Freehand Sketching Interfaces: Early Processing for Sketch Recognition

Shu-xia Wang, Man-tun Gao, and Le-hua Qi

Northwestern Polytechnical University, Xi'an, China, 710072
wangshuxia@nwpu.edu.cn

Abstract. Freehand sketching interfaces allow the user to directly interact with tasks without worrying about low-level commands. The paper presents a method for interpreting on-line freehand sketch and describes a human-computer interface prototype system of freehand sketch recognition (FSR) that is designed to infer designers' intention and interprets the input sketch into more exact 2D geometric primitives: straight lines, polylines, circles, circular arcs, ellipses, elliptical arcs, hyperbolas and parabolas. According to whether the stroke needs to be segmented or not, it is divided into single primitives and composite primitives correspondingly. Based on open/closed characteristic and semi-invariant, conic type and category of freehand sketch were defined for subdividing conic curve. Recognition approach for composite-primitive consists of three stages. The effectiveness of the algorithm is demonstrated preliminarily by experiments.

Keywords: Freehand sketching interface, Recognition, Segmentation, Stroke.

1 Introduction

Conceptual design is an early stage of the design process, in which designers use various sketches with vague and imprecise geometry to rapidly express their creative ideas. However, conceptual designers still tend to prefer paper and pencil to a computer aided design (CAD) system, for expression, communication and record of their ideas. The reason often given is that the user interfaces is not suitable for sketching design. Freehand sketching is a natural and powerful means of interpersonal communication. On-line freehand recognition is useful as a human-computer interface in situations where a human would draw sketches and the computer would interpret them. The goal of perceptual processing is to allow the user to perform complicated tasks with a minimum amount of explicit control. In traditional command-based interfaces, the user must decompose a task into a sequence of machine-understandable, fine command operations, and then input the commands one by one. Freehand sketching interfaces try to avoid this process and allow the user to directly interact with tasks without worrying about low-level commands.

Landay [1] described a sketch-based approach for the early stages of user interface design. His work allowed designers to quickly record design ideas in a tangible form. His electronic sketches possessed the advantages normally associated with computer-based tools, making them well suited to user interface design. Eggli [2] proposed a

J. Jacko (Ed.): Human-Computer Interaction, Part II, HCII 2007, LNCS 4551, pp. 161–170, 2007.

solid modeler incorporating a sketching tool; their system is three dimensional but the sketching itself is always constrained to some plane, thereby avoiding the problematic inverse-projection reconstruction phase. A similar system for designing solid objects using interactive sketch interpretation is described [3-5]. Several systems for modeling by sketching already exist. System Sketch [7] allowed the user to create simple geometrical scenes using a set of predefined gestures. There are also systems for sketching of CAD drawings [8]. These systems were a valuable help in creating models consisting of geometrical primitives, but they cannot be used to model free-form shapes, on which this work concentrates. Reference [9] was aimed especially on modeling of simple free-form objects. The system was able to inflate 2D closed curves into 3D objects, which can be consequently edited using simple gestures. During the whole modeling process the system maintained a polygon mesh of the model, displayed in a non-photo realistic way. Reference [10] presented a novel method for interpreting overtracing freehand sketch. The overtracing strokes were interpreted as sketch content and were used to generate 2D geometric primitives. The approach consisted of four stages: stroke classification, strokes grouping and fitting, 2D tidy-up with endpoint clustering and parallelism correction, and in-context interpretation. Strokes were first classified into lines and curves by a linearity test.

All of above methods intending to obtain new user interfaces, however, sketch recognition is important for a novel interface appearing. In this paper, we focus on the early processing for sketch recognition. In the section 2 we present an algorithm for stroke preprocessing by an adaptive threshold. Section 3 presents several classifiers for classification of single-primitives. If the stroke is composite primitive, the recognition approach is depicted in the section 4. Experimental results and conclusions are respective in the section 5 and 6.

2 Stroke Preprocessing

We adopt on the split and merge algorithm of polygonal approximation by [6]. This algorithm is composed of two steps. The split step consists in the subdivision of the stroke into line segments whose endpoints coincide with those of the original. When the maximum approximation error in terms of distance (ε_{max}) is no longer superior to a given threshold (ε_{12}) the subdivision process stops. The merge step consists in the fusion of two line consecutive segments, when the error inherent to the resulting line segment (ε_{max}) is not superior to the maximum admissible error (ε_{12}).

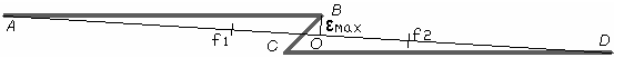


Fig. 1. ε_{12} is correlated with whether the error sign is changed $|f_1f_2|=2|Of_1|=L_1/3$

This algorithm depends on the given threshold ε_{12}. In fact, if the threshold value ε_{12} is constant can arose some problems in the recognition process [11]. We constructs a measure function for threshold value ε_{12} , which is related with the distance between first point and last point of stroke (L_1), length of stroke (L), pen width (D), and whether the error sign is changed in scope$[f_1, f_2]$ (See Fig.1). It is defined as

$$\varepsilon_{12} = C \cdot \left(L_1 / L + n \right) \cdot D, \tag{1}$$

where n is a coefficient (=0.5). In the first step, if the error sign is changed in the scope, let C=2.0, otherwise let C=1.0. In the second step, let C=2.0.

3 Single Primitive Recognition

To preserve the original information of sketch, our methods extend ploy line to primitive shapes. Primitive shapes are divided into polyline curve (straight line and polyline) and conic curve (circle, circular arc, ellipse, elliptical arc, hyperbola and parabola). We design several classifiers for classification of single-primitives in this section. Polyline classifier can recognize whether a stroke is polyline curve or conic section. If a stroke is conic, the recognition processes consist of three classifiers: open/closed classifier, residual error-based classifier, type and category classifier.

3.1 Polyline Curve Classifier

Given a stroke which contains sequences of sampling points $\{ p_i; i=0, 1, 2, \ldots, k\}$ and preprocessing points $\{ s_i; i=0, 1, 2, \ldots, u\}$ $(u \le k)$, where $p_i=(x_i, y_i) \in \{ p_i\}$. Let $d(q_{i,j})$ denote a perpendicular distances from the point $q_{i,j}(X_{ij}, Y_{ij})=p_{v(ij)} \in \{ p_i\}$ to the line segment that connects the points s_{i-1} and s_i $(1 \le i \le u)$, where $q_{i,j} =(X_{ij}, Y_{ij})=p_{v(ij)} \in \{ p_i\}$ and let $v(ij)$ denote a serial number that is expressed by Eq.(3). According to the distance definition for a point to a line, we have

$$d\left(q_{ij}\right) = \frac{\left| A_i X_{ij} + B_i Y_{ij} + C_i \right|}{\sqrt{A_i^2 + B_i^2}}. \tag{2}$$

$$v(ij) = \sum_{z=0}^{i-1} \left(N_z - 1 \right) + j, \quad 1 \le i \le u, \quad 0 \le j \le N_i - 1, \tag{3}$$

where N_i is the number of sampling points from s_{i-1} to s_i and let N_0=1. A measurement function of fitting error of polyline curve (ε_z) is obtained as

$$\varepsilon_z = \frac{1}{L} \cdot \sum_{i=1}^{u} \left(\frac{\sum_{j=0}^{N_i - 1} d\left(q_{ij}\right)}{N_i} \right), \tag{4}$$

where L is the length of stroke.

If fitting error of polyline curve ε_z is not superior to a maximum admissible error (ε_{z1}), the stroke is recognized as polyline curve, otherwise can be conic curve or composite primitives. Let u=1; we will interpret the polyline curve as a straight line. Otherwise the stroke is a polyline.

3.2 Conic Curve Classifier

If a stroke can be conic section, the recognition processes consist of three classifiers: open/closed classifier, residual error-based classifier, type and category classifier.

Open/closed classifier

Open/closed classifier is designed for detecting whether a stroke is conic section and whether is open or closed if the stroke is conic section. A closed conic can be circle or ellipses, otherwise can be circular arcs, elliptical arcs, hyperbolas and parabolas. The open/closed detecting method can cope with over-tracing sketching.

Let the central point be denoted as O, which is an average of the coordinates of sampling points of freehand conic curve. Given a preprocessing point s_i, a *radius* is the line segment that connects O and s_i. Constructing all radii that connect O to its consequent points s_0, s_1, s_2,…,s_u in turn, we have a sequence of radii, named as O–s_0, O–s_1, O–s_2, …, O–s_u. Let the symbol "<<" stands for the geometrical relation "left-of" and ">>" stands for "right-of" between any two continuous radii viewed in clockwise. We see that these radii form either one of two patterns:

1) in clockwise direction: O–s_0 << O–s_1 << O–s_2<<…<<O–s_u,
2) in counterclockwise direction: O–s_0 >> O–s_1 >> O–s_2 >>…>>O–s_u.

They are illustrated in Fig. 2.

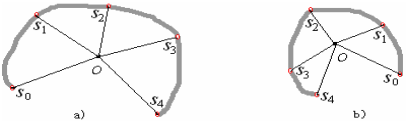


Fig. 2. Examples of the radii on a freehand conic curve: a) in the clockwise direction and b) in the counterclockwise direction

Through the above radii test, we can classify whether the stroke is not conic section, then we proceed the possible conic section by the following method. Let the angle value from the point s to the point p in the stroke be denoted as $O_{s,p}$ in Fig.3.a). We construct the open/closed algorithm for the possible curve section.

1) Let $O_{s0,su} < 1.85\pi$; the stroke is open curve and go to step 4.
2) Let $1.85\pi<O_{s0,su}<2.15\pi$; Firstly, calculate the distance from the first point s_0 to last point s_u. If the distance is small enough (determined by an adaptive threshold) the stroke is closed curve and go to step 4). Otherwise, if the stroke angle value $O_{s0,su}$ is less than 2π, the stroke is open curve and go to step 4.
3) Let $O_{s0,su}>2\pi$; search two points p_s and p_e which makes $O_{s0,ps}$ and $O_{su,pe}$ to approximate 2πin the sampling points $P=\{p_1, p_2, \ldots p_k\}$ (See Fig.3.b)). According to the distance definition for two points, we have Eq.(5). If $D(s_0, p_s)$ and $D(s_u, p_e)$ satisfy Eq.(4), the stroke is closed curve and go to step 4.
4) Stop.

$$D(s,z)=\frac{d(s,z)}{d(o,s)}<\frac{1}{4\pi}\cdot o_{s0,su}\ . \tag{5}$$

$$d(s,z)=\sqrt{(x_s-x_z)^2+(y_s-y_z)^2}. \tag{6}$$

Fig. 3. Examples for a angle value $O_{s,p}$ *in a)*and open/closed test for conic$|O_{s0,su}|>2\pi$in b)

Residual error-based classifier

A general conic section in the x-y plane is given by:

$$q(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0, \tag{7}$$

where real constant coefficients a, *b*, *c*, *d*, *e*, and *f* are not all zero. In practice, it can be assumed that f≠0 because we can easily program to avoid strokes passing exactly through the origin. The equation can therefore be normalized with *f*=1.

The strokes can be recognized as imaginary curves that have not path by the least-squares fitting. To avoid imaginary curves, this paper adopts Least Median Squares (LMS) algorithm that constraint conditions consider not only residual error square but also recognition result is not imaginary curve. The method effectively avoids vibration influence and enables the algorithm to robust. The least median square fitting conic curve model is represented as

$$\hat{q}_i = \hat{a}_i x_i^2 + \hat{b}_i x_i y_i + \hat{c}_i y_i^2 + \hat{d}_i x_i + \hat{e}_i y_i + f, \tag{8}$$

where $\hat{q}_i$, $\hat{a}_i$, $\hat{b}_i$, $\hat{c}_i$, $\hat{d}_i$, $\hat{e}_i$ are estimated values of q_i, a_i, b_i, c_i, d_i, e_i respectively. Using LMS obtain linear equations defined as

$$\begin{bmatrix} \sum_{j=1}^{m} x_{ij}^4 & \sum_{j=1}^{m} x_{ij}^3 y_{ij} & \sum_{j=1}^{m} x_{ij}^2 y_{ij}^2 & \sum_{j=1}^{m} x_{ij}^3 & \sum_{j=1}^{m} x_{ij}^3 y_{ij} \\ \sum_{j=1}^{m} x_{ij}^3 y_{ij} & \sum_{j=1}^{m} x_{ij}^2 y_{ij}^2 & \sum_{j=1}^{m} x_{ij} y_{ij}^3 & \sum_{j=1}^{m} x_{ij}^2 y_{ij} & \sum_{j=1}^{m} x_{ij} y_{ij}^2 \\ \sum_{j=1}^{m} x_{ij}^2 y_{ij}^2 & \sum_{j=1}^{m} x_{ij} y_{ij}^3 & \sum_{j=1}^{m} y_{ij}^4 & \sum_{j=1}^{m} x_{ij} y_{ij}^2 & \sum_{j=1}^{m} y_{ij}^3 \\ \sum_{j=1}^{m} x_{ij}^3 & \sum_{j=1}^{m} x_{ij}^2 y_{ij} & \sum_{j=1}^{m} x_{ij} y_{ij}^2 & \sum_{j=1}^{m} x_{ij}^2 & \sum_{j=1}^{m} x_{ij} y_{ij} \\ \sum_{j=1}^{m} x_{ij}^2 y_{ij} & \sum_{j=1}^{m} x_{ij} y_{ij}^2 & \sum_{j=1}^{m} y_{ij}^3 & \sum_{j=1}^{m} x_{ij} y_{ij} & \sum_{j=1}^{m} y_{ij}^2 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \\ e_i \end{bmatrix} = -f \begin{bmatrix} \sum_{j=1}^{m} x_{ij}^2 \\ \sum_{j=1}^{m} x_{ij} y_{ij} \\ \sum_{j=1}^{m} y_{ij}^2 \\ \sum_{j=1}^{m} x_{ij} \\ \sum_{j=1}^{m} y_{ij} \end{bmatrix}. \tag{9}$$

$$\min_{\hat{X}s} \underset{i}{med} \left(q_i - \hat{q}_i \right)^2. \tag{10}$$

Once the coefficients *a*, *b*...*f* have been solved by minimizing residual error square median value using Eq.(13), we can construct a measurement function of conic fitting error (ε_e) , which is calculated as follow:

$$\varepsilon_e = \frac{1}{L} \cdot \frac{1}{k+1} \cdot \sum_{i=0}^{k} |q_i|. \tag{11}$$

where *L* is the length of stroke. If conic fitting error ε_e is not superior to a maximum admissible error (ε_{el}), the stroke can be conic, otherwise is composite primitive.

Type and category classifier

Once these coefficients have been solved in Eq. (7), some conclusions regarding the conic curve may be derived using the following definition. Denote:

$$I_1 = a + c\,, \qquad I_2 = \begin{vmatrix} a & \frac{b}{2} \\ \frac{b}{2} & c \end{vmatrix}, \qquad I_3 = \begin{vmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{vmatrix}, \tag{12}$$

where I_1, I_2 and I_3 are conic curve invariants. This paper presents conic type and category of freehand sketch based on open/closed characteristics and invariants (See Table 1). In this table, ε_x is a type threshold, and ε_y is a circle threshold. If the stroke is closed through the open/closed classifier, $\alpha=0$, otherwise $\alpha=1$.

Table 1. Types and categories of freehand conic classified

Type	Category	Condition
Elliptic $(I_2 > \varepsilon_x)$	Ellipses	$I_1 \cdot I_3 < 0;\ \ I_1^2 - 4I_2 \geq \varepsilon_y;\ \ \alpha = 0$
	Elliptical arcs	$I_1 \cdot I_3 < 0;\ \ I_1^2 - 4I_2 \geq \varepsilon_y;\ \ \alpha = 1$
	Circles	$I_1 \cdot I_3 < 0;\ \ I_1^2 - 4I_2 < \varepsilon_y;\ \ \alpha = 0$
	Circular arcs	$I_1 \cdot I_3 < 0;\ \ I_1^2 - 4I_2 < \varepsilon_y;\ \ \alpha = 1$
Hyperbolic $(I_2 < -\varepsilon_x)$	Hyperbolas	$I_3 \neq 0$
Parabolic $(\lvert I_2 \rvert < \varepsilon_x)$	Parabolas	$I_3 \neq 0$

4 Segmenting Composite Primitive

If a stroke is not single primitive through the above method, enters into the processing of composite primitives. Recognition approaches consist of three stages: stroke classification, stroke segmentation and single primitive recognition.

4.1 Classification of Composite Primitive

While freehand sketching, a stroke is a single primitive commonly, and numbers of segment is not too much despite it is a composite primitive. This paper classifies composite primitive into non-ultra composite primitive or ultra-composite primitive based on convex-concave characteristic of freehand sketch such as Fig. 4. Non-ultra composite primitive is composed of single primitives. Ultra-composite primitive is composed of single primitives and non-ultra composite primitive. Classification of composite primitives effectively reduces the stroke complication and overcomes the question that stroke segmentation need long time.

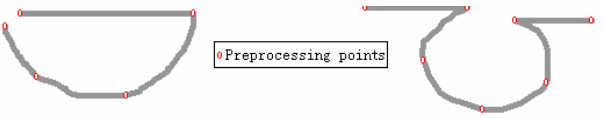


Fig. 4. Examples of non-ultra composite primitive at left and ultra-composite primitive at right

4.2 Segmentation of Ultra-Composite Primitive

Ultra-composite primitive is composed of single primitives and non-ultra composite primitive. So the aim of segmentation is that ultra-composite primitive is segmented as single primitives and non-ultra composite primitives. The algorithm of segmentation for ultra-composite primitive is followed.

1) Defined a sequence of vectors $\{v_i; i=1, 2, \dots, u\}$, where $v_i = \overline{s_{i-1}s_i} \in \{v_i\}$.
2) A sequence of signs $\{r_i; i=1, 2, \dots, u\text{-}1\}$, where r_i is the sign of $v_i \times v_{i+1}$.
3) Divide R into some groups where there are consecutive same sign. In Fig. 5$\{\{s_2\}\{s_3, s_4, s_5, s_6\}\{s_7\}\{s_8\}\{s_9, s_{10}, s_{11}, s_{12}, s_{13}\}\{s_{14}\}\}$.
4) Recognize segmenting points based on the numbers of preprocessing points is most in a unit length. In Fig. 4, the segmenting points:s_2, s_7, s_8, s_{14}.
5) Obtain the sub-strokes from the segmentation points. In Fig. 5, the sub-strokes are$\{\{s_1, s_2\}\{s_2, s_3, s_4, s_5, s_6, s_7\}\{s_7, s_8\}\{s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}\}\{s_{14}, s_{15}\}\}$.
6) Stop.

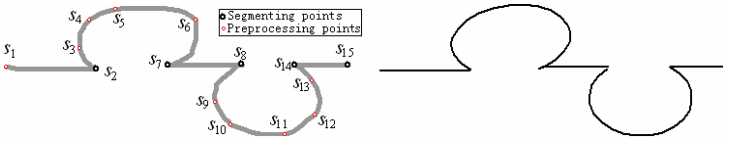


Fig. 5. Example of segmentation of ultra composite primitive: the recognition result at right

4.3 Segmentation of Non-ultra Composite Primitive

The aim of segmentation is that non-ultra composite primitive is segmented as several single primitives. The segmentation algorithm for ultra-composite primitive is expatiated by Fig. 6.

1) Defined a sequence of corresponding single span sub-stroke $L_0=\{l_1, l_2, \dots l_u\}$ where l_i represents a sub-stroke from preprocessing points s_{i-1} and s_i. And an empty sequence of single spans L_1.
2) Pop the first one from L_0 and push it into L_1until L_0 can be recognized as a single primitive using the methods in the section 3.
3) Obtain the segmentation point s_i that is the communal points of L_0 and L_1.
4) If L_1 can be recognized as a single primitive using residual error-based classifier in the section 4, go to step 6.
5) If not, L_1 is processed as a new stroke and go to step 1.
6) Stop.

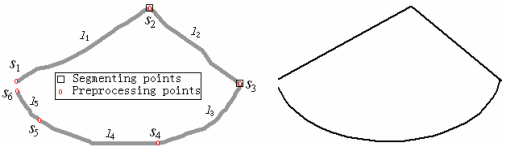
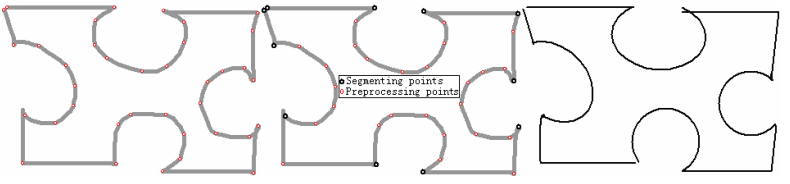


Fig. 6. At left the sequences of preprocessing points and corresponding single span sub-strokes are marked in original sketch and the result of segmentation at right



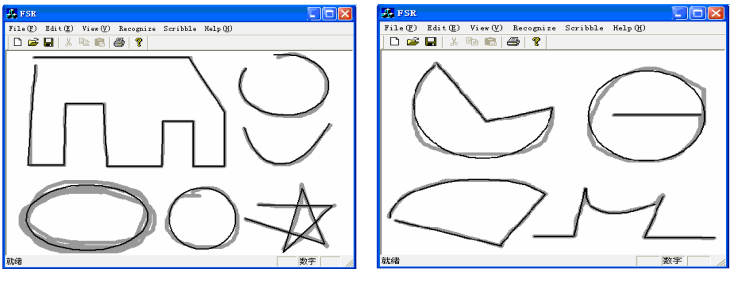
a) Stroke preprocessing b) Stroke segmenting c) Sub-strokes recognizing

Fig. 7. The process of composite primitive recognition

Through classification and segmentation of the composite primitives, it is segmented as several single primitives. Using the recognition of single primitives in the section 4, we can recognize the composite primitives (See Fig. 7).

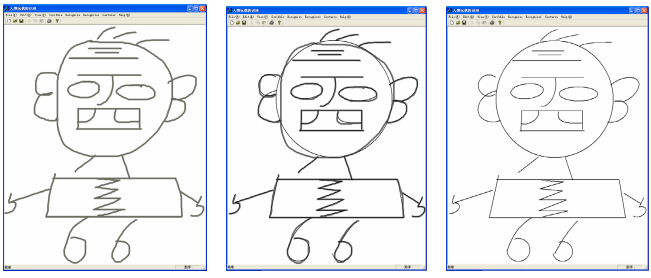
5 Experimental Results

The freehand sketch recognition system FSR developed in the above method uses the mouse to sketch on the screen. While sketching, data input is received as a sequence of mouse positions from pressing, moving the mouse button to releasing it. These data represent a freehand curve consisting of several single primitive. Fig. 8 shows examples of recognizing single primitive and composite primitive. A recognition result of a freehand sketch doll is shown in Fig. 9.



a) Single primitives recognition b) Composite primitives recognition

Fig. 8. Examples of recognizing single primitives and composite primitives



a) Sketch input b) Recognition process c) Recognition result

Fig. 9. Example of recognition result of a freehand sketch doll

6 Conclusion

The paper presents a novel method for interpreting on-line freehand sketch, and develops a human-computer interface prototype system (FSR) for assisting designers during conceptual design stages, which make system interface easy and friendly to use. According to whether the stroke needs to be segmented or not, it is divided into single primitives and composite primitives correspondingly. We design several classifiers for classification of single primitives. If a stroke can not be recognized as single primitives through the above method, it enters into the processing of composite primitives. Recognition approach consists of three stages: stroke classification, stroke segmentation and single primitive recognition. The system (FSR) has been tested with a number of sketched inputs of 2D geometry. The effectiveness of the algorithm is demonstrated preliminarily by experiments. Our recognition approach is based on stroke geometry characteristic and not non-geometry characteristic (speed, acceleration). Therefore it can make the user to sketch freely and easily and can be extended to the off-line recognition. Our method that tolerates high degrees of uncertainty and vague ideas does not rely on any domain-specific knowledge, and therefore it can be easily integrated with other high-level applications.

Acknowledgments. This work was supported by the grants from the Natural Science Foundations of Provinces of Shaanxi and Jiangxi (Project No.2002E$_2$24 and 0311018).

References

1. Landay, J.A., Myers, B.A.: Sketching interfaces: Toward more human interface design. IEEE Computer 34(2), 56–64 (2001)
2. Eggli L., Brüderlin B. P., Elber G.: Sketching as a solid modeler tool, Third Symposium on Solid Modeling and Applications, ACM SIGGRAPH, pp. 313–321 (1995)
3. Pugh, D.: Designing solid objects using interactive sketch interpretation. Computer Graphics Symposium on Interactive 3D Graphics 25(2), 117–126 (1992)
4. Hu, H.-q.: Freehand Sketch-based Three-dimensional CAD System, Zhejiang University (2004)

5. C.-x. Ma.: Research on Gesture Description and Sketch Design System in Conceptual Design PhD thesis, Chinese Academy of Sciences (2003)
6. Pereira, J.: Ambiguous calligraphic interaction for modeling environments, PhD thesis, FEUP (April 2004)
7. Zeleznik, R.C., Herndon, K.P., Hughes, J.F.: Sketch: An Interface for Sketching 3D Scenes. In: SIGGRAPH '96 Conference Proceedings, ACM, 30(4) 163–170 (1996)
8. Shpitalni, M., Lipson, H.: Classification of sketch strokes and corner detection using conic sections and adaptive clustering. Journal of Mechanical Design, Trans ASME 119, 131–135 (1997)
9. Igarashi, T., Matsuoka, S., Tanaka, H., Teddy: A sketching interface for 3D freeform design. In: Proceedings of SIGGRAPH 99 pp. 409–416 (1999)
10. Ku., D.C., Qin, S.-F., Wright, D.K.: Interpretation of Overtracing Freehand Sketching for Geometric Shapes, WSCG' (2006), Plzen, Czech Republic, 2 (2006)
11. Wang, S.-x., Gao, M.-t., Qi, L.-h.: Online Recognition of Freehand Sketching Using Conic Curves, Transaction of Northwestern Polytechnical University, 2 (2007)