EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2011), pp. 1–8
T. Hammond and A. Nealen (Editors)

Is the iPad useful for sketch input? A comparison with the Tablet PC

S. MacLean, D. Tausky, G. Labahn, E. Lank, and M. Marzouk

Cheriton School of Computer Science, University of Waterloo, Canada

Abstract

Despite the increasing prevalence of touch-based tablet devices, little attention has been paid to what effects, if any, this form factor has on sketch behaviours in general and on sketch recognizers in particular. We investigate the title question through an empirical study in the context of mathematical expression recognition. Using a corpus of thirty expressions drawn on Tablet PC and iPad by thirty writers, we show that characteristics of sketching and drawing differ depending on platform. While recognition is most accurate on the Tablet PC due to its technical superiority, recognition is feasible on mobile touch-based devices. However, there are characteristics of sketching on multi-touch tablets that differ, and these physical characteristics of writing impact recognition accuracy. Together, our observations inform the broader Sketch Recognition community as they design systems targeted to multi-touch tablets.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Systems]: Information interfaces and presentation—User interfaces

1. Introduction

Tablet based computers are popular devices for applications that can make use of two dimensional input and the ability of sketching smooth curves in two dimensions. Examples of such applications include mathematics, music, drawing programs and environments for marking up documents. While the current generation of Tablet PC has been a popular platform for two-dimensional input for the last ten years, more recently multi-touch tablets such as the iPad, the Android tablet, and the Playbook have supplanted the Tablet PC as a candidate for two-dimensional input.

The popularity of the iPad, Google Android and RIM's Playbook devices marks a significant shift from earlier tablet interfaces designed around the Tablet PC. While Tablet PCs were designed for stylus interaction, the newer devices make use of multi-touch interfaces designed for finger interaction. However, even with such a shift, there is significant demand for sketching applications on multi-touch tablets. A quick perusal of the *App Store* for Apple's popular iPad tablet displays many examples of sketching and drawing applications. Adobe's Ideas and Autodesk's SketchBookX are popular examples of these sketching and drawing applications [Aut11].

For sketch recognition researchers, it is clear that new sketch applications may wish to target multi-touch tablet devices alongside the more traditional Tablet PC form factor. However, despite years of developing interfaces and techniques based on the Tablet PC interface, there is a notable absence of research on how these sketch technologies work on touch based platforms. An exploration of related literature for information on how the drawing task changes when moving to multi-touch (does speed, size, or legibility change?), on the expected performance of recognition algorithms (how much worse should we expect it to be?), and on user attitudes toward these multi-touch tablets as a platform for drawing (do users find the platform usable? compelling?) gives little guidance. Questions such as: "Is the 'fat-finger problem' [WFB*07] significant?" and "Is a stylus necessary for reliable recognition on multi-touch tablets?" (see Figure 1) remain unanswered. This implies a clear need for studies of multi-touch versus tablet as a target platform for sketch, studies similar in nature to those of Apte and Kimura [AK93] comparing the mouse and tablet.

Surveying all possible sketch domains is clearly not possible. Amongst the many sketch based applications for tablet based computers, the ability to input and work with mathe-

matical expressions has been a popular target for research. A number of current math recognition systems have the ability to work with symbolic and numeric input that are available for the Tablet PC platform including for example MathBrush [LLM*08], MathPaper [ZMLL08] and FFES [ZBC02]. However, math recognition requires considerable computing resources, something not present in the current generation of multi-touch tablets. We know of no math recognition software other than our own which is available for iPads.

On the other hand, there are properties of mathematical notation that make it well suited to a study of sketching on Tablet PC versus multi-touch tablet platforms. Mathematical notation is two dimensional and, as such has not been a natural fit for one dimensional interface devices based on keyboards and mice. Mathematical expressions contain a variety of symbols (thus testing for object recognition) and a variety of spatial relationships (thus testing for spatial recognition). It is also possible to determine if a given mathematical expression has been recognized correctly or even a close-to-correct measure for such expressions. Finally, mathematical manipulation as done by pen and paper also lends itself to many tablet-friendly gestures. For example, crossing out common factors in rational fractions or deleting common terms on both sides of an equation is typically done via commands in one dimensional environments but involves no more than a simple crossing-out gesture on pen and paper or touch tablet.

Our study led to both expected and unexpected results. The fact that, at the level of mathematical notation recognition specifically, we found recognition was more accurate on the Tablet PC than on the iPad was expected. Similarly it was not surprising that writer-dependent training of our symbol recognition system produced large increases in accuracy on all platforms, though these increases were most dramatic on the Tablet. Less expected was the observation that the Tablet platform's advantage was limited to problems requiring discrimination of fine detail, such as symbol recognition. Coarser features such as spatial relationships were recognized equally well on all platforms, suggesting that the Tablet's superiority is more a function of its relative technical sophistication than of the form factor itself. While the Tablet PC outperformed the multi-touch tablet in both accuracy and writing speed, it was surprising that pens designed specifically for multi-touch tablet devices (see Figure 1, where the second stylus is one designed for the iPad) were of little benefit on multi-touch tablets. For example, symbol recognition accuracy on the iPad was higher when drawing with a finger than with an iPad stylus. As well, writing speed did not differ significantly with finger from speed with iPad stylus.

The rest of this paper is organized as follows. The next section introduces basic background on the Tablet form factor and the 'fat-finger problem'. Section 3 describes the methodology of our study and gives an overview of our recognition system. Section 4 discusses how we interpreted the results of our experiments, and provides math recognition accuracy rates. Following that, Section 5 analyzes some physical characteristics of the expression transcriptions we collected, and describes how those characteristics affected the lower-level classification systems included in our math recognition system, explaining the accuracy results in more detail. Finally, we point out our main findings in Section 6, and offer some problems whose solution would likely dramatically improve recognition accuracy on touch-based devices.

Figure 1: *An Tablet PC stylus (top) and an iPad stylus (bottom).*

2. Background

Currently, tablet computing platforms can be broadly divided into two categories - the Tablet PC platform, which is based on a conventional PC architecture and mobile multi-touch tablets such as the Apple iPad. The Tablet PC platform is a conventional laptop, coupled with the Microsoft Windows operating system and a high resolution electromagnetic digitizer. Typical Wacom digitizers used on Tablet PCs provide spatial resolution of over 1000dpi and a sampling frequency of 133Hz. Although some Tablet PCs support touch and multi-touch, the primary mode of interaction is with a stylus. Extensions to the Microsoft operating system allow developers to use built-in ink collection and handling data structures and character recognizers.

In comparison, multi-touch tablets have significantly reduced hardware and processing capabilities when compared to Tablet PC computers. As well, the capacitive based touch screen of the iPad captures much lower resolution ink sampling (at most 132 dpi), compared to the Tablet PC, at a far lower sampling rate that is determined by the operating system as it responds to events. Furthermore, while capacitive styli do exist, the tips are necessarily large when compared with their electromagnetic counter-parts. The primary method of interacting with such tablets is by the multi-touch interface, using your fingers to directly gesture on the screen. While some of the differences between these platforms, notably processor power and screen resolution, may change with time, the paradigm of finger or capacitive stylus drawing will, we argue, persist on multi-touch tablet devices. It is the effect of these features on input recognition that we examine in this paper.

One commonly cited problem with multi-touch interaction is the 'fat-finger problem' [WFB*07, WWC*09], a problem that arises from the relatively broad profile of a finger relative to the pixel-based contact point registered on a multi-touch screen. The pixel that represents the contact point with the display is always occluded by the finger. Furthermore, the fat-finger phenomenon persists even when moving to a capacitive sylus designed for multi-touch tablets. To enable high capacitance readings upon contact with the display, a capacitive stylus is broader. It presents a profile more like that of a crayon than like that of a traditional stylus.

While researchers have studied the design of widget sets to address widget interaction on multi-touch surfaces [WFB*07, WWC*09], as we note earlier, we are aware of no studies contrasting sketch input and recognition behaviour on multi-touch tablet versus traditional Tablet PC platforms.

3. Methodology

3.1. Data collection

Thirty students participated in our data collection study. They were recruited using posters and were given a $10 gift certificate in exchange for transcribing mathematical expressions. Participant data was collected on three different computing configurations: A Tablet PC using a stylus, an Apple iPad using a stylus and an Apple iPad using your finger. Participants sketched the same 30 expressions on each platform, sketching all the expressions on one platform before switching to the next platform. The study was fully factorial, with the ordering of the platforms being switched for each participant, such that an equal number of participant performed each plausible ordering.

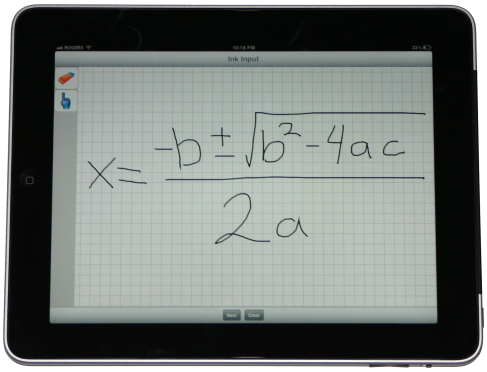

Figure 2: *Our data collection software on the Apple iPad.*

The equations that participants were asked to transcribe consisted of 30 mathematical expressions derived from a first year introductory Calculus textbook [HHGWM98]. The average equation contained 12 symbols. Examples of some of the equations are shown below. The entire set is available in an online appendix at http://www.scg.uwaterloo.ca/mathbrush/publications.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$A = \frac{1}{\sqrt{2\pi}} \int_0^b e^{\frac{-x^2}{2}}\, dx$$

Each session was organized as follows: One of three tablet platforms was placed in front of each participant, displaying the transcription interface shown in Figure 2. The functionality of the interface was described to them and demonstrated by the researcher conducting the data collection. The thirty expressions to be transcribed were organized in a binder, printed one expression per page. Participants were asked to draw expressions presented on each page in the binder, then click 'Next' to clear the screen and proceed to the next expression. They could also click 'Clear' to clear the display and redo an expression. The order of the expressions was the same for all participants.

Once the participant had completed all 30 expressions on a particular platform, they were rotated to a different platform and told to repeat the transcription task. Participants were asked in written instructions to write as legibly as they would on an assignment they would hand in for evaluation. It was observed part way through the data collection study that some participants were not writing legibly. We therefore changed the protocol to include both an oral and written reminder to write legibly. Participants were told that if they did not recognize a symbol, or were unsure how to draw a symbol, they should ask for help. Aside from requested assistance, no feedback or advice was offered during transcription. They were not advised on how to write or on how to interact with the data collection software.

Once the participants had completed transcribing the 30 calculus equations on all three platforms, participants were asked to transcribe a series of randomly generated mathematical equations on the Tablet PC platform for the remainder of the one-hour session. The purpose of this second task was simply to gather additional examples of characters and spatial relationships. The techniques used to generate the random equations are discussed in [MTL*09]. Our collection software recorded the x, y position and system timestamp associated with each sampled ink point.

3.2. Recognition system architecture

Our recognition system is based on a relational grammar formalism that describes how symbols may be arranged into spatial relationships to produce well-formed mathematical expressions [ML10]. Within the system, there are two classification subsystems. The symbol classification subsystem decides which input strokes should be grouped together into distinct symbols, and identifies what symbols those stroke groups represent. The relation classification subsystem determines which spatial relationships apply to a pair of stroke

groups, each representing potential subexpressions of the input expression. The relations indicate general writing directions (e.g., horizontal, vertical, superscript, subscript, containment). The grammar formalism integrates these two subsystems by specifying how individual symbols and subexpressions combine via the spatial relations into larger mathematical expressions. The grammar model also attaches semantic interpretations to these symbol and subexpression arrangements.

Because of ambiguities in handwritten input, it is unrealistic to expect perfect accuracy in the recognition subsystems. As such, they each report several candidates, along with confidence scores. We say that a particular symbol or relation classification decision is *correct* if the top-ranked candidate is the right one. If the decision was not correct, but the correct decision appeared as a candidate in the subsystem's output, we call the decision *ranked*.

Our recognition system reports interpretations of the input expression in decreasing order of confidence. It can report interpretations of any subset of the input, allowing users to correct the recognition results. For example, if a user draws the expression $A^x + b$, but it is incorrectly recognized as $A^X tb$, then the user may correct the expression to an addition, $A^X + b$, and correct the upper-case X to the correct symbol x, provided these alternatives were identified by the system as valid parses.

4. Results

Given the recognition architecture described in the previous section, the transcriptions we collected were annotated with ground truth identifying which strokes correspond to which symbols, and which groups of strokes relate to one another by spatial relationships. For example, consider the expression shown in Figure 3. Assuming that the strokes are identified by integers starting from 0 and increasing from left to right, the ground truth can be written schematically as

{0}	SYMBOL	'e'
{1}	SYMBOL	'l'
{2}	SYMBOL	'n'
{3,4}	SYMBOL	'x'
{5,6}	SYMBOL	'$=$'
{7,8}	SYMBOL	'x'
{0}	SUPERSCRIPT	{1,2,3,4}
{0,1,2,3,4}	HORIZONTAL	{5,6}
{5,6}	HORIZONTAL	{7,8}

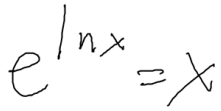

Figure 3: *Expression demonstrating our ground-truth format.*

We discarded any transcriptions which were illegible, incomplete, or contained cursive writing, which our recognizer does not currently support. Generally, if a human operator could not decide on an appropriate ground-truth assignment to a transcription, it was discarded. Of the 900 possible expressions, this left 794, 778, and 803 available for testing in the Tablet PC, iPad (pen), and iPad (finger) configurations, respectively.

Using this ground truth, we evaluated our math recognizer on the data collected under each of the three configurations. To determine the recognizer's accuracy, we measured, for each expression, how many corrections were required to be made through the recognition interface to obtain the correct recognition result. This measurement was taken automatically by a program that simulates a user interacting with the recognition system. If an expression is recognized perfectly, then it requires zero corrections. We say such a result is *correct*. Otherwise, the program identifies which symbols and/or subexpressions were recognized incorrectly and requests alternative parses from the recognizer for the appropriate subsets of the input. It searches the alternatives for symbols or subexpressions matching the expression's ground truth, and, if it finds them, corrects the recognizer's output. These corrections may change mistaken symbol identities, or they may change the expression's structure and semantics. If, after making some corrections, the program obtains the correct result, we say the result is *attainable*. Otherwise, the result is *incorrect*.

For example, the first expression in Figure 4 was recognized as $\int x^n \, dx = \frac{1}{n+1} x^{n+1}$ and so counts as correct. In the second expression, the π symbol was recognized as an upper-case Π. The lower-case symbol was available as an alternative, so the expression was attainable with one correction. In the third expression, the first closing parenthesis was recognized as the number 7, causing the expression to be recognized as $x^3 + y^3 = \left(x + y^7 \left[x^z - xyT\right] z\right)$. The correct parse was discovered by the recognizer, but without sufficient confidence for it to be among the top candidates. It was attainable after two corrections to the expression structure and three to incorrectly recognized symbols.

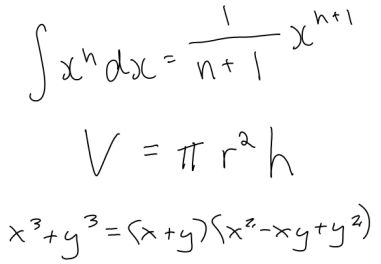

Figure 4: *Expressions demonstrating the classification of our test results.*

It is important to distinguish between failures in symbol classification and failures in relation classification. This is especially important since our system does not allow for extraneous ink (e.g., small dots from accidental finger or pen contact), and uses a model-based approach to symbol classification (preventing recognition of visually similar symbols that contain differing numbers of strokes).

Particularly in writer-independent testing, poor symbol classification accuracy can prevent expression recognition from succeeding. If the correct symbol identities are not reported as candidates by the symbol recognizer, then the test result will be incorrect no matter how many corrections the testing program makes. We call such expressions *infeasible*. Figure 5 shows two infeasible transcriptions. In the first, the left hand side was intended to be x, but the participant did not lift the pen, resulting in a symbol that looks more like α, which the symbol classification system could not identify as x. Note that this particular writer consistently wrote x symbols in this way, so it was not a transcription error. In the second transcription, there is some extraneous ink around the plus symbol, which our recognizer was forced to interpret incorrectly since it lacks a model for noise or extra ink.

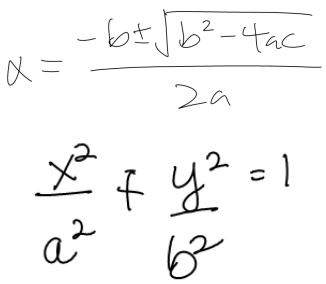

Figure 5: *Unknown symbol allographs and extraneous ink cause expressions to be classified as infeasible.*

To better identify the effects on recognition of each configuration, we evaluated the recognizer under three different scenarios. The first, called *default*, simply ran each expression in the corpus through the testing program. Figure 6 illustrates the recognizer's accuracy in each configuration for this scenario. Since the number of usable transcriptions is similar between scenarios, the results are reported as percentages rather than raw expression counts.

The low feasibility rate in the default scenario indicates that symbol recognition failure often prevented the recognizer from obtaining the correct parse. The two remaining scenarios were designed to avoid this problem so as to isolate the effects of each configuration on relation classification. In the *pretrain* scenario, we used a writer-dependent training strategy. Prior to running a participant's transcriptions through the test program, we added up to ten samples of each symbol to the database of symbol models. These samples were extracted from the participant's transcriptions of

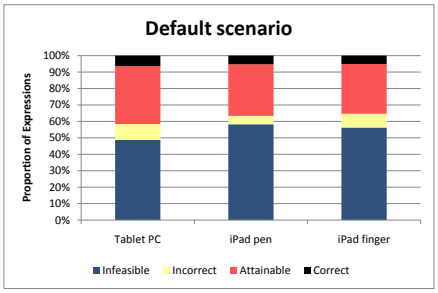

Figure 6: *Recognition accuracy in the default scenario.*

randomly-generated expressions. The relation classifier was not directly affected by this training step. One participant did not provide random expression transcriptions and was omitted from this scenario. Figure 7 shows the recognizer's accuracy for the pretrain scenario.

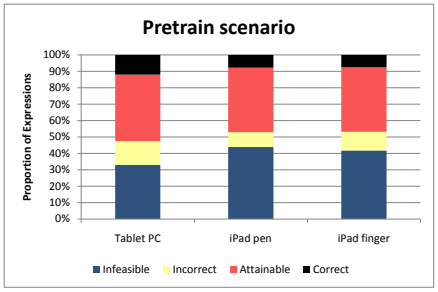

Figure 7: *Recognition accuracy in the pretrain scenario.*

The final scenario, called *perfect*, focused on relation classification accuracy by bypassing the symbol recognition phase altogether. In it, the correct symbol identities and bounding boxes were extracted from expression ground truth and passed directly to the math recognizer. Figure 8 shows the recognizer's accuracy for the perfect scenario.

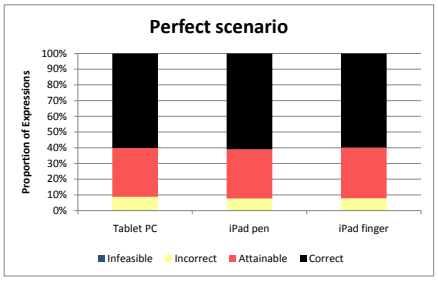

Figure 8: *Recognition accuracy in the perfect scenario.*

Although these recognition rates are relatively low, it should be noted that math recognition is a difficult problem. These rates are comparable to those reported by other

researchers both when ignoring symbol recognition (e.g., [GC04, ZBC02]), and under comparable training regimes (e.g., [AMVG09]).

5. Analysis

During our experiments, we collected a large amount of data concerning the behaviour of our recognizers as well as the physical characteristics of the transcriptions. We also tested the symbol and relation classification subsystems on isolated symbols and subexpression groups extracted from the transcriptions. These tests measured classification accuracy independently of any complicating factors introduced by the math recognition system as a whole. We will make some observations about the transcriptions, then use those facts to interpret the behaviour of our recognition system.

5.1. Physical transcription characteristics

The physical characteristics we measured are broadly divisible into four categories: time (average time per pen-down stroke, average time between strokes, average time per expression transcription), speed (average pen-tip speed), average number of sample points per stroke, and physical stroke size (average arclength and bounding box size).

Virtually all participants took less time per expression and per stroke on the Tablet PC than in either iPad configuration. Most also took less time per expression and per stroke when using a stylus on the iPad than when using their finger. However, time per stroke was only roughly correlated with time per expression. The time taken *between* strokes was a much better predictor, being nearly perfectly correlated with time per expression.

Repeated Measures ANOVA was performed to analyze the effect of platform on drawing speed and symbol size (height). Speed was calculated in inches per second on the displays, and was reported as an average speed for each participant. Figure 9 depicts the mean and standard deviation of users' pen speeds (i.e. the mean of means and the standard deviation of means). When analyzing average drawing speed across users on the different platforms, Mauchly's Test of Sphericity indicated that the sphericity assumption was violated ($p < 0.001, \varepsilon = 0.574$); therefore Greenhouse Geisser correction was applied to analyze within subjects effect. Repeated measures ANOVA indicates that average drawing speed differs significantly depending on platform ($F_{1.149,33.313} = 18.173, p < 0.001$). Post-hoc tests with Bonferroni correction indicated that the Tablet PC was significantly faster than both the stylus and finger on the iPad ($p < 0.001$). No significant difference was found between stylus or finger on the iPad ($p = 0.923$).

Similarly, height was calculated as the average height of a mathematical symbol in the expressions drawn. The average height of symbols (in inches) is depicted in Figure 10. Similar to drawing speed, we found that Mauchly's

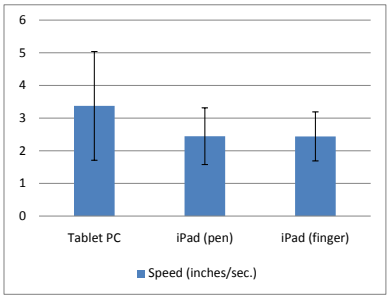

Figure 9: *Average pen speed in each configuration.*

Test of Sphericity indicated that the sphericity did not hold ($\varepsilon = 0.052$). Repeated measures ANOVA using Greenhouse Geisser correction indicates that the average height of symbols differs significantly depending on platform ($F_{1.105,32.042} = 27.886, p < 0.001$). Post-hoc tests using Bonferroni correction indicate significant differences between average heights across all platforms ($p < 0.01$ in all cases).

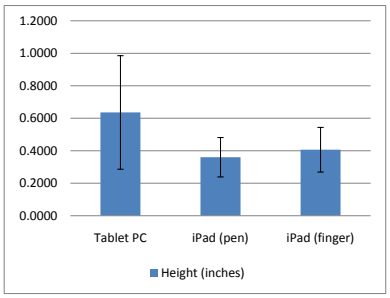

Figure 10: *Average stroke height.*

5.2. Relations between physical characteristics and recognition accuracy

Each of these physical characteristics had an impact on the quality of recognition, which often varied between configurations. The single greatest predictor of recognition accuracy was the time taken between strokes. In general, though, as the time taken to write expressions increased, so too did all accuracy measurements. This is not surprising, as slower writing is often neater and easier to read for people as well. Notably, this observation did not apply to as great an extent when writing with a finger on the iPad.

The number of points per stroke affected each configuration differently. On the Tablet PC, transcriptions were recognized more accurately the fewer points they had. This can be understood by noting that strokes with more points are longer and more complex, and therefore more succeptible to recognition errors and precision loss from downsampling.

On the iPad, so few points were generally available that downsampling was not employed by the recognition subsystem. More sample points in these configurations led to more discernable symbol shapes and bounding boxes, which enabled higher recognition accuracy by both classification subsystems, and hence by the math recognition system as a whole.

Typically, stroke size and writing speed increased or decreased together. Faster and larger writing tended to be recognized more poorly by both the symbol and relation classifiers, hence reducing the attainability and correctness rates of the math recognizer. One exception was that the proportion of symbol classification decisions that were correct increased for larger, more quickly-drawn symbols on the iPad. There are two reasons for this exception. First, the "fat finger problem" causes small symbols to be drawn inaccurately, especially given the relatively poor spatial resolution of the iPad touchscreen. Second, our experiments showed that the faster one draws on an iPad, the faster its operating system reports tracking events, increasing the sampling rate. This makes available more points over the same length of ink, which, as we saw above, improves recognition accuracy on the iPad.

5.3. Comparison of scenarios and configurations

The math recognizer's overall performance was clearly best in the perfect scenario, with over 90% of expression attainable, and about two-thirds of those correct. This result was expected, as that scenario offers a best-case scenario to the math recognizer in which there is no ambiguity between symbol candidates and the top-ranked candidate is always correct. In the perfect scenario, there was no significant difference in the math recognizer's accuracy between the three configurations. This was because the accuracy of the relation classifier – the only classifier required in the perfect scenario – was comparable in all three configurations for most participants. The participants for whom there was a significant difference between configurations were split with roughly half having more accurate classification on the Tablet PC, and other half having more accurate classification on the iPad.

Comparisons are more interesting between the default and pretrain scenarios. As one might expect, math recognition accuracy was significantly better in the pretrain scenario, driven by improvements in symbol classification accuracy. Writer-dependent training of the symbol recognizer caused roughly 33% more feasible expressions than in the default scenario. The proportion of feasible expressions that were correct or attainable increased in spite of the fact that the feasible transcriptions of the pretrain scenario generally contained more complex math expressions and symbol shapes than those of the default scenario. Writer-dependent classifier training is therefore an effective strategy for increasing recognition accuracy in all configurations.

More specifically, the attainability rate increased by 25-35% between the default and pretrain scenarios in all configurations. However, while the proportion of expressions recognized correctly nearly doubled in the tablet configuration, it only increased by about 50% in both iPad configurations. This discrepancy was due to differences in the symbol classifier's response to training on each platform. Although the proportion of ranked decisions made by the symbol classifier increased a similar amount in all configurations, the proportion of correct decisions increased nearly twice as much on the Tablet PC compared to the iPad. In this sense, the pretraining step was much more effective on the Tablet PC. Comparing the number of corrections to symbol classification errors between the default and pretrain scenarios further demonstrates the effectiveness of pretraining on the Tablet PC: on average, over 35% fewer corrections were required on the tablet in the pretrain scenario, while the number of corrections required on the iPad only dropped by 10-15%.

Overall, recognition was most accurate on the Tablet PC, significantly less accurate on the iPad with a pen, and somewhat less accurate again on the iPad with a finger. We expected this order at the outset of the experiment, believing (correctly) that the Tablet PC's high resolution offered intrinsically better recognition capabilities, and (somewhat incorrectly) that writing with a pen on the iPad would be more natural and amenable to recognition than writing with a finger. The Tablet PC's advantage was not as great as we initially suspected, though, and was limited almost exclusively to its superiority in symbol classification. In the default scenarios, the proportion of symbol classification decisions that were correct was roughly 20-25% higher on the tablet than on the iPad, and the gap increased in the pretrain scenario. But the proportion of ranked decisions was only 6-8% higher in the default scenario, and that gap shrank after pretraining. The difference in the attainability rate between the configurations was therefore only about 5% in both scenarios, although more corrections to symbol classification errors were required on the iPad.

Surprisingly, symbol classification accuracy was higher on the iPad when writing with a finger than when using the stylus. This was likely due to the larger, slower writing style that participants used when writing with their fingers, which led to more accurate recognition on the iPad, as explained above. Even so, overall math recognition accuracy was higher on the iPad when participants wrote with a pen. This apparent contradiction is explained by the fact that our classification subsystem tests were performed on isolated symbols and subexpression groups. In the context of expression recognition, it is possible for incorrect stroke groups to be proposed for symbol recognition, or for the correct stroke group to not be proposed. (For example, if the strokes were drawn too far apart.) It is also possible for relation classification confidence to be higher for incorrect subexpression groups than for the correct ones, thereby causing the math recognizer to choose an incorrect subexpression structure. These types of problems are exacerbated by the "fat finger"

phenomenon, which prevents writers from knowing exactly where they are inking, and causes strokes to be closer together or farther apart than intended.

6. Conclusions and Future Work

Multi-touch devices like Apple's iPad seem to be natural platforms on which to deploy sketch-based interfaces. We endeavoured to discover whether, given the relatively low resolution and limited processing power of these devices, such interfaces are, in fact, feasible, focusing on our research area of math expression recognition. We collected transcriptions of thirty mathematical expressions, each written once by thirty writers on the Tablet PC, on the iPad with a stylus, and on the iPad with a finger. By analyzing the results of running the transcriptions through our math recognition system, it is apparent that multi-touch devices are a viable platform for sketch-based interfaces, though this claim comes with some caveats and requirements for future work.

As expected, recognition accuracy was significantly better on the Tablet PC than in either iPad configuration. But, accuracy was not as poor on the iPad as we had anticipated given the huge difference in temporal and spatial resolution between it and the Tablet PC. Our relation classifier performed equally well on both platforms, suggesting that many coarse features can be adequately recognized on touch devices using "out-of-the-box" techniques. And although the Tablet PC platform was significantly better for for recognizing fine features, such as those required for symbol classification, its advantage was not as large as we initially suspected it would be.

Writer-dependent training successfully improved recognition rates in all configurations, but it benefited the Tablet PC disproportionately due to the Tablet's superior resolution. Given a method for capturing finer-scale details on the iPad, whether through more powerful hardware or clever algorithms, we expect that the higher rate of correct decisions observed on the Tablet PC should disappear.

The above observation of our mathematics recognizer's behaviour is further validated by observations on the basic trajectory observations of writing on the two platforms. Our experiments demonstrated that certain physical properties of writing engendered more accurate recognition on the iPad. In particular, a writing style that we might characterize as "methodical but purposeful" in which ink strokes are placed carefully, but drawn quickly, led to the best recognition results on that platform. An interesting question is therefore, what interfaces can we design that will encourage users to write in this style? This question is especially important on mobile touch-based devices, on which the limited display area permits somewhat more control over users' input options than typical Tablet PC programs.

Acknowledgements

The authors wish to thank John Mitchell for his help with organizing and running the data collection sessions.

References

[AK93] APTE A., KIMURA T. D.: A comparison study of the pen and the mouse in editing graphic diagrams. In *Proceedings of 1993 IEEE Symposium on Visual Languages* (1993), pp. 352–357. 1

[AMVG09] AWAL A.-M., MOUCHERE H., VIARD-GAUDIN C.: Towards handwritten mathematical expression recognition. In *Document Analysis and Recognition, International Conference on* (Los Alamitos, CA, USA, 2009), IEEE Computer Society, pp. 1046–1050. 6

[Aut11] AUTODESK INC.: SketchBookX for the ipad, 2011. http://itunes.apple.com/us/app/sketchbook-mobilex/id327376639?mt=8. 1

[GC04] GARAIN U., CHAUDHURI B.: Recognition of online handwritten mathematical expressions. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34*, 6 (Dec. 2004), 2366–2376. 6

[HHGWM98] HUGHES-HALLETT D., GLEASON A., WILLIAM MCCALLUM E. A.: *Calculus: Single and Multivariable*, second ed. Wiley, 1998. 3

[LLM*08] LABAHN G., LANK E., MACLEAN S., MARZOUK M., TAUSKY D.: MathBrush: A system for doing math on pen-based devices. In *Proceedings of the 2008 The Eighth IAPR International Workshop on Document Analysis Systems* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 599–606. 2

[ML10] MACLEAN S., LABAHN G.: *Recognizing handwritten mathematics via fuzzy parsing*. Tech. Rep. CS-2010-13, School of Computer Science, University of Waterloo, 2010. 3

[MTL*09] MACLEAN S., TAUSKY D., LABAHN G., LANK E., MARZOUK M.: Tools for the efficient generation of hand-drawn corpora based on context-free grammars. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling* (New York, NY, USA, 2009), SBIM '09, ACM, pp. 125–132. 3

[WFB*07] WIGDOR D., FORLINES C., BAUDISCH P., BARNWELL J., SHEN C.: Lucid touch: a see-through mobile device. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2007), UIST '07, ACM, pp. 269–278. 1, 3

[WWC*09] WIGDOR D., WILLIAMS S., CRONIN M., LEVY R., WHITE K., MAZEEV M., BENKO H.: Ripples: utilizing per-contact visualizations to improve user interaction with touch displays. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (New York, NY, USA, 2009), UIST '09, ACM, pp. 3–12. 3

[ZBC02] ZANIBBI R., BLOSTEIN D., CORDY J. R.: Recognizing mathematical expressions using tree transformation. *IEEE Trans. Pattern Anal. Mach. Intell. 24* (November 2002), 1455–1467. 2, 6

[ZMLL08] ZELEZNIK R., MILLER T., LI C., LAVIOLA J.: MathPaper: Mathematical sketching with fluid support for interactive computation. In *Smart Graphics*, Butz A., Fisher B., KrÃijger A., Olivier P., Christie M., (Eds.), vol. 5166 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 20–32. 2