Dasher - A Gesture-Driven Data Entry Interface For Mobile Computing

*David J. Ward*, †Alan F. Blackwell, ‡David J.C. MacKay

Cavendish Laboratory, Computer Laboratory, Cavendish Laboratory
University of Cambridge, University of Cambridge, University of Cambridge
U.K., U.K., U.K.
July 12, 2001

Running head title: Dasher

Abstract

Existing devices for communicating information to computers are either bulky, slow, or unreliable. Dasher is an interface incorporating language modelling and driven by continuous two-dimensional gestures, e.g. a mouse, a stylus, or eye-tracker. Tests have

---

1David Ward is a Physicist with interests in Human Computer Interaction, Language Modelling and Neural Networks; he is a graduate student in the Department of Physics at the University of Cambridge, UK.

2Alan Blackwell is a Psychologist and Engineer with interests in Human Computer Interaction and Software Product Design; he is a lecturer in the Computer Laboratory at the University of Cambridge, UK.

3David MacKay is a Physicist with interests in machine learning and information theory; he is a reader in the Department of Physics at the University of Cambridge, UK.
shown that, after an hour of practice, novice users reach a writing speed of about 20 words per minute while taking dictation. Experienced users achieve writing speeds of about 34 words per minute, compared with typical ten-finger keyboard typing of 40–60 words per minute.

Although the interface is slower than a conventional keyboard, it is simple to use, and could be used on personal data assistants and by motion–impaired computer users. Dasher can readily be used to enter text from any alphabet.
Contents

1 THE INFORMATION CONTENT OF TEXT AND OF HAND MOVEMENTS 6

2 RELATIONSHIP TO PREVIOUS WORK 7

2.1 Previous Solutions .............................................. 8
Miniature and Rearranged Keyboards .............................. 8
Gestural Alphabets ................................................. 10
Dynamic Selection .................................................. 10

2.2 Previous Work On Dasher ................................... 11

3 DASHER 11

3.1 Entering Text ................................................... 11

3.2 How the Probabilistic Model Determines the Screen Layout 12

3.3 Dynamics of the Interface .................................. 13

3.4 Correcting Mistakes .......................................... 16

3.5 Benefits of Continuous Gestures and Language Modelling 17

3.6 Horizontally Modified Display .............................. 18

3.7 Vertically Modified Display ................................. 18
4 ALTERNATIVE CHARACTER SETS

4.1 Capitalization ................................................................. 19
4.2 Japanese ................................................................. 19

5 THE LANGUAGE MODEL .................................................. 20

6 EMPIRICAL EVALUATION .................................................. 21

6.1 Evaluation Approach .................................................. 22
6.2 Pilot Experiment ......................................................... 22
6.3 Method ................................................................. 23
Subjects ................................................................. 23
Task ................................................................. 23
Apparatus ................................................................. 23
Procedure ................................................................. 24
Configuration ................................................................. 24
Analysis ................................................................. 25
6.4 Results ................................................................. 26
Text-entry Speeds ..................................................... 26
Writing Errors ................................................................. 26
The Power Law ................................................................. 27

7 INFORMATION RATE OF DASHER .................................. 28

7.1 Current Information Rate of Dasher .................................. 28
7.2 Potential Information Rate of Dasher ........................................ 28
Time-Delay Model ........................................................................... 29

8 APPLICATION TO MOBILE COMPUTING .................................. 31

9 DISCUSSION .............................................................................. 32
  9.1 Comparison to Other Devices .................................................. 32
  9.2 Attention Demand ................................................................. 32
  9.3 Potential Writing Speed .......................................................... 32
  9.4 Suitability to Mobile Text Entry .............................................. 33

10 FURTHER WORK ...................................................................... 34
  10.1 Eye-tracking ........................................................................ 34
  10.2 Modelling ........................................................................... 35
  10.3 Japanese ............................................................................. 35

11 CONCLUSION .......................................................................... 36

NOTES ......................................................................................... 36
  Acknowledgements ...................................................................... 36
  Support ..................................................................................... 36
  Author’s Addresses .................................................................... 37

REFERENCES ............................................................................. 37
1 THE INFORMATION CONTENT OF TEXT AND OF HAND MOVEMENTS

Existing devices for communicating information to computers are either bulky, slow to use, or unreliable. Dasher is a new interface incorporating language modelling and driven by continuous two-dimensional gestures, e.g. a mouse, touchscreen, or eyetracker. Dasher is easy to learn, attains moderate writing speeds and is small enough to be implemented on personal data assistants.

Conventional keyboards require the user to make one or two gestures per character (one for lower-case and two for upper-case characters). Each gesture is a selection of 1 from 80 keys, so the keyboard has the capacity to read about $\log_2(80) = 6.3$ bits per gesture. However, the entropy of English text is roughly 1 bit per character (Shannon, 1993), so existing keyboards are inefficient by a factor of six. This inefficiency manifests itself in the fact that when typing, people make errors that are clearly not English.

The keyboard is inefficient for two reasons. First, text is usually highly redundant, yet users are forced to type most or all of the characters in a document. Second, keyboards register only contact events, whereas humans are capable of fine motor movements. Human motor capabilities dictate an upper limit for finger tapping frequency, and hence the information rate achievable from successive key presses, no matter how small the distance between keys. Continuous pointing gestures have the potential to convey more information through high-resolution measurement of position.
According to MacKenzie’s analysis of Fitts’s experiments measuring one-dimensional pointing speed and accuracy, the rate at which information can be conveyed by one-dimensional pointing is about 8.2 bits per second (MacKenzie, 1992a).

So using just one finger in a one-dimensional pointing environment, it should be possible to write English at a rate of 8.2 characters per second. For this to be achieved we must connect the finger to a good probabilistic model of the English language. This character rate corresponds to about 100 words per minute. How close we can get to this figure depends on the quality of our model of English, and on whether we can map finger gestures to text in a way that people can easily learn.

2 RELATIONSHIP TO PREVIOUS WORK

Personal organizers with touch-sensitive screens are small and convenient to carry, but at present, data is usually entered using miniature keyboards or a slow handwriting system. Speech recognition systems are almost as fast as conventional keyboards (Jecker, 1999), but are not always socially acceptable in a crowded office and users may also suffer from voice fatigue.

Gesture–based text entry systems involve two significant tradeoffs: between potential efficiency and training time, and between device size and character-set size. Research on keyboards for conventional typing has addressed both efficiency versus training (Norman & Fisher, 1982) and the effect of keyboard size (Sears, Revis, Swatski, Crittenden, & Shneiderman, 1993). In conventional keyboards, efficiency depends on relative key size (according
to Fitts’ law (MacKenzie, 1992b)) and on the mapping between movement over the keyboard configuration and English letter sequences. Learning speed appears to be a function of similarity to the QWERTY layout (Gordon, Henry, & Massengill, 1975), which has almost ubiquitous familiarity, even in non-typists (Norman & Fisher, 1982).

The difficulty of resolving these tradeoffs has led to predominance of the QWERTY keyboard, even in the face of mechanical improvements such as the Dvorak keyboard. However, in the case of hand-held devices no single text-entry system is dominant.

2.1 Previous Solutions

Approaches to text entry in hand-held devices can be grouped into three broad categories; miniature and rearranged keyboards; gestural alphabets; and dynamic selection techniques.

Miniature and Rearranged Keyboards

The challenge in this approach is to present as much of the alphabet as possible while retaining sufficiently large keys. The number of keys is often reduced, and some selection mechanism is used to toggle between alternate sets. The ‘Half-QWERTY’ device uses half of the QWERTY keyboard and the space bar toggles between the two halves. Evaluation showed typing speeds of 34.7 words per minute after ten sessions, each lasting 50 minutes (Matias, MacKenzie, & Buxton, 1993). The ‘Fitaly’ keyboard arranges keys in a compact square optimized for minimal hand travel with one-finger typing (MacKenzie, Zhang, & Soukoreff, 1999).
The T9 text input system uses the conventional mapping of the 26 alphabetic characters onto a 12-key telephone keypad, with a dictionary to help disambiguate alternative readings (Silfverberg, MacKenzie, & Korhonen, 2000). Both Fitaly and T9 are available as alternative text entry systems for the Palm Pilot. T9 is also increasingly well-known as a predictive text entry system used in mobile phones. An experiment with 20 subjects compared novice and expert mobile phone text users (James & Reischel, 2001), using two different styles of text: chat messages and newspaper extracts. For chat messages, experts achieved 26 words per minute with T9, versus 11 wpm for multi-tap. Novices achieved 11 wpm with T9 versus 10 wpm with multi-tap.

A mathematically optimized keyboard has been proposed in which keys are arranged on a hexagonal grid (Hunter, Zhai, & Smith, 2000). Predicted text entry speed is 42 to 44 wpm. Recent evaluation (Zhai & Smith, 2001) reports writing speeds of 8.9 to 9.7 wpm.

Some evaluation has been done to measure the effect of using standard QWERTY keyboard layouts reduced in size on a one-finger touch screen (Sears et al., 1993). Typing speed for experts after 30 minutes was 32.5 words per minutes with a large keyboard (246 mm wide), and dropped to 21.1 words per minute on a very small one (68 mm wide).

It is also possible to reduce keyboard size with ‘chorded’ schemes having fewer keys, but in which multiple keys are pressed at one time. Performance of the ternary chorded keyboard (TCK) (Kroemer, 1992) was about 16 words per minute after 600 minutes training.
Gestural Alphabets

Most hand-held pen devices are supplied with software to recognize handwritten characters, possibly using a special alphabet. The gesture sets range from those that are designed to be very efficient to those that emphasize ease of learning. Early devices attempted, with limited success, to recognize users’ natural handwriting - and hence required no learning on the part of the user (as in the case of the Apple Newton). Current systems – Graffiti (MacKenzie & Zhang, 1997) for the Palm Pilot, and Jot for Windows CE (also available on the Pilot) – improve efficiency while maintaining some resemblance to alphabetic shapes to assist learning. Researchers have also proposed more efficient alphabets such as Unistrokes, which maps simple gestures to common characters regardless of mnemonic similarity (Goldberg & Richardson, 1993). The inventors of Unistrokes estimate a rate of 40.8 words per minute, without any empirical testing.

Dynamic Selection

Dynamic selection techniques share characteristics of miniature keyboards and gestural alphabets. The user moves the pen in the direction of a selection region, which either selects a character or reveals a further set of alternatives. Examples include T-Cube (Venolia & Neiberg, 1994) and Quikwriting (Perlin, 1998). Quikwriting is commercially available for the Palm Pilot.

An unusual alternative approach has been to start with a standard QWERTY keyboard, but then rearrange the keys after each keystroke to place the most probable next letters
near the centre. Results with the FOCL keyboard were around 10 words per minute after 10 sessions of 15 minutes (Bellman & MacKenzie, 1998).

These dynamic selection devices are subject to a tradeoff between efficiency and ease of learning. The arrangement of nested regions or sets of alternatives must favour the selection of common characters, while still being learnable. Dasher offers an alternative dynamic selection technique which is easier to learn, but improves efficiency by dynamically setting the size of the selection targets according to a language model. The system can be used with arbitrary alphabets; special characters and capitalization can be included without modification.

2.2 Previous Work On Dasher

We first described Dasher version 1.3 in (Ward, Blackwell, & MacKay, 2000). The present paper includes the results from that paper, for completeness. In addition, this paper includes: (a) application to mobile computing; (b) details of the dynamics of Dasher; (c) modifications included in Dasher 1.5; (d) alternative character sets; and (e) theoretical analysis of maximum information rate. The current version of Dasher is 1.6.

3 DASHER

3.1 Entering Text

[Figure 1 about here.]
We first describe how Dasher is used to enter the word ‘the’. Figure 1(a) shows the initial configuration, with an alphabet of 27 characters displayed alphabetically in a column. There are 26 lower case letters and the symbol ‘_’ represents a space. The user writes the first letter by making a gesture towards the letter’s rectangle. The trails show the user moving the mouse towards the letter ‘t’.

In a continuous motion, the point of view zooms towards this letter (figure 1(b)). As the rectangles get larger, possible extensions of the written string appear within the rectangle that we are moving towards. So if we are moving into the ‘t’, rectangles corresponding to ‘ta’, ‘tb’, …, ‘th’, …, ‘tz’ appear in a vertical line like the first line. The heights of the rectangles correspond to the probabilities of these strings, according to the language model. In English, ‘ta’ is quite probable, ‘tb’ is less so, ‘th’ is very probable. So it is easy to gesture our point of view into ‘th’ (figure 1(c)), and from there into ‘the’ (figure 1(d)).

Dasher has a website, http://wol.ra.phy.cam.ac.uk/djw30/dasher/ which has a non-interactive demo. Dasher can be downloaded for Microsoft Windows and UNIX platforms.

3.2 How the Probabilistic Model Determines the Screen Layout

[Figure 2 about here.]

In a given context, we display the alphabet of possible continuations as a column of characters as shown in figure 2. The division of the right-hand vertical is analogous to arithmetic coding (Bell, Cleary, & Witten, 1990). Let our alphabet be \( A_X = \{a_1, a_2, \ldots, a_I\} \). We divide the real line \([0,1)\) into \( I \) intervals of lengths equal to the probabilities \( P(x_i = a_i) \).
We subdivide the interval $a_i$ into intervals denoted $a_ia_1, a_ia_2, a_ia_3, \ldots, a_ia_I$, such that the length of the interval $a_ia_j$ is

$$P(x_1 = a_i, x_2 = a_j) = P(x_1 = a_i)P(x_2 = a_j|x_1 = a_i).$$

(1)

The language model described in section 5 assigns the probabilities. The interval $[0,1)$ can be divided into a sequence of intervals corresponding to all possible finite length strings $x_1x_2\ldots x_n$, such that the length of an interval is equal to the probability of the string given our model.

This sequence of intervals corresponds to the alphabetically ordered sequence of books in Borges’s ‘Library of Babel’ (Borges, 2000), with the size of each book being proportional to the probability of its contents under the language model. The user writes by gliding into the library and selecting the desired book.

In figure 2, the rectangles are shown as squares, but we can use rectangles with any aspect ratio.

3.3 Dynamics of the Interface

[Figure 3 about here.]

The Steps Parameter, $S$

The pointing device controls a continuous zooming-in of the users’ point of view. The interface zooms in so that the place under the pointer passes through the cross-hair $S$ frame.
updates later (figure 3).

The left–right coordinate controls the rate of zooming–in, and the vertical coordinate determines the point on the right–hand vertical that is being zoomed into. The intuitive idea is that the user should point where they want to go.

**Dynamical equations**

We use two co–ordinate systems to describe the dynamics of the interface. The first co–ordinate system specifies the *visible interval* on the real line (introduced in section 3.2). The second are the *screen co–ordinates*.

In the initial configuration, the *visible interval* is [0,1). The visible interval after $i$ screen updates can be specified by the centre of the interval, $C_i$, and the height, $H_i$ (figure 4). In the initial configuration,

\[
C_0 = 0.5, \quad H_0 = 1.
\]

[Figure 4 about here.]

[Figure 5 about here.]

Figure 5 shows the co–ordinate system for the pointer $(m_x,m_y)$. The position of the cross–hair is $(k_x,0.5)$. The following dynamical equations give the next interval, $(C_{i+1}, H_{i+1})$, in
terms of the current interval, \((C_i, H_i)\):

\[
    r = \frac{m_x}{k_x},
\]
\[
    r' = r^{1/S},
\]
\[
    H_{i+1} = r' H_i,
\]
\[
    C_{i+1} = C_i + H_i (0.5 - m_y) \frac{1 - r'}{1 - r}.
\]

**Setting the Steps Parameter, \(S\)**

The steps parameter determines how responsive the interface is; the smaller it is, the faster the point of view can be changed. Inexperienced users usually require a relatively large value of \(S\), say 70. A high value of \(S\) makes the system relatively unresponsive to large or inaccurate movements. As users improve, \(S\) can be decreased, enabling higher writing speeds.

However, when measuring their writing speed, we decided not to allow users to set \(S\), otherwise their performance would be subject to their ability to choose this parameter as well as their skill in using the interface.

The speed of Dasher depends on the frame-rate and CPU power so the appropriate value of \(S\) is system dependent. We designed an on-line algorithm to set \(S\) to a suitable value.

Let \(x\) be the normalized distance of the pointer from the right hand side of the screen,

\[
    x = \frac{m_x}{k_x}.
\]
This normalization ensures that $x$ is scale invariant.

Figure 6 qualitatively shows a typical user’s response to different values of $S$. At point A on the graph, $S$ is relatively small. The interface is too responsive and the user keeps the pointer close to the cross-hair, $x = 1$. At B, $S$ is relatively large. The user is able to hold the pointer near the right hand vertical, $x = 0$.

Experience shows that good typing speeds are obtained in the range $0.1 < \bar{x} < 0.3$. Therefore, we used the following update rule for $S$:

\begin{align}
\text{if } \bar{x} > 0.3, \quad & S \rightarrow S + 1; \\
\text{if } \bar{x} < 0.1, \quad & S \rightarrow S - 1.
\end{align}

We alter the time-scale of adaptation by adjusting the size of the sample, $N$. In practice, the time-scale was around 15 seconds.

### 3.4 Correcting Mistakes

Inexperienced users of Dasher often inadvertently follow a path other than that corresponding to the desired text. This generally happens when the desired string contains a high probability string followed by a low probability string. The user may react too late, inadvertently continuing forward motion to accept a high probability string in the neighbourhood of the desired text. After entering a high-probability neighbour, the desired text may not
even be visible.

There are a number of ways in which we could handle corrections. Currently, the dynamics allow the user to ‘back up’ by pointing to the left of the cross-hair. The higher the leftward offset from the cross-hair, the faster the zooming-out.

[Figure 7 about here.]

In figure 7, the user has written the string ‘might’ instead of ‘mean’. To alter this string, the user points to the left side of the display. Then the user points at the desired string, ‘mean’. An alternative mechanism, not yet tested, is to use an extra character to indicate that a misspelling has occurred.

3.5 Benefits of Continuous Gestures and Language Modelling

Dasher is driven by continuous gestures, so inaccurate gestures can be compensated for by later gestures, the way a driver keeps a car on the road. When using a conventional keyboard we select one character per gesture but in Dasher some gestures select more than one character and therefore Dasher has the potential to convey information at higher rates than a keyboard. Figure 8 shows completions of the string ‘object’. Grammatically correct continuations can easily be selected with a single gesture

[Figure 8 about here.]

The language model make spelling mistakes less likely.
3.6 Horizontally Modified Display

When testing Dasher version 1.0, many users found that they were not able to see an adequate history and that the rate of zooming-out was too slow.

In figure 9 the user writes ‘laboratory’ and the letters ‘la’ pass out of the display before finishing the word. In version 1.5, we put the horizontal coordinates of the squares through a non-linear mapping before rendering them on the screen. This mapping is linear on the right hand side, but logarithmic on the left hand side. The effect can be seen by comparing figures 9(a) and 9(b), which display the same interval on the real line. This display makes it easier for users to go back and correct mistakes.

If the user points to ‘l’ of laboratory using the modified display, the ‘l’ will pass through the cross-hair in S frame updates. With a square aspect ratio, it will take more frame updates to reach the ‘l’ when pointing at the same screen co-ordinates. Therefore, the modified display gives a higher rate of zooming-out.

3.7 Vertically Modified Display

We also put the vertical co-ordinates through a non-linear transformation as shown in figure 10. This has advantages of increasing the maximum speed of vertical scrolling, and gives users more time to react before a desired letter disappears out of view.
4 ALTERNATIVE CHARACTER SETS

It is easy to incorporate any character set into Dasher. We give two examples.

4.1 Capitalization

We can extend the character set introduced in section 3.1 to 53 symbols by adding capital letters (figure 11). In Dasher version 1.5, the lower and upper case letter are interleaved. When an expert user took dictation and wrote at 42 words per minute with lower case and 41 words per minute with both upper and lower case. It appears that the performance of Dasher is not significantly altered, even though size of the character set has doubled. Notice how the language model predicts capitals where they are probable (after ‘Mr_’) and lower case letters after ‘Mr_K’, ‘Mr_J’, etc.

4.2 Japanese

Japanese is written in three different ‘scripts’.

- **Kanji** are Chinese characters or ideograms. Each character represents a word.

- **Hiragana** is a system in which each symbol represents a spoken syllable.

- **Katakana** is a phonetic alphabet, similar to Hiragana, but used for writing words of non-Japanese origin.
There are thousands of Kanji symbols, so for our first prototype, we decided to use the Hiragana character set. In the future we plan to make a combined Hiragana-Kanji system in which Kanji characters are selected by their relative probability as the completion of a phonetic Hiragana spelling. Hiragana consists of 83 symbols, shown in figure 12. The symbols are listed in their conventional order for computers.

While Hiragana keyboards are available, many Japanese users type on conventional QWERTY keyboards. Each Hiragana symbol has a Romanisation, e.g. ka, ki, ku. Typically, two key presses are required to enter one Hiragana symbol.

The Hiragana version, ‘JDasher’, show in figure 13 is incorporated into Dasher versions 1.5 onwards. One ordering of characters is identical to that shown in figure 12. The character set can be split into 10 classes, based on sound. For example, the characters (ka, ki, ku, ke, ko) and (sa, shi, su, se, so) form two different classes. In Dasher, we gave each class a fixed colour to aid the users’ search. In an alternative character set, we separated the diacritical marks from the base characters. Evaluation of JDasher is continuing.

5 THE LANGUAGE MODEL

When choosing a language model for Dasher we considered two qualities: how well the language model compresses text and the time taken to compute the probability of a character.
Dasher must calculate many probabilities each time the screen updates.

The model in Dasher version 1.6 is based on a popular text compression algorithm called prediction by partial match (PPM) (Bell et al., 1990). PPM is a context-based algorithm it uses the preceding characters to predict the next one. The maximum size of the context is the order of the model. We use a variant of the algorithm called PPM5D+ (Teahan, 1995), which is fifth order, and can compress most English text to around 2 bits per character. There are slightly better algorithms (Gilchrist, 2000) but PPM5D+ is simple and fast.

Given a context, we compute the probabilities of all the symbols in the alphabet. The conditional probabilities determine the intervals in equation 1.

When using Dasher, it is difficult to select a character that has a very small probability. Therefore we add a small fixed probability $\delta$ to every character and then renormalize the probabilities. In Dasher 1.6, the default value of $\delta$ is 0.002.

As the user enters text the characters can be fed back into the PPM algorithm, hence adjusting the future probabilities in accordance with that user’s vocabulary.

6 EMPIRICAL EVALUATION

The evaluation below was carried out on Dasher version 1.3. The results were also presented in (Ward et al., 2000).
6.1 Evaluation Approach

An objective of this research was to assess Dasher in a realistic text-entry task. Dasher requires uninterrupted visual attention, so we measured the writing speed for dictated text.

Dasher has been used with a variety of position controllers. For the experiment, we originally hoped to use a stylus on a touchscreen the size of a hand-held computer. However, our original touchscreen required a contact force of 2 ounces. This is difficult to maintain when using a constant gestural interface such as Dasher, so we used a standard mouse in the experiment described below.

6.2 Pilot Experiment

An initial approach to the evaluation was for the user to transcribe text passages spoken by a speech synthesizer. The speed of speech was dynamically controlled to stay just ahead of the words being entered by the user.

The results of the three-subject pilot experiment were used to estimate an appropriate length for each experimental session, to design the session format, and estimate the number of subjects and training sessions that would be required to observe significant learning. Subjects found the speech synthesizer hard to understand, so we used recorded human speech for the main experiment.
6.3 Method

Subjects

The main evaluation experiment involved ten experimental subjects, recruited from the students and staff of the Cavendish Laboratory. All subjects had vision corrected to normal and spoke English as their first language. Subjects were paid for participating in the experiment. None of the subjects had previous experience with Dasher.

Task

The experimental task, as in the pilot experiment, was to enter text dictated from Jane Austen’s *Emma*. We selected 18 extracts at random from the total 883 Kbytes of text in *Emma* to form the test set. The remaining 866 Kbytes of text was used to train the language model.

Austen has a distinctive writing style. This allowed us to test Dasher in a reasonable simulation of typical usage where the language model would have been trained on previous prose written by the same user.

Apparatus

Platform           Pentium II 300 MHz running Linux 2.0.38
Monitor           38cm LCD display
Input Device       mouse

The dictated passages were recorded as a series of audio files, with each file containing a
short phrase. The text entered by the subject was monitored so that as he or she wrote the penultimate word in each phrase, the next audio file was played automatically. This allowed subjects to enter text continuously, with dictation proceeding at a comfortable speed. A simple synchronization algorithm compensated for misspelt words when identifying the end of a dictation phrase. Subjects could press a key with their free hand to repeat the last phrase if it was forgotten or misheard. There was no limit to the number of repeats.

**Procedure**

Subjects completed 6 experimental sessions of approximately half an hour each. Each session consisted of three exercises. Subjects first took dictation using Dasher for 5 minutes. They then took dictation using the keyboard (conventional typing) for 2 minutes. Finally they took dictation using Dasher for another 5 minutes. Sessions were generally spaced at daily intervals, with no more than two on any day and no more than three days between sessions. Subjects were instructed at the start of each exercise to write as fast as possible, and were told that they could correct simple mistakes within the current word. They were told not to correct mistakes in previous words.

**Configuration**

Dasher was configured with the following parameters:
The $\delta$ parameter, (defined in section 5), specifies a lower limit to the probability of a character. For each exercise, the Steps parameter $S$ (defined in section 3.3) was initialized to 60.

**Analysis**

The control software counted the characters entered in each period of dictation. It also counted word-level errors; a deletion, insertion, or replacement of a word (possibly with a misspelt version of the word) – each counted as an error. The Viterbi algorithm (Bellman, 1957; Viterbi, 1967) was used to determine the minimum number of errors which, when applied to the target text, produced the text entered by the user.

Data on character entry speed and proportion of errors were collected for the twelve periods of Dasher use, and for the six periods of conventional typing. We compared improvement in performance over the twelve periods of Dasher use in order to evaluate the effect of practice on Dasher performance. The interpolated conventional typing tests allowed us to estimate what proportion of this practice effect could be attributed to practice with the experimental dictation software, as opposed to practice with the Dasher text entry method.
The keyboard task should be viewed as a control condition for the experimental procedure, rather than a serious attempt to measure keyboard speeds under realistic conditions of use.

6.4 Results

Text-entry Speeds

[Figure 14 about here.]

All subjects successfully completed the 6 sessions (figure 14). The dictation texts were used in the same order for all subjects. As a result, variation in vocabulary between the texts resulted in consistently lower speeds for some exercises. For example exercise 8 included a particularly uncommon word which caused users to hesitate. As a result, the raw data shown figures 14, 17, and 20 include apparent dips in writing speed during those exercises.

To investigate these correlations further, we plot information rate rather than writing speed (figure 15). The correlations between subjects over exercises 7, 8, and 12 are no longer apparent because uncommon words have a higher information content, and hence information rate is relatively continuous despite discontinuities in writing speed when they are encountered.

[Figure 15 about here.]

Writing Errors

[Figure 16 about here.]
Errors were expressed as the percentage of words written differently from the test text. The rate of keyboard errors in figure 16 show that users took a little while to get used to the dictation system (assuming that their typing skill did not improve during the experiment). However, after the third keyboard exercise, the errors in both Dasher and keyboard exercises change very little.

Towards the end of the evaluation the average error rate is 3% when using Dasher and 7% when using the keyboard. On 10 occasions, Dasher users made no errors while keyboard users invariably made errors.

The Power Law of Practice

The power law of practice (Newell & Rosenbloom, 1981) predicts that the writing speed $dN/dt$ should increase with training time $t$ as

$$\log \left[ \frac{dN}{dt} \right] = C + D \log (t),$$

(11)

where $C$ and $D$ are constants. From figure 17 we can see that the data give a reasonably good fit to the power law. The gradient $D = 0.32 \pm 0.05$ (related to the learning rate) is similar for all 10 subjects, but $C = 3.22 \pm 0.45$ (the log rate after one minute) varies more among subjects. The users with faster initial speeds were generally the fastest after an hour training. Consequently, we can estimate a users’ future typing speed with reasonable accuracy, based on a small amount of use, say 15 minutes.

[Figure 17 about here.]
7 INFORMATION RATE OF DASHER

7.1 Current Information Rate of Dasher

One of the authors has used Dasher for substantially longer than the experimental training period; a total of a few hours use, mostly during development testing rather than sustained text entry. During a number of trials with the experimental dictation text from Jane Austen’s *Emma*, he achieved an average speed of 170 characters per minute (34 wpm).

The experimental texts have an information content of 1.7 bits per character with respect to the model. Expert performance of 170 characters per minute therefore corresponds to an information rate of 4.8 bits per second.

7.2 Potential Information Rate of Dasher

Apart from the language model, two factors might limit the rate at which a user can enter information with Dasher. First, steering Dasher in the required direction involves visual motor control similar to pole-balancing; the timescale of the eye-to-pointer feedback loop imposes a maximum writing rate.

[Figure 18 about here.]

Second, a limit in Dasher might be the time required for the user to search among the presented strings. We performed an experiment to estimate the maximum writing speed of Dasher when this visual search is not required, so only the first factor applies. The required sequence of squares was highlighted in a strongly-contrasting colour (figure 18), and the
‘expert’ user guided Dasher along this sequence as fast as possible. The required sequence of rectangles and their sizes were identical to those displayed when writing the experimental text from *Emma*. The information content of the required writing path is therefore identical to that of the experimental text.

When operating Dasher as shown in figure 18, the same user was able to ‘write’ at a rate of 228 characters per minute. This rate corresponds to an information rate of 6.5 bits per second which is 1.3 times faster than dictation speed when visual search is required. We conclude that visual search is not a major limit in the present implementation.

**Time–Delay Model**

We now present a quantitative model of the information rate limit imposed by the visual–motor feedback loop. A person is trying to track an object at $y(t)$ with a pointer at $u(t)$. The object runs away from the pointer in accordance with:

$$\frac{dy}{dt} = l(y - u),$$  \hspace{1cm} (12)

where $l$ is the exponential growth rate of the deviation, which is proportional to the writing rate. We model the person as a tracker with a delay:

$$u(t) = y(t - \tau).$$ \hspace{1cm} (13)
Substituting for \( u(t) \) in equation 12,

\[
\frac{dy}{dt} - ly(t) + ly(t - \tau) = 0. \tag{14}
\]

This is a type of delay differential equation with well known properties (Bellman & Cooke, 1963). Analysis by Laplace transform shows that \( y(t) \) is stable if and only if \( l\tau < 1 \).

The tracking problem is identical to using Dasher, where \( l \) is the expansion rate of Dasher, assumed constant. If the visible interval at time \( t = 0 \) has size 1, then after time \( t \), the visible interval has size

\[
p = \exp(-lt). \tag{15}
\]

The information content is defined as \( -\log_2(p) \), so the information rate of Dasher is

\[
\frac{-\log_2(p)}{t} = \frac{l}{\log_e(2)} \text{ bits per second.} \tag{16}
\]

Applying the stability condition \( l\tau < 1 \), we find an expression for the maximum information rate, \( M \).

\[
M = \frac{1}{\tau \log_e(2)} \text{ bits per second.} \tag{17}
\]

If we assume an eye–to–hand reaction time of 175 milliseconds (Rosenbaum, 1991), then the maximum information rate is \( M = 8.2 \) bits per second. This theoretical model agrees
reasonably well with our measured information rate of 6.5 bits per second for our most expert user.

In Dasher version 1.0, users could drive the interface arbitrarily fast by holding the mouse near the right hand vertical. From the analysis of the time-delay model, it would seem appropriate to set an upper limit to the rate of expansion so that the dynamics are always stable. In version 1.6, there is an option to set this maximum information rate.

8 APPLICATION TO MOBILE COMPUTING

We used a variable sized window on a 38 cm LCD touch screen to simulate the use of Dasher on a Personal Data Assistant (PDA). This touch screen was not ideal, as it requires an unreasonably large 2 ounces of force.

The size of the output display was varied from 50×50 pixels (1.6cm by 1.6 cm) to 600×600 pixels (20cm by 20cm). Our screen had 30 pixels per cm, similar to the 27 pixels per cm on the Handspring Visor, a Palm–compatible PDA. For each size, an experienced user performed 10 dictation tasks. We plot the mean writing speed in figure 19.

The writing speed decreases rapidly when the screen size is decreased below 150×150 pixels. Increasing the screen size beyond 200×200 pixels does not appear to be advantageous.
9 DISCUSSION

9.1 Comparison to Other Devices

Where previous researchers have reported improvement of performance over time, we can compare Dasher with other devices.

[Figure 20 about here.]

We have extracted data from a number of earlier published evaluations of text entry interfaces and compare the results to Dasher (figure 20).

In a relatively short training time, writing performance with Dasher already exceeded that of many methods. If we extrapolate a power law function fit (a straight line on figure 20), it seems that with further practice, Dasher performance may also equal the fastest method using QWERTY keyboard derivatives.

9.2 Attention Demand

It is clear that Dasher requires sustained visual attention from the user. This requirement is in contrast to conventional keyboards which can be used without visual attention after sufficient training.

9.3 Potential Writing Speed

The performance of Dasher is determined by two factors. The first is how well the language model can compress the text being entered. Shannon (Shannon, 1993) estimated the entropy
of English to be 1 bit per character but PPM typically compresses to 2 bits per character. A perfect language model should increase the speed of Dasher by a factor of 2 which would result in a writing speed similar to that of a QWERTY keyboard.

The second factor is the user interface, which can currently convey 4.8 bits per second to the computer (section 7.1). Earlier we cited an upper bound estimate of 8.2 bits per second for human pointing performance. The following parameters are important:

- the aspect ratio of Dasher and/or the individual rectangles;
- the positioning of the characters on the screen;
- the pointing device;
- the frame-rate of Dasher (averaging 40 frames per second in the experiment);
- the minimum pixel threshold for displaying new rectangles.

There is a tradeoff between the last two factors; the values will be influenced by the performance of the system.

### 9.4 Suitability to Mobile Text Entry

The operation of Dasher is performed without a keyboard, using a single continuous stream of motor input. In section 8 we determined that writing speeds are fairly constant at screen sizes of 150×150 and above. Many PDAs have screens of at least 200×200 so we believe that Dasher could be used on PDA.
Dasher has recently been ported to a PDA. Details on our progress, including a download for Pocket PC’s (ARM, MIPS, SH3 processors), can be found at the Dasher website http://wol.ra.phy.cam.ac.uk/djw30/dasher/.

The combination of stylus and touchscreen is highly suited to Dasher. The user simply points at the letters that they want to select. This mode of operation should prove to be faster than a mouse, where the user is using an indirect method of pointing.

Dasher could be configured so that the viewpoint moves only while the pen is in contact with the screen. The language model could be trained on an existing dictionary, emails or other documents. As the user writes, new words and phrases can be automatically added to the model.

\section{FURTHER WORK}

10.1 Eye-tracking

An eye-tracking device is currently being tested as an alternative input device to Dasher, enabling hands-free text entry. The system could be especially beneficial to users with limited hand movement.

When Dasher is used with an eye-tracker, users simply look at the string they want to write. We hope that this will be easier and more natural to use than eye-tracking interfaces in which the users must consciously direct their gaze toward action targets, for example (Salcucci & Anderson, 2000).
10.2 Modelling

Since the experimental work, a new language model has been implemented. This model also uses the PPM algorithm, but it is word-based, rather than character-based. A dictionary is used to supplement the statistical model. Compression performance on English text is around 10% better than the character model. We observed improvements in typing speed of around 10%.

We are also exploring latent-variable models which should be able to learn new languages more rapidly.

10.3 Japanese

We carried out a small survey of Japanese people to find out the typing speed for Hiragana entered with a QWERTY keyboard. A typical typing rate is about 60 to 120 Hiragana per minute.

We trained a language model on 200k of Japanese text, and estimated the information content of Hiragana to be 2.5 bits per character. The current rate of entering information with Dasher is about 4.8 bits per second, so we might expect a writing speed of 2 Hiragana per second or 120 Hiragana per minute.

We hope to perform an experiment to determine the performance of the Hiragana version of Dasher.
11 CONCLUSION

Dasher is a novel text entry interface that exploits the information redundancy of the English language, and the human capacity to convey high information rates through fine motor control.

The operation of Dasher is simple, and immediately evident to new users. Furthermore Dasher has a rapid learning rate that is comparable to alternative text entry methods.

We think Dasher shows promise as a keyboard–less text entry system both in its absolute writing speed and ease of use.

NOTES

Acknowledgements

Dasher was conceived by David MacKay and Mike Lewicki and would have not been possible without Ousterhout’s Tcl language (Ousterhout, 1994).

We would like to thank the experimental subjects.

Thanks to T. Matsumoto, T. Tokieda and T. Fujiyoshi for helpful discussions.

Support

This work was supported by the Gatsby foundation and by Applied Science Laboratories.

Alan Blackwell’s research is funded by the Engineering and Physical Sciences Research Council, under EPSRC grant GR/M16924 ‘New paradigms for visual interaction’.
Author’s Addresses

David J. Ward, Cavendish Laboratory, Madingley Road, Cambridge, CB3 0HE, United Kingdom. Email: djw30@mrao.cam.ac.uk

Alan F. Blackwell, Computer Laboratory, Cambridge, CB2 3QG, United Kingdom. Email: Alan.Blackwell@cl.cam.ac.uk

David J.C. MacKay, Cavendish Laboratory, Madingley Road, Cambridge, CB3 0HE, United Kingdom. Email: mackay@mrao.cam.ac.uk

REFERENCES


List of Figures

1. (a) Initial configuration. (b),(c),(d) Three successive screen shots from Dasher showing the string ‘the’ being entered. The symbol ‘_’ represents a space. 43
2. Arithmetic Coding .......................................................... 44
3. Dynamics of the interface, assuming the user holds the pointer at a constant screen location. After $S$ frame updates, the point of view has zoomed in so that the point under the mouse is now under the cross–hair. 45
4. Screen update. The left diagram shows a view of the whole interval. The point of view is specified by a visible interval on the real line. The interval has a centre, $C_i$ and a height, $H_i$. The left figure shows three subsequent intervals and on the right, the point of view. 46
5. Pointer coordinates. The height of the display is one screen unit. 47
6. Mean horizontal pointer position, $\bar{x}$. The graph qualitatively shows a typical user’s response to different values of $S$. 48
7. Correcting mistakes. The user has written ‘might’ instead of ‘mean’. In figure (a), the user points at ‘m’ to go back. In figure (b), the user points at ‘mean’. The open arrowhead indicated the start of the motion; the closed arrowhead the end. 49
8. A single gesture can select several characters. Continuations of the string ‘object’ include ‘objection’, ‘objects’, ‘object_and’ and ‘object_of’. 50
9. The user writes ‘laboratory’. (a) Square aspect ratio. (b) Modified display obtained by a non-linear mapping of the horizontal coordinates. 51
10. Figure to demonstrate vertically modified display. (a) Square aspect ratio. (b) Modified display obtained by a non-linear mapping of the vertical coordinates. 52
11. Dasher with capital letters. ................................................. 53
12. Shift-JIS character set for Hiragana. ..................................... 54
13. JDasher - Hiragana character set. ....................................... 55
14. Effect of practice on writing speeds for ten subjects. Each exercise was 5 minutes long. 56
15. Dasher information rate. Each exercise was 5 minutes long. 57
16. Writing Errors ................................................................. 58
17. Logarithmic plot to test the Power Law of Practice. Dasher results are plotted for all ten subjects. 59
18. Modified version of Dasher to measure writing speed without visual search. 60
19. Touch screen writing: effect of screen size. 61
Comparison of Dasher to other devices. Hand printing required the user to print letters on a touchscreen. In QWERTY-tapping, a stylus selects characters from an on-screen QWERTY keyboard. ABC-tapping uses a stylus with letters in alphabetical order. Half-QWERTY uses half a QWERTY keyboard, with the other half accessible with a special key. Bellman is a keyboard designed with a probabilistic character layout strategy. TCK2 is a version of the ternary chorded keyboard. OPTI is a soft-keyboard with an optimized layout.
Figure 1: (a) Initial configuration. (b),(c),(d) Three successive screen shots from Dasher showing the string ‘the’ being entered. The symbol ‘_’ represents a space.
Figure 2: Arithmetic Coding
Figure 3: Dynamics of the interface, assuming the user holds the pointer at a constant screen location. After $S$ frame updates, the point of view has zoomed in so that the point under the mouse is now under the cross-hair.
Figure 4: Screen update. The left diagram shows a view of the whole interval. The point of view is specified by a *visible interval* on the real line. The interval has a centre, $C_i$ and a height, $H_i$. The left figure shows three subsequent intervals and on the right, the point of view.
Figure 5: Pointer coordinates. The height of the display is one screen unit.
Figure 6: Mean horizontal pointer position, $\bar{x}$. The graph qualitatively shows a typical user’s response to different values of $S$. 
Figure 7: Correcting mistakes. The user has written ‘*might* instead of ‘*mean*’. In figure (a), the user points at ‘m’ to go back. In figure (b), the user points at ‘*mean*’. The open arrowhead indicated the start of the motion; the closed arrowhead the end.
Figure 8: A single gesture can select several characters. Continuations of the string ‘object’ include ‘objection’, ‘objects’, ‘object_and’ and ‘object_of’.
Figure 9: The user writes ‘laboratory’. (a) Square aspect ratio. (b) Modified display obtained by a non-linear mapping of the horizontal coordinates.
Figure 10: Figure to demonstrate vertically modified display. (a) Square aspect ratio. (b) Modified display obtained by a non-linear mapping of the vertical coordinates.
Figure 11: Dasher with capital letters.
Figure 12: Shift-JIS character set for Hiragana.

<table>
<thead>
<tr>
<th>0x8290</th>
<th>あいいうえおかがきぎくぐけ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x82A0</td>
<td>ごこさざしすせそそそたたち</td>
</tr>
<tr>
<td>0x82B0</td>
<td>ちっつててとどなるぬのはばば</td>
</tr>
<tr>
<td>0x82C0</td>
<td>ひびぶぶへべべほほほみむめ</td>
</tr>
<tr>
<td>0x82D0</td>
<td>もややゆゆよよよりりれろゎゎわわゎ</td>
</tr>
<tr>
<td>0x82E0</td>
<td>ん</td>
</tr>
<tr>
<td>0x82F0</td>
<td>とん</td>
</tr>
</tbody>
</table>
Figure 13: JDasher - Hiragana character set.
Figure 14: Effect of practice on writing speeds for ten subjects. Each exercise was 5 minutes long.
Figure 15: Dasher information rate. Each exercise was 5 minutes long.
Figure 16: Writing Errors
Figure 17: Logarithmic plot to test the Power Law of Practice. Dasher results are plotted for all ten subjects.
Figure 18: Modified version of Dasher to measure writing speed without visual search
Figure 19: Touch screen writing: effect of screen size.
Figure 20: Comparison of Dasher to other devices. Hand printing required the user to print letters on a touchscreen. In QWERTY-tapping, a stylus selects characters from an on-screen QWERTY keyboard. ABC-tapping uses a stylus with letters in alphabetical order. Half-QWERTY uses half a QWERTY keyboard, with the other half accessible with a special key. Bellman is a keyboard designed with a probabilistic character layout strategy. TCK2 is a version of the ternary chorded keyboard. OPTI is a soft-keyboard with an optimized layout.