Nomon:

Efficient communication with a single switch

Technical Report

Tamara Broderick

St. John's College
Inference Group
Cavendish Laboratory
University of Cambridge

This report extends the thesis submitted for the degree of:
Master of Philosophy, University of Cambridge
June 2009

Declaration

I hereby declare that my thesis, entitled "Nomon: Efficient communication with a single switch" and submitted for the MPhil in Physics, is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other university.

I further state that no part of my thesis has already been or is being concurrently submitted for any such degree or diploma or other qualification.

Except where explicit reference is made to the work of others, this thesis is the result of my own work and includes nothing that is the outcome of work done in collaboration. This thesis does not exceed fifteen thousand words in length.

Signed: _____ **Date:** _____

Tamara Broderick
St. John's College
Cambridge
June 15th, 2009

Abstract

Nomon is a new selection method that allows users to carry out computer actions normally achieved by pointing and clicking. This method has diverse applications such as text entry, mobility control, internet browsing, and drawing. Among these, text entry is arguably the most important computer-based activity an individual, disabled or otherwise, will engage in. As such, and in order to make a comparison with existing methodologies, this paper focuses on a writing application with the Nomon selection method: the Nomon Keyboard. In a comparison study, sixteen novice users wrote 35% faster with this method than with a popular commercial scanning system, and an experienced user achieved speeds of 10 words per minute. All study participants enjoyed using the Nomon Keyboard at least as much as the scanning software.

Acknowledgements

First, I would like to express my gratitude to my supervisor, David MacKay. Not only does his clarity of insight make every discussion illuminating, but his openness to unusual ideas allowed this thesis to exist in the first place.

I am indebted to the entire Inference Group for a warm welcome, helpful conversations, and constructive feedback. I am especially grateful to Philipp Hennig, Carl Scheffler, and Philip Sterne for bearing with me through equation-filled boards of various types, to Keith Vertanen for being my go-to on HCI methodology, and to Patrick Welche for sharing his extensive computing knowledge.

Thanks to all of the participants in my experimental studies. And thanks to my examiners, Per Ola Kristensson and Geoffrey Hinton, for their careful reading and thoughtful comments.

Of course, I would not be in Cambridge without the generous funding of the Marshall Scholarship. I would also like to thank St. John's College for their support.

I am very grateful to Sensory Software, Inc. for their donation of a 60-day free trial of The Grid 2 and for their helpful tech-support services. Finally, I would like to acknowledge some of the free software that has contributed to this thesis: Python (especially the Tkinter, SciPy, and PyLab modules), GNU/Linux, Dasher, Firefox, LaTeX, GIMP Image Editor, etc.

Contents

1 Introduction

Assistive technology for individuals with motor disabilities is primarily concerned with the problems of communication and movement. The goal is to allow these individuals to, in the former case, write or speak more quickly and, in the latter case, navigate autonomously. For the most severe motor impairments, an individual's motor capabilities can be as limited as a single gesture with timing information. The gesture could be controlled blinking, tongue movement, or finger movement. It is often the case that these movements are registered when a button, or switch, is clicked. Thus, "single-switch" describes the single repeated-gesture input. In what follows, we will use the term "click" as a shorthand for switch activation via an arbitrary single gesture.

Since single-switch movements are so alien to our usual means of computer interaction, software designed to interpret these clicks tends to be very specialized; for example, a program may focus on writing with a pre-defined alphabet, playing a specially designed game, or manipulating a picture in pre-defined, stereotyped ways. Not all single-switch systems are so limited. Software such as The Grid 2 (Sensory Software International Ltd., 2008), EZ Keys (Words+, Inc., 2000, 2004), and One-Button Dasher (MacKay *et al.*, 2004) all allow the user to customize the set of selection options. However, it is often the case that customizing the interface is difficult to accomplish with a single switch. A caretaker or other able-bodied individual may need to step in to help. The Gnome Onscreen Keyboard (GOK) (Sun Java Desktop System Documentation Team, 2004) goes a step further and eliminates the middle man in the customization process. When a new application is invoked, GOK is able to generate a new set of selection options "on the fly" so that the user can immediately access the application.

Nonetheless, these systems still have limitations. Consider One-Button Dasher; the language in which the user writes is fully customizable, and the program could be used to write to a command line. But Dasher is a writing program, not a general selection program; it cannot translate a single-switch input into graphical user interface (GUI) interaction. The Grid 2, EZ Keys, and GOK enable GUI interaction. However, the set of selection options must be arranged in a grid format. And if the software is accessing another application, the option grid will be separate from the application's interface. This kind of interaction is not cumbersome for navigating through a file system or writing. But it is less well-adapted to drawing, certain forms of game-play, and web browsing. In any case, it is less direct than normal point-and-click selection.

In contrast, Nomon—the program described by this thesis—can be used to choose any point on a two-dimensional screen. Selection of options in a grid, then, is just a special case of this more-general selection method. Theoretically, Nomon allows a single-switch

user to carry out any computer actions normally achieved by pointing and clicking. This functionality opens up drawing, game-play, and web browsing. But the method is also suited to writing, operating system interaction, and mobility control.

In what follows, we focus on a writing application for two reasons. First, verbal communication is arguably the most important task any individual performs on a daily basis and thus the most important application for a single-switch user. Second, while Nomon serves a more general purpose than writing, most existing single-switch software is specialized for writing (for reason one); thus, writing allows a comparison of the Nomon selection method with other methods.

With these considerations in mind, we begin by describing two very different single-switch writing methods in Section 2.1. Section 2.2 outlines an information-theoretic framework in which to consider the information rate of single-switch input methods. Having provided a brief overview of current single-switch communication research and a mathematical model of optimal performance, we proceed to describe the Nomon method—its appearance, functionality, and application—in Section 3.

The three experimental studies in the subsequent sections compare the Nomon Keyboard application to the most common single-switch writing method (scanning), described in Section 2.1.1. We begin by looking at how able-bodied individuals interact with the Nomon selection method in Section 4 in order to best choose fixed parameters of the system and also to inform our mathematical models. The principal study (Section 5) is a 16-subject, balanced, within-subjects comparison of Nomon and scanning for novice users. The final study (Section 6) is a sensitivity analysis of the adjustable parameters from the comparison.

We end with descriptions of further Nomon applications in various stages of development (Section 7).

2 Background

For the reasons above, we concern ourselves with the problem of generating text with a single-switch input. To be more explicit, we take a single-switch input to be an ordered sequence of activation times. The user is assumed to control the time of activation of the switch, but not the time of release. This definition excludes the well-known Morse code method from our discussion as this method uses duration to distinguish two separate clicks; indeed, some individuals write with Morse code directly from two (Glennen and DeCoste, 1997) or more (Hsieh and Luo, 1999) switches.

In the following, we consider other prominent single-switch writing interfaces and, to help us evaluate performance, a mathematical model for the information rate attainable

with a single switch. In the literature, performance is traditionally assessed by measuring the writing rate as words per minute, where a word is defined as five consecutive characters. Other quantities of interest include proportion of errors, e.g. errors per word, and number of clicks required for output, e.g. clicks per character.

2.1 Writing Interfaces

The following two writing methods both require just one switch, in the strict sense defined above. Further, unlike Morse code, they require no special memorization or knowledge and can be used immediately, with minimal instruction, to generate text.

2.1.1 Scanning

Scanning is by far the most popular single-switch method of writing. Only two methods for single-switch text entry are discussed in *The Handbook of Augmentative and Alternative Communication* (Glennen and DeCoste, 1997): scanning and Morse code. Morse code is mentioned only briefly after scanning. The vast majority of single-switch commercial writing software operates via scanning in some form.

Method Typically, scanning presents options in a rectangular grid. Subsets (e.g. rows) of the options are highlighted sequentially, with the set being updated at regular intervals. Upon a click, the set of highlighted subsets change. If an option subset of size one was highlighted at click time, that option is selected.

The most popular subset-selection mechanism is row-column scanning for some roughly balanced grid height and width. In this case, the software begins by highlighting the rows sequentially, starting with the top row and iteratively moving to the next row at fixed time intervals (a.k.a. *scanning delays*). To select a row, the user clicks when a row is highlighted. Then the options in the selected row are highlighted one at a time; that is, starting with the option in the first column, the highlight iteratively moves to the next column at the same fixed time intervals as before. The user clicks while an option is highlighted to select that option. Thus, a selection always requires exactly two clicks.

Some scanning interfaces will cycle through rows or columns only for a fixed number of iterations; after this number is reached, the system times out. Then the user must supply a click to begin highlighting again.

Word prediction may be added to a scanning interface. While most cells in the grid will typically contain a fixed character or set of characters, a word-prediction cell will display a whole word that may depend on what has been written thus far. These cells can be accessed in the same way as any other cell. An example scanning grid appears in

Figure 1: One of the pre-packaged scanning grids for writing in The Grid 2 commercial software (Sensory Software International Ltd., 2008). The four blank cells on the far left are designed to hold word completions, which dynamically adjust according to what has been written. The characters and functions in the remaining cells are fixed.

Figure 1. The tan and blue cells on the right are fixed, but the white cells on the left dynamically display words as appropriate during writing.

Performance Given the limited nature of a single-switch input, we expect writing speed for a ten-fingered, able-bodied individual to be much slower with any single-switch interface than with a standard QWERTY keyboard. We explore these limitations more rigorously in Section 2.2. Following our intuition on the matter for now, the principal concern of such a writing application must be to maximize writing speed.

Table 1 summarizes a variety of studies that have examined able-bodied users generating text via a scanning interface. We first consider text entry rate in words per minute; in particular, we are interested in the total number of words (where "word" means any sequence of five characters) in the output text divided by the total time to generate that text. While the range of reported text entry rates in words per minute initially seems large, some care must be taken with the highest and lowest entry-rate values in the table. The entry rates from Koester and Levine (1994) include "timing errors, selection errors, and error corrections" in the character count. A timing error occurs when the user does not select the desired option on the first opportunity (i.e. the scanning highlight passes it by). A selection error occurs when an option besides the desired one is selected. Error correction involves selecting a special backspace character and then writing a new character. We expect this measure to be much higher than a standard words-per-minute writing rate. Nonetheless, the Koester and Levine (1994) study still reveals a relative effect; including word prediction does not improve writing speed. The authors suggest that including word-prediction options may decrease writing rate as a result of the time users spend searching word lists.

Reference	System				Performance		
	Pred	Mx	Sel	Delay	Subj	Entry	Error
Koester and Levine (1994)	·	freq	RC	adju	3	[8.5][1,2]	[≤ 5][3]
Koester and Levine (1994)	√	freq	RC	adju	3	[8.2][1,2]	[≤ 5][3]
Damper (1984)	·	freq	RC	0.5	thy	≤ 6.5	0
Simpson and Koester (1999)	·	freq	RC	adap	4	[4.5][4]	[≤ 5][3]
Evreinov and Raisamo (2004)	·	freq	3	adap	8	[≈ 4.2][5]	0.6–33
Simpson and Koester (1999)	·	freq	RC	adju	4	[4.1][4]	[≤ 5][3]
Szeto *et al.* (1993)	·	freq	RC	NA	4	[3.2][6]	0.5–4
Baljko and Tam (2006)	·	freq	RC	≥ 0.75	12	[3.1][5]	[6.6][7]
Venkatagiri (1999)	·	freq	RC	1.0	sim	2.5	0
Venkatagiri (1999)	·	alpha	RC	1.0	sim	2.1	0
Baljko and Tam (2006)	·	freq	H	≥ 0.75	12	[1.8][5]	[18.5][7]

Table 1: A summary of studies addressing the performance of able-bodied individuals using a scanning interface to write. Each row corresponds to a single (experimental, simulated, or theoretical) result, and results are ordered by reported text entry rate. A legend is provided in Table 2. Additional notes on the entry and error rates appear here: [1] "Timing errors, selection errors, and error corrections" are included in the character count. [2]The time elapsed does not include time to read the prompt. [3]Incorrect selections, both corrected and uncorrected, count as errors; the number of errors is divided by number of selections, not written characters. [4]Only includes correct output characters in the character count. [5]A special "stroke" beside the single switch was used to reach additional functions (such as backspace) or skip ahead in the selection hierarchy. [6]No error correction allowed. [7]Definition of error not given.

Column	Symbol	Meaning
Pred	$\surd$	Word predictions available for selection in scanning grid.
	$\cdot$	No word predictions available.
Mx	freq	Scanning grid arranged according to unigram frequencies.
	alpha	Scanning grid arranged alphabetically.
Sel	RC	Row-column highlighting.
	H	Huffman highlighting (Baljko and Tam, 2006).
	3	3x1 mutable grid (Evreinov and Raisamo, 2004).
Delay	#	Fixed scanning delay in seconds.
	adju	Scanning delay could be adjusted by the user.
	adap	Scanning delay automatically adapted to the user.
	NA	Scanning delay not specified.
Subj	#	Number of experimental subjects from which the performance measures shown here were derived.
	thy	Performance determined from theory.
	sim	Performance determined by computer simulation.
Entry	#	Mean entry rate in words (word = 5 characters) per minute.
	$\leq$ #	Upper bound on entry rate.
	$\approx$ #	"Most of the participants achieved" (Evreinov and Raisamo, 2004) this speed.
Error	#	Mean percentage of incorrect characters in the output text.
	$\leq$ #	Upper bound on error rate.
	#–#	Range of mean errors across trials (Szeto *et al.*, 1993) or characters (Evreinov and Raisamo, 2004).

Table 2: A legend for the summary of scanning performance studies in Table 1.

Conversely, the Venkatagiri (1999) study has particularly low numbers, perhaps due to the scanning delay being set to a full second. These results were derived by simulation, and they represent a user who is completely familiar with the frequency ordering and never makes mistakes. Therefore, the quoted speeds would be twice as high at a delay of 0.5 seconds. Again, though, the study allows us to assess a relative effect: how much does a frequency-ordered matrix improve writing speed performance. The speeds shown in the table suggest a 19% faster writing speed can be achieved with a frequency-ordered matrix under perfect conditions. Given the idealized simulation used to attain this number, 19% may represent a maximum improvement for more realistic users.

Another performance measure of interest to users for whom movement is difficult is the number of clicks per character. Any grid arrangement for row-column scanning without word completion will necessarily require at least 2 clicks per character since two clicks are necessary to make a selection. Some row-column interfaces require a further third click to begin the sequential highlighting process after each selection. The number of clicks per character will also be increased by mistake corrections and clicks to start scanning again after time-outs.

There is no relative advantage for different matrix orderings in this measure. However, we do expect the inclusion of word prediction to reduce the number of clicks per character. For one system (Koester and Levine, 1996), the authors report a click reduction of 28% from including word prediction. In (Koester and Levine, 1994), selection reduction was reported at 36.4%. In another study (Lesher *et al.*, 1998), the authors consider a variety of grids, some including dynamic letter arrangement as well as word prediction. The authors measure, via simulation, the total number of scanning iterations and clicks combined, and they report savings in the combined measure as high as 57.7%. Since it is impossible to design a grid that contains every alphabetic character and has an expected wait of fewer than two iterations, these studies suggest that a rate of around 1.3 clicks per character (calculated as 65% of our lower bound on click rate) is required in row-column scanning.

2.1.2 One-Button Dasher

A more recently developed single-switch writing method is based on the information-efficient Dasher (Ward *et al.*, 2000; Ward and MacKay, 2002) method for writing with continuous gestures.

Method The original Dasher program (Ward *et al.*, 2000) works by zooming in the direction of the desired text string along a line in which all possible text strings are arranged alphabetically; the direction for zooming is determined by the placement of a pointer, which is accomplished with a continuous movement, e.g. of a mouse. In a two-

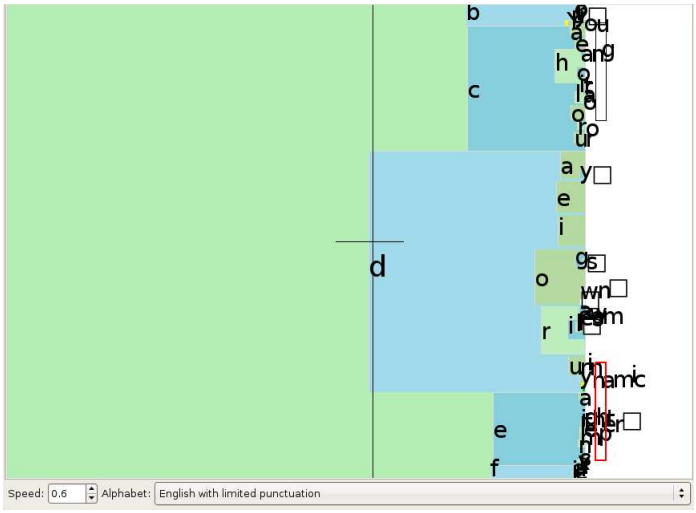

Figure 2: The zooming display for One-Button Dasher's "one-click" mode. The display zooms into the upper right or lower right corner continuously, with the direction of zoom indicated by the red rectangle. The zoom direction is toggled by clicking.

button dynamic version of Dasher (MacKay, 2007), the user adjusts the zooming direction by clicking the first button precisely when the desired text string is near an indicator at the top of the screen and clicking the second button precisely when the desired text string is near an indicator at the bottom of the screen.

While the preceding Dasher versions do not fit our single-switch input specification, One-Button Dasher (MacKay *et al.*, 2004; MacKay and Ball, 2006; MacKay, 2007) requires a single click with no duration information. There are two principal modes in which Dasher can be operated with a single switch. In "one-click" mode (MacKay and Ball, 2006), zooming is always toward the top or bottom of the screen, and clicking toggles which direction is currently used. This mode is shown in Figure 2. In "two-click" mode (MacKay, 2007), the two different switches of two-button dynamic Dasher are replaced by a single switch with two different activation patterns. Thus, to choose the indicator at the top of the screen, the user clicks, waits for time t_{upper} (typically, a very short time), and then clicks again. The same procedure occurs at the other indicator, only now with time t_{lower} waiting period. Time t_{lower} is also very short but different enough from t_{upper} so as to easily distinguish the upper and lower click sequences.

Performance There is no performance information yet available for the two-click mode of One-Button Dasher. The performance measures that exist for one-click mode rely on the user having access to a second "cheat button" to reverse zooming and thus undo mistakes (MacKay and Ball, 2006). Under these conditions, an expert in continuous-motion Dasher achieved approximately 10 words per minute and 0.4 clicks per character

in one-click mode, far superior to the same measures for row-column scanning cited above in Section 2.1.1.

In the latest Dasher software release (4.10.1), reverse zooming can be accomplished with a cheat button but also with a long-hold or quick triple-click. Only when this last option is used does the interface fully conform to our single-switch definition. However, it may be the case that the text-entry and click rates are somewhat different in the adjusted interface.

2.2 Information-Theoretic Interpretation

The Dasher software, including the one-button version, is designed to be information efficient in that it comes close to the maximum communication rate theoretically attainable with a single switch. To find the information rate for single-switch input, we begin here by considering the discrete, noiseless approximation of the information rate examined in the paper introducing One-Button Dasher (MacKay *et al.*, 2004). We then develop a more detailed approximation introducing continuous time and click-time uncertainty.

2.2.1 Discrete, Noiseless Model

MacKay *et al.* (2004) start with two assumptions about a user's clicking ability: (1) that a user requires a recovery time D after making a click and (2) that, for some precision g, a user clicks within $[t - g/2, t + g/2]$ when t is the desired click time. The results of the study in Section 4 validate these assumptions; participants in this study exhibited a relatively long necessary time between clicks (D) and a relatively short noise-model standard deviation ($\propto g$).

Finding the maximum information rate that can be generated by such a user can then be solved in the same way as Exercise 6.18 on p. 125 of MacKay (2003). In particular, a series of clicks can be associated with a string of characters from a binary alphabet, whose two symbols have durations D and g, respectively. The method that optimizes the information rate uses the first symbol with some probability B and the second symbol with probability $1 - B$.

We must still find the best value of B. The probability that the user waits ng time after the initial D recovery time takes the form of a geometric distribution: $p(n) = (1-B)^n B$, $n \in \{0, 1, 2, \ldots\}$, with parameter B. The entropy of p is $H(p) = H_2(B)/B$, where $H_2(B) = B \log_2 (B^{-1}) + (1 - B) \log_2 ((1 - B)^{-1})$ is the usual binary entropy function. If we assume that a user clicks in the middle of a block of length g, the average time between clicks is $L(p) = D + g/2 + g \cdot (1 - B)/B$. Alternative assumptions might be that the user clicks at the beginning of a block ($L(p) = D + g \cdot (1 - B)/B$) or the end of a

block ($L(p) = D + g + g \cdot (1 - B)/B$). Call these three assumptions the middle-click, early-click, and late-click assumptions. The information rate $R(B)$ is the entropy divided by the expected time between clicks. Under the middle-click assumption, the derivation in Appendix A yields

$$R(B) = \frac{H_2(B)}{BD + (1 - B/2)g}.$$

The maximum information rate is $\max_B R(B)$.

2.2.2 Continuous, Noisy Model

When the click distribution around the desired time is normal, as is approximately the case in Section 4, it is not immediately obvious what proportionality constant to use to find g from the standard deviation in the formulation above. Therefore, we develop the following model, similar to the version above, that now accounts for the normal noise distribution directly. We assume that the user's click time follows a normal distribution with standard deviation γ around the desired click time. Here we let this noise distribution have mean zero. Motivated by the same study (Section 4), we further assume that the user requires a recovery time D after making a click, as in the previous model.

In this continuous and noisy case, we take $I(X;Y)$ to be the continuous mutual information between the input variable X (when the user wants to click) and the output variable Y (when the user actually clicks). With $L(Y)$ as the expected time between clicks, the information rate is $I(X;Y)/L(Y)$. For the above assumptions, we again have that the users clicks with zero probability at times before D; for times beyond D, the optimal input density can be derived with variational methods and takes the form of an exponential distribution: $f(t) = \beta e^{-\beta t}$ with some parameter β. For this density, the derivation in Appendix A yields the information rate $\rho(\beta)$.

$$\rho(\beta) = \frac{-\log_2\left(\sqrt{2\pi}\gamma\beta\right) + (2\ln(2))^{-1}\left(1 - \gamma^2\beta^2\right)}{D + \beta^{-1}}$$

It follows that the maximum information rate is $\max_\beta \rho(\beta)$.

Comparing this approximation with the discrete, noiseless one confirms the intuition that an appropriate choice of g is the 95% interval of the normal distribution ($g = 2 \cdot 1.96\gamma$). We also confirm that, in the discrete approximation, it is most appropriate to assume the user clicks in the middle of the block. With $g = 3.92\gamma$ and the middle-click assumption, the maximized information rate under both models is almost identical across a range of D values (left panel of Figure 3). We confirm that this is an appropriate proportionality constant and that the middle-click assumption is best in the right panel of Figure 3. Here, we plot the value of g/γ that minimizes the squared error between

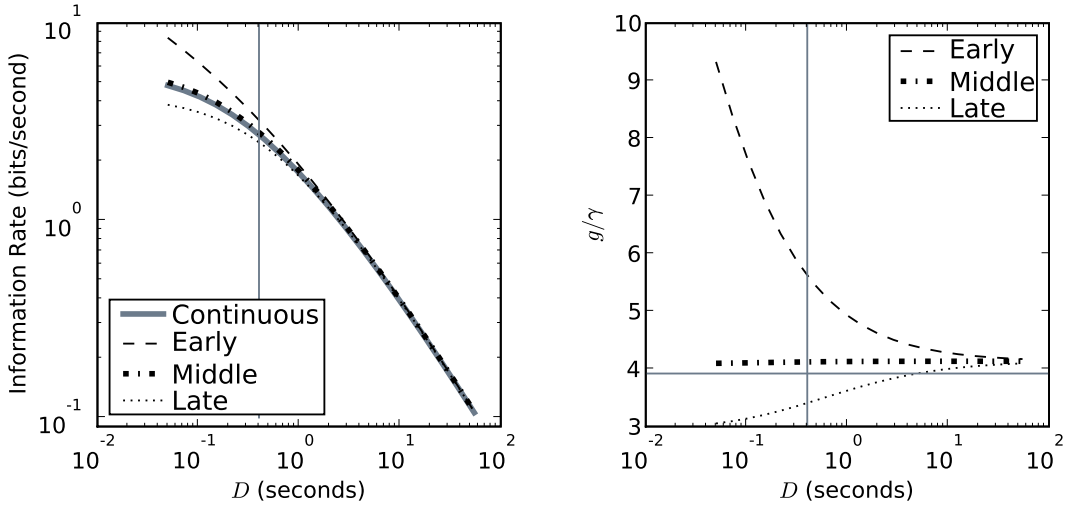

Figure 3: *Left*: Maximized information rates for continuous and discrete approximations as a function of recovery time D, with $g = 3.92\gamma$ fixed. Three discrete approximations are shown: the approximations under the early-click, middle-click, and late-click assumptions. Based on the novice in Section 4, we set $\gamma = 0.06$ and plot a line at $D = 0.4$. *Right*: For a fixed recovery time D and $\gamma = 0.06$, we find the g that minimizes squared error between the discrete and continuous models: $[\max_\beta \rho(\beta) - \max_B R(B)]^2$. The squared-error-minimizing g (scaled by γ^{-1}) is shown for the discrete approximation under the early-click, middle-click, and late-click assumptions. Again, we plot lines at $D = 0.4$ and at $g/\gamma = 3.92$ for comparison.

the two model results, $[\max_\beta \rho(\beta) - \max_B R(B)]^2$, across the same range of D values and across the three click-time assumptions. The curve corresponding to the middle-click assumption remains close to $g/\gamma = 3.92$ across the full range of D values.

2.2.3 Discrete, Noisy, Periodic Model

The previous models assume an arbitrary single-switch input. However, in practice, single-switch interfaces often exhibit periodicity (Sections 2.1.1 and 3). That is, for some period T in the interface, clicking at time y is the same as clicking at y plus any integer multiple of T. Therefore, we now assume that X (the desired clicking time) has support on $[0, T)$. We use elements of Sections 2.2.1 and 2.2.2 in our user model. As usual, we assume that the user requires a recovery time D after making a click. After time D, we divide the real line into adjacent intervals of length T. Each interval of length T we further subdivide into blocks of length $g = 3.92\gamma$; as an approximation, we assume this subdivision can be done exactly. Given a particular T-length interval, we assume that the user always clicks in the desired block of length g. Then we form a geometric distribution over the length-T intervals in the following way. In Section 2.2.2, we noted that the user

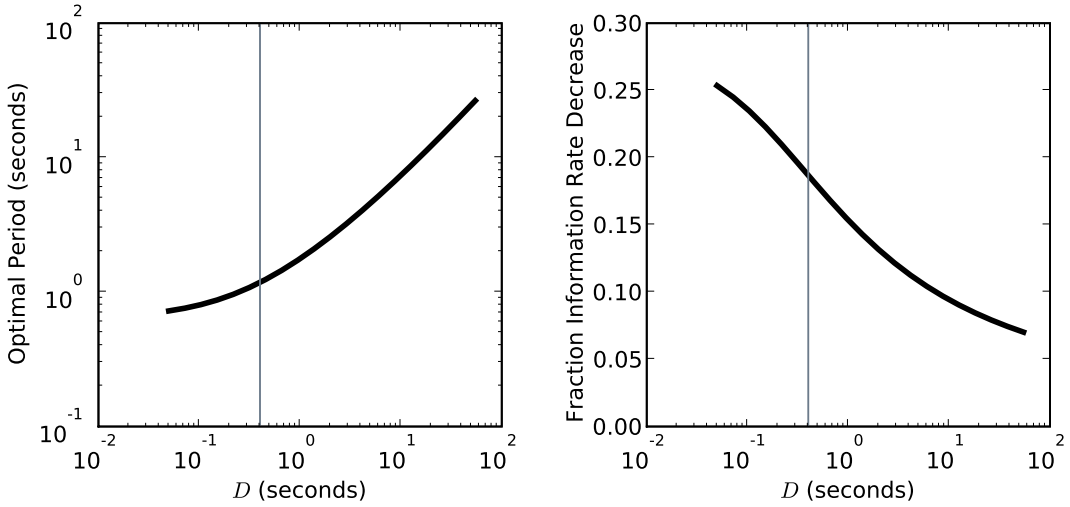

Figure 4: *Left*: Optimal period T in the discrete, noisy, periodic model as a function of D with $\gamma = 0.06$ fixed. *Right*: For each recovery time D and $\gamma = 0.06$, we find the highest information rate $\hat{R}^*$ in the discrete, noisy, periodic model and the highest information rate ρ^* in the continuous, noisy model. We plot the information rate loss due to periodicity, $1 - \hat{R}^*/\rho^*$, as a function of D.

actually clicks within the desired length-g interval with 95% probability. Therefore, we take the geometric distribution success probability of clicking in a particular length-T interval to be 95%.

For these specifications, the optimal input distribution over the $\approx T/g$ blocks of length g in $[0, T)$ is uniform. And the information rate, derived in Appendix A, is

$$\hat{R}(T) = \frac{\log_2(T/g)}{D + T/2 + T \cdot 0.05/0.95}$$

The maximum information rate is $\max_T \hat{R}(T)$. The optimizing period for a range of D values (with $\gamma = 0.06$ fixed) is shown in the left panel of Figure 4. For the novice user, we see the optimal period length is approximately 1 second. The right panel of Figure 4 shows the decrease in the optimal information rate due to assuming a periodic interface (rather than the more general interface in Sections 2.2.1 and 2.2.2). In particular, this quantity is calculated by finding the optimal information rate $\hat{R}^*$ in the discrete, noisy, periodic model and the optimal information rate ρ^* in the continuous, noisy model. The fractional decrease in information rate is then $1 - \hat{R}^*/\rho^*$. Since periodicity imposes an additional constraint on the model, the highest information rate here is always lower than the highest rate in the previous models.

3 Nomon

The methods described in Section 2.1 can all be applied to writing but feature varying degrees of generality outside of text-entry applications. While One-Button Dasher is almost exclusively a writing interface, it is easily adapted to output any character sequence, as long as the set of input characters can be ordered consistently (e.g. alphabetically). Scanning generalizes to allow selection of any options arranged in a grid. While some software requires grid formation by able-bodied individuals, systems such as the Gnome Onscreen Keyboard can generate grids for new applications automatically.

Into this family of single-switch interfaces, we introduce Nomon. At its most general, Nomon is a method for selecting any points on a two-dimensional computer screen. We here consider it in its full generality and then illustrate how this selection method may be applied to create a writing interface.

3.1 General Nomon

We begin by examining the most general version of Nomon, which serves as a substitute for any point-and-click activity on a computer screen. After considering the user experience, we look at the mechanism that allows Nomon to function. All Nomon applications share the features described here.

3.1.1 User's View

The trademark of any Nomon application is a set of small *clocks*. These clocks serve as the selection mechanism and may be superimposed on any background. In particular, each clock usually appears alongside its corresponding screen option. For instance, in Figure 5, each clock appears above a numbered option. In the writing application described below (Section 3.2) and shown in Figure 8, one clock appears immediately to the left of each letter on the screen. In any Nomon application, the user selects a screen option by selecting its corresponding clock in the following way.

Operation On each clock, there is a black *moving hand* rotating at a fixed speed. While the speed of rotation is shared by every clock, the angular position of the moving hand at any given moment is different across the clocks on screen. Every clock also has a red line indicating *noon*. In order to select an option, the user begins by finding its corresponding clock. Next she clicks, as precisely as possible, when the moving hand is at noon. Every clock *offset*—the initial counterclockwise angle from noon to the moving hand—will reset. This action is repeated until the desired clock—and, by extension, the desired option—is shown to have been selected.

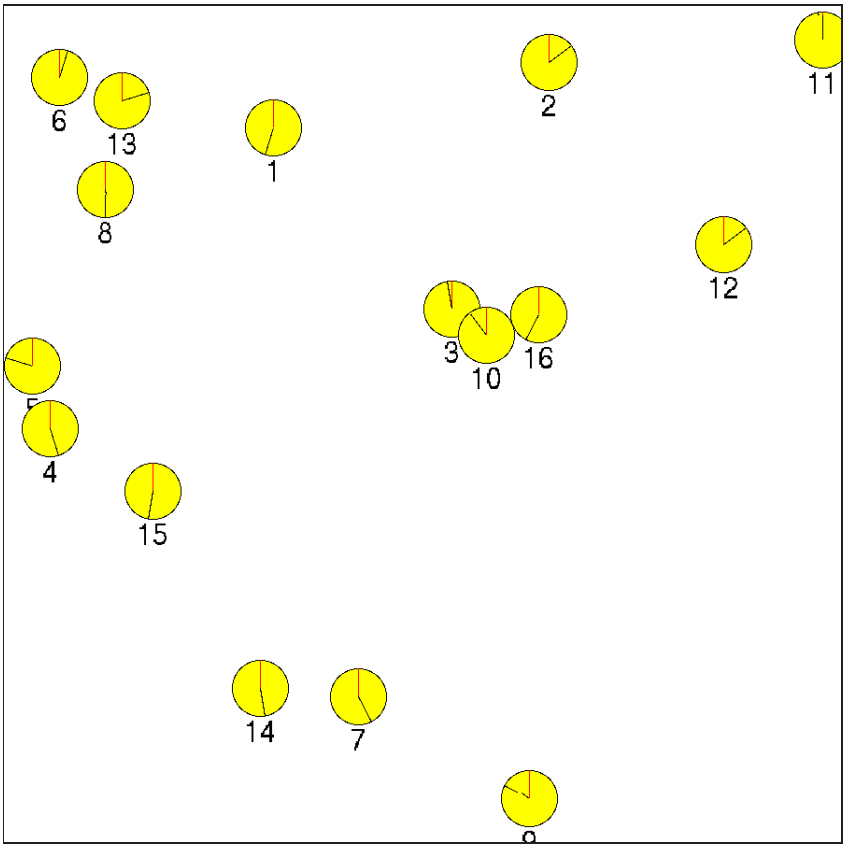

Figure 5: A Nomon selection scheme for numbered options on a screen. Here, Nomon clocks appear immediately above their corresponding options.

The number of clicks required to make a selection is variable as it depends both on how precisely the user clicks and on how many alternatives are available. This flexibility allows efficient operation for a wide range of users and inputs. The interface adapts to extremely precise users by allowing them to click very few times to make a selection. It also allows less-precise users extra clicks so that mistakes are unlikely to lead to errors in the final choice. The desired probability of error is one of the adjustable parameters of the software.

The required number of clicks may also depend on the set of options; indeed, we often have prior information about what is likely to be chosen. For instance, in a writing interface, we usually expect that the letter `a` is more likely than the letter `x`. By incorporating this information (Section 3.2.2), we allow users to write more quickly.

Display Cues Due to the variability in the number of clicks to make a selection, it is important that the Nomon interface clearly demonstrate when a selection has been made. Display cues may take various forms, including visual and spoken notifications. For instance, upon any selection, the entire visual interface may flash briefly. The selected

option may be highlighted to indicate its choice. Spoken output may also be produced; in the case of a keyboard application, hearing the chosen letter or word spoken frees the user from spending time checking the written output.

Display cues can also provide information about progress toward a selection. Consider the posterior probability of a given option—equivalently, of its corresponding clock—given the history of clicks since the last selection. In Nomon, the color of a clock's face indicates the posterior probability of that clock, computed explicitly in Section 3.2.2. Specifically, clocks with posterior probability above a certain cut-off (more likely clocks) are colored yellow, and clocks with probability below this cut-off (less likely clocks) are colored white. Immediately after a selection, there is no click information for the next selection yet, so a clock's face color indicates that clock's prior probability. Clock face colors serve as a progress indicator. As clicking proceeds, typically the desired clock turns or stays yellow, and the other clocks increasingly become white. A poorly-timed click may cause the desired clock to turn white. In any case, a white clock signals that more careful clicking is required to choose this option.

Controls Since Nomon is a method that enables arbitrary point-and-click actions, interface controls can be selected in exactly the same way as other options. Of particular use, for instance in writing applications, is an undo function. By putting an option for `Undo` somewhere on the screen and assigning it a corresponding clock, the last action may be undone simply by selecting this clock. Program environment options such as `File`, `Edit`, and `Minimize` may be selected in the same way. Parameters, such as the rotation speed of the clocks, may also be adjusted with Nomon. For instance, we can establish a sliding scale for each parameter; one clock pushes the scale up (e.g. higher clock rotation speed) and another clock pushes the scale down (e.g. lower clock rotation speed).

3.1.2 How It Works

Having described the key points of the user's experience of Nomon, we now describe how Nomon functions, particularly how it decides when a election has been made. Below we explore the details of how the clocks work.

Clock Design and Schedule On a high level, the Nomon clock is designed to take advantage of a user's clicking precision. Motivated by the single line-segment into which One-Button Dasher zooms, we would like to place such a line-segment (or other selection device) at every desired selection location on a screen. In order to make the selection space continuous, we wrap the line segment around and thereby arrive at the clock shape. Further, fixing the noon marker while updating the clock angular offsets after each click

allows us, effectively, to completely reorder any set of selections without any additional cognitive load for the user. While row-column scanning could theoretically benefit from grid rearrangement after every selection, such a dynamic grid is very difficult to use in practice (Darragh *et al.*, 1990).

The clock shape and moving hand make the Nomon interface periodic in its operation. In Section 2.2.3, we found that the optimal input click distribution across the period of a periodic single-switch interface is uniform. We also found that, when the input click distribution in uniform, there exists an optimal rotation period. As a simple measure to approximate a uniform clicking distribution over the clocks, we set each clock's speed to be uniform throughout the period. Given that there exists one optimal period for a periodic single-switch interface, we set all of the clocks to rotate at the same average speed. Thus, all clocks rotate at the same speed at all times.

Selection Decisions Consider that the user is about to make a series of clicks in pursuit of a single option. Before the user begins, we have some prior probabilities over the possible clock choices. For instance, in a writing application, we may expect certain words to be more likely. Or we may expect users to choose `Undo` often. We assign a set of probabilities reflecting our beliefs to each clock c of C clocks on the screen: $p(c)$. We also assume we know the probability distribution of the user's r^{th} click time t_r around noon: $p(t_r|c)$. This probability distribution depends on the chosen clock, which defines the location of noon.

After the R^{th} click, we may calculate the posterior probability of each clock given the clicks thus far using Bayes' theorem: $p_{c,R} = p(c|t_{1:R}) \propto p(c) \prod_{r=1}^{R} p(t_r|c)$. Under a stricter use of Bayes' theorem, the times at which the user did not click are also a (weak) source of information about the desired clock, but we ignore this source here. Ordering the probabilities $p_{c,R}$, we can find the highest probability $p_{(C),R}$. It is natural to set some error rate p_{error} in advance and declare that the highest-probability clock, (C), is the winner when its posterior probability is higher than $1 - p_{\text{error}}$. The chosen clock is then communicated to the application (e.g. keyboard, drawing pad) so that it may update accordingly. For instance, a drawing program may need to draw a new line after a point is selected. Then the process begins again and allows the user to select among the next set of options.

In practice, though, we store the unnormalized log probabilities for each $p_{c,R}$. Therefore, the error criterion just described would require exponentiating every stored value and summing over the results. To find a criterion that requires less computation, first define the odds $\alpha = (1 - p_{\text{error}})/p_{\text{error}}$. Above, we proposed to declare clock (C) the winner when $p_{(C)} > 1 - p_{\text{error}}$. It would be equivalent to declare (C) the winner when

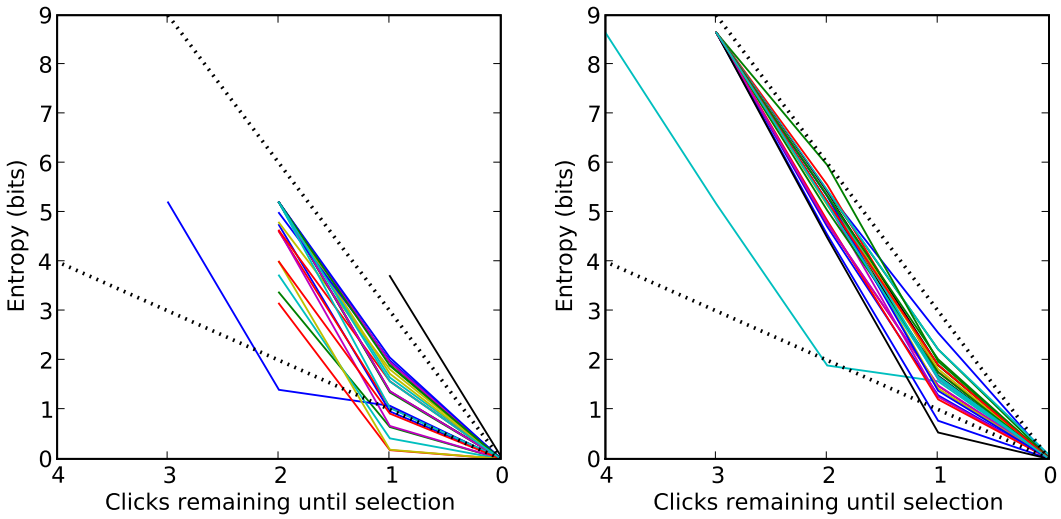

Figure 6: Each panel shows the entropy of the estimated probability distribution over clocks as a function of clicks remaining to selection for a particular Nomon application. Each solid line represents a single selection process. Dotted lines decreasing to zero at respective rates of 1 (lower) and 3 (upper) bits per click are illustrated for reference. *Left*: 25 selections on the Nomon Keyboard (Section 3.2): 30 clocks, non-trivial prior $p(c)$, clock period 2.0 seconds, switch input from joystick button. *Right*: 25 selections on a Nomon drawing application (Section 7.1): 401 clocks, uniform prior $p(c)$, clock period 2.0 seconds, switch input from space bar. Data was generated by the experienced user (the author) from the studies below (Sections 4.1.1, 5.1.1, 6.1).

$p_{(C),R} > \alpha \sum_{c \neq (C)} p_{c,R}$.

Now consider an approximation to this strict criterion where we substitute $p_{(C-1),R}$ for the sum: $p_{(C),R} > \alpha p_{(C-1),R}$. This new criterion can be evaluated directly by finding the difference between the two highest (unnormalized) log probabilities. Assuming the $p_{c,R}$ distribution is well-behaved, α still corresponds to some desired lower bound on the odds of correctly making the desired selection. We check that this two-top-score criterion behaves as desired below. As in the strict case, we can choose the selection error probability by choosing the value of α. Moreover, we can vary α according to the context, or from clock to clock; for example, a higher α value can ensure greater safety for critical actions.

In order to help $p_{(C),R}$ and $p_{(C-1),R}$ diverge and satisfy our error criterion as quickly as possible, we choose the relative clock offsets carefully according to the following heuristic. The most probable and second most probable clock start with moving hands at 180° angular separation. The next two most probable clocks start at 90° and 270° relative to the most probable clock, and so on.

Entropy and Scaling We examine the practical implications of these design specifications for Nomon operation in Figures 6 and 7. The evolution of the entropy of the $\{p_{\cdot,R}\}_c$ distribution as clicking progresses is shown for some example Nomon applications in Figure 6. The lines in the left panel chart the entropy changes during the selection processes for 25 consecutive selections with the Nomon Keyboard (Section 3.2). The non-trivial prior $p(c)$ of the Nomon Keyboard (Section 3.2.2) is evident in the different starting entropies of the lines. In this set of selections, reaching the stopping criterion usually requires two clicks and sometimes requires one or three. Similarly, the right panel of Figure 6 shows 25 lines, one for each of 25 consecutive selections from a Nomon drawing application (Section 7.1). The drawing application features more than 10 times as many clocks as the keyboard. In this case, the starting entropies are all equal due to the uniform prior $p(c)$. All selections but one were made with exactly three clicks.

A quick calculation shows that, with a uniform prior, the total entropy of the initial clock probability distribution scales logarithmically with the number of clocks: $\sum_{c=1}^{C} C^{-1} \log_2 C = \log_2 C$. Comparing the highest initial entropies (since the uniform distribution maximizes entropy) in the left panel of Figure 6 to the initial entropy in the right panel, we see a difference of about 3.5 bits. This agrees with $\log_2(401/30) = 3.7$ as expected. In Figure 6, we also see that the entropy decrease due to a click is roughly constant—except on the final click of a selection, when the achievable decrease is bounded by the remaining distributional entropy. If the entropy decrease per click is, in fact, constant, then the number of clicks should scale logarithmically with the number of clocks.

Selection Error Figure 6 confirms that our stopping criterion takes effect when the entropy of the clock distribution has been reduced dramatically; it remains to check in more detail that this criterion elicits the desired behavior from the Nomon system. Figure 7 illustrates how our choice of the top-two-scores stopping criterion $p_{(C),R} > \alpha p_{(C-1),R}$ compares to the stricter criterion $p_{(C),R} > \alpha \sum_{c \neq (C)} p_{c,R}$ in practice. In particular, we take $p_{(C),R}/p_{(C-1),R}$ as an estimate of the odds of a correct selection in the former case, and we take $p_{(C),R}$ as an estimate of the probability of a correct selection in the latter case. For α set to 99, the probability of error estimated with $p_{(C),R}$ and $p_{(C-1),R}$ is always less than 0.01 by construction. The fact that it can often be much less than 0.01 confirms that any α setting provides only an upper bound on the error rate. In this case, it happens that the average error estimate calculated just from $p_{(C),R}$ and $p_{(C-1),R}$ is 0.001, a full order of magnitude less than the bound.

As expected, the point error estimates calculated just from $p_{(C),R}$ and $p_{(C-1),R}$ are always less than the stricter error estimate calculated using the full distribution. While the top-two-score estimates are always less than 0.01, full-distribution estimates reach

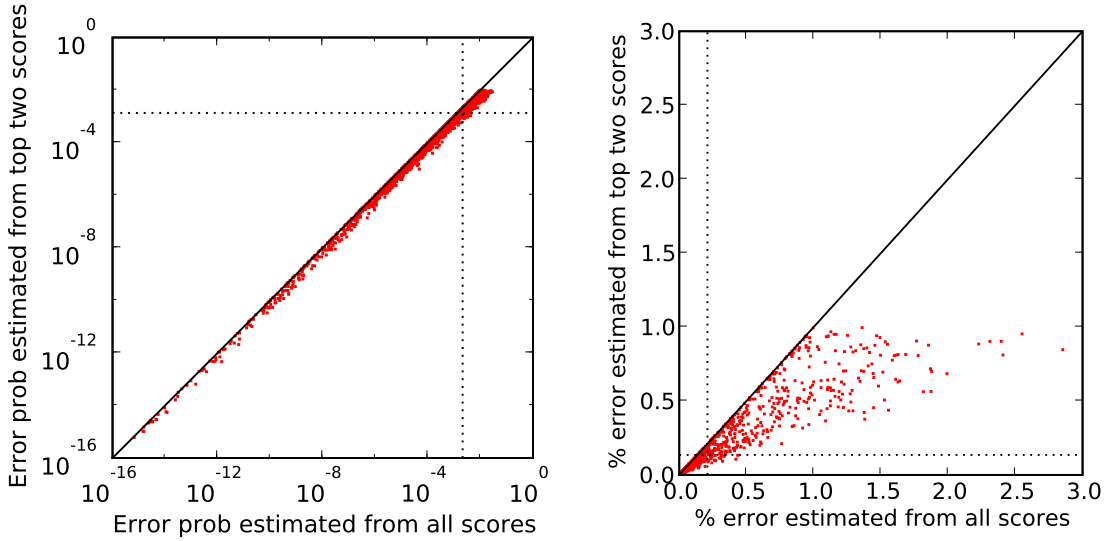

Figure 7: The same plot is shown in both panels but with a different axis scaling in each. The two coordinates of each data point correspond to results from two different methods of estimating the probability of error; both estimates were calculated after a selection was made. Points represent 1,714 selections using the top-two-scores criterion with $\alpha = 99$ (0.01 error probability). Dashed lines give the average error estimate of each method, and a solid line with unit slope is illustrated for reference. Data was generated by the experienced user from the studies below (Sections 4.1.1, 5.1.1, 6.1); the interface was the Nomon Keyboard with non-trivial prior $p(c)$, clock period 1.1, and switch input from the joystick button.

nearly 0.03. Nonetheless, the average full-distribution error estimate is, at 0.002, still about an order of magnitude lower than the stated 0.01 bound.

We find that the empirical error rate agrees roughly with the estimated rate and is well within the stated bound. In generating the data for Figure 7, selecting a special `Undo` clock (Section 3.2.1) signified that the previous selection was a mistake. Two `Undo`-clock selections were never made in a row. Of $1,714$ selections, 3 were `Undo`—an error rate of about 0.002.

Click Distribution In the above description of Nomon selection decisions, we implicitly assumed that we know the distribution of the user's clicks around noon. In general, we would like a user to be able to start a Nomon application right away without spending a long time generating data for such a distribution, a time-consuming and boring task. To that end, we start with an initial belief $\hat{g}_0$ for the distribution density and learn the user's clicking behavior while he or she is using the interface. At any point, the learning could be stopped, and the user could proceed with the fixed distribution generated in this manner. Note that $\hat{g}_0$ is not used as a prior in the strict Bayesian sense but rather

serves as an initial guess to be utilized by the subsequent learning heuristic.

Our $\hat{g}_0$ assignment is broad and slightly offset from zero: $\mathcal{N}(0.05T, (0.14T)^2)$, where T is the clock period. The setting depends on clock period T so that it does not approach a uniform distribution for small T. The distribution choice, with initial setting $T = 2.0$, is based on a study of click distributions for two volunteers (Section 4). The choices of initial T and $\hat{g}_0$ are examined in more detail in Section 4.3. We update the $\hat{g}_0$ distribution with a (modified) Parzen window estimator—with width given below—and a damping factor λ that allows learning to continue over time. After any selection is made, we update the distribution estimate with the data from some number n_{offset} of selections back in time (e.g. $n_{\text{offset}} = 2$). This offset allows the user to choose `Undo` after a selection, in which case we do not use the clicks toward this selection for learning. Once a selection occurred n_{offset} rounds in the past, we assume that it was correctly chosen. With the clock choice c known for the s^{th} selection, we are able to calculate click times around noon $t_{s,r}$ for each click that was made toward this selection. We treat these as data from the distribution g we are estimating. To calculate our estimate $\hat{g}_s$ for g after the s^{th} selection, we make use of the unnormalized distributions $\tilde{g}_s$.

$$\tilde{g}_s(t) = \lambda \tilde{g}_{s-1}(t) + \sum_{r=1}^{R_s} \mathcal{N}\left(t; t_{s,r}, \hat{\sigma}^2_{\text{NS,s}}\right), \tag{1}$$

where

$$\tilde{g}_0(t) = n_\lambda \hat{g}_0(t). \tag{2}$$

The update equation (1) specifies that, after each selection, $\tilde{g}$ is damped by the factor λ. The next term in Eq. 1 is a sum over clicks r leading to the s^{th} selection. Within the summation is a normal density centered at the click time $t_{s,r}$, as in Parzen window estimation. The width for this Parzen-window term is given by $\hat{\sigma}_{\text{NS},s}$, which is derived from the normal scale rule estimate for the Parzen window (Wand and Jones, 1995). That is,

$$\hat{\sigma}_{\text{NS},s} = 1.06 n_\lambda^{-0.2} \hat{\sigma}_s,$$

where $\hat{\sigma}_s^2$ is the standard unbiased variance estimator obtained from the last n_λ clicks before the s^{th} round (Silverman, 1986). This rule is known to oversmooth distributions that are not approximately normal. However, user trials suggest the click distribution is usually unimodal and fairly bell-shaped (Section 4). Moreover, oversmoothing at worst results in a slightly more conservative number of clicks until a decision is made. For a longer discussion of the normal scale rule, see Appendix B.

The factor $n_\lambda = (1 - \lambda)^{-1}$ in Eq. 2 is an effective number of samples derived from the damping factor. Using this factor and the unnormalized update in Eq. 1, we ensure that

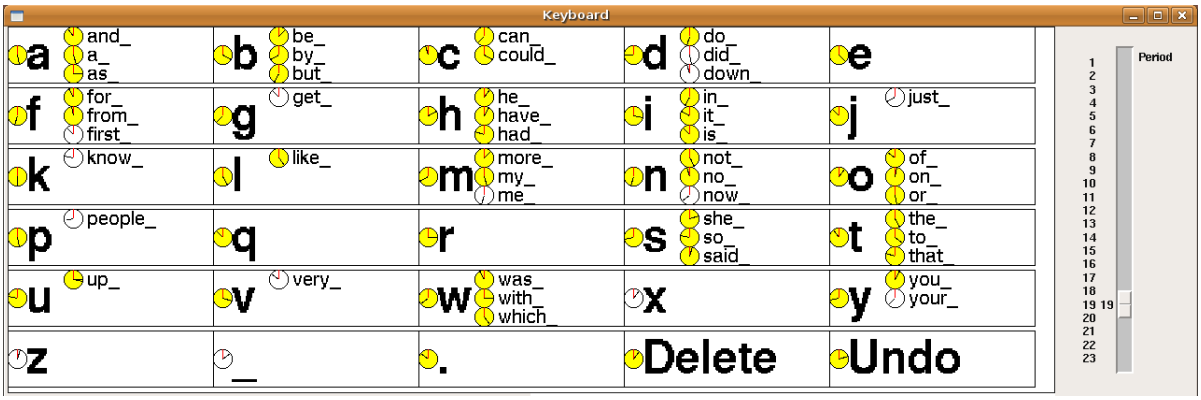

Figure 8: A keyboard application employing the Nomon selection scheme. A clock appears immediately to the left of each option, where options include characters, words, a character-deletion function, and an undo function. The slider at the right controls how long the moving hands of the clocks take to complete a full rotation.

the initial estimate $\hat{g}_0$ dominates $\hat{g}_s$ even after the first few selections. Without the n_λ factor, the Parzen window term for the first click, $\mathcal{N}\left(t; t_{1,1}, \hat{\sigma}^2_{\mathrm{NS},1}\right)$, would have essentially equal weight with the initial estimate.

This method allows us to save the distribution and update it quickly and easily during operation of the application. As a result, users can start the Nomon application immediately, without a waiting or calibration period, but they can also enjoy an experience tailored to their abilities. Indeed, we see that a user need not be perfect about hitting noon (or any offset) exactly. Their personal offset, reflecting reaction time, is learned by this method rather than hard-coded and, as long as it is not too close to 6 o'clock, will make no difference to program operation. The precision around this personal offset determines the number of clicks necessary to make a selection.

3.2 Keyboard Application

The preceding discussion applies to all Nomon applications. We now consider a particular application, a keyboard employing Nomon, in further depth.

3.2.1 User's View

Figure 8 shows the visual display of the Nomon Keyboard. Each small rectangle in the upper visual area represents a *key* on the keyboard. The key labels appear in bold and larger font. They range from `a`, `b`, `c`, to four non-alphabetic special keys: `_`, `.`, `Delete`, and `Undo`. Each letter of the alphabet shares its key with up to three word completions. All of these options are explained in more detail below.

The display features the distinctive Nomon clocks. Each clock appears directly to the

left of its associated selection. As explained in Section 3.1.1, any option may be chosen by clicking whenever the moving hand of the corresponding clock is near noon.

Special Characters As the user makes selections on the keyboard, text will appear in a textbox outside of the keyboard. The four special keys affect this text much like their counterparts on a normal keyboard. The underscore key '_' is just a representation of the space character. And '.' is just a period. `Delete` is a single-character backspace. If nothing appears in the text box, `Delete` has no effect. `Undo` undoes the previous selection. If the previous selection was a character or word completion, all characters added during the selection will be removed. If the previous selection was a `Delete`, the lost character will be returned. `Undo` has no effect if nothing else has yet been selected. In all non-trivial cases, the previous selection appears in green text just below the `Undo` button, and any text that would be undone if `Undo` were selected next is colored green in the main text box. All other text in the text box is colored black. If the last action was `Delete`, the word "Delete" appears in green text just below the `Undo` button, and there is no green text in the text box.

Word Completions At all times while writing with Nomon, a context is implicitly defined to be the string of alphabetic characters to the left of the text box cursor. The words surrounding each alphabet key on the keyboard are the most probable words that begin with the concatenation of the context and that letter. For instance, if the user has written "alphabet_so", then the context is "so", and the words within the `u` key will begin with "sou"—for example, "sound", "soup", and "source". If the user has written nothing, the words within each key will begin with that key's character. Only words that are likely (i.e. have a probability greater than some threshold) given the context will appear, so often a key may contain no word completions.

The `Delete` and `Undo` keys change the context. If the text box contents are "alphabet_soup_" before `Delete` is selected, the context will change from the empty string to "soup" after `Delete` is selected. After `Undo` is selected, the context reverts to its state before the last selection was made.

Controls In this particular instantiation of the Nomon Keyboard, the screen flash upon selection may be turned on or off. Also, dragging the righthand slider with the mouse adjusts the clock rotation speed. Alternatively, clicking the up arrow button moves the slider up by one mark (to a slightly faster speed) and clicking the down arrow button moves the slider down by one mark (to a slightly slower speed).

Both of these controls could be adjusted with Nomon clocks; this more integrated

interface was not pursued here in order to simplify the experiment described in Section 5.

3.2.2 How It Works

While the description above summarizes the user's experience of the Nomon Keyboard, it remains to fill in some details: which word completions appear on the screen and how to choose the prior over the clocks.

Our solution in both cases is informed by a training corpus of words. In this particular instantiation, we used the British National Corpus word-frequency list (Kilgarriff, 1998, 1997). Specifically, the data is composed of an alphabetic list of words w and their frequencies f_w. The count f_w is the number of occurrences of w in a sample set of written and spoken British English. We discarded single-character words besides "I" and "a". Of the remaining words, we kept only those appearing with some small minimum frequency. As the unused words are so rare, this step principally affects just the pre-processing time of the program.

Word Completions The word completions that appear depend first on the context, which is defined as the string of alphabetic characters from the end of what has been written so far to the last space, period, or beginning of the text box. The context may be empty. Necessarily, then, the context will be some list of letters $l_1 \cdots l_N$ of length $N \geq 0$. If any word completions appear next to key l' on the keyboard given this context, they must be prefixed by the string $l_1 \cdots l_N l'$. Since there at most three places for words next to l', the greatest number of word completions that appear is the minimum of three and the number of words in the corpus beginning with $l_1 \cdots l_N l'$.

We make one further restriction. Note that we may define a frequency for any context $f(l_1 \cdots l_N) = \sum_w f_w \mathbb{1}\{l_1 \cdots l_N \text{ prefixes } w\}$. That is, $f(l_1 \cdots l_N)$ is the number of times that the context appears at the beginning of a word in the corpus. Finally, then, we include only those words w satisfying the previous restrictions that also satisfy $f_w > \alpha_{\text{word}} f(l_1 \cdots l_N)$ for some small α_{word} chosen in advance. Since $f(l_1 \cdots l_N)$ is the total frequency of words prefixed by the current context, this criterion roughly states that the conditional probability of a word given the context, estimated as $f_w / f(l_1 \cdots l_N)$, must be greater than some lower bound α_{word}. It was judged that $\alpha_{\text{word}} = 0.001$ yields a reasonable balance between displaying common words and not cluttering the screen. The words that are displayed are ordered by probability.

Clock Priors To assign priors $p(c)$ to the clocks, we first assign fixed probabilities to the special characters. The priors on the letters and words are then chosen using Laplace smoothing out of the remaining probability mass p_{alpha}. Specifically, suppose the context

after the previous selection is $l_1 \cdots l_N$. Let $\mathcal{W}_{\text{on}}$ be the set of word completions appearing on screen. Assign $C_{\text{on}} = 26 + |\mathcal{W}_{\text{on}}|$, the total number of letters and word completions on screen. Then if $c(l')$ is the clock corresponding to letter l',

$$p(c(l')) = p_{\text{alpha}} \times \frac{f(l_1 \cdots l_N l') + 1}{f(l_1 \cdots l_N) + f(\mathcal{W}_{\text{on}}) + C_{\text{on}}} \tag{3}$$

If w is a word prefixed by the context $l_1 \cdots l_N$, and if $c(w)$ is the clock corresponding to w,

$$p(c(w)) = p_{\text{alpha}} \times \frac{f(w) + 1}{f(l_1 \cdots l_N) + f(\mathcal{W}_{\text{on}}) + C_{\text{on}}}. \tag{4}$$

In both equations above,

$$f(\mathcal{W}_{\text{on}}) = \sum_{w' \in \mathcal{W}_{\text{on}}} f(w').$$

The fact that the frequencies of word completions appearing on screen are not subtracted from the frequency of the letter in Eq. 3 reflects the belief that novice users will sometimes choose a letter instead of a word completion even when their desired word appears on the screen.

The non-uniform prior described in part by Eqs. 3 and 4 is used after every selection except for an `Undo`. After `Undo` is selected, the prior over all clocks is uniform. Novice users in a pilot study of the Nomon Keyboard preferred the uniform prior to a more informative prior after an `Undo` selection.

A different prior than the one described above might be more appropriate for an expert Nomon user. After a non-`Undo` selection, the prior might reflect the belief that the user will select a letter only when the desired word is not available:

$$p_{\text{expert}}(c(l')) = p_{\text{alpha}} \times \frac{f(l_1 \cdots l_N l') + 1 - \sum_{w \in \mathcal{W}_{\text{on}}} f(w) \mathbb{1}\{l_1 \cdots l_N l' \text{ prefixes } w\}}{f(l_1 \cdots l_N) + C_{\text{on}}}$$

$$p_{\text{expert}}(c(w)) = p_{\text{alpha}} \times \frac{f(w) + 1}{f(l_1 \cdots l_N) + C_{\text{on}}}.$$

Also, an informative prior after an `Undo` selection could take into account information gained during the selection preceding `Undo`. In all that follows, however, only the novice prior of Eqs. 3 and 4 is used.

4 Click Distribution Study

In this section, we seek to understand how the click distribution of users around the noon marker depends on the clock period and the initial angular offset of the desired

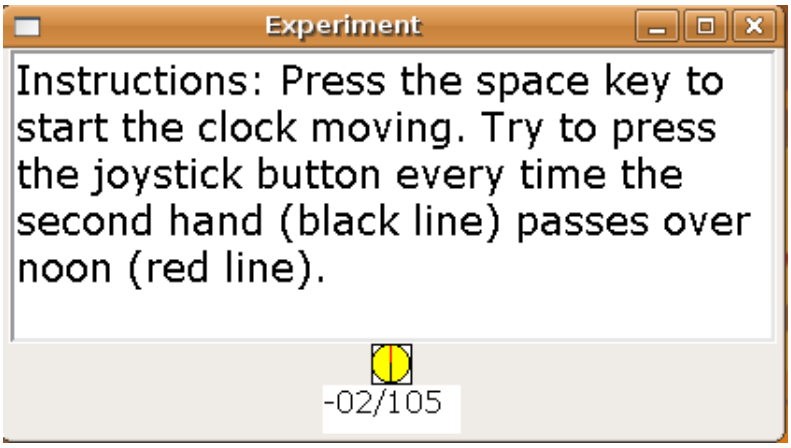

Figure 9: Instructions and clock for the switch distribution study. Below the clock is a counter, which took negative values for the two practice switches before counting the principal 105 switches in the block.

clock. The results of this study help us to choose reasonable values for the clock period, the angular offset of the most probable clock, and the click-distribution initial guess $\hat{g}_0$ (Section 3.1.2).

4.1 Method

4.1.1 Participants

We recruited two volunteers, screened for motor and cognitive difficulties; in particular, neither participant had dyslexia or RSI. Both participants were right-handed. One volunteer had never used Nomon or any single-switch interface. The other volunteer was an experienced (> 10 hours experience) Nomon user (the author). Volunteers agreed to participate in the study without compensation.

4.1.2 Apparatus and Software

All sessions took place on a desktop running Ubuntu 8.04 with Intel Pentium 4 CPU 2.80GHz. The desktop was connected to a 19 inch color screen. The physical screen size was 375×301 mm, with resolution 1280×1024 pixels. The single-switch hardware device was the trigger button—reached with the first finger of the right hand—of a Logic3 Tornado USB joystick. None of the other joystick inputs was used.

The experimental window (Figure 9) was docked in the center of the screen. It measured 123×69 mm (420×237 pixels). The top half gave a quick summary of the instructions. In the bottom half, there was a Nomon clock of radius 10 pixels above a counter showing how many clicks of the total had been achieved so far.

4.1.3 Procedure

Each volunteer completed the following experimental procedure in a single session lasting less than two hours, inclusive of break times. The session was composed of 42 blocks for the novice, and 35 blocks for the experienced user. Each block consisted of 107 button presses. Participants could rest for as long as desired between blocks; both participants spent approximately 30 minutes of the session in breaks. A participant began a block by pressing the space bar, at which point the clock hand would begin rotating. The block finished when the participant had pressed the joystick button 107 times (2 practice rounds followed by 105 rounds of data collection). During the block, the participant was asked to press the joystick button when the moving hand was above noon. After each press, the moving hand would reset to a random offset from noon, as described below.

We are interested in generating histograms of click times relative to noon for different combinations of angular offset θ_0 and period of clock rotation T. To that end, we consider values of θ_0 arranged evenly around the circle (and exclude noon): $2\pi \cdot \frac{i}{7}$ for $i \in \{1, \ldots, 7\}$. And we consider values of T of the form $2.0 \cdot 0.9^{3j}$. For the novice, $j \in \{-1, \ldots, 4\}$. For the experienced user, $j \in \{0, \ldots, 4\}$.

At each value of T, there are seven experimental blocks (hence, 35 total for the experienced user and 42 total for the novice user). We could set the initial offset of every round (i.e. every request for a click) within the i^{th} block (of the seven total blocks) to be the i^{th} θ_0 value above. This setup would give us 105 data points to form a histogram of click times relative to noon for each (T, θ_0) pair.

However, the design just proposed is problematic since a user could simply press in rhythm instead of reacting to the angular offset at each round. In usual Nomon operation, the user cannot predict the next clock offset after a click. Instead, then, during each block we form a list with five repeats of each value of θ_0 (for a total of 105 values of θ_0). We randomize this list to determine the sequence of θ_0 for the 105 rounds within a block. At the end, when data from all seven blocks at each T value has been compiled, we still have 105 data points to form a histogram of click times relative to noon for each (T, θ_0) pair, but the user is not able to predict the angular offset of any round.

4.2 Results

The histograms resulting from the set of 105 points for each (T, θ_0) combination (for a total of 35×105 experienced points and 42×105 novice points) are shown in Figure 10. The experienced user (upper panel) has more precise click distributions whereas the click distributions of the novice (lower panel) are wider. Only clicks within 0.25 seconds of noon are shown. Clicks that occurred during the first full clock revolution are shown in

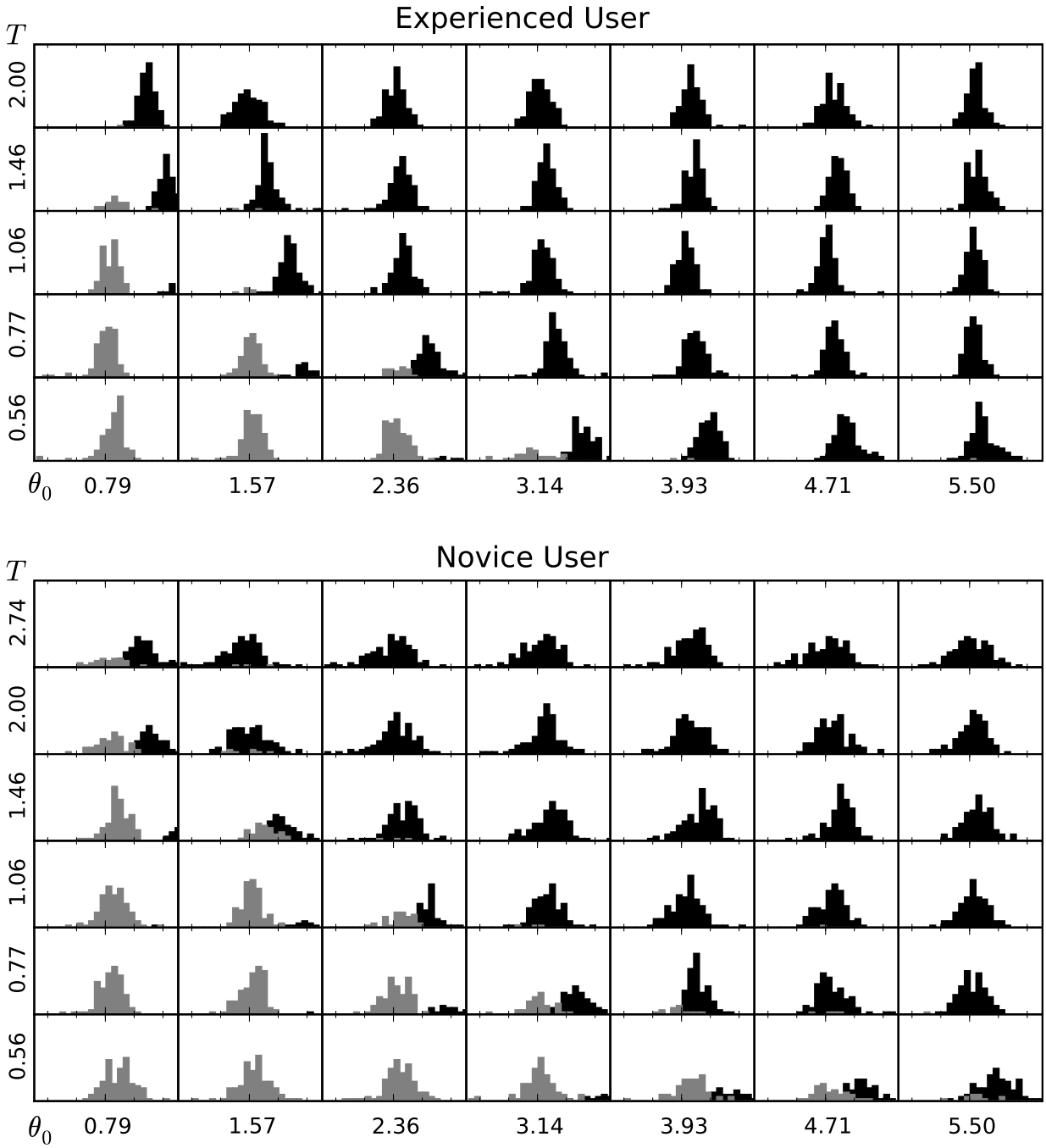

Figure 10: Histograms of click frequency around noon for the two participants in the click distribution study. The *upper* panel shows the results for the experienced user, and the *lower* panel shows the results for the novice. In either panel, each row corresponds to a different clock rotation period T in seconds, and each column corresponds to a different clock rotation starting angle θ_0 in radians, measured counterclockwise from noon. The horizontal axis of any individual plot ranges from -0.25 sec to 0.25 sec. Clicks that occurred within the first full clock revolution are in black; clicks in the second full revolution are superposed in gray.

black, and clicks that occurred during the second revolution are superposed in gray.

The relative precision of the experienced user is evident in the standard deviations of these distributions. After calculating an empirical standard deviation for each distribution in the figure, we can take the median across these standard deviation values. The median (across 35 values) for the experienced user was 0.04 seconds, with range 0.03 to 0.09 seconds. The median standard deviation (across 42 values) for the novice user was 0.06 seconds, with range 0.04 to 0.14 seconds. In both cases, distribution means were actually quite wide-ranging. The median and range of the experienced-user means were 0.03 and $[-0.01, 0.17]$ seconds respectively. The median and range of the novice-user means were 0.02 and $[-0.03, 0.09]$ seconds, respectively.

There is a clear divide between the region where the participant was able to plan sufficiently to click during the first revolution and the region where the participant felt compelled to wait until the second revolution. On the boundary, an unusually late-offset black distribution appears with a gray distribution that corresponds more closely to the norm; in these cases, the participant sometimes chose a click time significantly later than noon and sometimes waited until the next revolution. The data are consistent with a reaction time of approximately 0.2 seconds for the experienced user (yielding predicted boundary periods of 1.6, 0.8, 0.5, and 0.4 for the first four smallest θ_0 in increasing order) and a reaction time of about 0.4 for the novice user (yielding predicted boundary periods of 3.2, 1.6, 1.1, 0.8, 0.6, 0.5 for the first six θ_0 in order).

4.3 Discussion

Parameter Settings Our first goal with this data is to choose reasonable values for the fixed parameter settings of the Nomon Keyboard application: namely, the clock period, the angular offset of the most probable clock, and the click-distribution initial guess $\hat{g}_0$ (Section 3.1.2).

Of the periods T tested, the novice seems to have performed at a similar level at periods of both 2.00 seconds and 2.74 seconds. In order to avoid the frustration of overly slow rotation speeds, we choose 2.00 seconds as the starting clock rotation period. This setting can be manually adjusted while writing. Novice users in a pilot study of the Nomon Keyboard confirmed that this was a comfortable initial setting.

To choose an initial offset θ_0, we note that the novice user found the final two periods (0.77 and 0.56 seconds) to be uncomfortably fast. Across the remaining periods tested, π is the smallest offset without a significant second-revolution (gray) component in the lower (novice) panel of Figure 10. While we therefore choose $\theta_0 = \pi$, this choice is somewhat unsatisfactory because θ_0, unlike T, cannot be adjusted. The click-time distributions in Figure 10 demonstrate that a better choice of θ_0 would vary with T according to the

user's reaction time. However, this reaction time does not account for the time it takes the user to find a desired clock or the time it takes the user to decide on her next action in practice. Therefore, we keep the simpler fixed-θ_0 choice in what follows.

For the click-distribution initial guess $\hat{g}_0$, we choose a normal form after noting the bell-shape of the Figure 10 distributions. However, we aim for a $\hat{g}_0$ much wider than the click distributions so as to encompass a range of clicking behavior. In particular, the range of experienced user means (see above) was 0.18 seconds, and the maximum experienced standard deviation was 0.09 seconds. These sum to 0.27 seconds. Similarly, the range of novice user means was 0.12 seconds, and the maximum novice standard deviation was 0.14 seconds. These sum to 0.26 seconds. To choose a broad $\hat{g}_0$ at $T = 2.00$ seconds, we take the standard deviation σ_{init} of $\hat{g}_0$ to be 0.28 seconds. Taking the novice user as a rough guide for the relation of mean to standard deviation, we choose $\mu_{\text{init}} = 0.10$ seconds. Finally, we allow $\hat{g}_0$ to adjust proportionally with the period so that experienced users can start the application at faster settings. Thus we arrive at the $\mathcal{N}(0.05T, (0.14T)^2)$ distribution quoted in Section 3.1.2.

Theoretical Performance Estimates In Section 2.2.2, we found the optimal information rate of a single switch as a function of waiting time D and noise standard deviation γ. It may be that there is no realistic interface through which an individual with waiting time D and click noise γ can actually achieve this rate. The idealization does serve, though, as an upper bound on the information rate of any single-switch interface. And any discrepancy may point to ways to improve existing interfaces.

In this study, we have derived reaction times from the boundary between the two types of click distributions as well as estimates of noise distribution standard deviation. With the model in Section 2.2.2, then, we can estimate idealized information-rate upper bound for these two users. For this purpose, take waiting time $D_{\text{experienced}} = 0.2$ seconds and noise standard deviation $\gamma_{\text{experienced}} = 0.04$ seconds for the experienced user. For the novice, take $D_{\text{novice}} = 0.4$ and $\gamma_{\text{novice}} = 0.06$. With these parameters, the model predicts that the experienced user can generate information at a rate of at most 4.6 bits/second, and the novice user can generate information at a rate of at most 2.7 bits/second.

To interpret these rates as words per minute, we calculate an approximate entropy rate (in bits/character) for our user. In particular, we assume that the user writes only with the words from the subset of the British National Corpus (Kilgarriff, 1998) employed by our language model and that the user writes a space after every such word. If the user also writes a period for 5% of characters, then we find that the user's entropy rate is 1.18 bits/character. This information rate yields maximum writing speeds of 47 words per minute for the experienced user and 27 words per minute for the novice user. While

such high speeds sound promising, unfortunately they have not been attained with our interface (Section 5.2.1). We explore this discrepancy in Section 5.3.

5 Comparison Study

While the Nomon selection scheme has a variety of useful applications, the Nomon Keyboard is the application that is most directly comparable to existing single-switch technologies. As communication is such a basic requirement, much time and effort has been devoted to the creation of efficient writing devices with a single-switch input (e.g. MacKay *et al.*, 2004; Sun Java Desktop System Documentation Team, 2004; Words+, Inc., 2004). In the following study, we compare Nomon to a particularly popular commercial system, The Grid 2 (Sensory Software International Ltd., 2008). Our method and performance measures follow Kristensson and Denby (2009) and MacKay and Ball (2006).

5.1 Method

5.1.1 Participants

We recruited sixteen participants from the university community across a wide range of academic disciplines. Their ages ranged from 22 to 39 (mean = 26, sd = 4). Eight were women, and eight were men. Participants were screened for motor or cognitive difficulties; in particular, no participant had dyslexia or RSI. None of the participants had used a scanning or Nomon interface before. No participant had regularly used any single-switch interface before. Fourteen of the participants had used word completion (e.g. on cell phones). Participants took part in two sessions and were paid £10 for each session. A participant could earn up to £10 in performance bonuses, described below.

In addition to the sixteen novice participants, an experienced user of Nomon and The Grid 2 ($>$ 10 hours writing with each interface) was run through the same experimental procedure for comparison. The experienced user (the author) was the same as in the previous study (Section 4); this participant was not paid.

5.1.2 Apparatus and Software

All sessions were run on a Dell Latitude XT Tablet PC with a partitioned hard drive. The 12.1 inch color screen had a physical screen size of 261×163 mm. The single-switch hardware device in all cases was the trigger button of a Logic3 Tornado USB joystick. Participants operated the trigger button with the first finger of their left or right hand. None of the other joystick inputs was used. For both writing interfaces, automated spoken feedback was provided as the user wrote.

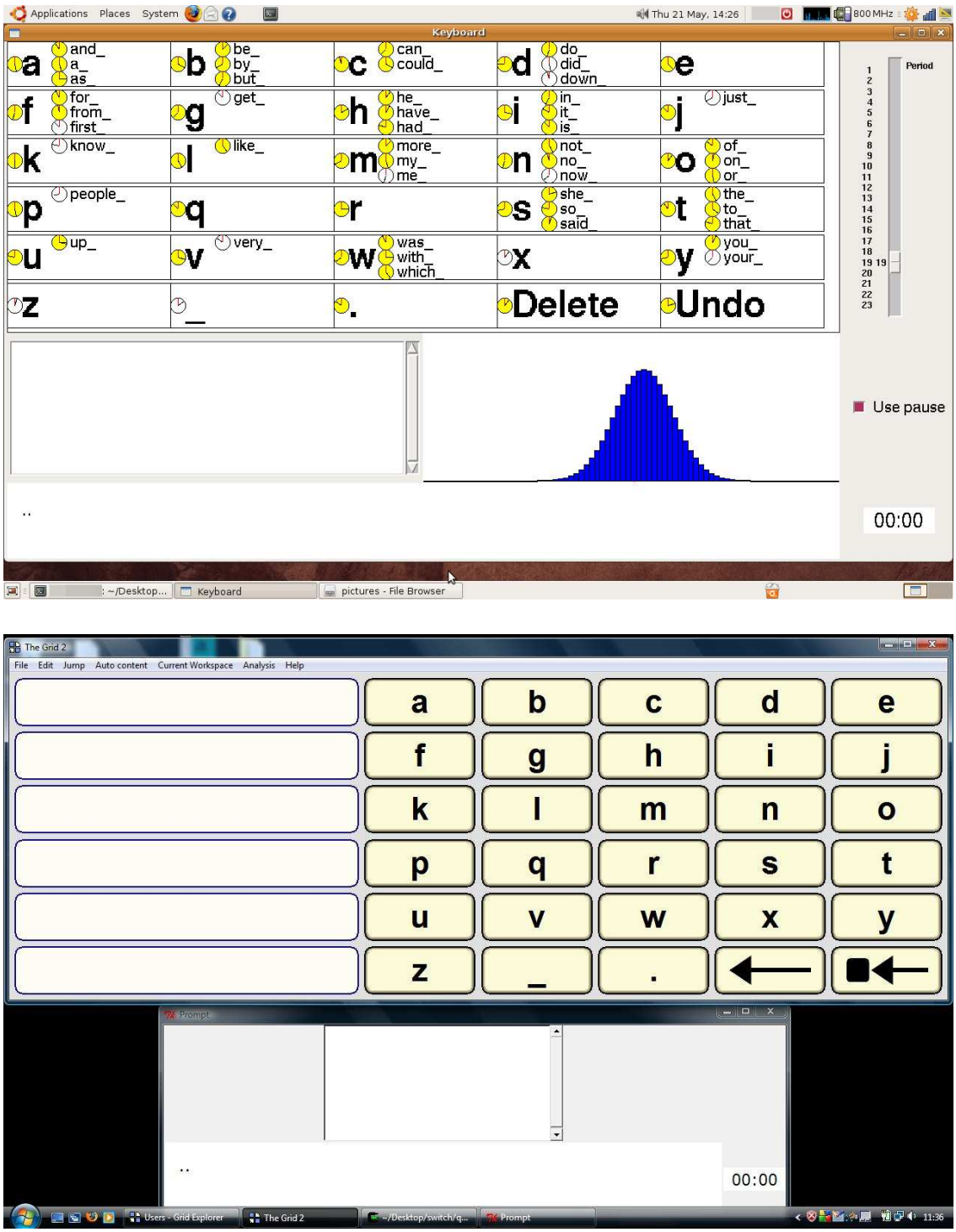

Figure 11: Screen appearance during the comparison study. *Upper*: Nomon Keyboard. *Lower*: The Grid 2.

Nomon Keyboard We ran the Nomon Keyboard (upper panel of Figure 11) on an Ubuntu 8.10 operating system running the Linux kernel. The screen resolution was 1280 $\times$ 800 pixels, and the physical size of the keyboard display was 224 $\times$ 85 mm (1125 $\times$ 416 pixels). The interface was docked in the upper part of the screen. A text box and phrase box were located below the keyboard in the same window. The keys of the keyboard were arranged in six rows and five columns. Each key contained a principal character, with letters in alphabetical order (across and then down) first, followed by four special characters: an underscore (representing space), a period, a character deletion function, and an undo function. Each letter key also contained up to three word completions.

The clock rotation period T could be set to $2.0 \cdot 0.9^i$ seconds for $i \in \{-4, -3, \ldots, 18\}$. Higher i corresponded to faster rotation. The initial setting of the period for novices was $T = 2.0$ ($i = 0$). The experienced user initially chose $T = 1.06$ ($i = 6$).

The Grid 2 We ran The Grid 2 (lower panel of Figure 11) on a Windows Vista Service Pack 1 operating system. The screen resolution was again 1280 $\times$ 800 pixels. The physical size of The Grid 2 display was 261 $\times$ 102 mm (1280 $\times$ 500 pixels). The interface was docked in the upper part of the screen, and the text box and phrase box were docked immediately below. Six word-completion boxes appeared on the left side of the main interface. The remaining space was divided into six rows and five columns of keys. Each key contained a single character. First were letters arranged in alphabetical order (across and then down), followed by an underscore, a period, a character deletion function, and a word deletion function. The studies in Section 2.1.1 suggest that an alphabetic layout does not lead to significantly worse performance than a frequency-ordered layout. We chose the alphabetic layout here on the assumption that it would be easier for the novices in our study to learn. Section 2.1.1 further suggests that word prediction does not significantly decrease writing rate though it may provide significant improvements in click rate.

The grid scanning delay d had settings at $0.1(10 - i)$ for $i \in \{\ldots, -1, 0, 1, \ldots, 9\}$. Higher i corresponded to faster scanning. The initial setting of the delay for novices was $d = 1.0$ ($i = 0$). The experienced user initially chose $d = 0.5$ ($i = 5$).

5.1.3 Procedure

The experiment consisted of two identical sessions, one for each interface. The starting interface was balanced across participants, and sessions were spaced at least four hours apart.

Each session proceeded according to the same schedule. The first ten minutes were introductory. First, the supervisor either explained or reviewed the experimental proce-

dure according to the session number. Then the participant was shown how to use one of the interfaces. The demonstration included basic writing, word completion, and error correction.

The next hour was divided into four 14-minute blocks, separated by short breaks. During the blocks, participants were asked to write phrases drawn from a modified version of the phrase set provided by MacKenzie and Soukoreff (2003), with British spellings and words substituted for their American counterparts. For each participant, a different random ordering of the initial phrase set was generated. Phrases appeared one at a time in the phrase box at the bottom of the screen. Once a participant finished a phrase, writing the period character twice would cause a new target phrase to appear and the text box to empty. Participants were instructed that no changes relevant to a particular phrase could be made after the two periods were written.

During the first three (of four) writing blocks, each participant was allowed to adjust the rotation-period or scanning-delay parameter at the end of each written phrase. In particular, immediately after writing two periods and receiving the new target phrase, the participant could increment or decrement i (defined in Section 5.1.2) by one. The experienced user incremented to $i = 7$ ($T = 0.96$) after two blocks using the Nomon Keyboard. This user similarly incremented to $i = 6$ ($d = 0.4$) after two blocks using The Grid 2. No other changes were made by this user.

Participants were informed at the beginning of the study that they could receive a £5 bonus for achieving a writing speed among the top half of novice participants for each interface. They were further informed that, for the purposes of the bonus, writing speed would only be measured during the final writing block. They were told that they would not be allowed to change the rotation -period or scanning-delay parameter during this block and thus would have to calibrate it as they saw fit during the previous blocks. Information about their own writing speeds across full blocks and also phrase-by-phrase was made available to participants during the break after each block.

5.2 Results

In total, 34 hours of data from 16 novice participants and 1 experienced participant were collected. We compared objective measures of the participants' performance between the two interfaces using a variety of measures: text-entry rate, error rate, clicks per character, and proportion of time spent in deletion functions. We also examined subjective ratings of the two interfaces given by the participants. In the following, we perform seven significance tests with a family-level significance of 0.05. Observing the Bonferroni correction, we perform each individual test at a significance level of $\alpha = 0.007$. Except when indicated otherwise, a one-way analysis of variance (ANOVA) test for repeated measures

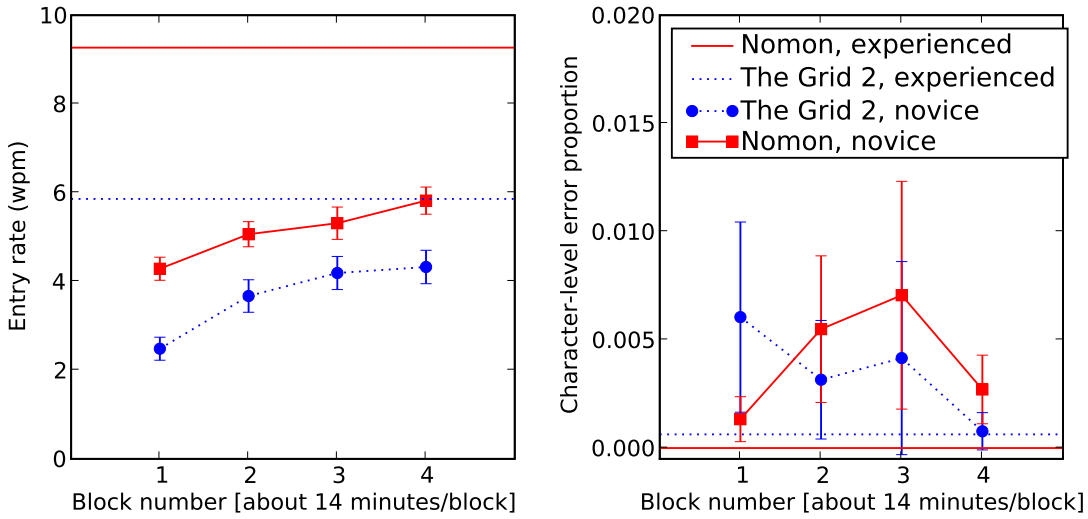

Figure 12: *Left*: Entry rate in words per minute across interface blocks. In both panels, error bars represent 95% confidence intervals for the novice user curves, and the average experienced user (author) performance is illustrated by horizontal lines for comparison. *Right*: Error rate measured as Levenshtein distance divided by target-phrase length (in characters) across interface blocks.

was performed wherever F values are quoted.

5.2.1 Text-entry Rate

As explained previously (Section 2.1), we calculate text-entry rate in words per minute, where a word is defined as five consecutive characters in the output text. At the beginning of each fourteen-minute block, the participants were asked to write two periods using the interface for that session. This action signalled that they were ready to begin and generated the first target phrase. Timing started once the two periods were written and stopped when the final two periods following the last phrase were written. The periods after each phrase were counted as characters in what follows, and the time spent writing them was counted as well.

The left panel of Figure 12 shows the novice participants' mean entry rates across the four blocks for each interface. Also shown, for comparison, is the performance of the experienced user. Participants wrote faster with Nomon during the first block ($F_{1,15} = 128.69$, $p = 9.2 \cdot 10^{-9}$). While the total session time was short for both interfaces, participants became faster with practice in both the Nomon session ($F_{3,45} = 58.50$, $p = 1.4 \cdot 10^{-15}$) and The Grid 2 session ($F_{3,45} = 122.17$, $p < 10^{-15}$). This learning is apparent in the left panel of Figure 12. In the final block we see that participants remained faster at writing with Nomon ($F_{1,15} = 134.59$, $p = 6.8 \cdot 10^{-9}$). In the fourth block, Nomon was 35% faster than the scanning interface; participants were writing at 4.3 words per minute

on average with The Grid 2 and 5.8 words per minute with the Nomon Keyboard. The speed of 4.3 words per minute is greater than the reported writing speeds using scanning from the majority of studies in Section 2.1.1.

The experienced participant was writing at speeds of 5.7 words per minute with The Grid 2 and 9.5 words per minute with the Nomon Keyboard in the fourth block. The speed of 5.7 words per minute is greater than the speeds reported by all but three of the scanning studies in Section 2.1.1; of these three studies, two used a different definition of writing speed, and one stated a theoretical upper limit on writing speed with scanning.

5.2.2 Error Rate

To find the error rate during a block, we begin by computing the Levenshtein distance d_i between the i^{th} target phrase in the block and the text written by the participant; d_i is also known as the edit distance. We define the error rate for the block to be $\sum_i d_i / \sum_i n_i$, where n_i is the number of characters in the i^{th} target phrase.

The right panel of Figure 12 illustrates participant error rates during the experiment. The experienced user attained a zero error rate except in the first scanning block. The zero error rate indicates that all text entered by the experienced user during these blocks was exactly the same as the target phrases. The non-zero error rate for scanning corresponds to a single error within the entire set of four blocks.

A glance at the output errors made by participants suggests that they were mostly due to poor recall of the target sentence. For instance, one participant pluralized "head" in "head_shoulders_knees_and_toes" (an edit distance of 1) and wrote "reading_week_is_almost_here" instead of "reading_week_is_just_about_here" (an edit distance of 8).

The average novice character-level error rate (over all blocks) for the Nomon Keyboard was 0.43%, and the average novice error rate for The Grid 2 was 0.34%. There was no significant difference in novice error rate between the two interfaces ($F_{1,15} = 0.71$, $p = 0.41$).

5.2.3 Click Rate

The click rate is the number of clicks per character. Other names for this measure include "keystrokes per character" (MacKenzie, 2002) and "gestures per character" (MacKay and Ball, 2006). The click rate is calculated as the number of button presses in a block divided by the number of characters in the output. The left panel of Figure 13 shows that novice participants, on average, spent around 1.5 clicks to generate a character in either interface. Specifically, the average novice rate (over all blocks) for the Nomon Keyboard was 1.58 clicks per character, and the average novice rate for The Grid 2 was 1.55 clicks

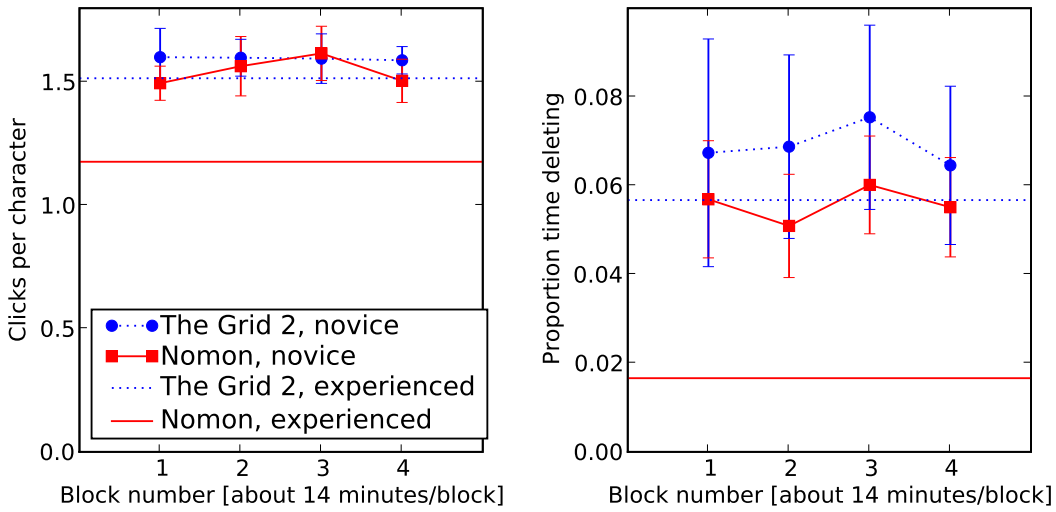

Figure 13: *Left*: Click rate in clicks per character across interface blocks. In both panels, error bars represent 95% intervals for the novice user curves, and the average experienced user (author) performance is illustrated by horizontal lines for comparison. *Right*: Proportion of total time spent correcting errors across interface blocks.

per character. There was no significant difference in novice click rate between the two interfaces ($F_{1,15} = 0.49$, $p = 0.49$).

However, the experienced user's data demonstrates that a much lower click rate can be achieved with the Nomon Keyboard. While the experienced user still required about 1.5 clicks per character in The Grid 2 (roughly in agreement with estimates from Section 2.1.1), the click rate approached 1 click per character for the Nomon Keyboard. For comparison, writing with a normal keyboard requires at least one key press for each character and thus at least 1 click per character (possibly more due to error correction). To compare to Morse code, we assume letter and space frequencies according to Lee (1999) and the Morse encoding of Words+, Inc. (2000); this encoding has the same click and hold sequences for alphabetic characters as ITU Radiocommunication Assembly (2004). In this case, a Morse code click rate estimate is 2.5 clicks per character without space and 2.7 clicks per character with space. These rates are two to three times as high as the Nomon click rates.

5.2.4 Deletion Time

While the number of clicks required to make a selection is variable in Nomon but constant in The Grid 2, there are two factors that contribute to click rate in both interfaces: word completions and mistake corrections. To investigate the contribution of mistake corrections, we can compare the proportion of time spent using the deletion and undo functions in the two interfaces. This measure has previously been used by Kristensson and Denby

Statement	Nomon	The Grid 2
	mean (sd)	mean (sd)
I liked writing using X.	**5.6** (1.4)	3.9 (1.5)
It was easy to select word completions (the, and, cat, ...).	**6.1** (0.7)	4.8 (1.3)
It was easy to correct errors.	**4.5** (1.8)	3.9 (1.7)
I had to look intensively at the text area (where the final text goes) to spot any errors.	**2.3** (1.6)	3.1 (1.8)

Table 3: Subjective ratings of the two interfaces by novice participants. Each response to the lefthand statements was on a scale from 1 (strongly disagree) to 7 (strongly agree). In the questionnaires, the interface name was substituted for 'X'. Mean responses are shown with standard deviations in parentheses. Boldface is used to highlight the means corresponding to a more positive user experience.

(2009) for comparing text-entry methods. The results from our study are illustrated in the right panel of Figure 13). Novices spent, on average across all blocks, 5.6% of their time deleting while using the Nomon Keyboard and 6.9% of their time deleting while using The Grid 2. The difference between interfaces for novice participants was not significant ($F_{1,15} = 2.09$, $p = 0.17$). In contrast, the experienced participant spent roughly the same proportion of time (5.7%) in The Grid 2 using these functions as novices do, but this participant spent much less time doing so in Nomon (1.7%).

Note that deletion and undo-function selection are not strictly the same as mistake correction. Specifically, the `Delete` function can be used in multiple ways in both programs. Not only is it useful for directly correcting mistakes, but it can also be used strategically to eliminate the space character (or further characters) at the end of a word completion. This complete-then-delete strategy may be helpful if a large enough portion of a word completion is a prefix of the desired word.

5.2.5 Subjective Ratings

The performance measures discussed thus far are all objective and do not take into account participants' opinions and preferences concerning the two interfaces. Particularly with a motor-impaired end user in mind, it is important that a method not cause discomfort and that it be easy to use.

We assessed participants' opinions with a questionnaire administered immediately after writing with an interface was completed. The questionnaires for each interface were identical (except for the name of the interface). Participants were asked to rate how much they agreed with a series of statements on a scale from 1 (strongly disagree) to 7 (strongly agree). These statements were largely the same as those in Kristensson and Denby (2009). A box labelled "Open Comments" occupied the remainder of the one-page questionnaire,

and participants were encouraged to write any thoughts about the interfaces there.

Participant mean responses to the statements appear in Table 3. Perhaps most significantly, participants liked using the Nomon Keyboard more than The Grid 2 scanning interface. More to the point, no participant in the study rated The Grid 2 more highly than the Nomon Keyboard on enjoyment; every participant liked using Nomon at least as much as The Grid 2.

Contributing factors for why the Nomon Keyboard was preferred became apparent in the remaining responses. Participants found it easier to select word completions and easier to correct errors with the Nomon Keyboard. These responses corroborate our objective findings in Section 5.2.4. Perhaps as a result of easy word completion and error correction, participants reported they did not need to check their progress in the output text box as closely with the Nomon Keyboard as with The Grid 2. This freedom may have allowed them to better concentrate on their writing task.

The subjective difference between the two interfaces in terms of perceived time spent looking at the text output area may be misestimated. This question was the only one where a lower numerical value corresponded to more positive user experience, so some participants who filled in a similar value across all questions may have been confused or misled by the format.

Open comments supported the conclusions drawn above. Participants emphasized the enjoyment aspect; they wrote that the Nomon Keyboard was "fun to use," "more fun to use" than the other interface, and "lots of fun;" conversely, The Grid 2 was "less fun to use." In terms of ease of use, they wrote that the Nomon Keyboard was "more user-friendly," "really useful," "easier to use" than the other interface, and "not as difficult;" conversely, The Grid 2 was "tiring, time consuming, not user-friendly."

Details provided suggest that while The Grid 2 was, for at least one participant, "more frustrating when an error was made," a repeated complaint was the difference in word completion arrangement. In The Grid 2, items must necessarily be arranged in a grid. It is natural (and standard in the pre-packaged grids with this software) to put all word completions in the same place so that they may be easily scanned. Moreover, interspersing word completions with the character keys is not feasible since such a large addition of selectable items would greatly reduce selection time in row-column scanning. In the Nomon Keyboard, however, interspersing word completions within the keys does not significantly reduce selection time; we saw in Section 3.1.2 that the number of clicks per selection scales logarithmically with the number of clocks. As a result, participants found Nomon word completions "easier to spot since they are placed in alphabetical order," liked having "more options [arranged] in a more structured way," and liked having "a lot more word completions to choose from." The Grid 2, in turn, was "a little more

difficult to use for word completion than the clocks interface because you have to quickly switch your focus on the screen to see which words are available then look back to the letters."

It is worth noting that the unusual nature of the Nomon clocks may be a bit disconcerting to participants at first glance. This impression is implicit in one participant's remark about the Nomon Keyboard, "Surprisingly, I found this more user-friendly." Similarly, another participant notes that, "The writing system looks intimidating when it first comes up on the screen but is actually very easy to use."

5.3 Discussion

The participant's remarks that the Nomon Keyboard seems "intimidating" at first suggest that there may be an initial psychological barrier to be overcome in introducing Nomon to new users. Even though this participant quickly discovered how simple Nomon was to operate, there was some initial surprise at the unusual clock idea. It is easier for users to understand how more traditional methods of communication for motor-impaired individuals, such as scanning, work. These methods perform no computation that cannot easily and concurrently be performed by the user.

However, even participants who have been exposed to the Nomon Keyboard and The Grid 2 for just one hour agree that it is worthwhile to overcome this barrier. Novice participants are 35% faster writing with the Nomon Keyboard than with The Grid 2 after three quarters of an hour of practice, and data from an experienced participant suggests users may require fewer clicks per character in the Nomon interface after further practice. Finally, despite any initial trepidation, novice participants quickly found using Nomon to be a fun experience, and more enjoyable than the alternative.

Theoretical Comparison We found that Nomon performs well in the sense that users reach higher writing speeds with Nomon than with a competitive writing system. However, Nomon no longer rates so favorably when compared with our previous estimates of optimal, idealized single-switch writing speed. In Section 4.3, we calculated idealized maximum writing speeds for experienced and novice single-switch users based on the model in Section 2.2.2. The novice rate of 5.8 words per minute from the experiment in this section is just 21% of the estimated optimal speed: 27 words per minute. The experienced rate of 9.5 words per minute is 20% of the estimated optimal speed: 47 words per minute.

While the idealized model optimizes over fully general single-switch communication, both Nomon and The Grid 2 have a periodic component. That is, in either interface, there are click times, separated by a constant duration, that convey exactly the same

information. In particular, clicking while a Nomon clock is at a particular angular offset conveys the same information no matter how many rotations have elapsed thus far. Such periodicity is important for usability as it allows the user to switch between tasks without losing their progress in Nomon. When this extra assumption is included, though, the maximum theoretical mutual information between the input and output variables will be lower than above. Therefore, the resulting theoretical bound on writing speed will be lower. Using the model in Section 2.2.3, we can quantify the expected information rate (and hence writing speed) drop. In both the novice and experienced user cases, the periodic information rate predictions are about 20% lower than the predictions above—specifically, 22 words per minute for the novice and 37 words per minute for the experienced user.

Nomon is further designed to handle selection of options at many different locations on the screen. Deciding which option to select, as well as finding the desired option, takes time. In this study, users needed to find the next character, and, if it was a letter, read the word completions. Searching time is not included in the theoretical model. It may be possible to design an interface for, say, writing that does not require searching on a screen; indeed, Dasher (Ward and MacKay, 2002) aspires to this goal. However, Nomon is designed to be useful in exactly those situations when screen selection *is* desired. Moreover, resting time and deciding time on the user's part cannot be eliminated.

Designing Nomon as a method for arbitrary point selection has another consequence; while Dasher can take advantage of the full entropy of any click by using stream coding, Nomon is limited to block coding. As a result, the final click in any sequence can, at most, reduce the entropy of the probability distribution over clocks to zero. See Figure 6 for an illustration of entropy changes as a function of clicks remaining to selection. If we suppose that the experienced user requires 2 clicks for each selection and that half of the entropy of the second click, on average, is wasted, the idealized optimal information rate for this user is further reduced by 25%.

Some remaining discrepancy may be due to imperfect parameter choices. In this study, participants were allowed to choose speed settings in both interfaces, but they might have chosen, for instance, an overly fast setting in an effort to win the monetary bonus. The following study (Section 6) suggests this was not the case for the experienced user. However, it was observed during trials that novices would sometimes choose rotation periods that forced them to wait for at least one full revolution before clicking. These users, in particular, might benefit from an automatic, adaptive clock rotation speed as well as an adaptive choice of initial offset for the highest-scoring clock. In this study, it was fixed.

On the other hand, the clock probability distributions and selection criterion were

tailored to novice use and common novice errors (Section 3.2.2). The screen flash and 400 millisecond pause after each selection was also intended as a novice teaching mechanism. During the pause, no clicks would register with the interface; this feature was designed to limit errors due to confusion or double-clicking. Though primarily intended for novice use, this pause was activated throughout all sessions in the studies described in this paper. It is possible that the experienced user could have written more quickly without such safeguards built into both the display and the mathematical model.

The five-fold discrepancy between true and idealized writing speed suggests that it may be worthwhile to examine the effects of each design aspect above (and even the clock representation) on information rate. These design choices may prove necessary for a realistic single-switch device, but we see there is potential at least for a much faster method.

6 Parameters Study

In the previous study, participants (including the experienced user) were allowed to adjust the parameter setting of each application as desired. For The Grid 2, the adjustable parameter was the scanning delay time, and for the Nomon Keyboard, the parameter was the clock rotation period. Personal parameter adjustment reflects the way a user would interact with these applications in personal use as neither interface has—at this moment—automatic, adaptive settings for the respective parameter.

However, there are trade-offs inherent in any setting. Indeed, while we have considered writing rate and clicks per character as performance measures thus far, there is somewhat of an inverse relation between these two. We might expect clicks per character to decrease (a desirable effect) as words per minute decreases (an undesirable effect) if only because we expect mistakes to decrease in this scenario. To better understand the nature of this trade-off, as well as the truly optimal levels of words per minute and clicks per character that may be achieved, we examine the experienced user's performance measures across a spectrum of parameter values.

6.1 Method

The single participant in this study was the experienced user (the author) from the comparison study. The apparatus and software remain the same.

As in the comparison study, each session was broken into 14-minute blocks. Rather than allowing the participant to adjust parameters after each phrase, though, the parameters were preset and constant across each block. In particular, for The Grid 2, the

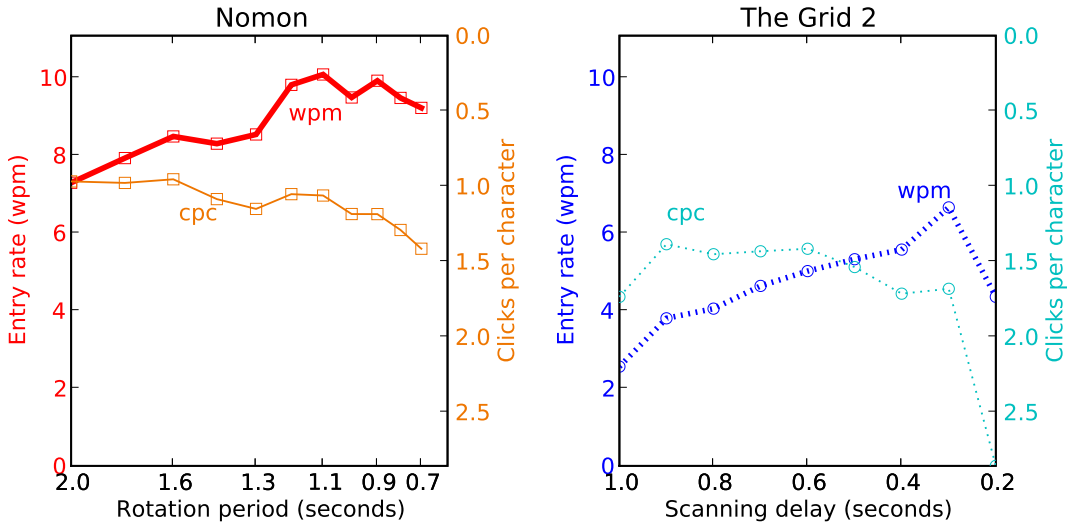

Figure 14: *Left*: Writing speed and clicks per character as a function of clock rotation period for the Nomon Keyboard. In both panels, the click axis is inverted so that "up" always represents better performance. *Right*: Writing speed and clicks per character as a function of scanning delay for The Grid 2.

scanning delay of the first block was set to $0.1(10 - i)$ with $i = 0$ (the initial setting for all novice participants in the comparison study). Each subsequent block had a scanning delay with i incremented by one from the previous block. Testing finished with $i = 8$, one setting beyond the smallest delay chosen by any participant in the final block of the comparison study.

Likewise, for the Nomon Keyboard, the clock rotation period was initially set to $2.0 \cdot 0.9^i$ with $i = 0$ (the initial setting for all novice participants in the comparison study). Each subsequent block had a clock rotation period with i incremented by one from the previous block. Testing finished with $i = 10$, one setting beyond the shortest rotation period chosen by any participant in the final comparison block.

6.2 Results

The following sections illustrate the trade-offs between writing speed and click rate as well as proportion of time spent in deletion.

6.2.1 Text-entry Rate

From the right panel in Figure 14, we see that while speeds between 5 and 6 words per minute are attainable by this participant with The Grid 2 at the comfortable scanning delays of 0.5 seconds and 0.4 seconds, a speed of 6.7 words per minute is attainable at a delay of 0.3 seconds. While this delay was uncomfortable for the participant in the

current trial, it might become less so with continued practice at this speed. Nonetheless, even this new, higher speed was lower than the lowest speed attained by the participant across Nomon Keyboard settings explored here (right panel of Figure 16). Indeed, even at the slow starting setting, the user was writing above 7 words per minute. All Nomon Keyboard speeds except for the lowest (at $T = 2.0$) are higher than all of the row-column scanning speeds cited in previous studies in Section 2.1.1.

This lowest Nomon Keyboard speed of 7 words per minute represents a drop-off of approximately one-third from the maximum speed with this interface. The Grid 2, however, was much more sensitive to parameter settings; the speed attained at the lowest setting was about two-thirds lower than the maximum speed. This difference between the interfaces is reasonable since, even at slow settings in Nomon, a clock is available for clicking during every rotation period. Scanning, by contrast, must cycle through every other option before returning to the desired one.

6.2.2 Click Rate

The expected trade-off between writing speed and clicks per character is evident for both interfaces and plotted in the left panel of Figure 16. The parameter settings at each point in that figure are made explicit in Figure 14. Though at high speeds, the Nomon Keyboard required as many as 1.5 clicks per character, not only did sufficiently low speeds bring this requirement down to 1.0 click per character, but there existed settings for which high speeds (above 10 words per minute) coincided with low click rates (about 1.0 click per character). In The Grid 2, on the other hand, click rate was mostly consistent around 1.5 clicks per character, with a spike at unmanageably fast parameter settings.

6.2.3 Deletion Time

Indeed, Figure 15 reveals one cause of the speed-click trade-off. In the right panel, we see that the time spent deleting became very high at the 0.2 second delay setting, resulting in the spike seen in the middle panel of Figure 14. The clicks in Nomon, however, were not due to time spent deleting, as evidenced by the left panel of Figure 15. Rather, the trade-off for this interface was a result of the nature of Nomon selections; at higher clock speeds, clicks must be less precise relative to the period, and therefore Nomon requires more clicks to be made per selection.

6.3 Discussion

While both Nomon and The Grid 2 exhibited a trade-off such that better click rates could be obtained at the cost of worse writing speeds, we see from Figure 16 that the Nomon

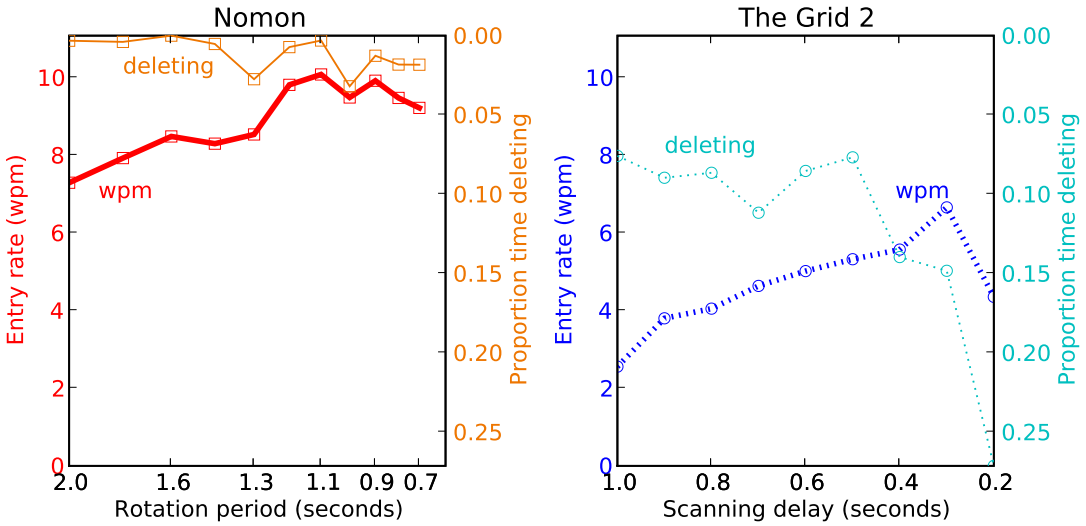

Figure 15: *Left*: Writing speed and proportion time spent deleting as a function of clock rotation period for the Nomon Keyboard. In both panels, the deletion axis is inverted so that "upper right" represents better performance. *Right*: Writing speed and proportion of time spent deleting as a function of scanning delay for The Grid 2.

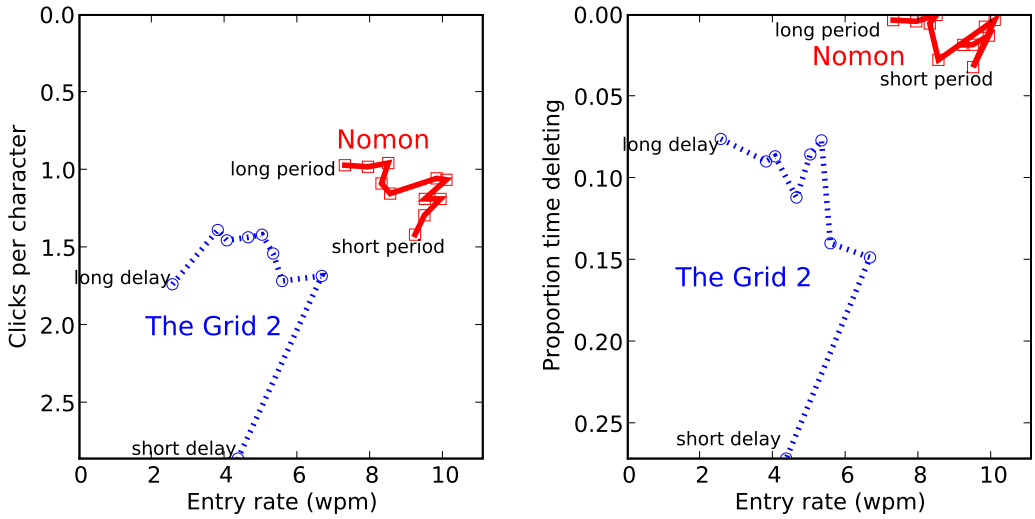

Figure 16: *Left*: Clicks per character as a function of writing speed across ordered parameter settings for the Nomon Keyboard and The Grid 2. In both panels, the vertical axis is inverted so that "up" always represents better performance. *Right*: Proportion of time spent deleting as a function of writing speed across ordered parameter settings for the Nomon Keyboard and The Grid 2.

Keyboard remained preferable in all measures across reasonable parameter settings.

7 Further Nomon Applications

The Nomon Keyboard writing application has been tested extensively by various individuals, not least those in the studies above (Section 4, 5, 6). However, this program represents just one possible application of the general Nomon selection mechanism. To illustrate the range of possibilities, we here consider other project proposals, in various stages of development.

Since writing features a limited set of discrete elements (characters), it is well-suited to single-switch selection. The following applications, on the other hand, are to varying degrees more difficult to formulate as single-switch problems. Indirect mechanisms must be introduced to allow GUI interaction or web browsing with scanning, but Nomon allows icons, links, and other GUI elements to be selected directly at their screen positions. Drawing is technically possible with a writing system if a user specifies pixel locations and drawing elements in code. Alternatively, single-switch drawing programs may allow pre-defined, stereotyped actions on a picture. Nomon, by contrast, lets the user directly select pixels and the location of common drawing elements such as lines, rectangles, and ovals.

7.1 Nomon Draw

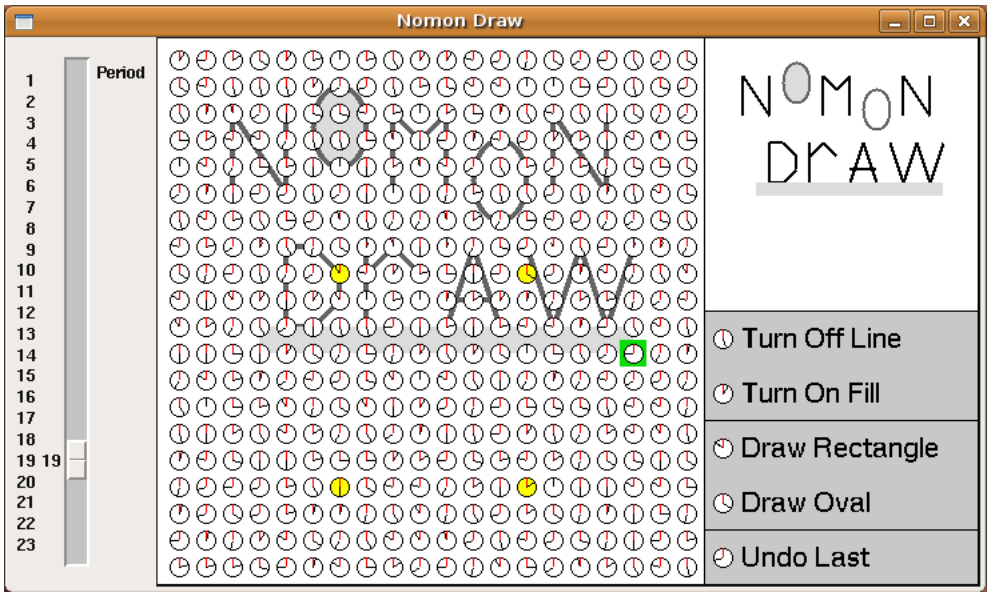

Figure 17: A Nomon-based drawing program in development. After two clicks, only four clocks are highlighted in yellow. The last selected pixel is surrounded in green.

A minimal version of a Nomon-based drawing program is depicted in Figure 17. Filling the screen with a grid of clocks allows a single-switch user to easily access any pixel in

the drawing area. For instance, for a 20 × 20 grid of clocks with some additional clock-accessible menu options, an experienced Nomon user was almost always able to access any clock with 3 clicks (Figure 6). This square grid of clocks appears at the center of Figure 17 with the usual rotation speed control to its left. The right side of the application features a clock-free copy of the drawing and a menu of common drawing options. Accessing pixels directly with clocks allows a variety of functionalities. Two pixels might define the two endpoints of a line, two corners of a rectangle, or two corners of the bounding rectangle for an oval. The default operation of the application is a line-drawing program. The user can move to a different pixel without drawing a line by toggling the "Turn Line On"/"Turn Line Off" menu option. The user can draw a rectangle or oval instead of a line by selecting one of these options before the second pixel. Whether the shape is filled with color can also be toggled by the "Turn Fill On"/"Turn Fill Off" option. Finally, "Undo Last" will remove the last item drawn (if an item was drawn after the last selection) and return the user to the starting pixel position before the last selection.

Some simple enhancements for future incarnations of this application include a menu for color options and another for line styles. Also useful would be clock-accessible options for saving pictures and for opening existing pictures for editing. Ideally, file names could be typed with the Nomon Keyboard.

7.2 Internet Browsing

Figure 18: Sketch of Nomon-based internet browsing.

The remaining two Nomon project ideas are not actively being pursued at the moment but rather serve as sketches to motivate future work. Figure 18 illustrates how links in

an internet browser might be accessed with Nomon clocks positioned alongside each option. There could also be a clock to access each text box—and open a writing program automatically—as well as clocks for scrolling at the top and bottom of a scrollbar.

7.3 Operating System

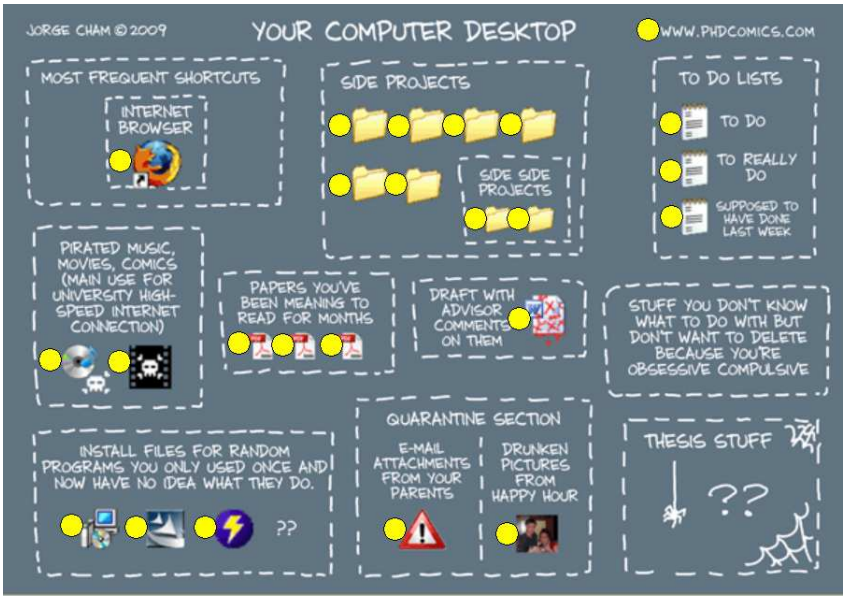

Figure 19: Sketch of Nomon-based operating system GUI interaction. Image modified from "Piled Higher and Deeper" by Jorge Cham (www.phdcomics.com).

Finally, we could adapt existing operating system GUIs with Nomon by placing clocks next to any relevant selection location, such as the desktop icons in Figure 19. Navigation bars, and window drop-down menus such as `File`, `Edit`, etc. could also be accessed in this way.

8 Conclusion

This paper focuses on a balanced, 16-subject comparison of novice use between the Nomon Keyboard and a popular commercial scanning system. We found that Novice users wrote 35% faster with the Nomon Keyboard and had more fun doing so. While writing can be accomplished in an intuitive way with existing single-switch systems, we have highlighted some applications that are less intuitive, if not impossible, with these systems—for instance, Nomon Draw. Pre-release versions of the Nomon Keyboard and Nomon Draw—as well as instructions for use and demonstration videos—are available free online at `http://www.inference.phy.cam.ac.uk/nomon/` under the GNU General

Public License. The Nomon Keyboard described in this paper corresponds to pre-release version 0.2.

Beyond these demonstrations of viability and potential, there remains much theoretical and practical work to be completed with Nomon. In this paper, we have developed a new theoretical model for the idealized information rate that may be achieved with a single switch. The gap between this model and the writing speed currectly attainable with single-switch methods suggests that there may be room for further significant improvements in these methods. In the case of Nomon, the initial form and evolution of the click distribution needs to be studied more carefully and justified more rigorously from an information-theoretic perspective. The same assessment holds for the positioning of the clocks' moving hands and the functionality of the `Undo` key. Even the clock representation might be changed to good effect. Practically, it remains to finish development of a Nomon drawing application and to integrate Nomon into a fully functional operating system and web browser. Most importantly, Nomon use by full-time single-switch users needs to be studied, and Nomon needs to be adapted to their patterns of usage.

A Information Rate Derivations

Discrete, Noiseless Model

This derivation is related to Exercise 6.18 on p. 125 of MacKay (2003).

Let B be the probability of the block with length D seconds; then the probability of the block with length g seconds must be $1 - B$. It follows that, after a block of length D (i.e. a click), the probability that n blocks of length g occur before the next D-block is $p(n) = (1 - B)^n B$. The entropy of p is

$$\begin{aligned} H(p) &= -\sum_{n=0}^{\infty}(1-B)^n B(n\log_2(1-B) + \log_2(B)) \\ &= -B\log_2(1-B)\sum_n[n(1-B)^n] - \log_2(B)\sum_n[(1-B)^n B] \\ &= -B\log_2(1-B)\frac{1-B}{B^2} - \log_2(B) \\ &= B^{-1}H_2(B) \end{aligned}$$

The average length $L(p)$ from the start of the sequence to the end of the D-block is

$$D + \sum_{n=0}^{\infty} ng(1-B)^n B \;=\; D + g\cdot\frac{1-B}{B}$$

If the click does not actually happen at the start of a block, but rather at the midpoint of the next g-block, then the length is $D + g/2 + g\cdot(1-B)/B$. For completeness, we might also consider selections at the end of a block; then the length would be $D+g+g\cdot(1-B)/B$.

Finally, the information rate $R(B) = H(p)/L(p)$ can be maximized numerically with respect to B.

Continuous, Noisy Model

This derivation follows the solution to Exercise 11.1 on p. 189 of MacKay (2003).

Now that the channel is noisy, we have an input variable X (when the user wants to click) and an output variable Y (when the user actually clicks). We wish to find the optimal probability density $f(x)$ of desired click times. Since, by assumption, the user never tries to click before D seconds after the previous click, let $V = X - D$ and $W = Y - D$. Let $f(v)$ be the probability density of waiting time $v + D$ in between clicks. The support of $f(v)$ is the positive real line. We further assume that the user clicks around the desired time with a zero-mean, γ^2-variance normal distribution, i.e. $p(w|v) = \mathcal{N}(w|v, \gamma^2)$.

We wish to maximize $I(X;Y)/L(Y) = I(V;W)/(D + L(W))$ with the constraint that $f(v)$ integrate to unity. It is enough to maximize $J(V;W)/(D + L(W))$, where $J(V;W) = \ln(2)I(V;W)$. We use the method of Lagrange multipliers.

$$\begin{aligned}
\mathcal{L} &= \frac{J(V;W)}{D + L(W)} - \lambda \int_v f(v)\, dv \\
&= \left[\int_w w\, p(w)\right]^{-1} \int_v f(v) \int_w p(w|v) \ln \frac{p(w|v)}{p(w)}\, dw\, dv - \lambda \int f(v)\, dv
\end{aligned}$$

Taking the functional derivative with respect to the density over V yields

$$\begin{aligned}
\frac{\delta \mathcal{L}}{\delta f(v')} &= \frac{1}{D + L(W)} \frac{\delta J(V;W)}{\delta f(v')} - \frac{J(V;W)}{(D + L(W))^2} \frac{\delta L(W)}{\delta f(v')} - \lambda \\
&= \frac{1}{D + L(W)} \left[\int_w p(w|v') \ln \frac{p(w|v')}{p(w)}\, dw - \int_v f(v) \int_w p(w|v) \frac{1}{p(w)} \frac{\delta p(w)}{\delta f(v')}\, dw\, dv\right] \\
&\quad - \frac{J(V;W)}{(D + L(W))^2} \left[\int_w w \frac{\delta p(w)}{\delta f(v')}\, dw\right] - \lambda \\
&= \frac{1}{D + L(W)} \left[\int_w p(w|v') \ln \frac{p(w|v')}{p(w)}\, dw - \int_v p(v) \frac{1}{p(w)} p(w|v')\, dw\right] \\
&\quad - \frac{J(V;W)}{(D + L(W))^2} \left[\int_w w\, p(w|v')\, dw\right] - \lambda \\
&\quad \text{since } p(w) = \int_v p(w|v) f(v)\, dv \\
&= \frac{1}{D + L(W)} \left[\int_w p(w|v') \ln \frac{p(w|v')}{p(w)}\, dw\right] - \frac{J(V;W)}{(D + L(W))^2} v' - \lambda' \\
&\quad \text{where } \lambda' = \lambda + 1, \text{ and } p(w|v') \text{ has mean } v'
\end{aligned}$$

Setting $\delta\mathcal{L}/\delta f(v')$ to the zero function,

$$\forall v', \quad \frac{J(V;W)}{D + L(W)} v' + \lambda'(D + L(W)) = \int_w p(w|v') \ln \frac{p(w|v')}{p(w)}\, dw$$

From the form of $p(w|v')$,

$$\begin{aligned}
\frac{J(V;W)}{D + L(W)} v' + \lambda' L(W) &= -\int_w (2\pi\gamma^2)^{-1/2} \exp\left[-\frac{1}{2\gamma^2}(w - v')^2\right] \ln[p(w)\gamma]\, dw \\
&\quad + \int_w (2\pi\gamma^2)^{-1/2} \exp\left[-\frac{1}{2\gamma^2}(w - v')^2\right] \left[-\frac{1}{2}\ln(2\pi) - \frac{1}{2\gamma^2}(w - v')^2\right] dw \\
&= -\int_w (2\pi\gamma^2)^{-1/2} \exp\left[-\frac{1}{2\gamma^2}(w - v')^2\right] \ln[p(w)\gamma]\, dw + \lambda''
\end{aligned}$$

Since this condition holds for all v', Taylor-expanding $\ln[p(w)\gamma]$ reveals that $\ln[p(w)\gamma]$

must be linear in w. Since $p(w|v) = \mathcal{N}(w|v, \gamma^2)$, we can achieve $\ln[p(w)\gamma]$ linear when $\ln[f(v)/\beta]$ is linear in v, for some β with inverse-time units (so as to ensure that the input to the logarithm is dimensionless). Thus, an optimal input distribution has exponential density: $f(v) = \beta \exp(-\beta v)$. In this case,

$$\begin{aligned}
p(w) &= \int_v f(v)p(w|v)\, dv \\
&= \beta \int_v \frac{1}{2\pi\gamma^2} \exp\{-\beta v\} \exp\left\{-\frac{1}{\sqrt{2\gamma^2}}(w-v)^2\right\}\, dv \\
&= \beta \exp\left\{\frac{1}{2}(\beta^2\gamma^2 + w\beta)\right\} \int_v \frac{1}{\sqrt{2\pi\gamma^2}} \exp\left\{-\frac{1}{2\gamma^2}\left(v - [\beta\gamma^2 + w]\right)^2\right\}\, dv \\
&= \beta \exp\left\{\frac{1}{2}(\beta^2\gamma^2 + w\beta)\right\}
\end{aligned}$$

So

$$\begin{aligned}
J(V;W) &= \int_v \int_w \mathcal{N}(w|v,\gamma^2) \ln \frac{\mathcal{N}(w|v,\gamma^2)}{p(w)}\, dw\, dv \\
&= \int_v \int_w \mathcal{N}(w|v,\gamma^2) \left[-\ln(\sqrt{2\pi}\gamma\beta) - \frac{1}{2\gamma^2}w^2 + \frac{1}{\gamma^2}wv - \frac{1}{2\gamma^2}v^2 - \frac{1}{2}\beta^2\gamma^2 - \beta w\right] dw\, dv \\
&= \int_v \left[-\ln(\sqrt{2\pi}\gamma\beta) - \frac{1}{2\gamma^2}(\gamma^2 + v^2) + \frac{1}{\gamma^2}v^2 - \frac{1}{2\gamma^2}v^2 - \frac{1}{2}\beta^2\gamma^2 - \beta v\right] dv \\
&= -\ln(\sqrt{2\pi}\gamma\beta) + \frac{1}{2}(1 - \gamma^2\beta^2)
\end{aligned}$$

And the information rate $\rho(\beta)$ is

$$\begin{aligned}
\rho(\beta) &= \frac{I(X;Y)}{L(Y)} = \frac{(\ln 2)^{-1} J(V;W)}{D + L(W)} \\
&= \frac{-\log_2(\sqrt{2\pi}\gamma\beta) + (2\ln(2))^{-1}(1 - \gamma^2\beta^2)}{D + \beta^{-1}}
\end{aligned}$$

The information rate can then be maximized numerically with respect to β.

Discrete, Noisy Model with Periodicity

Now that we are dealing with discrete random variables, let m index the g-length interval in $[0, T)$ when the user wants to click, and let n index the unconstrained g-length interval after the initial D recovery time when the user actually clicks. Thus, $m \in \{0, M-1\}$ for $M \approx T/g$, and $n \in \{0, 1, 2, \ldots\}$. We can express n as the ordered pair (i_n, j_n) defined by the constraints $n = i(n) + j(n)M$ and $i(n) \in \{0, M-1\}$. We further assume, as in the discrete, noiseless model (Section 2.2.1 and Appendix A), that $p(i(n)|m) = \mathbb{1}_{\{i(n)=m\}}$.

And we put a geometric distribution with success probability $q = 0.95$ on the periods: $p(j(n)) = (1-q)^{j(n)}q$. Then

$$\begin{aligned} p(n) &= \sum_{m=0}^{M-1} p(n|m)p(m) \\ &= p(j(n)|i(n))p(i(n)|i(n))p(i(n)) \\ &= (1-q)^{j(n)}q \cdot 1 \cdot p(i(n)) \end{aligned}$$

So the mutual information between the input and output is

$$\begin{aligned} I(X;Y) &= I(m;n) \\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{\infty} p(n|m)p(m)\log_2\frac{p(n|m)}{p(n)} \\ &= \sum_{m=0}^{M-1}\sum_{j=0}^{\infty} p(j(n)|m)p(m)\log_2\frac{p(j(n)|m)}{(1-q)^{j(n)}q \cdot p(m)} \\ &\quad \text{using the convention } 0\log_2 0 = 0 \\ &= \sum_{m=0}^{M-1}\sum_{j=0}^{\infty} p(j(n)|m)p(m)\log_2\frac{1}{p(m)} \\ &= \sum_{m=0}^{M-1} p(m)\log_2\frac{1}{p(m)} \end{aligned}$$

This mutual information is maximized by the uniform distribution $p(m) = M^{-1} \approx g/T$.

The expected time between clicks is

$$\begin{aligned} L(Y) &= D + g\sum_{n=0}^{\infty} np(n) \\ &= D + g\sum_{j=0}^{\infty}\sum_{i=0}^{M-1}[i+Mj]p(i(n))p(j(n)) \\ &= D + g[M/2 + M \cdot 0.05/0.95] \\ &\approx D + T/2 + T \cdot 0.05/0.95 \end{aligned}$$

Combining these results, we find the information rate.

$$\begin{aligned} \hat{R}(T) &= \frac{I(X;Y)}{L(Y)} \\ &\approx \frac{\log_2(T/g)}{D + T/2 + T \cdot 0.05/0.95} \end{aligned}$$

The information rate can be maximized numerically with respect to T.

B Normal Scale Rule

Consider a Parzen-window estimate $\hat{g}_h$ of some density g. For kernel distribution K, bandwidth h, and data points $\{x_i\}_{i=1}^n$ drawn from g, the estimator function is defined to be

$$\hat{g}_h(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x - x_i}{h}\right)$$

The kernel distribution $K = \mathcal{N}(0, 1)$ is a standard choice (Wand and Jones, 1995), and we use it here. The bandwidth h is a free parameter. We wish to determine an appropriate h.

The *asymptotic mean integrated square error* (AMISE) of $\hat{g}_h$ is an estimate of how well $\hat{g}_h$ approximates g. It is defined as follows.

$$\text{AMISE}(\hat{g}_h) = \frac{1}{nh}R(K) + \frac{1}{4}h^4\mu_2^2(K)R(g'')$$

In this definition, the function g'' is the second derivative of g. The functional R takes a function argument f and returns $\int_{x\in\mathcal{X}} f^2(x)\ dx$, where $\mathcal{X}$ is the support of f. The functional μ_2 returns the second moment of its density-function input.

The value of h that minimizes $\text{AMISE}(\hat{g}_h)$ is easily found.

$$h^* = \left(\frac{R(K)}{\mu_2^2(K)R(g'')n}\right)^{1/5} \tag{5}$$

In order to calculate h^*, we need to calculate $R(K)$, $\mu_2(K)$, and $R(g'')$. Problematically, g is meant to be an unknown density; indeed, it is the density we are attempting to approximate. The normal scale rule sidesteps this dilemma by calculating h^* as in Eq. 5 under the assumption that g is a normal density with variance γ^2. Then we have $R(g'') = 3/(8\sqrt{\pi}\gamma^5)$, where γ can be estimated from the data (Wand and Jones, 1995). Call this estimate $\hat{\gamma}$. Finally, for K a zero-mean, h^2-variance normal distribution, $R(K)/\mu_2^2(K) = (4\pi)^{-1}$. Number-crunching finally yields

$$h_{\text{NS}} = 1.06\hat{\gamma}n^{-1/5}$$

in agreement with Silverman (1986).

There are more careful ways of calculating h^* from Eq. 5 than the normal scale rule, but this rule allows quick calculation of the kernel bandwidth—an important considera-

tion in our application.

References

Baljko, M. and Tam, A. (2006). Indirect text entry using one or two keys. In *Assets '06: Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 18–25, New York, NY, USA. ACM.

Damper, R. (1984). Text composition by the physically disabled: A rate prediction model for scanning input. *Applied Ergonomics*, **15**, 289–296.

Darragh, J. J., Witten, I. H., and James, M. L. (1990). *The Reactive Keyboard: A Predictive Typing Aid*, volume 23. IEEE Computer Society, Los Alamitos, CA, USA.

Evreinov, G. and Raisamo, R. (2004). Optimizing menu selection process for single-switch manipulation. In *Proceedings of the 9th International Conference on Computers Helping People with Special Needs (ICCHP 2004)*, pages 836–844, Paris, France. Springer-Verlag Berlin/Heidelberg.

Glennen, S. and DeCoste, D. (1997). *The Handbook of Augmentative and Alternative Communication*. Singular Publishing Group, San Diego, CA, USA.

Hsieh, M. and Luo, C. (1999). Morse code text typing training of a teenager with cerebral palsy using a six-switch Morse keyboard. *Technology and Disability*, **10**(3), 169–173.

ITU Radiocommunication Assembly (2004). International Morse code.

Kilgarriff, A. (1997). Putting frequencies in the dictionary. *International Journal of Lexicography*, **10**(2), 135–155.

Kilgarriff, A. (1998). BNC database and word frequency lists. Available from http://www.kilgarriff.co.uk/bnc-readme.html. Accessed on 18 March, 2009.

Koester, H. and Levine, S. (1994). Learning and performance of able-bodied individuals using scanning systems with and without word prediction. *Assistive Technology*, **6**(1), 42–53.

Koester, H. and Levine, S. (1996). Effect of a word prediction feature on user performance. *Augmentative and Alternative Communication*, **12**, 155–168.

Kristensson, P. O. and Denby, L. C. (2009). Text entry performance of state of the art unconstrained handwriting recognition: a longitudinal user study. In *CHI '09: Proceedings of the 27th International Conference on Human Factors in Computing Systems*, pages 567–570, New York, NY, USA. ACM.

Lee, E. S. (1999). Essays about computer security. Technical report, Centre for Communications Systems Research, University of Cambridge.

Lesher, G., Moulton, B., and Higginbotham, D. (1998). Techniques for augmenting scanning communication. *Augmentative and Alternative Communication*, **14**(2), 81–101.

MacKay, D. J. (2007). Another one-button dynamic mode for Dasher: 'two-click mode'. Technical report, Cavendish Laboratory, University of Cambridge.

MacKay, D. J. and Ball, C. J. (2006). Dasher's one-button dynamic mode—theory and preliminary results. Technical report, Cavendish Laboratory, University of Cambridge.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press. Available from http://www.inference.phy.cam.ac.uk/mackay/itila/.

MacKay, D. J. C., Ball, C. J., and Donegan, M. (2004). Efficient communication with one or two buttons. In R. Fischer, R. Preuss, and U. von Toussaint, editors, *Maximum Entropy and Bayesian Methods*, volume 735 of *AIP Conference Proceedings*, pages 207–218, Melville, NY, USA. American Institute of Physics.

MacKenzie, I. S. (2002). KSPC (keystrokes per character) as a characteristic of text entry techniques. In *Proceedings of the Fourth International Symposium on Human Computer Interaction with Mobile Devices*, pages 195–210, Heidelberg, Germany. Springer-Verlag.

MacKenzie, I. S. and Soukoreff, R. W. (2003). Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, pages 754–755, New York, NY, USA. ACM.

Sensory Software International Ltd. (2008). *The Grid 2: Version 2.4*. Sensory Software International Ltd.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, London, UK.

Simpson, R. and Koester, H. (1999). Adaptive one-switch row-column scanning. *IEEE Transactions on Rehabilitation Engineering*, **7**(4), 464–473.

Sun Java Desktop System Documentation Team (2004). *GOK Manual V2.3*. Sun Microsystems, Inc.

Szeto, A. Y. J., Allen, E. J., and Littrell, M. C. (1993). Comparison of speed and accuracy for selected electronic communication devices and input methods. *Augmentative and Alternative Communication*, **9**(4), 229–242.

Venkatagiri, H. S. (1999). Efficient keyboard layouts for sequential access in augmentative and alternative communication. *Augmentative and Alternative Communication*, **15**(2), 126–134.

Wand, M. P. and Jones, M. C. (1995). *Kernel smoothing*. Chapman & Hall/CRC, London, UK.

Ward, D. J. and MacKay, D. J. C. (2002). Fast hands-free writing by gaze direction. *Nature*, **418**(6900), 838.

Ward, D. J., Blackwell, A. F., and MacKay, D. J. C. (2000). Dasher—a data entry interface using continuous gestures and language models. In *UIST '00: Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, pages 129–137, New York, NY, USA. ACM.

Words+, Inc. (2000). *EZ Keys^{TM} User Manual*. Words+, Inc.

Words+, Inc. (2004). *EZ Keys^{TM} XP Product Specification Sheet*. Words+, Inc.