Proc. of IMechE, Part B, J. of Engineering Manufacture, Vol 215, pp. 111-116, 2001.

A Conceptual Design Tool: Sketch and Fuzzy Logic Based System

Sheng-Feng Qin, David K. Wright* and Ivan N. Jordanov**

Department of Manufacturing Eng., Loughborough University, Loughborough, LE11 3TU,
*Department of Design, Brunel University, Surrey, TW20 0JZ,
**DERC, University. Of Wales Institute Cardiff, CF 5 2SG,

Abstract: A real time sketch and fuzzy logic based prototype system for conceptual design has been developed. This system comprises four phases. In the first one, the system accepts the input of on-line free-hand sketches, and segments them into meaningful parts by using fuzzy knowledge to detect corners and inflection points on the sketched curves. The fuzzy knowledge is applied to capture user's drawing intention in terms of sketching position, direction, speed and acceleration. During the second phase, each segmented sub-part (curve) can be classified and identified as one of the following 2D primitives: straight lines, circles, circular arcs, ellipses, elliptical arcs or B-spline curves. Then, 2D topology information (connectivity, unitary constraints and pairwise constraints) is extracted dynamically from the identified 2D primitives. From the extracted information, a more accurate 2D geometry can be built up by a 2D geometric constraint solver. The 2D topology and geometry information is then employed to further interpretation of a 3D geometry. The system can not only accept sketched input, but also users' interactive input of 2D and 3D primitives. This makes it friendly and easier to use, in comparison with 'sketched input only', or 'interactive input only' systems. Finally, examples are given to illustrate the system.

Key words: Conceptual design, fuzzy logic, sketch, curve segmentation, curve fitting, geometric modelling.

Notation

$\boldsymbol{Q}_i$	Position vector
β_i	Directional deviation at point $\mathbf{Q}_i$
β_0	Basic threshold
β_s	Adaptive speed tolerance
β_l	Adaptive linearity tolerance.
m	Support length
$\boldsymbol{V}_{i,j}$	Vector from point $\boldsymbol{Q}_i$ to point $\boldsymbol{Q}_j$
S	Sketching speed at point $\boldsymbol{Q}_i$
L_p	Forward linearity

L_n	Backward linearity
S_a	Average sketching speed over one stroke
ϑ	Speed constraint at point $\boldsymbol{Q}_{i.}$

1. Introduction

Conceptual design is at an early stage of the design process with the characteristics of fuzzy problems, tolerating high degree of uncertainty and vague ideas. Decisions made during this stage have significant influence on the cost, performance, reliability, safety and environmental impact of a product. Even the highest standard of detail design can never compensate a poor concept. Early design ideas are represented as 2D and 3D sketches, which may be vague and imprecise [1, 2, 3]. During this stage, designers search for structures, generate ideas, and try out different solutions, while at the same time these activities are guided by sketching out ideas [4]. As designers view a drawing, thoughts come to mind or model in progress, which can alter their perceptions and suggest new possibilities. The emerging representation allows them to explore avenues that could not be foreseen, and ideas are generated along the way [5]. Currently there exists little computational support for the early stages of geometric design. Various CAD systems, such as advanced parametric and feature based systems, have been developed to support 2D drafting and 3D modelling of products, but they usually require complete, concrete and precise definitions of the geometry, which are only available at the end of the design process. Also, conceptual designers still tend to prefer paper and pencil to CAD systems for effective expression, communication, and recording of new ideas. The reason for this includes low overhead of a single-tool interface (pencil), lack of special knowledge needed to draw, ease with which many kinds of changes can be made, and the fact that precision is not required to express an idea. Nevertheless, pencil and paper are still imperfect. After many changes, the paper can become cluttered, and also multiple access and input from distributed users is difficult. Drastic alterations such as showing the model from different viewpoints require new drawings, and collections of drawn objects cannot be transformed as a unit. In contrast, computer models do not have these disadvantages [6].
To support an early stage of geometric design and to improve the speed, effectiveness and quality of the design decision, studies in [7, 8] indicate that a Computer Aided Conceptual Design (CACD) systems must allow sketched input, must have a variety of interfaces, should be able to recognise features and manage constraints.

2. On-line Sketching and Segmentation

2.1 On Line Sketching

The input of design sketches into computers can be done by scanning paper-based sketches, or by on-line sketching. During the conceptual geometric design stage, first method can only enter final design results in the design sketch form for 3D reconstruction, and it can’t produce input-responses in real time to obtain interface effects of “What you draw is what you get“ in 2D and 3D. This means the designer can’t use feedback from one change to guide the next change, although this feedback is very useful for the conceptual design. Therefore, this input method was not adopted in our system, and instead, the second method was chosen. During sketching, information about drawing positions, directions and dynamic parameters, in terms of speed and acceleration, can be determined in real time. One stroke can be represented by a set of successive points with some dynamic parameters.

2.2 Data Collection and Filtering

While sketching, one obtains as input data, a sequence of mouse positions from pressing a button, moving the mouse while the button is still pressed, and from releasing it. The mouse positions are measured in the screen co-ordinate system with the lower-left corner as origin, one unit being a screen pixel. This data represents a freehand curve. In order to obtain the speed, the system uses a distance between two adjacent points, and a constant time interval for machine capturing events. If the value of the time interval is assumed as unity, then the value of the distance can be used as a speed measure. Subsequently, the acceleration is calculated on the basis of the speed.

2.3 On-line Segmentation

To allow sketched input in a more natural way, a stroke that includes more than one geometric primitive is acceptable in this system. Thus, perfect segmentation of sketch strokes into straight lines and other sub-curves is a prerequisite for obtaining the best sketch recognition and interpretation, because errors in segmentation might propagate to feature extraction and classification.

For the past two decades, many off-line algorithms based on polygonal approximation and dominant point detection have been developed, most of which use straight lines to approximate the edge pixels. The fit is made to reduce a chosen error criterion between the approximation and the original curve [9]. However, the piecewise linear approximation of digital curves is scarcely useful for segmenting curves in a meaningful and compact form [10]. Several works [10, 11, 12] have been published on the segmenting planar curves into straight lines and conic arcs. The approaches can be briefly divided into two major categories, namely edge approximation and break point detection. In the first method, curves are segmented into straight lines and conic arcs by repeatedly fitting and segmenting, based on some "goodness-of fit" [11]. The edge approximation procedures employed in recent research are mainly splitting (or region decomposition) and merging (or region growing), and split-and-merge [12]. In the break point detection ways, break points are usually detected at curvature extrema or at points of rapid curvature change [11, 10]. Generally speaking, the edge approximation method leads to heavy computation, whereas the break point detection approach is sensitive to noise. Also, all the above methods are focused on either line-fit or combinations of lines and general elliptical arcs fitting, which exclude free form curve descriptors.

Although many off-line algorithms have been proposed, most of them are not accurate enough or fast enough to be used in on-line systems [13]. On-line sketches have not only the properties of low-level images, but also have some dynamic features in terms of drawing direction, speed, and acceleration. One advantage of the on-line curve segmentation over the off-line segmentation is that dynamic information can be employed to assist the segmenting process. Another advantage is the close interaction between the user and the machine. Users can thus correct any recognition error immediately as it occurs. In this system, an intelligent and adaptive threshold segmentation technique is used, on the basis of fully exploiting the properties of dominant points and fuzzy heuristic knowledge in terms of sketching speed, and acceleration [14].

The fuzzy logic based segmentation algorithm can be briefly described in 4 steps:

Step1: Compute the directional deviation β_i at point $\boldsymbol{Q_i}$ with an adaptive support region, based on k-cosine curvature measure, and perform nonmaxima suppression [15]. A directional deviation β_i at point $\boldsymbol{Q_i}$, ranging from 0 to 180 degrees, is defined by a dot product of two-unit directional vectors, that is,

$\beta_i = \arccos(\boldsymbol{V}_{i-m,i} \bullet \boldsymbol{V}_{i,i+m})$, $i = m, m+1, m+2, \ldots, N-m$,

where

$$\boldsymbol{V}_{\mathrm{i-m,\,i}} = (\boldsymbol{Q}_i - \boldsymbol{Q}_{i-m}) \;/\; ||\boldsymbol{Q}_i - \boldsymbol{Q}_{i-m}||,$$

$$\boldsymbol{V}_{\mathrm{i,\,i+m}} = (\boldsymbol{Q}_{i+m} - \boldsymbol{Q}_i) \;/\; ||\boldsymbol{Q}_{i+m} - \boldsymbol{Q}_i||,$$

and m is defined as a support length (number of points from the central point $\boldsymbol{Q}_i$); N is the number of sketched points.

Step 2: Find obtuse corner point; if β_i is larger than 90 degree. The first and last points of a curve are regarded as default obtuse corner points;

Step 3: Detect acute corner points between two adjacent obtuse corner points by applying adaptive threshold and fuzzy knowledge, with respect to drawing speed, acceleration, and curve's linearity. Let us define the linearity as a ratio of the distance between two curve points to the accumulative arc length between the two points. An adaptive threshold for the directional deviation β_i at point $\boldsymbol{Q_i}$ consists of three parts: basic threshold β_0 (about 10 degrees), adaptive speed tolerance β_s (an adaptive function of sketching speed S at $\boldsymbol{Q_i}$), and adaptive linearity tolerance β_l (a non-linear function of the forward linearity Lp of a curve between the previous corner point and $\boldsymbol{Q_i}$, and the backward linearity L_n of a curve between $\boldsymbol{Q_i}$ and the next corner point). Point $\boldsymbol{Q_i}$ will be treated as an acute corner point, if the angle β_i is larger than *(*β_0

+ β_s + β_l), and speed and acceleration at this point can meet corresponding constraints. The fuzzy functions are shown in Figure 1.

Step 4: Determine whether a sub-curve between two neighbouring corner points, can be fitted with a straight line by computing its linearity. If yes, do line fitting, and then go to the next sub-curve. If not, detect inflection points within this sub-curve. If the sub-curve does have some inflection points, it will be treated as a B-spline.

3. Classification and Identification of 2D Primitives

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 three-step procedure (Figure 2). Here, classifying a curve is based on three preference orders [16]: linearity, convexity, and complexity of a shape, not only on complexity, as in [14, 17]. They use a fit-and-test method to classify sketches. The simplest primitive type is considered first, so their systems initially test if a straight line is a good fit to the stroke data. If this fails, a circular arc is next considered, and so on. The proposed classification can quickly classify curves and thus save system time. This makes it suitable for on-line applications.

After the classification, each curve should be identified and fitted with a meaningful 2D primitive or a B-spline segment to represent its corresponding sketching points, by a set of specific parameters [18]. In general, the fit is based on weighted least-square routines.

4. Generation of 2D Geometry

Initially during this stage, the fitted 2D primitives are firstly corrected into proper position, in accordance with their unitary relations. The unitary relations are properties of a single primitive on its own, and they apply to lines, ellipses, arcs, and elliptical arcs. For example, the system examines the slope of the straight line to see if it is close to one of a set of special directions: horizontal, vertical, isometric projection of principle axes. If it is close, the straight line will be assigned corresponding unitary relation codes and then will be adjusted to the special direction. Secondly, the system will correct 2D primitives' position according to their pairwise relations. Pairwise relations are geometric properties shared by two primitives. Currently, the system supports the parallelism and perpendicularity relations between two straight lines and two ellipses, or two elliptical arcs. Finally, the system connects these primitives under the constraints of connectivity.

5. Interpretation of 3D geometry

After the correction of the 2D primitives, the 2D geometry has its correct topology connections and the correct primitives themselves. The problem remaining is how to recognise 3D objects from 2D topology and geometrical information. From previous research [19], it is believed that design with features will bring some significant benefits for the design process itself and for the further manufacturing process. So, the system combines solid modelling methods with feature based design methods to develop a 3D inference engine for machined parts.

Our 3D recognition engine expresses recognition knowledge in knowledge rules, and integrates them into a programme by selection structure statements. The system examines combinations of 2D sketched elements and topology information (connectivity information from 2D) to infer a 3D feature. Different features have different inference rules; for example, general extrusion objects feature a closed profile and an extrusion edge. Therefore, the system first examines whether a closed profile exists, and then finds the direction of extrusion. Finally, the system determines where the closed profile comes from (reference plan) by checking if its centroid is within the projection area of a boundary plan of previous objects, or vice versa. Once a specific feature type is found, and the reference plan and extrusion direction are known, the system obtains all necessary 3D information and can produce a 3D feature. The 2D connectivity information here is used to find a closed profile and to determine which line is an extrusion line. For example, inference rules for box features are given in pseudo-code below:

```
Rules for a box feature:
 IF
- the feature is composed of a closed profile and one extrusion line AND
- the closed profile is composed of 4 lines (two pair parallel line) AND
- the extrusion direction has been determined (by the extrusion line) AND
- the reference plan has been found (default reference is XOZ, XOY, or YOZ plans corresponding to different extrusion directions)
THEN a box feature is defined.
```

In order to illustrate the essential features of the proposed approach, a prototype system has implemented box features, cylindrical features and simple revolution features.

6. Examples and Conclusions

This system has been implemented on Windows'95 by using Visual C++ and Open GL. On-line sketch segmentation and 2D curve identification are illustrated in Figure 3. The system first found four obtuse corner points (marked with small triangles). Secondly, it detected one acute corner point (marked with a small circle). Then it identified straight-line segments and further detected two inflection points (signed with small squares). Finally, it classified two non-line segments into an elliptical arc and a free-form curve correspondingly.

Figure 4 shows a conceptual model of a camera, which consists of a body box, zoom units (two cylinders), a flash-lamp (a box) and a button (a cylinder). The body box and zoom units are input by on-line sketches because of their greater size. An original sketch for the body box is shown in Fig. 4(a), and its 3D recognition result is in Fig. 4(b). Further sketch on the body box for a zoom unit (Fig. 4(c)) is interpreted as a cylinder (Fig. 4(d)). In the same way, the other zoom unit is created (a cylinder). The button and flashlight are, however, relatively small, and it is therefore easier to define them (Fig. 4(e)) by means of interactive 2D primitive input (for the button), and interactive 3D primitive's projection input (for the flashlight). The final recognised 3D model is given in Fig. 4(f).

The results show that the fuzzy logic based system can interpret users' intention on 2D and 3D geometry correctly and effectively. This system gives users greater freedom to quickly specify 2D and 3D geometry, comparing with those with sketched input only [19]. This mixed automatic and interactive design environment can encourage users with poor sketching skills to use it for creative design tasks. In principle, the system has the potential capability of supporting 3D surface design.

References

1 **Tovey, M.** Drawing and CAD in industrial design, *Int. J. Design Studies*, 1989, **10**(1), 24-39.

2 **Eisentraut, R., Günther, J,** and et al, Individual styles of problem solving and their relation to representations in the design process, *Int. J. Design Studies*,1997, **18**(4), 50-69.

3 **Tovey, M.** Styling and design: intuition and analysis in industrial design, *Int. J. Design Studies*, 1997,**18**(1), 5-31.

4 **Van Dijk, C.G.C.** New insights in computer-aided conceptual design, *Int. J. Design Studies*, 1995, **16**(1), 62-80.

5 **Dorsey, J.,** and **McMillan, L.** Computer Graphics and Architecture: State of the Art and Outlook for the Future, *ACM SIGGRAPH Computer Graphics*, 1998, **32**(1), 45-48.

6 **Zeleznik, R. C.** SKETCH: An Interface for Sketching 3D Scenes, *ACM SIGGRAPH Proceedings*, 1996, 163-170.

7 **Lipson, H.** and **Shppitalni, M.** Optimization -based reconstruction of a 3D object from a single freehand line drawing, *Computer-Aided Design*. 1996, **28**(4), 651-663.

8 **Eggli, L., Hsu, C.Y.** and et al, Inferring 3D models from freehand sketches and constraints, *Computer-Aided Design*, 1997, **29**(2), 101-112.

9 **Ramer, U.E.** An Iterative Procedure for the Polygonal Approximation of Plane Curves, *Computer Graphics and Image Processing,* 1972, **1**,244-256.

10 **Chen, J. M., Venture, J. A.** and **Wu, C. H.** Segmentation of Planar Curves into Circular Arcs and Line Segments, *Image and Vision Computing*,1996, **14**, 71-83.

11 **Albano, A.** Representation of Digitized Contours in Terms of Conic Arcs and Straight-line Segments, *Computer Graphics & Image Process*, 1974, **3**, 23-33.

12 **Rosin, P. L.** and **West, G. A.** Nonparametric Segmentation of Curves into Various Representations, TPAMI, 1995, **17**(12), 1140-1153.

13 **Wan, W.** and **Venture, J. A.** Segmentation of Planar curves into Straight-line Segments and Elliptical Arcs", Graphics Model and Image Process, 1997, **59**(6), 484-494.

14 **Chen, C.L.P** and **Xie, S**. Freehand drawing system using a fuzzy logic concept, *Computer-Aided Design*, 1996, 28 (2), 77-89.

15 **Qin, S. F**., **Wright, D. K**. and **Jordanov, I. N.** Intelligent Recognition of 2D Design Sketches, *In Proc. Of the IASTED Int. Conference on Intelligent System and Control (Ed. M. H. Hamza)* , Santa Bardara, CA. USA, 1999, pp. 246-251.

16 **Qin, S. F., Wright, D. K.** and **Jordanov, I. N.** Intelligent Classification of Sketch Strokes, *In Proc. Of CACUK'99 (Eds: H. Hu and M.H. Wu),* Derby, UK,1999, pp. 189-194.

17 **JenKins, D. L.** and **Martin, R. R.** Applying constraints to enforce users' intentions in free-hand 2-D sketches, *Intelligent System Engineering,*1992, **1**(2) , pp. 31-49.

18 **Qin, S. F., Jordanov, I. N.** and **Wright, D. K** Freehand drawing system using a fuzzy logic concept, *Computer_Aided Design*, **31**(5), 359-360.

19 **Hwang, T.** and **Ullman, D.** Recognizing features from freehand sketches, *Computers in Engineering*, 1994, **1**, 67-78.

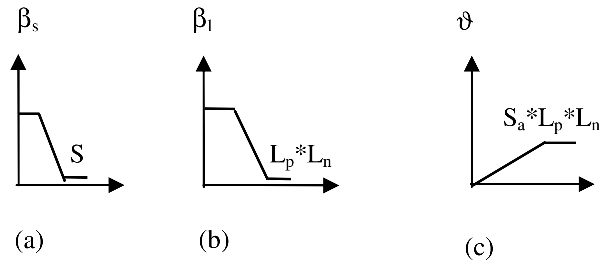

Figure 1. Fuzzy functions: (a) speed tolerance;
(b) linearity tolerance; (c) speed constraint

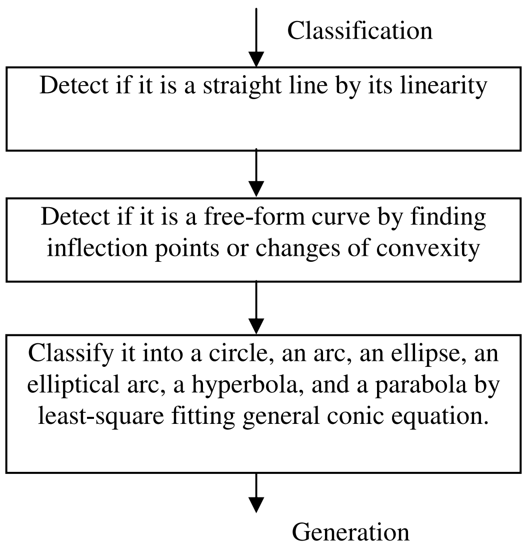
Classification
Detect if it is a straight line by its linearity
Detect if it is a free-form curve by finding inflection points or changes of convexity
Classify it into a circle, an arc, an ellipse, an elliptical arc, a hyperbola, and a parabola by least-square fitting general conic equation.
Generation

Figure 2. A curve classification procedure

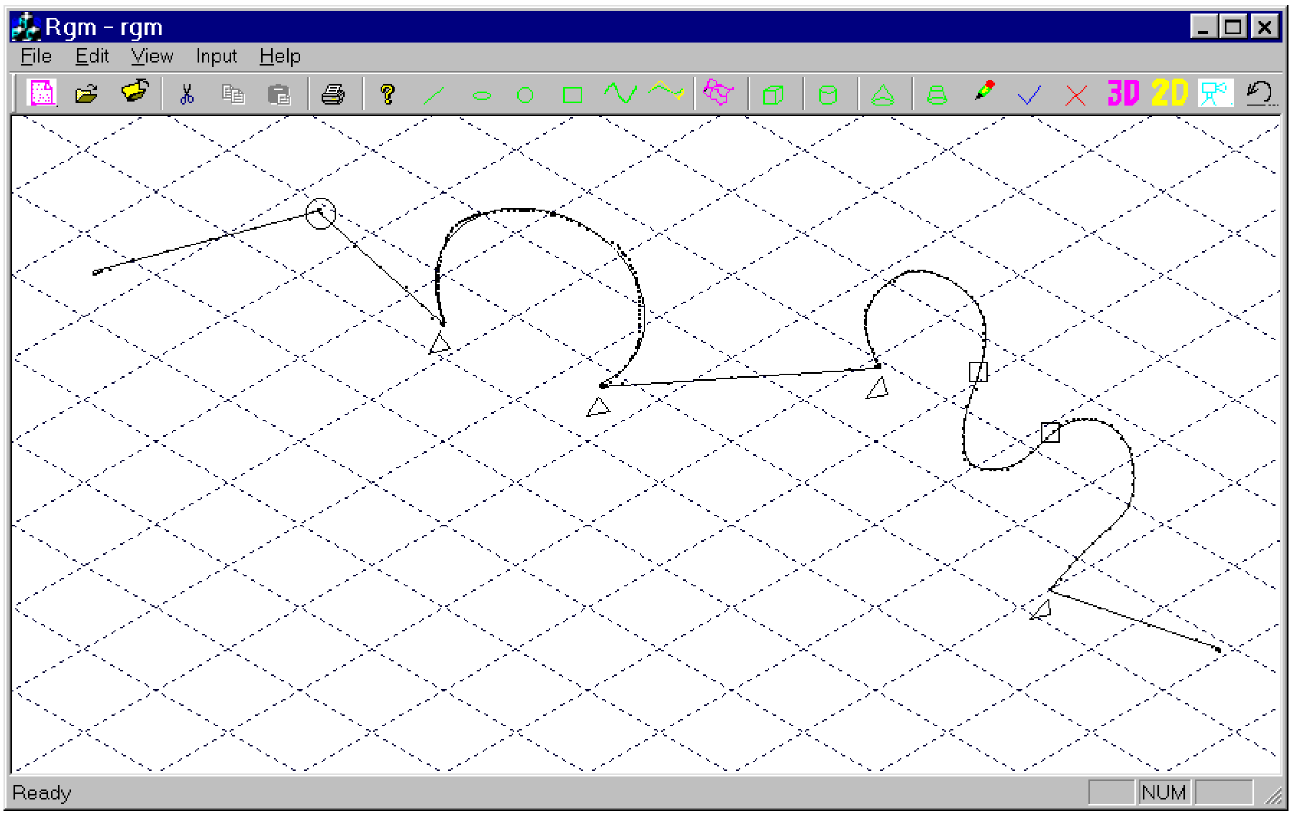

Figure 3. Curve segmentation and identification

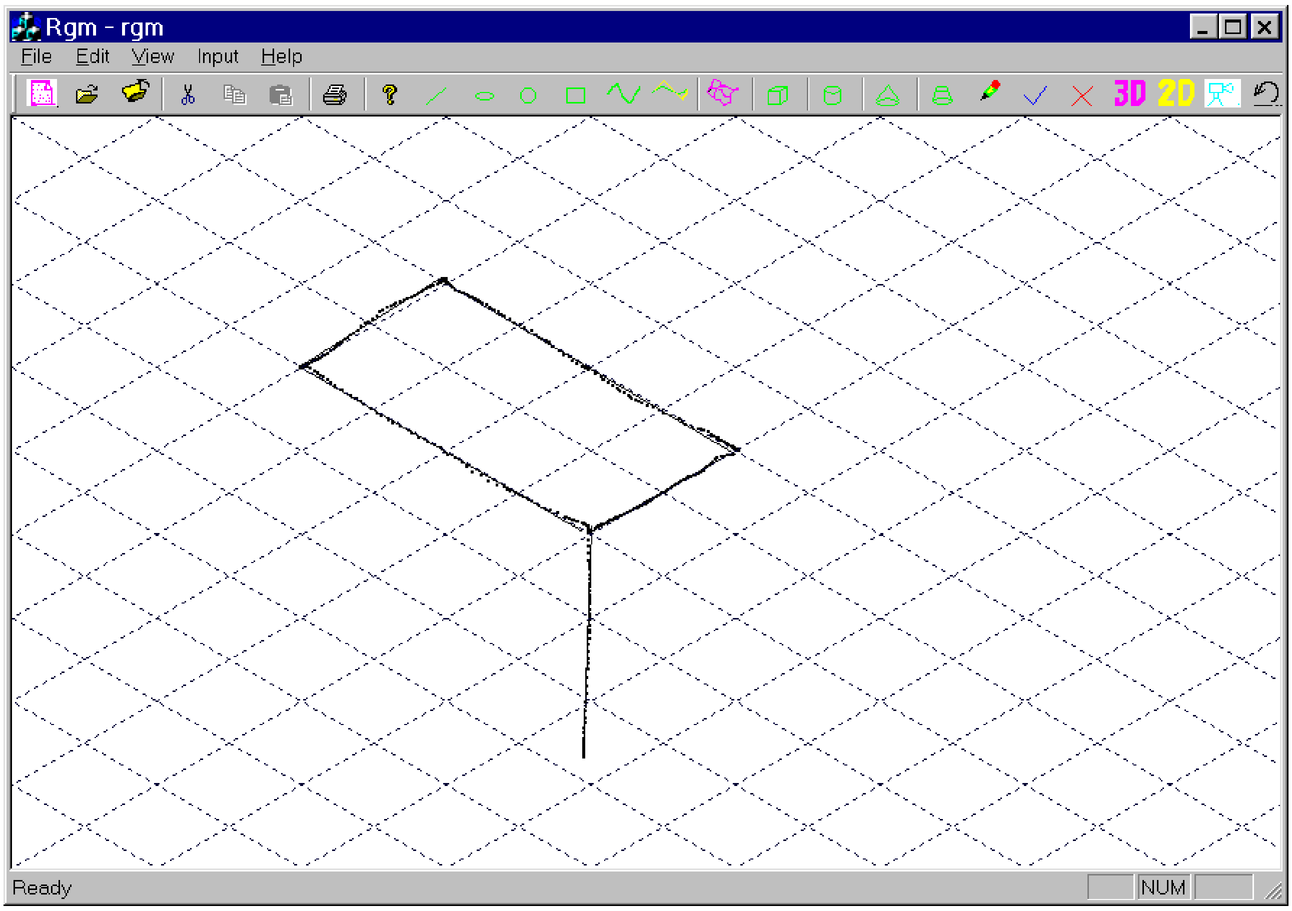

Fig. 4(a)

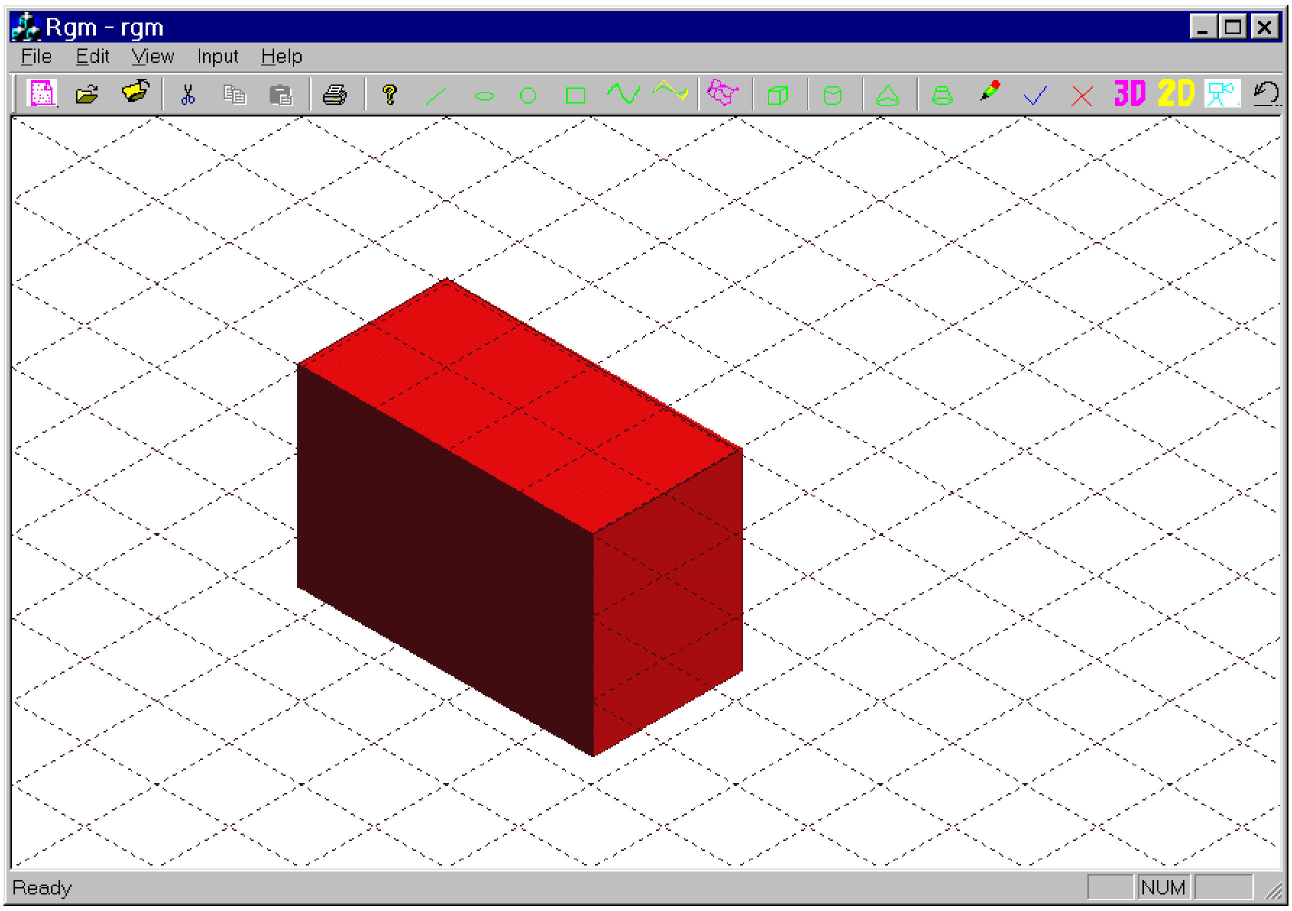

Fig. 4(b)

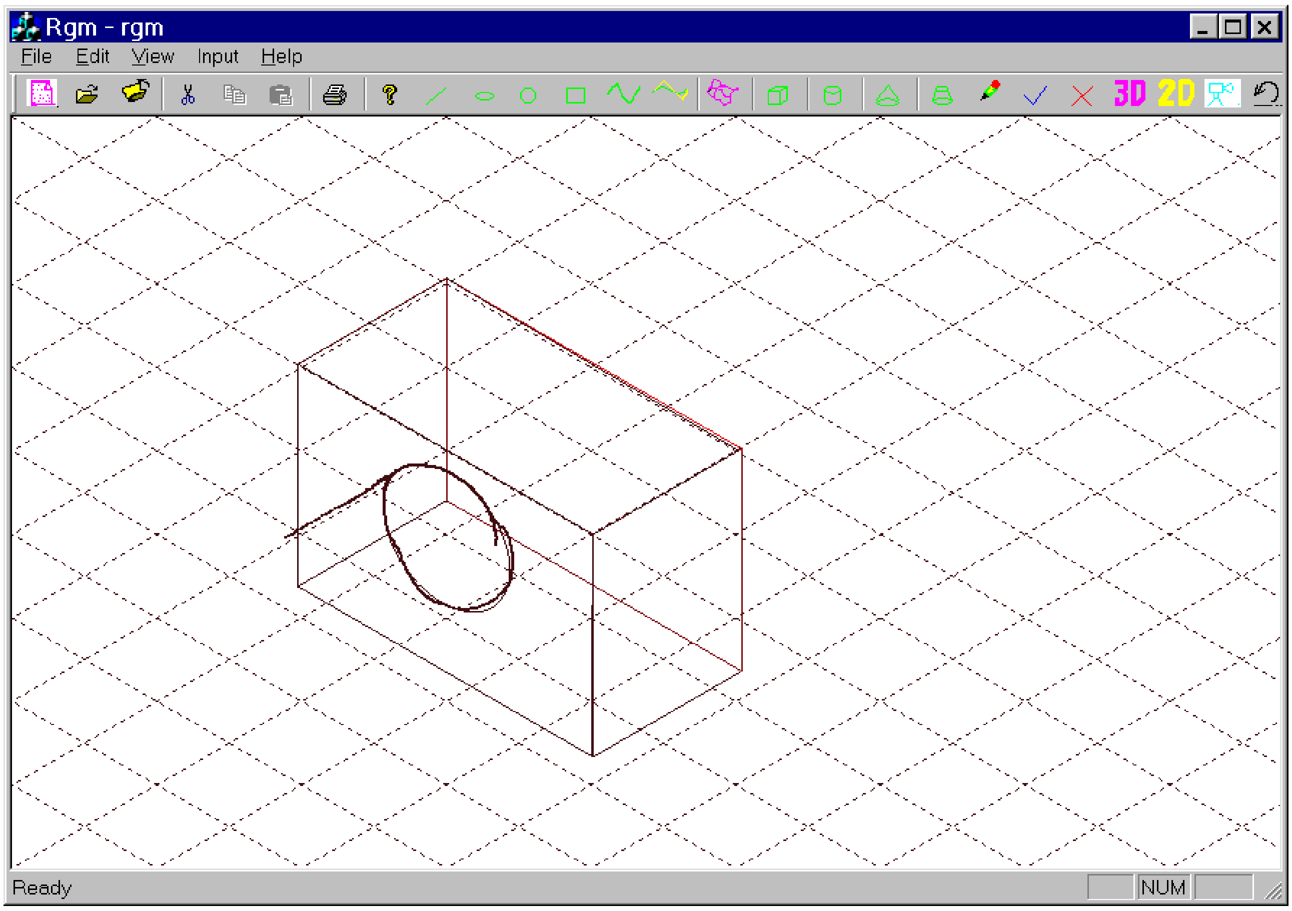

Fig. 4(c)

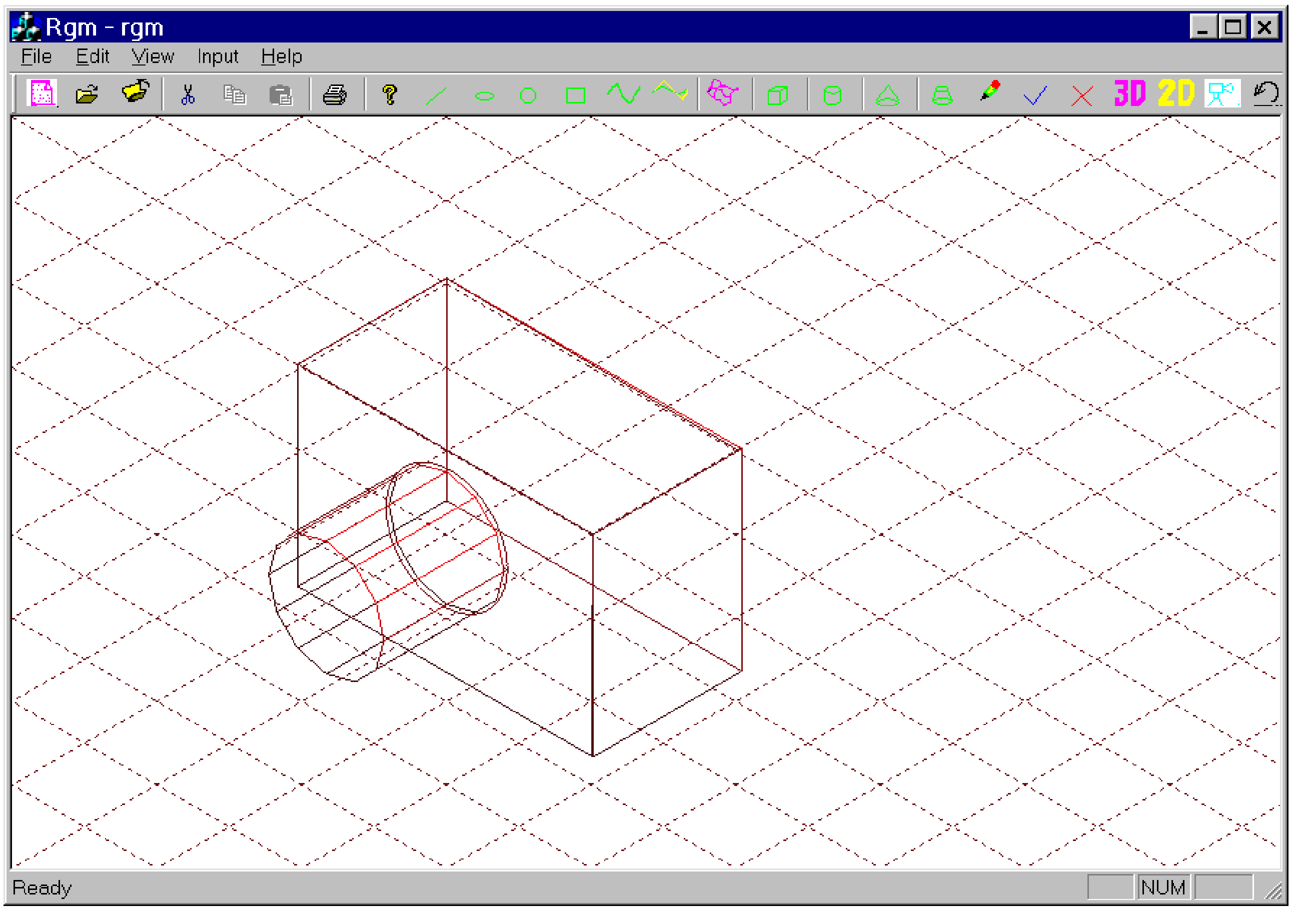

Fig. 4(d)

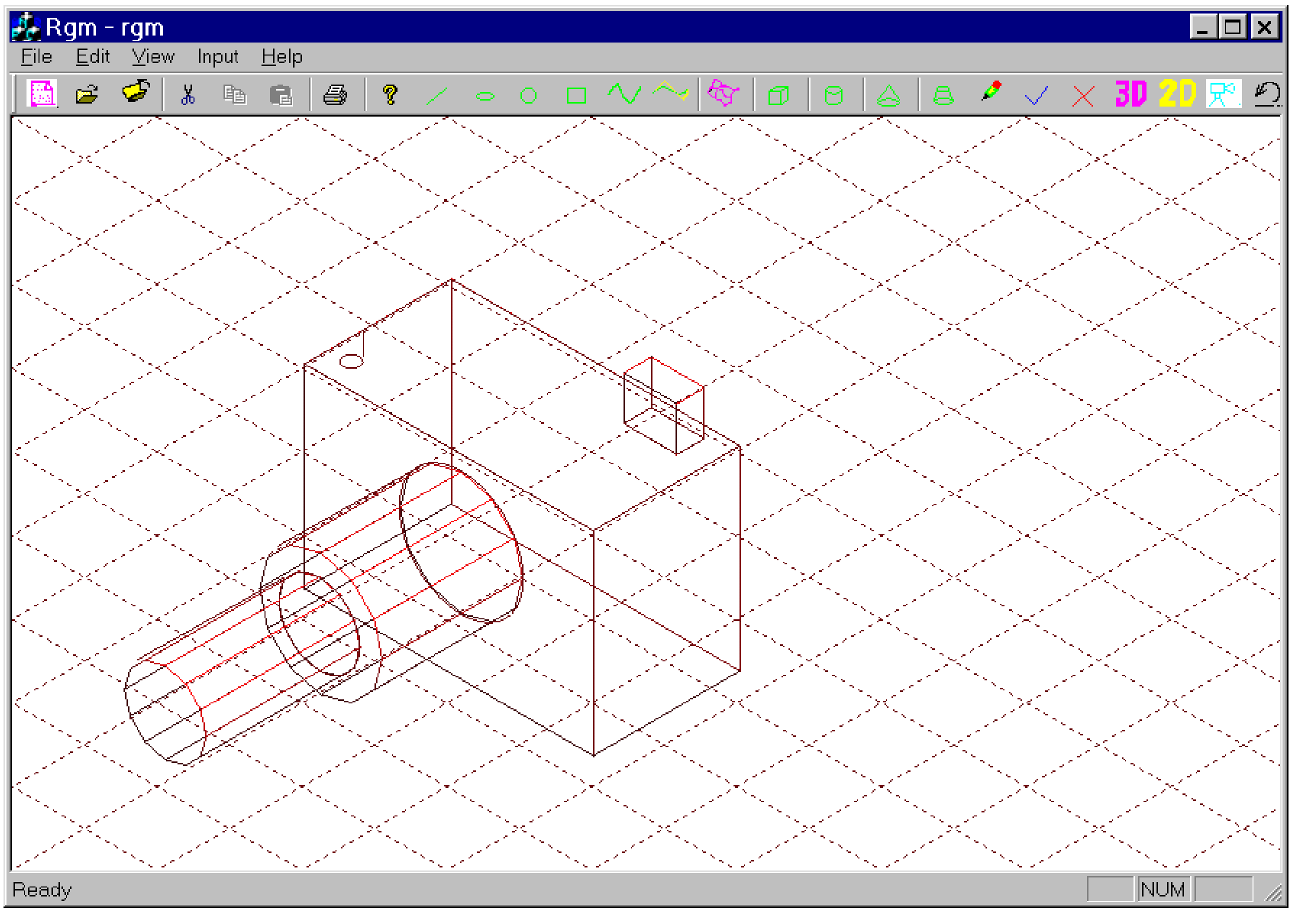

Fig. 4(e)

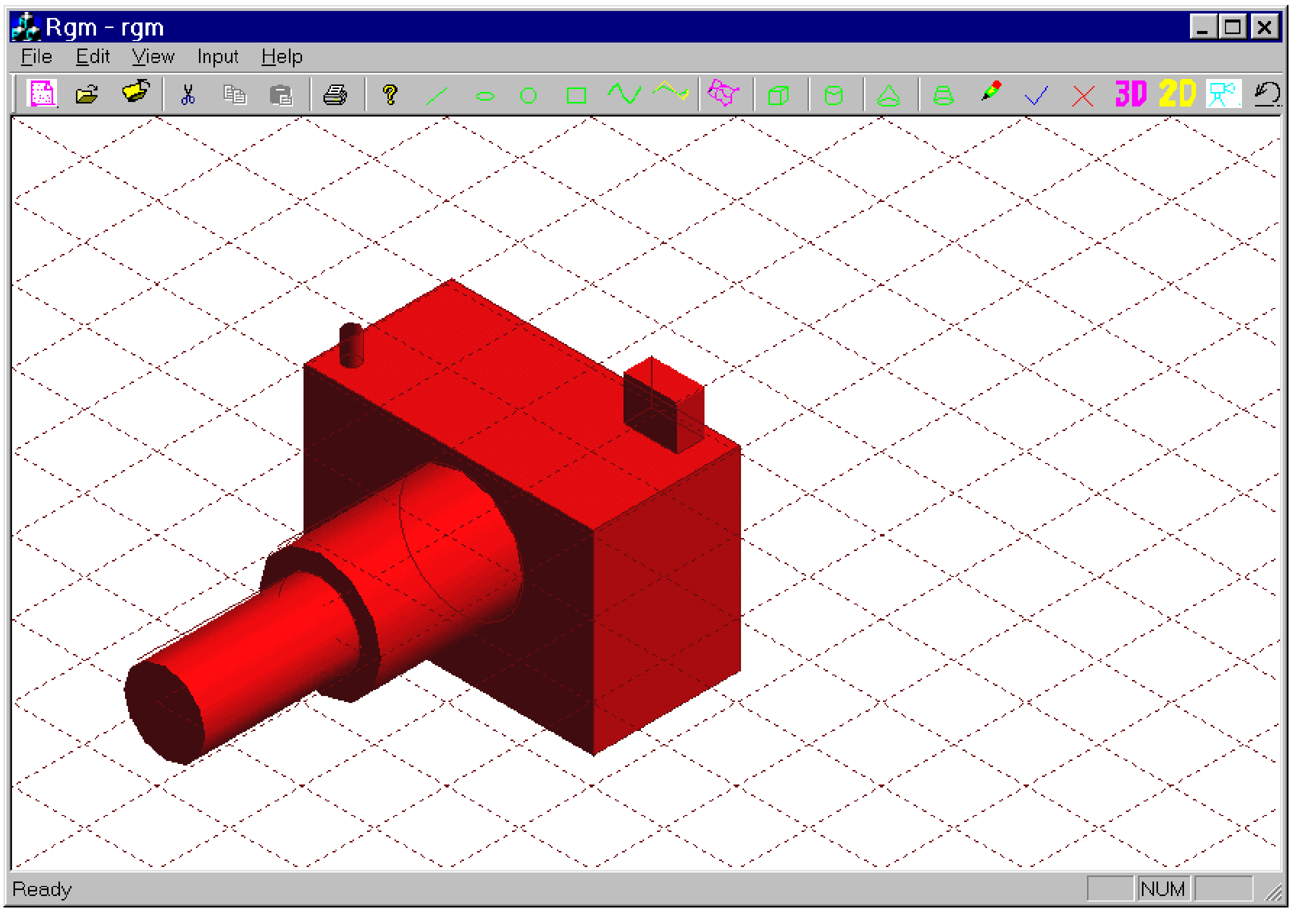

Fig. 4(f)

Figure 4. 3D conceptual models of a camera: (a) sketches for the body box; (b) recognised 3D box; (c) sketching on the body box; (d) cylinder interpretation; (e) interactive input; (f) a shaded model.