# Sketching as a Solid Modeling Tool

*Lynn Eggli*
*Beat Bruderlin*
*Gershon Elber*

UUCS-94-029

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

November 9, 1994

## *Abstract*

This paper describes 'Quick-sketch', a 2-d and 3d modeling tool for pen based computers. Users of this system define a model by simple pen strokes drawn directly on the screen of a pen-based PC. Lines, circles, arcs, or B-spline curves are automatically distinguished and interpreted from these strokes. The system also automatically determines relations, such as right angles, tangency, symmetry, and parallelism, from the sketch input. These relationships are then used to clean up the drawing by making the approximate relationships exact. Constraints are established to maintain the relationships during further editing. A constraint maintenance system, which is based on gestural manipulation and soft constraints, is employed in this system. Several techniques for sketch based definitions of 3d objects are provided as well, including extrusion, surface of revolution, ruled surfaces and sweep. Features can be sketched on the surface of a 3d object, using the same 2d and 3d techniques. This way, objects of medium complexity can be sketched in seconds. The system can be viewed as a front-end to more sophisticated modeling, rendering or animation environments, serving as a hand sketching tool in the preliminary design phase.

# Sketching as a Solid Modeling Tool

Lynn Eggli, Beat Bruderlin, Gershon Elber*
University of Utah, Department of Computer Science.
*The Technion, Haifa, Israel

## Abstract

This paper describes 'Quick-sketch', a 2d and 3d modeling tool for pen based computers. Users of this system define a model by simple pen strokes drawn directly on the screen of a pen-based PC. Lines, circles, arcs, or B-spline curves are automatically distinguished, and interpreted from these strokes. The system also automatically determines relations, such as right angles, tangencies, symmetry, and parallelism, from the sketch input. These relationships are then used to clean up the drawing by making the approximate relationships exact. Constraints are established to maintain the relationships in further editing. A constraint maintenance system, which is based on gestural manipulation and soft constraints, is employed in this system. Several techniques for sketch based definitions of 3d objects are provided as well, including extrusion, surface of revolution, ruled surfaces and sweep. Features can be sketched on the surfaces of 3d objects, using the same 2d- and 3d techniques. This way objects of medium complexity can be sketched in seconds. The system can be viewed as a front-end to more sophisticated modeling, rendering or animation environments, serving as a hand sketching tool in the preliminary design phase.

## 1 Introduction

In recent years much effort has gone into making graphical design tools more efficient, providing object-oriented graphical user interfaces with mice. This effort succeeded in making CAD available to drafts people who aren't computer specialists, and also improved their productivity. Nevertheless, these CAD systems are far away from being optimal for an application in the initial design phase. It still takes hours of significant effort to create the three dimensional models for new designs, illustration purposes, or for animation. This is in contrast to the seconds or minutes it takes to quickly sketch an idea on paper that is good enough to convey the essentials of a new concept. With the work presented here, we tried to combine some of the sketching techniques engineers are already familiar with, with the power of computers to build a design tool for two and three dimensional objects. After a brief survey of related work in section 2, section 3 describes the basic functions of interpreting pen strokes as shapes, such as lines, circles, arcs or B-spline curves, and also how geometric relationships are recognized from the sketch, which are used to clean up the drawing and to establish constraints. Section 4 describes in some detail the criteria applied in interpreting the information, and also the type of feedback the system provides to the user. Section 5 describes the constraint system that underlies the editing system; it is based on soft constraints and gestural manipulation. Section 6 describes a couple of sketch based 3d techniques developed for Quick-sketch. A brief description of the software components employed in the prototype implementation is given in section 7.

## 2 Related Work

In this paragraph related work in sketch interpretation, and the use of constraints in design, which has to some degree influenced our development, is described.

The paper "Design Capture System: Capturing Back-of-the-Envelope Sketches" by Hwang and Ullman [6,7] is interesting both for the system implemented and for the background research behind their system. The authors performed an extensive study of mechanical engineers in action. They videotaped mechanical engineers solving ill-defined problems for over 46 hours. One of their chief observations was the central role that sketching plays in the design process. It was hypothesized that sketching is an extension of visual memory. It was also noted that these professionals possess considerable expertise in sketching. This talent is generally unexploited in contemporary CAD systems. This study concluded that to be effective, a CAD system must allow sketched input, have a variety of interfaces, recognize features, and manage constraints. From these criteria they built a system they felt would be useful to engineers. Their system has two phases. The first is a two-dimensional stroke recognition system. Sketched strokes are interpreted as lines, arcs, circles, ellipses, etc. These primitives are accumulated until they can be recognized as a 3D feature. The features are then placed in a coherent three-dimensional topology. New features can be built upon older ones. The limitation with this system is, that it will only recognize a limited set of features (blocks, cylinders and spheres), sketched from a fixed viewpoint.

"Designing Solid Objects Using Interactive Sketch Interpretation (Viking)" [9] lets the user sketch a whole line drawing of an object in 2D, and then attempts to interpret it into 3D. This modeler uses hints like shading of non-visible lines and previous interpretations of the object to guide the interpretation. Additionally, the user can specify constraints on the objects. Sketched segments are automatically aligned, where appropriate, making input easier. The object can be viewed from any view point, and the user can make modifications to any side of the object. There are possibly a few limitations with this system: Forcing the user to shade non-visible line segments seems tedious and error-prone. Furthermore, it seems questionable that the system could successfully interpret a large complicated drawing all at once. Constraints are added in a fill-in-the-blank template instead of using full graphical interaction. The constraints are solved using a relaxation technique which can be quite slow and unpredictable.

"A User Interface Model and Tools for Geometric Design" [1]. In this thesis the author outlines an architecture for graphical user interfaces. He then uses this architecture in a system that builds B-spline curves from sketched data, using an incremental knot removal algorithm. In the next phase, the user is able to edit the curve's control-polygon, once again by sketching. This system considers how the manipulation stroke crosses a segment (or multiple segments) of the control polygon, and determines the user's intent of changing the shape from that input.

"Constraint Objects -- Integrating Constraint Definition and Graphical Interaction" [5]. This paper describes a constraint-based modeling system taylored to be highly interactive. Constraint objects and parameter objects are used to simulate degrees of freedom between objects. In this system the user selects a point. From this point, a constraint-dependency graph is constructed in a non-deterministic fashion. The dependency graph is evaluated while the user drags the selected point. This system also automatically derives constraints from construction operations, such that constraint specification, construction operations, and interactive dragging can be mixed freely. The one drawback this system has is that the response generated from the non-deterministic dependency graph is not always intuitive. It is sometimes difficult to guess exactly how the system will react to a drag operation.

The "DeltaBlue Algorithm: An Incremental Constraint Hierarchy Solver" [4] is an incremental constraint solver. In this system, a new solution to a system of constraints and variables is based on previous solutions. Changes between solutions are assumed to be small. Emphasis is placed on speed over generality. The programmer is able to assign weights to the constraints as they are added to the system. These weights range from an inviolable "hard" constraint to the weakest default "stay" constraint. These constraints form a hierarchy. When a constraint is added or deleted, the algorithm determines which constraints to satisfy (and how) from this hierarchy.

## 3 Interpreting Pen Strokes as Geometric Shapes

A user of the 'Quick-sketch' system draws with a pen, directly on the pressure sensitive LCD screen of a laptop PC. Two-dimensional lines, circular arcs, full circles, and B-spline curves can be sketched. The stroke is sampled as a sequence of points from which the program interprets the type of shape, using some mode dependent preference function. Once the type is decided, the closest fit to the stroke is determined using different numerical techniques (a least square fit approach is taken for lines and B-spline curves; circles and circular arcs are determined using a gridding technique). See figures 1 and 2:

**Figure 1**. Example of sketching primitive shapes with a pen.

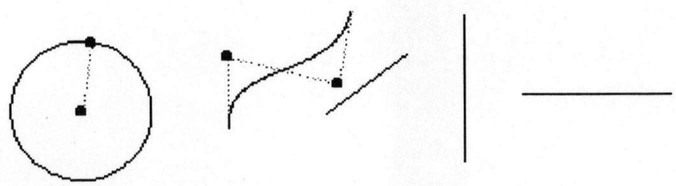


**Figure 2**. The program interprets the stokes as circle, cubic Bezier, line, horizontal and vertical lines.

**Figure 3**. sketching a sequence of strokes.

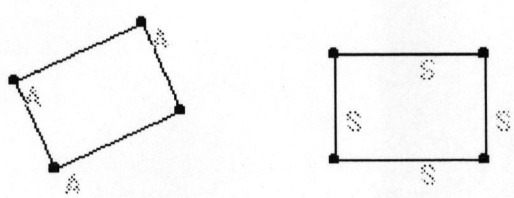



**Figure 4**. The program automatically recognizes adjacencies and the right angles, and cleans up the drawing. Two exact rectangles are created (the second one is axis-aligned). Angle constraints 'A' and slope constraints 'S' are associated with the points and lines, keeping these relations intact when the geometry is changed by dragging.

Once the closest fitting primitives have been found, the system tries to interpret certain relationships between them (e.g whether two curves are adjacent and whether two adjacent lines are at a right angle. If such a relationship is found within tolerance, the parameter of the primitives are altered to establish an exact relationship. Figures 3, 4, 10 and 11 show the effect of the system with a few examples.

## 4 System Behavior and Feedback

'Quick sketch' applies a number of criteria to interpret the type of shape and the relationships between objects. Since the sketched shapes are not usually exact, the system has to apply tolerance in interpretation. If the interpretation does not correspond to the users intention, there needs to be an easy way of recognizing and changing it. This paragraph describes the concepts in some detail.

The interpretation is taking into account the closeness $\varepsilon$ to the exact shape, the speed of the input $v$, a user settable tolerance $\tau$ which is attributed to the user's skill, the length of the stroke (significance $\sigma$), and the user preference (mode $\mu$) that depends on the type of drawing.

Modes: Examples for preference modes are for instance:

*Technical drawing mode*, preferring lines and circular arcs, with right angles and tangencies, parallel lines an concentric circles.

*Symbol mode*, preferring horizontal and vertical lines, symmetric shapes, right angles, parallelism, semi- and quarter- circles.

*Free form mode,* preferring B-spline curves and tangencies.

In each mode a higher tolerance is applied to the preferred attributes, making them a more likely choice.

Significance: The longer a curve is the more accurate it's relative global features will be drawn, since the user will have a chance to do some midcourse correction. For instance, a short horizontal line, in the average, will deviate by a larger angle from an exact horizontal line than a long one. The system multiplies 1/sqrt(length of the line) with the tolerance applied to angles. Similarly other properties are derived using a length dependent tolerance function.

Speed: Sketching an object (for instance a rectangle) very quickly, will be less accurate, in general, than drawing it more carefully and slowly. The speed of a stroke is taken into consideration by multiplying all the tolerances with the speed. So, the faster an object is drawn, the higher is the tolerance used in the interpretation and thus it is more likely that a correction is applied. This approach is assuming that the faster strokes are more sloppy, and in reality should represent some more accurate shape. This behavior can also be exploited, for instance, to draw a line that has a slight slope, and therefore should not be interpreted as horizontal by drawing such a line more carefully, and deliberately.

Skill: The speed factor is compensated somewhat by a user settable skill factor. The more skilled a designer is the less his or her drawing should be attempted to correct, even if drawn fast.

*Cleaning up the drawing and constraints*: After their determination, the sytem tries to satisfy the relationships with the *least amount of change*. This paradigm is applied on a case by case basis, using some heuristic rules, as shown in the following example: The angle between the two lines in figure 5 is close to a right angle (within tolerance). The angle can be corrected by rotating the second (vertical) line about the point of incidence. This, however, would make the other end of the line jump unexpectedly, and would suprise the interactive user, even if a right angle was intended. The system uses an alternative solution, namely to rotate the second line about its center point, and to elongate the first line, to maintain incidence between the lines. Generally, the system attempts more distributed small changes, rather than causing one big local change. After the correction is made, a right angle constraint is established between the two lines.

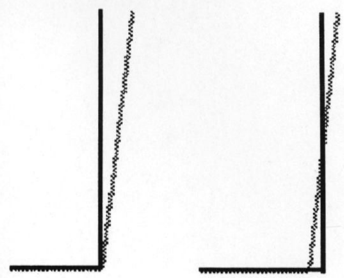


**Figure 5**. Snapping to a right angle constraint (2 possibilities shown)

*Correction of automatically made assumptions*: The criteria applied in making automatic interpretation work together naturally, and make the behavior of the system very predictable. Nevertheless, there will always be cases where the system does a wrong interpretation of the users intent. It is therefore essential to provide the designer with clearly understandable feedback about the interpretations made, and a simple way of correcting them, if they are wrong. Each interpretation made by the system will result in highlighting of corresponding icons that easily identify the choice. The user can quickly unselect the highlighted constraint and/or select others instead, and the program instantly reinterprets the stroke.

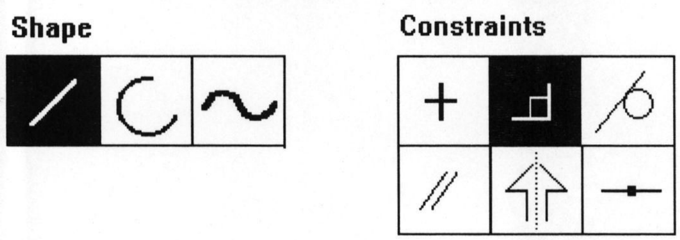



**Figure 6** Highlighting the system's choice: The last stroke was interpreted as a line with a right angle constraint.

Using the context sensitive menu interaction after-the-fact is a very efficient way of interaction especially for pen based computers. Since a pen provides a lot of dexterity, in the vast majority of cases the automatic interpretations are correct and easy to predict. With mouse based interaction it is necessary to first tell the system each time whether we want to draw a line, B-spline curve, or a circle, and then a few limited options of determining the shape interactively, by dragging the provided controls into place, are offered. With a pen we just draw the object and let the system figure out the type of object, the parameters, and the relationships to other objects. Only the menus/icons relevant to the objects drawn will then be displayed, so the user can make possible corrections. Together these approaches speed up the design significantly, as the examples show.

## 5 Constrained Manipulation, Gestural Manipulation, and Soft Constraints

Quick sketch allows for interactive manipulation of sketches, by dragging the control points displayed in the drawing. The previously established constraints are maintained during dragging. 'Gestural manipulation' is used to disambiguate the interaction for underconstrained drawings, taking the direction of the stroke into account. Implicit (soft) constraints have also been introduced to achieve even better predictable behavior when manipulating underconstrained drawings. The following example shows how the system reacts when dragging points of a profile, containing a circular arc tangent to two lines, and two right angles at both ends (figures 7 and 8).

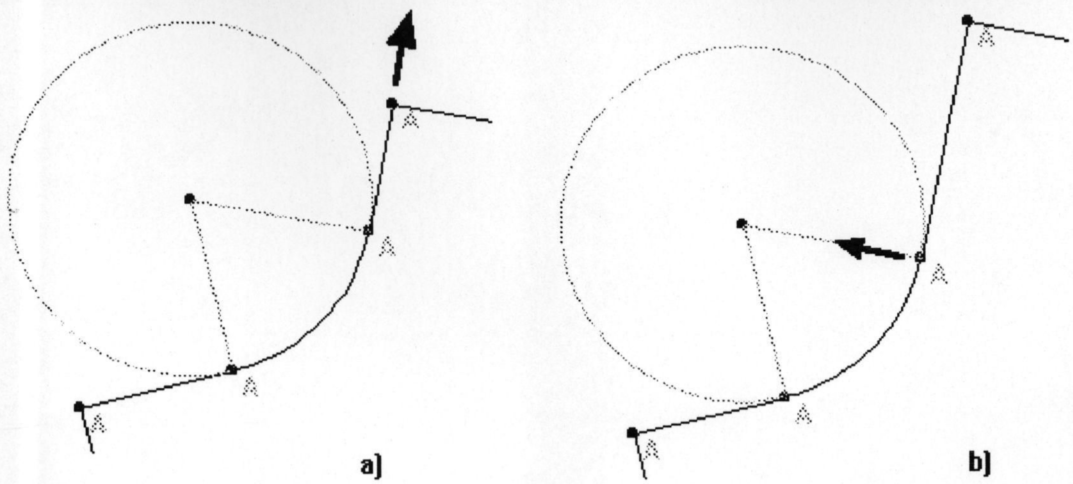



**Figure 7 a)** Dragging the point at the top upward (see direction of arrow) will stretch the upper portion of the profile (see figure 7 b). Notice that all the previously imposed constraints, such as right angles, and tangencies, are maintained during all manipulations. **b)** Dragging a point on the peripery of the circle inward causes the circle to shrink in size (see figure 8a)

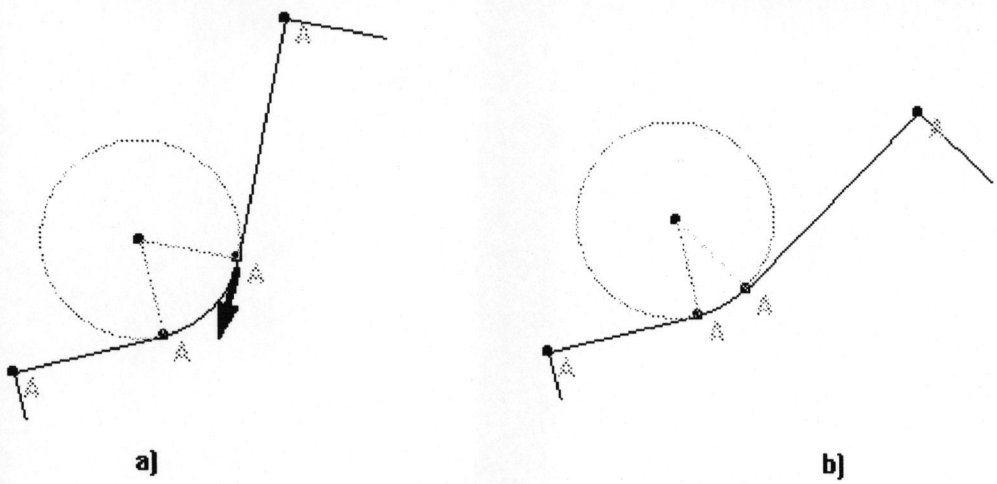



**Figure 8 a)** Dragging that same point tangential to the circle causes a change in the opening angle. **b)** The change causes the upper half of the profile to be rotated about the circle center due to the constraints.

The idea behind gestural manipulation is to use the direction of the pen stroke (gesture) in determining which effect a manipulation will have. In simple terms, the direction of the stroke is compared with the directions of the lines adjacent to the point picked. In case of circles or B-spline curves the control polygons are used as a reference (drawn dimly in the illustrations). If the stroke is (within tolerance) along such an existing line, the constraint solver tries to achieve a one dimensional degree-of-freedom motion in this direction. Gestural manipulation for 3d interaction was preiviously proposed in [8]. Due to the increased dexterity of a pen over a mouse, this type of interaction gains importance.

The possible reaction to manipulation, despite using gestures, may still be ambiguous, especially, if only a few constraints are defined. The following example shows how implicit constraints can be used to obtain predictable behavior, even in highly underconstrained situations. The polyline in figure 9 has no explicit constraints. Implicit constraints automatically define distance constraints between pairs of points connected by a line. Angle constraints between two adjacent lines, position constraints for each point, and a slope constraint for each line are also defined implicitly. In contrast to explicitly defined constraints, these implicit constraints are so called soft constraints, i.e. they are only observed if they do not contradict any explicit (hard) constraints. Each type of soft constraint has a mode dependent weight associated with it. The weight is interpreted as a 'penalty' for violating that constraint. For hard constraints the penalty is infinity (the ultimate penalty). When manipulation an object the constraint solver tries different ways of transforming the objects while dragging. For any given plan, some of the soft constraints will have to be violated. In the planning phase the weight of each violated constraint is summed up. Several plans are analyzed, and in the end the one with the least penalty is used. Giving different weights to different types of constraints will effect different behaviors in interactive situations. In our system we do not put the burden to assign weights to the individual constraints on the user, but rather provide predefined sets that can be associated intuitively with some geometric behavior. The weights are summarized in the following table:

| *Mode* / constraint type: | position | slope | distance | angle |
|---|---|---|---|---|
| *rigid* | 0 | 0 | 10 | 10 |
| *bend* | 5 | 0 | 10 | 0 |
| *stretch* | 5 | 10 | 0 | 10 |
| *free* | 5 | 0 | 0 | 0 |

In rigid mode, a high penalty is associated with violation of distance and slope constraints but no penalty is associated with position and slope constraints, causing concatenated primitives (e.g. in a profile) to be translated or rotated as a rigid object.
in stretch mode no penalty is associated with distance violations but slopes and angles carry a high penalty. The penalty on position constraints keeps the transformation more local which causes the obects to be streched locally.
In bend mode angle constraints have no penalty but the system tries to maintain distances causing a kind of a bending or shearing transformation.
In free mode only position constraints carry a penalty causing free local deformations, maintaining neither angles nor distances.

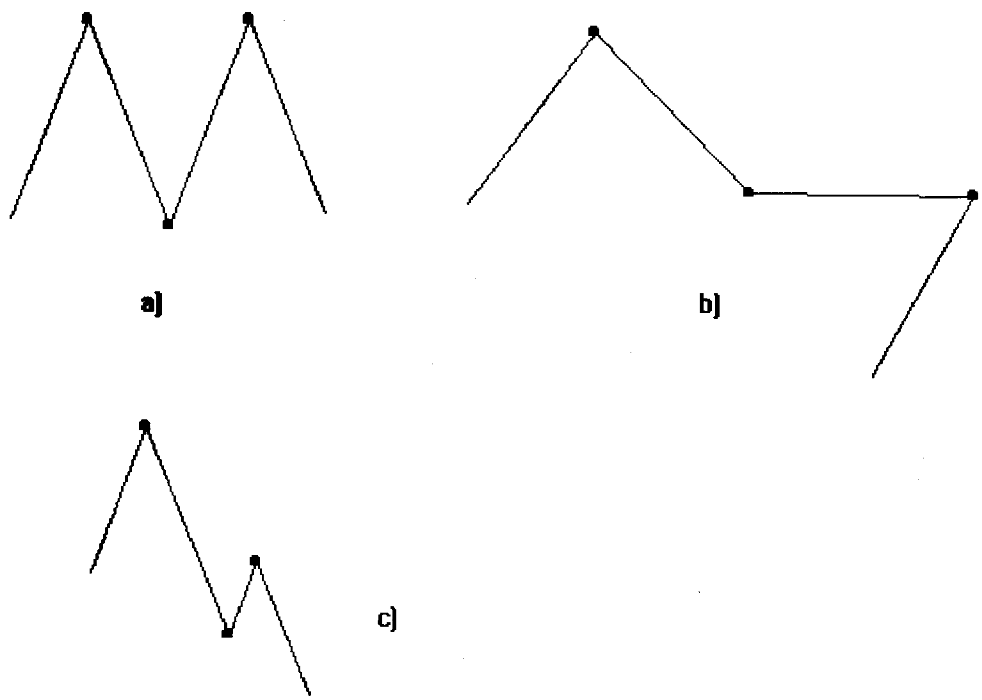



**Figure 9** Examples: The polyline (a) is edited in bend mode (b), or in strech mode (c)

The way the soft constraints are treated in the constraint solver bears some resemblance to the hierarchical constraint solver described in [4], however, the solving mechanism used here is quite different. The details of the mechanism cannot be described in the space provided here, but instead we refer to [3].

## 6 Sketching in 3D

To model three dimensional objects by sketching, the system currently allows the following techniques: Two-dimensional profiles can be extruded by sweeping them along a straight line. Lines and curves can be sketched on planar faces of existing objects. This way, features can be quickly added to objects. All the previously introduced 2d techniques also work for any new profile sketched onto a face (eg. detection of parallelism, snapping to right angles, tangent circles, constrained manipulation etc.) An example sequence for sketching a 3d object can be seen in figures 10, .. 13.

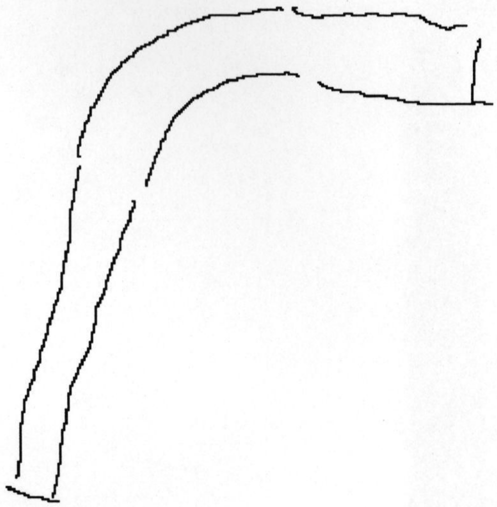


**Figure 10**. We start with sketching a 2d profile of a mechanical part.

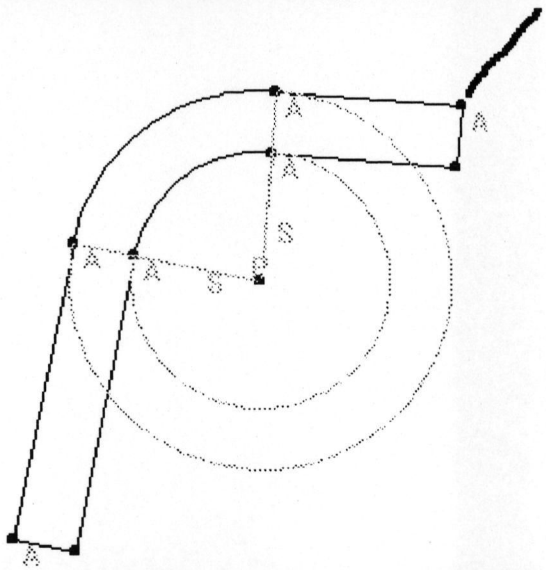
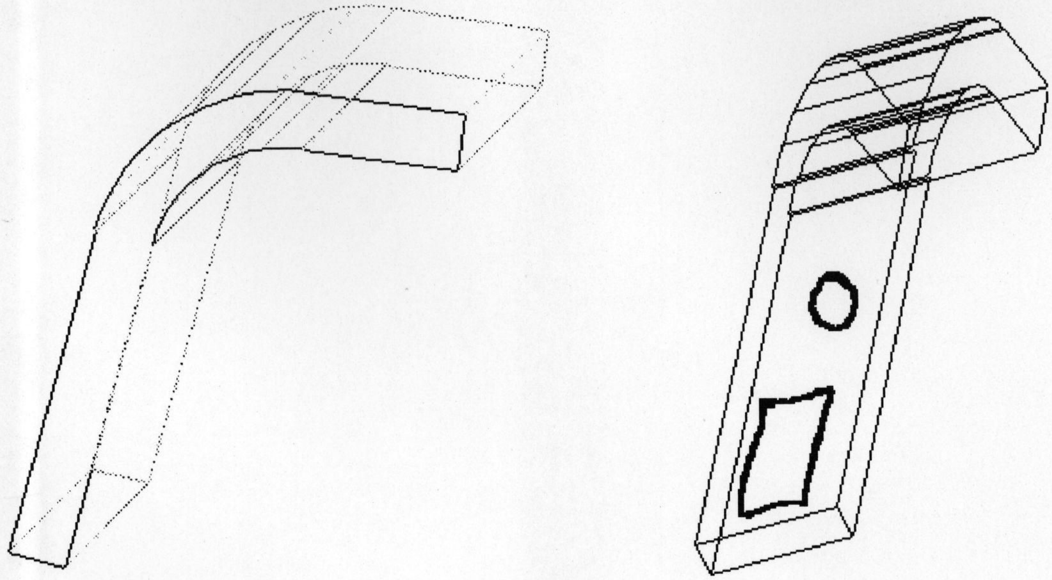




**Figure 11**. The pen strokes of figure 10 are recognized as circular arcs and lines. Also, the lines are reconized to be tangent to the circles. Parallel lines, concentric circles, and right angles are interpreted. The drawing is cleaned up instantly, and constraints are established. The 2d profile can then be extruded into the third dimension by a stroke (near the top right), resulting in a solid object (see figure 12).



a]　　　　　　　　　　　　　　　　b]

**Figure 12 a)** The resulting 3d part. **b)** Features can be sketched onto any flat surface facing the viewer in the current perspective.

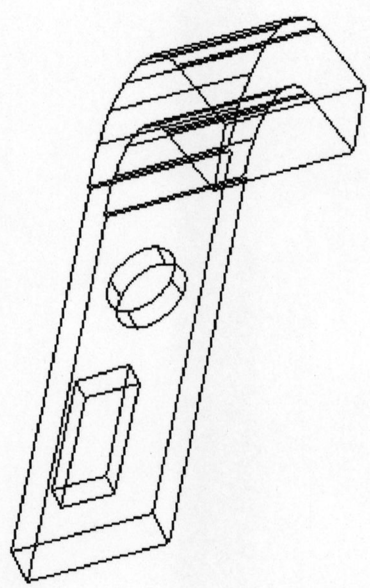


**Figure 13**. These features are also cleaned up by establishing right angles, and aligning them with the boundary lines.

The interactive modeling of this solid from scratch took about 35 seconds, which was even slightly faster than trying to sketch the same object with pencil on paper and has the additional advantage of having a full 3d CAD model that can be edited, dimensioned, and rendered, etc.

In addition to extrusion of features, we adopted the following standard surface modeling techniques into the sketching environment:

**Ruled surfaces** can be defined between two sketched curves. (see fig. 14, 15). A sketched cross section can be swept along a sketched curve (figure 16), creating a **sweep surface.** A **surface of revolution** can be created, by simply sketching two approximately symmetric silhouette lines (figure 17).

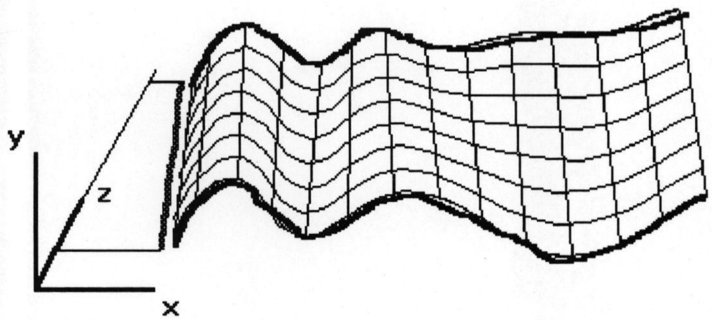



**Figure 14**. Ruled surfaces: Interpolating between two spline curves. The two curves are first interpreted in the x/y-plane. The stroke between the two curves is then used to determine the depth. It is first projected onto the x/z plane and then its projection onto the z-axis determines the offset between the two interpolated curves in the z-direction, assuming that it is parallel to the x/z-plane.

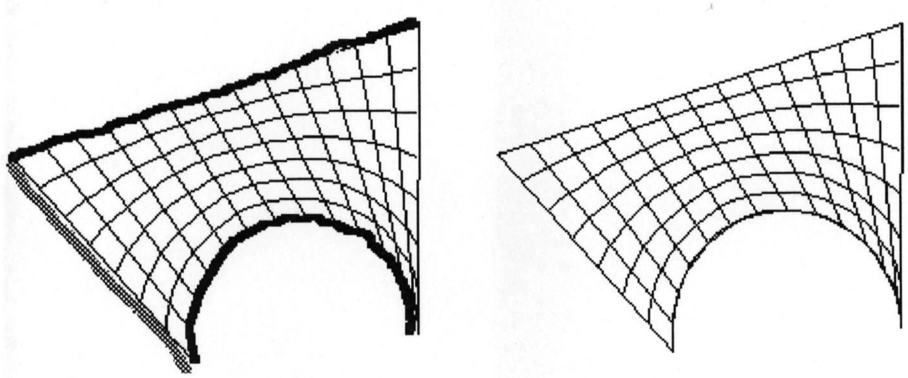
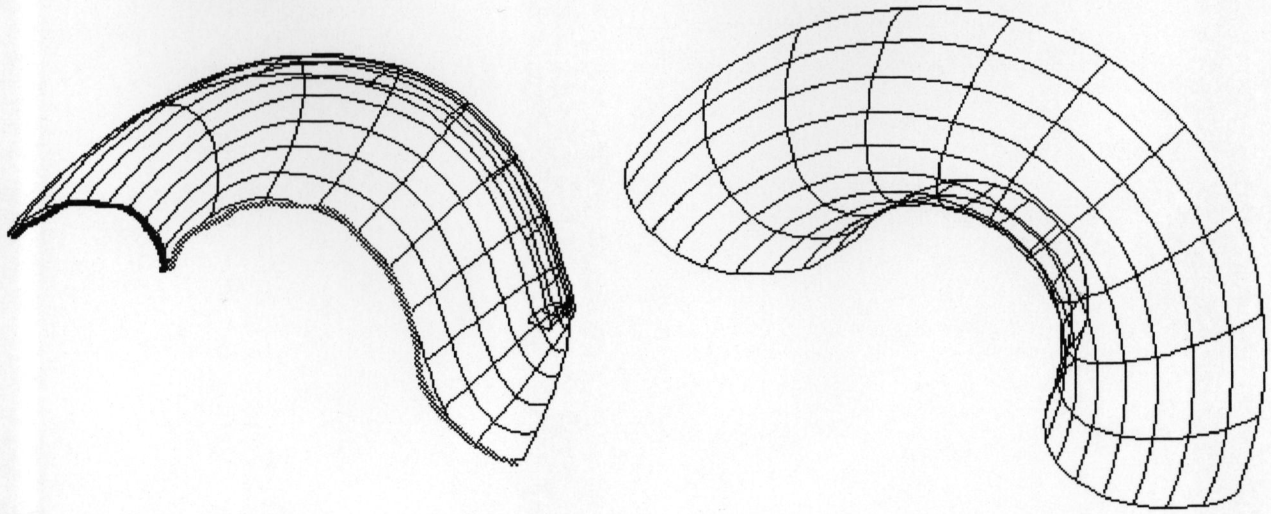


**Figure 15** A ruled surface is created by interpolating a straight line and a circle.



a) b)

**Figure 16** Sweep surfaces: Sweeping a circular arc along a B-spline curve creates a sweep surface, as shown in (a). The stroke for the cross section curve is interpreted as a projection onto the x/y plane, and the sweep cuve is projected onto a plane parallel to the x-z plane. In (b) a circular arc is swept along another circular arc, creating a portion of a torus.

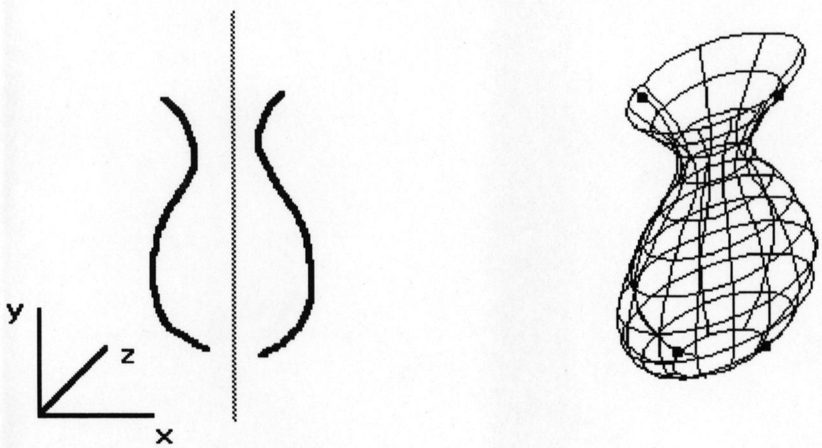



**Figure 17.** A surface of revolution can be created by sketching two approximately symmetric silhouette lines in the x-y plane. The system determines the symmetry axis to be parallel to either the x-axis or the y-axis.

## 7 Implementation

The prototype implementation of Quick-sketch was realized with GDI, a portable graphical user interface toolkit, and the 'IRIT' computer aided geometric design package.

The 'IRIT' libraries [2] provide us with the necessary mathematical tools, such as least square fit, calculating B-splines curve of arbitrary orders and degrees of freedom through the sampled points of a pen stroke. Functions for creating surfaces of revolution from two silhouette curves, for constructing sweep surfaces from cross section and axis curves, and for Boolean sum surfaces or Coons patches are also available in IRIT.

The GDI library [10] provides the graphical user interface to the system. GDI consists of 2d gadgets and 3d interactors and display functions have been implemented as C++ classes in a portable way. An interactive geometric modeling system has been implemented, using GDI. The portability of the library allows this modeler to run on PCs under Windows 3.1, as well as Unix, with X-windows (with or without Motif), virtually unmodified.

## 8 Conclusion and Outlook

Our experience drawn form the development of 'Quick-sketch' is that in 2d the automatic interpretation of a pen stroke is a powerful technique. "Reading the users mind" can be done reliably, since the input device (pen) is also two dimensional. The extra control and dexterity provided by the pen allows for these new techniques that would have been unpractical with a mouse. The automatic interpretation is generally correct, and in the few cases where it is wrong it can be corrected with a press of a button. The way of designing with sketches feels very natural and is also very efficient.

In three dimensional design, the situation is much more difficult. Interpreting an arbitrary 2d input as a 3d object is too ambiguous, in general. We decided against this idea, and instead, we developed specific drawing techniques that have an unambiguous interpretation in 3d. These techniques are mostly adaptions of conventional 3d techniques for a sketch based environment, in combination with the new 2d sketching and manipulation techniques. In the future we plan to add many more techniques specifically for free-from surface design, for which sketching can be a powerful technique. Each of these techniques will be quite simple and therefore certainly limited. However, we feel that a transparent combination of a few such techniques will add up to an extremely poweful tool.

The use of geometric constraints has proven to be a powerful tool to express design intent, especially in the preliminary design phase, where the exact shape is not generally known. The geometric relationships can be derived from the sketch input in many cases. These constraints become part of the designed object. In the later design phases the shape of the model can be refined and modified, for instance, by adding more features, adding more dimension information, and editing the constraints.

Our goal is to make sketching with the computer as natural as using paper and pencil, and even more efficient. It should be possible to jot down an idea in a few minutes or even seconds, avoiding the tediousness of current drafting packages. Sketching makes it worthwile using the computer in the preliminary design phase, during which ideas are still developed. Up until now, in this stage designers still use pencil and paper. Only after an idea is almost completely thought out it has to be transferred to the computer, by hand. With the new technology being developed, a three dimensional model is in digital format even during the conceptual phase, and can directly be used for refinement in the later stages of design.

## Acknowledgements

This work was supported in part by grant No. 92-00223 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel.

## References

[1] Banks, M. A user interface model and tools for geometric design. Master's thesis, University of Utah, 1989.

[2] Elber, G. IRIT 4.0 User's Manual.

[3] Eggli, L. Sketching with constraints. Master's thesis, University of Utah, 1994.

[4] Freeman-Benson, B. Maloney, J. The delta-blue algorithm. Tech. rep. 99-11-09, University of Washington, 1988

[5] Hsu, C.-Y, Bruderlin, B. Constraint-objects - integrating constraint definition and graphical interaction, Proceedings Second Symposium on Solid Modeling and Applications, 1993.

[6] Hwang, T. Ullman, D. The design capture system: Capturing back-of-the envelope sketches. Journal for Engineering Design 1, 4, 1990.

[7] Hwang, T. Ullman, D. Recognizing featues from freehand Sketches. Computers in Engineering - 1994 Vol. 1, ASME.

[8] Nielson, G, and Olsen, Jr., D. Direct Manipulation techniques for 3d objects using 3d locator devices. Proceedings of the 1986 Workshop on Interactive 3D graphics (1987)

[9] Pugh, D. Designing solid objects using interactive sketch interpretation. Siggraph Proceedings, 1992.

[10] Salem, M. Skowronski, S. and Bruderlin, B. GDI reference manual. Tech rep. UUCS-92-031, University of Utah, 1992.