J. Computer-Aided Design. Vol.32, No.14, pp. 851-866, 2000.

From on-line sketching to 2D and 3D geometry: A fuzzy knowledge based system

S. F. Qin*, D. K. Wright[a], and I. N. Jordanov[b]
School of Product & Engineering Design, University of Wales Institute, Cardiff CF5 2YB, UK
[a] - Department of Design, Brunel University, Runnymede Campus, Egham, Surrey TW20 0JZ, UK.
[b] - Design Engineering Research Centre, University of Wales Institute, Cardiff CF5 2YB, UK

Abstract

The paper describes the development of a fuzzy knowledge based prototype system for conceptual design. This real time system is designed to infer user's sketching intentions, to segment sketched input and generate corresponding geometric primitives: straight lines, circles, arcs, ellipses, elliptical arcs, and B-spline curves. Topology information (connectivity, unitary constraints and pairwise constraints) is received dynamically from 2D sketched input and primitives. From the 2D topology information, a more accurate 2D geometry can be built up by applying a 2D geometric constraint solver. Subsequently, 3D geometry can be received feature by feature incrementally. Each feature can be recognised by inference knowledge in terms of matching its 2D primitive configurations and connection relationships. The system accepts not only sketched input, working as an automatic design tools, but also accepts user's interactive input of both 2D primitives and special positional 3D primitives. This makes it easy and friendly to use. The system has been tested with a number of sketched inputs of 2D and 3D geometry.

Keywords: Conceptual design; Geometric modelling; Fuzzy knowledge

1. Introduction

Conceptual design is an early stage of the design process, characterised by a fuzzy knowledge of the design requirements and constraints, and tolerating high degrees of uncertainty and vague ideas. A rapid geometric modeller for supporting conceptual design process is highly demanding, because few CAD tools are suitable for this stage of the design process, in which designers use various sketches with vague and imprecise geometry to rapidly express their creative ideas. Besides, conceptual designers still tend to prefer paper and pencil, to a CAD system, for effective expression, communication and record of their ideas. The reason most often given for this is that the interface is not suitable for sketching very basic ideas. To support this early stage of geometric design and to improve the speed, effectiveness and quality of the design decision, studies [1 – 5] indicate that a computer aided conceptual design system must allow sketched input, and must have a variety of interfaces, recognising features and managing constraints.

This paper presents the development of a sketch based CAD system interface for assisting designers during conceptual design stages. The system captures designers' intention and interprets the input sketch into geometrically more exact 2D vision objects and further 3D models. It could also allow designers to specify a 3D object or a scene quickly, naturally, and accurately.

The problem of inputting a 3D object from 2D data, e.g. from several orthographic views, has been addressed by many researchers [6-8]. They aimed to produce a solid model given a complete drawing of the target object which contains depth information, and concentrated on matching vertices between views, or just producing face information. Our aim is to allow designers to input a quick sketch for just a single view. We do not demand several sketch views for expressing their 3D design ideas. Some works [9-11], coming from the computer vision community, aim to reconstruct objects from line drawings, extracted from single perspective view image, rather than sketched by a user. Perspective projections are difficult to sketch, and too error prone to be used for quick sketching by hand. For reconstruction from a single isometric projection, two main methods, namely labelling schemes and optimisation approaches have received considerable attention in computer vision society [12]. Huffman [13] and Clowes [14] set forth the first labelling scheme valid for polyhedra. The Huffman-Clowes labelling method classifies line segments into three categories: convex edges; concave edges; and occluding edges. Given this labelling, there is only a finite number of ways in which lines can meet at junctions. The labelling scheme is based on line labels and junction library to interpret line drawings. Kanade [15], Sugihara [16], and several others [5, 12] have extended the Huffman-Clowes labelling scheme based on junction libraries. Lamb and

* Corresponding author. Tel: +44 (0) 2920 416495, fax: +44 (0) 2920 416946, E-mail: sqin@uwic.ac.uk

Bandopadhay [17] presented a system for interpreting a 3D object from a rough line drawing. Their system uses heuristic rules plus labelling information. In general, labelling methods require a hidden-line removed 2D view of a 3D object, and are not suitable for handling inaccurate drawings and possible missing entities [2]. For an optimisation approach, it requires a complete wireframe drawing as input, and the use of an optimisation method gradually assigning the depth of each vertex from an initially flat drawing to a 3D wire-frame. Then minimising the standard deviation of the angles between connected lines, as in Leclerc's [18], and Marill's [19] works, or identifying and formulating geometrical regularities and seeking their associated 3D configuration, as in the work of Lipson and Shpitalni [2]. These approaches neither take into account drawing errors, nor attempt to tidy up the drawings.

The direct input of depth, whilst creating the sketches, is also investigated. Fukui [20] developed a system for transforming 2D into 3D data, face by face, referring to the geometry of connected faces that are previously transformed. If there is only one, or there is no adjacent face, the viewing direction is referred to. In principle, curved shapes cannot be input directly by this method. Pugh [21] proposed an algorithm that applies geometric constraint satisfaction to the labelling scheme to generate a 3D object description. This description is consistent with both, the designer's line-drawing, and set of geometric constraints, either derived from the line drawing, or placed by the designer. This system produces solid models directly, but not in a natural for the users way. Furthermore, it seems not suitable for interpreting large complicated drawings all at once. Hwang and Ullman [22, 23] developed a design capture system, which has two phases: 2D stroke recognition; and 3D feature recognition. In the first phase, sketched strokes are interpreted as lines, arcs, circles, ellipse, etc. These primitives are accumulated, until they can be recognised as a 3D feature. New features can be built upon previous ones. Their system still uses some junction features, such as *arrow_head* and *duck_claw* to inference a box feature. They did not employ a general modelling feature of extrusion object, in terms of a closed profile with an extrusion edge. Thus, their system has difficulties in applying an inference knowledge method for a box structure to a general combined extrusion object. The weakness of the system is that it recognises a finite number of features: box and cylinder. To construct a complicated design, a large number of features are required. Eggli, Hsu, Bruderlin and Elber [1] described a 'Quick-sketch' system which can interpret 2D sketch into 2D lines, circles, arcs or B-spline curves, and build up geometric relationships. Then it infers 3D models from 2D shapes and constraints. Their system seems questionable in successfully inferring large complicated objects, because it does not interpret 2D ellipses and uses a vague projection co-ordinate system.

This paper presents a profile of the developed system and details of a 2D relationship inference engine, and 3D recognition. After describing the system structure, segmentation and classification of sketch input, identification and generation of 2D primitives and B-spline curves are studied. In section 4 a 2D relationship inference engine is investigated, and then 3D recognition is presented. Discussion with some examples, and conclusion remarks are made in the final section.

2. System description

In this paper, development of a sketch-based CAD system for conceptual design is described. The system flow chart is shown in Figure 1. In the first phase, the system gets a sequence of input data from mouse button presses, mouse motion and mouse button release events. From this data, information about the speed, acceleration, direction, angle, and accumulative chord length is extracted. This information is used in the following processing to infer users drawing intentions, and then to filter unintentional and redundant points. During the segmentation phase, the input sketch is divided into several sub-curves by locating segmentation points (points connecting two meaningful sub-curves). In the third stage, each of the curve segments is classified and recognised. Then, the corresponding precise 2D primitives or B-spline curves are identified and generated. At this stage, 2D primitives can also be quickly inputted by selecting from a 2D menu. After that, 2D relationships (connectivity, parallelism or perpendicularity) between the primitives is conducted. Subsequently, a 2D geometry can be received by a 2D geometric constraint solver, based on the relationship information. Finally, this 2D geometry (primitives and connections) is accumulated until it can be recognised as a 3D object or feature. The features are placed in a 3D space and new features can be built upon previous ones.

3. Segmentation and classification of sketched input

The systems for on-line sketching and interpretation, referenced in previous section, have no segmentation process because each stroke is assumed to correspond to a single entity. However, this assumption simplifies and limits a variety of applications. For example, several connected lines can be drawn with one stroke on pen-paper based sketches.

To allow sketched input in more natural way, in our system, one stroke input can include more than one geometric primitive. Thus, precise segmentation of the sketch strokes into straight lines and other sub-curves is prerequisite for obtaining the best sketch recognition and interpretation. Errors in the segmentation phase might propagate to false feature extraction and classification.

3.1 Curve segmentation

In our system, an intelligent and adaptive threshold segmentation technique is used on the basis of fully exploiting properties of dominant points, and fuzzy heuristic knowledge in terms of sketching speed and acceleration. We combine fuzzy logic ideas from [24] with a hybrid approach [25], that emphasises on accuracy and speed to segment curves by finding acute and obtuse corner points, and inflection points. For an obtuse corner point, directional deviation ranges from 90° to 180°, and for an acute corner point the deviation is less than 90°. As for inflection point, an identifying feature is a change in the curve convexity. Details of this segmentation process are given in [26].

Our fuzzy knowledge based segmentation algorithm can be briefly described in four steps:

Step 1 Compute the directional deviation β_i at point i with an adaptive support region, based on k-cosine curvature measure, and perform nonmaxima suppression;
Step 2 Find obtuse corner point, if β_i is larger than 90 degree;
Step 3 Detect acute corner points between two adjacent obtuse corner points, by applying adaptive threshold and fuzzy knowledge, with respect to the drawing speed, acceleration, and curve's linearity;
Step 4 Specify inflection points, if it is necessary, during the following classification process.

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. We classify a curve according to three preference orders: linearity; convexity; and complexity of a shape, not just by complexity as in [24, 27]. In comparison with the referenced works, this classification method brings advantages of reducing computational burden and complexity. Details of this classification are presented in [28]. A curve classification briefly follows a four step procedure:

Step 1 Detect a straight line by its linearity;
Step 2 Detect a free-form curve by finding inflection points or convexity changes;
Step 3 Determine a spiral line (free-form curve) by checking self-intersection point;
Step 4 Classify into a circle, an arc, an ellipse, an elliptical arc, a hyperbola, or a parabola by least-square fitting general conic equation, or further identification for free form curve.

3.2 Identification and generation

After the classification, each curve should be identified and fitted with a meaningful 2D primitive or a B-spline segment, representing the corresponding sketching points. To find the best coefficients (a, b, c) of a line segment equation: $aX + bY + c = 0$, a weighted least-square (LS) routine is used. For a conic curve, we investigate weighted LS fitting with some normalisation techniques, based on algebraic distances, because geometric distances are difficult to evaluate. After the LS fitting, a conic curve can be further classified into a circle, an arc, an ellipse, or an elliptical arc. Then, the corresponding 2D parameters are received. For a line, we obtain the line equation and two end points. For an elliptical arc, we get the centre point, two radii, direction, and start and end angles [29]. A circle and an arc are special cases for elliptical arcs. When free-form curves are classified, B-splines are used to fit a set of sketched points. Once a 2D primitive or a B-spline segment is specified, it is displayed on the screen.

3.3 System behaviour and 2D interaction

Nevertheless, there will always be cases where the system makes a wrong interpretation of the user's intention. It is therefore essential to provide designers with a simple way of correcting erroneous interpretations. In our system, the user can quickly click on *x*-icon and then select an intended shape icon from the toolbar icon menu. Then the system refits sketched points with the specified shape.

On the other hand, users can utilise the icon menu to input 2D primitives quickly and more accurately, as in any 2D CAD system. With these tools, users can mix freehand sketching and interactive 2D input to quickly specify 2D primitives. Some features, such as fillet elliptical arcs might be difficult to sketch. The reason for that could be individual's low-level sketching skill or vibration from sketching devices. Anyway, the system can enable users to mix the two input methods to any content they want. If the system could only accept sketched input, users with poor sketching skills are likely to be dissatisfied.

All 2D primitives are stored in a drawing history database in time order. The following figures show some examples (sketches to the left, and fitted curves to the right). Figure 2 shows examples of finding obtuse segmentation points between different primitives: lines, circular arcs, elliptical arcs and free-form curves. For detecting acute corner points, examples are shown in Figure 3. Figure 4 is employed to demonstrate a B-spline fitting and inflection point finding. Examples of identification of circles and arcs from general ellipse fitting are shown in Figure 5. The system is checking the sketch for closed arcs (circular or elliptical) before the generation. If an closed arc is over-drawn, the system will make the two end points to meet together, as shown in the middle graphics of Figure 5. If the angle formed from the centre point of the arc to its two end points is too small, for example 5°, the system will change the arc to a whole circle or ellipse, as in the case shown just

above the bottom one in Figure 5. The example at the bottom is a normal arc identification.

4. Relationship inference engine

Once the closest fitting primitives have been found, the system's relationship inference engine tries to infer certain relationships between them. Relationships can be classified into three categories: connectivity; unitary; and pairwise relations [27].

The inference engine firstly searches for connectivity relations. It looks at the end points of a pair of primitives, and determines whether they are within a certain adaptive distance tolerance. If they are, the two end points of the two primitives are connected. In this case, relation code 1 is assigned (default is 0, meaning free end). The adaptive distance tolerance is related to the lengths of lines or radii of arcs. Then, the inference engine tries to infer second or third type relations for a free end. Second type connection is touching relation, in which an end point of a primitive falls on the path of the other primitive. The relation code in this case is 2. The third type connection (relation code 3) is tangent relation, in which one end of a primitive is tangent to another primitive, as between lines, circles, ellipses, arcs, or elliptical arcs. The fourth type connection is an ellipse tangent to, or on the path of the other primitive. Its relation code is 4. For example, when sketching slot features from a box or a cylindrical object, users will meet the second type connection, and when silhouette lines are drawn to express a cylindrical object or feature, the third type connection will be obtained. The type 4 connection is met, when an ellipse from a projection of a section circle touches the path of a silhouette curve to express a revolution feature.

Unitary relations are properties of a single primitive. The unitary relations apply to lines, ellipses, arcs, and elliptical arcs. For the lines, the engine examines the slope of the straight line, to see if it is close to one of special directions: horizontal; vertical; and isometric projection of principle axes. If it is close, to the straight line will be assigned corresponding unitary relation code: HOR, VER (or ISO-Y), ISO-X, or ISO-Z. Subsequently, this line will be changed to its corresponding direction. For an ellipse, we check the direction of its axis to determine if it is close to one of the special directions. If so, the ellipse gets the same unitary relation codes as in the line cases. In the case of circular arcs, the angles formed from the centre of the arc to its two end points are examined to check if they match any of the special direction angles. If so, the angle subtended by the arc will be changed. For an elliptical arc, we first check its direction, as for an ellipse, and then examine its start and end angles, as for a circular arc.

Pairwise relations are geometric properties shared by two primitives. Currently, the system supports parallelism and perpendicularity relations between pairs of lines, ellipses, or elliptical arcs. Each line, or ellipse, may have only one pairwise relation with previous primitive: parallelism or perpendicularity. Once this relation is found, the system will stop further backward search for that type relation.

After all these relations are examined, the inference engine will clean up the drawing by using the relations as geometric constraints. First, the engine corrects the primitives in accordance with unitary relations and with the least amount of local changes. Then, the engine gives correction for the primitives, according to the pairwise relations and connectivity constraints. Finally, the 2D geometry with its relationships is configured, using the inference engine.

Figure 6 shows sketches of a house with a square door, an elliptical window, etc. Before tidying up, the primitives are not clearly connected. For example: the elliptical window is not quite vertical (or horizontal); the left wing line of the roof is not parallel to the grid lines; the sun ray lines do not start exactly from the circle to outwards, etc. However, after tidying up, the 2D geometry from Figure 6 becomes more exact, which is shown in Figure 7.

5. Recognition in 3D

After the 2D correction, the 2D geometry has its correct primitives and topology connections. The problem remaining is how to recognise 3D objects from a 2D topology and geometrical information. From previous research [23], it is believed that design with features will bring some significant benefits for the design process itself and further for the manufacturing process as well. Therefore, the system combines solid modelling methods with feature based design methods to develop a 3D inference engine for machined parts. This method brings the following advantages:

- It processes input model data rapidly and efficiently. Once a specific feature is recognised, it is not necessary to continue with inputting the complete model data. For example, if a closed 2D profile with one extrusion edge is recognised as a extrusion object, further input of any edges will be unnecessary. In this case, the user can input any extrusion edge, no matter whether it is visible edge or not. So, the user is not interactively interrupted in order to determine which edges should be inputted;
- Once the design is finished, the feature model is available. This makes unnecessary the creation of a feature model by decomposing the solid model for further manufacture processing;
- It simulates parametric modelling methods in the design process. The design is conducted feature by feature in temporal order. The constraints and geometric variables are easy to set up and if the design system is used as an interface to a

commercial parametric CAD system, such as, Pro/Engineer, we can effectively integrate the conceptual design process with more detailed design.

Although the number of features required for modelling complex mechanical parts is huge, it can be reduced significantly if the main focus is on the conceptual design, because at this stage rough design ideas are expressed mainly by solid primitives and their Boolean operations: union; subtraction; and intersection.

In order to apply features to the design process, the system first recognises feature information from the 2D freehand sketches. It then transforms the recognised feature to its proper 3D position. Once a feature is created, the user can examine all features in a wireframe model or in a shaded solid model. The user can continue to add features based on wireframe or shaded model. Users may, therefore, begin to sketch in a real 3D world. In this way, the system can support an iterative creative design process: thinking; creating; and evaluating.

5.1 Features and system setting

In order to recognise 3D features, the system makes following assumptions:

- 2D input is a isometric drawing. The origin of the isometric projection co-ordinate system is the same as the origin of the display window (lower-left corner). The system selects isometric drawing for two reasons: first one, that parallel lines in the objects appear as parallel lines in the drawing; and second one, that edges parallel to the principle axes are drawn with lengths proportional to the actual dimensions of the objects (about 0.8165 of the actual dimensions);
- The projection co-ordinate system has the same scale as the display system (default value is 1);
- Dimension unit is a screen pixel.

In general, the system can recognise the following features:

- Box feature, which can be transformed into protrusion of a rectangular shift (Fig. 8(a)), or depression of a rectangular hole, or slot (Fig. 8(b));
- Cylindrical feature, which can be recognised as a cylindrical shift (Fig. 8(c)), or a hole including blind and through holes (Fig. 8(d));
- Revolution feature, which can be any revolution, solid or hollow (Fig. 8(e));
- Spherical feature, which express a solid ball;
- Ruled surface feature (Fig. 8(f));
- Sweeping surface (Fig. 8(g));
- Modified feature, which can be a chamfer or a fillet (Fig. 8(h));
- Complex extrusion feature.
- Due to limited developing time, this prototype system has implemented box features, cylindrical features and simple revolution features.

5.2 Knowledge representation

Our 3D recognition engine expresses its recognition knowledge in knowledge rules and integrates them into a programme by conditional statements (*if-then*). 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. General extrusion objects feature a closed profile and extrusion edges. The closed profile may consist of only one ellipse, or two pair parallel lines, or combined line segments with arcs. Therefore, the system first examines whether a closed profile exists, then finds the direction of extrusion (for box features, this direction information can come from the direction of the extrusion edge, as for cylindrical features, this direction information can be obtained from the direction of the ellipse). Finally, the system determines where the closed profile comes from (reference plane) by checking if its centroid is within a projection area of a boundary plane of previous objects. Once a specific feature type is found and the reference plane and direction information are known, the system obtains all the necessary 3D information and can produce the 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, if the closed profile (representing a face) is an ellipse, which has either second, or third type connection with a straight line, the feature might be a cylindrical shift (Fig. 8(c)), or a hole (Fig. 8(d)). Alternatively, this could be a revolution feature if the ellipse touches the edge line with a type four connection (Fig. 8(e)). In this case the ellipse represents a section circle, and the extrusion edge represents one of the silhouette lines. For a type 4 connection, the corresponding feature might be a general revolution object if the edge is a curved line. Some inference rules 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 lines)
- *and* the extrusion direction is determined (by the extrusion line)
- *and* the reference plane is found (default reference is XOZ, XOY, or YOZ plane correspondingly to the special extrusion directions)

then
- a box feature is defined.

Rules for a sweeping surface could be:

if
- the feature is composed of two curves
- *and* the two curves are end-connected
- *and* the two reference planes for two curves belong to one box feature.

then
- a sweeping feature is found.

Once a feature is created, it is stored in an Object-Oriented database. Its class construction function will record features size and position parameters (3D shifting and rotating parameters), and will produce information about all boundary faces, both in 3D and in the projection plane. All this information is used for determining the reference planes.

5.3 Interaction in 3D

Similarly to the 2D input, icon menus can be used to input 3D primitives and mix freehand sketching with interactive 3D input, to quickly specify 3D primitives. For example, the users can quickly specify a 3D box by drawing a diagonal line for a top face and subsequently dragging the face vertically, to produce height information. From the menus, the users can input vertical cylinders, semi-cones, or cones.

On the other hand, the system provides assistant grid lines in accordance with the isometric projection. These grid lines can help users to sketch in 3D.

Theoretically speaking, users can correct the 3D feature recognised by the engine, if it is wrong. However, correcting a 3D feature is more difficult than correcting a 2D fitting, therefore, this function needs further refinement.

6. Discussion and conclusion

With series of figures (Fig. 9(a) to Fig. 9(f)) we demonstrate the process of producing an object that combines a cylinder with a box. First, we sketch two ellipses and two lines for a cylinder (Fig. 9(a)). Fig. 9(b) shows the recognised 2D primitives. After 3D recognition, we receive a 3D cylinder, and continue to sketch a box over the wireframe model of the cylinder (Fig. 9(c)), obtaining five 2D lines for the box (Fig.9(d)). Again, 3D recognition is performed, and finally, the combined object is shown in wireframe (Fig. 9(e)) and in shaded model (Fig. 9(f)).

We use this prototype system to deal with a conceptual design geometric model of a lathe machine. This geometric model consists of 10 parts: two base parts; a headstock; a spindle; a gear box; a lead screw; a feed rod; a carriage; a tailstock; and a cross slide. The spindle is expressed as a cylindrical feature. The feed rod is modelled as a revolution object. Most of the parts are presented as box features. Fig. 10 shows a wireframe model of the lathe (with some 2D input geometry). The shaded model is shown in Fig. 11. When working on this model, we draw some features on the previous wireframe model, and some features on the shaded model, because it is easier to draw on the 3D faces.

Fig. 12 describes a model of a scene including a desk and a small bench. Fig. 13 shows its shaded model. The left desk foot is a semi-cone, formed from revolution feature. On the desk, there is a vertical shelf (a thin box), which holds a horizontal lump tube (a cylinder). A small clock (cone feature) is situated on the shelf. There is also a small bench in front of the desk, modelled by three box features.

This system is implemented on Windows'95, using Visual C++ and Open GL. A part of the system has also been developed on SGI workstation and UNIX platform, in C++, Motif and Open GL. The results show that the fuzzy knowledge based system can interpret users' intention on 2D and 3D geometry satisfactorily. From real-time sketches, the system can give proper segmentation and curve fitting in variety of 2D shapes: straight lines, circles, arcs, ellipses, elliptical arcs, spiral lines, spring lines and free-form curves. The 2D relationship engine can generate 2D connectivity, unitary constraint, and pairwise constraint information. This information is used for tidying up the 2D geometry and for inferring a 3D object. After the 2D cleaning up, a rule-based 3D recognition is conducted. The system combines interactive input of 2D and 3D primitives, with sketched input recognition. This system gives users greater freedom to quickly specify 2D and 3D geometry, than those with sketched input only [23]. 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 a potential capability of supporting 3D surface design. It can model scenes, which are difficult for the labelling schemes and optimisation-based methods, although it requires corresponding recognition knowledge for the different features.

References

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

[2] Lipson H, Shpitalni M. Optimisation-based reconstruction of a 3D object from a single freehand line drawing, Computer-Aided Design 1996;28(8):651-663.

[3] Lipson H, Shpitalni M. A new interface for conceptual design based on object reconstruction from a single freehand sketch. Annals of the CIRP 1995;44(1):133-136.

[4] Dorsey J, McMillan L. Computer graphics and architecture: State of the art and outlook for the

future. ACM SIGGRAPH- Computer Graphics 1998;32(1):45-48.
[5] Grimstead IJ, Martin RR. Creating solid models from single 2D sketches. In: Proceedings of the Third Symposium on Solid Modeling Applications, ACM SIGGRAPH 1995:323-337.
[6] Yan QW, Chen CP, Tang Z. Efficient algorithm for the reconstruction of 3D objects from orthographic projections. Computer-Aided Design 1994;26(9):699-717.
[7] Masuda H, Numao M. A cell-based approach for generating solid objects from orthographic projections. Computer-Aided Design 1997;29(3):177-187.
[8] Dori D. A scheme for 3D object reconstruction from dimensioned orthographic view. Engineering Application of Artificial Intelligence 1996;9(1):53-64.
[9] Horaud R. New methods for matching 3-D objects with single perspective views, IEEE Pattern Analysis and Machine Intelligence 1987;9(3):401-411.
[10] Hale BJ, Burton RP, Olsen DR, Stout WD. A three-dimensional sketching environment using two-dimensional perspective input. Journal of Imaging Science and Technology 1992;36(2):188-196.
[11] Ulupinar F, Nevatia R. Constraints for interpretation of line drawings under perspective projection. Computer Vision, Graphics and Image Processing: Image Understanding 1991;53(1):88-96.
[12] Wang W, Grinstein G. A survey of 3D solid reconstruction from 2D projection line drawing. Computer Graphics Forum 1993;12(2):137-158.
[13] Huffman DA. Impossible objects as nonsense sentences. In: Meltzer, Michie D, editors. Machine Intelligence, Edinburgh: University Press, 1971;6:295-323.
[14] Clowes MB. On seeing things. Artificial Intelligence 1971;2(1):79-112.
[15] Kanade T. Recovery of the three dimensional shape of an object from a single view. Artificial Intelligence 1980;17:409-460.
[16] Sugihara K. Machine interpretation of line drawings. Cambridge, Massachusetts: MIT Press, 1986.
[17] Lamb D, Bandopadhay A. Interpreting a 3D object from a rough 2D line drawing. In: Proceedings of Visualization'90, pp. 59-66, 1990.
[18] Leclerc YG, Fischler MA. An optimization-based approach to the interpretation of single line drawings as 3D wire frames. International Journal of Computer Vision 1992;9(2):113-136.
[19] Marill T. Emulating the human interpretation of line-drawings as three-dimensional objects, Int. Journal of Computer Vision 1991;6(2):147-161.
[20] Fukui Y. Input method of boundary solid by sketching. Computer-Aided Design 1988;20(8):434-440.
[21] Pugh D. Designing solid objects using interactive sketch interpretation, ACM SIGGRAPH-Computer Graphics Special Issue on 1992 Symposium on Interactive 3D Graphics, 1992:117-126.
[22] Hwang T, UIIman D. The design capture system: Capturing back-of-the envelope sketches. Journal for Engineering Design 1990;1(4):339-353.
[23] Hwang T, UIIman D. Recognize features from freehand sketches. ASME Computers in Engineering 1994;1:67-78.
[24] Chen P, Xie S. Freehand drawing system using a fuzzy logic concept, Computer-Aided Design 1996;28(2):77-89.
[25] Wan W, Venture J. Segmentation of planar curves into straight-line segments and elliptical arcs. Graphics Model and Image Process 1997;59(6):484-494.
[26] Qin SF, Wright DK, Jordanov IN. Intelligent recognition of 2D design sketches. In: Hamza M, editor. Proceedings of the International Conference on Intelligent Systems and Control, Santa Barbara, California, USA, October 28-30 1999:246-251.
[27] Jenkins DL, Martin RR. Applying constraints to enforce users' intentions in free-hand 2-D sketches. Intelligent Systems Engineering 1992;1(1):31-49.
[28] Qin SF, Wright DK, Jordanov IN. Intelligent classification of sketch strokes. In: Hu H, Wu M, editors. Proceedings of CACUK, Derby, UK, September 24-25 1999:189-194.
[29] Qin SF, Jordanov IN, Wright DK. Freehand drawing system using a fuzzy logic concept, Computer-Aided Design 1999:31(5):359-360.

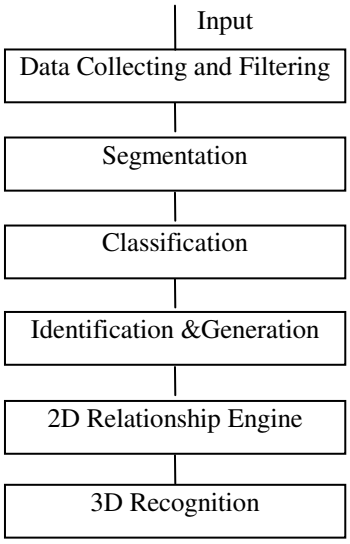


Figure 1 System flowchart

Figure 2. Finding obtuse corner

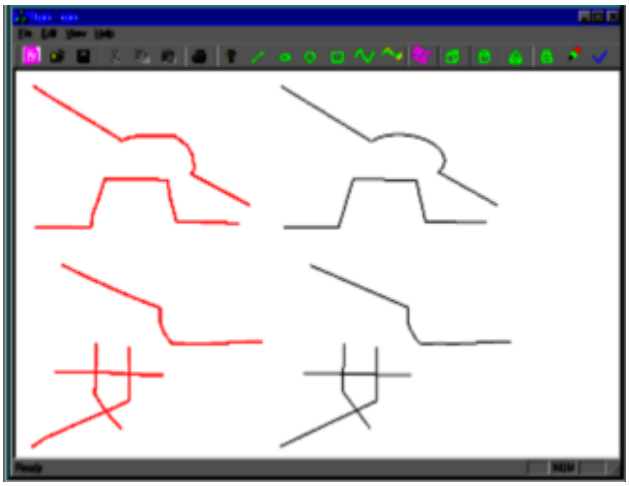


Figure 3. Finding acute corner

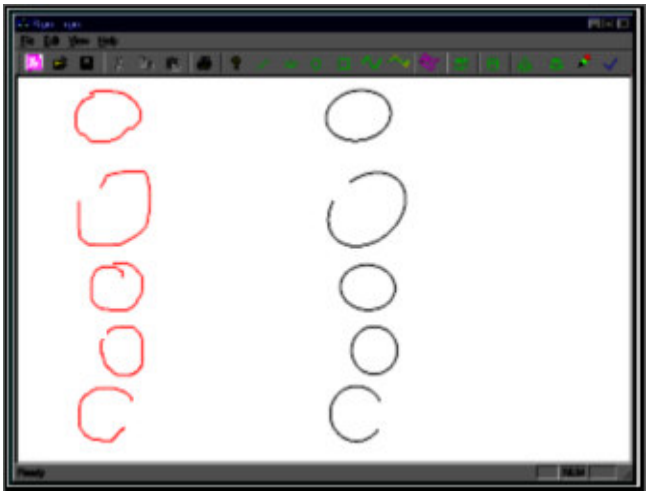


Figure 5 Circle and arc fitting

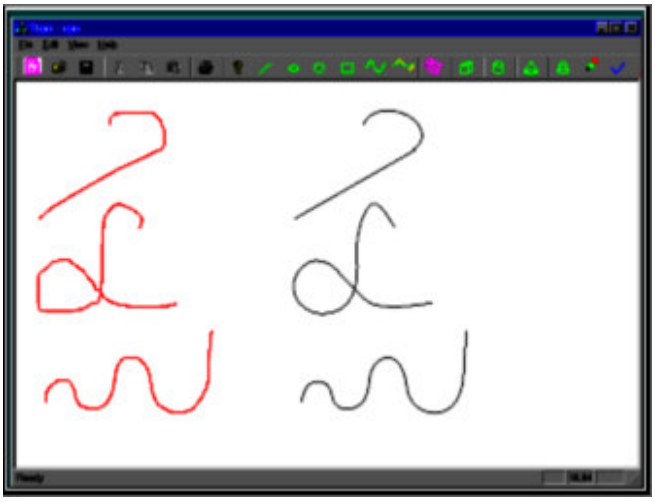


Figure 4. Curve fitting and segmenting

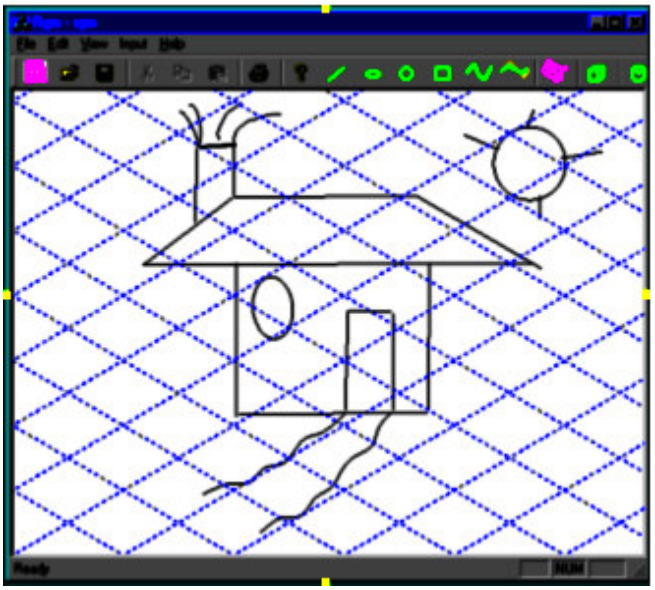


Figure 6. 2D primitives before tidying up

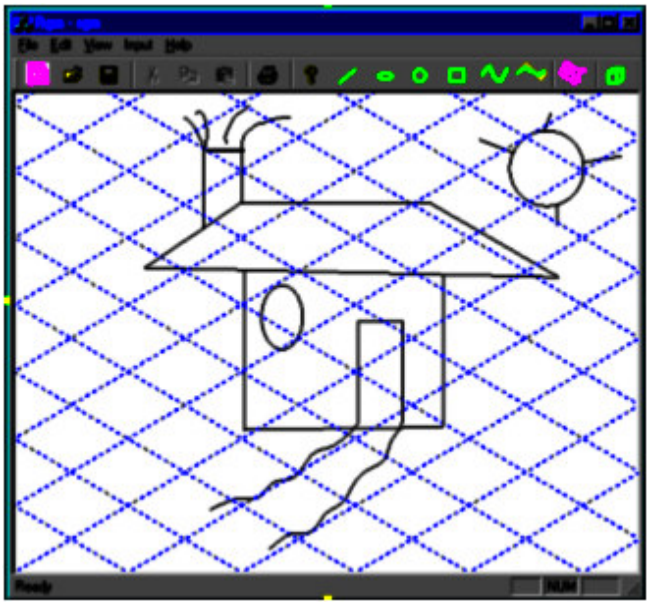


Figure 7 2D Geometry after tidying up.

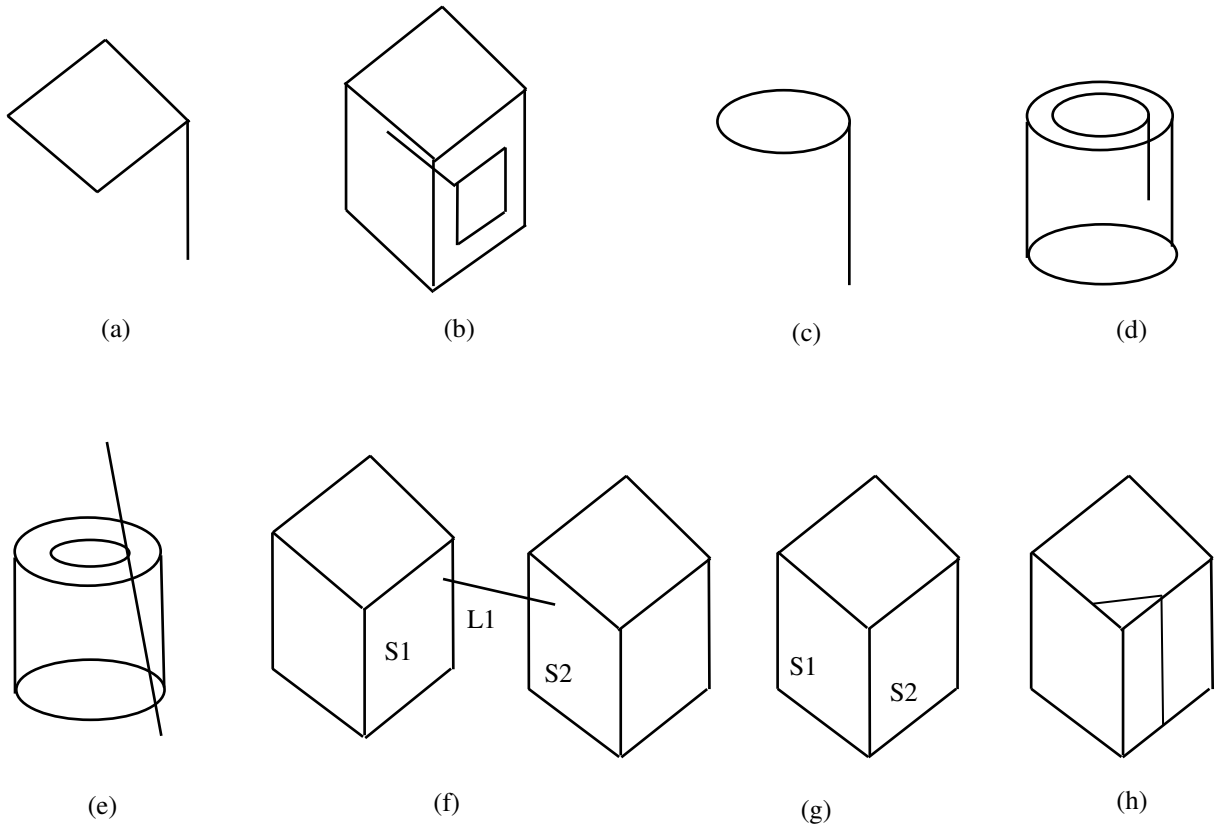


Figure 8. Figure 8. Features: (a) box; (b) rectangular hole; (c) cylindrical shift; (d) cylindrical hole; (e) revolution; (f) ruled surface; (g) sweeping surface; (h) chamfer

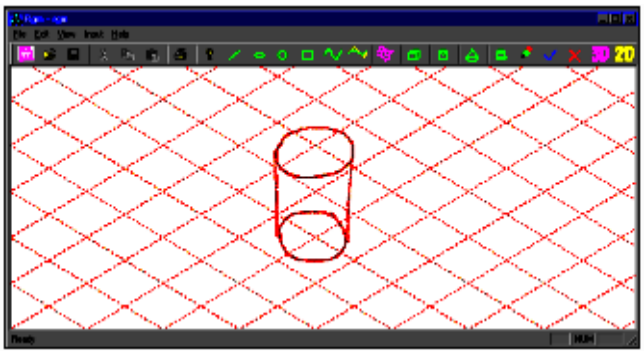


Figure 9. (a)

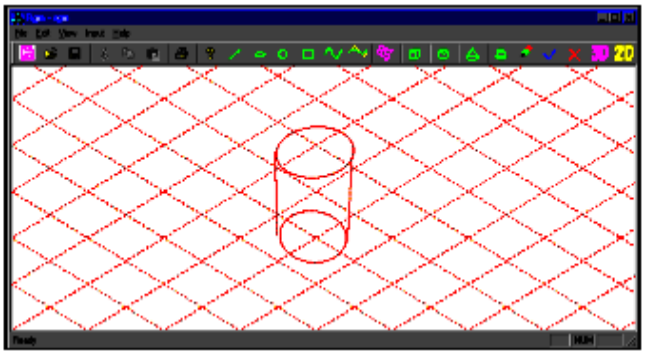


Figure 9 (b).

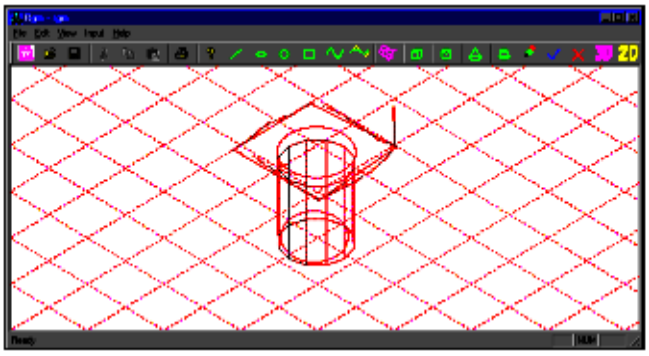


Figure 9 (c)

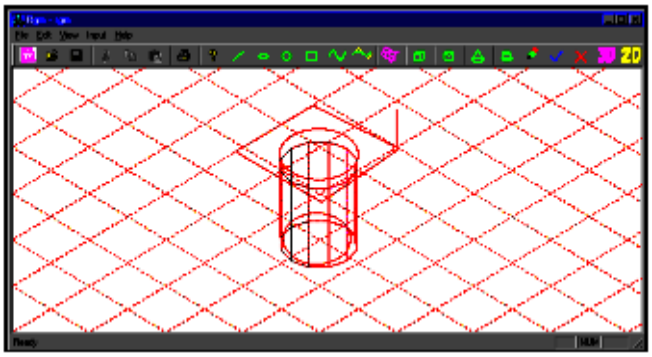


Figure 9 (d)

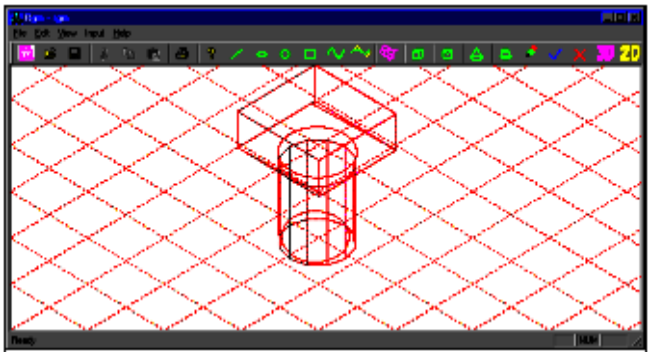


Figure 9 (e).

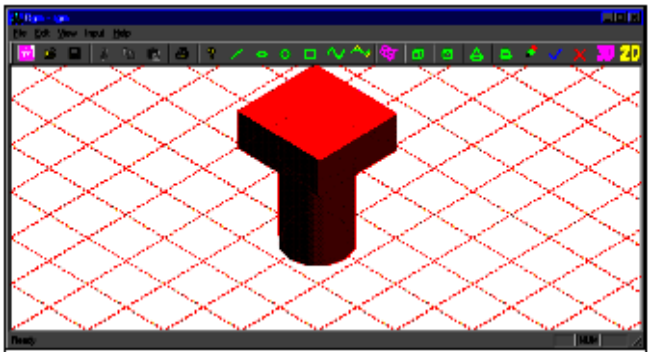


Figure 9 (f)

Figure 9. Processes of building objects: (a) sketch of a cylinder; (b) 2D primitives; (c) sketching on the previous wireframe model; (d) 2D primitives for the box; (e) wireframe model of the combined objects; (f) shaded model for the combined objects.

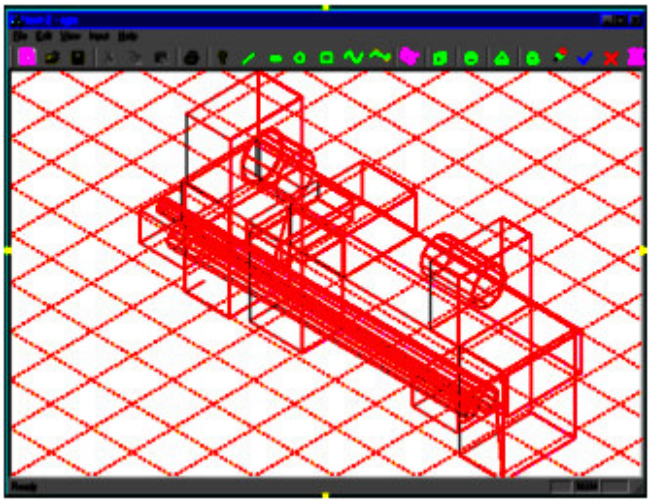


Figure 10 A wireframe model of a lathe

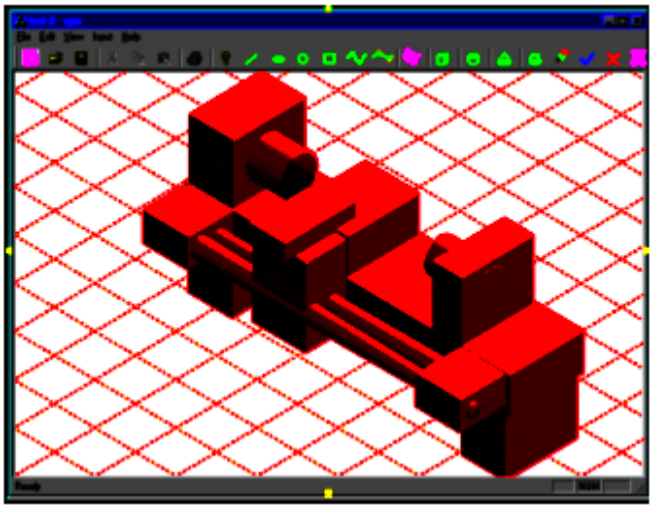


Figure 11 A shaded model of a lathe

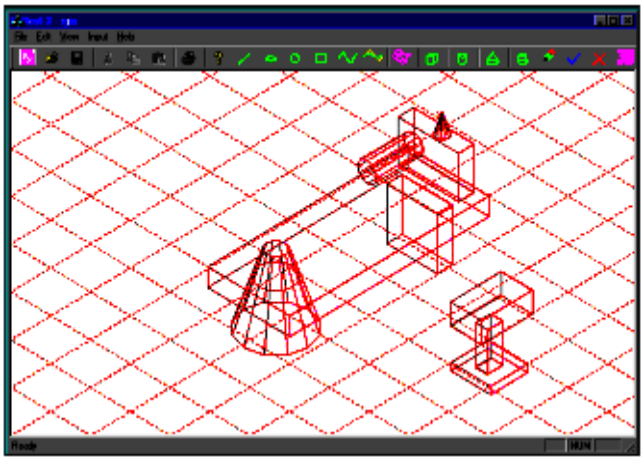


Figure 12 A wireframe model of a scene

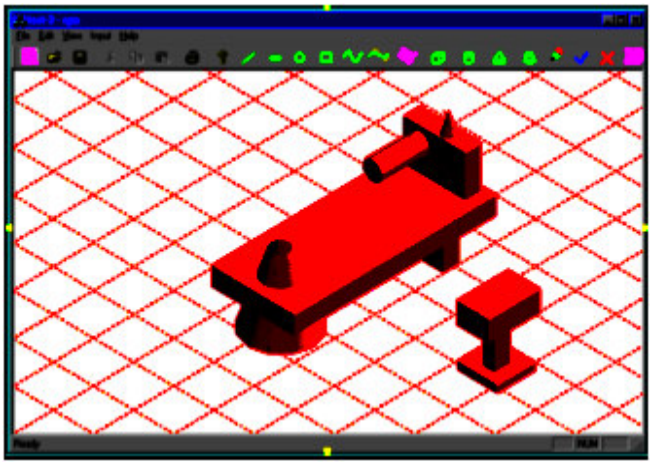
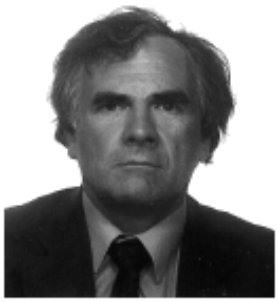


Figure 13 A shaded model of a scene

Author photographs and biographical notes

Sheng Feng Qin is a Research Assistant and a PhD candidate at the University of Wales Institute Cardiff (UWIC). He gained his BEng (1983) and MSc (1988) in CAD from the Southwest Jiaotong University, China. He was an Associate Professor at East China Jiaotong University and an academic visiting scholar at the University of Birmingham from 1996 to 1997. He did his first year PhD in the Brunel University in 1998. 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, subsequently becoming Senior Research Fellow. At UWIC, he was Visiting Professor in the Design Engineering Research Centre (DERC) and Research Director of School of Product & Engineering Design. His teaching and research interests are in CAD, computer simulations of biomechanical systems; virtual prototype of products; as well as CSCW and CAL methodolgies.

Ivan N. Jordanov is a Senior Research Fellow at DERC, UWIC, UK. He obtained BSc (1976), MSc (1978) and PhD (1988) from the Technical University of Sofia (TUS), BG. He was an Associate Professor at the Department of Programming and Computer Applications at TUS. His teaching and research activities include: Programming and Application Development; Optimisation of Dynamic Systems; and Neural Networks Learning.