

A Directional Stroke Recognition Technique for Mobile Interaction in a Pervasive Computing World

Vassilis Kostakos & Eamonn O'Neill

*Department of Computer Science, University of Bath,
Bath BA2 7AY, UK*

Tel: +44 1225 384432 / 383216

Email: {vk, eamonn}@cs.bath.ac.uk

This paper presents a common gestural interface to mobile and pervasive computing devices. We report our development of a novel technique for recognizing input strokes on a range of mobile and pervasive devices, ranging from small devices with low processing capabilities and limited input area to computers with wall-sized displays and an input area as large as can be accommodated by motion-sensing technologies such as cameras. Recent work has included implementing and testing our stroke recognition technique and associated object-tracking by camera. Ongoing and future work includes optimizing our stroke recognition and camera-based object-tracking techniques, developing input based on human body tracking and running extended usability evaluations.

Keywords: stroke recognition, gestural interfaces, pervasive computing, ubiquitous computing, mobile interaction.

1 Introduction

One of the most exciting developments in current HCI is the shift in focus from computing on the desktop to computing in the wider world. Computational power and the interfaces to that power are moving rapidly into our streets, our vehicles, our buildings and our pockets. The combination of mobile/wearable computing and pervasive/ubiquitous computing is generating great expectations.

We face, however, many challenges in designing human interaction with mobile and pervasive technologies. In particular, the input and output devices and methods

of using them that work (at least some of the time!) with desk-bound computers are often inappropriate for interaction on the street.

To take a simple example, a keyboard does not transfer well from the office to the street. Simplistic solutions seen with several current mobile and wearable devices, such as making the keyboard much smaller, create their own usability problems. Similarly, usability problems are introduced by taking an interface designed for desktop monitors, such as a desktop operating system's graphical front-end, and putting it, with minimal changes, on a Pocket PC with a tiny display area and very different input devices.

Physically shrinking everything including the input and output devices does not create a usable mobile computer. Instead, we need radical changes in our interaction techniques, comparable to the sea-change in the 1980s from command line to graphical user interfaces. As with that development, the breakthrough we need in interaction techniques will most likely come not from relatively minor adjustments to existing interface hardware and software but from a less predictable mixture of inspiration and experimentation. For example, Brewster and colleagues have investigated overcoming the limitations of tiny screens on mobile devices by utilizing sound and gesture to augment or to replace conventional mobile device interfaces [Brewster 2002; Brewster et al. 2003].

2 Searching for New Interaction Techniques

In our current research, we assume that there will be an increasing convergence between mobile/wearable computing and pervasive computing. The latter, being built into the environment, for example in a shop or library or on a bus, can have very large processing and data storage power and (at least in the fixed location examples) very fast, high bandwidth connections to other distributed resources. In addition, input and output devices can take many different forms, from traditional keyboards and monitors to huge displays and touch screens.

In contrast, and notwithstanding continuing improvements in hardware and communications technologies, mobile and wearable computers are likely to continue to suffer from relatively weaker data storage, processing power and network connectivity.

As the technologies mature, we must learn to play to the strengths of each form. For many applications, the user may want to use, say, the wall display in the shop with the high bandwidth connection rather than the tiny display on her PDA with its relatively poor connectivity. For other applications, the user may prefer to take advantage of the characteristics of her mobile device. Indeed, some applications may be most usable through simultaneous use of a combination of pervasive and mobile computing power.

A persistent problem with the usability of mobile computing has been the conflation of the physical characteristics of the device with the characteristics of the interface between the user and the computing functionality which the device delivers. For example, as mobile and wearable devices become ever smaller, their display areas, which typically serve as both input and output devices, become ever smaller and less usable. The legitimate desire to make mobile and wearable devices

more mobile and easier to wear through miniaturization will render these devices less and less usable so long as the interface and its associated interaction techniques continue to be conflated with the physical characteristics of the device itself.

In attempting to resolve this dilemma, we are exploring ways of decoupling the interaction techniques from the devices. In the context of converging mobile and pervasive technologies, this means developing interaction techniques that will work with devices ranging from the smallest wearable computer or smart ring to a wall-sized display driven by a powerful fixed-location computer in a shop or street. In this paper, we focus on our work to develop an input technique that will satisfy these criteria.

3 Stroke Recognition for Mobile and Pervasive Interaction

Given the inadequacies of traditional desktop input techniques, i.e. mouse and keyboard, in a pervasive computing environment and, even more so, with mobile and wearable computing, there has been considerable research investigating alternative techniques. Prominent amongst these is gesture or stroke based input [Pirhonen et al. 2002]. This has formed the basis for many of the input techniques used with PDAs, whether in the form of touchscreen strokes to perform commands or in the form of alphabets, such as Graffiti on the Palm range of PDAs.

Moreover, stroke recognition predates PDAs by quite a while. One of the first applications to use some sort of stroke recognition was Sutherland's Sketchpad [Sutherland 1963]. The idea of mouse strokes as gestures dates back to the 1970s and pie menus [Callahan et al. 1998]. Since then, numerous applications have used similar techniques for allowing users to perform complex actions using an input device. For instance, design programs like [Zhao 1993] allow users to perform actions on objects by performing mouse or pen strokes on the object. Recently, Web browsing applications, like Opera¹ and Mozilla², have incorporated similar capabilities. Guimbretiere et al. [2001] show how FlowMenu [Guimbretiere & Winograd 2000] may be used with large wall displays. FlowMenu is very similar to pie menus. Unless the FlowMenu has been displayed, any pen stroke is interpreted as simple mouse input, using a simple down-move-up event model.

There is a number of current open source projects which involve the development of stroke recognition, including Mozilla, Libstroke³, XScribble⁴, and WayV⁵.

The latter is a library created for recognizing characters as well as strokes. It is based on a technique called point density analysis which uses matrix mathematics. In its latest version it has included a second 'backup' method for recognizing strokes, which implements a form of directional recognition. This method imposes an $n \times n$ matrix on the stroke and assigns every stroke point to a cell in the matrix. By comparing the relative position of two boxes which contain consecutive stroke

¹<http://www.opera.com/features/mouse/>

²<http://optimoz.mozdev.org/gestures/>

³<http://www.etla.net/libstroke/libstroke.pdf>

⁴<http://www.handhelds.org/projects/xscribble.html>

⁵<http://www.stressbunny.com/wayv/>

points, a sequence of directions is produced, and is used to assist the point density analysis algorithm in the recognition of the stroke.

The Mozilla browser uses a simpler technique. Each point of the gesture is compared to the previous one, and one of four directions is generated (U, D, L, R), while discarding consecutive U's, D's etc. Then, the sequence is compared against a table of stroke signatures, and if no exact match is found, then only the last 2 and then the last 3 elements of the direction signature are used. If that fails, then the signature is processed for diagonals, simply by replacing consecutive L's and D's by '1' (for diagonally left-down), R's and D's by '3', L's and U's by '7', and R's and U's by '9'. Then, this modified signature is checked against a table for matches.

Learning techniques have been applied to stroke recognition, with some success. For instance, Boukreev⁶ has implemented stroke recognition using neural networks. This technique involves recording the path of the stroke, smoothing it to base points, translating it to the sines and cosines of the points' angles, and then passing these values to a neural network. The neural network will try to recognize the stroke, and in the process of doing so, will actually improve its recognizing capability.

Its range of uses over the past three decades illustrates a key characteristic of stroke recognition as an input technique: it is not tightly bound to a particular device. Our aim in this research is to exploit this characteristic to develop an input technique that can be used seamlessly across the wide range of devices in a mobile-populated, pervasive computing world.

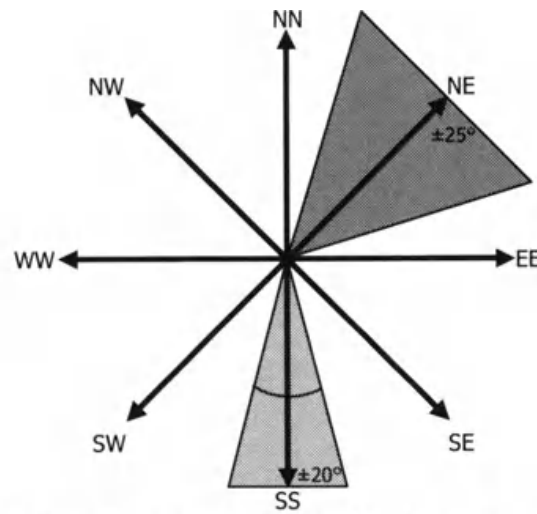
The diverse characteristics of such devices, and potential future devices, impose key requirements on such an interaction technique. At one end of the scale, the user may wish to interact with a device as limited in processing power and surface area as a smart ring or credit card, perhaps using a stylus to make the gestures. At the other end of the scale, the user may wish to interact with a wall-size display, perhaps using the smart ring itself, or indeed using just the user's hand, to make the gestures in the air. Igarashi et al. [2000] present a framework for using wall-sized displays using pen input. They describe how defining different application behaviours can provide a means of dealing with input strokes in different ways.

The work reported here presents a technique for recognizing input strokes which can be used successfully on devices with very low processing capabilities and very limited space for the input area. The technique is based on the user's denoting a direction rather than an actual shape and has the twin benefits of computational efficiency and a very small input area requirement. We have demonstrated the technique with mouse input on a desktop computer, stylus and touchscreen input on a wearable computer and hand movement input using real-time video capture.

4 Directional Stroke Recognition

We have developed a technique for directional stroke recognition. As its name implies, this is a technique for recognizing strokes based solely on their direction. Other characteristics of a stroke are not used. For instance, the position of a stroke is of no importance, nor are the relative positions of several strokes.

⁶<http://www.generation5.org/aisolutions/gestureapp.shtml>



For consistency purposes, every direction is represented by a two letter combination. Therefore, North, East, South, and West are represented as NN, EE, SS, and WW respectively.

Figure 1: Calculating the direction of a line.

4.1 Stroke Recognition

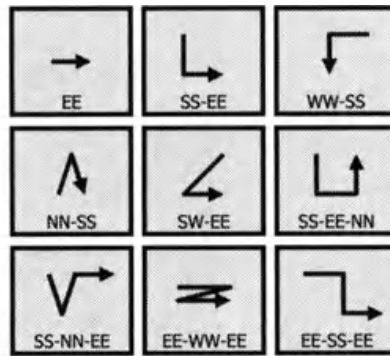
The first step of our stroke recognition method is to collect the input data. Typically, the data for a stroke performed by the user is a set of coordinates. Our method regards the input stroke as an ordered set of lines. Each line consists of a ‘fromPoint’ and a ‘toPoint’. Using these two coordinates, we can calculate the direction of each line as shown in Figure 1.

Humans tend to be more accurate at drawing vertical and horizontal lines than diagonals [Pirhonen et al. 2002], especially when on the move. Therefore, by adjusting the relative angle for acceptance, e.g. a variation of 25° for diagonals and 20° for other strokes, we may accommodate inaccuracy in stroke direction.

At this stage we have a stream of ‘directions’, for example: “SS, SS, SS, SS, WW, WW, WW, WW, NW, NW, NW”. The next step is to remove noise from this stream. This is achieved by setting a threshold as a percentage of the length of the whole stroke. This threshold is applied by removing any sequence of identical directions that does not reach the threshold. So, for example, for a threshold value of 10% and a stroke recorded as a stream of 40 directions, any contiguous sequence of fewer than 4 identical directions would be removed.

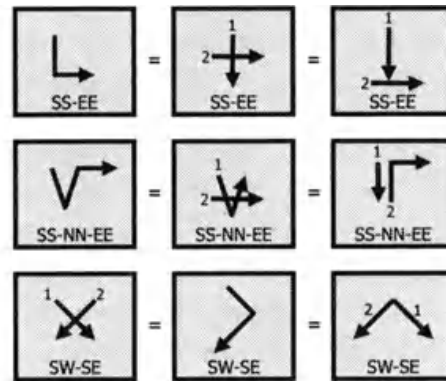
It is worth noting that this method performs very badly when given a stroke which resembles a curve. Because a curve is a sequence of lines which continuously changes direction, our method would calculate that the whole curve is noise, and thus would not be able to recognize it.

Having removed the noise, we then reduce adjacent appearances of a given direction to just one occurrence. At this point, we are left with a ‘signature’ that looks, for example, like “SS, WW, NW”. Using the signature that we have derived from the stroke, we can execute predefined operations. Some sample strokes along with their signatures are shown in Figure 2.



Note that strokes like NN-SS and EE-WW-EE do not need to form an angle, but are shown like this for illustration purposes.

Figure 2: Some strokes and their signatures.



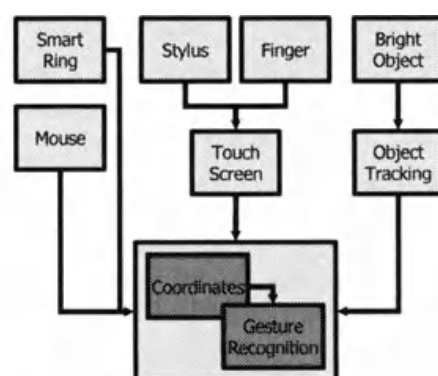
For any possible stroke, users may decide to break up the stroke into any number of sub-strokes, and then perform each sub-stroke independently, regardless of its relative position to the rest of the sub-strokes.

Figure 3: Different strokes with identical signatures.

4.2 Multi-stroke Gestures

The next development of our method was to recognize gestures that consist of more than one stroke. In order to allow users to perform multi-stroke gestures, the GUI has to allow for a short ‘timeout’ period, in which the user is able to stop drawing a stroke and start drawing a new stroke. For example, in the case of pen input, the user would be able to draw a stroke, lift the pen, and within the timeout period start drawing a new stroke. In this case, a special symbol may be used in the input stream to denote that the pen was lifted.

Having allowed for gestures consisting of more than one stroke, we have introduced an interesting characteristic. Now, different gestures may map to the same signature. Thus, a signature and, in turn, an operation can have more than one way of being accessed. For example, a stroke that looks like L and a gesture that looks like a cross may have the same signature, as shown in Figure 3.



Any method and input technique that can produce a meaningful set of coordinates may be used with our stroke recognition technique.

Figure 4: Stroke recognition used with various techniques.

This sort of flexibility is beneficial when users may be working with multiple devices, each with different form factors and characteristics. In the case where screen size is limited, users may choose to decompose a gesture as finely as they wish, even into separate single-line strokes. The single-line strokes may be performed on top of each other, thus requiring less space on the screen. On the other hand, users with enough space may choose to perform one long composite stroke in order to save time.

It may be argued that the number of possible operations is limited when many gestures are mapped to the same signature. Although this is true, we believe that in many cases the flexibility provided by our method outweighs the need for a plethora of different operations. The optimum solution is probably to allow the user to choose between different gestures mapping to the same or different signatures, perhaps according to the visual similarity of gestures as described in [Long et al. 2000]. Rubine [1991], for example, demonstrates an approach to training a system for various gestures. This is a focus of our ongoing work.

5 Real-time Video Capture of Hand Movements as Input

In principle, our stroke recognition method deals with pure coordinates and nothing more. Therefore, any input technique could work with our stroke recognition method, as long as there is a meaningful way of deriving a set of coordinates from the input technique. This supports our aim of developing a flexible interaction technique that can be used across multiple devices and platforms (Figure 4). For example, in the converging world of mobile and pervasive computing, our user may at one moment wish to interact with her PDA using a common set of gestures and in the next moment move seamlessly to interacting with the wall display beside her using the same set of gestures. At one moment the PDA provides the interaction area on which the gestures are made using a stylus; in the next moment, the PDA itself becomes the ‘stylus’ as our user makes the gestures in the air with the PDA while interacting with the wall display. As a proof of principle, we implemented a real-time object tracking

technique that we then used along with our stroke recognition algorithm as an input technique.

For our prototype we implemented an algorithm that performs real-time object tracking on live input from a Web camera. The user can select a specific object by sampling its colour, and the algorithm tracks this object in order to generate a series of coordinates that describe the position of the object on the screen or, to be precise, the position of the object relative to the camera's view. We then pass these generated coordinates to our stroke recognition algorithm, which then proceeds with the recognition of the strokes.

Due to the characteristics of our stroke recognition method, the coordinates may be supplied at any rate. So long as this rate is kept steady, the stroke recognition is very successful. Thus, despite the fact that our object tracking algorithm is not optimal, it still provides us with a useful prototype.

Object recognition is performed using HSL (Hue – Saturation – Luminosity) sampling. The object to be tracked is described in terms of HSL based on its colour. We then apply a varying threshold of approximately 5% to the live video input, which results in certain pixels being identified as belonging to the object. We then perform a second pass in order to identify which region has the highest density of object pixels, and from this we derive the object's centre. These coordinates are then passed on to the stroke recognition algorithm.

An issue that we had to address was how to allow for an act corresponding to lifting the stylus from a touch screen. Our initial approach has been that the user can, for example, hide the object within the palm of her hand. This does not limit functionality in terms of stroke recognition — remember that any signature can be performed as one long stroke — and contributes to the similarity of input methods across platforms and devices. A potentially less cumbersome solution to this would be to allow for an 'invisible' light, such as infrared, to be emitted from a small handheld object and to be used by the object-tracking camera. Such an object could be a dedicated input device for interacting with pervasive computing facilities and could emit light when squeezed or held at a certain orientation. Alternatively, and more in line with our general vision of integration across mobile and pervasive devices, a PDA or other device that can provide an input area (e.g. a touchscreen) for gestures could also act as a 'stylus' by emitting infrared on user demand. This would allow for seamless transitions, using a common gesture alphabet, between interacting with a mobile device and interacting with surrounding pervasive computing devices. We are currently developing this functionality as part of our ongoing research.

6 Conclusions and Future Work

We have incorporated the stroke recognition method described here in a test harness application. This has given encouraging results. For testing purposes, the stroke recognition method was implemented as a library in C++, and was used by an application written in Microsoft Visual C++. The testing platforms were:

- A standard desktop PC with conventional mouse input.
- A Xybernaut MA V wearable computer with touchscreen and pen input.

- A PC with 61 inch plasma display and camera picking up hand input.

The results have been very positive. For mouse and pen-based input the recognition rate peaked for a threshold of about 9%. For multi-stroke gestures, a timeout of 800 milliseconds produced the best results. Gestures consisting of up to five single-line strokes were readily recognized. The real-time video capture algorithm also produced encouraging results. We tested the method on both standing users with the camera facing them, and on sitting users with the camera above them facing down to their table. The main problem we faced was the speed at which objects were recognized and coordinates generated. We used a bright yellow tennis ball as our testing object. With the current implementation, the users had to move the tracked object quite slowly, because fast movements were not processed well. On average, for video input of dimensions 320×240 pixels, one frame every 200ms was processed. Although the strokes were recognized, it was frustrating for the users to move their hands at such speeds.

The main bottleneck in the existing implementation is rather naïve processing of the camera input. We have drawn in colleagues with expertise in computer vision in order to optimize this. Our collaboration with them should lead to improvements in usability by removing the current need for relatively slow hand gestures when using camera-based object tracking.

For further improvements at the subsequent stage of stroke recognition, we are developing a mechanism for adjusting the threshold for every sub-stroke, in order to increase the recognition rate. This will also allow for more sub-strokes to be recognized.

Our ongoing work includes developing and evaluating a range of applications which use the stroke recognition method. Usability evaluation issues include the question of mapping multiple gestures to one or more signatures. Further work will evaluate alternative strategies for this mapping, including user-definable strategies.

The stroke recognition method is usable on small devices with limited processing capabilities and small input areas. Typical current PDAs stand at the upper end of this scale. The method is of potential value for even smaller devices, such as rings, buttons and smart cards, for example. Moving in the opposite direction in scale, the method can also be applied to live graphic input and on wall-sized display areas. As we have demonstrated, a user may hold an object or even attach it to her fingers as, for example, a ring, in order to perform strokes in the air, on a table or on an interactive surface [cf. Rekimoto 2002].

Any other way of providing a set of coordinates can also be used with our stroke recognition method. To expand the range of input options in a pervasive computing environment, we are exploring the generation of coordinates by users' hand gestures without an object acting as a 'stylus'. To this end, we are beginning research with colleagues in sports science to develop alternative input methods using body tracking technologies. Thus, our stroke recognition method has great potential value in exploring and developing common interaction techniques that will allow us successfully to take the human-computer interface out into our streets and wider lives.

References

- Brewster, S. A. [2002], Overcoming the Lack of Screen Space on Mobile Computers, *Personal and Ubiquitous Computing* **6**(3), 188–205.
- Brewster, S. A., Lumsden, J., Bell, M., Hall, M. & Tasker, S. [2003], Multi-modal “Eyes Free” Interaction Techniques for Wearable Devices, in G. Cockton & P. Korhonen (eds.), *Proceedings of CHI'03 Conference on Human Factors in Computing Systems, CHI Letters* **5**(1), ACM Press, pp.473–80.
- Callahan, J., Hopkins, D., Weiser, M. & Shneiderman, B. [1998], An Empirical Comparison of Pie vs. Linear Menus, in M. E. Atwood, C.-M. Karat, A. Lund, J. Coutaz & J. Karat (eds.), *Proceedings of the CHI'98 Conference on Human Factors in Computing Systems*, ACM Press, pp.95–100.
- Guimbretiere, F. & Winograd, T. [2000], FlowMenu: Combining Command, Text and Data Entry, in M. Ackerman & K. Edwards (eds.), *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology, UIST2000, CHI Letters* **2**(2), ACM Press, pp.213–7.
- Guimbretiere, F., Stone, M. & Winograd, T. [2001], Fluid Interaction with High-resolution Wall-size Displays, in J. Marks & E. Mynatt (eds.), *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology, UIST2001, CHI Letters* **3**(2), ACM Press, pp.21–30.
- Igarashi, T., Edwards, W. K., LaMarca, A. & Mynatt, E. D. [2000], An Architecture for Pen-based Interaction on Electronic Whiteboards, in S. Levialdi, V. di Gesu & L. Tarantino (eds.), *Proceedings of the Working Conference on Advanced Visual Interfaces*, ACM Press, pp.68–75.
- Long, A. C., Landay, J. A., Rowe, L. A. & Michiels, J. [2000], Visual Similarity of Pen Gestures, in T. Turner & G. Szwillus (eds.), *Proceedings of the CHI'00 Conference on Human Factors in Computing Systems, CHI Letters* **2**(1), ACM Press, pp.360–7.
- Pirhonen, A., Brewster, S. & Holguin, C. [2002], Gestural and Audio Metaphors as a Means of Control for Mobile Devices, in D. Wixon (ed.), *Proceedings of CHI'02 Conference on Human Factors in Computing Systems: Changing our World, Changing Ourselves, CHI Letters* **4**(1), ACM Press, pp.291–8.
- Rekimoto, J. [2002], SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces, in D. Wixon (ed.), *Proceedings of CHI'02 Conference on Human Factors in Computing Systems: Changing our World, Changing Ourselves, CHI Letters* **4**(1), ACM Press, pp.113–20.
- Rubine, D. [1991], Specifying Gestures by Example, in J. J. Thomas (ed.), *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, ACM Press, pp.329–37.
- Sutherland, I. [1963], Sketchpad: A Man–Machine Graphical Communication System, in *Proceedings of the Spring Joint Computer Conference*, IFIP, pp.329–46.
- Zhao, R. [1993], Incremental Recognition in Gesture-based and Syntax-directed Diagram Editors, in S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel & T. White (eds.), *Proceedings of INTERCHI'93*, ACM Press/IOS Press, pp.95–100.