Dynamics and probabilistic text entry*

John Williamson[1]
[1] Department of Computing Science,
University of Glasgow,
Glasgow G12 8QQ,
Scotland, UK.
jhw@dcs.gla.ac.uk

Roderick Murray-Smith[1,2]
[2] Hamilton Institute,
National Univ. of Ireland, Maynooth,
Co. Kildare,
Ireland
rod@dcs.gla.ac.uk

June 11, 2003

Abstract

We present a gestural interface for entering text on a mobile device via continuous movements, with control based on feedback from a probabilistic language model. Text is represented by continuous trajectories over a hexagonal tessellation, and entry becomes a manual control task. The language model is used to infer user intentions and provide predictions about future actions, and the local dynamics adapt to reduce effort in entering probable text. This leads to an interface with a stable layout, aiding user learning, but which appropriately supports the user via the probability model. Experimental results demonstrate that the application of this technique reduces variance in gesture trajectories, and is competitive in terms of throughput for mobile devices. This paper provides a practical example of a user interface making uncertainty explicit to the user, and probabilistic feedback from hypothesised goals has general application in many gestural interfaces, and is well-suited to support multimodal interaction.

1 Introduction

Text entry is an important part of all human computer interfaces, and is particularly important for communication between humans via computer. However, entry on mobile devices can be problematic compared to established keyboard-based desktop systems. The restricted size and reduced processing power of mobile devices, and the changing contexts in which the devices are used are all obstacles to efficient text entry. Current approaches include virtual keyboards (Kolsch and Turk 2002), handwriting recognition (Plamondon and Srihari 2000), and gesture-based interfaces; the latter is of interest here.

*Technical Report TR-2003-147, Dept. Computing Science, University of Glasgow

Various interfaces for gestural text entry have been devised, including fixed layout approaches such as (Mankoff and Abowd 1998) and (Perlin 1998), and dynamic layout approaches, as in Dasher (Ward *et al.* 2000) and in (Bellman and MacKenzie 1998). Of these, only the latter systems support a probabilistic model for increasing accuracy and throughput. Support for a probabilistic model is vital for optimal performance; in particular the best use of the limited bandwidth available can be made only if the uncertainty in language is adequately represented.

Systems that dynamically optimize letter layouts can impair learning, as the constantly changing interface lacks the stability needed to learn to perform automatic movements. Although initial learning times may be very short, the transition from a novice to an expert user is often slow or impossible. At the novice level, the user is totally dependent on feedback, while at an expert level, control is more open-loop with rapid, learned responses producing desired control actions. If the configuration changes significantly with varying contexts, learning and producing such automatic high-speed responses is more difficult.

In this paper, we describe a system which uses continuous gestures to produce text. The handling qualities of the system are dynamically altered according to a time-varying probability model. The gestures for letter sequences remain stable, supporting user learning and high-speed, open-loop gesturing. However, the changing control properties reduce the effort required to choose highly probable sequences, and so there is a direct relation between the information content of a sequence and the effort that must be expended by the user.

2 Design

2.1 Layout

Each symbol is coded as a pair of primitive gestures, these "gestures" being movements in one of six directions, allowing thirty-six symbols. Such a division leads to a hexagonal layout, with two stages: selecting a letter group and selecting a letter. As hexagons form a regular tiling, gestures for letter sequences are represented as paths through such a plane (see Figure 1). Recognition involves selecting points in space such that the Voronoi tessellation (using the L2 norm) is hexagonal. Crossing cell boundaries in this tessellation triggers transitions in a finite-state model, which outputs symbols in response.

2.2 Control

The interface can be controlled with a number of input devices; mice and accelerometers are used in the prototypes presented here. The tessellation must be effectively unbounded to permit all combinations of symbols to be entered, and so input deflection is mapped to velocity allowing apparently infinite range of movement. A nonlinear transfer function, with a dead-zone around zero (see (Jagacinski and Flach 2003)) is used to help stabilize the control.

The handling qualities of the system are manipulated by simulating a nonlinear landscape on the selection plane. The local system dynamics are altered by a vector force-field which is computed from the current probabilities. This field is conditioned on the current context, where the

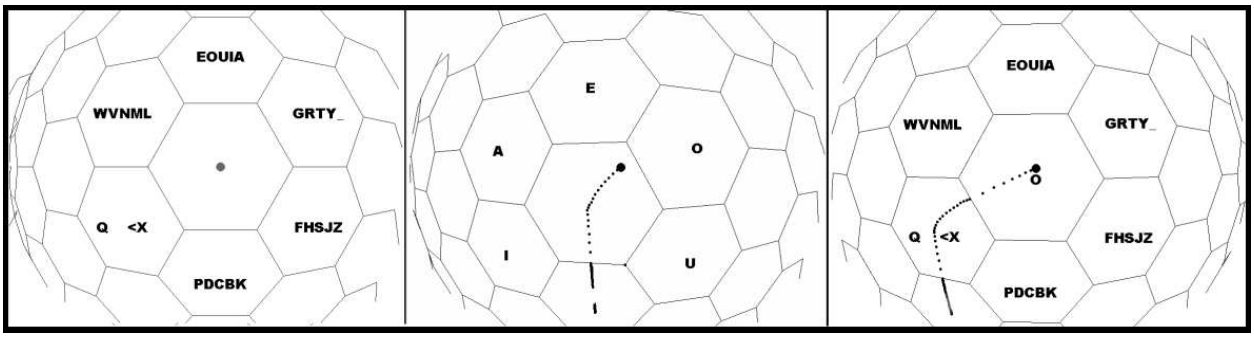


Figure 1: The hexagonal layout. Each letter is assigned to a group, within which it is associated with a particular edge. In this example, producing "o" requires an upwards then up-right movement.

context may incude position, velocity, acceleration, and the probability of a letter given the current prefix. Given the state x of the system, we have

$$\dot{x} = A(c)x + B(c)u \tag{1}$$

where u is the control action, and $A(c)$ and $B(c)$ are context-varying state and control matrices, conditioned on the context c.

The force vectors require greater control effort on the part of the user to move into low probability areas; this can be thought of as a system of hills and valleys guiding the user away from improbable regions of the state space.

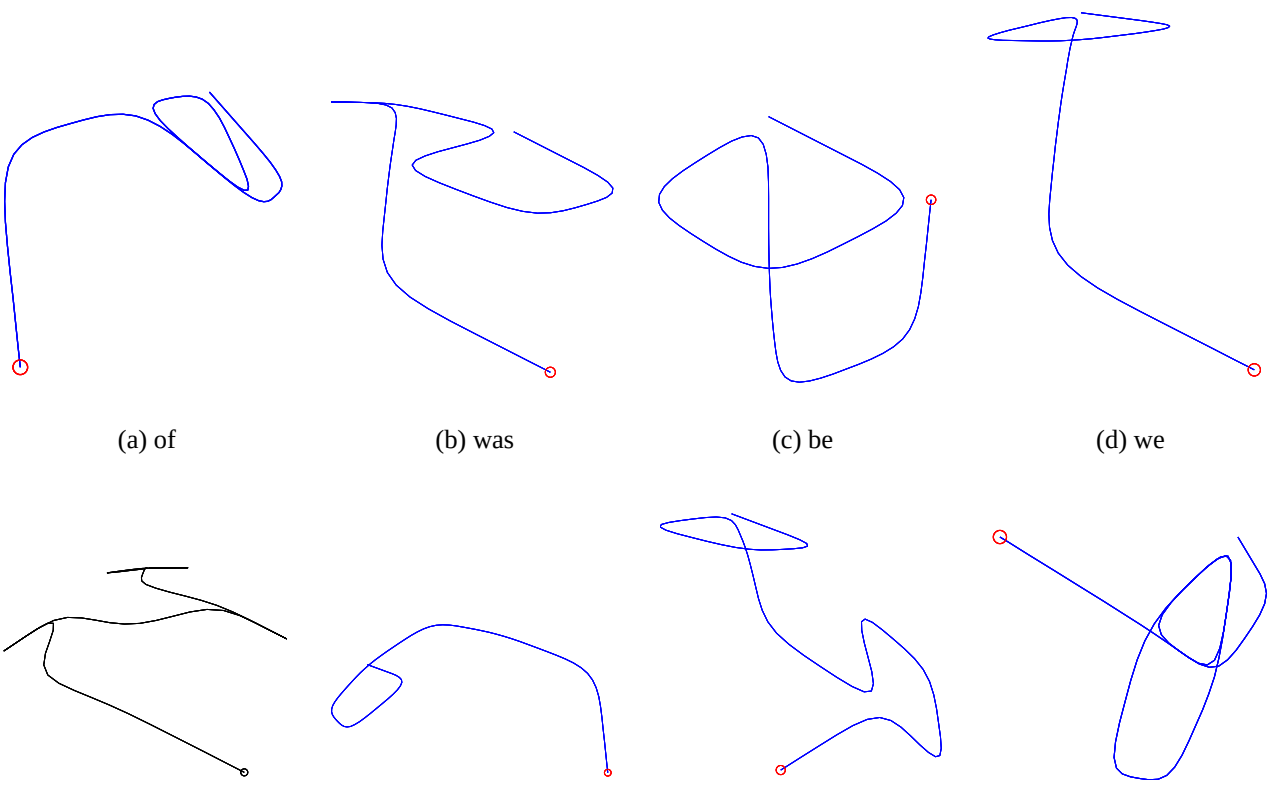
Given six discrete probabilities for each possible transition, $p_1, .., p_6$, the force at any point is given by a function with squared-exponential decay from the vertices. Each force is applied from the two vertices which form the boundary across which the transition can occur. The magnitude of the force applied at each point is given by:

$$f = \frac{k}{2} \sum_{i,j} e^{\frac{d(v_{i(j)})^2}{s}} p_i \tag{2}$$

where $d(v_{i(j)})$ is the Euclidean distance from the jth ($j = 1..2$) vertex of the ith ($i = 1..6$) edge, k is a constant scaling the magnitude of the forces, and s is a constant specifiying the width of the density around the vertex.

An example landscape is shown in Figure 3, after "q" has been entered, which shows the deep valley towards the letter group containing "u".

In addition to this field, fixed forces are applied at the vertices, repelling the user from these points. This limits ambiguous transitions, and forces the user to make a conscious choice at these decision points. These forces are applied as above, having squared exponential decay, but with constant magnitude. The forces act along the direction from the user to the vertex, avoiding the slingshot effect that would occur if the forces were normal to the density.



(a) of (b) was (c) be (d) we

(e) little (f) am (g) take (h) side

Figure 2: Eight common English words, and paths through the hexagonal space that will produce them. These paths are generated via cubic splines (see Section 2.5)

.

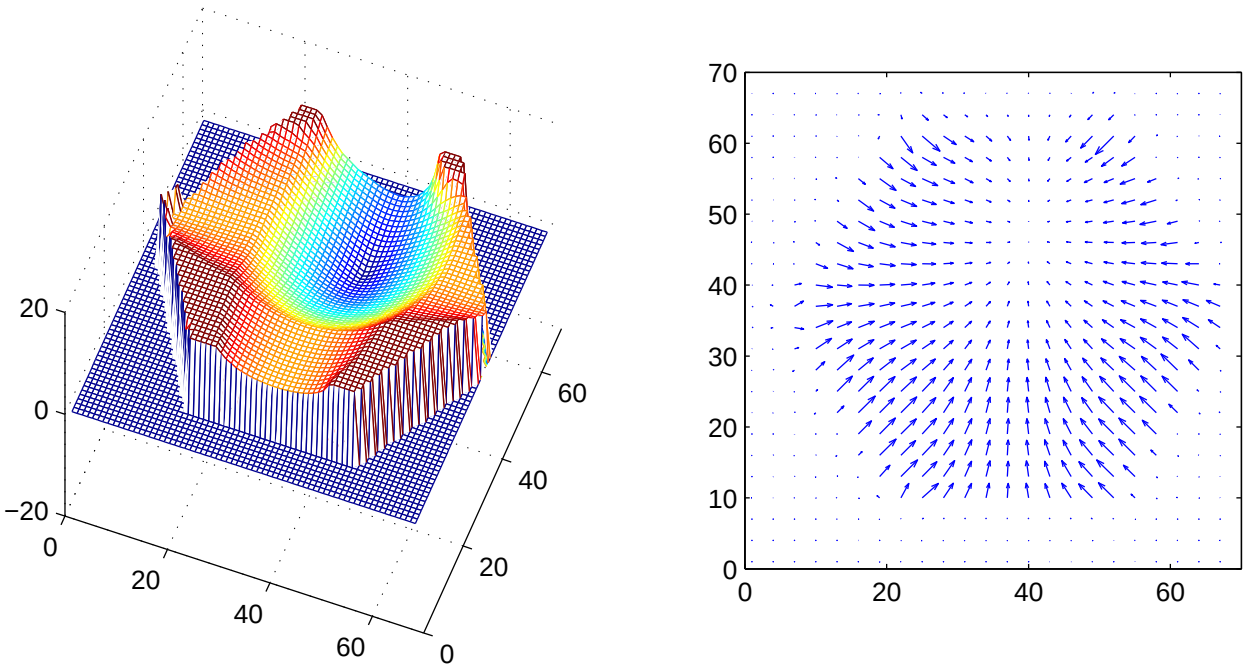


Figure 3: The vector field (right) produced after the letter “q” is entered, and its magnitude, shown as the surface plot on the left. The group boundary leading to “u” is at the top of these diagrams.

2.3 Probability model

A simple language model (based upon partial predictive matching, see (Cleary and Witten 1984, Cleary *et al.* 1995)) is used to produce $p(\text{letter}|\text{prefix})$ (referred to as $p(l|pr)$) on a per-word basis. A tree with probability information is generated from a corpus (in this case texts from Project Gutenberg (Hart 2003)). For simplicity, no grammar or word-level model is used, although this would be likely to improve performance significantly (Lesher and Rinkus 2002). More complex language models can easily be incorporated in this framework.

The probability model is extended to include the dynamics of the cursor. The velocity and acceleration of the cursor are numerically estimated. The probability of heading into a hexagon is then given by

$$\frac{\cos\theta + 1}{2}, \tag{3}$$

where θ is the angle between the movement vector and the center of the hexagon being tested. The probability of each hexagon is given by

$$p(h) = p(h|pr)p(h|v)p(h|a) \tag{4}$$

. If a transition into hexagon h represents a single letter l_h then

$$p(h|pr) = p(l_h|pr), \tag{5}$$

otherwise

$$p(h|pr) = \sum_{i=1}^{6} p(l_{hi}|pr), \tag{6}$$

where l_{hi} is the ith letter in the letter group selected by a transition into h.

2.4 Autocompletion and prediction

Potential autocompletions can be predicted using Monte-Carlo sampling. Starting from the current prefix a potential symbol is selected randomly, weighted according to $p(l)$. This letter is concatenated to the prefix, and the process repeated until the end-of-word symbol is produced. The probability of the sequence is evaluated as a by-product of this process.

Each of these word/probability pairs is stored in a list ranked by probability. The sampling is repeated k times, with $k \approx 300$ in the current implementations. The top autocompletion is then presented, and the autocomplete action can be initiated either by a specific button press (in the case of a mouse) or a simple shake gesture (for orientation sensors).

The display can show the path which would generate the current autocomplete possibilities. Fitting a cubic spline through the medians (the centers of edges) of the hexagons gives a smooth path which will generate a given letter sequence. Displaying these splines for the top autocompletes shows the paths of possible completions, giving a background awareness of the "word density" at any point in state space. It also facilitates the learning of smooth trajectories for words. We are currently extending this feedback to an audio display, based on ideas we presented in (Williamson and Murray-Smith 2002), where we describe a system for audio display of time-varying probabilities. The use of audio is important for mobile devices, where screen space is at a premium, and users visual load is often already high.

2.5 Layout optimization

The layout used in preceding examples was chosen to aid learning , by grouping letters in a logical manner (such as grouped vowels). Given a source corpus, it is possible to optimize the layout to minimize some cost function, given a model of the user's movement. The cost of a particular layout is

$$c_t = \sum_{i=1}^{n} p(w_i)c(w_i) \tag{7}$$

where n is the number of words in the dictionary, and c is the cost for each word. For the sake of computational efficiency implemetations prune the cost evaluation, letting n be the top ranked few hundred words from the corpus.

The cost function used should minimize some aspect of effort on the part of the user; here we penalize the sum of squared j-th derivatives of the trajectory representing the word, i.e we have:

$$c(w_i) = \int_0^t \left(\alpha_j \left(\frac{d^j x}{dt^j} \right)^2 + \beta_j \left(\frac{d^j y}{dt^j} \right)^2 \right) dt. \tag{8}$$

In the implementations the third derivative is penalized. This is based on a minimum-jerk model (Flash and Hogan 1985), in contrast to the linear-segment model proposed in (Isokoski 2001). Finally, a model of the user's movement is required; we approximate it with a cubic spline path. This simple approximation is justified experimentally in Section 3.1. We then numerically optimize the layout to minimize c_t.

2.6 Implemenations

The system has been implemented running on a desktop PC with a mouse and with an InterTrax accelerometer, and on the PocketPC platform with an accelerometer (see Figure 4).

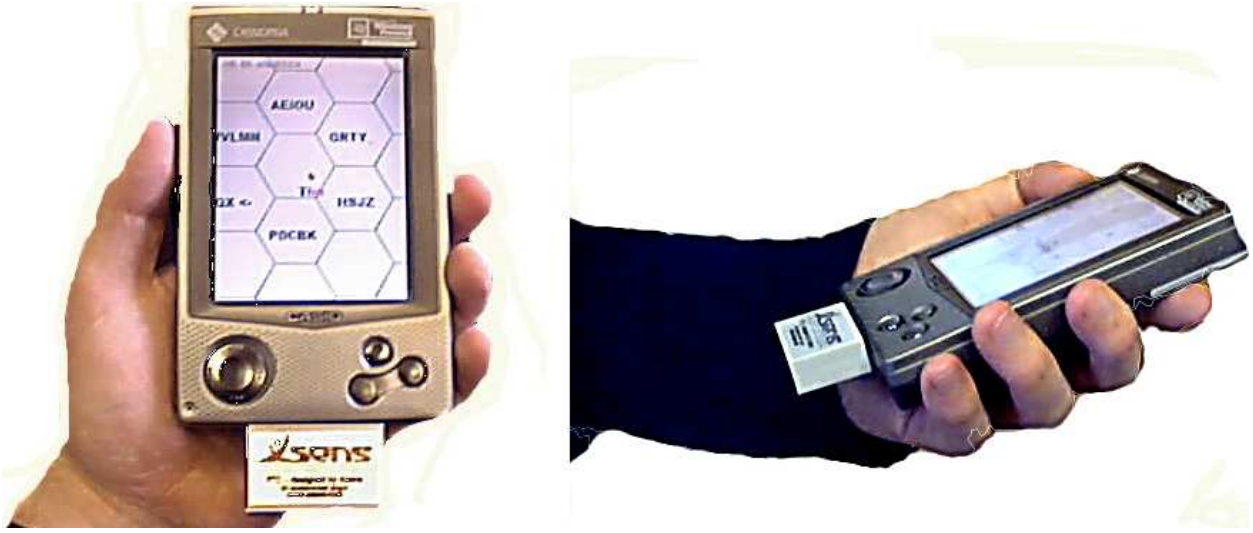


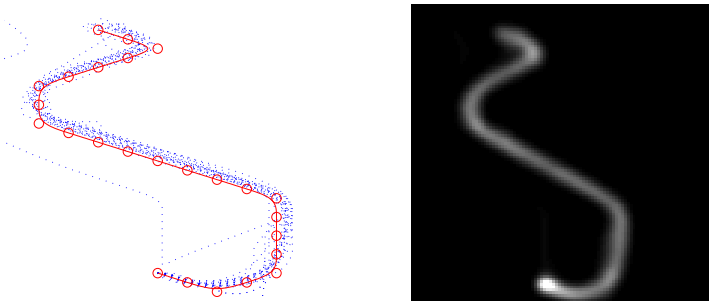
Figure 4: The system running on Cassiopeia E115 with a minature accelerometer

3 Results

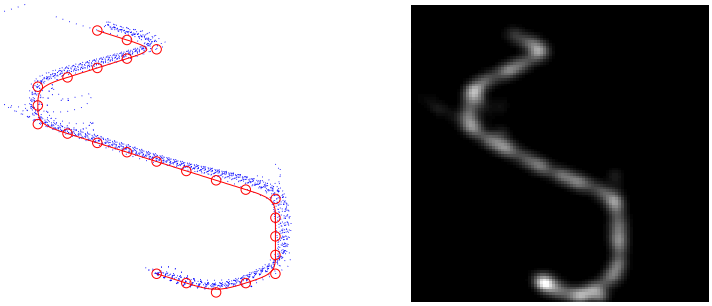
In throughput testing, one of the authors achieved around 10–12 words per minute with earlier versions of the system. This is the rate for perfect transcription of a hundred words of written text (rather than groups of five characters per minute), including error-correction time. In this case, the user had around 30 hours of use with the layout used for the test. Speeds of around 17wpm are achievable with current versions, for free-form text entry. It should be borne in mind that the layout used for these tests was not optimized (see Section 2.5).

3.1 Spatial effects

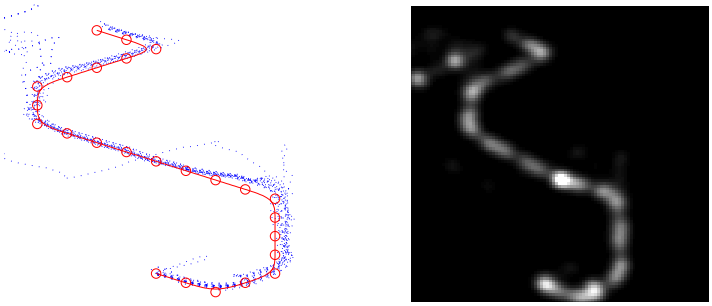
Figure 5 shows twenty trajectories for the word "hello", as performed by one of the authors as force model is adjusted. The four experimental conditions are: forces applied as previously de-



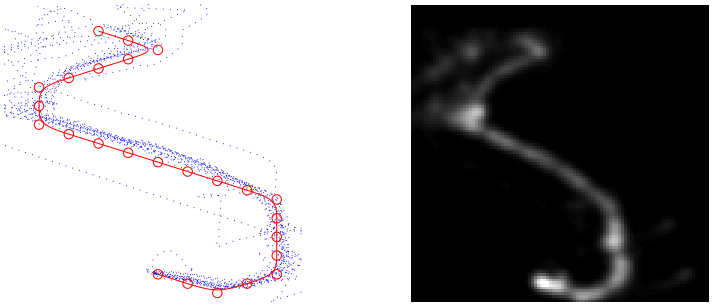
(a) No forces applied



(b) Forces applied as described previously



(c) Forces with double magnitude



(d) Forces with probabilities inverted

Figure 5: Trajectories from twenty repetitions of the gesture for “hello” with varying forces. On the left panel, dashed lines show measured trajectories, circles indicate the centers and medians of the hexagons, and the solid line indicates a cubic spline fit through the medians. The right panel shows a density plot produced by summing each of the data points, after convolving with a smoothing window, onto a mesh (higher density areas are lighter). When velocity is lower the local density will increase, assuming equal path density.

scribed; forces not applied; forces applied with double magnitude; forces applied as normal but with probabilities inverted.

Also shown is a cubic spline fit through the medians of the hexagons. It is apparent that the spline fit is a reasonable approximation; the cubic spline is within the distribution of points on the trajectory for most of the path. Exceptions occur at significant decision points where the user follows a less constrained path.

The intention of the force model is to increase accuracy and speed in performance. If the hypothesis that accuracy would increase is to be verified, then the distribution of the trajectories should be narrower for the cases where forces are present than when they are not. This can be seen when comparing Figure 5(a) (no forces) with Figure 5(b) (with forces). Increasing the forces should amplify these effects; this is apparent in Figure 5(c).

To illustrate the effect of the choice of language model on the performance of the system, Figure 5(d) shows the result of inverting the probabilities in the language model (p becomes $1-p$). This results in a significant increase in the deviation from the ideal path, particularly towards the end when the model is confident of its predictions, and so is opposing most strongly. The vertex and friction forces are as in the other tests, and so all changes of performance can be attributed to the change in the language model.

3.2 Temporal effects

The right-hand panel in Figure 5 shows the effect of the forces on the timing of the gesture. Without forces applied (Figure 5(b)) the path is smooth, without any significant pauses or accelerations (except at the start). The two runs with forces applied normally and at double strength show a strongly periodic movement. This periodicity is significantly diminished in the example with inverted forces, even though the forces are of the same magnitude as Figure 5(a). This enforced periodicity may be due either to a change in the control strategy pursued by the human, or may simply be a by-product of the changing system dynamics; more testing will be required to separate these issues.

Whichever is the case, it is a potentially powerful feature. A periodic interface allows for task interleaving; this is important for mobile devices where interaction may be occurring while occasional attention is required elsewhere. The periodicity of motion may be a useful metric for estimating performance in an adaptive system – it seems possible that confident users will produce more regularly timed movements than users who are relying more heavily on feedback control. Rhythmic movement can also be of use in feedback presentation, particularly in the audio modality, allowing for structured output which requires less constant attention.

4 Conclusions

We have created a text entry system based on continuous gestures performed on a regular tessellation, and demonstrated how dynamically altering the handling qualities of the system given a probabilistic model of context can improve performance. Testing shows that the variance of trajectories for probable sequences can be reduced using this method. Further systematic user trials will

be required to establish the effects at the various stages of learning.

This control-based approach supports users without constraining them, resisting low probability actions but not preventing them. This creates a correspondence between the information content of a sequence and the expenditure of energy on the part of the user. It also facilitates a smooth transition from unskilled, feedback-dependent users, to skilled users performing automatic, open-loop movements.

Our system uses the probability of the hypothesised goals compatible with the current context, to provide feedback directly to the user or by adapting the local dynamics of interaction. This is a general technique of interest to the whole area of gesture-interface design, improving throughput and supporting exploration and learning in new users.

Acknowlegements

Both authors are grateful for support from EPSRC grant Modern statistical approaches to off-equilibrium modelling for nonlinear system control GR/M76379/01, and Audioclouds: three-dimensional auditory and gestural interfaces for mobile and wearable computers GR/R98105/01. The Multi-Agent Control Research Training Network EC TMR grant HPRN-CT-1999-00107. We thank Xsens for the use of the P^3C accelerometer.

References

Bellman, T. and I. S. MacKenzie (1998). A probabilistic character layout strategy for mobile text entry. In: *Proceedings of Graphics Interface '98*. pp. 168–176.

Cleary, J.G and I.H. Witten (1984). Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications* **32**(4), 396–402.

Cleary, J.G., W.J. Teahan and I. H. Witten (1995). Unbounded length contexts for ppm. In: *Proceedings DCC'95*. pp. 52–61.

Flash, T. and H. Hogan (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neuroscience* **5**(7), 1688–1703.

Hart, M. (2003). Project gutenberg. Available at http://promo.net/pg/.

Isokoski, P. (2001). Model for unistroke writing time. In: *CHI*. pp. 357–364.

Jagacinski, R.J. and J.M. Flach (2003). *Control theory for humans : quantitative approaches to modeling performance*. L. Erlbaum Associates. Mahwah, N.J.

Kolsch, M. and M. Turk (2002). Keyboards without keyboards: A survey of virtual keyboard implemenations. In: *Proceedings of Sensing and Input for Media-centric Systems*.

Lesher, G.W. and G.J. Rinkus (2002). Leveraging word prediction to improve character prediction in a scanning configuration.. In: *Proceedings of the RESNA 2002 Annual Conference*.

Mankoff, J. and G. D. Abowd (1998). Cirrin: A word-level unistroke keyboard for pen input. In: *ACM Symposium on User Interface Software and Technology*. pp. 213–214.

Perlin, K. (1998). Quikwriting: Continuous stylus-based text entry. In: *ACM Symposium on User Interface Software and Technology*. pp. 215–216.

Plamondon, R. and S. N. Srihari (2000). On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(1), 63–84.

Ward, D. J, A. F. Blackwell and D. J. C. MacKay (2000). Dasher - a data entry interface using continuous gestures and language models. In: *UIST'00*. pp. 129–137.

Williamson, J. and R. Murray-Smith (2002). Audio feedback for gesture recognition. Technical Report TR-2002-127. Dept. Computing Science, University of Glasgow.