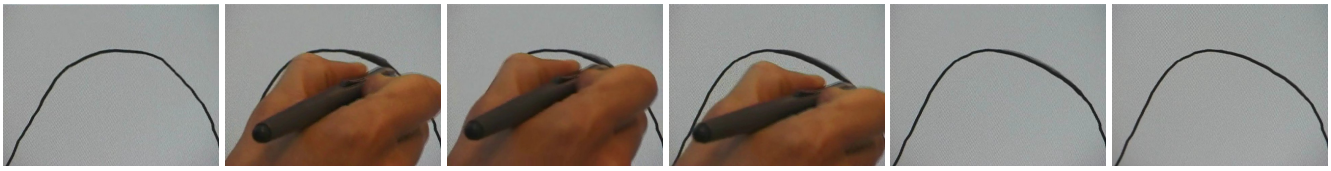
Sketching with a Low-latency Electronic Ink Drawing Tablet

Alex Henzen Neculai Ailenei*
Philips Emerging Display Technologies
Jan Campertstraat, 5
Heerlen (the Netherlands)

Fabian Di Fiore Frank Van Reeth†
Hasselt University
Expertise Centre for Digital Media
transnationale Universiteit Limburg
Wetenschapspark, 2
BE-3590 Diepenbeek (Belgium)

John Patterson‡
Dept. of Computing Science
Glasgow University
17 Lilybank Gardens
Glasgow G12 8QQ (Scotland, UK)



(a) (b) (c) (d) (e) (f)

Figure 1: Shows a curve as drawn by the nervous hand tool. Shot (a) showing the original curve. Shots (b–d) show the artist as s/he modifies another corner of the curve, dragging this corner inwards whilst also smoothing it out. Shot (e) shows the curve as the old curve begins to fade with shot (f) showing the final lie of the curve.

Abstract

Drawing on paper is an experience which is still unmatched by any input device for drawing into a computer in terms of accuracy, dexterity and general pleasantness of use. This paper describes a paper-like drawing tablet which uses electronic ink as its output medium with stylus-based touchpanel input. The device mimics the experience of drawing in a manner which can be adjusted to approach the feel of different kinds of paper. We discuss further some basic issues which need to be addressed in managing interfacing to such a device, specifically the avoidance of the legacy of mouse-oriented point-and-click interfaces which have influenced GUI design for so long. We see a sketch-based model for interaction, based on free-form curve drawing, as being the way forward but new interaction models are required. The tablet is initially intended to serve as an input-device for cartoon drawing and editing, so the product of any sketching process has to be presented to the rest of the animation data-path in terms of a conventional curve model, here Bézier chains. We discuss models for achieving this without having to resort to legacy curve-editing techniques which have no counterpart in drawing on paper or in the repertoire of the traditional animator. Potential uses of these interaction techniques go well beyond supporting the cartoon drawing application.

CR Categories: I.3.1 [Computer Graphics]: Hardware Architecture—Input devices; I.3.3 [Computer Graphics]: Picture/Image generation—Line and Curve Generation; I.3.6 [Computer Graphics]: Methodology and Techniques—Ergonomics; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques

Keywords: Input devices, Line and Curve Generation, Interaction Techniques, Electronic Ink

*e-mail: {alex.henzen, neculai.ailenei}@philips.com
†e-mail: {fabian.difiore, frank.vanreeth}@uhasselt.be
‡e-mail: jwp@dcs.gla.ac.uk

1 Introduction

Capturing drawings for computer processing has always been problematic, especially for sustained work as is typical in cartoon animation. There are basically two routes, scanning and direct input. Scanning involves drawing on paper, then capturing the drawing with an accurate flatbed scanner. This results in a raster image which has to be subjected to further processing before any 'creative' steps can be applied to it. Direct input involves making the drawing through direct computer interaction with a drawing program and a matched input device, usually a mouse or passive tablet-and-stylus. This results in an internal data structure which defines the appearance of the drawing and from which the entire history of the process of making the drawing is potentially available. In particular the drawing can be defined in a so-called vector or stroke-based format in which the primitives are straight lines and free-form curves along with associated attributes like control-point coordinates, line thickness, colour etc. This structure facilitates subsequent processing steps and makes automatic in-betweening — the generation of intermediate drawings between key poses, as required in large numbers in cartoon animation — feasible. By contrast deriving an equivalent structure from the raster image of a scanned-in drawing is not possible by purely automatic means, and the lengthy and tedious task of reconstructing it interactively is simply not practical.

Ergonomic considerations have forced the developers of cartoon animation software to favour scanning-in drawn artwork and implementing a truncated data-path which leaves the in-betweening step as an unautomated process [Patterson and Willis 1994]. This puts an absolute constraint on improving the productivity of an animation studio with computer support and disallows any process which could rely on in-betweening or the vector form, such as sta-

ble rendering of highly textured brush strokes. While technical considerations overwhelmingly support direct input and vector artwork the artist's environment is in practice unacceptably unpleasant for sustained work.

This paper introduces a new form of interactive drawing tablet, Electronic Ink, which has the potential to support a drawing experience far more like pencil and paper than other interactive displays. Functionally it is very similar to an LCD display backed by a tablet and stylus, like the Wacom Cintiq™ or as found in a TabletPC, but ergonomically it is very different. The display medium is E-ink™ which is a thin composite layer whose (subtractive) colour can be switched electrically from one stable state to another. When the E-ink layer is placed over its switching electrodes it forms a surface which is as rigid as its supporting layers and which, when switched, shows a high-contrast, high-resolution image with negligible parallax when illuminated with ambient light — in fact quite similar to paper.

Electronic ink drawing tablets offer the possibility of readdressing the whole question of direct input but at the same time pose the question of appropriate models of interaction more starkly than with any other interactive display. If the medium is to be seen as paper-like it should not behave in a very un-paper-like way. Furthermore, its specific advantages should be exploited properly, particularly the virtual absence of parallax which makes the traditional point-and click approach assuming the use of clumsy tools to move a cursor around particularly inappropriate. Sketching would seem to be an appropriate metaphor, which in an animation context mainly involves curve drawing and editing. Unfortunately the usual models for curve editing are heavily biased towards point-and-click and optimised for manipulation with clumsy interaction tools like mice. Essentially a curve — as close to the intended shape as can be managed in a single interaction — is introduced into the drawing area, then manipulated into place with controls based on the mathematical formulation of the curve or curve chain. While this approach mitigates the clumsiness of mouse interaction, nothing less like the experience of drawing with pencil on paper, the very experience which most attracts artists to a profession which promises endless drawing, can be imagined.

In this paper we will be exploring new sketch-based models for interacting with and editing free-form curves, here specifically Bézier chains. The rest of the paper is structured as follows. In Section 2 we describe the electronic ink technology and the intention behind its adoption, in Section 3 we discuss the basic approach to curve drawing and methods of managing interaction, and in Section 4 we introduce two sketch-based models for curve editing. Results are exemplified in Section 5. Section 6 is our conclusions section in which we also set contexts for our work. It should be noted that the electronic ink devices we are working with at the time of writing are engineering prototypes and that our interaction models are currently being explored in simulation. In the end our models are only really practical in an interactive environment sustained by fully developed electronic ink panels and that development in turn is influenced by the models they have to sustain.

2 Electronic Ink Technology

When committing any type of artistic expression to paper, there are many factors influencing the end result. Among those are the drawing/painting tool and the paper's characteristics, e.g. softness or roughness. In the animation context it is still the case that most artwork is created on paper and subsequently digitised for what is in effect post-production. Some attempts have been made with direct input, where touchpanels and digitisers are used in combination

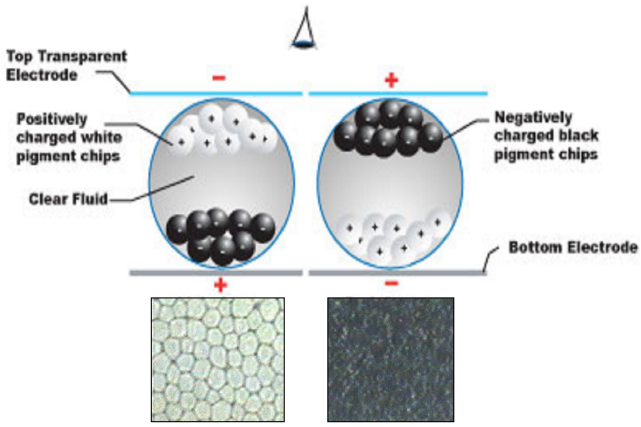


Figure 2: Microencapsulated Electrophoretic Display.

with computer displays but this is more usually employed in 3D animation where using a computer is not optional.

Vansichem et al. focused on the simplicity of the classic pen as an input device for drawing computer animation [Vansichem et al. 2001]. Here they used pen input to create the geometrical data needed for the automatically generation of the in-betweens. In doing this they focused on creating a digital curve drawing tool that works in a natural and intuitive way, which implies that the tool has to work in real-time and that the movement of the pen has to be closely tied to the resulting curve. However they did not take into account the difference in hand-eye coordination between drawing with a pen on paper and drawing with a digital pen on a tablet while watching the screen. Attempts have been made to combine a flat panel LCD display with a touchpanel but basic shortcomings fail to relieve the unpleasantness of current forms of direct input. Complaints include excessive parallax, highly quantised spatial resolution (both of which work against accuracy of input capture), unacceptable backlighting, and an unpaper-like feel (too smooth and slippery). To overcome these problems, a display was required which does not show these problems. One such display is the electronic ink panel in which the display is provided by an electrophoretic layer and input via a commercially available pressure-sensitive electromagnetic panel, of the kind available commercially, behind the display.

The original drawback with electronic ink was that it was not optimised for fast response. Full, accurate switching from black to white or vice-versa could take around 1 second which is well outside the 40 ms response time usually necessary to be considered an 'immediate' reaction. For correct reproduction of grey-levels, it is necessary to let a display update finish uninterrupted before a new display update can be initialised. New update strategies have been developed to allow continuous display updates, to the point that a user will not object to the time lag between input and actual reproduction [Henzen et al. 2004]. Considerably faster responses than those achieved at present will be possible in the near future [Whitesides et al. 2004]. These await a further development effort.

2.1 Electrophoretic Layer

The display layer used in the panel is manufactured by the E-Ink Corporation. It is based on microencapsulated electrophoretic layers, as shown in Figure 2, where two charged particle types, white and black, can be switched by means of an electric field. The result

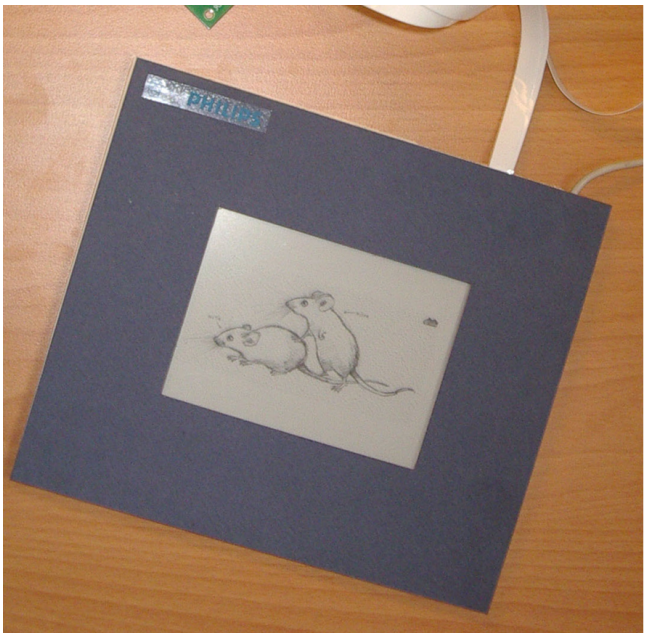


Figure 3: Prototype pane.

is a black-white switching layer, an example of which is shown in Figure 3. A layer of this kind is laminated onto a dedicated backplane and driven by specially designed electronic circuits [Henzen et al. 2002]. Although not specifically designed to have a fast response time, the response is fast enough to generate sufficiently good optical response directly following the stylus tip. The underlying principle is that the transition from black state to white state and vice versa does not take much time; 300 ms usually suffices. However, the accuracy of the grey level (16 grey levels) is not very good. This means that updates can be fast, but no strict requirements on accuracy should be posed. On the other hand, updates can be accurate, as long as sufficient time is invested. The drawing tablet therefore cuts corners when speed is required, and will restore accuracy once time is available.

2.2 Panel Engineering

The touch sensor used is a prototype panel employing electromagnetic signals to detect position and status of a special, passive stylus. The panel delivers information about stylus position, pressure and inclination, as well as the position of up to four switches incorporated in it. In tests no interaction between input sensor and output display has been detected which had been a concern.

Strategies for driving electronic ink panels were, until now, based on accuracy in reproducing grey levels [Zehner et al. 2003]. Since accuracy requires effort (time), update used to be slow, and new input was inhibited until the full image was written. If drawn input must be processed, it is unacceptable to suspend processing new touchpanel input until the previous information has been fully updated. Instead, the module accepts input data continuously, and the pixel addressing starts as soon as the coordinates are received.

(a) (b)

Figure 4: a) Smooth Hand. b) 'Nervous' Hand.

2.3 Operating Environment

The software driver is adapted to respond in the same manner as a normal digitiser input, but the display on top is provided with the drawn input data, thus directly representing the drawing on screen. One goal is to make the experience of using the device as paper-like as possible. Currently the sense is one of drawing on aluminium which is certainly rigid enough. By contrast the medium used in an LCD panel is soft and the thick transparent protective layer which causes the parallax problem has to be reinforced to stop it bending under stylus pressure distorting what is seen beneath. The support of this glass layer is a significant and persistent problem for this type of display as increased size requires increased thickness and worse parallax. For the electronic ink panel the problem is somewhat different as the surface needs to resist the passage of the stylus and feel less rigid. The former may be achieved in a number of ways, by coatings or by providing an intrinsically rough protective layer otherwise needed for sealing purposes. The latter can be achieved by providing a (slightly) compressible core to the stylus whose function is otherwise solely to press on the very slightly compressible resistor used to provide pressure readings. These aspects are to be the subject of an evaluation process to be conducted shortly.

3 Free-form Curve Model

If one goes to the lengths of trying to simulate the experience of drawing on paper the most obvious metaphor for the interaction model is one which is like sketching with pencil and paper. The basic interaction element is thus not a point, but a stroke, although internally strokes will be represented in terms of points (e.g. polyline vertices or control points). When drawing or writing users tend to make two different kinds of strokes, which collectively can be said to be done either with a smooth hand or a 'nervous' hand. Figure 4 shows two drawings, one executed by a smooth hand (a) and the same drawing executed by a nervous hand (b).

To make the point Figure 4 is a modification of animators' practice [Patterson and Willis 1994]. An animator would typically produce a drawing like that of Figure 4(b) in which the correct line path would be seen from repeatedly going over the same line. While the results of each stroke would be apparent in the clutter the line path would also be apparent as a black core. A 'clean-up' artist would then take the drawing and produce something like Figure 4(a) by tracing the paths with a smooth hand. A certain amount of interpretation is required because while the black core will contain the desired

line direction it may not constrain it tightly. A few animators have trained themselves to use a smooth hand consistently, often because of the constraints of computer systems, and for them the smooth mode will always be enough, but animators from an unconstrained or traditional training commonly use the nervous form to pin down the 'lay' of their line.

For automated in-betweening it is essential to be able to extract the core line as a free-form curve, i.e. to fit a curve model as closely as possible to a set of points, for example a polyline approximating the core line. Our freeform curve model is that of a Bézier chain and the curve control points are what are in-betweened (so there is a correspondence problem which is resolved by ensuring matching chain links in corresponding curves which are identified through corresponding hierarchies). Curve fitting is done at the same time as the artist draws a stroke, so this supports a 'smooth' hand model at a minimum. Some curve fitting techniques for the purpose of interpreting hand drawn strokes are introduced by Baudel [Baudel 1994] and Schneider [Schneider 1990]. We however are mainly interested in creating curves in an intuitive and real-time manner.

In our system the creation of a stroke is done interactively by sampling a stylus along the trail of the stroke. This only happens when the tip of the stylus touches the touchpanel. To allow for real-time high-level manipulation of the stroke the individual pixels that make up the stroke are not used, in contrast to their use in the nervous stroke algorithm. Instead, a high-level internal representation, using cubic Bézier curves, is created, using the classical formula quoted in the following equation.

$$f(t) = (1-t)^3 B_1 + 3t(1-t)^2 B_2 + 3t^2(1-t)B_3 + t^3 B_4 \quad (1)$$

While sampling the stylus we simultaneously perform an iterative curve-fitting technique based on least-squares error estimation which is done 'on the fly' while the curve is being drawn, using the solution of Vansichem et al. [Vansichem et al. 2001].

When fitting a cubic Bézier spline to a set of n data points, we need to determine its control points B_1, B_2, B_3 and B_4 so that the difference between the spline and the corresponding data points (P_1, P_2, ..., P_n) is as small as possible. The start and end point (B_1 and B_4) of the spline can be chosen in function of the smoothness represented by the parameter s: the number of shared data points.

$$\begin{aligned} B_1 &= P_s \\ B_4 &= P_{(n-s)} \end{aligned} \quad (2)$$

The two other control points (B_2 and B_3) can be constructed by means of least square minimisation [Press et al. 1995].

$$S = \sum_{i=1}^{n} (P_i - f(t_i))^2$$
$$\frac{\partial S}{\partial B_3} = 0, \quad \frac{\partial S}{\partial B_2} = 0 \quad (3)$$

In order to use this method we must find the t-values of equation 1 along the cubic Bézier spline, which correspond to the data points. These values $t_1, \ldots, t_n$ are found as an approximation of the chord length between each pair of data points.

Let $d_{i,j}$ be the distance between P_i and P_j:

$$d_{i,j} = |P_i - P_j| \quad (4)$$

The total chord length is:

$$L = \sum_{i=s}^{n-s-1} d_{i,i+1} \quad (5)$$

We now can calculate the t-values and use them in the least square minimisation:

$$\begin{aligned} t_s &= 0 \\ t_i &= (\sum_{j=s}^{i-1} d_{j,j+1})/L, \quad \forall i > s \\ t_i &= -(\sum_{j=s}^{i+1} d_{j,j-1})/L, \quad \forall i < s \end{aligned} \quad (6)$$

Substituting Equations 1, 4, 5 and 6 in Equation 3 yields the following equations:

$$\begin{aligned} B_1 \sum t_i(1-t_i)^5 + 3B_2 \sum t_i^2(1-t_i)^4 + 3B_3 \sum t_i^3(1-t_i)^3 \\ + B_4 \sum t_i^4(1-t_i)^2 - \sum P_i t_i(1-t_i)^2 = 0 \\ B_1 \sum t_i^2(1-t_i)^4 + 3B_2 \sum t_i^3(1-t_i)^3 + 3B_3 \sum t_i^4(1-t_i)^2 \\ + B_4 \sum t_i^5(1-t_i) - \sum P_i t_i^2(1-t_i) = 0 \end{aligned} \quad (7)$$

The values of B_2 and B_3 are determined by solving this set of equations.

4 Curve Drawing and Editing

To actually draw the spline, the t-values, $t_1 \ldots t_n$, which are found as an approximation of the chord length between each pair of data points, are substituted in Equation 1. This results in a series of data points which lie on the Bézier spline. Tracing out these data points using hardware accelerated graphics primitives (i.e. line-segments) yields a visually smooth curve.

Regarding the editing, a suitable model for interaction in a sketching metaphor is proposed by Di Fiore et al. [Di Fiore et al. 2004]. A basic principle of the interface is that it does not require a cursor (whose presence would be quite un-paper-like), another principle that explicit mode-change interactions should be minimised, typically by making reversible inferences. Response to an action appears immediately under the stylus as a mark. Accordingly if controls or mode-changes are required the user can call these up by making a stroke-based sketch which specifies the new mode. This can be done anywhere in the drawing area. Since the nervous style is not generally used for writing of any form nor for making iconic drawings like arrows it is assumed the smooth stroke form is used in these contexts. The interface requires the use of the concept of a currently selected object and strokes will have associated with them their recentness of selection (this is sometimes needed to recognise sketch icons). The concepts of grouping, and the joining of separated but nearby lines within tolerances, are also required.

4.1 Line Connectivity

Although this paper is not addressing anything above the lowest level of curve editing we need to consider briefly the issue of line connectivity modelling and its role in minimising the need for explicit mode changing. Even for single smooth strokes it is possible to derive a non-simple connectivity model even before editing is considered, e.g. simulating erasing and redrawing.

Our connectivity model may be described (here informally) in terms of a directed graph whose edges represent the individual strokes making up a portion of the drawing being worked on, and whose nodes, which may have any in- or out-degree greater than zero, represent stroke junction points. We will model such a graph

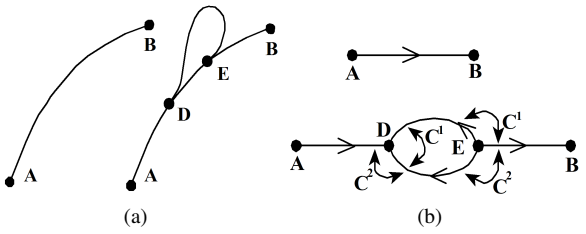


Figure 5: a) Line *AB* and an edit. b) Connectivity graphs.

here diagrammatically although there are many well-known ways of modelling directed graphs efficiently for computation. A node or an edge may be decorated with various parameters and here we need to specify a direction (not necessarily the original direction in which the stroke is made), the degree of continuity associated with the strokes modelled by the edges at their junction points (the nodes), and a reference pointing to the stroke model itself. This graph serves two purposes. It shows how individual strokes relate to one another and any closed regions they might define, also it shows how to reconstruct the drawing precisely. Transformations to the drawing or to parts of it are applied directly to the connectivity model from which all the relevant parameters can be found and when the graph is subsequently interpreted to yield a drawing the results of the transformation will be seen. The connectivity graph is not however a semantic model as it is derived entirely from the drawing process itself. Semantics can be imposed externally by means beyond the scope of this paper. The graph is however a working hypothesis of the line structure and connectivity which is updated in the light of user-supplied evidence of an implicit or explicit kind. As can be seen from the interface discussion we want to avoid explicit correction as much as possible but it is always allowed as an option.

We consider as an example the drawing and editing of a smooth line *AB*. The original drawn line is shown in Figure 5(a)(left). The connectivity model for this stroke is shown in Figure 5(b)(top). If a section (say section *DE* in Figure 5(a)(right)) was deleted from *AB* and a new segment added then the connectivity graph would be restructured into three sections, *AD*, *DE* and *EB*, with appropriate annotations reflecting the continuity established between the sections in the edit. The curve model for *AB* also needs to be modified to provide Bézier end-points at *DE* where the relevant continuities to be enforced are associated with the connectivity graph.

If instead the loop *DE* was simply drawn over *AB* then, inferring connectivity and continuity from proximity (spatial proximity for connections and tangential alignment to a tolerance for continuity), the outcome would be the drawing in Figure 5(a)(right) and the connectivity graph in Figure 5(b)(bottom). Here the continuity inferred (C^1, C^2) is shown as labels. Directions on the lines indicate how to interpret the continuity conditions.

The point is these graphs are models for the local stroke structure and that even inferred-mode edits with smooth lines can swiftly affect the structure of these graphs. Structures of this kind can arise even more easily with the nervous hand model, so connectivity graphs are an essential part of the 'line hypothesis' used in this model. The further management of these graphs is beyond the scope of this paper.

The simplest control can be effected by a single stroke, and here such single strokes are overloaded to allow a variety of mode change operations without stepping too far away from the sketching on paper metaphor. A smooth stroke consisting of a short straight line (within tolerances) crossing a nervous stroke would normally

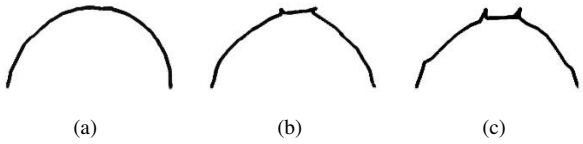


Figure 6: Screen shots from prototype showing the effect of a curve crumpling when points are not removed. a) Original line. b) Artist attempts to manipulate curve decreasing its size, crumpling effect starts to show. c) Artist continues to shrink curve, crumpling effect very apparent.

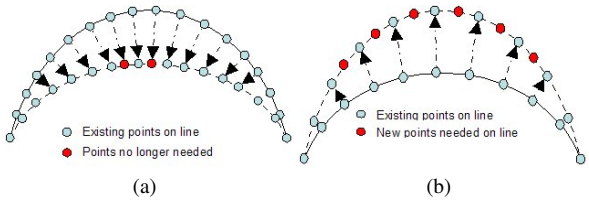


Figure 7: Preserving line stability. a) Example shows how points are pushed closer together as a curve is reduced in size; eventually points will start to crumple and so must be removed. b) Shows how points are separated as the radius of a curve is increased. Eventually the line will become faceted and so new midpoints must be added between existing points.

indicate selection of the nervous stroke. One attractive feature of drawing icons directly is that their meaning can be modified with specific modifiers so that a quite rich mode set can be build up from a small number of primitives requiring few drawings to be learned. Here a mode change can be retrospective ('I forgot to do the mode change before I started') or prospective depending on a simple qualifier (a single stroke through the icon would be enough). The risk of confusion over mode changes can be avoided by showing the current mode icon in a discreet part of the display. After invocation the icon or stroke fades out after a short interval and thus disappears from the main drawing area.

Alternative techniques for manipulating parametric curves are described in [Fowler and Bartels 1993; Grimm and Ayers 1998; Zheng et al. 1998; Raymaekers et al. 2002].

4.2 Line Stability

When the artist is editing a spline, the problem of line stability needs to be addressed.

In the situation where the artist is reducing the size of a curve, the number of points must be reduced to prevent a crumpling effect occurring (see Figure 6). A validity check checks when points are too close to each other (see Figure 7(a)), flagging a point for removal if so. This prevents crumpling of the curve.

In the occurrence of the situation when the artist attempts to increase the length of a curve by dragging it outwards, more points must be added to the line hypothesis to prevent the curve losing its shape and appearing faceted (see Figure 7(b)). This is done by checking that the distance between the start point and newly found middle point is less than a globally defined constant.

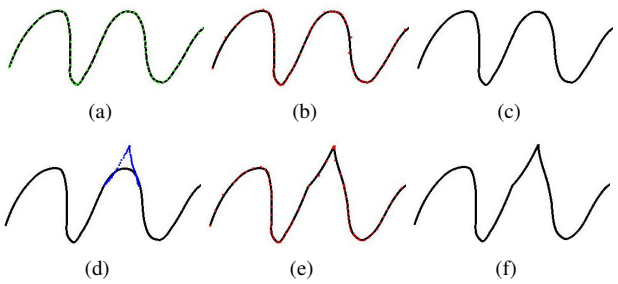


Figure 8: Example of editing a free-form stroke. a–c) Initial drawn free-form stroke. Shown with data points, control points, and as depicted to the user. d) New data points as indicated by the user using the stylus. e) Refitting of the original free-form stroke, shown with control points. f) Final result after editing the original stroke as depicted to the user.

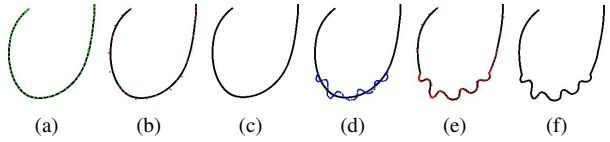


Figure 9: Example of editing a free-form stroke. a–c) Initial drawn free-form stroke. Shown with data points, control points, and as depicted to the user. d) New data points as indicated by the user using the stylus. e) Refitting of the original free-form stroke, shown with control points. f) Final result after editing the original stroke as depicted to the user.

4.3 The Smooth Hand Model

Our main concern here is free-form curve drawing and editing at a level below general considerations of interfaces, although interfacing issues inevitably arise. A stroke is originally effected by making a single gesture with the stylus in contact with the drawing surface. Non-zero pressure indicates the stroke is being made and its completion is indicated by pressure returning to zero. In the drawing surface this results in the usual coordinate pairs being generated from samples taken at regular short intervals of 5 ms (200 samples/sec), as in Figures 8(a) and 9(a). While the stroke is being made the least-squares process discussed in the previous section is run continuously on the generated points to derive the curve control descriptors, as shown in Figures 8(b) and 9(b). This is a real-time process which results in pleasant smooth-looking lines (e.g. Figures 8(c) and 9(c).

Our models so far cater exclusively for smooth strokes but there is no recognised or clearly identified sketch-based method for editing them. In a pencil-and-paper world one might imagine erasing a line and amending it. Accordingly here we can perform the erasing function by delimiting a segment of the drawn curve with single strokes (since two are required this disambiguates selection) and re-draw the curve using the greyed-out portion of the line as a sight guide (e.g. Figures 8(d) and 9(d)). Ends close enough to the edit points can be snapped after the stroke is completed (e.g. Figures 8(e) and 9(e). Completion of the editing stroke is again signalled by zero pressure (within a tolerance) on the stylus tip.

One problem with a drawing-based approach to defining the descriptors of a free-form curve model is that of continuity. In a Bézier chain there is an opportunity to enforce varying degrees of continuity at each joint or knot in the chain but the model leaves it up to the application as to what degree of continuity is enforced or how it is specified. Typically first-degree geometric continuity would be enforced and higher degrees (up to second degree parametric) promoted if the resulting curve smoothness justified it (e.g. by comparing tangents and matching to a tolerance limit). The real problem is C^0 continuity, which appears as a corner. In practice it is impossible to draw a corner without stopping the pencil moving. It may be still in contact with the drawing surface but it needs also to be stationary for at least two sample periods, which will yield two coordinate pairs which are identical within noise tolerance.

4.4 The Nervous Hand Model

The nervous hand model is handled somewhat differently, and is more like painting albeit with a pencil-like 'brush'. The intention is to derive a curve, which here we refer to as the 'line hypothesis', which matches the trajectory of the curve of which the combination of marks infers to the artist. The order in which the artist makes these marks is not taken into account directly, rather the line hypothesis runs through the most intensely coloured[1] region joining the end-points and follows what is taken to be the appearance of the line. The line is 'edited' by adding colour close to the currently hypothesised trajectory so that the line 'drifts' into the region now perceived to be the most intensely coloured. This is therefore a model in which an artist can alter the lay of the line in terms of manipulating the accumulation of the intensity of the mark. What the artist perceives as the core line is the core line and this core can be manipulated by adding pigment alongside it.

The brush footprint of the pencil consists of a disc defining the initial colours. Pressure determines the disc radius (discs are precalculated). This disc is swept along the polyline trajectory defined by the sample points interpolating pressure values to make the right selection. When a disc footprint overlaps with another the affected pixel values are altered additively. Once painted each pixel proceeds to fade to background at a specified globally-defined rate. A polyline is continuously fitted to the darkest part of the line and does not attempt to follow stroke trajectories but instead finds the minimum energy state as instantaneously determined by the isochromic surface defined by the local pixel values. The polyline acts as a precursor to a Bézier chain by providing points to which the chain is fitted and it is this chain which provides the actual core line hypothesis. For reasons of feedback pixels on the core line are constantly refreshed and the polyline precursor is not suitable for this purpose as it can cause an untended curve to drift towards its local centre of curvature. So it is the Bézier chain which provides the trajectory which is refreshed by redrawing the stroke over its current path frequently enough to avoid fading being seen. The effect of this redrawing is to reinforce the currently defined trajectory which can now be changed by resetting fading pixels near to it with another stroke or another part of the same stroke action. The effect is to drag the trajectory into the most recently reinforced region, also reinforcing the rest of the trajectory.

Painting is carried out in the usual way placing a foreground colour f on top of a background b. The resulting colour $c(p)$ is determined by the linear interpolation formula:

$$c(p) \quad = \quad f \cdot p + b \cdot (1-p) \tag{8}$$

so sketching with a black pencil ($f = 0$) on a white background ($b = 255$, say) shows a mark which gets blacker for larger values of

[1] We will describe this as 'colour' or 'pigment' because our model allows for any foreground or background colour but the e-ink surface only shows grey pigment (16 levels) on a light background (or vice-versa).

p. New pixel values determined by p_{new} are painted into the framestore conventionally with a circular 'brush' or pencil footprint, but each pixel value $c(p)$ in the footprint increments the current value $c(p_{old})$ according to the formulæ

$$p_{new} = p_{old} + p \quad (9)$$
$$c(p_{new}) = c(min(p_{new}, 1)) \quad (10)$$

The pencil footprint is usually modelled in terms of fixed-point values of the parameter p of which, while it can exceed 1, the associated colour value $c(p)$ is clamped to $c(1)$. Typically this is of the form of a disc of values with a maximum value in the centre tapering to 0 at the edge. While it might be possible to select these values stochastically or otherwise simulate pencil behaviour realistically there is a risk of confusing the trajectory finder so the model of a truncated cone of uniformly tapering values is used.

The 'nervous' hand algorithm is described by the following pseudocode:

No	Function	Thread
1	Initialise search structures	1
2	**repeat**	1
2a	Paint in new pixel values	1
2b	Update line hypothesis	2
2c	Refresh framestore with stroke along line	3
2d	Fade pixels	1
	until smooth line state	1

Pixels are faded at a constant rate k which causes them to fade according to $\frac{dp}{dt} = -k$ i.e. a pixel fades according to $p_{new} = max(p_{old} - k \cdot t, 0)$. This constant k allows for global adjustment of the fade rate.

The line hypothesis is the currently assumed trajectory for the stroke. This corresponds to a single segment of the line connectivity diagram referred to earlier and consists of a Bézier chain model for the stroke and highlights the end points and any continuity conditions which exist in respect of other lines which share the end-points. As can be seen from the example in Figure 5, a single line initially drawn as a (quasi-)continuous entity may have end-points inserted in it because of a later edit. Although the connectivity diagram will now show the line as consisting of several segments each with their own end-points these points are in fact interior points and the continuity of the original line is noted and retained for enforcement in any refreshing process.

The polyline approximation (and the chain itself) are managed by the *update line hypothesis* process described in the following pseudocode:

No	Function
1	Confirm end-points
2a	**if** new end-point **then**
2b	Position new line segment (polyline)
2c	**else**
2d	Find local maximum (polyline)
3	Fit Bézier chain

The `Find local maximum` process is the one which is usually invoked. This assumes the existence of a previous solution and the delimiting of the sub-region of the brush footprint which has been updated by user action since the previous line hypothesis trajectory was refreshed. The boundaries of this sub-region are extended to include the nearby elements of the line hypothesis. Each control point in the line hypothesis falling within the update region is moved in succession as closely along the line normal as pixellation allows until it encounters a local maximum. This starts with the control point nearest the centre of the update region and propagates outwards until no further point movements are required, i.e. the next points are already at their maxima and are outside the update region. If a control point is moved the mid points between the points may themselves move off local maxima and if so they become control points themselves and are moved accordingly. This splitting process is carried out after all the pre-defined control points have been moved and may involve control point removal if three or more control points form a straight line within pixellation error.

The `Position new segment` process extends a line hypothesis, taking the previous end-point as a start point, so it also initialises a line hypothesis. This is a recursive process which makes an initial hypothesis that the line is a straight line between its end-points, then tests to see if the mid-point lies on the maximum value traversed by the normal at the mid-point. If not the line is split, the mid-point repositioned on the maximum and the process repeated for each half of the split line.

The behaviour of this algorithm is to refine the nervous hand into a curve in a way which allows the artist to control the refinement process. To an artist the control process is just like the process of refining the lay of a line by repeated overdrawing, a familiar and standard technique which in this case needs the accuracy of the electronic ink drawing surface to make it work properly. An example of a simulation of the algorithm on a LCD backed-tablet of approximately half the spatial resolution of an e-ink panel is shown in Figures 10–12. The final step refines to a polyline with integer control points, rather than a Bézier chain, so the resulting line looks less smooth than it could do.

Figure 10: An example of a simulation of the algorithm on an LCD backed-tablet of approximately half the spatial resolution of an e-ink panel. a) Initial 'nervous' line. b) Redraw reinforcing top part of the line. c) Line partly faded. d) Final lie of the line as reinforced.

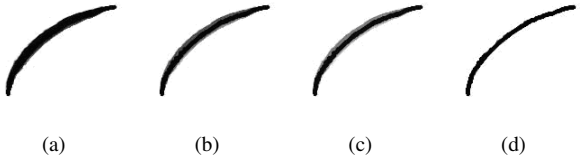


Figure 11: An example of a simulation of the algorithm on an LCD backed-tablet of approximately half the spatial resolution of an e-ink panel. a) Initial 'nervous' line. b) Line after initial fading. c) Redraw reinforcing top part of line. d) Final lie of the line.

A nervous stroke evolves into a smooth stroke, so any combination which is allowed for smooth strokes is also allowable with nervous strokes or nervous and smooth stroke combinations, e.g. using nervous strokes to edit smooth strokes. The initial assumption is that a smooth stroke is being drawn but the nervous stroke model is invoked as soon as the line trajectory executes a close to 180^o turn or the user makes a new stroke (lift-move-press) along the direction of a curve provisionally modelled as smooth. (An 180^o turn is a potential indicator of an end-point except in close proximity

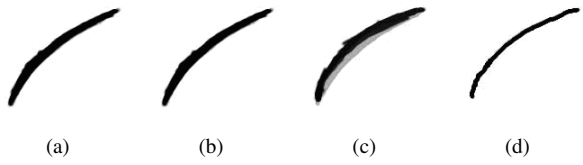


Figure 12: An example of a simulation of the algorithm on an LCD backed-tablet of approximately half the spatial resolution of an e-ink panel. a) Initial 'nervous' line. b) Line after initial fading. c) Redraw reinforcing middle of line. d) Final lie of the line.

of a previously existing part of the current line being drawn.) Some precautionary initialisation assists the model reconstruction process and any curve data already available can be used as previous iteration data. The situation in which the user draws a smooth curve, then starts to draw over it, is ambiguous as what might be intended in a drawing in the style of Figure 8(b). This will only become clear if the new line trajectory moves sufficiently far away from the overdrawn line to trigger a change in the connectivity model rather than the line model (or both). This is detected in terms of the intersection of the footprint of successive brush updates with previously set pixels. If this footprint intersection is identical to the footprint of successive brush updates in the absence of other set pixels then the trajectory has become disconnected. If the intersection contains additional set pixels then it is still connected. Conditions such as *connected* (as here), *end-point hypothesis* (as above) and *corner hypothesis* (where the stylus is stationary for at least two samples) are used to manage the line connectivity graph continuously during the nervous hand interaction.

The nervous stroke can thus be seen to be a form of a smooth stroke which allows for interactive editing in a manner intended to suit animators. Although complicated to implement and awkward to fit into a curve interaction model which relies on making simple inferences to avoid explicit mode changes the nervous hand model gives closure to the range of curve management techniques we are presenting here for sketching.

5 Results

The results section contains screen shots of the nervous hand program as drawn by the developer and frames taken from a video made of the developer sketching with the tool, it should be noted that the developer is not an artist by trade and hence the curves shown are not those of a professional.

Long sweeping curves such as those in Figures 10–12 were drawn using the nervous hand tool. The curve found is a fairly smooth representation of the original curve drawn.

Smaller curves can be drawn as in the example shown in Figure 13. The curve is first sketched and the line core is found; the artist then redraws the top left corner dragging it outwards and the top right corner dragging it inwards to demonstrate the flexibility of the curve. The tighter curve still appears smooth and as can be seen, is easily modified.

The examples shown in Figures 14 and 15 show the nervous hand tool in action.

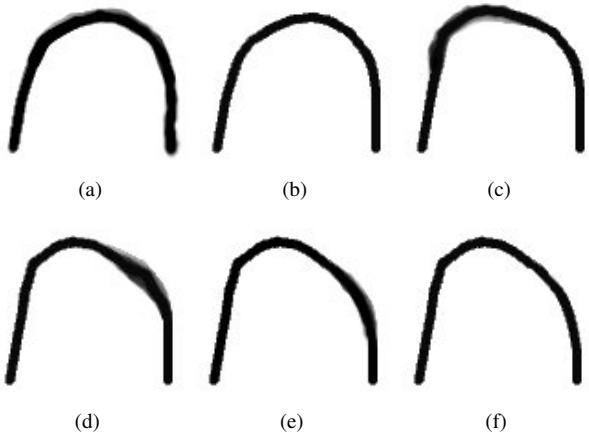


Figure 13: Shows a tighter curve drawn using the nervous hand technique. a) Shows the line as originally drawn by the artist with fading beginning to take place with (b) showing the curve as found by the tool. Shot (c) shows the corner of the curve edited, moving it outwards and shots (d–e) show the right corner dragged inwards. Shot (f) shows the final lie of the curve.

5.1 Testing

Testing was an ongoing process in the development of the nervous hand system. Testing was carried out by the developer throughout implementation using the Electronic Ink Drawing Tablet as well as a WACOM CintiQ 15x graphics tablet with stylus. By testing in this way the developer was able to see how accurately the hypothesised line followed a line drawn by an artist and also the smoothness and stability of the line.

Various shapes were drawn throughout testing to test various aspects of the code. Long sweeping curves were sketched using the nervous hand style to test the accuracy and smoothness of the cores found, these were then edited making the curves smaller and bigger to test to what extent the algorithm allowed the curve to be moulded. Corners of different angles were also drawn to see how sharp an angle the system would find, and sinusoidal lines of varying sharpness were drawn also checking how tight a curve could be. Spirals were drawn and other similar shapes concatenated together testing the flexibility of the tool.

Extremely long curves and lines were used to test that the implementation of the algorithm could cope with finding large numbers of points in a fast time ensuring the speed of the tool. Lines were also drawn using a smooth hand style of drawing to see how accurately the system would find these lines. Shapes drawn using the system were compared to similar sketches on paper confirming or disproving the accuracy of the algorithm as these are ultimately what the system would replace.

5.2 Evaluation

Results for all large and medium curves demonstrated that the tool works very well for curves of this size, showing that the correct core of the line could be found and that curves could be manipulated to any extent the artist needed. The curves were shown to be smooth.

Results for small curves and sharp corners however showed that a corner hypothesis is required that would detect the shape of a curve, modifying the distance between points found according to

the curves severity. If a corner hypothesis was implemented, it would be possible to draw small curves and corners with the system finding points close together in these regions, but also draw larger curves with points found at a greater distance so that they appear smooth.

When drawing circles, the model only allows a circle to be drawn if its endpoints are set after drawing the circle only once. This does not fit with methods used by some artists who often repeatedly draw round a circle to get the most circular shape. Because of this an endpoint hypothesis should be included in the model to allow endpoints to change as the artist draws. This endpoint hypothesis should also deal with the problem of endpoints on a line so that they can be modified as the artist modifies the line.

Other points noted during the testing phase were that it is difficult to draw perfectly straight lines with the tool. It is expected that this problem would be greatly reduced with a higher definition graphics tablet; however, if a perfectly straight line is required by the artist then existing tools for drawing straight lines could be used in such a situation.

Tests were also undertaken to prove the speed of the system. This was done by drawing very long curves and lines that filled the entire drawing space to check that the system could continue to function as required under strain. These tests however were run without trouble and the system functioned as normal despite the high loads.

We refer the reader to [Foss 2005] for more details about the testing, evaluation and future extensions.

6 Conclusions

Electronic ink pads have the potential to be able to simulate the experience of drawing on paper to an unprecedented degree and certainly in ways which would be inappropriate for other reactive gesture capture devices. In a sketching role they pose subtle problems for interaction management while facilitating solutions which are novel to managing curve drawing. The intention has been to try to fit as closely as possible with animators' traditional drawing skills, to retain for them as much of the experience of drawing on paper as is possible, and to retain in automation those aspects of their traditional work environment that appeal to them the most.

What artists will make of this is the subject of current evaluations being conducted with professional animators within the IST project CUSTODIEV, which provides the means to carry out this work. CUSTODIEV is researching new models for animation and its deployment which best engage with traditional animation skills, of which the reported work is a part.

Acknowledgements

We are grateful to the European Commission for their funding of IST 37116 CUSTODIEV. Part of the research at the Expertise Centre for Digital Media is funded by the ERDF (European Regional Development Fund), the Flemish Government and the Flemish Interdisciplinary institute for Broadband Technology (IBBT).

We would also like to thank Andrew Foss for helping us implementing the nervous hand tool.

References

BAUDEL, T. 1994. A mark-based interaction paradigm for free-hand drawing. In *Proceedings of User Interface Software & Technology (UIST)*, 185–193.

DI FIORE, F., VANDOREN, P., AND VAN REETH, F. 2004. Multimodal interaction in a collaborative virtual brainstorming environment. *Lecture Notes in Computer Science LNCS series. First International Conference on Cooperative Design, Visualization and Engineering (CDVE2004) LNCS3190* (September), 47–60.

FOSS, A. 2005. *The Nervous Hand.* Bachelor's thesis, Department of Computer Sciences. University of Glasgow.

FOWLER, B., AND BARTELS, R. 1993. Constraint based curve manipulation. *IEEE Computer Graphics and Applications 13*, 43–49.

GRIMM, C., AND AYERS, M. 1998. A framework for synchronized editing of multiple curve representations. In *Proceedings of EUROGRAPHICS*, 31–40.

HENZEN, A., PITT, M., YASUI, M., DIJKMAN, W., AMUNDSON, K., ZEHNER, R. W., AND GATES, H. 2002. Development of active matrix electronic ink displays for smart handheld applications. In *Proceedings of International Displays Workshop (IDW)*, 227–230.

HENZEN, A., AILENEI, N., VANSICHEM, G., VAN REETH, F., AMUNDSON, K., AND ZEHNER, R. 2004. An electronic ink low latency drawing tablet. *Society of Information Display (SID): digest of technical papers XXXV*, 1070–1073.

PATTERSON, J. W., AND WILLIS, P. J. 1994. Computer assisted animation: 2D or not 2D? *The Computer Journal 37*, 10, 829–839.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1995. *Numerical Recipes in C*. Cambridge University Press, ISBN: 0-521-43108-5.

RAYMAEKERS, C., VANSICHEM, G., AND VAN REETH, F. 2002. Improving sketching by utilizing haptic feedback. In *Sketch Understanding: Papers from the 2002 American Association for Artificial Intelligence (AAAI2002) Spring Symposium. Technical Report SS–02–08*, 113–117.

SCHNEIDER, P. 1990. *Graphics Gems I.* Academic Press Inc., ch. An algorithm for automatically fitting digitized curves, 612–627.

VANSICHEM, G., WAUTERS, E., AND VAN REETH, F. 2001. Real-time modeled drawing and manipulation of stylized cartoon characters. In *Proceedings of the IASTED International Conference on Computer Graphics and Imaging*, IASTED, 44–49.

WHITESIDES, T. H., WALLS, M., PAOLINI, R., SOHN, S., GATES, H., MCCREARY, M., AND JACOBSON, J. 2004. Towards video-rate microencapsulated dual-particle electrophoretic displays. *Society of Information Display (SID): digest of technical papers XXXV*, 133–135.

ZEHNER, R. W., AMUNDSON, K., KNAIAN, A., ZION, B., JOHNSON, M., AND ZHOU, G. 2003. Drive waveforms for active matrix electrophoretic displays. *Society of Information Display (SID): digest of technical papers XXXIV*, 842–845.

ZHENG, J. M., CHAN, K. W., AND GIBSON, I. 1998. A new approach for direct manipulation of free-form curves. In *Proceedings of EUROGRAPHICS*, 327–334.

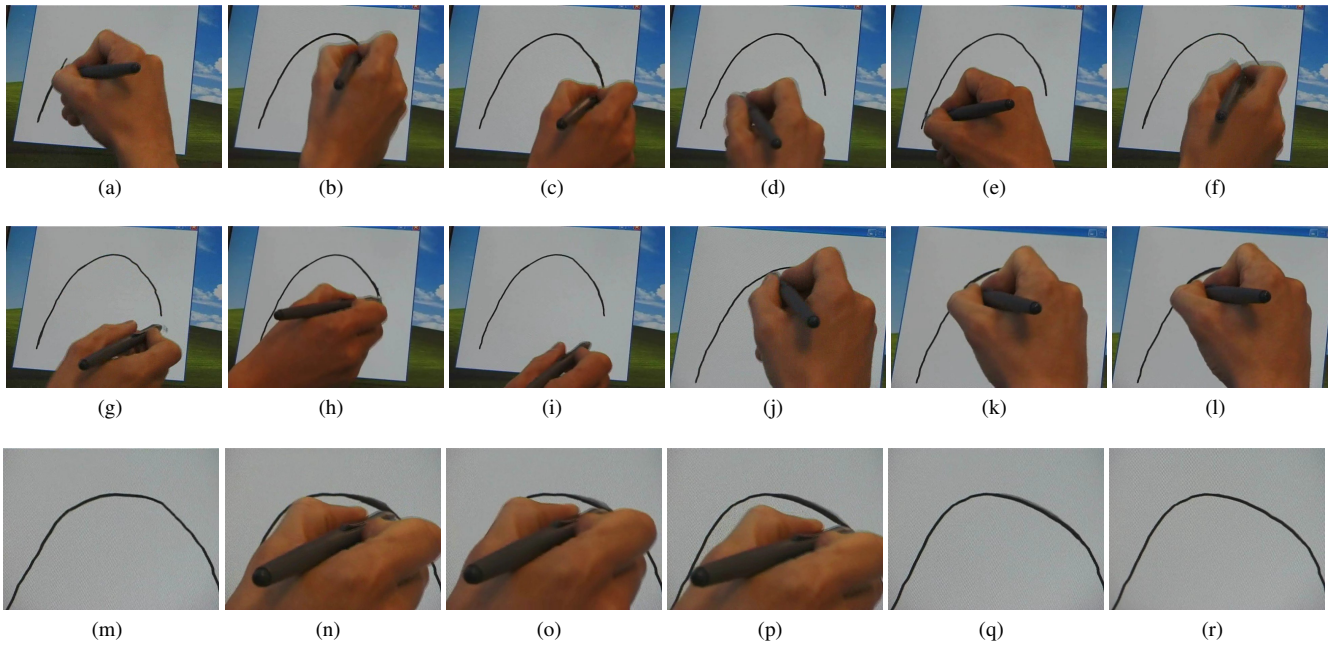


Figure 14: Shows a curve as drawn by the nervous hand tool. Shots (a–c) show the artist sketching the curve, shot (d) shows the line core with outer pixels starting to fade. Shots (e–i) show the artist as s/he begins to tidy the line up making minor changes. Shots (j–l) show the artist as s/he starts to manipulate a corner of the curve by dragging it outwards with shot (m) showing the slightly modified curve. Shots (n–p) show the artist modifying another corner of the curve, dragging this corner inwards whilst also smoothing it out. Shot (q) shows the curve as the old curve begins to fade with shot (r) showing the final lie of the curve.

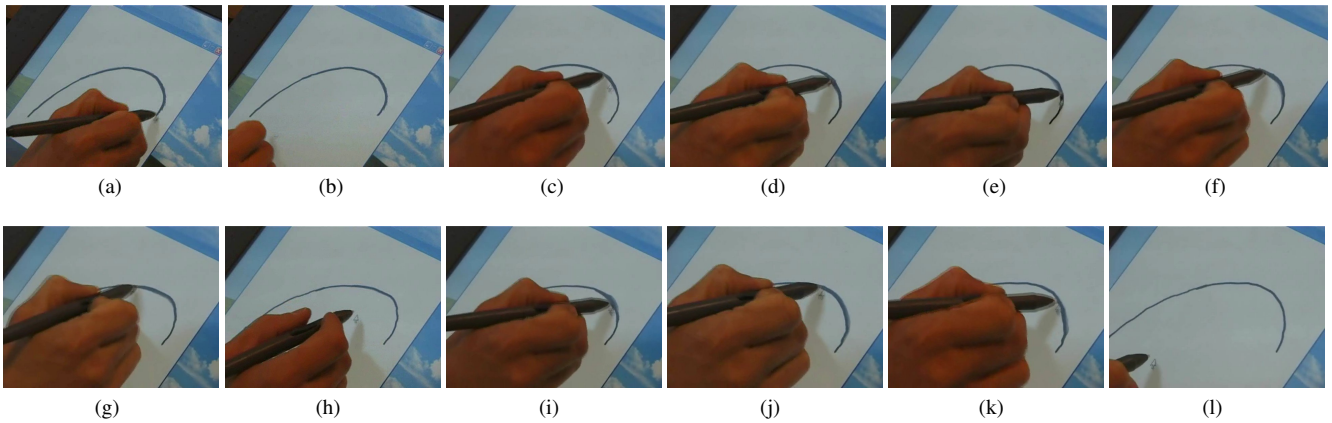


Figure 15: Shows a curve as drawn by the nervous hand tool. Shot (a) shows the curve as first drawn with shot (b) showing the curve with all but the core line faded. Shots (c–g) show the artist dragging the top right part of the curve outwards with shot (h) showing the core line found after dragging had finished. The artist then demonstrates the curve being dragged back inwards in shots (i–k) and shot (l) shows the final lie of the curve.