided visual feedback on such *failure to recognize* errors. The (previously) inputted gesture field, in the top-right corner of Figure 5, displayed a special symbol (see Figure 6) in case of such accidental interactions.

If the recognized gesture did not match the presented gesture the system classified this as a *misrecognition* error. Similar to almost all gesture-based techniques, the custom software displayed the misrecognized gesture in the inputted gesture field. For example, when *O* was misrecognized as *Q*, the system displayed *Q* in the inputted gesture field. Auditory feedback, in form of a *ding* sound was provided for both *failure to recognize* and *misrecognition* errors.

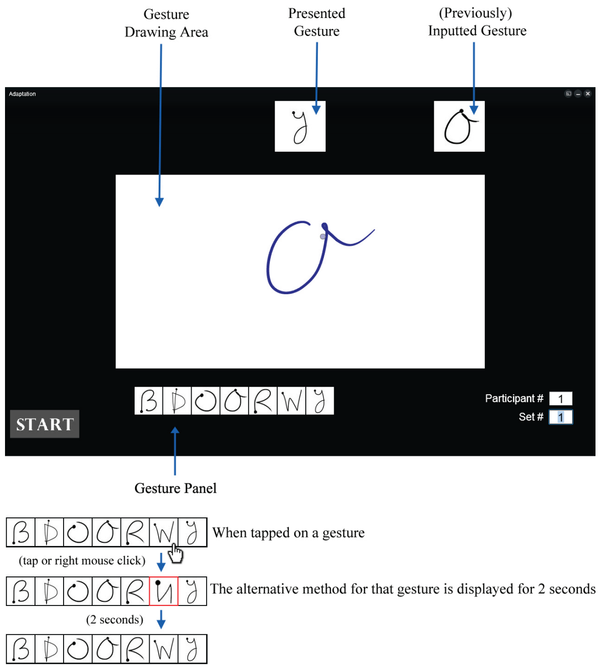

Gesture Drawing Area
Presented Gesture
(Previously) Inputted Gesture
Adaptation
Participant #
1
Set #
1
START
Gesture Panel
When tapped on a gesture
(tap or right mouse click)
The alternative method for that gesture is displayed for 2 seconds
(2 seconds)

Figure 5: The custom software used during the studies. The to-be-inputted letter is presented using the *primary* gesture. To discover the alternative gesture for that letter, one has to tap on the corresponding *primary* gesture in the bottom panel.

### 3.6.1 Raw Recognition Error Rate

In a pilot with eight novice users, four female, average 21 years, all right-handed, where each user inputted the seven *Graffiti* gestures for forty times with the custom software ***without error injection***, the system recorded 1% "system" errors, composed of 0.3% *failure to recognize* and 0.7% *misrecognition* errors. In other words, the overall accuracy rate was 99%, which matches prior work [35].

## 3.7 Misrecognition Error Injection

The main purpose of this work is to investigate (if and) how users adapt to *misrecognition* errors. Thus, a few *primary* gestures were randomly selected during the studies and injected with *synthetic* misrecognition errors at different rates. That is, the system intentionally misrecognized these gestures at given rates. For instance, if the *primary* gesture for the letter *D* was injected with 5% synthetic *misrecognition* error, then five out of hundred times the system would intentionally misrecognize this gesture and would randomly display a similar gesture in the inputted gesture field, such as *B*, *C*, *O*, or *Q*. The system injected *misrecognition* errors instead of *failure to recognize* errors, as *misrecognition* is the most common type of error in gesture-based techniques [24, 29]. ***Only*** the *primary* gestures were injected with these errors**.**

We accounted for any potential bias in simulated gesture recognition errors by randomly selecting a different set of letters for error injection for each participant. Another design constraint for our user studies is that with increasing gesture set size, error occurrences naturally decrease, which makes such errors then progressively harder to study. Consequently, we used only seven letters and tuned the gestures well. As mentioned above, in the absence of injected errors, users encountered only 1% "system" errors. Such a small error rate is well below what can be studies in short-term studies. Consider that 1% errors means that system errors occur only on one out of hundred letters entered. In our studies, participants entered 630 gestures within an hour or more. Thus they would see only 6-7 errors, which is too small to study adaptation.

Figure 6: The symbol displayed in the (previously) inputted gesture field on accidental interactions.

## 3.8 Bypassing Injected Misrecognition Errors

Our pilots indicate that users deal with *misrecognition* errors in two different ways [2]. They either draw a faulty gesture slowly or start using an alternative method (if available). The first approach affects one's entry speed, as it takes more time to input gestures. In contrast and assuming that the alternative method is not more complex than the *primary* one, the second approach does not compromise entry speed. Prior work provides a methodology for classifying gestures into simple, medium, and complex categories [31]. Given this, the latter approach is a better choice for experiments, as entry speed will vary less.

Besides, with only a single "faulty" gesture set (and no alternatives), users are effectively stuck. If they fail to recognize the failure patterns, they either adapt or abandon the system. In many real world situations, they would most probably abandon the system, as there are other ways to achieve their tasks. Consequently, many recent real world systems, see above, include gesture variations (alternative gestures) as a solution this problem. We chose to follow this idea.

To address the issues around speed, our studies ***indirectly*** discouraged participants from drawing gestures slowly. First, users were informed prior to the studies that taking more time to draw a gesture might not enhance the system's recognition rate. Second, and in the practice period prior to the main studies, most users would realize that an inputted gesture does not have to be an exact match of the displayed one for the system to recognize it—so that (subconsciously) they would be much less motivated to draw gestures slowly.

### 3.9 Seven Letters vs. Short English Phrases

Early on, we decided against the use of short English phrases in the studies. Two reasons motivated this. First, using English phrases would require injecting errors based on letter frequencies to maintain uniformity. This needlessly complicates and lengthens the studies. Second, a pilot study indicated that inputting English phrases with an untrustworthy gesture-based system causes a high level of user frustration, which may negatively bias study results.

We also decided against using a complete gesture alphabet. The reason is that users need to experience *enough* injected errors within 60 to 90 minutes to be able to adapt to the system. The use of seven letters assured that each letter appeared a sufficient number of times. This does not invalidate this work, as the focus here is on how users adapt to injected *misrecognition* errors and not (directly) on how text entry performance is affected.

### 3.10 Justification for a Short-term Study

While it is important to understand gradual adaptation over time, short-term usability is today a strong determinant in product success. If users do not see reliable enough performance in the short term, a product is likely to fail. Consequently, long-term investigations are interesting, but do not help in situations where users get frustrated up-front. This is a global issue that gesture recognizers have to contend with today.

### 3.11 Justifications for the Performance Metrics

The following metrics were calculated during the studies.

- **Alternative Method Usage (AMU):** The rate (%) at which the alternative method was used to input letters. As users were free to use either the *primary* or the alternative method to input/re-input a letter, this metric enable us to measure the rate at which users adapted to the alternative gestures.
- **Input Time ($T^{h}_{input}$)**: This represents the average time (in milliseconds) it took to input a letter [3]. This metric captures the performance aspect of learning. We also use this to analyse performance across different *misrecognition* rates.
- **Gestures per Character (GPC):** This denotes how many gestures it took on average to input a letter [34]. As most unistroke methods have dedicated gestures for all English letters, a flawless system will require a GPC of one, providing there was no human error. This was calculated to provide an overall picture of the input process, and to check whether the more faulty letters yield higher GPCs compared to less faulty ones, as one might expect.

## 4 User Study 1

This user study investigated users' adaptation behaviour for injected *misrecognition* error rates between 0 and 30%.

### 4.1 Participants

Twelve participants, aged from 21 to 30 years, average 24.5 (SD = 2.61), participated in the user study. They were recruited through online social communities, local university e-mailing lists, and by posting flyers on campus. None of them had prior experience with pen-based devices. All were unaware of the existence of *Unistrokes* and *Graffiti*. Seven of them were female and one was a left-hand pen user. They all received a small compensation for participating.

### 4.2 Apparatus

The custom application described earlier was used during this study. It was developed with the default *Bamboo Mini* SDK 2.1. The application was running on a 15.4" *Compaq* Presario C700 Notebook PC at 1280×800 pixel resolution. Participants interacted with the application through a digital pen on a *Wacom* Bamboo Pen and Touch Graphic Tablet, as illustrated in Figure 7. The device's 14.73×9.14 cm active area was calibrated with respect to the application window. Its multi-touch input capability was disabled to permit participants to rest their hand on the surface while using the pen. The orientation of the tablet and the default firmware was adjusted to accommodate for left- and right-handedness. The custom application logged all interactions with timestamps and calculated user performance directly.

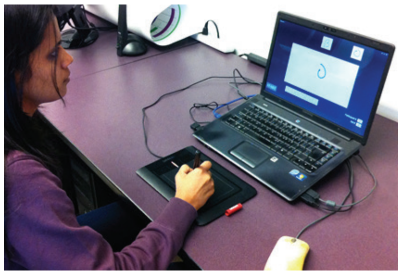


Figure 7: A participant drawing gestures using a digital pen on a Bamboo Pen and Touch Graphic Tablet.

### 4.3 Procedure and Design

The experiment setup and software was first demonstrated to users. The experimenter verified that participants understood the primary (*Graffiti*) and the alternative (*Unistrokes*) gestures, the *failure to recognize* and the *misrecognition* errors, and knew how to discover alternative gestures.

A practice period followed the demonstration. During practice, participants were asked to input the seven letters five times using the *primary* method without error injection. The intent was to familiarize them with the setup. This also gave them some experience with how similar the presented and the performed gestures needed to be for the system to recognize them accurately. Participants were able to extend the practice period (at most twice), as desired.

The main user study started roughly two minutes after the practice. In that part, participants inputted letters in random order and each of the seven letters occurred ninety times. Thus, each participant inputted in total 630 letters. Three out of the seven letters were randomly picked by the system and injected with 10, 20, and respectively 30% *synthetic misrecognition* errors. That is, in ten, twenty, and thirty out of hundred attempts the corresponding letters were intentionally misrecognized by the system. That is, the system displayed a similar letter instead of the recognized one, as discussed above. Only three letters were injected with synthetic *misrecognition* errors, to ensure that the faulty letters do not dominate the overall input process.

The letters were displayed one at a time on the screen. Participants had to input each presented letter using the pen and the graphic tablet using predominantly the *primary* method (*Graffiti*). They were informed that, unlike in the practice period, the system might not be entirely reliable. That is, it may misrecognize some of the letters, even when they were inputted correctly. However, they were not informed about error rates or the number of letters where synthetic *misrecognition* errors were injected.

A gesture was recorded from the moment one touched the graphic tablet with the pen (touch-down) to the moment it was lifted (touch-up). Upon completion of input, the recognized and the next to-be-inputted letters were displayed on the screen automatically, as illustrated in Figure 5. Participants were asked to input the gestures as fast as possible, but to focus more on the accuracy. That is, they were encouraged to reduce the *misrecognition* errors, any way they saw fit, even if it compromised their input speed. They were informed that they ***could*** use the alternative method (*Unistrokes*) to input a frequently misrecognized letter, if they felt that this would improve (or is improving) recognition accuracy. But they were ***neither forced nor instructed*** to use the alternative*s*. Users had to keep inputting a gesture until it was correctly recognized by the system. On correction attempts, no synthetic recognition errors were injected to reduce the potential for overly frustrating tasks. Thus, users who did not want to use alternative*s* could use the *primary* method on correction attempts. Auditory and visual feedback was provided, as described earlier. To minimize interruptions, participants were permitted to take at most two three-minute breaks during the study, as necessary. Given that participants entered 630 letters in the whole session, this gave them enough time to create a good mental model of the system and its errors. After all, each participant the set of faulty letters was constant for each participant. Upon completion of the study, they were asked to fill out a short questionnaire, where they were asked to list the frequently misrecognized letters.

The study used a within-subjects design, where the within-subjects factor focused on the 0, 10, 20, and 30% injected *misrecognition* rates. The dependent variables were GPC, AMU (%), and $T^{h}_{input}$ (milliseconds).

## 4.4 Results

The whole user study lasted from sixty to ninety minutes including the demonstration, practice, and breaks. Upon completion of the study, 59% participants were able to recognize all three error prone letters, 33% had recognized the two most error prone letters, while the remaining 8% recognized only the most error prone one. Thus, about 8% of the users did not adapt to the two less faulty letters where fewer misrecognitions were injected. Consequently, they also did not use the alternative method to input those two letters.

D'Agostino Kurtosis tests on the dependent variables revealed that the data were normally distributed. Also, a Mauchly's test confirmed that the data's covariance matrix was circular in form. Hence, repeated-measures ANOVA was used for all analysis. All statistically significant results are presented with effect size ($\eta^2$).

To identify learning, the data was segmented into blocks of ten appearances of each letter during the study. That is, the average of every ten times a letter was presented to the users to input was used to observe improvements over time. As all letters appeared exactly ninety times per participant, there were nine segments for each letter. Note that we only compared different data points from the same segment to isolate the effect of *misrecognition* rates. As the learning process is gradual [25, 26], users adapt more to a (faulty) system when they spend more time with it. Hence, comparisons between different data points from different segments may be misleading or suffer from bias.

### 4.4.1 Alternative Method Usage (AMU)

An ANOVA on the data revealed that there was a significant effect of injected *misrecognition* rate on AMU ($F_{3,11} = 5.56$, $p < .005$, $\eta^2 = .40$). Average AMU for 0, 10, 20, and 30% injected *misrecognition* rates were 8.5, 31.85, 27.59, and 55.10%, respectively. Figure 9 illustrates this. A Tukey-Kramer test showed that the 30% injected *misrecognition* rate had significantly higher AMU than 0, 10, and 20%.

For all injected *misrecognition* error rates, power functions were fitted to the data to model the power law of practice [5]. This is illustrated in Figure 9, where the horizontal axis represents the segments and the vertical axis represents the average AMU during that segment. Recall that there were four letters where no *misrecognition* errors were injected (0%), compared to one letter for each injected *misrecognition* rates (10, 20, and 30%). Hence, the 0% data points average the AMU of the ten appearances of the four non-faulty letters (10×4 appearances). We also tried fitting linear functions to the data (0%: $R^2 = 0.87334$, 10%: $R^2 = 0.63659$, 20%: $R^2 = 0.95331$, and 30%: $R^2 = 0.51826$), but they did not correlate as well as the power functions (0%: $R^2 = 0.92358$, 10%: $R^2 = 0.84447$, 20%: $R^2 = 0.96004$, and 30%: $R^2 = 0.73612$).

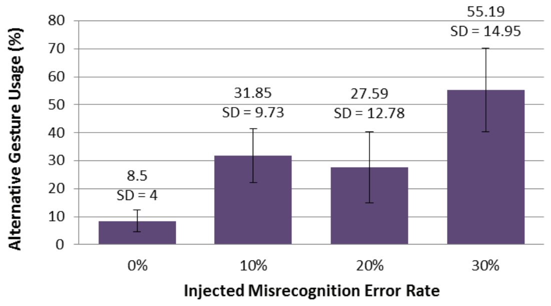


Figure 8: Average Alternative Method Usage (AMU) over all investigated injected misrecognition error rates. Error bars represent ±1 standard deviation.

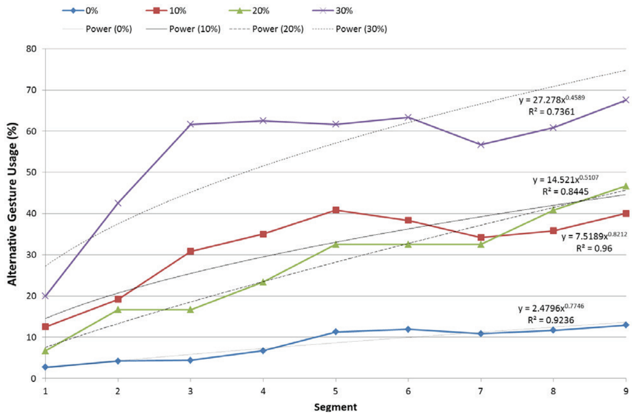


Figure 9: Average Alternative Method Usage (AMU) by injected *misrecognition* rates and segments.

### 4.4.2 Input Time ($T^{h}_{input}$)

There was global learning, as the average time over all letters to input a gesture ($T^{h}_{input}$) correlated well with the power law of learning [5], over all letters ($y = 750.07x^{-0.109}$, $R^2 = 0.7564$). An ANOVA on the data failed to identify a significant effect of injected *misrecognition* rate on $T^{h}_{input}$ ($F_{3,11} = 1.68$, $p > .05$). $T^{h}_{input}$ for 0, 10, 20, and 30% injected *misrecognition* rates was 652, 627, 707, and 593 milliseconds, respectively. Figure 10 illustrates this.

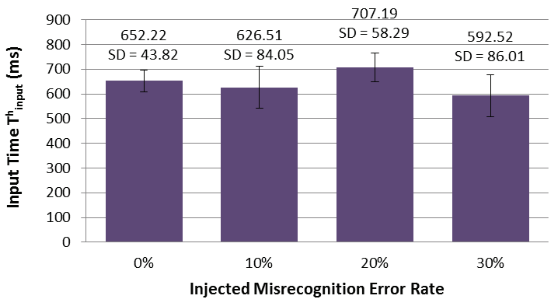


Figure 10: Average Input Time ($T^{h}_{input}$) over all investigated injected misrecognition rates. Error bars represent ±1 standard deviation.

### 4.4.3 Gestures per Character (GPC)

An ANOVA identified a significant effect of injected *misrecognition* error rate on GPC ($F_{3,11} = 4.39$, $p < .05$, $\eta^2 = .20$). A Tukey-Kramer test revealed that 30 and 20% injected *misrecognition* rates yielded significantly higher GPCs compared to 0 and 10%. Average GPC for 0, 10, 20, and 30% injected *misrecognition* rates were 1.11, 1.25, 1.4, and 1.37, respectively. Figure 11 illustrates this. Yet, the data over all letters did not correlate with the power law of learning [5], ($y = 1.2638x^{0.0092}$, $R^2 = 0.1001$).

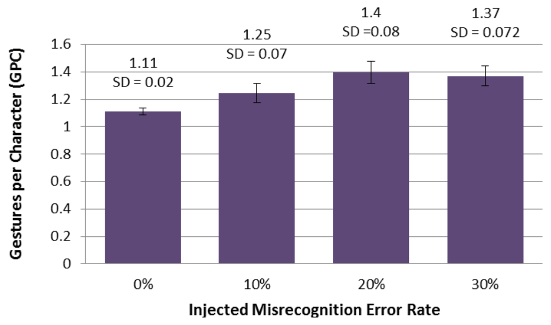


Figure 11: Average Gestures per Character (GPC) over all investigated injected misrecognition rates. Error bars represent ±1 standard deviation.

## 4.5 Discussion

The results show that the use of the alternative method increased over time. Figure 9 illustrated average AMU by injected *misrecognition* error rates and segments, where one can see that participants learned to use the alternative method to input those letters where synthetic *misrecognition* errors were injected, relatively faster compared to the reliable letters. A Tukey-Kramer test showed that the alternative method was used substantially more frequently for the most faulty letter (30% injected *misrecognition* rate) compared to the less faulty ones (0-20% injected *misrecognition* rates). This verifies the hypothesis that users adapt to a gesture-based technique's *misrecognition* errors and that this adaptation rate depends on how frequently they occur. That is, users adapt to an error faster if it occurs more frequently.

Note that even for letters with 0% injected *misrecognition* rate, some users chose to use the alternative gesture, see Figure 9. To investigate this behaviour, we speculated that this was due to the visual similarity between some error prone letters and alternative gesture. Consequently, we investigated if users started to use the alterative gestures for letters that were visually similar to the faulty ones, such as *D* when *B* was more error prone. Yet, we failed to find any notable relationship. One possible reason is that some users treated the whole system as faulty and thus started using alternative gestures for all letters. However, we do not have enough data to validate this hypothesis.

There was no significant effect of injected *misrecognition* rate on $T^{h}_{input}$. Instead, participants learned to input *all* letters faster with time, despite the injected *misrecognition* error rates. This validates to some degree the assumption discussed earlier that switching input methods mid-study–from primary to alternative gesture to adapt to a faulty letter - does not affect entry speed in a significant manner.

One interesting trend visible in Figure 9 is that users adapt to the 10 and 20% injected *misrecognition* rates roughly the same way, while adaptation to 0 and 30% seem distinct. One can speculate that this is because users perceive 10 and 20% injected *misrecognition* rates almost the same way, while 30% was perceived as *too* erroneous. User feedback data also supports this, as most users responded that they were only able to differentiate between the 10 and 20% injected *misrecognition* rates towards the end of the study. This behaviour is similar to earlier results on text entry on faulty keyboards, where 10 and 20% were also not found to be significantly different [3].

A significant effect of injected *misrecognition* rate was identified on GPC. Evidently, 30 and 20% injected *misrecognition* rates yielded significantly higher GPCs compared to 0 and 10%. This is not unexpected as error correction was forced during the study. Thus, participants often had to make multiple attempts to input letters where synthetic *misrecognition* errors were injected. This is also apparent in Figure 11, where one can see the increase in average GPC with increasing injected *misrecognition* rates.

## 5 USER STUDY 2

We conducted a second user study to further observe user adaptation to injected *misrecognitions* and to investigate how the results apply at relatively lower error rates. This study investigated users' adaptation behaviour for injected *misrecognition* error rates from 0 to 10%.

### 5.1 Participants

Twelve participants, aged from 18 to 34 years, average 23.83 (SD = 4.74), took part in the study. They were recruited through online communities, local university e-mailing lists, posting flyers on campus, and by word of mouth. None of them had prior experience with pen-based devices and eleven of them had no knowledge of *Unistrokes* and *Graffiti*. One knew about these techniques, but had never used them. Six of them were female and one was a left-hand pen user. All received a small compensation for participating.

### 5.2 Apparatus, Procedure, and Design

The same apparatus as the first user study were used. The study also used the same procedure and design, as described earlier. The difference is that this study investigated lower injected *misrecognition* error rates (0, 5, 7.5, and 10%).

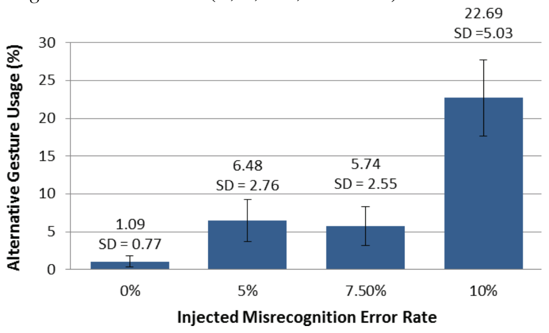


Figure 12: Average Alternative Method Usage (AMU) over all investigated injected misrecognition rates. Error bars represent ±1 standard deviation.

### 5.3 Results

The whole user study lasted from fifty to ninety minutes including the demonstration, practice, and breaks. Upon completion of the study, 25% participants were able to recognize all three error prone letters, 58% recognized the two most error prone letters, and the remaining 17% recognized only the most error prone one. Thus, about 17% of the users did not adapt to the two less faulty letters, where fewer *misrecognition* errors were injected. Consequently, they also did not use the alternative method to input those two letters.

D'Agostino Kurtosis tests on the dependent variables revealed that the data were normally distributed. Also, a Mauchly's test confirmed that the data's covariance matrix was circular in form. Thus, repeated-measures ANOVA was used for all analysis. All statistically significant results are presented with effect size ($\eta^2$).

As in the first study, the data was segmented into blocks of ten appearances of each letter for learning analyses.

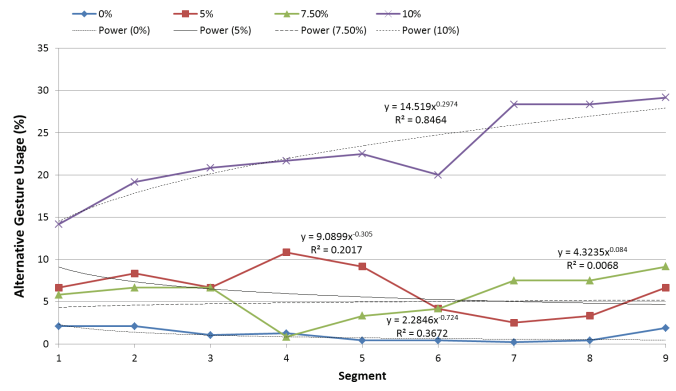


Figure 13: Average Alternative Method Usage (AMU) by injected misrecognition rates and segments.

### 5.3.1 Alternative Method Usage

An ANOVA revealed that there was a significant effect of injected *misrecognition* rate on AMU ($F_{3,11} = 3.52$, $p < .05$, $\eta^2 = .20$). Average AMU for 0, 5, 7.5 and 10% injected *misrecognition* rates were 1.09, 6.48, 5.74, and 22.69%, correspondingly. Figure 12 illustrates this. A Tukey-Kramer test failed to identify groupings. Yet, a (statistically somewhat weaker) Duncan's test identified two groups, 0-7.5% and 10%.

For all injected *misrecognition* rates, the data was again fit with power functions to analyse learning. Figure 13 illustrates this, where the horizontal axis represents the segments and vertical axis represents the average AMU during that segment. As above, the 0% condition is averaged across the four non-faulty letters. We also tried to fit linear functions to the data (0%: $R^2 = 0.24341$, 5%: $R^2 = 0.2467$, 7.5%: $R^2 = 0.13909$, and 10%: $R^2 = 0.83695$), yet the power functions yielded marginally better results (0%: $R^2 = 0.3672$, 5%: $R^2 = 0.20167$, 7.5%: $R^2 = 0.00681$, and 10%: $R^2 = 0.84642$).

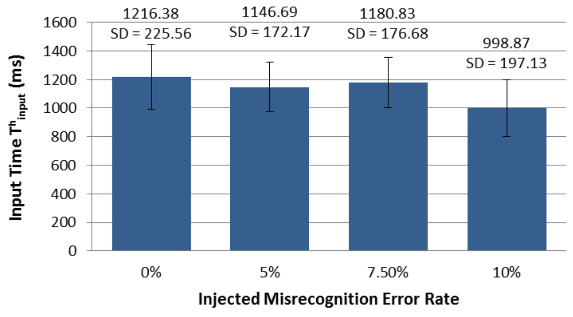


Figure 14: Average Input Time ($T^h_{input}$) over all investigated injected misrecognition rates. Error bars represent ±1 standard deviation.

### 5.3.2 Input Time ($T^h_{input}$)

An ANOVA failed to identify a significant effect of injected *misrecognition* rate on $T^h_{input}$ ($F_{3,11} = 1.34$, $p > .05$). Average $T^h_{input}$ for 0, 5, 7.5, and 10% injected *misrecognition* rates were 1216, 1147, 1181, and 999 milliseconds, correspondingly. Figure 14 illustrates this. Similar to the first user study, the data over all letters correlates very well to the power law of learning [5], ($y = 1534.9x^{-0.22}$, $R^2 = 0.9574$).

### 5.3.3 Gestures per Character (GPC)

An ANOVA on the data identified a significant effect of injected *misrecognition* rate on GPC ($F_{3,11} = 5.33$, $p < .01$, $\eta^2 = .20$). A Tukey-Kramer test revealed that the 10% injected *misrecognition* rate yielded a significantly higher GPC than the 0% injected *misrecognition* rate. Average GPC for 0, 5, 7.5, and 10% injected *misrecognition* rates were 1.07, 1.16, 1.21, and 1.31, respectively, as illustrated in Figure 15. Yet and again similar to the first user study, no strong learning effect was identifiable, ($y = 1.2619x^{-0.044}$, $R^2 = 0.6529$).

## 5.4 Discussion

The results of the study are mostly comparable to the results of the first study: there was a significant effect of injected *misrecognition* rate on both AMU and GPC, but not on input time ($T^h_{input}$). Substantial learning effects were observed for AMU and input time ($T^h_{input}$), but not for GPC. Figure 13 illustrates average AMU by injected *misrecognition* rates and segments. Similar to the first study, one can see there that participants learned to use the alternative method to input letters, where synthetic *misrecognitions* were injected, more frequently relatively faster than the other letters. Also, the 10% injected *misrecognition* condition, common to both studies, yielded comparable AMU (32% and 23%) and GPC (1.25 and 1.31) values, which shows that the results of the two experiments are reasonably consistent. Figure 9 and Figure 13 illustrate that users adapted to the 10% *misrecognition* condition nearly the same way. Therefore, results of this study further strengthen the initial hypothesis and extend the findings towards lower injected *misrecognition* rates.

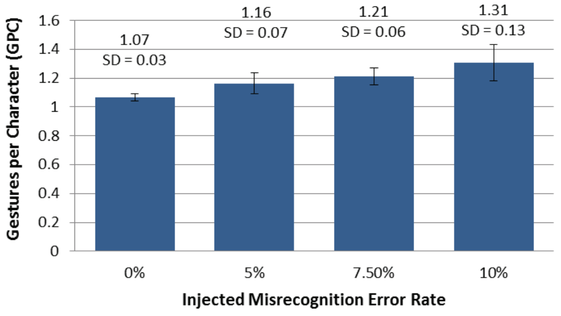


Figure 15: Average Gestures per Character (GPC) over all investigated injected misrecognition rates. Error bars represent ±1 standard deviation.

Figure 13 shows that adaptation to 0, 5, and 7.5% injected *misrecognition* rates were relatively slower than 10%. The results indicate that this is mostly due to insufficient exposure. In the post-study questionnaire, most (75%) participants responded that they managed to identify the 5 and 7.5% faulty letters only shortly before the study ended. This is also apparent in Figure 13, where one can see a distinct trend in adaptation through an increased alternative method usage for these letters during the last three segments, especially for the 5% condition. An ANOVA on the data from these segments identified a significant effect of injected *misrecognition* rate on AMU ($F_{3,11} = 3.96$, $p < .05$, $\eta^2 = .20$). A Tukey-Kramer test identified two statistically different groups for the last three segments: 0% and 5-10%. A statistically weaker Duncan's test identified three statistically different groups for the last three segments: 0%, 5-7.5%, and 10%. Note that the average input time ($T^h_{input}$) was higher during the second study, compared to the first. This is presumably due to the inclusion of relatively more inexperienced users.

## 6 Overall Discussion and Implications

Overall, we observed that users learn to use alternative gestures more quickly, if the primary gestures are faultier. This validates our primary hypothesis. It also complements findings in psychology [9, 10], skill acquisition [28], and user interface research [8, 11, 27] that imply that user interfaces requiring greater efforts from users may facilitate the transition to recall-based expert behaviour. After all, faulty gestures increase user effort to some degree. Marking menus are an example how this concept could be applied [17]. To force users to recall the direction of the intended menu item, they delay the display of the pie menu content. This affects interaction time for novices, but facilitates the transition to expert level [8]. However, this cannot be applied directly in our context, as marking menus do not provide for alternative gestures.

Our results also indicate that gesture recognizers should achieve substantially more than 90% accuracy to appear less (or maybe even in-)distinguishable from a "perfect" system. This is similar to results for keyboard based text entry [3]. The fact that users adapt to unreliable gesture recognizers by using an alternative method for inputting letters that are frequently misrecognized by the system should encourage developers to provide users with an alternative gesture set along with the primary one. Systems should also permit users to swap a primary gesture with an alternative one, and vice versa. A more advanced system could even keep track of the primary and alternative method usage for each letter and might then even propose a switch for letters that are frequently inputted with the alternative method. This may increase the overall recognition accuracy, providing that the recognition rate is higher for the alternative method than the primary one. For this, more distinct gestures would be a good choice as alternatives, as our results showed that users adapt to alternative gestures for frequently misrecognized letters, even when the alternative gestures are relatively less intuitive (and harder to discover) than the primary gestures. We speculate that such a feature can be applied not only on text entry but also to other gesture systems, such as natural user interfaces and application launchers.

Looking across both studies, one interesting observation is that about half of the users were unable to identify all faulty gestures of the system within about an hour. We speculate that this is likely due to different cognitive strategies or personality types. Investigating this is a topic for future work.

## 7 Conclusion

This article presented the results of two empirical studies that identify that users gradually adapt to *misrecognition* errors of a unistroke-based text entry technique's gesture recognizer and that this adaptation rate depends on how frequently such errors occur. That is, users adapt to an error faster if it occurs more frequently. For injected *misrecognition* rates below 10%, no clear pattern for adaptation was observed.

## 8 Future Work

In the future we may conduct a longitudinal study to investigate this range more closely over longer periods. Other potential future work may investigate if our results apply for multistroke recognizers as well. Finally, based on the fact that users' adaptation to *misrecognition* errors is dependent on how frequently they occur, a mathematical model could be developed to predict adaptation rates.

## 9 Acknowledgements

We would like to express our gratitude to NSERC, the GRAND NCE, and York University for funding the research.

## References

[1] Anthony, L. and Wobbrock, J. O. $N-Protractor: A fast and accurate multistroke recognizer. In *Proc. GI '12*, Canadian Information Processing Society (2012), 117-120.

[2] Arif, A. S. and Stuerzlinger, W. How do users adapt to a faulty system? In *CHI '12 Workshop on Designing and Evaluating Text Entry Methods*, 11-14, 2012.

[3] Arif, A. S. and Stuerzlinger, W. Predicting the cost of error correction in character-based text entry technologies. In *Proc. CHI '10*, ACM (2010), 5-14.

[4] Buxton, W. Chapter 7: Touch, gesture & marking. In *Readings in Human Computer Interaction: Toward the Year 2000*, Morgan Kaufmann, 1995.

[5] Card, S. K., Moran, T. P. and Newell, A. *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum, 1983.

[6] Castellucci, S. J. and MacKenzie, I. S. (2008) Graffiti vs. Unistrokes: An empirical comparison. In *Proc. CHI '08*, ACM (2008), 305-308.

[7] Castellucci, S. J. and MacKenzie, I. S. UniGest: Text entry using three degrees of motion. In *CHI EA '08*, ACM (2008), 3549-3554.

[8] Cockburn, A., Kristensson, P. O., Alexander, J., and Zhai, S. Hard lessons: Effort-inducing interfaces benefit spatial learning. In *Proc. CHI '07*. ACM (2007), 1571-1580.

[9] Craik, F. I. M. and Lockhart, R. S. Levels of processing: A framework for memory research. *Journal of Verbal Learning and Verbal Behavior 11*, 6 (1972), 671-684.

[10] Craik, F. I. M. and Tulving, E. Depth of processing and the retention of words in episodic memory. *Journal of Experimental Psychology: General 104*, 3 (1975), 268-294.

[11] Ehret, B. D. Learning where to look: Location learning in graphical user interfaces. In *Proc. CHI '02*. ACM (2002), 211-218.

[12] Goldberg, D. and Richardson, C. Touch-typing with a stylus. In *Proc. CHI '93*, ACM (1993), 80-87.

[13] Hafner, K. Did Bill Gates really say that? *The New York Times: Bits*, Mar 25, 2008. http://nyti.ms/PFiPkO.

[14] Isokoski, P. A minimal device-independent text input method. Technical Report. University of Tampere, Finland, Report A-1999-14, 1999.

[15] Karam, M. and Schraefel, M. C. Investigating user tolerance for errors in vision-enabled gesture-based interactions. In *Proc. AVI '06*, ACM (2006), 225-232.

[16] Költringer, T. and Grechenig, T. Comparing the immediate usability of Graffiti 2 and virtual keyboard. In *CHI EA '04*, ACM(2004), 1175-1178.

[17] Kurtenbach, G. and Buxton, W. The limits of expert performance using hierarchic marking menus. In *Proc. CHI '93*, ACM (1993), 482-487.

[18] Labahn, G., Lank, E., Marzouk, M., Bunt, A., MacLean, S., and Tausky, D. MathBrush: A case study for pen-based interactive mathematics. In *Proc. SBIM '08*. Eurographics Association (2008), 143-150.

[19] LaLomia, M. User acceptance of handwritten recognition accuracy. In *Proc. CHI '94*, ACM (1994), 107-108.

[20] Li, Y. Gesture search: A tool for fast mobile data access. In Proc. UIST '10, ACM (2010), 87-96.

[21] MacKenzie, I. S. and Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice. *Hum.-Compute Interact.*, 17 (2002), 147-198.

[22] MacKenzie, I. S., Chen, J., and Oniszczak, A. Unipad: Single stroke text entry with language-based acceleration. In *Proc. NordiCHI '06*, ACM (2006), 78-85.

[23] Mankoff, J. and Abowd, G. D. Cirrin: A word-level unistroke keyboard for pen input. In *Proc. UIST '98*, ACM (1998), 213-214.

[24] Mankoff, J. and Abowd, G. D. Error correction techniques for handwriting, speech, and other ambiguous or error prone systems. Technical Report, Georgia Tech., Atlanta, GA, USA, GVU-99-18, 1999.

[25] Mazur, J. E. and Hastie, R. Learning as accumulation: A reexamination of the learning curve. *Psychological Bulletin 85*, 6 (1978), 1256-1274.

[26] Newell, A. and Rosenbloom, P. S. Mechanisms of skill acquisition and the law of practice. In *Cognitive Skills and Their Acquisition*, Lawrence Erlbaum, 1981.

[27] Riche, Y., Riche, N. H., Isenberg, P., and Bezerianos, A. Hard-to-use interfaces considered beneficial (some of the time). In *CHI EA '10*. ACM (2010), 2705-2714.

[28] Schmidt, R. A. and Bjork, R. A. New conceptualizations of practice: Common principles in three paradigms suggest new concepts for training. *Psychological Science 3*, 4 (1992), 207-217.

[29] Shilman, M., Tan, D. S., and Simard, P. CueTIP: A mixed-initiative interface for correcting hand-writing errors. In *Proc. UIST '06*, ACM (2006), 323-332.

[30] Tappert, C. C. and Cha, S. English language handwriting recognition interfaces. In *Text Entry Systems: Mobility, Accessibility, Universality*, Morgan Kaufmann, 123-137, 2007.

[31] Tu, H., Ren, X., and Zhai, S. A comparative evaluation of finger and pen stroke gestures. In *Proc. CHI '12*, ACM (2012), 1287-1296.

[32] Vatavu, R.-D., Anthony, L. and Wobbrock, J.O. Gestures as point clouds: A $P recognizer for user interface prototypes. In *Proc. ICMI '12*, ACM (2012), 273-280.

[33] Venolia, D. and Neiberg, F. T-Cube: A fast, self-disclosing pen-based alphabet. In *Proc. CHI '94*, ACM (1994), 265-270.

[34] Wobbrock, J. O., Myers, B. A., and Kembel, J. A. EdgeWrite: A stylus-based text entry method designed for high accuracy and stability of motion. In *Proc. UIST '03*, ACM (2003), 61-70.

[35] Wobbrock, J. O., Wilson, A. D., and Li, Y. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proc. UIST '07*, ACM (2007), 159-168.

[36] Wobbrock, J.O. and Myers, B.A. Gestural text entry on multiple devices. In *Proc. ASSETS '05*, ACM (2005), 184-185

[37] Zhai, S. and Kristensson, P. Shorthand writing on stylus keyboard. In *Proc. CHI '03*, ACM (2003), 97-104.