*Journal. Pattern Recognition*, Vol 34, No. 10, pp. 1885-1893, 2001.

# On-Line Segmentation of Freehand Sketches by Knowledge-Based Nonlinear Thresholding Operations

Sheng-feng Qin[*], David K. Wright[1], and Ivan N. Jordanov[2]
Department of Manufacturing Eng. Loughborough University, Loughborough, LE11 3TU, UK
[1]Department of Design, Brunel University, TW20 OJZ, UK
[2]DERC, University of Wales Institute, Cardiff CF5 2YB, UK

***Abstract.*** This paper describes an on-line procedure for segmenting freehand sketches into 2D primitives and smooth curves by knowledge-based nonlinear thresholding operations. Drawing position, direction, speed and acceleration information is used for establishing fuzzy knowledge and inferring user's intention. The procedure can be divided into three stages: (1) obtaining corner points based on the fuzzy knowledge and adaptive thresholds, which includes detecting "obtuse" and "acute" corner points; (2) finding smooth join points, during which stage we use both curve linearity and curvature; and (3) classifying the sketches into the corresponding curves according to curve linearity and changes of curve convexity. The method is illustrated and tested with a number of sketches.

***Key words:*** Heurestic knowledge, on-line sketches, curve segmentation and classification, conceptual design.

## 1. Introduction

A variety of CAD systems, such as advanced parametric and feature based systems, have been widely accepted by design engineers for detailed design. However, current CAD systems are not sufficiently developed to be used intuitively for conceptual design where designers use various sketches with vague and imprecise geometry to rapidly express their ideas [1-3]. Our current research investigates a knowledge based 3D rapid geometric modeller from 2D perspective projection sketches [4]. The segmentation of on-line freehand sketches is one of the most important elements in this research, because errors in the segmentation might propagate to feature extraction and classification errors. In engineering design, most parts are bounded by flat, quadric or free form surfaces. Their 3D perspective projections consist of line segments, circles or arcs, ellipses or elliptical arcs, or free form curves. Hence, we are interested in segmenting 2D sketches into 2D primitives and smooth curves.

[*] Corresponding author: Tel: +44 1509 263171 ext 4227; Fax: +44 1509267725; Email: s.f.qin@lboro.ac.uk

For the past two decades, many curve segmentation techniques have been proposed, most of which use straight lines to approximate the edge pixels in an image [5-9]. Chen, Ventura and Wu [10] emphasised that the piecewise linear approximation of digital curves is scarcely useful for segmenting curves in a meaningful and compact form. Describing the digitised curves in terms of conic arcs and straight-line segments is very important as stated in [11]. Several algorithms have been developed for segmenting planar curves into conic arcs and straight lines. These approaches can be briefly divided into two major categories, namely break point detection and edge approximation. In the first method, break points are usually detected at corner points for which the tangent to the contour is discontinuous, or smooth join points for which the tangent is continuous, but the curvature is discontinuous [10]. The chain-code based segmentation schemes have been used to detect break points [10, 12, 13]. The 8-directional code is applied in common to represent the tangent at a point. Generally, these coding schemes offer some advantages, including simplicity and speed. However, they are very sensitive to noise due to a limited number of directions. In the second method - edge approximation, planar curves are segmented into straight lines and conic arcs by repeatedly using fitting and segmenting, based on some "goodness-of-fit" [14, 15]. The "split-and merge" edge approximation schemes are the main techniques employed in recent research. They are based on an iterative approximation of a curve by straight segments and a classification of merged segments [16, 17]. Rosin and West [18] developed a non-parametric procedure for segmenting planar curves into lines, circular arcs and elliptical arcs, but Kanatani [19] pointed out the existence of a theoretical difficulty, underlying the curve segmentation problem studied by Rosin and West. One major problem of edge approximation is that it does not fully exploit the properties of dominant points, needing heavy computation.

In addition to the properties of low-level images with significant noise, on-line sketches also have some dynamic features in terms of drawing direction, speed, and acceleration. Chen and Xie [20] use geometric information, such as change of angle as well as acceleration and speed information, to find turning points. Their work, however, did not show sufficient robustness and is focused mainly on curve classification. The goal of the work being described in this paper is to improve Chen and Xie's method for segmenting on-line perspective projection sketches into 2D primitives and smooth curves with respect to computational efficiency, accuracy and robustness. We use an adaptive threshold segmentation technique, based on heuristic knowledge in terms of sketching behaviour: speed and acceleration, sketching direction, and object geometric linearity.

In Section 2 we present our method for detecting corner points. A method of finding meaningful smooth join points is discussed in Section 3. A simple classification procedure is then given in Section 4, and in Section 5 we demonstrate examples and case studies of segmented curves, including an example of 3D CAD applications. Finally, in Section 6 conclusion based on qualitative comparisons with other methods is given.

## 2. Detection of corner points

### *2.1 Data collection and filtering*

While sketching, input data is received as a sequence of mouse positions from pressing the mouse button, moving the mouse when the button is still pressed, and releasing it. This data represents a freehand curve that may consist of several sub-curves which are expected to be segmented as line segments, circular arcs, elliptical arcs or free form curves. According to our observations, the drawing speed for each sub-curve will increase from zero, or a small value, to a normal value, then will remain steady, and finally will decrease when a segmentation point is approached. After the segmentation point, the speed will increase again. These form a speed cycle and a corresponding acceleration cycle for each sub-curve (Figure 1). Thus, the density of sample points around the segmentation points will become greater than their density in the middle of a sub-curve.

In order to estimate the drawing speed, we employ a distance between two adjacent points and a constant time interval for machine capturing events. If the time interval is assumed to be a unity, then the value of the distance can be used as a speed measure. The acceleration is determined from the rate of speed change.

When sketching is very slow, two consecutive digitized points are very close to each other. These extraordinarily close consecutive points will not only cause heavy computational effort, but will also bring problem in the computation of tangents to the sketched curve. For this reason, we apply a simple filtering scheme to eliminate such unnecessary points. A minimum threshold such as 3.5 units (screen pixels) is chosen for the distance constraint between two successive digitized points. All points within the minimum distance are deleted. Nevertheless, if this rule is applied to points near possible segmentation points, some candidate segmentation points might be missed. In order to keep the candidate points during the filtering process, a knowledge-based threshold rule is employed. That is, if a distance between two successive points is less than the threshold and the acceleration at the current point is non-negative, the current point will be eliminated. This pre-filtering reduces the number of points to be restored and processed, and ensures that the vector between two post-filtered successive points will give a good approximation for the sketching direction. In this way, we obtain a sequence of points $\{Q_i \mid i=1, 2, 3, \ldots, n\}$ with attributes of speed, acceleration and accumulative chord length. The data collection is simply denoted by $\{Q_i\}$ for convenience, where $i$ is a index.

### *2.2 Estimating directional deviation*

Corner points have an important property that the tangent to the contour at the point is discontinuous. We use directional deviations to reflect this property, and apply an adaptive threshold technique to find the corner points.

A directional deviation $\beta_i$ subtended by two unit vectors $\boldsymbol{V}_{i-1}$ and $\boldsymbol{V}_i$ at point $Q_i$ (Figure 2) is defined as

$$\beta_i = \arccos(\boldsymbol{V}_{i-1} \bullet \boldsymbol{V}_i), \tag{1}$$

where $\mathrm{V}_{i-1} = \dfrac{\overrightarrow{Q_{i-m}Q_i}}{\left\|Q_{i-m}Q_i\right\|}$, $\mathrm{V}_i = \dfrac{\overrightarrow{Q_iQ_{i+m}}}{\left\|Q_iQ_{i+m}\right\|}$, and

$$m = round\,(2{*}S_a / S + 0.5). \tag{2}$$

In (1) index $i$ starts from 2. An adaptive support length $m$ which defines the number of points from the central point $Q_I$, given with (2), is chosen to avoid occurrences of segmentation points being too close and to overcome unintentional drawing deviations producing sharp turns at several starting and ending points. The value of $m$ is determined as a non-linear adaptive function of an average sketching speed $S_a$ and a sketching speed $S$ at point $Q_i$ (eq.2). In this equation the average speed $S_a$ is equal to the total cumulative arc length divided by the number of steps $(n-1)$. It is clear from (2) that $m$ will be 1 if $S >> S_a$; and its value will be 3 if $S_a = S$, which avoids taking the neighbouring step point to evaluate the angle change.

The theoretical value of $\beta_i$ for lines is zero, and its value for curves is determined by two tangent lines at two adjacent points on the curve. The closer the two points the smaller the value of $\beta_i$.

The adaptive support length $m$ varies in narrow boundaries in order to reflect the speed changes. When sketching at relatively high speed, $m$ takes relatively smaller values. When drawing at relatively lower speed, $m$ has relatively greater values.

After determining the support length $m$ and computing the angle $\beta_i$ for each point, the system will perform nonmaxima suppression. Following the suppression, the angle $\beta_i$ will keep its original value if point $Q_i$ is a local maximum (dominant) point; the angle $\beta_i$ will be suppressed into zero if the point $Q_i$ is not such a point.

According to the directional deviation, we may classify corner points into two categories: acute corner points and obtuse corner points. As shown in Figure 3(a), we begin by drawing a line $L_1$ perpendicular to $\boldsymbol{V}_{i-1}$, and lines $L_2$ and $L_3$ symmetrical to the vector $\boldsymbol{V}_{i-1}$. If both $\boldsymbol{V}_{i-1}$ and $\boldsymbol{V}_i$ are located on the same side of line $L_1$, the corresponding angle $\beta_i > \pi/2$. In this case, the current point is an obtuse corner point denoted as of type C1, which is a real corner point with possibility or membership value of 1 as shown in Figure 3(b). If $\boldsymbol{V}_i$ is between the two lines $L_1$ and $L_2$, or between $L_1$ and $L_3$, then $\beta_0 < \beta < \pi/2$ (where $\beta_0$ is a threshold), the current point is classified as an acute corner point denoted as of type C2 (which may occur due to drawing speed, acceleration, and curve's linearity). If $V_i$ is between lines $L_2$ and $L_3$, the corresponding value $\beta_i < \beta_0$, and the current point is unlikely to be a real corner point, thus it has a smaller membership value, but it still might be a smooth join point.

Corner points are detected in two steps. The first step is to find obtuse corner points, and the second one is to dectect acute ones.

*2.3 Finding obtuse corner points*

The first point $Q_1$ will be assigned as the first obtuse corner point by default. Then, we compute a directional derivation $\beta_i$ at point $Q_i$ using (1). If $\beta_i > \pi/2$, we recognize the point $Q_i$ as an obtuse corner point and save the point index $i$ in a corner point index array $B$. The last point $Q_n$ will be assigned as the last obtuse corner point. The array $B$, $\{B(p), p=1, 2, \ldots, N\}$ will contain indexes for all $N$ obtuse corner points in the order of appearance, as shown in Figure 4.

*2.4 Detecting acute corner points*

We detect acute corner points within each span of obtuse corner points $Q_{B(p)}$ and $Q_{B(p+1)}$, where p = 1, 2, … N-1. It was noted that the detection and segmentation results are not satisfactory when only a fixed threshold for the directional deviation $\beta$ is used for various drawing intentions and objects. Thus, an adaptive threshold for the directional deviation $\beta$ is proposed, considering the drawing speed, acceleration and object linearity. Let us define the linearity of a curve segment as a ratio of the distance between two segmentation points to the cumulative arc length between the two points. Linearity of the curve segment between obtuse corner point $Q_{B(p)}$ and the current point $Q_c$ is denoted $L_p$, and $L_n$ is the linearity of the curve segment between the $Q_c$ and $Q_{B(p+1)}$ (Figure 4). The adaptive threshold $\beta_t$ consists of three parts: basic threshold $\beta_0$, speed adaptive tolerance $\beta_s$, and linearity adaptive tolerance $\beta_l$. These adaptive tolerances are represented by their non-linear functions that are based on heuristic knowledge and computational efficiency, and can reflect a user's intention. As a result, we have chosen the triangular functions shown in Figure 5.

The basic threshold $\beta_0$ reflects the drawing skills and pixel-based drawing features. Theoretically speaking, the directional derivation should be zero for a straight line, but it is usually different from zero for a digitised straight line, because a digitised curve is just an approximation of its corresponding mathematical curve, and it has some round-off errors. Therefore, $\beta_0$ is chosen based on the average directional derivation angles of normal sketches (excluding the segmentation points). The speed adaptive tolerance $\beta_s$ (Figure 5(a)) depends on the drawing speed. Based on observations, the faster the sketching the smoother the curve, thus, the directional derivation will give more accurate estimation of the true geometric parameters. So, the faster the sketching speed the smaller the angle tolerance. As a result, we use an adaptive function of $S$ to obtain $\beta_s$, which is:

$\beta_s = (10-S)*2+1$; and if $S > 10$, $\beta_s = 1$.

Another tolerance is the linearity adaptive tolerance $\beta_l$ (Figure 5(b)), because directional derivations are different for lines and curves, thus, the tolerance with linearity $L_p*L_n$ should have a smaller value for an angle between two lines, than between one line and one curve, or between two curves. Before the sketches are interpreted, we use the linearity of a sub-curve to evaluate the possibility of being a straight line. The threshold $\beta_l$ is computed from a non-linear function of variable $L_p* L_n$, which is $\beta_l = 10+(1- L_p* L_n)*50$.

Considering the speed and acceleration constraints, we noted that when drawing close to the position that corresponds to a corner point, the sketching speed tends to be lower than or equal to the average speed $S_a$ as shown in Figure 1(a); it also varies with different sketched objects. To capture this feature, an adaptive speed constraint $\vartheta$ (Figure 5(c)), is introduced by using a triangle function with respect to $S_a$ and the linearity $L_p*L_n$. Regarding the acceleration constraint, as shown in Figure 1(b), a user will slow down the sketching before a pre-assumed segmentation point and then will speed up after that point. Thus, sketching at a segmentation point would be with negative or zero acceleration, which forms the acceleration constraints.

Subsequently, we apply the following rules to find acute corner points:

(1) Compute average speed $S_a$ for the current segment between two obtuse corner points;

(2) Generate an adaptive tolerance $\beta_s$ and basic tolerance $\beta_0$ ;

(3) Compute the linearities $L_p$ and $L_n$;

(4) Determine an adaptive tolerance $\beta_l$;

(5) Give angle threshold by $\beta_t = \beta_0 + \beta_s + \beta_l$;

(6) Compute speed threshold $\vartheta$;

(7) Decide if point $Q_i$ is an acute corner point by the conditions: $\{(\beta_i >= \beta_t)$ and $(acceleration <= 0)$ and $(speed < \vartheta)\}$.

(8) Save indexes of obtuse corner and acute corner points in an array D { D($f$), $f$ = 1, 2, …, $G$}. The array D will keep indexes for all G corner points in the order of index values.

## 3. Detection of smooth join points

In each span between two adjacent corner points $Q_{D(f)}$ and $Q_{D(f+1)}$, we examine smooth join points, using both curve curvature and curve linearity. We define the curvature $\boldsymbol{K}_j$ at point $Q_j$ as the cross product of two vectors $\boldsymbol{V}_{j-1}$ and $\boldsymbol{V}_j$ with the sign of the subtended angle:

$\boldsymbol{K}_j = \boldsymbol{V}_{j-1} \times \boldsymbol{V}_j$,

where $D(f) < j < D(f+1)$, and $\boldsymbol{V}_{j-1}$ and $\boldsymbol{V}_j$ are given with (1).

Based on the curvature $\boldsymbol{K}_j$, drawing direction and curvature magnitude are obtained. If all $\mathbf{K}_j > 0$, the drawing direction is counter-clockwise, and if all $\boldsymbol{K}_j < 0$, the drawing direction is clockwise. If $\boldsymbol{K}_j = 0$, the drawing direction is along a straight-line. There are two types of smooth join points: external and internal tangent join points. For the external case, two joined curves are on different sides of the tangent line, or one curve (line) is on the tangent line. In the internal case, both curves are on the same side of the tangent line. Our system ignores the internal tangent join points, because we deal with elliptical arcs and free-form curves, which have varied curvatures and infinitely many internal tangent join points. At external tangent points, curvatures change their signs or their value to 0, which means the convexity of the curve changes from convex to concave or vice versa. An example of a smoothly joined compound curve with changing $\boldsymbol{K}_j$ is shown in Figure 6. The sketch is on the right (the fitted curve is moved 2 units to the right) and the corresponding graph of $\boldsymbol{K}_j$ against the arc length is on the left. It can be seen that each smooth join point features the sign change of $\boldsymbol{K}_j$. For straight line sketching, the $\boldsymbol{K}_j$ varies from positive to negative territory around zero.

We use the dot product of $\mathbf{K}_j$ and $\mathbf{K}_{j+1}$ to detect candidates for smooth join points. The linearity of the curve is then used to delineate these candidates, because some of the candidates may be on a straight-line segment. Afterwards, we utilise the refinement process to further deal with those candidates, because some candidates are introduced due to sketching defects, such as hand vibration, which does not reflect the user's intention. The rules adopted are as follows:

(1) Determine if the linearity for a curve between $Q_{D(f)}$ and $Q_{D(f+1)}$ is larger than a threshold such as 0.95. If so, go to the next span;

(2) Detect candidates for smooth join points if dot product of $\boldsymbol{K}_j$ and $\boldsymbol{K}_{j+1}$ is equal to or less than zero;

(3) In the case of more than one candidate, the current one will be recognized as a true candidate only if the linearity of the sub-span between the current candidate and previous segmentation point is less than a threshold 0.95.

(4) Finally, eliminate extraneous candidates based on the arc length between the successive points. The current candidate will be deleted if the arc length to the previous one is less than 5% of the arc length

between the two adjacent corner points. Users can change this percentage according to their sketching skills and individual requirements.

Conditions 3 and 4 are applied to filter most of the high frequency inflection points due to hand or mouse vibrations. After detecting all candidates for smooth join points of the current span, a refinement process is conducted. Figure 7(a) shows an example of sketches with high frequency smooth join points along with an intended elliptical arc (dashed line). The refinement process of a candidate point $Q_s$ is illustrated in Figure 7(b).

The refinement process consists of the following four steps:

(1) Sub-divide the current span into sub-spans by the candidate points.

(2) Divide each sub-span by a step specified by the user, from 5% to 25% arc length of each sub-span, depending on the user's sketching skills and the precision requirement. Taking 25% as an example, consequently, five sub-dividing points $\{P_r \mid r = 0, 1, 2, 3, 4\}$ around the candidates can be obtained, as shown in Figure 7(b).

(3) Calculate the curve curvature $\boldsymbol{K}_t = \boldsymbol{U}_t \times \boldsymbol{U}_{t+1}$, $(t = 1, 2, 3)$ at a large scale, where $\boldsymbol{U}_t$ is a unit vector from $P_t$ pointing to $P_{t-1}$;

(4) The candidate $Q_s$ can be determined as a meaningful inflection point, only if the dot product of $\boldsymbol{K}_t$ and $\boldsymbol{K}_{t+1}$ is less than zero.

Finally, we have obtained all segmentation points (obtuse corner points, acute corner points and inflection points).

## 4. Classification

To find suitable 2D primitives that fit a segment of sketches, it is very important to classify a sub-curve into a line, a circular arc, or a free form curve.

The following steps and rules are adopted:

(1) Detect if the curve is a straight line by checking its linearity. If the linearity is greater than the threshold (e.g. 0.95), the sub-curve is a line (the threshold vary with the sketching skill); If not, the system goes to the next step;

(2) Determine if the curve is a free-form curve by checking for the existence of a smooth join point. If the curve has some smooth join point, the curve is a free-form curve. If not, the system goes to the next step;

(3) Determine if the curve is a free-form curve by checking for self-intersection. If not, the curve is a general conic section curve; the system goes to the next step;

(4) Classify a general conic curve into a circle, an arc, an ellipse or an elliptical arc by use of a general conic fitting and classification.

## 5. Results and Discussion

### *5.1 Case Studies*

Both, synthetic data and real-time sketches were generated to test the proposed procedure. A synthetic contour is created: a spiky shaped sketch of Figure 8. The synthetic shape is a combination of fifteen lines (drawn clockwise from the bottom), consisting of seven obtuse corners and seven acute corners. To obtain the synthetic data, a sequence of straight line segments was generated, and then filtered by distance threshold 3.5. The line sequence is specified in Table 1. A default value zero for the speed information was assigned to each point. Accordingly, the adaptive speed tolerance was ignored. These testing data have a segmentation result shown in Figure 8. Small circles in Figure 8 mark all acute corner points. The other segmentation points are obtuse corner points (unmarked). The result shows that the system successfully detects all obtuse and acute corners.

Figure 9 illustrates detection of obtuse corner points of on-line sketching. Dot graphs display initial sketches. Line graphs (shifted 2 units to the right for clear views ) show the results of segmentation and corresponding fits. Drawing speed varies from very slow (drawings on the lower-left side) to very fast (sketches on the lower-right side). Connected primitives include straight lines, arcs, elliptical arcs, and free-form curves. In the two left columns, drawings are produced by sketching in different directions (clockwise vs. counter-clockwise, top-to-bottom vs. left-to-right). This indicates that this system can work in direction-independent way. As it can be seen from this figure, all obtuse corner points are found.

Figure 10 is employed to give examples of finding smooth join points and acute corners. In this figure, the first drawing at the upper-left corner gives an example of a combination of detecting corners and finding smooth join points. Acute corners are marked with the smaller circles. Candidates of smooth join points are marked with short-vertical line bars. If the candidates are confirmed by the refinement process as real smooth join points, they will be marked with two concentric circles. The graphs show that some candidates of smooth join points are not real segmentation points (marked only with short-vertical line bars and without two concentric circles), because they do not match the users' intention. In the middle column, there are two drawings with extra acute points. These acute points are detected as unintentional, caused by fast "U"-turn drawing or mouse vibration. This implies that the system cannot guarantee 100% accuracy in detecting segmentation points.

Actually, it is hard for any system to find a perfect solution, because segmentation techniques are basically '*ad hoc'* and highly problem dependent. Thus, it is very demanding for a system to provide a way for correcting a wrong solution. Our system allows users to correct mistakes in segmentation and corresponding erroneous fits by interaction with the system.

*5.2 An example from CAD applications*

This on-line sketch segmentation technique has been applied in our on-line 3D-recognition system. After on-line sketches are segmented properly, the 3D-recognition system interprets 3D features based on 2D configurations of sketched input.

Figure 11 shows a shaded model of a camera, which is built incrementally from on-line 2D sketches and its corresponding 3D recognition.

**6 Conclusion**

It is difficult to make direct comparison with other methods for curve segmentation, since most of them are based on either chain-codes, or polygon approximation. On the other hand, they do not use any dynamic information. Nevertheless, here we give some qualitative comparison.

(1) According to the angle or curvature detection, this system can represent and accept arbitrary angles to be examined, whereas chain code based systems [12] can only present limited directions (4 or 8). This suggests that our system can work more accurately. The directional deviation $\beta_i$ is considered as only depending on the relative positions over the support length *m*, so, more robustness to noise effects is expected for this measurement. Moreover, the evaluation of each point is independent of the characteristics of the whole contour, taking only local properties into account. The self-adjustment of the support length *m* implies some advantages: (a) the algorithm works well for features of different sizes in the same contour, and (b) - it works automatically. Otherwise, if the support length is set to be very large, the algorithm will miss fine details, and if it is very small, the algorithm will be more sensitive to noise. By constrast, Chen and Xie use in their work [20] a fixed support length equal to 1, which implies that their method is more sensitive to noise than ours.

(2) Dominant points based methods [6], are concentrated at local maxima points of curvatures. Once the local maxima points are obtained, they are automatically regarded as segmentation points. Actually, some of them are not real segmentation points if segmented curve is considered at a large scale. The problem is that there are no adaptive evaluation measurements for an individual dominant point to check if it is a real segmentation point. As a result, these approaches detect extra segmentation points easily. In our system,

for each point an adaptive evaluation measurement (threshold: $\beta_0 + \beta_s + \beta_l$ and speed constraint $\vartheta$) is employed to check whether a local maximum point is a real corner, considering the drawing speed, acceleration, and linearity. Thus, the system has good potential to capture real segmentation points and users' intention. The segmentation points determined by our system are just a sub-set of dominant points. In comparison with [20], where the authors used some dynamic information in terms of drawing speed and acceleration by determing fuzzy memberships of *AngleSmall*, *AngleNormal* and *AngleLarge* (see their Figure 3). Unfortunately, they used a set of fixed values, such as 0, 30 and 60 as thresholds, to evaluate the memberships, which could be not sufficiently robust, because different primitives or even the same primitives with different scales will have different theoretical angle values. In contrary, we employ linearity adaptive tolerance to measure the difference in the theoretical angle values for the different primitives. It can be concluded from the above discussion, that selected nonlinear adaptive functions including the linearity tolerance are practical and useful.

(3) Edge approximation based approaches [14], [18] are dependent on the number of types of primitives to be recognised. If a free-form curve is included for recognition, these methods will have difficulties. Moreover, these approaches lead to heavy computation, making them unsuitable for on-line applications.

(4) In comparison with FFDS system [20], our classification procedure can improve computing efficiency by ($T$-$1$) times roughly (assuming $T$ types of curves to be classified), whether FFDS classifies a sub-curve by computing $T$ possibility values for a sub-curve. In our system, once a curve is classified (based on current criterion, such as linearity for a line), it is not necessary to check for another type. The FFDS also does not deal with smooth join points.

As a result of the comparison with the other methods, it can be concluded that our corner point detection algorithm does not involve heavy computation. Its ability to automatically vary the support region, $m$, and the adaptive threshold $(\beta_0 + \beta_s + \beta_l)$ enables the system to perform reliably and robustly even when the object consists of multiply sized features. This can be seen from the results shown in Section 5. Furthermore, its ability to interactively correct unintentional segmentation points makes it very practical.

Generally speaking, this knowledge-based system with nonlinear thresholding operations can interpret user's intention satisfactorily. From on-line sketches, the system can give proper segmentation and curve fitting in a variety of 2D shapes: straight lines, circles, arcs, ellipses, elliptical arcs, and free-form curves. Furthermore, those primitives are applied in 2D and 3D CAD applications. The system is suitable for dealing with vague and imprecise sketching information.

**Acknowledgements**

We would like to thank the anonymous referees for the valuable and very helpful comments improving the quality of this paper.

## References

[1] L. Eggli, C-Y. Hsu, B. D. Bruderlin, G. Elber, Inferring 3D models from freehand sketches and constraints, Computer-Aided Design 29 (2) (1997) 101-112.

[2] H. Lipson, M. Shpitalni, Optimization-based reconstruction of a 3D object from a single freehand line draw, Computer-Aided Design 28 (4) (1996) 651-663.

[3] T-S Hwang, D. G. UIIman, Recognize features from freehand sketches, Computer in Engineering 1 (1994) .67-78.

[4] D. K. Wright, S. F. Qin, Design framework of a 3D rapid geometric modeller, Proc. the 1998 Lancaster International Workshop on Engineering Design, Lancaster, UK, 1998, pp. 253-266.

[5] T. Pavlidis, S. L. Horowitz, Segmentation of plane curves, IEEE Trans. Computer 23 (8) (1974) 860-870.

[6] C. H. Teh, R.T. Chin, On the detection of dominant points on digital curves, IEEE Trans. Pattern Analysis and Machine Intelligence 11 (8) (1989) 859-872.

[7] Y. M. Sharaiha, N. Christofides, An optimal algorithm for the straight segment approximation of digital arcs, CVGIP Graph. Models and Image Processing 55 (1993) 397-407.

[8] W.Y. Wu, M.J. Wang, Detecting the dominant points by the curvature-based polygonal approximation, CVGIP Graph. Models and Image Processing 55 (1993) 79-88.

[9] B. K. Ray, K. S. Ray, A new split-and-merge technique for polygonal approximation of chain coded curves, Pattern Recognition Letters 16 (1995) 161-169.

[10] J.M. Chen, J. A. Ventura, C.H. Wu, Segmentation of planar curves into circular arcs and line segments, Image and Vision Computing 14 (1996) 71-83.

[11] W. Wan, J. A. Ventura, Segmentation of planar curves into straight-line segments and elliptical arcs, Graph. Model and Image Processing 59 (6) (1997) 484-494.

[12] O. Baruch, M. H. Loew, Segmentation of two-dimensional boundaries using the china code, Pattern Recognition 21 (1988) 581-589.

[13] W. Kim, R. H. Park, Contour coding based on the decomposition of line segments, Pattern Recognition Letters 11 (1990) 191-195.

[14] A. Albano, Representation of digitized contours in terms of conic arcs and straight-line segments, Computer Graph. Image Processing 3 (1974) 23-33.

[15] C. Ichoku, B. Deffontaines, J. Chorowicz, Segmentation of digital plane curves: A dynamic focusing approach, Pattern Recognition Letters 17 (7) (1996) 741-750.

[16] T. Pavlidis, S. T. Horowitz, Segmentation of plane curves, IEEE Trans. Computer, C-23,(1974) 860-870.

[17] J. A. Ventura, J.M. Chen, Segmentation of 2-dimensional curve contours, Pattern Recognition 25 (10) (1992) 1129-1140.

[18] P. L. Rosin, G. A. W. West, Non-parametric segmentation of curves into various representations, IEEE Trans. Pattern Analysis and Machine Intelligence 17 (12) (1995) 1140-1153.

[19] K. Kanatani, Non-parametric segmentation of curves into various representations-Comment, IEEE Trans. Pattern Analysis and Machine Intelligence, 19 (12) (1997) 1391-1392.

[20] C. P. Chen, S. Xie, Freehand drawing system using a fuzzy logic concept, Computer Aided-Design, 28 (2) (1996) 77-89.

## Figure list

Sheng-feng Qin*, David K. Wright[1], and Ivan N. Jordanov[2]

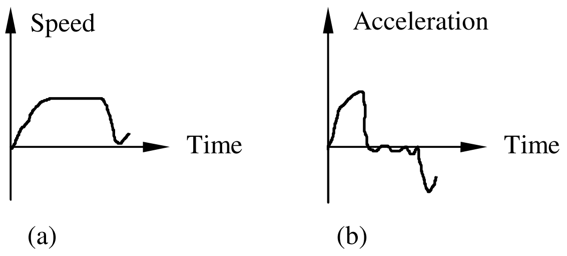

Figure 1. Speed and acceleration cycles:
(a) speed cycle, (b) acceleration cycle

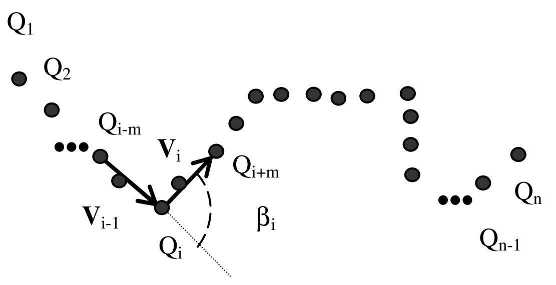

Figure 2. Representation of directional deviation $\beta_i$ with respect to data points and unit vectors

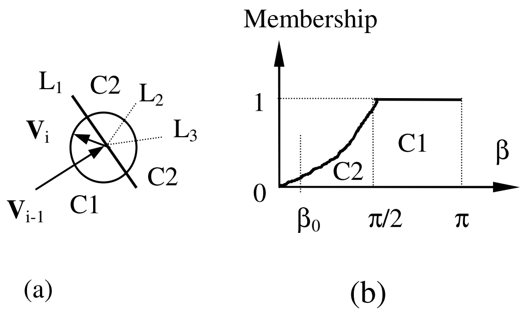

Figure 3. Classification and membership function for corner points:
(a) classification with vector positioning, (b) membership function

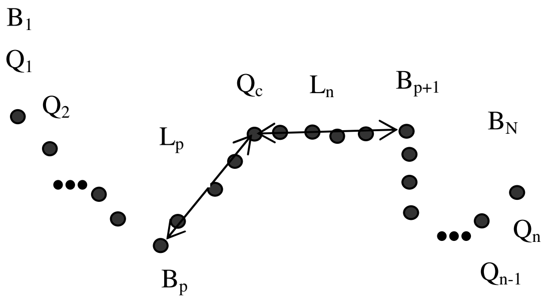

Figure 4. Representation of corner indexes with respect to data points and unit vectors

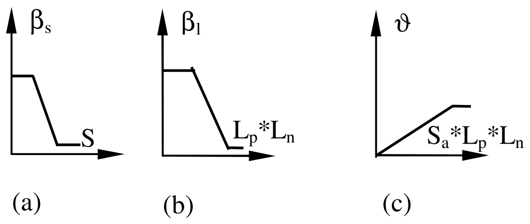

Figure 5. Adaptive threshold functions for $\beta_s$, $\beta_l$, and $\vartheta$
(a) speed threshold function, (b) linearity threshold function, (c) speed limitation threshold function

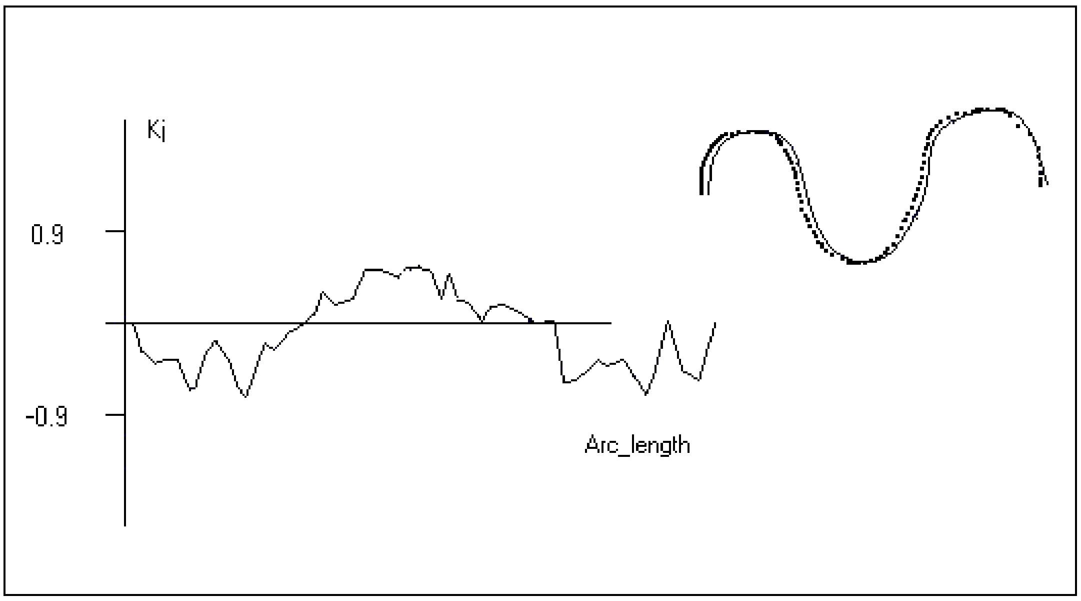

Figure 6. Curvature $\boldsymbol{K_j}$'s variation along a free-form curve.

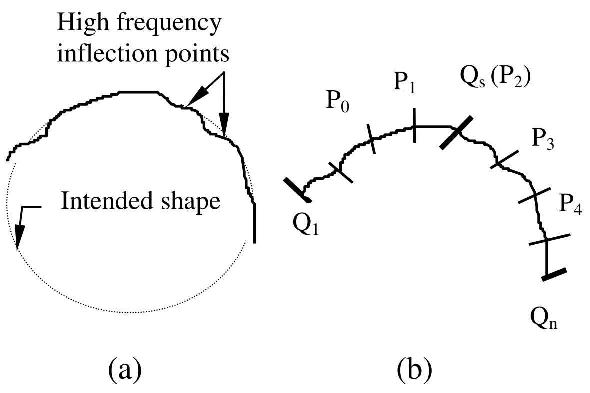

Figure 7. Refinement process for smooth join point detection: (a) high frequency inflection points, (b) refinement process for a candidate point

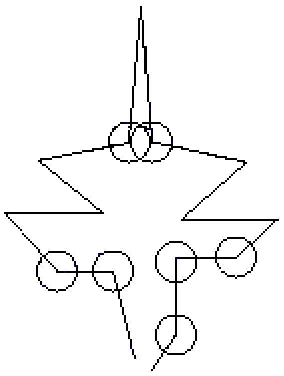

Figure 8. Segmentation of a spiky shaped sketch

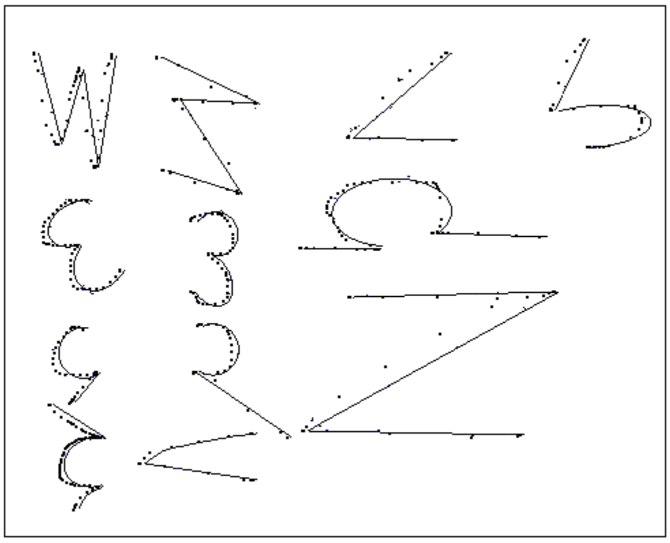

Figure 9. Detection of obtuse corner points

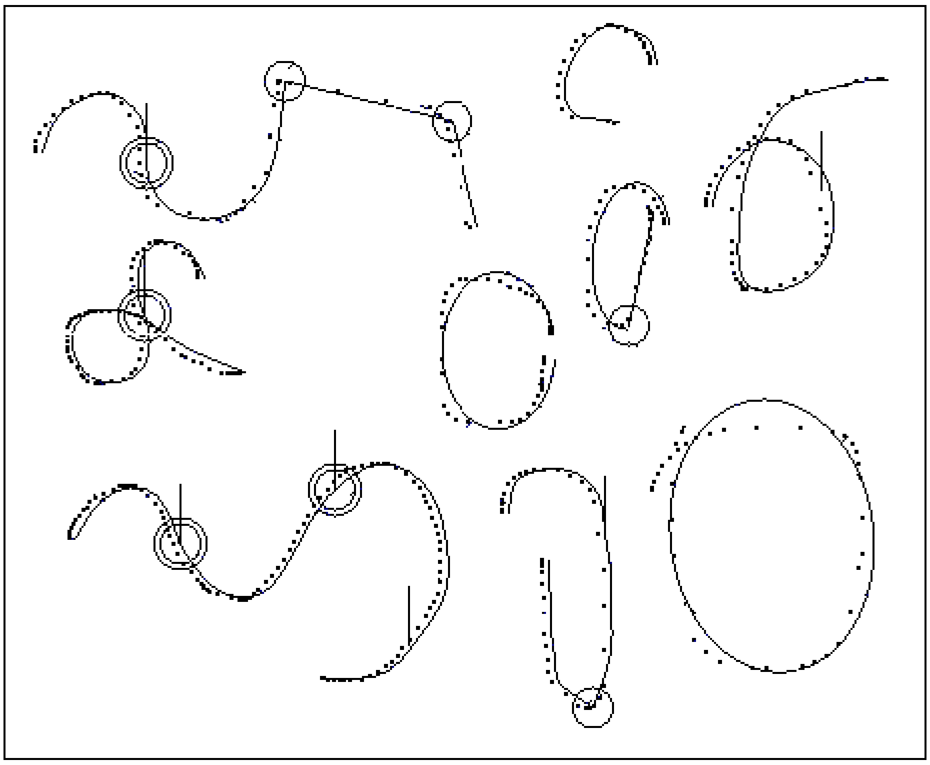

Figure 10. Detection of smooth join points and acute corners

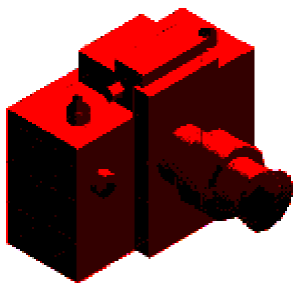

Figure 11. A shaded model of a camera built from 2D sketches

Table 1. Sequenced line segments for the spiky shaped sketch

| Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Direction (degrees) | 105 | 180 | 130 | 0 | 140 | 10 | 85 | -85 | -10 | -140 | 0 | -140 | 180 | -90 | -125 |
| Length | 50 | 30 | 40 | 50 | 45 | 50 | 70 | 70 | 50 | 45 | 50 | 30 | 30 | 40 | 25 |

Biographical Sketch

**Sheng Feng Qin** is a Research Associate at the Loughborough University. He gained his BEng (1983) and MSc (1988) in CAD from the Southwest Jiaotong University, China. He obtained his PhD from the University of Wales in 2000. He worked as a lecture and an associate professor in East China Jiaotong University from 1989 to 1996. His research interests include CAD/CAM, CIMS, feature-based modelling, and Multimedia Applications.

**David K. Wright** is currently a Reader in the Department of Design, Brunel University, and head of Interaction Design Research Group. He gained his PhD in Physics at the University of Warwick. His teaching and research interests are in CAD, computer simulations of biomechanical systems; virtual prototype of products; as well as CSCW.

**Ivan N. Jordanov** is a Senior Research Fellow at Design Engineering Research Centre, University of Wales Institute Cardiff, UK. He obtained BSc (1976), MSc  (1978) and PhD (1988) from the Technical University of Sofia (TUS), BG. His teaching and research activities include: Programming and Applications Development; Optimisation of Dynamic Systems; and Neural Networks Learning.