Intelligent Classification of Sketch Strokes

Shengfeng Qin, *Member, IEEE*

Abstract **— This paper presents an intelligent method for classifying pen strokes in an on-line sketching system. The method, based on adaptive threshold and fuzzy knowledge with respect to curve's linearity and convexity, can identify sketch strokes (curves) into lines, circles, arcs, ellipses, elliptical arcs, loop lines, spring lines and free-form B-spline curves. The proposed method has proven to be fast, suitable for real-time classification and identification.**

Keywords **— Fuzzy knowledge, sketch recognition, curve classification, curve fitting.**

I. INTRODUCTION

THE current CAD systems, operated by icons, tool-bars and menus, have not brought CAD close enough to product conceptual design stages in which designers use various sketches with vague and imprecise geometry to rapidly express their creative ideas.

This paper presents a sketch based interface in a CAD system to assist designers during conceptual design stages by attempting to capture the designers' intention and interpret sketches into more geometrically exact 2D vision and further into 3D models. The paper focused on the intelligent stroke classification.

The role on-line classification of sketch strokes is to convert the original digitized pen strokes in sketch into the intended 2D geometric objects and group them together it necessary. Rubine [3] investigated statistical pattern recognition techniques to specify gestures by examples. Jenkins and Martin [4] used a fit-and-test method to classify sketches. This system excluded ellipses and elliptical arcs. Chen and Xie [6] applied a fuzzy logic concept to approximate pen strokes. Yu [2] used mean shift to recognise freehand sketches. All the above method has their merits and flaws such as scalability.

In our research, the system gets a sequence of 2D input data from mouse button and movement events. From this data, information about the speed, acceleration, direction, angle, and accumulative chord length is extracted. This information is used to infer the user drawing intention, and then unintentional and redundant points are filtered. The filtered sketches are then segmented into several sub-curves if any (Qin et al, 2001). Finally, each of curve segments is classified and recognised. This paper presents details of classification and identification.

S. F. Qin is with the School of Engineering and Design, Brunel University, Middlesex, Uxbridge, UB8 3PH, UK; (e-mail: sheng.feng.qinj@brunel.ac.uk; phone:+44-1895-266335; fax: +44-1895-269763).

II. CLASSIFICATION

In order to find suitable 2D primitives for fitting a segment of sketches, it is very important to be able to correctly classify a sub-curve as a line, a conic curve or a free form curve. A curve classification follows a four-step procedure (Figure 1).

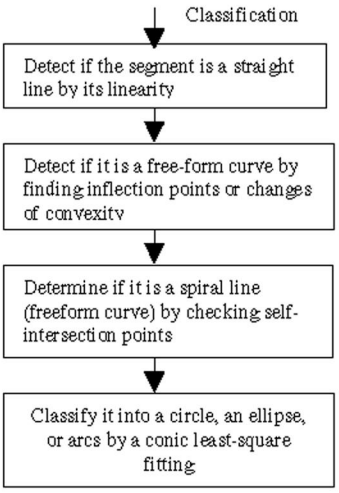


Figure 1. Curve classification procedure

A. Classification of lines

A curve was classified according to three preference orders: linearity (or straightness), convexity, and complexity of shape, not only by complexity, as in [4,6]. We considered the curve classification as an ill-condition problem, depending on applications. For example, a curve shown in Figure 2 (a) represents a straight line. But, if it is viewed at a fine scale, changes of convexity will be seen. Theoretically speaking, a curve with some changes of convexity cannot be a straight line or any conic section. To avoid this scale problem and quickly classify a curve as a straight line, we firstly computed the curve's linearity, and then used it to classify the curve. The linearity of a sketch segment is a value of the distance between two end points divided by its accumulative chord length between them. For example, the linearity for a strict straight line segment should be 1. So, before a sub-curve is interpreted, its linearity is used to evaluate the possibility to be a straight line. For instance, if a value of linearity is 0.95, it means that the corresponding curve has a 95% possibility to be treated as a line. In our system, if a curve's linearity is greater than a threshold of 0.98, the curve will be identified as a straight line. This threshold can be selected by users according to their drawing skills. In contrast to least squares line fitting and testing, computing linearity is easier. This linearity threshold can be computed by an

1-4244-0049-X/05/$20.00 ©2005 IEEE

adaptive threshold function of average drawing speed. If users draw the sketch very fast or roughly, the lower threshold is expected. If they draw the sketch carefully and slowly, the higher threshold should be exploited. This is because slow drawing implies that the users are paying more attention to each point, and hence, each point has a more accurate geometric meaning. Furthermore, the better the drawing skills, the lower the threshold.

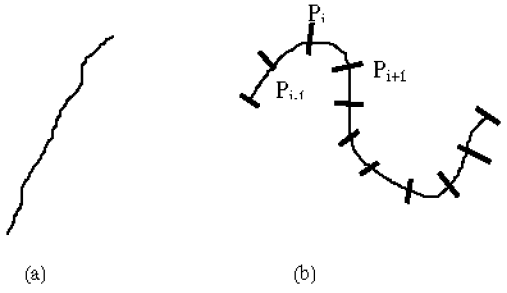


Figure 2. Linearity and convexity

B. Classification of free-form curves

Secondly, if a curve is not a straight line, we check whether it has changes of its convexity from convex to concave or vice versa by detecting inflection points and changes in the drawing directions. If the curve has some inflection points, it will be classified as a free-form curve. Sometimes, the system may fail to find inflection points; In this case, the system will further detect changes of convexity by the following steps (Figure 2 (b)):

(1) Sub-divide the curve by its arc length. Stepping length could be 5~10% of the total of its accumulative chord length. This step length could be selected by users in accordance with precision requirement and scales.
(2) Compute a convexity vector as a cross product of two vectors: $\mathbf{K}_i = \mathbf{V}_{i,i-1} \times \mathbf{V}_{i+1,i}$ where $\mathbf{V}_{i,i-1} = \mathbf{P}_i - \mathbf{P}_{i-1}$, $\mathbf{V}_{i+1,i} = \mathbf{P}_{i+1} - \mathbf{P}_i$. From the vector $\mathbf{K}_i$, the drawing direction and convexity of the shape can be received. If all $\mathbf{K}_i > 0$, they point out of the screen (or paper), which means the drawing direction is anti-clockwise along with a left-hand bend. If all $\mathbf{K}_i < 0$, the drawing direction is clockwise along with a right-hand bend. If $\mathbf{K}_i = 0$, the drawing direction is along a straight-line. At change points of convexity, the two adjacent $\mathbf{K}_i$ should change their signs, which means the convexity of the curve changes from convex to concave, or vice versa.
(3) Use the dot product of $\mathbf{K}_i$ and $\mathbf{K}_{i+1}$ to detect changes of convexity, if the product is negative;
(4) Classify a curve as a free-form curve if it has some changes of convexity.

Thirdly, if there are no inflection points in a curve and no changes in its convexity, but some self-intersection points exist, the curve would be identified as a free form curve representing a loop curve. If any of the arc lengths, which are out of the closed segment, is greater than 20% of the total arc length, the curve should be a free-form curve. Figure 3 (a) shows a normal over-drawn case, in which the two arcs lengths, out of closed segment, are less than the threshold. Figure 3(b), 3(c), and 3(d) give examples of loop curves, in which at least one arc length is greater than the threshold on *one side* and on *two sides*.

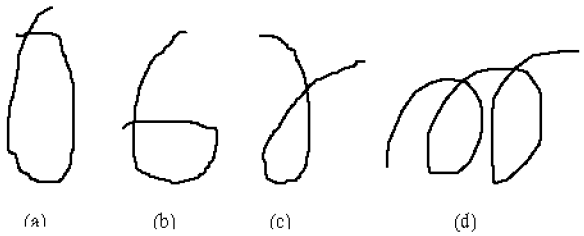


Figure 3. Classification of conic sections and freeform curves.

In order to find if a curve has self-intersections, a simple procedure with two nested loops is adopted. For the outer loop, let the variable i go from the start point to the end point with a step of 1. For the inner loop, let the other variable j go from $i+2$ and increments by 1, until the end point is reached. Within the inner loop, we first find the intersection point between line segments (P_i, P_{i+1}) and (P_j, P_{j+1}). If the intersection point is on either of the two line segments, a self-intersection point is found. Then we record the intersection position at points i and j, and assign $i=j+2$ to break the inner loop and continue in the outer loop. Finally, all self-intersection points are extracted. This procedure is quite simple and practical.

C. Classification of conics

Finally, if a curve is not a freeform curve, it will be fitted with a general conic equation (see details in Section 3). It can then be further classified as a free form curve, an ellipse (including a circle), an arc, and a hyperbola, or a parabola. This step mixes the fitting and classification.

III. Identification of Conic and B-spline Curves

After the classification, each curve should be fitted with a meaningful 2D primitive or a B-spline curve to represent its corresponding sketching points. In general, let the number of sketch points be n.

A. Conic curves

After a sketched segment is considered as a conic curve, the problem remaining is to fit a conic section to a set of sketching points $\{S_j, j=1,\ldots, n\}$. A general conic section can be described by the following equation:

$$Q(x, y) = ax^2 + 2hxy + by^2 + 2gx + 2fy + c = 0.$$

Many parameter estimation techniques [7] in vision community are developed for finding the coefficients of this equation. In this paper, we investigated the weighted Least squares fitting based on algebraic distances, because

geometric distances are difficult to evaluate. Thus, the fitting problem is to minimize a function,

$$E = \sum_{1}^{n} w_i(ax_i^2 + 2hx_iy_i + by_i^2 + 2gx_i + 2fy_i + 1)^2.$$

Where w_i is a weight for point (x_i, y_i). We assume that w_i is positive. It is relatively large for low speed drawing, and relatively small for high speed drawing because human intention represented at each point, in terms of speed, is different. Normally, the higher the drawing speed, the less the contribution of the intention is. To obtain the minimum value of *E*, its five partial derivatives with respect to parameters: *a, h, b, g,* and *f* are set to 0. We then obtain five simultaneous linear equations (the symbol Σ implies summation for *i* from 1 to *n*):

$$\begin{aligned}
&(\sum w_i x_i^4)a + (\sum w_i x_i^3 y_i)h + (\sum w_i x_i^2 y_i^2)b \\
&+(\sum w_i x_i^3)g + (\sum w_i x_i^2 y_i)f + (\sum w_i x_i^2) = 0, \\
&(\sum w_i x_i^3 y_i)a + (\sum w_i x_i^2 y^2{}_i)h + (\sum w_i x_i\, y_i^3)b \\
&+(\sum w_i x_i^2 y_i)g + (\sum w_i x_i\, y^2{}_i)f + (\sum w_i x_i\, y_i) = 0, \\
&(\sum w_i x_i^2 y^2{}_i)a + (\sum w_i x_i\, y^3{}_i)h + (\sum w_i y_i^4)b \\
&+(\sum w_i x_i\, y^2{}_i)g + (\sum w_i y^3{}_i)f + (\sum w_i y^2{}_i) = 0, \\
&(\sum w_i x_i^3)a + (\sum w_i x_i^2 y_i)h + (\sum w_i x_i\, y_i^2)b \\
&+(\sum w_i x_i^2)g + (\sum w_i x_i\, y_i)f + (\sum w_i x_i) = 0, \\
&(\sum w_i x_i^2 y_i)a + (\sum w_i x_i\, y^2{}_i)h + (\sum w_i y_i^3)b \\
&+(\sum w_i x_i\, y_i)g + (\sum w_i y^2{}_i)f + (\sum w_i y_i) = 0,
\end{aligned}$$

By solving the above equations, the coefficients (*a, h, b, g,* and f) can be obtained. Once these coefficients are found, we use the equation for computing the LS fitting error. If the mean fitting error over *n* points is greater than a threshold value of 0.001 (obtained from tests), the sub-curve will be classified as a free-form curve. If not, categorising a given conic (1) into one of the three possible forms, can be carried out using the following three invariants (Bowyer and Woodwark, 1983):

$$\begin{aligned}
\Delta &= a(bc - f^2) - h(hc - gf) + g(hf - gb), \\
\delta &= ab - h^2, \\
s &= a + b
\end{aligned}$$

If $\Delta = 0$, then the conic degenerates into a line(s), or a point (which may not always exist), otherwise:
if $\delta < 0$, the conic is a hyperbola;
if $\delta = 0$, the conic is a parabola;
if $\delta > 0$ and $\Delta s < 0$, the conic is an ellipse.

The five parameters of a general ellipse: the central point (X_C, Y_C), the two radii (R_a, R_b) and the directional angle θ of its major axis, can be received as follows :

$$\begin{aligned}
x_c &= (bg - fh)/(h^2 - ab), \\
y_c &= (af - gh)/(h^2 - ab),
\end{aligned}$$

$$\begin{aligned}
\theta &= \frac{1}{2} arctg\,(\frac{2h}{a-b}) \\
R_a &= (-c'/a')^{1/2} \\
R_b &= (-c'/b')^{1/2}
\end{aligned}$$

where

$$\begin{aligned}
a' &= a\cos^2\theta + b\sin^2\theta + 2h\sin\theta\cos\theta \\
b' &= a\sin^2\theta + b\cos^2\theta - 2h\sin\theta\cos\theta \\
c' &= ax_c^2 + by_c^2 + 2hx_cy_c + 2gx_c + 2fy_c + c
\end{aligned}$$

If the ratio of $R_a/R_b \approx 1$, we classify the ellipse as a circle, and simply take the centre of the ellipse as its centre. The average of R_a and R_b, is then the circle radius.

B. Freeform curves

For freeform curves, a B-spline curve is usually used to fit a set of points, it has flexible local control properties. There are two main ways for curve fitting: interpolation and approximation. In general, the interpolation is easier than the approximation. When interpolating, the number of control points is automatically determined by a chosen degree and the number of data items. The corresponding knot vector can be obtained in advance. Also, there is no curve error to be checked. When approximating, we don't know in advance how many control points are required to obtain the desired accuracy ε; the value of ε is usually not known, and hence the approximation methods are generally iterative (Piegl, 1995). For real-time application, interpolation approaches are more suitable.

For a given set of sketch points $\{S_j\}$, we intent to interpolate several key points $\{D_i\}$, $i = 0, 1, \ldots, m$ of $\{S_j\}$ with a *p*th degree B-spline curve, that is

$$C(u) = \sum_{i=0}^{m} N_{i,p}(u) P_i$$

where P_i are the $m+1$ unknown control points. $N_{i,p}(u)$ is *p*th-degree B-spline basis function.

If we give a parameter value $\hat{u}_k$ for each key point D_k, and select an appropriate knot vector $U=\{u_0, u_1, \ldots, u_{m+p+1}\}$, we can build up $(m+1) \times (m+1)$ system of linear equations

$$D_k = C(\hat{u}_k) = \sum_{i=0}^{m} N_{i,p}(\hat{u}_k) P_i \qquad (1)$$

In our system, we chose cubic B-spline fitting, that is, p=3. For the key points, we prefer to use as few as possible, in order to fit the sketched curve with respect to the computational efficiency. Each curve between two adjacent inflection points (including two end points and change points in convexity) is sub-divided into *SEG* segments with an equal arc length. Consequently, two end points and all (*SEG*-1) sub-dividing points are considered as key points. Clearly, for each sub-curve, there are (*SEG*+1) key points. Thus, the number of key points for a whole curve, *m*, can be determined by the number of sub-curves. The default value for *SEG* is 8, but it can be selected by the user.

To solve equation (1), a knot vector must be determined in advance. We assume that the parameters in the knot vector lie in the range [0, 1]. In this system, we

use a chord length based method with average technique, recommended by [7], to construct the knot vector. First $\hat{u}_k$ is computed by

$$\hat{u}_0 = 0, \hat{u}_n = 1$$
$$\hat{u}_k = \hat{u}_{k-1} + \left| l_k - l_{k-1} \right| / d, k = 1,2,..., m-1, \qquad (2)$$

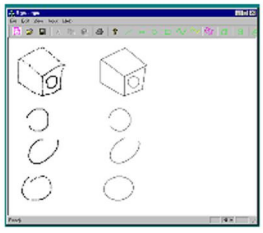
Where d is the total chord length, and l_k is accumulative chord length at point k. Then the following technique of averaging is adopted

$$u_0 = u_1 ... = u_p = 0, u_{m+1} = ... = u_{m+p+1} = 1,$$
$$u_{j+p} = \frac{1}{p} \sum_{i=j}^{j+p+1} \hat{u}_j, j = 1,2,..., m-p. \qquad (3)$$

Finally, The equations (1), (2) and (3) are combined to determine the control points and fitted cubic B-spline.

IV. Experimental Results and Discussions

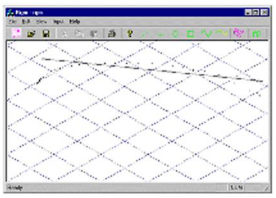
This simple and robust classification approach has been implemented in a sketch-based conceptual design system. Figure 4a is employed to demonstrate identification of lines, arcs, circles, ellipses and elliptical arcs (sketches to the left and fitted curves to the right). The top sketches show examples of straight lines and a whole circle. The next two sketches illustrate the application of the approach for general conics fitting and classification of arcs and elliptical arcs. The last one gives an over-drawn case of an ellipse with an open gap between starting part and ending part. The system classify it a close curve by computing its starting and ending angles properly. Figure 4b gives examples of loop curve classifications by using self-intersection points. The top sketch is with short arc lengths, out of the closed section, it is treated as a normal over-drawn case, and classified as an ellipse. The next is with a long arc length, out of the closed section, it is regarded as a free form curve. The middle one demonstrates loop curve fitting with multiple self-intersections. The bottom sketch is without any inflection or self-intersection point, and would be fitted with a LS ellipse, but the error of this fitting is greater than a given threshold 0.001, thus, it is finally fitted with a B-spline curve. The sketch just above the bottom one, is with some inflection points or changes of convexity, and is identified as a free-form curve with a B-spline fitting.



(a) Primitives

(b) Loop curves

Figure 4. Curve classifications



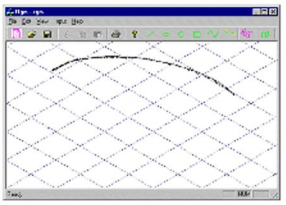


Figure 5a. Fast sketching Figure 5b. Slow sketching

Figure 5a and Figure 5b are used to show the effects of the drawing speed. The two sketches look like arc shapes with similar curvatures. However, the first sketch was drawn very fast, which usually means that users pay relatively little attention to it. The weighting for it was a lower value and finally, it was classified as a straight line. The second sketch was produced slowly, which means that users intended to draw an arc. Thus, bigger weightings were applied to it, and it hence was fitted with an arc. In this way, the system can intelligently interpret the users' intention.

V. Conclusion

This classification method employs some heuristic knowledge in terms of linearity and existence of inflection points, or changes of convexity, in order to quickly classify straight lines and free form curves, and then let only conics alone. The results show that it is very practical. Adaptive weighting and threshold scheme for linearity add a useful feature for capturing user intent. Thus, the system can intelligently classify curves.

Furthermore, the proposed method can distinguish real free form curves from various over-drawn cases of conics, by checking self-intersection conditions. This allows users to draw in a more natural way in terms of various over-drawn sketching. In the published work of [6], [4], It seems that no attention was given to over-drawn cases.

The examples show that for real-time sketches, the system can give proper classification and curve fitting in variety of 2D shapes: straight lines, circles, arcs, ellipses, elliptical arcs, spring and loop lines and free-form curves. This technique is therefore suitable for dealing with vague and imprecise information from sketching.

References

[1] A. Bowyer, J. Woodwark, A programmer's geometry, Butterworths. England, 1983.

[2] B. Yu Recognition of freehand sketch using mean shift. IUI'03, 204-210, 2003.

[3] D. Rubine Specifying gestures by examples. Computer Graphics, 25(4): pp. 329-337, 1991,

[4] D.L. Jenkins, R.R. Martin, Applying constraints to enforce users' intention in freehand 2-D sketches. J. Intelligent Systems Eng., Vol 1, pp. 31-49, 1992.

[5] L. Piegl, The NURBS book, Springer-Verlag, New York. 1995,

[6] P. Chen, S. Xie, Freehand drawing system using a fuzzy logic concept. J. CAD. 28, pp.77-89, 1996.

[7] P.L. Rosin, A note on the least squares fitting of ellipses. J. Pattern Recognition Lett. 14, pp.799-808, 1993.

[8] S.F. Qin, D.K. Wright, I.N. Jordanov, Online segmentation of freehand sketches by knowledge-based nonlinear thresholding operations. J. Pattern Recognition. 34, pp. 1885-1893, 2001.