Ticker: An Adaptive Single-Switch Text Entry Method for Visually Impaired Users

Emli-Mari Nel, Per Ola Kristensson, and David J. C. MacKay

Abstract—Ticker is a probabilistic stereophonic single-switch text entry method for visually-impaired users with motor disabilities who rely on single-switch scanning systems to communicate. Such scanning systems are sensitive to a variety of noise sources, which are inevitably introduced in practical use of single-switch systems. Ticker uses a novel interaction model based on stereophonic sound coupled with statistical models for robust inference of the user's intended text in the presence of noise. As a consequence of its design, Ticker is resilient to noise and therefore a practical solution for single-switch scanning systems. Ticker's performance is validated using a combination of simulations and empirical user studies.

Index Terms—Single-switch systems, accessibility, augmentative and alternative communication, Bayesian inference

1 Introduction

A *single-switch user* is someone whose primary means of communication relies on a single-switch input device, triggered by an action such as blinking, raising an eye brow, flexing of the wrist, jostling a knee, sniffing, sipping and puffing on a straw, or thinking of an activity such as tennis [1], [2], [3].

Switch events are challenging to capture due to involuntary actions from users, which necessitates a sophisticated noise-tolerant system. When triggering a switch with, e.g., an eye blink, it might not always be clear if it was intentional, especially if the user has involuntary head motions. A sophisticated gesture-detection algorithm is typically necessary. Such a detection algorithm will inevitably have a non-zero error rate. In addition to the inherent difficulty to automatically identify a gesture, several other noise sources can corrupt the user's switch events.

In [4] the importance of modelling noise sources is highlighted and categorized. For example, timing errors versus unintentional clicks, where the activation of a switch is referred to as a *click*. *Switch noise* may cause spurious click detections (false positives), or failures to detect a click (false negatives). *Click-timing noise* causes the observed click time of a switch event to be earlier or later than intended. We will assume that click-timing noise and switch noise operate independently of each other.

Scanning systems are the most prevalent single-switch systems in the literature. With a typical scanning system, the user selects a row and then a column in a grid configuration; see Fig. 1.

• *The authors are with the University of Cambridge, Cambridge CB2 1TN, United Kingdom. E-mail: {en256, pok21, djcm1}@cam.ac.uk.*

Manuscript received 22 May 2016; revised 20 Apr. 2018; accepted 4 June 2018. Date of publication 16 Aug. 2018; date of current version 10 Oct. 2019. (Corresponding author: Emli-Mari Nel.)
Recommended for acceptance by M. Betke.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2018.2865897

In [4] an in-depth background review of techniques that led to *Ticker*, our proposed audio scanning system for single-switch users, are given. The framework in [4] allows one to model a probabilistic sequence of user actions if the probability distributions quantifying the noise sources are known. One can sample from the noise distributions to simulate performance. Even though such noise sources might not be a complete representation of the reality it can help developers to make their software robust against the most common problems in practice, and thereby save hours of field work.

Some drawbacks of standard scanning systems such as Grid2 [5] are highlighted and analyzed via simulation in [4]:

1) By design, a linear increase in a user's average response time requires a linear increase in scanning delay, causing the text entry rate to reduce significantly. The *scanning delay* refers to the time (measured in seconds) it takes to present a selection option to the user. Hence, standard scanning systems, that have to increase the scanning delay as such, are referred to as the *slow-scan method*.
2) Similar to 1), the scanning delay has to be increased linearly with an increase in standard deviation of the click-timing distribution to keep the probability of error constant.
3) Finally, by design, there is no way to control the probability of error due to spurious clicks; they can only be corrected after they occurred.

In [4] a *fast-scan method* is proposed. The method is intended for users who can click precisely (resulting in a narrow click-timing distribution), but have a long average response delay (which is assumed to be known through measurement). The click-timing distribution is then used to infer the intended letter after a whole row/column has been scanned. Since there is a model that can be used to do inference it is not necessary to increase the scanning delay at each cell. The scanning delay is decoupled from the average click-timing delay at all cells except the last in the group,

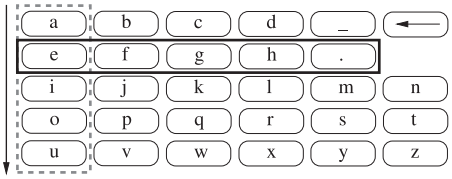

Fig. 1. A typical scanning interface. To select the letter "h", at least two clicks are necessary. In the first phase all rows are scanned. The first click selects the desired row associated with the subset *efgh.* after hearing the corresponding audio cue (e.g., the audio recording of the letter "e"). Thereafter the individual letter keys of the selected row are scanned in sequence. The second click selects the desired letter "h". Grid2 [5] provides software which enables one to select letters using this configuration. In audio mode, the user will typically have to memorize the contents of each subset which is not necessary in visual mode.

which means one can decrease the delay at all cells except the last, thereby increasing the overall text entry rate. The effect of this slight modification to the slow-scan method is validated through simulation.

Note that the fast-scan method works exactly like the slow-scan method, except that the whole row or column is scanned before the system decides which cell to select. If the wrong subset/row is selected after the first click, this has to be undone in exactly the same way than in the slow-scan method.

We wish to apply the latter idea of including the noise source model in the interface design to reduce error corrections and long scanning delays. We work with the same noise sources defined in [4]. The fast-scan method models only the latency as part of the interface design, enabling the system to "scan faster" compared to the standard method (if the user has a large response latency). In Ticker, we also include an explicit model for false- positives and negatives. We also defer the inference decision until the end of the word instead of the end of a letter. We wish to control the probability of error through probabilistic modelling, so that reliable communication can be achieved without a reduction in text-entry rate.

Ticker infers the user's intentions from the timing of a user's clicks during audible presentations of the alphabet. By design, our model can take the whole distribution into account (e.g., also the standard deviation in the case of a Gaussian). Any click-timing distribution that is continuous in time can be accommodated.

The alphabet is spoken twice at a high speed, using a different (but fixed) ordering of the letters at each repetition. The user is tasked to select a letter twice for redundancy purposes. The system has two measurements to try to infer the intended letter. The letter order is shuffled (but fixed) so that the user's selection delay can be accounted for. The two letter sequences are fixed to reduce the users cognitive load.

Even when the user's click times are imprecise (the click-timing distribution is broad), the system can often detect which letter was intended.

Similar to the fast-scan example, mentioned earlier, we decouple the dependence of the scanning delay on the average response delay, except for the last scan, which in our case, happens at the end of the word. This means that we can potentially achieve an even higher text entry rate than the fast-scan method in the same setting where it outperforms the slow-scan method.

A challenging problem in practice is to present the alphabet to the user at a high speed such that it is still audible. Ticker uses stereophonic sound recordings from several different speakers, each pronouncing a subset of the alphabet. The use of different voices at different perceived spatial locations can potentially help the user to process the presentation of letters at a higher speed. An illusion is created that the alphabet is presented at a much slower speed, if the user is able to focus on a particular voice. Section 2.1 describes how this illusion can be achieved through an example of the system in two-channel mode.

Ticker's use of stereophonic sound recordings exploits the *cocktail-party effect*; the ability of humans to filter out a chosen signal from a range of simultaneous auditory stimuli [6], [7]. The use of this technique results in a system that is hoped to be intuitive and easy to learn, and can be used in conjunction with the more commonly used approach of altering the frequencies of the sounds (which might take some time to get used to).

Making use of the cocktail-party effect provides a way to parallelize a serial input. When using five audio streams (five people speaking simultaneously), one can, in theory, make the text entry rate five times faster compared to one person reading the alphabet serially to the user. However, using five audio streams as input to an audio text entry method is unexplored. It can create perceptual difficulties, which can lead to difficulties in interpreting the user input, especially if the design doesn't make it easy for the user to focus on a particularly audio channel [8].

We could not find other real world applications where more than three audio channels are used. Several references (see e.g., [8]) note that it becomes increasingly difficult to switch one's attention to a different voice when more than two voices speak simultaneously. We therefore investigate whether it is possible to make use of more than three audio channels in the context of this application. More specifically, we compare human performance in the same conditions for three, four and five audio channels.

A useful aspect of Ticker is that it is *adaptive*, because all distributions are re-trained after each word selection. If, for example, the user's average click-timing delay drifts systematically, the learning algorithm is designed to translate the click-timing distribution accordingly. This can be useful if, e.g., a user is tired at the end of a writing session, causing him/her to respond with a larger latency than usual.

Our key contributions are:

a) the design of a statistical model that handles several types of noise that inevitably occur in practice as part of the interface design;
b) the design of appropriate simulations to validate robustness to long user response times and false positives as part of the design interface;
c) empirical evidence indicating that some able-bodied users, and one impaired user; could select letters using Ticker in five channel mode;
d) empirical evidence indicating that human performance does not differ significantly between 3, 4, and 5 audio channels;
e) open-source software libraries [9].

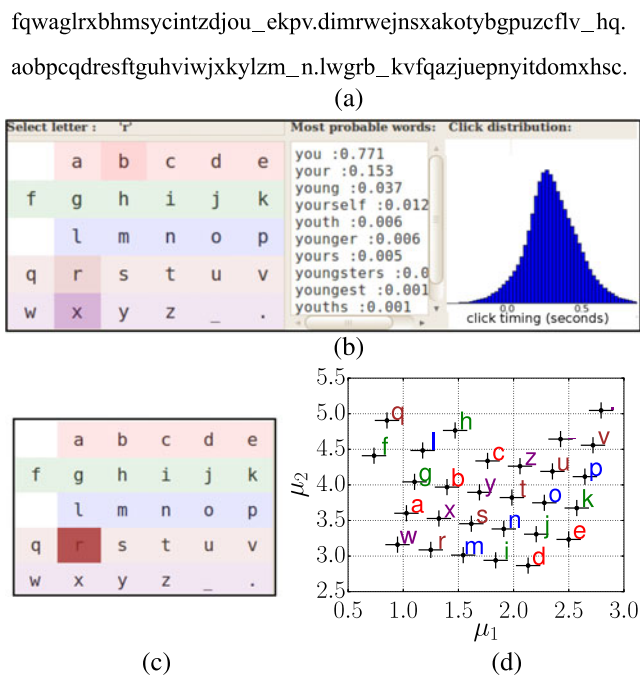

Fig. 2. (a) Two example composite audio sequence that can be presented to the user. The top/bottom sequence was optimized for usage in five/two channel mode, respectively. (b) An illustration of the system's state after receiving one click while trying to write *r* of "*your_*" in five channel mode from the top composite sequence in (a). The first repetition of the alphabet is shown from top to bottom, $\{fqwag, \ldots\}$, where letters with the same color occur at the same sound source. The letters are highlighted according to their probabilities. The user aimed for "r", but clicked slightly late, causing "x" to be the most likely candidate, followed by "b" and "r". (c) The color intensities indicate that more certainty is provided by the second click compared to (b), i.e., it is more obvious that the user aimed for "r". (d) Click times that are possible for the first/second repetition of the alphabet μ_1/μ_2 is shown on the horizontal/vertical axis (measured in seconds). The start- and end times of each symbol's sound file (black lines) are plotted for the first and second repetition of the alphabet, as derived from the top composite sequence in (a). That is, the composite sequence is again $\{fqwag, \ldots\}$. The starting time of "s" is at about 1.5s/3.4s during the first/second repetition of the alphabet. The possible click times are measured from the beginning of audio file of the composite sequence. Letters within the same channel have the same color and correspond to the colors in (b).

2 Ticker

2.1 User Interface

Ticker is a new audio-based single-switch text entry method that uses a noise model to adapt to the user's capabilities. Ticker uses three main strategies to cope with noise:

1) statistical models to represent noise,
2) a predictive language model,
3) an audio-interface that repeats the alphabet to the user R times in a fixed pseudo-random sequence, referred to as the *composite audio sequence*. This paper focusses on $R = 2$.

The user selects a word from a pre-defined dictionary in two steps: 1) Letter candidates are selected sequentially, and 2) the posterior word probabilities are updated after each letter candidate selection. If the posterior probability of any particular word is above a pre-defined threshold then that word is selected.

Fig. 2a presents two example composite audio sequences. In one-channel mode, one voice will read this sequence to the user, starting at the beginning with "f". The sequence consists of two alphabet repetitions. The order of the second repetition is different from the first. The user selects one letter at a time to write "is_": While listening to the composite audio sequence, the user will select the first letter, "i", by activating the binary switch ("clicking") after hearing "i". Since "i" is included twice in the composite sequence, two clicks are expected. If at least one activation occurs the system will proceed to the next letter, updating and comparing the posterior probabilities of the words. If no click is received, the user will have the opportunity to click again. Note that "i" is not explicitly selected. If the click-timing model is not misspecified "i" will become a likely first-letter candidate after the word probabilities are updated. The character selections become explicit once the word is selected. This idea is similar in spirit to the T9 mobile interface.

In extreme circumstances the user might reach the end of his/her word without any word selection made by the program. This can happen, if, e.g., the user mostly clicked only once per letter. In such cases, the user has to resume clicking from the beginning of the word. In practice, it was found that the user rarely has to repeat more than one character before the system becomes certain of the user's intentions. Longer words can typically be inferred before their ends are reached (due to the language model's influence), reducing the average number of clicks to less than 2 clicks per character.

One can potentially increase the information rate by using multiple audio channels to parallelize the composite sequence into groups called *clips*. We only use horizontal directional perceptions. That is, a sound source can be perceived to be located to the left or to the right of a user. To indicate the phase offset of the sound (i.e., the sound source location) we use the normalized notation $\phi_e \in [-1, 1]$, where $e \in \{1, \ldots, E\}$ and E is the number of sound source locations.

In two-channel mode, the user is expected to wear head phones. Two voices will read the alphabet at two normalized stereo locations, -1 (audible in their left ear), and 1 (audible in their right ear). The second composite sequence in Fig. 2a starting with "a" can be read to the user in sequence like before. The voices will alternate from one letter to the next, e.g., the first letter "a" will be read by one voice in the left ear, after which "o" will be read by the second voice in the right ear. The composite sequence is designed so that "a", and "o" will always be read by the same voice at the same audio location. When the user focusses on the voice associated with the target letter (fixed by design), all other voices will be ignored by virtue of the cocktail party effect. In two channel mode this will cause the illusion that the alphabet is presented to the user at half the speed of the one channel mode.

Fig. 2b shows how the first composite sequence in Fig. 2a is designed to enable usage in five channel mode: The user will hear letters from the set $\{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}\}$ when focussing on the voice associated with the color red.

To successfully deploy the use of stereophonic sounds combined with the cocktail-party effect, we utilized results from experimental psychological studies on speech intelligibility [6]. It was found that, if the sound sources are not clearly distinguishable, the brain tends to filter out some parts of the audio sequence completely. The most prominent techniques used to increase speech intelligibility were: Every other voice was recorded from a different gender. We

varied the pitch of the voices in different channels considerably. We have also created a constant "rhythm" within each channel, which helps considerably to stay focussed on a voice and to improve the user's click-timing precision.

If the composite sequence becomes inaudible because the sound files overlap too much in one channel mode, the clips can still be audible when the number of channels is increased and the user is able to focus on a specific voice. The input is therefore parallelized: at composite level many things happen simultaneously, but at a specific clip level fewer sounds overlap; see, for example, Fig. 2d: The whole alphabet is presented twice in approximately 5s, as indicated by μ_2, the possible click times associated with the second alphabet repetition. Each letter is therefore presented to the user in about 96ms. The composite sequence for one channel may be inaudible. That is, if the first composite sequence in Fig. 2a is played to the user at the same speed without making use of stereophonic sounds, it might be difficult to make sense of the sequence, as many of the sound files overlap.

Figs. 2b and 2c provide a visualization of the program's state during an attempt to select the fourth letter *r* in "*your_*". From the shown color intensities a few letters seem to be likely after the first click, with "x" being the most probable.

The posterior probabilities of all words in the box labelled "most probable words" are shown. After three implicit letter selections, the word "*you_*" has higher prior mass compared to "*your_*", because it is used more frequently in English. It is therefore at the top of the word list before processing the fourth character.

Although "r" is the most likely fourth letter after the second click (shown in Fig. 2c), it is not explicitly selected. Instead, the posterior probabilities of all words are updated after processing the two clicks. This update will place "*your_*" at the top of the word list, instead of "*you_*", because "r" has a much higher probability compared to "_". The user will progress to select "_", in which case the posterior probability of "*your_*" should be well over 0.9, making it clear that the user is not aiming for words like "*yours_*" or "*yourself_*". If the posterior probability of "*your_*" is above 0.9, the system will select it, at which point all the letter selections become explicit.

After each word selection, the click-timing distribution is updated. The click-timing distribution is initialized with a Gaussian. The system then trains a non-parametric distribution after a short calibration/training phase, where the user is required to write "yes_" (in which case the eight true click times are known). Fig. 2b shows the click-timing distribution after the calibration/training phase. The distribution is quantified by a histogram, and is clearly not Gaussian any more (compared to the Gaussian that is used to initialize calibration/training). The resulting distribution is unimodal with an asymmetric narrow peak. Since the training is initialized with a Gaussian prior, and the training algorithms applies some smoothing, many bins can be used to represent the distribution. Section 3.5 provides more detail regarding the derivation of the non-parametric distribution.

Some central concepts in this paper revolve around the computation of the distance between a received click time and the click-timing distribution. In time, some letter are said to be "close" to each other. If the click-timing distribution is a very narrow Gaussian, the distance will amount to an euclidean norm in an R dimensional space, e.g., Fig. 2d provides the 2D visualization of Fig. 2a. Using the euclidean norm as an example distance metric: If we scale each axis so that the length of each file is one unit long (all files are assumed to have the same unit file length), "x" and "r" are next to each other (separated by one unit) in the first dimension. In the second (y) dimension they are separated by 6 units, resulting in a total distance of 6.1 units between them. The minimum distance between any two letters is 4.1 units.

For the same aforementioned example, $\{\mathrm{g}, \mathrm{l}, \mathrm{x}, \mathrm{b}\}$ are the four nearest neighbors to "r" during the first repetition of the alphabet, but during the second repetition the nearest neighbors are $\{\mathrm{i}, \mathrm{m}, \mathrm{w}, \mathrm{e}\}$. After receiving the first click time "x" can have a large probability (with the intention of selecting "r"), but because it is far from "r" during the second repetition its probability is likely to drop significantly.

In [4] we have illustrated that the fast- and slow scan methods cope with broad click-timing distributions by making the scanning delay longer. To deal with the click-timing precision as part of the interface it is necessary to include more degrees of freedom. On a letter selection level, we have added another degree of freedom through the second alphabet repetition. This becomes apparent by evaluating the composite sequence in 2D. An explicit evaluation in 2D is necessary because, if the sequence is not carefully considered, the second dimension can be ineffective, e.g., if the first and second repetition is exactly the same.

More degrees of freedom are created by deferring the letter selection decisions to the end of a word, and making use of a language model. Theoretically, this creates even more resilience to the noise sources that are difficult to cope with in standard scanning systems, but does require the user to be able to spell.

We illustrate with a small example how Ticker manages to be more resilient to false positives and click-timing precision compared to standard scanning systems, even if a language model is not used.

The user is expected to click twice for each letter, which is also the case for a standard scanning system. In a standard scanning system, however, the minimum distance between letters is always 1 unit. This means, by design, if the click-timing distribution is a broad Gaussian, Ticker has a smaller probability of error for the same unit file length.

If the user has a long response time, and can click precisely, one does not have to decrease the text entry speed in Ticker to keep the probability of error constant (except for the last letter scan). This means, that in theory, if the probability of error has to be the same for both Ticker and a standard scanning system, and that two clicks per character are required, one should be able to communicate faster when using Ticker (provided the sound file lengths can be made very small).

In the presence of false positives, a standard scanning system will always select the spurious click immediately after it was received. In Ticker the selection will be determined by the probabilities. After the presentation of the composite sequence there will typically be three click times to process, if one spurious click was received. With a good noise model, the probability of the intentional letter should be much higher compared to the unintentional selection associated with a false positive. We therefore wait a bit to

gather evidence, and argue that this might be a more practical solution for an impaired user than to immediately make a spurious selection that has to be fixed.

The composite audio sequence is computed through an optimization procedure, which is discussed in [10].

3 Ticker Noise Models

With each letter selection, the same composite audio sequence is presented to the user in T seconds. This sequence contains R repetitions of each character. The more the user manages to click near their desired letter, the more confident the system can be about the user's intentions. Increasing R will generally decrease the probability of error (tolerate more noise) but also decrease the text entry rate. We mainly focus on $R = 2$ in this paper.

After each composite audio sequence (T seconds), M received click times $\mathbf{t} = \{t_1, \ldots, t_{\mathrm{M}}\}$ are analyzed, where $t_m \in [0, T]$ and $m \in \{1, \ldots, M\}$. A complicated part of this research was to find an adequate model for $P(\mathbf{t}, M \,|\, \ell, \boldsymbol{\theta})$, where ℓ is the intentional letter, and the model parameters are represented by $\boldsymbol{\theta}$. The model must distinguish between intentional and spurious clicks, and has to determine if the intentional click times were further corrupted in some way (e.g., occurring later than intended).

The composite sequence is quantified by the set of integers $\{(a, r) : a \in \{1, \ldots, A\}, r \in \{1, \ldots, R\}\}$. Each index (a, r) can be mapped to a symbol/letter $\mathrm{ch}(a, r) = \ell_r = \ell$, where ℓ can be one of A unique letters in the alphabet, and ℓ_r refers to the rth repetition of ℓ.

Inference in Ticker revolves around the following inverse probability (using Bayes' theorem)

$$P(\ell \,|\, \mathbf{t}, M, \boldsymbol{\theta}) = \frac{\pi_\ell P(\mathbf{t}, M \,|\, \ell, \boldsymbol{\theta})}{\sum_{\forall \ell'} \pi_{\ell'} P(\mathbf{t}, M \,|\, \ell', \boldsymbol{\theta})}, \tag{1}$$

where $\boldsymbol{\theta}$ is the set of parameters, $|\,\mathbf{t}\,| = M$ is the number of clicks received, and $\pi_\ell = P(\ell)$ is a prior distribution over letter ℓ. Likewise, $\pi_{\ell'}$ is the prior over letter ℓ'. We will use C to denote the number of true clicks, and $N = M - C$ to denote the number of spurious clicks.

The derivations of this section concentrate on developing a model for $P(\mathbf{t}, M \,|\, \ell, \boldsymbol{\theta}) = P(\mathbf{t} \,|\, \ell, \boldsymbol{\theta}, M) P(M \,|\, \boldsymbol{\theta})$, initially by assuming a Uniform prior, π_ℓ, for each letter ℓ, and assuming $\boldsymbol{\theta}$ is known.

3.1 Simplified Noise Model

A useful property of a generative model is that one can draw samples from it. One can analyze a sequence of samples to see if they match reality, and to see how the system behaves when they are used as input to the system. The generative model used in Ticker, allows one to sample sequences of click times. Each click time can be intentional or spurious.

We make use of a sampling procedure in later sections to simulate Ticker's performance and compare that to a simulation of a standard scanning system. During the comparison, the noise sources (probability distributions) for both systems are the same, and we would like to evaluate the effect on both systems if the parameters of the distributions are varied.

In this section we start with a simple generative noise model, and derive our final model from it. To generate samples, consider that the following process generates a single click and its associated click time:

1) Flip a coin with bias g to decide if a click should be spurious (with probability g) or true (with probability $1 - g$).
2) If the click from step 1 was classified as spurious, sample its click time $t_m \sim \mathcal{U}(0, T)$. Otherwise flip a coin with bias f to decide if the true click should be falsely rejected (with probability f) or not (with probability $1 - f$).
3) If the non-spurious click from step 2 was accepted, generate its click time by first sampling a discrete index r uniformly from $1, \ldots, R$, where R is the number of times the alphabet was repeated. The user is assumed to have some average response time latency Δ with standard deviation σ. We assume the same Δ and σ for all letters. We define $\mu_{\ell_r} \in [0, T]$ as the beginning of the sound file associated with ℓ_r (the rth repetition of ℓ). Subsequently sample $t_m \sim \mathcal{N}(\Delta + \mu_{\ell_r}, \sigma)$.

By repeating the above steps M times, we obtain a generative model for a simple click-timing distribution. For example, it can accommodate situations where all clicks are spurious (false positives), in which case $t_m \sim \mathcal{U}(0, T), \forall m$ or where all clicks are true clicks $t_m \sim \mathcal{N}(\Delta + \mu_{\ell_r}, \sigma) \forall m, r$. Of course, when receiving click times from a real hardware switch, the corresponding hypothesis that generated each click time is unknown. We therefore marginalize (sum) over all possible hypotheses to obtain a distribution that directly models the times of received clicks. The resulting model after marginalization is

$$P(\mathbf{t}, M \,|\, \boldsymbol{\theta}, \ell) = M! \prod_{m=1}^{M} \Big[(1 - f)(1 - g) p_{m\ell} + \frac{g}{T} \Big], \tag{2}$$

where

$$\boldsymbol{\theta} = \{\{\boldsymbol{\theta}_{\ell_1}, \ldots, \boldsymbol{\theta}_{\ell_R}\}, g, f, T, R\}, \tag{3}$$

$$\boldsymbol{\theta}_{\ell_r} = \{\Delta + \mu_{\ell_r}, \sigma\}, \tag{4}$$

$$p_{m\ell} = \frac{1}{R} \sum_{r=1}^{R} \mathcal{N}(t_m \,|\, \boldsymbol{\theta}_{\ell_r}). \tag{5}$$

The normalization factor $M!$ results from the constraint that time always increases, so that $t_1 < \ldots < t_M$.

If $M = 1$, $R = 2$, $g = 0$, and $f = 0.5$, Equation (2) will be a simple mixture of two Gaussians. The latter mixture model is shown in Fig. 3a, where $\ell = \mathrm{b}$, and $\Delta = 0$, so that each Gaussian is centered at the beginning of the sound file associated with the letter "b".

If $M = 2$, $R = 2$, $g = 0$, and $f = 0.5$, Equation (2) will be a sum of four Gaussian products, each representing a hypothesis of how the two click times could have been generated

$$\begin{aligned} h_1 &= \mathcal{N}(t_1 \,|\, \boldsymbol{\theta}_{\ell_1})\, \mathcal{N}(t_2 \,|\, \boldsymbol{\theta}_{\ell_2}), \\ h_2 &= \mathcal{N}(t_1 \,|\, \boldsymbol{\theta}_{\ell_2})\, \mathcal{N}(t_2 \,|\, \boldsymbol{\theta}_{\ell_1}), \\ h_3 &= \mathcal{N}(t_1 \,|\, \boldsymbol{\theta}_{\ell_1})\, \mathcal{N}(t_2 \,|\, \boldsymbol{\theta}_{\ell_1}), \\ h_4 &= \mathcal{N}(t_1 \,|\, \boldsymbol{\theta}_{\ell_2})\, \mathcal{N}(t_2 \,|\, \boldsymbol{\theta}_{\ell_2}). \end{aligned} \tag{6}$$

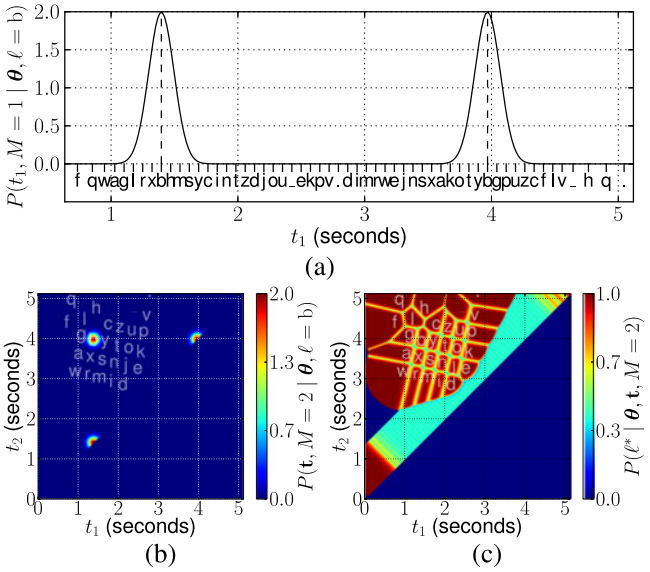

Fig. 3. A depiction of $P(\mathbf{t}, M \mid \boldsymbol{\theta}, \ell = \mathrm{b})$ from Equation (2), where the composite audio sequence corresponds to Fig. 2a, $\sigma = 0.2$ s, $T \approx 5$ s, $\Delta = 0$ s, $f = 0.5$, and $g = 0$. (a) $M = 1$, $\ell = \mathrm{b}$, where $\mu_{\ell_r} + \Delta$ for each letter is indicated with a labelled line. (b) $M = 2$ and $\ell = \mathrm{b}$. (c) $P(\ell^* \mid \boldsymbol{\theta}, t_1, t_2, M)$ is computed from Equations (7) and (1); see the text for detail.

The mixture model $P(\mathbf{t}, M \mid \boldsymbol{\theta}, \ell) = \frac{1}{2}\sum_{k=1}^{4} h_k$ is illustrated in Fig. 3b. Note that the click-time probability is zero where $t_2 < t_1$. There is a full sphere associated with h_1. However, probability mass is also assigned to the other hypotheses, leading to the half spheres on the diagonal of the figure. We specifically refer to h_3 and h_4 as the *same-letter hypotheses*, as they imply that the same Gaussian can be responsible for both click times.

Note from Figs. 3a and 3b that the alphabet is presented to the user rapidly (twice in 5 seconds). When looking closely at Fig. 3a, the two Gaussian bumps associated with "b" imply that two letter groups that can easily be confused are $lrxbhms$ and $otybgpu$. If $\sigma > 0$, it may be challenging to infer the user's intentions when $M = 1$ without the help of a language model (even if $\Delta = 0$). However, none of the letters in the latter letter groups co-occur except for "b", which means that the second click should greatly improve confidence for the user's intentions.

Fig. 3c plots the maximum posterior letter probabilities $P(\ell^* \mid \boldsymbol{\theta}, t_1, t_2, M)$ for the shown click-timing pairs $\{t_1, t_2\}$, where Equation (1) is computed for each pair $\{t_1, t_2\}$ with $\pi_\ell = 1/A$. The letter with the highest posterior probability is then selected for that pair, i.e.,

$$\ell^* = \arg\max_{\ell}(\{P(\ell \mid \boldsymbol{\theta}, t_1, t_2, M), \forall \ell\}). \tag{7}$$

Hence, for each pair there is a corresponding "best letter" ℓ^*, which automatically leads to a Voronoi-type of diagram [11] in the top-left part of the figure. Each Voronoi cell is labelled with its corresponding ℓ^*, and the decision boundaries are shown in light green. If e.g., the received click-time pair $\{t_1, t_2\}$ falls inside cell labelled "f", the probability of "f" will be almost 1.0 except on the boundaries where the probabilities are closer to 0.5.

The artifact on the diagonal is a result of the same-letter hypotheses. This can e.g., cause confusion for click-times close to the letter "d", where the maximum posterior probability can be as high as 0.7 that the user selected another letter that is far from its nearest neighbors (shown in Fig. 2d). Due to the same-letter hypotheses, the current model is only plausible if the false-positive rate is extremely low and the click-timing distribution associated with each ℓ_r is a good approximation for the user's click-timing accuracy.

A second problem with the model in Equation (2) is the hypothesis that all the received clicks may be considered as intentional even if $M > R$.

A third problem is that the false-positive rate is independent of T. In a more plausible model, one would expect the number of false positives to increase as T increases. The remaining part of this section concentrates on addressing the three problems above.

3.2 Poisson Process: Spurious Clicks

A homogeneous Poisson process [12] can be used to model noise in a way that takes into account a *rate* λ of false-positives per unit time. One way to construct a Poisson process is to discretize time into a set of bins (of equal width), so that no more than one event can take place in any particular bin. In our problem the probability that two false positives are detected almost simultaneously is negligibly small, allowing the construction of a Poisson process to model spurious click-times. The result is that the number of spurious clicks N in a finite time interval T always has a Poisson distribution

$$P(N \mid T, \lambda) = \frac{(\lambda T)^N e^{-\lambda T}}{N!}, \tag{8}$$

where λ is the average number of false positives per unit time. The longer we wait, the smaller the probability that $N = 0$. The Poisson process addresses the third problem mentioned at the end of Section 3.1.

Furthermore,

$$P(\mathbf{t} \mid N, \lambda, T) = N! \cdot \left(\frac{1}{T}\right)^N \cdot \delta(\,|\,\mathbf{t}\,| - N), \tag{9}$$

where the normalizing constant $N!$ results from the time constraint $t_1 < t_2 < \ldots < t_N$. It follows that

$$P(\mathbf{t}, N \mid \lambda, T) = P(\mathbf{t} \mid N, \lambda, T) P(N \mid \lambda, T) = \lambda^N e^{-\lambda T}, \tag{10}$$

where $N = |\,\mathbf{t}\,|$.

When using the Poisson process to model the false positives, the collection of parameters from Equation (3) is modified by replacing g with λ

$$\boldsymbol{\theta} = \{\{\boldsymbol{\theta}_{\ell_1}, \ldots, \boldsymbol{\theta}_{\ell_R}\}, f, \lambda, T, R\}. \tag{11}$$

where Equation (4) defines $\boldsymbol{\theta}_{\ell_r}$.

3.3 Deriving the Click-Timing Distribution

The second problem that was mentioned at the end of Section 3.1, the hypothesis that all M clicks can be intentional, can be addressed by assuming that the user never clicks more than necessary to make a selection. This assumption can be expressed with a Binomial distribution

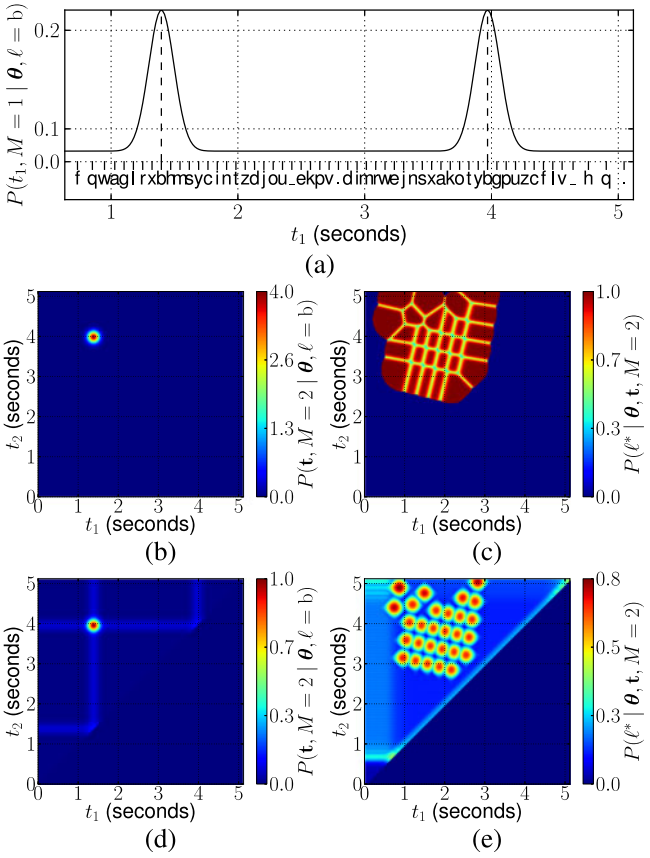

Fig. 4. The differences to Fig. 3 are pointed out. First, Equation (2) was replaced with Equation (13) to compute $P(\mathbf{t}, M \mid \boldsymbol{\theta}, \ell = \mathrm{b})$, with $\boldsymbol{\theta}$ as defined by Equation (11). (a) The effect of switch noise is shown, whereas Fig. 3a was drawn without switch noise: $f = 0.5$ and $\lambda = 0.3$/s. (b)–(c): Compared to Fig. 3, there is only one 2D Gaussian, and there are no artifacts on the diagonal in (c). (d)–(e): Adding switch noise to (b)–(c): $f = 0.5$ and $\lambda = 0.3$/s.

$$P(C \mid R, f) = \frac{R!}{C!\,(R-C)!} f^{R-C}(1-f)^{C}, \tag{12}$$

where C is the number of true clicks intended by the user, R is the number of times the alphabet was repeated to the user, and f is the probability of falsely rejecting a click. Equation (12) assigns zero probability to the event that $C > R$. The probability that a spurious and a true click happen simultaneously is assumed to be negligibly small, so that $P(M \mid N, C) = \delta(M - (N + C))$.

It is assumed that a true click and a set of spurious clicks are generated independently from each other. An auxiliary binary vector $\mathbf{n}$ of length M is introduced to label each click time t_m as either a false positive ($n_m = 1$) or a true click ($n_m = 0$). There are $\frac{M!}{C!\,N!}$ possible assignments of these labels—we assume a Uniform distribution over them.

False negatives can lead to uncertainty regarding which letter repetition was responsible for a true click. One can construct another auxiliary binary vector $\mathbf{c}$ of length R for each combination of C and $\mathbf{n}$, to associate each true click in $\mathbf{n}$ with a repetition r of a letter. It follows that $c_r = 1$ for the hypothesis that t_m was generated with the intention of selecting the rth repetition of ℓ, which can only happen if $n_m = 0$. There are $\frac{R!}{(R-C)!\,C!}$ possible cases to consider; again, we assume a Uniform distribution over these.

To address the same-letter hypotheses problem mentioned in Section 3.1, all click-times of true clicks are henceforth assumed to have been generated by a unique Gaussian distribution. For the two click-time example in Section 3.1, this will have the implication that only h_1 and h_2 will be evaluated. In fact, the latter of the two hypotheses is also unnecessary, as it suggests that a click-time from the Gaussian associated with ℓ_2 can be observed before the one associated with ℓ_1. We therefore also omit this hypothesis. For the general case, a true click time from a Gaussian associated with ℓ_r cannot be observed before the ones from $\ell_1, \ldots, \ell_{r-1}$. The overlap between any pair of distributions associated with the same letter is assumed to be insignificantly small. Note that this is clearly the case in Fig. 3a, where the two Gaussians associated with "b" do not overlap significantly, even for a relatively large σ. This implies that a click from the second Gaussian is exceedingly unlikely to occur before a click from the first Gaussian.

All assumptions are combined to compute

$$P(\mathbf{t}, M \mid \boldsymbol{\theta}, \ell) = e^{-\lambda T} \sum_{C=0}^{C'} \lambda^N \cdot f^{R-C} \cdot (1-f)^C \cdot p_{C\ell}, \tag{13}$$

where $C' = \min(R, M)$, $N = M - C$, $\boldsymbol{\theta}$ is given by Equation (11),

$$p_{C\ell} = \sum_{\mathbf{c},\mathbf{n}} p_{\mathbf{z}\mathbf{t}\ell}, \tag{14}$$

$$p_{\mathbf{z}\mathbf{t}\ell} = \prod_{m=1}^{M} \prod_{r=1}^{R} \mathcal{N}(t_m \mid \boldsymbol{\theta}_{\ell r})^{g_{\mathbf{z}mr}}; \tag{15}$$

$$g_{\mathbf{z}mr} = \delta(1 - n_m) \cdot \delta(1 - c_r) \cdot \delta\left(\sum_{m'=1}^{m} \mathbf{n}_{m'} - \sum_{r'=1}^{r} \mathbf{c}_{r'}] \right); \tag{16}$$

where $\mathbf{z} = \{\mathbf{n}, \mathbf{c}, C, N\}$, $t_1 < \ldots < t_M$ and $\mu_{\ell_1} < \ldots < \mu_{\ell_M}$.

Fig. 4 illustrates the changes made to Fig. 3 by replacing Equation (2) with Equation (13).

Fig. 4a illustrates that the Gaussian bumps are less peaked due to the non-zero "floor rate" to compensate for false positives, where for $M = 1$, a false positive can only result if the false negative probability is non-zero. Figs. 4d 4e illustrates the effect of switch noise for $M = 2$. The light blue regions in Fig. 4d correspond to the hypothesis that one click is a true positive and the other a false positive. The switch-noise parameters were chosen quite large to make the effect of switch noise obvious for illustration purposes: $\lambda = 0.3$/s and $f = 0.5$ will result in an average of 1–2 false positives every time the alphabet is presented to the user (in 5 seconds), and true clicks will be ignored 50 percent of the time.

3.4 Language Modelling

Our language model is similar to Nomon [13]. We also use a slightly modified version of the British National Corpus word-frequency list [14], and keep track of the posterior probability of all words. If the posterior probability of any word is above a certain threshold, that word is selected. After each explicit/implicit letter selection in Nomon/Ticker, the posterior probability of the word is updated.

We have to keep track of which letter the user intends to select at any specific time for each word hypothesis in order to compute $P(\mathbf{t}, M \mid \boldsymbol{\theta}, \ell)$. This is slightly more complicated than in Nomon, where one letter is selected at a time. If the user tries to write "is_" using Ticker, and the system has not selected the word when the user has implicitly selected the space, it is assumed that the user will start over with the word.

The equations for the posterior word probabilities, and $P(\mathbf{t}, M \mid \boldsymbol{\theta}, \ell)$ are derived in [10]. Note that the current version of Ticker does not make provision for words that are not defined in the dictionary. If the user wishes to add a word, an assistant will have to include it manually in the "text" file that contains all the words, which is then uploaded by Ticker.

3.5 Training the Click-Timing Model

The parametric Gaussian distribution assumption to model the user's click-timing precision can be restrictive, especially when learning how to use the system. A novice user might click a bit early during the first reading of the clip (in anticipation to the well-known alphabet sequence), and click slightly later during the second (less familiar) reading. This discrepancy can cause a mismatch between the assumed unimodal click-timing distribution and the actual bimodal click-timing distribution. It is possible that the two modes do not overlap, causing one click time to always be classified as a false positive, which will, in turn slow down the entry rate significantly.

To allow for asymmetric and multi-modal click-timing distributions, a similar approach to Nomon [13] is followed: the Gaussian distribution that has been used to represent the user's click-timing distribution is converted into a non-parametric distribution. The main difference between our approach and that of Nomon is that we take more noise sources into consideration, which introduces more latent variables to account for. To deal with these latent variables during training, we make use of the Expectation-Maximization (E-M) algorithm [15]. The result is a histogram as shown in Fig. 2b. For this example, the learned distribution is asymmetric with a narrow peak, but depending on the user's response time, the distribution can assume any shape.

Similar to Nomon [13], the click-timing distribution is retrained after each word selection, which allows it to dynamically adapt to systematic drift. The training algorithm and all equations can be found in [10].

4 Simulation Results

We compare the first and second-order statistics of the text entry rate ($\#scans$), the total number of clicks ($\#clicks$), and the number of character errors ($\#errors$) between a simulation of a standard scanning system [4] and Ticker. The standard scanning system we compare against is summarized in Fig. 1: The configuration/layout is the same, and the scanning system is assumed to be used in audio mode. In [4] we have developed a Markov chain model for the standard scanning system described in Fig. 1. The Markov chain model allows one to simulate a large universe of user actions when using a standard scanning system, and incorporates all the noise sources defined in this paper (the click-timing model and false negatives/positives). This allows us to compare both systems in the same noise conditions. Our implementation of the standard scanning system that was used in all user trials, as well as the simulation model used to compare against Ticker are available online [9].

All statistics are derived on a per word basis. The number of clicks to write a word can be normalized with the length of the word to measure the number of click per character (cpc). Numerical performance measurements are made by processing one word $\mathbf{w}_k^{\mathrm{x}}$ at a time from a list of words that constitute a phrase set of K words, $W^{\mathrm{x}} = \{\mathbf{w}_1^{\mathrm{x}}, \ldots, \mathbf{w}_K^{\mathrm{x}}\}$.

It is assumed that the user immediately corrects any errors, and does not proceed to the next letter if an error is not corrected. For each ground-truth word $\mathbf{w}_k^{\mathrm{x}}$, the simulation ends with a corresponding output $\mathbf{w}_k^{\mathrm{y}}$, encapsulating three possible scenarios:

1) the selected word is correct, $\mathbf{w}_k^{\mathrm{y}} = \mathbf{w}_k^{\mathrm{x}}$;
2) an erroneous word was selected, $\mathbf{w}_k^{\mathrm{y}} \neq \mathbf{w}_k^{\mathrm{x}}$, where $\mathbf{w}_k^{\mathrm{y}} \neq \emptyset$;
3) system failure, $\mathbf{w}_k^{\mathrm{y}} = \emptyset$. This happens if the system can not cope with the noise, making it inaccessible.

Ticker results are superimposed on the standard scanning system results from [4] to allow for a direct comparison via simulation.

4.1 Simulating Ticker

Ticker is simulated in 1- and 5-channel mode. For each letter in $\mathbf{w}_k^{\mathrm{x}}$, the number of false positives, the number of true clicks and the click times of all the synthetic clicks are generated from their corresponding distributions. These samples are then used as input to Ticker, pretending that they come from a real user. A measure of Ticker's performance is then obtained through a numerical approximation after repeating the above procedure 1,000 times, and computing expectations from the measurements.

For each $\mathbf{w}_k^{\mathrm{y}}$, $\#clicks$ and $\#scans$ are immediately available (as one sample of a numerical approximation), and $\#errors$ is computed as the minimum edit distance between $\mathbf{w}_k^{\mathrm{x}}$ and $\mathbf{w}_k^{\mathrm{y}}$. Similar to the scanning system simulation, a time-out error occurs if the simulation runs for longer than κM audio sequences and $\mathbf{w}_k^{\mathrm{y}} = \emptyset$, where $M = |\mathbf{w}_k^{\mathrm{x}}|$. In all simulations $\kappa = 5$. Unlike scanning systems, no output characters can exist when the system fails. Hence, $\#errors = M$ and $wpm = 0$ in such a case.

For the Ticker simulation we henceforth represent the scanning delay with T_{S}^*. The scanning delay of the scanning system is represented by T_{S}. In both systems T_{S} and T_{S}^* are varied to change the speed at which the alphabet is presented to the user.

The recorded audio file of each letter is 210 ms. The scanning delay is increased by adding a waiting time after a letter is played. The scanning delay is decreased when a new sound file is played to the user before the previous one was finished, i.e., if the scanning delay is less than 210 ms.

All relevant simulation parameters for this paper are summarized as

$$\boldsymbol{\theta}_{\mathrm{S}} = \{\Delta, \sigma, f, \lambda, T_{\mathrm{S}}, T_{\mathrm{S}}^*\}, \tag{17}$$

where Δ, σ is the average and standard deviation of the user's response time, representing the parameters of the

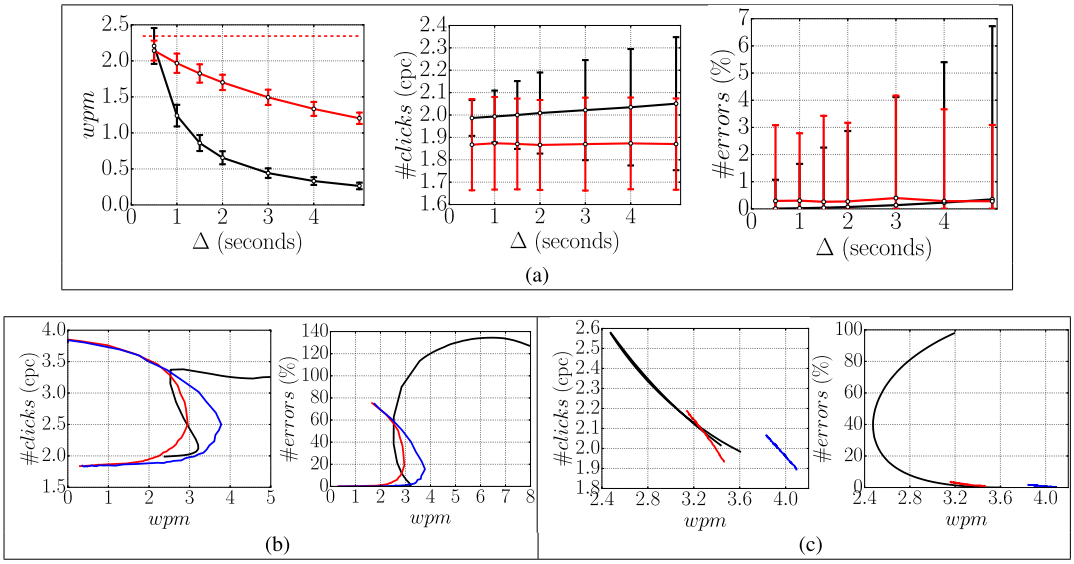

Fig. 5. (a) The click-time delay (Δ) is varied for $\sigma = 50$ ms, $f = 0.05$, $\lambda = 0.001$/s, $T_S = 0.5$ s and $T_S^* = 70$ ms. Ticker (5-channel mode) results are shown in red. The red solid lines indicate the average results (with error bars at one standard deviation). The dashed red line indicates the expected text entry rate without the unnecessary waiting time of $\Delta + 3\sigma$ s at the end of each composite audio sequence. Ticker results are superimposed on the results from [4] (black), where for the scanning system $T_S = \max(0.5, \Delta + 3.0\sigma)$ seconds. (b) The effect of varying the scanning delays T_S and T_S^* are investigated for $\Delta = 0$ s, $\sigma = 100$ ms, $f = 0.05$, $\lambda = 0.001$/s. Average entities for Ticker in 5-channel mode (red) and Ticker in 1-channel mode (blue) are superimposed on the results for the scanning system (black) from [4]. (c) The effect of varying λ is investigated for $\Delta = 400$ ms, $\sigma = 50$ ms, $f = 0.05$, $T_S = 300$ ms, $T_S^* = 42$ ms. Average entities for Ticker in 5-channel mode (red) and Ticker in 1-channel mode (blue) are superimposed on the results for a standard scanning system (black) from [4].

Gaussian click-timing distribution, f is the probability of a false negative, λ is the false positive rate, and T_S, T_S^* is the scanning delay of the standard scanning system and Ticker, respectively. Similar noise conditions therefore exist for both Ticker and the scanning system, and one can easily sample from the Gaussian click-timing distribution.

A Gaussian click-timing distribution is assumed throughout the simulations. We didn't assess the additional performance improvement offered by the non-parametric click-timing distribution mentioned in Section 3.5.

4.2 Simulation Parameters

During all simulations, results are computed for the pangram $W^x =$ "the quick brown fox jumps over the lazy dog." The parameter ranges $\Delta \in [0.25, 3]$ s, and $\sigma \in [50, 200]$ ms were tested. Following the Grid2 manual [5] we assume the boundary $\Delta = 3$ s, $\sigma = 200$ ms is associated with a slow speed setting. The other extreme boundary $\Delta = 0.25$ s, $\sigma = 50$ ms, represents an exceptional able-bodied user (see e.g., [13]).

In theory one should obtain similar performance with the scanning system and Ticker if $T_S^* = 0.14T_S$. For example, if $T_S = 300$ ms and $T_S^* = 42$ ms a text entry rate of 4.7 wpm should, in theory be possible.

One can, in theory, obtain a factor 4-5 speedup by using five stereophonic sound channels instead of one. One could, also potentially achieve much higher entry rates by making use of shorter sounds. For example, a clear "click" sound is only 10 ms instead of the recorded 210 ms for a fast letter pronunciation.

4.3 Simulation Results

The first simulation compared robustness to variations in latency Δ between Ticker and the scanning system. Results are shown in Fig. 5a.

A waiting time of $\Delta + 3\sigma$ s at the end of each composite audio sequence is included in the simulation to reflect the current implementation of Ticker. This delay is there because of software implementation issues that can easily be amended in a future release. The shown text entry rate for Ticker is therefore slightly lower than what it should be (red dashed line), as this delay should be included at the end of a word.

Both systems have the same starting point so that one can evaluate the effect of increasing the scanning delay of both systems without an offset. In [4] it is shown that the scanning delay T_S has to be at least as long as Δ for a standard scanning system, otherwise the user will always select the wrong cell. The scanning delay is therefore linearly increased with Δ, while the other noise sources are fixed. The scanning system's text entry rate decreases at a much faster rate compared to Ticker because the delay happens at every scan, and shows that Ticker is more resilient to an increase in Δ compared to a standard scanning system.

The second simulation tested the influence of varying T_S^*/σ. Fig. 5b indicates that the behavior of Ticker and standard scanning systems differ in failure mode. That is, when the scanning delay becomes small relative to σ. In Ticker, time-out errors occur, where many clicks are used to generate no output word (with an error of 100 percent and a text entry rate of zero). In scanning systems, many erroneous characters lead to the system failure, as the text entry rate and error rate are both high. This means that in the scanning system the noise causes erroneous outputs at a faster rate than the user can correct it. Note that $\#error > 100\%$ if an erroneous word is selected which is longer than the word the user is supposed to select.

If a reasonable accuracy ($\#error < 5\%$) and click rate ($\#clicks \leq 2$) are required, all three systems have similar performance (2.5-3 wpm), with Ticker in 1-channel mode slightly outperforming the rest.

Scaling the scanning delay in any of the systems will not have an effect on the shape of the curve as the curve is varied as a function of T_S^*/σ, making the performance measurement scale invariant.

Ticker in 5-channel mode performs slightly worse than the 1-channel version. The latter robustness to noise of the 1-channel version can be attributed to the nearest neighbor of each letter (in the composite audio sequence) which is further away from it, on average, compared to the 5-channel version. As mentioned above the waiting time at the end of the composite sequence should be deferred to the end of the word (a practical implementation issue), which should increase the text entry rate of Ticker.

A useful insight from this experiment is that, if the user clicks imprecisely (large σ) neither the five-channel version of Ticker, nor the standard scanning system might be viable text entry methods. A possibility is Ticker in one channel mode, and in severe cases, the alphabet has to be repeated more than twice.

The third simulation tests the effect on both systems if spurious clicks are randomly injected into the system. Such false positives are not expected to originate from the click-timing distribution but from an unreliable switch recording device. In this experiment it is assumed that the corresponding distribution is a Poisson process defined in Section 3.2. The Poisson Process has the same false positive rate for both systems when they are compared. Results are shown in Fig. 5c.

Similar initial speeds ($T_S = 300$ ms, $T_S^* = 42$ ms, $\lambda = 0$) for both systems have been chosen. Ticker in 5-channel mode and the standard scanning start with similar text entry speeds ($\lambda = 0$), so that effect of $\lambda > 0$ can be seen more clearly for comparison. For the same reasons as explained for the second simulation, Ticker in 1-channel mode theoretically outperforms Ticker in 5-channel mode. The Ticker performance measurements do not significantly change for any of the tested values of λ. It is therefore clear from the third simulation that Ticker is, in general, significantly more robust to false positives than the standard scanning system.

5 Design Validation

First, we validate the basic design assumptions in a stereophonic-sound controlled experiment with able-bodied participants. Then we validate that the design works in its intended context by evaluating Ticker with a non-speaking individual with motor disabilities who is unable to communicate without human assistance using a standard hierarchical scanning system. Third, we validate the modelling assumptions by training a single able-bodied user to use both Ticker and a standard scanning system at expert performance level with closed eyes and little audio guidance. This enables us to validate the predictions of the modelling reported in the previous section by comparing predictions against an empirical expert human performance envelope.

5.1 Stereophonic-Sound Controlled Experiment

In theory, five or more audio channels have the most potential for high text-entry speeds, since multiple inputs are effectively presented to the user in parallel. However, we could not find any verification that so many audio channels is a possibility in the context of our application. As mentioned in Section 1, going beyond two audio channels where the cocktail party effect applies can make it exceedingly hard for a user to change their focus from one voice to another [6], [7], [8]. These problems are enhanced when increasing the number of audio channels. Some design recommendations from [6] were integrated into Ticker to help the user switch their focus to a particular voice; see Section 2.1. The most important decision is probably that each target letter is always associated with the same voice. We aim to test the efficacy of the design choices associated with changing focus in this section.

Since the literature points out that human performance typically decreases with an increase in the number of channels (as one can naturally expect), we believe that our design is validated if there is no significant difference in human performance between 3, 4 and 5 channels in the same conditions. If, e.g., the alphabet is presented to the user at the same speed but the number of channels are increased from 4 to 5, we isolate the effect of increasing the number of channels by measuring and comparing the human performance.

The main focus of the user trials in this section is to compare human performance between channels. However, we also discuss absolute performance in 5-channel mode. We specifically compare the performance of able-bodied users to the performance of a non-speaking individual with motor disabilities in similar conditions.

The user trials were carried out as a controlled experiment with a within-subject design with two independent variables: 1) the speed at which the alphabet is presented to the user (with three levels: slow, medium and fast) and 2) number of audio channels (with three levels: 3, 4 and 5); and three dependent variables: entry rate, error rate and number of clicks per character.

The speed level defines how much successive sounds in the composite sequence overlap. In "slow" mode the composite sequence is presented to the user in a longer time compared to the other modes causing the successive sounds in the composite sequence to overlap less. More specifically, in fast mode the alphabet is presented twice to the user in 9s. If word completions are ignored this means that the user can not write faster than 1.33 wpm. On average, a word is five characters long, and word completions are unlikely to occur (many words have this length). One can therefore expect the average text-entry speed rate be about 1.33 wpm if the composite sequence is repeated only once per character, and the user clicks twice on target per character. Similarly, in "medium"/"slow" it takes 11.2 s/12.5 s to play the alphabet twice, which can result, at most, in text-entry rates of 1.07 wpm/0.96 wpm in the absence of word completions. We call the aforementioned expected text-entry rates the *baseline text-entry rates*.

The *baseline number of clicks* is 2 cpc, which can be achieved without word predictions, and if the user clicks twice on target for each character. The *baseline error rate* is 0%.

It is important to note that we derive the text-entry rate only from the length of the composite sequence, and the number of times it has been presented to the user. We therefore exclude all other system delays, and other audio cues that can be used to assist the user. This makes it easier to directly compare to simulations, and other methods such as Grid2.

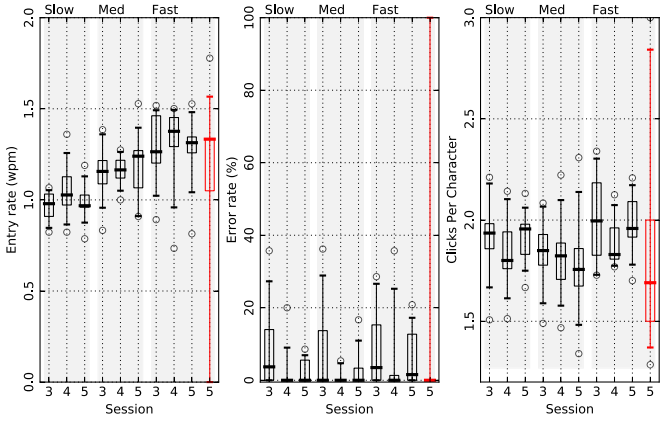

Fig. 6. Box-and-whisker plots for the multi-channel stereophonic user trials indicating the 5th, 25th, 50th, 75th and 95th percentile values. Results are plotted for each channel, and each speed setting (as indicated by the labels on the x-axis at the top and bottom of each plot). Results are computed over all words and over all participants. For example, the text entry rate plot on the left shows that 50 percent of users could select words at 1wpm when phrases were presented to them in 3-channel mode in "Slow" mode. Outliers are shown as circles. Black lines indicate results for able-bodied participants in the stereophonic-sound controlled experiment. The performance results obtained from a non-speaking individual with motor disabilities are shown in red.

5.1.1 Method

Entry rate was measured in words-per-minute (wpm), with a word defined as five consecutive characters, including spaces. Error rate was measured as the minimum edit distance between the response text and the stimulus text, divided by the number of characters in the stimulus text. We recruited 12 able-bodied participants from a university campus via convenience sampling.

The experiment was carried out as a single two-hour session, consisting of a 30-minute practice session and a 90-minute testing session. To reduce fatigue, the testing session was interleaved with breaks (30 minutes total). In the practice session the participant listened to the composite audio sequence with visual assistance, and practised selecting a few letters. In the testing session the number of channels were tested in random order and the speed was incremented slowly. Participants were exposed to roughly seven minutes per speed setting. To reduce cognitive load participants were visually shown the prompted phrase, with the current target letter highlighted.

Participants were also given a chart indicating which letters occurred in which channel. However, to successfully select the letters they had to understand how the system worked, and they had to correctly identify the letters when they heard it. Phrases were processed one word at a time. Thus, if the participant selected a word, the participant couldn't go back to change his selection, and had to proceed to the next word. The phrases were drawn from a widely used phrase set for text entry experiments [16].

The experiment was carried out similarly to the simulations described in Section 4.1. The participant was allowed to listen for as many composite audio sequences as desired before clicking. The participant was allowed only $2M$ composite audio sequences where one or more clicks occurred, otherwise a time-out failure was assumed (M is the length of the input word). In case of failure, wpm was set to zero and $\#errors$ was set to 100 percent, and the system proceeded to the next word.

TABLE 1
The Analysis for the Potential Effect of the Number of Audio Channels on the Text-Entry Rate

Fast	$F_{2,22} = 1.729$	$\eta_p^2 = 0.136$	$p = 0.201$
Medium	$F_{2,22} = 0.222$	$\eta_p^2 = 0.020$	$p = 0.803$
Slow	$F_{2,22} = 0.327$	$\eta_p^2 = 0.029$	$p = 0.725$

TABLE 2
The Analysis for the Potential Effect of the Number of Audio Channels on the Error Rate

Fast	$F_{1.352,14.870} = 3.948$	$\eta_p^2 = 0.264$	$p = 0.056$
Medium	$F_{1.371,15.080} = 3.338$	$\eta_p^2 = 0.233$	$p = 0.077$
Slow	$F_{2,22} = 0.808$	$\eta_p^2 = 0.068$	$p = 0.459$

TABLE 3
The Analysis for the Potential Effect of the Number of Audio Channels on the Clicks per Character

Fast	$F_{2,22} = 1.015$	$\eta_p^2 = 0.084$	$p = 0.379$
Medium	$F_{2,22} = 0.518$	$\eta_p^2 = 0.045$	$p = 0.603$
Slow	$F_{2,22} = 2.053$	$\eta_p^2 = 0.157$	$p = 0.152$

5.1.2 Results

The results from the controlled experiment are summarized in Fig. 6. We analyzed the potential effect of the audio channels using a General Linear Model repeated measures analysis of variance at initial significance level $\alpha = 0.05$ with Greenhouse-Geisser correction for violation of sphericity. The analyses, summarized in Tables 1, 2, and 3, revealed no significant differences in either entry rate, error rate or clicks-per-character.

The latter analyses revealed no significant differences in either entry rate, error rate or clicks-per-character. By inspection of the box-and-whisker plots in Fig. 6 and taking into account the low effect sizes (η_p^2 above), it appears unlikely that the number of audio channels would have an effect if this experiment would be replicated. Hence, we keep the null hypothesis and conclude that the number of audio channels (3, 4, or 5) is unlikely to exhibit a large effect on entry rate, error rate or the number of clicks per letter for the particular speed configurations considered in the experiment.

Human performance therefore does not seem to degrade substantially when increasing the number of channels from three to five, thereby validating our design choices related to assisting the user to change focus to a different voice. Note that the tested speeds were rather conservative. That is, there is overlap between the sound files, but we haven't tested the speed saturation point. The result from this experiment motivates further research to measure the saturation point for all channels, in order to better quantify the speed benefits of the proposed audio parallelization. An experiment can be constructed to measure the fastest speed attainable for the number of audio channels (for a reasonable error rate). The number of channels should also be increased up to breaking point.

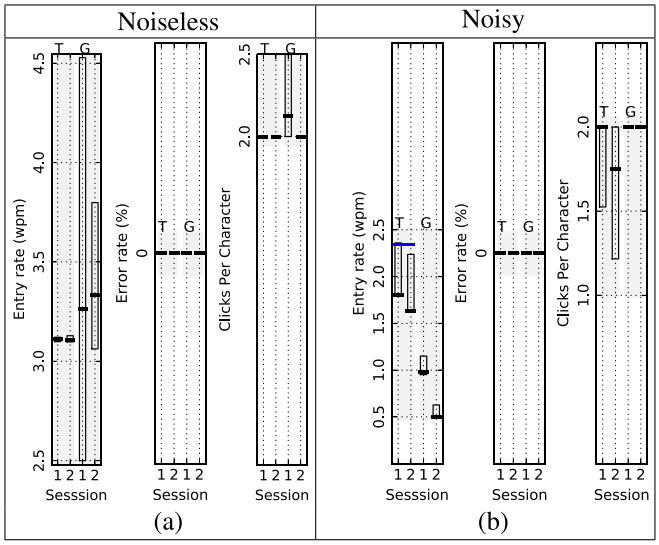

Fig. 7. Box-and-whisker plots of the audio pilot study, comparing Ticker (T) and a standard scanning system such as Grid2 (G), indicating the 25th, 50th, and 75th percentiles. Results for two sessions are shown in (a)-(b). Each session was 15 minutes long. Each session is numbered (x-axis). (a) Results for a trained participant (with at least two hours of practise) communicating in an environment with little noise. T_{S}^{*} and T_{S} were varied. The first session was recorded when the user had at least one hour of practise. The last session was recorded when the user could comfortably use the system blindfolded. (b) Results for the same participant in (a) but simulating a non-speaking individual with motor disabilities (by including synthetic noise and using the system blindfolded). Session 1 presents a user who can click precisely, but with some latency; $\Delta = 0.8$ s, $\sigma = 50$ ms, $\lambda = 0$, $f = 0$, $T_{\mathrm{S}}^{*} = 70$ ms, and $T_{\mathrm{S}} = 1.4$ s. The latency during Session 2 was increased and some false positives were randomly generated with $\Delta = 1.5$ s, $\sigma = 50$ ms, $f = 0.1$ and $\lambda = 1/3$/s, $T_{\mathrm{S}}^{*} = 70$ ms, and $T_{\mathrm{S}} = 2.1$ s. The theoretical maximum speed that can be attained when using Ticker without the unnecessary waiting time at the end of the composite audio sequence is shown in blue.

5.2 Non-Speaking Individual with Motor Disabilities

In addition to the controlled experiment we also carried out a case study with a non-speaking individual with motor disabilities who was unable to communicate on his own using the standard scanning system Grid2. This user communicated mostly by raising his eyebrows in an interactive conversation with his carer. The carer could also guess well what he tried to say after he selected a few letters. We automated this process using an Impulse switch attached to the user's eyebrow muscle and connected to Ticker.

The Impulse switch is quite prone to false positives and drift, especially if the user communicates for a while and his body temperature slightly increases. Since this end-user had vision problems all visual cues had to be replaced with audio cues.

We trained the end-user to use Ticker in four 2-hour sessions. During the last session the end-user was able to select 20 words (four phrases) at a rate of 1.3 wpm. No time-out errors occurred, and four of the 20 words were wrong. However, due to the context one could easily see which words the end-user meant. For example, "throb_" were selected instead of "three_" from the phrase "three_two_one_zero_blast_". All the other words were selected correctly.

Results are also shown in red in Fig. 6. Note that the median text-entry rate is 1.3wpm, thereby corresponding to the baseline text-entry rate defined earlier in this section.

A video of the participant using Ticker during one of the sessions is provided as supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2018.2865897, where the user writes the word "friend_". To reduce the cognitive load slightly, some audio cues were provided to tell the user which letter to select. The user had to be able to select letters from the sound files and had to know how the interface works in order to write words successfully.

5.3 Model Validation by Expert User

In order to further validate our modelling assumptions and observe the performance envelope of Ticker in relation to a standard hierarchical scanning system, such as Grid2, we recruited a single able-bodied participant for a model validation exercise.

We trained this participant to reach expert performance in both systems. For each system, training and testing were done in 4–5 hours, in several sessions and over several days. During training, the user gradually practised to memorize the alphabet layout and practised to write at fast and slow speeds. Eventually the participant became an expert user of both Ticker (5-channel mode) and the scanning system (audio mode) with closed eyes, and with little audio guidance. We used a phrase set specifically designed for testing text entry methods intended for non-speaking individuals with motor disabilities as stimuli [17] and added synthetic noise to simulate the practical realities of noisy single-switch systems (see Fig. 7b for more information regarding the synthetic noise).

In both systems the target phrase was read out the user. Each word constituting the target phrase was then processed in succession. After an audio prompt read out "new word", the target word was read to the user. In both systems, the letter index was presented to the user after two successive group scans in which no clicks were received. For example, the audio cue was "second letter" if the user waited for two group scans and the user already clicked for the first letter.

In Grid2 a time-out error was generated after $2M$ output characters (as part of an unsuccessful attempt to select the target word), where M was the number of characters in the target word. Similarly, in Ticker, a time-out error was generated if clicks were received during $2M$ scans of the composite audio sequence (allowing the user to listen to many group scans, but not to keep on clicking for an arbitrarily long time). A full-stop (".") or space ("_") resulted in a word selection causing the system to proceed to the next word in the target phrase.

During the first testing phase, the speed for both systems were gradually increased, until it felt too fast, in which case it was slowed down again. Once optimal speed for the participant was determined, synthetic noise was added to both systems. Ticker noise parameters were set to the synthetic noise parameters, and automatically refined during calibration and training according to the participant's abilities.

At the end the participant was effectively trained to represent an expert user on the system, able to use the system blindfolded with synthetic noise. This allows us to empirically observe an estimation of the human performance envelope of Ticker. The results are presented in Fig. 7.

Comparing the text-entry rate of Session 1 in Fig. 7b to Fig. 5a validates that the expert user's performance results closely reflect the simulation results for similar noise parameters. That is, for Ticker, $\Delta \in [0.8, 1.4]$s results in $wpm \in [1.7, 2.1]$ in Fig. 5a. A similar text-entry range was achieved in Session 1 of Fig. 7b. Likewise, for the scanning system, $wpm \approx 1.0$ in Fig. 7b and and Fig. 5a (close to the point where $\Delta = 1.4$ s). Note that a value close to $\Delta = 1.4$ s has to be used for the scanning system in Fig. 5a. This result corresponds to the scanning delay that was used to generate Fig. 7b, compensating for both the synthetic latency of 0.8 s and the users average response time which was unknown.

In Fig. 7b the theoretical maximum speed without unnecessary waiting time at the end of each composite audio sequence is shown in blue. Even with this waiting time, Ticker was on average about twice as fast as the scanning system during the first session, and more than three times faster during the second session (with similar click- and error rates).

The expert user found the long scanning delay in the scanning system, which is necessary for a long latency, cumbersome to use. The small number of false positives that were generated were difficult for him to cope with, as an arbitrary (and distracting) selection was sometimes made for him. In Ticker, one hardly notices the false positives, as no selection is made that has to be undone. The user hardly noticed the speed decreasing due to the unnecessary waiting time at the end of each composite audio sequence when using Ticker. Without any noise, the user found both systems easy to use, although it took about 30 minutes longer to learn Ticker.

6 Conclusions

We have presented Ticker: an audio-based noise tolerant single-switch text entry system for non-speaking individuals with motor disabilities. We have shown by modelling that Ticker exhibits similar performance to existing single-switch scanning systems when no noise is present. However, Ticker is likely to outperform scanning systems in the presence of switch- and click-timing noise—especially in the presence of false positives and long latencies. Such noise sources are common in practice due to imperfect devices failing to reliably detect switch events and due to cognitive and motor errors arising due to user-specific disabilities.

The performance modelling was validated by observing a single expert user's performance in both Ticker and using a standard scanning system and comparing predicted versus actual results.

We also feasibility tested Ticker with a non-speaking individual with motor disabilities who was unable to communicate on his own using a standard scanning system. Using Ticker this user was able to select letters from 20 words (four phrases) on his own at a rate of 1.3 wpm.

In addition, we have carried out an experiment that reveals that users can easily make use of multiple audio channels to select letters using Ticker. This result can serve as a general solution principle for other user interface designs; specifically when there is a need to use more than three stereophononic sound sources.

We hope Ticker will inspire further research in augmentative and alternative communication interfaces that leverage models of the uncertainty in user's interaction in order to increase the communication rate.

Acknowledgments

We thank Mick and his son Bill Donegan who connected us with several users, and assisted us with every visit to them. We specifically thank two users, Michael and Sara for their valuable feedback. Tom Ash initiated the project and suggested the name Ticker. We thank Christian Steinruecken for revising, and Philipp Hennig for proofreading the manuscript before its first submission. We thank Robert Fanner for his help with the software. The work was funded by the Aegis EU project and the Gatsby Charitable Foundation.

References

[1] S. T. Grafton, "Unlocking communication with the nose," *Proc. Nat. Acad. Sci. United States America*, vol. 107, no. 32, pp. 13 979–13980, Aug. 2010.

[2] K. Grauman, M. Betke, J. Lombardi, J. Gips, and G. Bradski, "Communication via eye blinks and eyebrow raises: Video-based human-computer interfaces," *Universal Access Inf. Soc.*, vol. 2, no. 4, pp. 359–373, 2003.

[3] D. S. Tan and A. Nijholt, Eds., *Brain-Computer Interfaces: Applying our Minds to Human-Computer Interaction*. London, U.K.: Springer, Jul. 2010.

[4] E. Nel, P. O. Kristensson, and D. J. C. MacKay, "Modelling noise-resilient single-switch scanning systems," 2017. [Online]. Available: https://arxiv.org

[5] Sensory Software International Ltd., "The grid 2 reference manual," [Online]. Available: http://sensorysoftware.com/downloads/, Accessed on: 2015.

[6] A. W. Bronkhorst, "The cocktail party phenomenon: A review of research on speech intelligibility in multiple-talker conditions," *Acta Acustica United Acustica*, vol. 86, pp. 117–128, 1999.

[7] A. W. Bronkhorst, "The cocktail-party problem revisited: Early processing and selection of multi-talker speech," *Attention Perception Psychophysics*, vol. 77, no. 5, pp. 1465–1487, Jul. 2015.

[8] L. J. Stifelman, "The Cocktail Party Effect in Auditory Interfaces: A Study of Simultaneous Presentation," MIT Media Laboratory, Cambridge, MA, 1997.

[9] E. Nel, "Ticker," 2017. [Online]. Available: https://github.com/singleswitch/ticker

[10] E. Nel, P. O. Kristensson, and D. J. C. MacKay, "The statistical model for ticker, an adaptive single-switch text-entry method for visually impaired users," 2018. [Online]. Available: https://arxiv.org

[11] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd Ed. Berlin, Germany: Springer, 2008.

[12] J. F. C. Kingman, *Poisson Processes*, vol. 3. Oxford, U.K.: The Clarendon Press Oxford University Press, 1993.

[13] T. Broderick and D. J. C. MacKay, "Fast and flexible selection with a single switch," *PLoS One*, vol. 4, no. 10, Oct. 2009, Art. no. e7481.

[14] A. Kilgarriff, "BNC database and word frequency lists," 1998. [Online]. Available: http://www.kilgarriff.co.uk/bnc-readme.html/

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.

[16] I. S. MacKenzie and R. W. Soukoreff, "Phrase sets for evaluating text entry techniques," in *Proc. Extended Abstracts ACM Conf. Human Factors Comput. Syst.*, 2003, pp. 754–755.

[17] K. Vertanen and P. O. Kristensson, "The imagination of crowds: Conversational AAC language modeling using crowdsourcing and large data sources," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2011, pp. 700–711.

Emli-Mari Nel was a postdoctoral researcher with Prof. David MacKay at the Cambridge Inference Group from 2009 to 2014. She is currently a researcher at Empiric Capital, a hedgefund in the Western Cape, South Africa, and also works as a freelance machine learning researcher.

Per Ola Kristensson is a University reader with the Department of Engineering, University of Cambridge and fellow of Trinity College, Cambridge. In 2013 he was recognized as an Innovator Under 35 (TR35) by MIT Technology Review and appointed a member of the Royal Society of Edinburgh Young Academy of Scotland. In 2014 he won the ACM User Interface Software and Technology (UIST) Lasting Impact Award and the Royal Society of Edinburgh Early Career Prize in Physical Sciences, the Sir Thomas Makdougall Brisbane Medal.

David MacKay received the PhD degree in 1992 from Caltech as a Fulbright Scholar, where his supervisor was John Hopfield. He was chief scientific adviser to the United Kingdom Department of Energy and Climate Change (DECC) from 2009 to 2014. He was elected a fellow of the Royal Society (FRS), in 2009. He was regius professor of engineering at the University of Cambridge from 2013 to 2016. In 2016 he was appointed a Knight Bachelor. He wrote two books: Sustainable Energy Without the Hot Air (which sold 40,000 copies and has been downloaded nearly half a million times); and Information Theory, Inference, and Learning Algorithms. His contributions in machine learning and information theory include the development of Bayesian methods for neural networks, the rediscovery (with Radford M. Neal) of low-density parity-check codes, and the invention of Dasher, a hands-free keyboard.