A new approach to avoiding the local extrema trap

A. McCabe[1] J. Trevathan[2]

(Received 30 August 2006; revised 4 July 2007)

Abstract

The Extremum Consistency algorithm avoids local maxima and minima in a specialised domain. The most notable difference between this approach and others is that it places a greater importance on the width or consistency of an extremum than on its height or depth (amplitude). Short term, high amplitude extrema are encountered in many typical situations (such as noisy environments or due to hardware inaccuracies) and cause problems with system accuracy. The Extremum Consistency algorithm is far less susceptible to these situations than hill climbing, convolution, thresholding, and tends to produce higher quality results. We describes the algorithm and present results from practical experimentation, which illustrates its superiority over other forms of local extrema avoidance in three real world applications.

See http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/110 for this article, © Austral. Mathematical Soc. 2007. Published October 7, 2007. ISSN 1446-8735

Contents

1 Introduction **C475**

2 The problem domain—motivation **C476**

3 The algorithm **C480**

4 Successful applications **C485**

4.1 Direction based handwritten signature verification C485

4.2 Velocity based handwritten signature and password verification C489

4.3 Physiology research—tracking fluctuations in infant face temperature C490

5 Conclusions **C490**

References **C491**

1 Introduction

This article presents a new algorithm for avoiding local maxima and minima in specialised environments. The embedding of this algorithm in a number of practical systems results in significant improvements in accuracy when compared with the use of other classical local extremum avoidance algorithms.

Throughout this article the discussion tends to focus on minima (also known as valleys or troughs) in the interests of brevity. Discussions can trivially be adapted to avoidance of maxima or peaks. Additionally, chiefly for convenience, this algorithm has been named Extremum Consistency (EC). The meaning behind the name is made clear later in the article.

The underlying problem lies in deciding whether a given extremum represents a true extremum. This is not a new problem to computer science, but it has never been approached in the particular way described here.

There are several existing algorithms for avoiding local extrema; however, none have proven to be sufficiently effective in the specific domain outlined in Section 2. The most appropriate of these existing algorithms were implemented in complete systems and the results are presented below.

Section 2 describes the problem domain and motivation for the algorithm, Section 3 presents the algorithm, and Section 4 outlines some of the specific software applications where the algorithm has been successfully applied. Section 5 presents the concluding remarks and directions for future work.

2 The problem domain—motivation

This problem domain is based on iterative improvement strategies that attempt to find maxima or minima while 'traversing' or 'stepping' along a certain stream.

Abstractly, what is required in this domain is for an algorithm to move in the direction of decreasing value until a minimum is reached. Once there, the minimum's location is recorded and, depending on the application, it either proceeds in the opposite direction looking for a corresponding maximum, or 'jumps' out of the minimum and begins the process again. This search continues until all of the minima (and maxima if required) are mapped and then processing of these points is performed (see Section 4 for more information on specific applications). Because of this pattern of searching, any false (in a sense, local) minimum encountered will adversely affect the performance of the system, not only because the recorded position is incorrect, but starting the search for the subsequent extremum too early may propagate that error.

The main problem therefore lies in detecting the *true* (or global) extrema

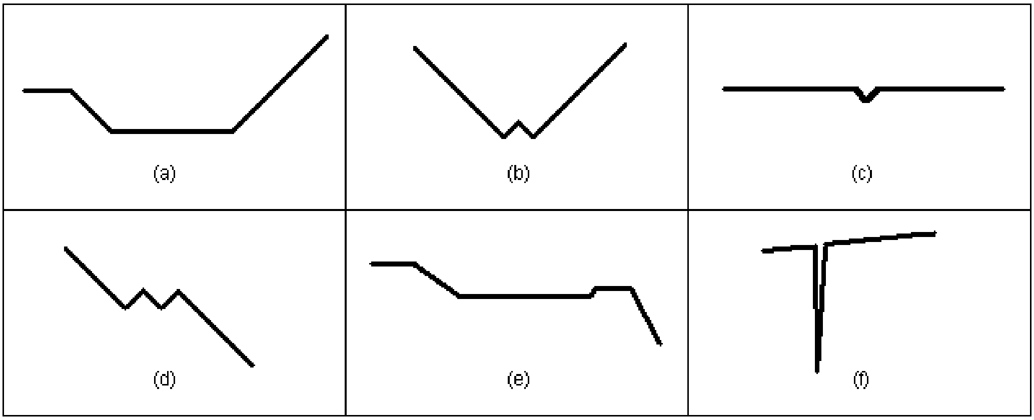


Figure 1: This figure represents some local minima situations that are typically encountered in processing the input stream. The horizontal axis represents increasing time and the vertical axis represents any of various stream types such as velocity, direction or temperature. Specifically, (a) contains a valid minimum, (b) contains only a single valid minimum (the appearance of two minima is caused by some noise in the stream, so the second should be ignored) and no others contain any true minima, according to our definition (the minima in (c) and (f) are probably a result of noise in this environment, rather than genuine minima, and the same could be said for (d) and (e)). An effective algorithm should reflect this.

in a volatile and often noisy environment. The EC algorithm is a specific approach to deciding whether a particular extremum represents a true extremum. Figure 1 pictorially shows some of the situations typically encountered in this environment. The horizontal axis in these diagrams represents time and the vertical axis represents various observations such as velocity, direction or temperature.

The initial motivation for this algorithm came when developing signature verification software [2]. The approach was based on detecting the order of 'turning points' (in other words, minima and maxima) in the pen tip direction and processing the locations of those turning points. The EC algorithm was implemented in this application with significant improvements in system accuracy. See Section 4 for more information on this and other applications.

The challenge for this and similar algorithms is to ignore meaningless small fluctuations that appear in the stream, while recording the *meaningful* fluctuations. The problem lies in distinguishing between the two. Local extrema can enter the data as a result of various aspects, such as quantisation noise, rounding problems (discussed below) or, in handwriting based applications (discussed in Section 4), something as simple as shaky hands or the writing surface itself. These local extrema can be of varying scale, making them more difficult for conventional algorithms to overcome.

The aforementioned rounding problem occurs due to the discrete resolution of hardware such as graphics tablets or pen based computers used to capture handwriting. The problem occurs when the actual handwriting path travels directly between two neighbouring pixels. The tablet must 'round' the pen tip location to the nearest pixel. Occasionally, slight variations in pen tip pressure cause the rounding to be done to a different pixel. Figure 2 shows an example of this. The resulting handwriting path then looks (to the system) like that shown in Figure 1(b) or Figure 1(c).

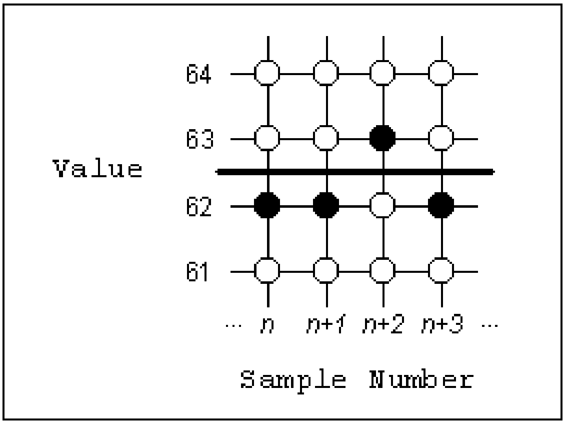


Figure 2: Situations like this are the result of the discrete resolution of the hardware used to capture a stream. The bold horizontal line represents the actual position of the data and the black dots represent the recorded values. Time is represented on the horizontal axis. This situation typically arises when the hardware is a graphics tablet or pen-based computer that rounds the position of the pen tip to the nearest pixel, but also arises with (say) temperature observations when the actual temperature is rounded to the nearest tenth of a degree for recording.

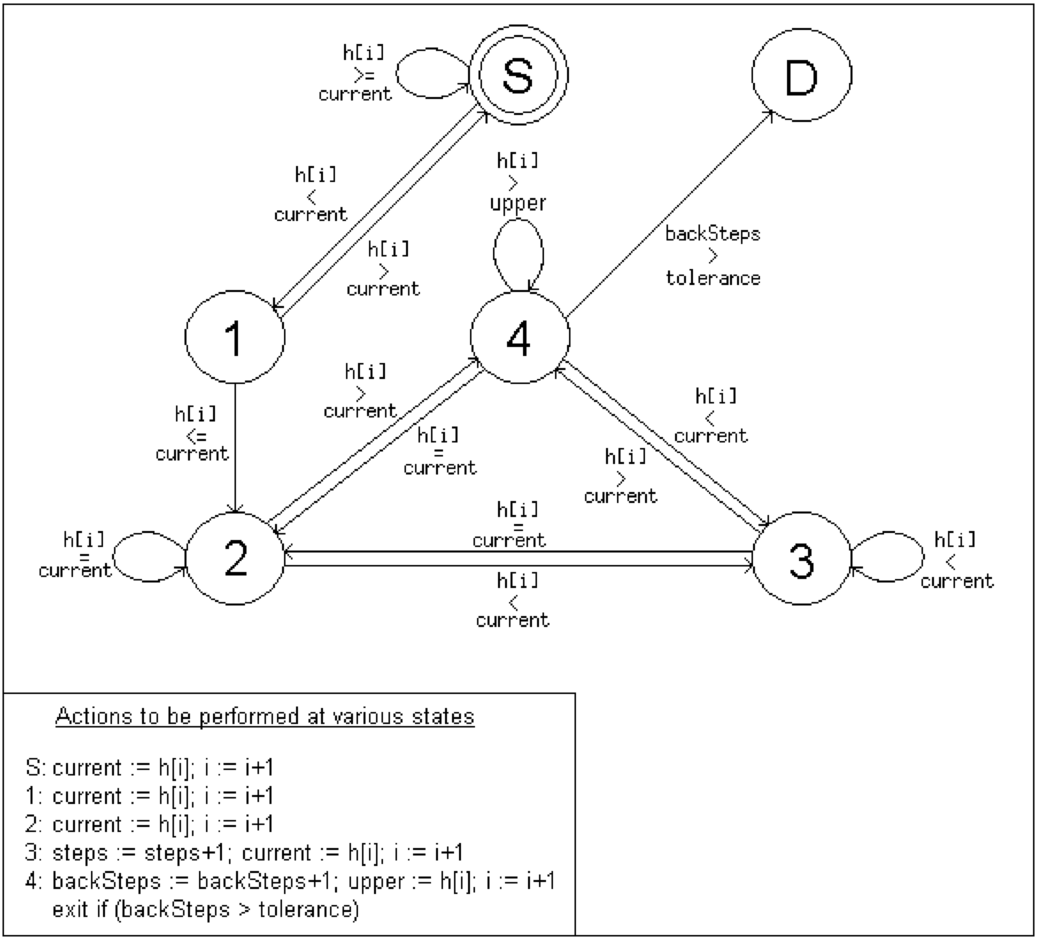
3 The algorithm

The algorithm is compact, requiring very little stored data (four integers), no search tree, and is implemented via a series of comparisons done while traversing the surface of the feature space. Additionally, the algorithm is only executed when a potential minimum is encountered so the effect on the efficiency of the overall system is slight.

The major difference between this algorithm and others is that it examines, primarily, the *width* (or perhaps more accurately the *consistency* or *duration*) of the minima. Most other algorithms (such as convolution and thresholding) place more emphasis on the *depth* of the minima. The use of the term *width* here differs slightly from an intuitive understanding of the width of a valley. The width of a valley is best explained by considering the initial valley downslope and the following upslope separately. It is defined as the number of 'steps' encountered in its traversal where a step, in a downslope, refers to a decrease in height below the value of the current minimum. The more of these decreases there are, the larger the number of steps. Once these two values are found, the width of the valley is defined as the *minimum* of the individual width values.

Figure 3 presents a finite state machine illustrating the EC algorithm's operation. The remainder of this section contains descriptions to accompany the illustrations.

Figure 4 illustrates an example of step and width calculation. Steps are defined as movements in the direction of a particular extremum—for example, in Figure 4 the movement between Time = 0 and Time = 1, Time = 1 and Time = 2, Time = 3 and Time = 4 and Time = 5 and Time = 6 each constitute a single step. The term 'backward steps' is now also defined as movements *away* from the extremum, beyond the current maximum. For example, in Figure 4, the movement between Time = 4 and Time = 5, Time = 6 and Time = 7 as well as between Time = 9 and Time = 10 can



Actions to be performed at various states

```
S: current := h[i]; i := i+1
1: current := h[i]; i := i+1
2: current := h[i]; i := i+1
3: steps := steps+1; current := h[i]; i := i+1
4: backSteps := backSteps+1; upper := h[i]; i := i+1
   exit if (backSteps > tolerance)
```

FIGURE 3: A finite state machine expressing the EC algorithm for finding the 'width' of, or number of 'steps' in, the initial downslope of a valley. The movements between vertices (states) are defined by the comparison between points in the input stream and the comparisons are included on the edges in the diagram. Additionally there are actions to be performed when some vertices are reached—these are also included in the diagram. Note that in this table, $h[i]$ refers to the ith element in the list of stream values h. Once the `backSteps` parameter exceeds the `tolerance`, the algorithm enters the dead state D and we have the width of the slope in the `steps` parameter. The width of the upslope is similarly calculable, with the inversion of various state transition conditions and the width of the valley itself is then the minimum of the downslope width and upslope width. Likewise, the width of peaks can be found with minimal modifications to the algorithm.

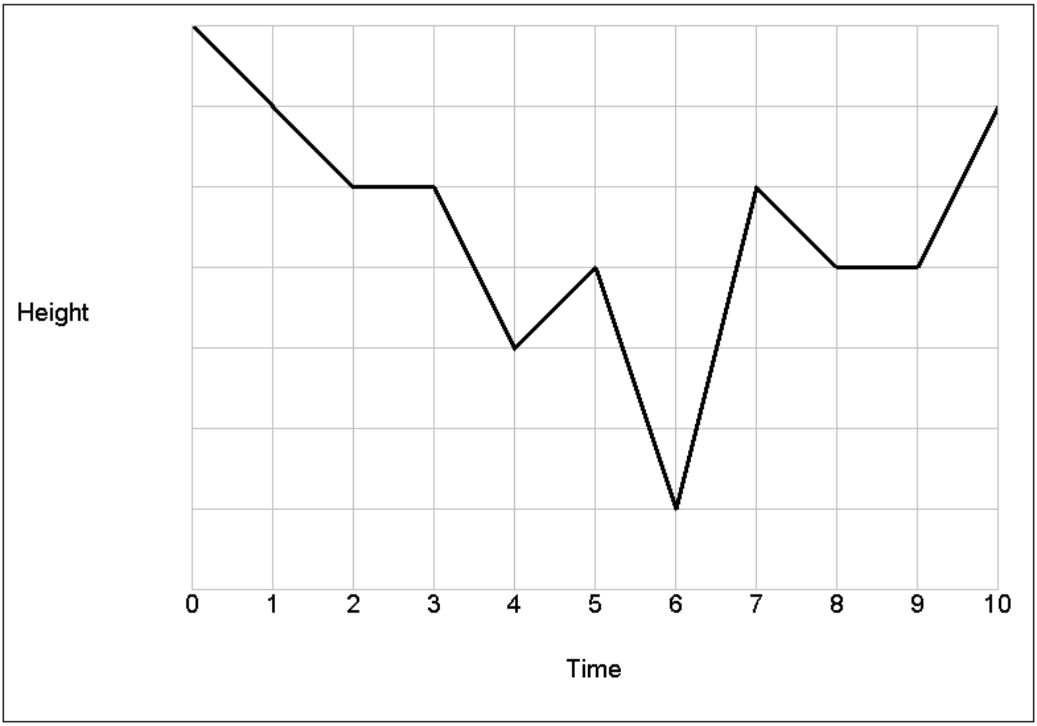


Figure 4: An illustration of step and width calculation. Valid steps occur between time points 0 and 1, 1 and 2, 3 and 4, and 5 and 6. Backward steps occur between time points 4 and 5, 6 and 7, and between 9 and 10.

be thought of as taking a backward step. A tolerance parameter determines how many backward steps are accepted before the algorithm terminates.

Upon termination we have the value for the slope's width. Tolerance then becomes a significant factor as it represents the amount of time spent looking for a 'better' minimum before giving up and accepting the one we have. If the minimum's location is the only information sought (as is often the case, depending on the application), the goal has been achieved and the algorithm ends. However, if the width of the entire valley is sought (as in the application described in Section 4.1) then the search for the top of the post-valley upslope takes place, starting from the valley floor.

Recall that the width of the entire valley is taken as the minimum of the width values of the downslope and the upslope. It is important to take the minimum because otherwise a very large downslope with a very small upslope would erroneously appear as a very large valley. For example, see Figure 5.

With the width of the valley obtained it is then just a matter of setting a threshold on which width sizes will be considered large enough to constitute a genuine valley (and similarly for peaks). The calculation of the optimal threshold was done during a training phase via brute force experimentation for each application. Results typically worsened considerably for a threshold value of five or over and for all applications in Section 4 a threshold value of just two proved to be globally optimal.

The final phase in experimentation involved taking both an extremum's height/depth and width value into account, in an attempt to obtain an even more accurate estimate of true extrema. The simplest and most successful combination method was to simply take the product of the valley width and depth (or height). The result of this was that deeper valleys were now considered 'better' minima than shallow valleys with similar width, which intuitively seems a more desirable effect. In the experiments conducted, this approach consistently provided the best overall results.

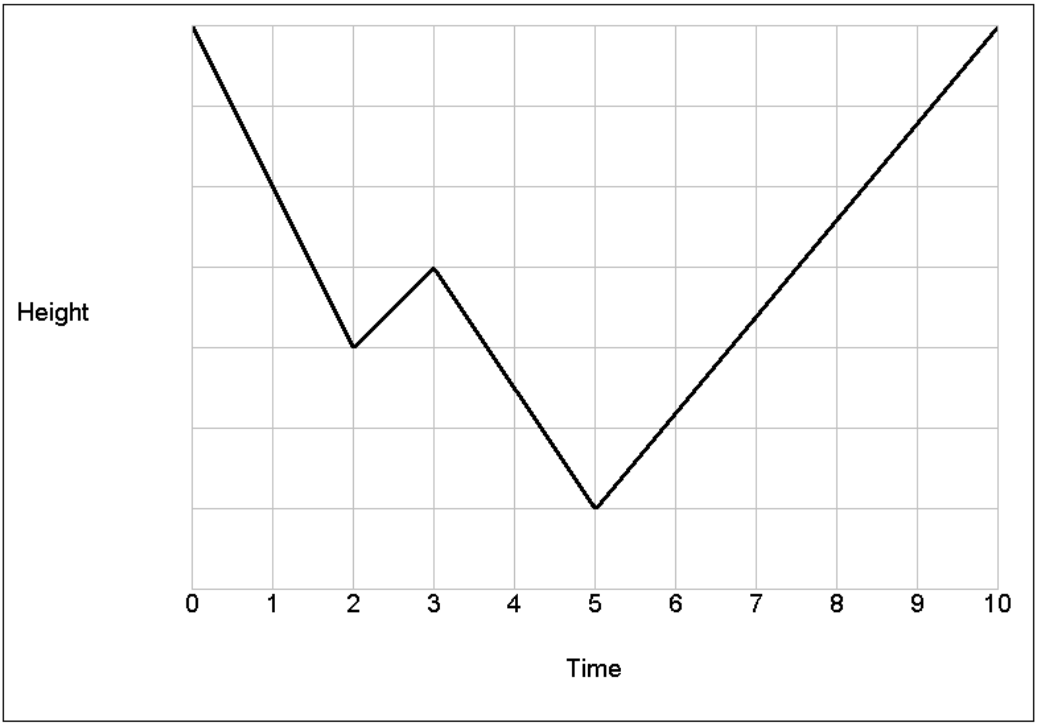


Figure 5: An illustration of a large downslope with a small upslope appears in the valley at Time $= 2$. In this case the valley itself is actually very shallow and using the maximum or mean slope size would make the valley appear erroneously large - the use of the minimum slope size is much more accurate. By placing a threshold on the width, this valley will ideally be ignored and that at Time $= 5$ would be taken as the more appropriate minimum.

Section 4 describes a number of projects in which the EC algorithm has been successfully implemented. Subsection 4.1 presents a detailed comparison of the accuracy and execution speed of this algorithm versus other approaches such as convolution.

4 Successful applications

It is worth noting at this point that the algorithm presented in Section 3 is used as a pre-processing filter, the output of which serves as input to another stage (for example, as part of a signature verification system). The only real way that success of the EC algorithm is measured is by evaluating the success of the resulting application as a whole. This section discusses some of the application areas in which the EC algorithm has been successfully employed.

4.1 Direction based handwritten signature verification

This was the first project to benefit from the application of the EC algorithm as it relies heavily on extrema detection. It involved the design and development of a dynamic signature verification system [2]. At its most basic level this system tracked the direction of the pen tip when performing a signature. Peaks and valleys (maxima and minima) were detected in both the horizontal and vertical directions, ordered and converted into a character string.

The initial attempt at valley detection was to implement a naive gradient descent algorithm that simply traversed in the direction of non-increasing value until the point where a greater value was encountered. This previous (lowest) point was then deemed to be the minimum (note that the surface is only one dimensional, so there is no choice as to which direction to take when traversing). The problem with this approach was that noise periodically appeared in the stream producing false extrema. Additionally there was a problem with the rounding of the pen tip location to the nearest pixel in the hardware device which also produced false valleys and peaks (see Figure 2). This often resulted in the character string becoming somewhat mis-representative of the signature, degrading the whole system's effectiveness.

The overall error rate (sum of false rejection and false acceptance rates) for the first version of the system was 6.9%. This improved dramatically to 2.9% with the implementation of the EC algorithm to better detect the valleys and peaks. Specifically the false rejection rate (the proportion of genuine signatures rejected as forgeries) was improved from 4.3% to 0.9% and the false acceptance rate (forgeries accepted as genuine signatures) was improved from 2.6% to 2.0%. As mentioned in Section 3, the final phase of EC involved combining the width of an extrema with its 'amplitude' (height/depth). The purpose of this being to assign a higher importance to extrema with higher amplitude, versus lower amplitude extrema of similar width. The best combination method was to take the product of the two values, which resulted in a significant improvement of the overall error rate to 2.3%.

In other attempts to improve the error rates, two more algorithms were implemented: simple removal of small valleys or peaks (called 'thresholding') and basic convolution of the data prior to gradient descent/hill climbing. Thresholding simply involved determining the distance (in pixels) between the peak's location and the preceding valley's location (that is, the depth of a valley or height of a peak). If this distance was below a specified threshold, then that peak was ignored and the traversal continued in the same direction.

Examples of situations where thresholding was successful are shown in Figure 1(b), (c) and possibly (d). However, the algorithm's overall performance (in terms of error rate) was quite poor. The reason is that a valley's *depth* alone, while containing useful information, is not the best validity indicator (at least in this environment), but rather the consistency/duration is more important.

Simulated annealing is an approach which avoids local extrema by 'jumping ahead' some random distance when an extremum is encountered, to try to find a 'better' extremum [7]. While this is very effective in some domains, it was envisaged that simulated annealing would not work well in this environment. This was because there are often long periods in which the stream value remains the same (plateaus—see Figure 1(e)), which can vary greatly in their duration. Small jumps ahead will work on many occasions, but not in situations such as this. Large jumps ahead will work in many situations also, but will tend to jump over smaller details. In the interest of experimentation, a modified simulated annealing algorithm was implemented [7]. The number and size of jumps were limited by threshold values, which were optimised through trial and error. The unsuitability of this approach was reflected in the poor error rate of 24.0%.

Convolution is considered one of the most useful method of 'smoothing out' or 'averaging' one-off 'bumps' or random noise while attempting to preserve those extrema which are truly indicative of the pen tip direction. The basic idea behind convolution is that a window of some finite length (convolution matrices are possible in environments of higher dimensionality) is scanned across the stream of values [1]. The output pixel is the weighted sum of the input pixels within the window where the weights can be adjusted to perform various filtering tasks—when smoothing is performed the weights are generally all equal.

After the input stream was convoluted the hill climbing/gradient descent approach was used to obtain the extrema. Multiple attempts were made with convolution using window sizes varying from one (the trivial case) up to fifty, with the optimal window size (that which resulted in the lowest error rate over the entire signature database) found to be five. Experiments were also conducted with the number of iterations of convolution performed, allowing for the possibility that the first convolution run did not smooth out all of the irrelevant extrema and further iterations were necessary. The best overall

Table 1: Error rates using various methods of overcoming false extrema in a specific signature verification environment. If there are parameters involved in the operation (such as convolution window size) then the parameters producing the lowest error rate were used to generate the results.

Technique	Error Rate
Simple Hill Climbing	6.9%
Thresholding	17.0%
Simulated Annealing	24.0%
Convolution and Hill Climbing	5.3%
EC	2.9%
Convolution and EC	3.4%
EC (width $\times$ height)	2.3%

results were obtained after a single iteration of convolution with the results progressively deteriorating with further iterations, indicating that some of the true extrema were being incorrectly smoothed out.

There was also some experimentation with combining convolution with the EC algorithm. This involved using convolution to smooth the input stream before applying EC to detect the extrema. This approach proved to be more successful than convolution with hill climbing but less successful than EC alone. The reason for this is most probably related to convolution smoothing out small but meaningful extrema.

Table 1 summarises the error rates of the implemented approaches used in the signature verification system. The EC algorithm is clearly superior in this environment.

The other advantage of the EC algorithm over convolution is execution speed. In a real time application like signature verification, execution speed becomes a serious issue. In order to perform convolution an entire extra layer of computation is required, as convolution of the raw data must be done prior to obtaining the extrema, whereas with the EC algorithm the checking is done at the same time as the search for extrema. Additionally, convolution can become quite expensive using a large window or with a large raw data size (for example, a typical data size in the application described in Section 4.3 is over 12,000 entries).

Empirical experimentation with the signature verification system has found that convolution causes an average slowdown of 20–25% (depending on system parameters). The number of extra calculations required in the EC algorithm compared to naive hill climbing is almost negligible with the slowdown of the signature verification system experimentally found to be less than 3%.

4.2 Velocity based handwritten signature and password verification

The EC algorithm was also used in a handwritten password verification system [3]. The first step in this system (as well as many other handwriting based systems like character recognition algorithms [5]) was to segment the writing stream into its conceptually significant or constituent parts, commonly known as *strokes*. The strokes are continuous 'pen down' segments of writing bounded by consecutive minima in the pen tip velocity. The approach then is to extract properties of these strokes and model these properties using a hidden Markov model or neural network.

An approach along these lines was presented previously in McCabe [3], which also made successful use of the EC algorithm. The most naive method of obtaining the velocity minima is a basic gradient descent algorithm. This was seen as an obvious application for the EC algorithm and it was implemented immediately. A simple gradient descent implementation was also performed for comparative purposes. Simple gradient descent produced a total error rate of 2.7% for password verification compared with 0.79% using EC with width only, and 0.64% when using EC with the product of width and height to do the segmentation. A similar EC implementation was also used as a method of signature segmentation in recent extensive studies [4, 8].

4.3 Physiology research—tracking fluctuations in infant face temperature

This physiological research project involved examining fluctuating infant facial temperatures and detecting the exact location of temperature maxima [6]. It is theorised that a climax of increasing facial temperature closely correlates with other physiological episodes. Our task was to accurately determine the occurrence of these maxima in real time.

Initially developed software implemented a simple hill climbing approach to determine the location (in time) of the temperature maxima. These times were correlated with the nearest occurrence of a particular physiological episode and the correlation value, or p-value, was 0.072. That is, the relationship was not significant.

The EC algorithm (width $\times$ height) was also implemented to detect the temperature maxima. The same correlation calculation method was used and the p-value improved to 0.001 (highly statistically significant). These results show that the EC algorithm provides a more accurate estimate of the true temperature maxima.

5 Conclusions

This article presents a novel local minima and maxima avoidance algorithm. The EC algorithm examines an extrema's *width* or *consistency* more so than the actual height. Short term peaks/valleys can be encountered in many situations (for example, noisy environments/hardware inaccuracies) and can cause system accuracy problems. The EC algorithm is far less susceptible to these anomalies than existing techniques and tends to produce higher quality results.

The EC algorithm has shown that it can be effective in various practical iterative improvement environments. This article discussed several specific large scale applications, and performed a comparison between the EC algorithm and hill climbing, convolution, thresholding and simulated annealing. The EC algorithm is a notable improvement over the other approaches in all of the explored applications.

Future work involves comparing the algorithm's effectiveness to other optimisation techniques such as genetic algorithms. Furthermore, the possibility of adapting the algorithm to multi-dimensional space will be explored. Should this be successful, it would be interesting to examine its utility in traversing error surfaces for more effective neural network training. We are also applying the EC algorithm to software bidding agents to enable correct measurements of maximum and minimum prices.

References

[1] Hirschman, I. and Widder, D. *The Convolution Transform.* Dover Publications, 2005. C487

[2] McCabe, A. Implementation and Analysis of a Handwritten Signature Verification Technique. In Proceedings of the *International Conference on Security and Cryptography*, (to appear), 2007. C478, C485

[3] McCabe, A. Markov Modelling of Simple Directional Features for Effective and Efficient Handwriting Verification. R. Mizoguchi and J. Slaney (Eds.): *PRICAI*, LNAI 1886, pp.801(1–12), 2000. C489

[4] McCabe, A. Handwritten Signature Verification Using Complementary Statistical Models. *Ph.D. Thesis*, James Cook University, 2004. C490

[5] Plamondon, R. and Srihari, S. On-line and Off-line Handwriting Recognition: A Comprehensive Survey. *TPAMI*, Vol. 22, No. 1, pp.63–84, 2000. C489

[6] Russell, M. J. and Vink, R. Increased Facial Temperature as an Early Warning in Sudden Infant Death Syndrome. In *Medical Hypotheses*, Vol. 57, No. 1, pp.61–63, 2001. C490

[7] Russell, S. and Norvig, P. *Artificial Intelligence—A Modern Approach (2nd Edition)*. Prentice Hall, 2002. C487

[8] Trevathan, J. and McCabe, A. Remote Handwritten Signature Authentication. In Proceedings of the *2nd International Conference on E-Business and Telecommunication Networks*, pp.335–339, 2005. C490

Author addresses

1. **A. McCabe**, Fern Computer Services Pty Ltd., Belfast, Ireland.
   mailto:alan@mymait.com

2. **J. Trevathan**, School of Mathematics, Physics and Information Technology, James Cook University, Australia.
   mailto:jarrod.trevathan@jcu.edu.au