ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

αμηον.

Techniques For Dynamic Interactive Sketch Recognition

Neal Baker

A thesis submitted in partial fulfilment of the requirements of the Council for National Academic Awards for the degree of Master of Philosophy

October 1990

This work was hosted, equipped, and funded by Plessey Networks and Office Systems Ltd., Nottingham, England, under Esprit project 295.

ABSTRACT

The objective of this work is to develop pattern recognition techniques which will render hand-drawn sketches to draughtsman-like quality, suitable for real-time operation in an interactive human/computer interface. The intended use is as a means of entering graphical information in the office environment.

The recognition process has been rationalised into discreet stages with the intention of a) reducing complexity, by simplifying the interface between different recognition functions, and b) providing a suite of recognition functions, suitable for use in a variety of sketch recognisers, of varying sophistication.

Drawing takes place with a pen on a digitising tablet. The position of the pen on the tablet is sampled at frequent time intervals, thus the pen path is digitised into a stream of points. Recognition stages envisaged are:-

Input Filtering	(chapter 5)
Stroke Segmentation	(chapter 4)
Curve Fitting	(chapter 6)
Stroke Analysis	(chapter 7)

Shape Matching, and Hierarchical Analysis (which are outside the scope of this work).

Input Filtering filters out sampled points which are determined to be too close to the previous sample to be useful to the stroke segmentation process.

Stroke Segmentation distinguishes straight from curved regions in a pen stroke. The angular variation in the pen path is monitored using a scan-along algorithm to classify regions. As soon as there is a break in classification Curve Fitting renders the recognised region of the stroke with a geometric line or curve.

Stroke Analysis compares a newly fitted line to previous lines in the sketch in an attempt to merge it with previous lines. This is in response to the observed drawing habit, that a user sometimes represents a single line by drawing it in several fragments. Detection at this stage, using heuristics, simplifies the task of Shape Matching.

Shape Matching takes a newly fitted, or merged line, and searches the previous lines in the sketch, to recognise elementary shapes from a given set of templates.

Hierarchical Analysis is a generic term for further stages of recognition, probably employing spatial relations between elementary shapes and lines, and grammars.

The stages of recognition addressed in this thesis are Input Filtering, Stroke Segmentation, and Stroke Analysis. Shape matching and hierarchical analysis, beyond the scope of study, are higher levels of recognition which can benefit from the preprocessing studied here.

Contents

1. INTRODUCTION

1.1 Objective of the Sketch Recognition

The objective is to render a hand drawn sketch to a drawing of draughtsman like quality, in real-time. It is intended for use in interactive drawing composition and be completely user independent. It is anticipated that pen driven editing will be used in conjunction with sketch recognition so that the recognised drawing can be corrected or amended in a highly interactive way.

1.2 Class of Sketch Considered

The word 'sketch' encompasses a wide variety of drawing subjects and styles, from an artist's shading kind of scribble to a draughtsmans careful calculated composition. I use the word sketch to mean an outline drawing, most likely to be found in a business document e.g. pie chart, histogram, tabular form, schematic diagram. It is not planned to consider gestures for drawing shading or filling. I hence define a sketch to be a series of hand drawn lines intended to represent geometric objects which may compose higher level contextually meaningful objects.

1.3 The Nature of the Problem

Because recognition takes place as a person draws the recognition process can take full advantage of information contained in the movement of the pen in relation to the movement so far. For example, at the instant the pen turns a

corner having previously been moving in a straight line, the straight line can be delimited, and the hand drawn element replaced with a geometric element. Likewise, when the pen begins moving straight having previously been describing a curve, the hand drawn curve can be replaced with a geometric curve. In this way a hand drawn stroke can be segmented and fitted with geometric line types. This is the first level of recognition, Stroke Segmentation.

Samples have shown that the level of human inaccuracy in sketches is such that at best only straight lines and curves can be distinguished. Curves of different types are completely ambiguous without the presence of contextual information. In the recognition of 'pie' and 'gate' sample sketches (sections B.2, B.4) it is not known in the Stroke Segmentation phase whether curves should be circular, elliptical, or other. Therefore, it is necessary in Stroke Segmentation to approximate a hand drawn curve by a piecewise fit with a single general curve type. The correct curve type can be replaced by a subsequent contextual level of recognition.

As sketching proceeds simple geometric objects become discernible such as rectangle, triangle, circle etc. and parts thereof. For example, a rectangle becomes discernible when its fourth side is completed. Therefore a phase of sketch recognition is needed which is invoked each time a stroke segment is complete. This is the Shape Matching phase, replacing a piecewise fit of line segments with a neat syntactic shape.

As elementary shapes and lines assemble, more complex objects become discernible making further refinements possible such as fitting different curve

types. Rules relating to connectivity, orientation, and proportional and positional relationships. This is a hierarchical process.

Many aspects of the recognition process depend on the context of the sketch. For example, requirements for orientation and connectivity vary considerably between pie charts and circuit diagrams. The lowest level of recognition, the Stroke Segmentation phase, is the only one which can be completely independent of context.

1.4 Strategy Overview

The process of structural pattern recognition can be considered as a series of discrete phases shown in figure 1-1 in order of increasing abstraction from the original hand drawn input. The Sketch Structure is a model of the recognised sketch, which evolves as drawing proceeds. It is manipulated by the surrounding recognition phases which maintain a hierarchical structure of data elements, viz: a complex shape recognised by Hierarchical Analysis is composed of shape elements and line elements; shape elements recognised by Shape Matching are themselves composed of line elements, which are recognised by Stroke Segmentation, and modified by Stroke Analysis.

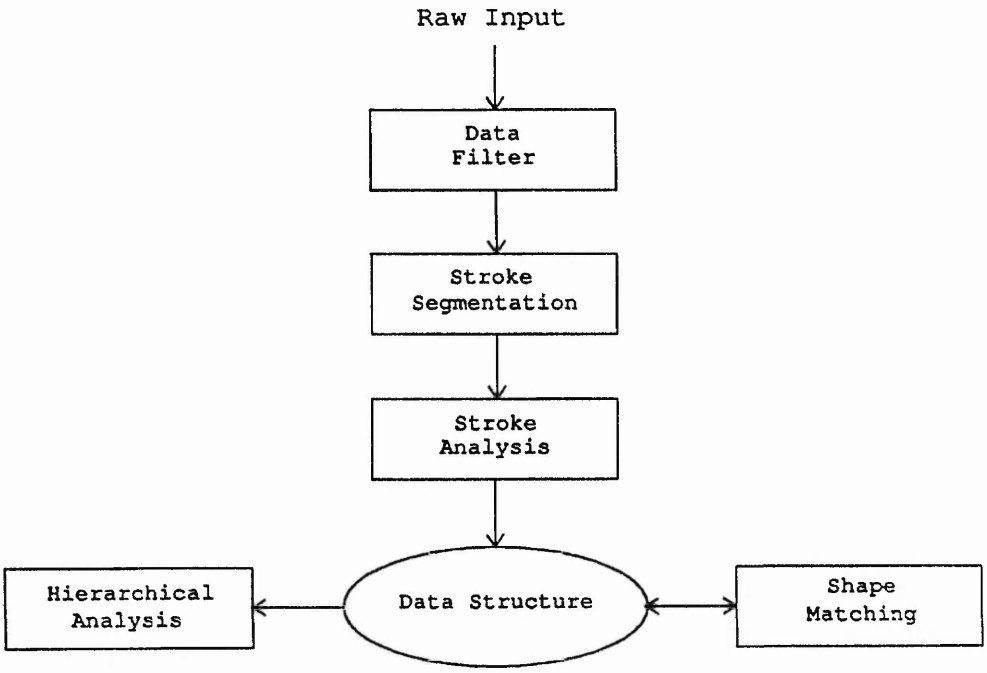


Figure 1-1

There are two advantages of rationalising the recognition process into functional modules with well defined interfaces. Firstly, it has enabled recognition to be developed in discrete phases, with minimal dependency on others. Secondly, having a range of recognisers of increasing degrees of refinement, makes them suitable for a range of uses, of increasing sophistication.

2. REVIEW OF RELATED WORK

To date there are precious few examples of recognisers that recognise sketches as they are drawn. Even the latest generation of flat screen and stylus workstations such as the Grid Pad and ABC Telestation do not enable the user to enter graphics by hand. A search through the IEE Inspec catalogue of publications at the start of this study found only one. A second was published later. Lessons learnt from these are described in the following sections.

Relevant techniques have also been found in static (OCR type) recognition of line structures where an image of a line structure is digitised into electronic form by a scanner, then skeletonising techniques are applied to produce a coordinate sequence describing the line structure. At that stage the data is in a form similar that from a digitising tablet, except that the pattern is complete and devoid of information about speed of pen movement. However, some of the techniques subsequently employed to recognise geometric line segments are relevant and are discussed in section 4 in considering techniques for stroke segmentation.

2.1 Review of HUNCH

2.1.1 Overview

HUNCH [3] was the sketch recognition system developed by the Architecture Machine Group, in Nicholas Negroponte's Department of Architecture, MIT, to investigate the entry of draughtsman quality drawings to a computer, by means

of rough hand sketching. Initially it was a post-process, taking as input a finished sketch. It was found that this mode of working would require sophisticated artificial intelligence to successfully determine the user's intentions in the sketch. Subsequently, the system was made interactive, so that:-

- the user was given constant feedback of the recognised rendition;
- the sketched evolved in relation to the rendition;
- the user could make corrections if the rendition was not what they intended.

2.1.2 Objectives of the Project

The objectives were to attempt to answer the following questions:-

- Does there exist a syntax of sketching, independent of semantics ?
- Could a machine make useful interpretations of a sketch without knowledge of the subject domain?

2.1.3 Methods of Recognition

The process of sketch recognition was identified to be a series of sequentially dependent tasks: segmentation, line classification, latching, and overtrace removal.

Sketches were done with pencil and paper. The strokes of a pencil sketching on a tablet, digitised at 100 samples per second, were captured to a file. The

recognition process was operated off-line (the user having no interaction with it), taking a digitised sketch file as input, and producing a rendered line drawing.

2.1.3.1 Segmentation and Line Classification

Initial assumption was that pen speed is a measure of the user's intent - a quickly drawn line is meant less literally than a slow one. The observation was made that pen speed decreases at corners, and this was intitially used to detect corners. Straight lines were fitted between corners.

Curves were considered to be a special case of corners. Angular variation was used as a secondary characteristic to pen speed, in distinguishing curves from gentle corners. A region of the pen path with little speed decrease, or low rate of change of curvature was interpreted as a curve.

These methods did not give the same interpretations as human observers, and were more successful for some users than others.

2.1.3.2 Latching

Where the ends of two or more lines lie close together it may be true that they were intended to join. Latching is the act of joining them. Using local endpoint criteria for detecting the need for latches HUNCH investigated the feasibility of latching in the absence of contextual or syntactic guidance.

An initial criteria for latch detection used a fixed proximity between endpoints. This often produced bizarre results. It was most unsuccessful with lines of very different lengths.

The next attempt adjusted the latching radius according to average pen speed, still assuming that pen speed was a measure of intended accuracy of a line. This was also unsuccessful.

It was concluded that pen speed does not always indicate intent.

Gridding was also rejected, since this requires the user to watch the display for a snap rather than concentrate on the paper. This would interfere with the mental processes involved in developing ideas on paper.

Indications about latching were that syntactic guidance is needed. No other work on local latching criteria was found in the literature.

2.1.3.3 Overtracing

The HUNCH team noted a sketching habit which they called 'overtracing', where a person uses several superimposed pen strokes to represent one line. It can be seen in the first few samples of houses in section B.3. The phenomenon was found to be often used to emphasise a line's certainty or importance, but not always. It may just be incidental. HUNCH attempted to distinguish between carefully drawn parallel lines, and overtracing. Overtraced lines can be merged into one. No details are given of methods for eliminating overtracing.

2.1.4 The Need For Context

To perform automatic sketch recognition completely successfully, the HUNCH team realised that the recognition system needed to be guided by sophisticated knowledge of the sketch context, and of the user's idiosyncrasies. For example, if a building is being sketched, knowledge about how buildings are structured, and the inter-dependences between constituent parts, are the kinds of knowledge that need to be modelled, and be used to guide even the lowest levels of recognition.

However, it was shown that by making the recognition process interactive with the user, misinterpretations that HUNCH made could be corrected by the user.

This showed that by exploiting the *user's* knowledge of his intentions, artificial intelligence can be avoided.

2.1.5 Interactive HUNCH

Line segmentation and fitting were done in real-time, as the user was drawing. Latching and overtrace removal were done in the background. The user sketched in relation to the recognised drawing. Editing facilities were provided for the user to make corrections. No comment was made on user acceptance.

2.1.6 Conclusions

The questions that the project sought to answer and the findings are as follows:-

- Does there exist a syntax of sketching, independent of semantics ?

  No, even the style of an individual will vary according to their mood, urgency etc., but sketching was confirmed to be a viable means of entering a draughtsman quality drawing.

- Could a machine make useful interpretations of a sketch without knowledge of the subject domain?

  Yes, an interactive process employing the user to modify the rendition to their intentions is the most feasible approach.

Further work is needed to employ contextual knowledge in the recognition process.

2.2 Review of MIRABELLE

2.2.1 Overview

MIRABELLE was a sketch recognition system, initially developed at the University of Nancy, France [16], and later enhanced by the addition of contextual guidance [19] at the Centre of Research and Information, Nancy. Here we consider the early stages of the recognition system, those of filtering out redundant points from the digitisation, and segmenting and classifying pen strokes.

The techniques that they rejected, and those that they chose are well described in [16] which is included in the appendix of this thesis. The techniques have been studied as part of this work, and the filtering algorithm tried out. Here I present a critique of weaknesses found that I hope to improve on.

2.2.2 Filtering Algorithm

The purpose of a filtering algorithm is to remove points from the input stream that are either redundant to the line segmentation process, or would cause inappropriate segmentations.

The filtering algorithm for Mirabelle discards a point if it is either,

a) within a certain distance of the previous point, OR

b) within an angle of about 15 to 20 degrees from the previous two points.

These citeria cause most points to be discarded. Discarding points means less work for the segmentation process, but care must be taken to retain corner points. Unfortunately, when the exact algorithm described in the paper was tried out, it was found to discard **all** corner points. Because the pen slows down at a corner, most of the points which you would subjectively identify as 'the corner' are bunched together. The criteria above discard points that are bunched together, even if they are over 20 degrees from the previous ones.

I used the angle criterion above to prevent filtering out corner points in this project, discussed fully in section 5.1 Filtering Criteria.

2.2.3 Stroke Segmentation and Classification

There is no segmentation of pen strokes in Mirabelle, a line is classified only when the pen lifts. There can be no rendition 'on-the-fly' as the pen turns a corner, or moves from straight to curved. The filtering algorithm produces a polygonal approximation of a pen stroke, which is then analysed to identify drawing primitives from a syntactic set.

Curves cannot be distinguished from a series of straight lines because the filtering algorithm discards most of the points describing a curve.

2.2.4 Latching

Rote latching was tried, joining the ends of lines that lie close. Findings concur with those of HUNCH, that latching can only be done reliably with contextual guidance.

3. SKETCH SAMPLES FOR OFF-LINE TESTING

Test patterns that are repeatable and deterministic were needed for developing and testing the recognition techniques. To ensure that they are representative, test patterns should be actual human sketches. Because a person cannot repeat a sketch with exactly the same patterns, samples of sketches were captured on a digitising tablet and stored as files of coordinate pairs, with the dynamics of pen movement preserved. These were used as repeatable test data. A captured sketch is a complete sketch, whereas the techniques being investigated here are required to work on an evolving, partial sketch. Therefore, they were tested and developed on the sketch samples and later refined on-line, and validated for real-time use.

Samples of sketches were taken from a variety of people: left-handed, right-handed, male, female, British, German, Italian, westernised ethnic, mainly from engineering and secretarial professions, but also from draughtsmen, nurses, a doctor, and a nun. Initially around 100 samples were taken, using different digitising tablets to ensure that the Stroke Segmentation techniques were not influenced by any particular tablet's characteristics. Calcomp, Numonics, and Penpad, tablets were used. The library was later increased to around 200 samples using the Numonics, which proved the most accurate and consistent. Each donor was asked to copy, or occasionally memorise and repeat each picture in a target set, shown in figure 3-1. Copying does not invite all the sketching habits associated with free-form origination of ideas on paper. For example, overtracing (emphasising a line by re-drawing over it) is noticeably

rare in the samples, but is common in free-form sketching [3]. Such habits that were not well represented in the samples were investigated on-line.

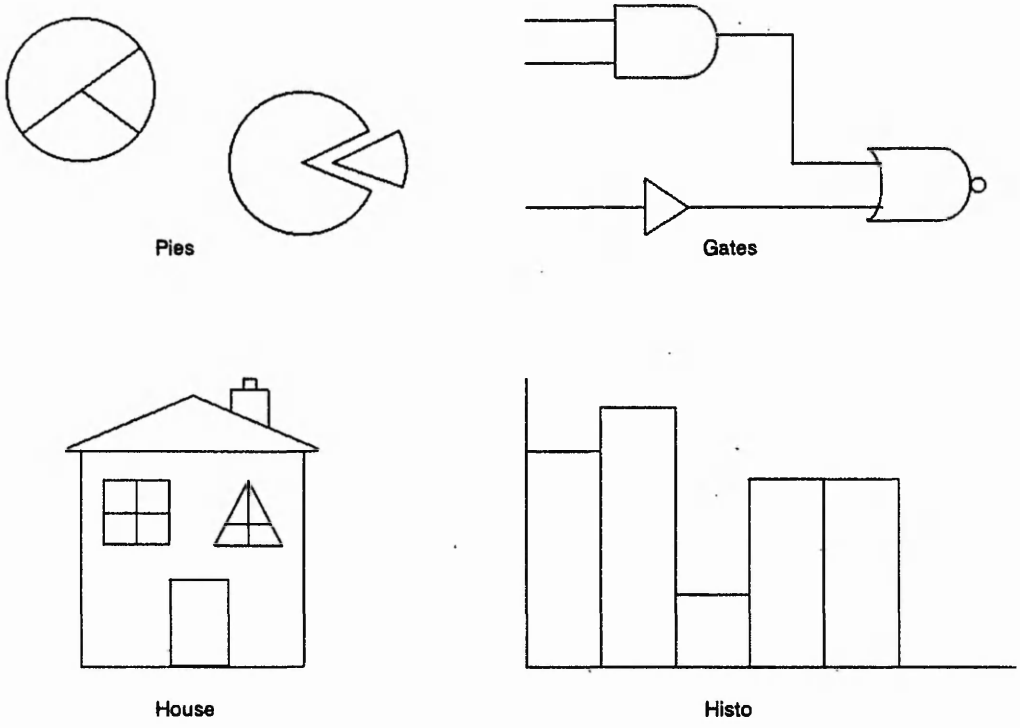


Figure 3-1 Targets for sample sketches.

The set of picture targets was designed to be representative of the kind of graphical objects that are likely to be encountered in office documents. Some examples of sketch donations, which have been used in developing the recognition techniques, are shown in figure 3-2. Analysing the ways that these sketches approximate the targets in figure 3-1, it can be seen aberrations and deformations in shape abound. Many lines, which are intended to be straight, are significantly bent. Single lines are drawn in separate parts. Most lines that should connect either overshoot, or fall short. Proportions in shapes are, at

best, approximate. Shapes and individual lines are skewed from there intended alignments. All of these aspects must be addressed by sketch recognition.

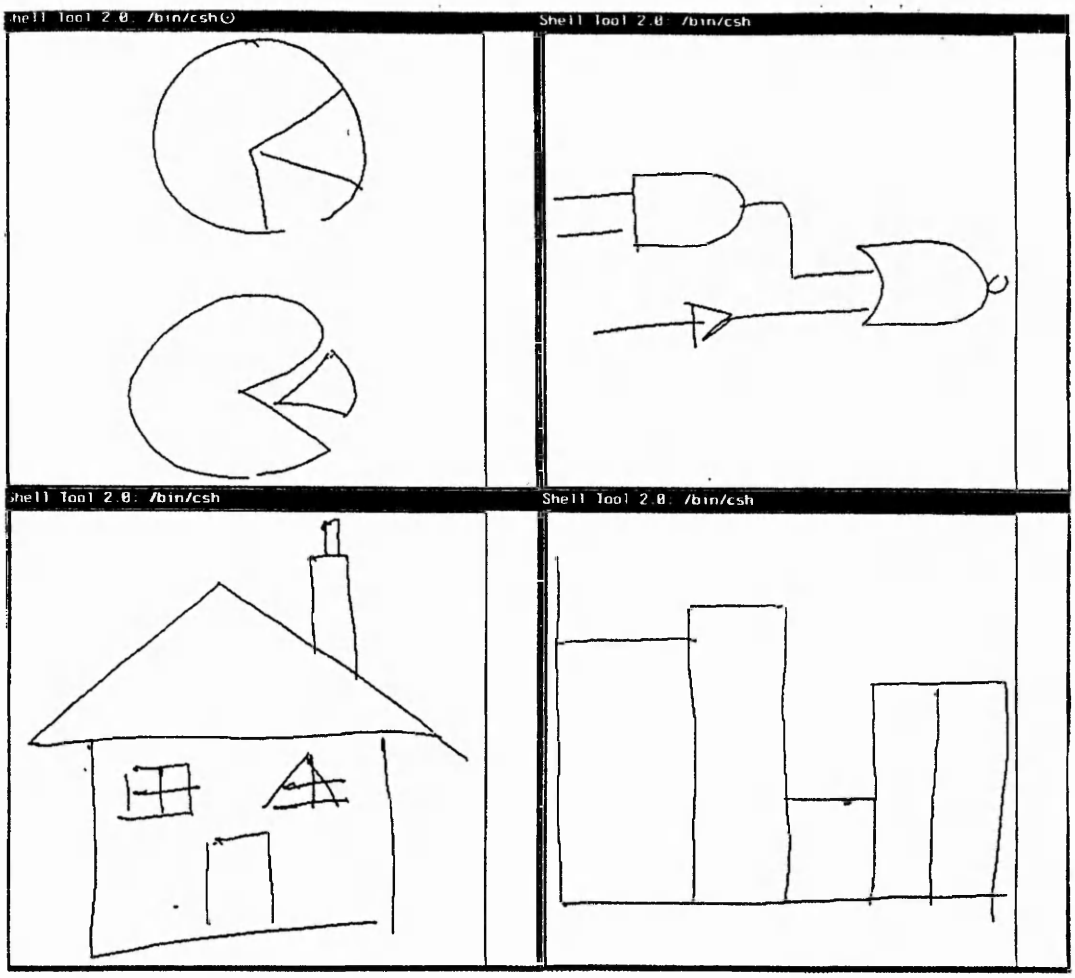


Figure 3-2 Examples of donated sketches

4. TECHNIQUES FOR STROKE SEGMENTATION

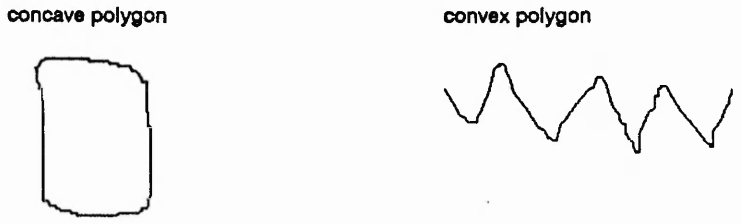
A technique for stroke segmentation is required which is suitable for the real-time dynamic rendition objectives of the system, preferably a technique which will react the moment the pen changes course mid-stroke. Several techniques exist that meet this requirement, we will discuss them briefly to consider their relative merits and suitability.

The HUNCH system analysed changes in pen speed and pressure, as well as angular variation between sampled points, as characteristics of pen movement. Speed and pressure were found to be unreliable indicators of the user's intent and expression.

Angular variation between sampled points gives a detailed measure of pen movement, but it is a local characteristic and does not differentiate major transitions in the pen's path from palsy in the human hand, nor minor deformations in a stroke. Several techniques exist which aggregate angular variation over a section of the pen's path in order to either classify sections into geometric types, or to identify major transitions, such as corners and bend points.

One such technique analyses the sum and range of variations in angles between consecutive pairs of sample points, in order to classify the line element drawn [2].

In a straight line the sum and range is small. In a steady curve the local angles are of similar sizes, so their range is small, and mostly of the same sign, so they accumulate a significant sum. In a polygon the distinguishing feature is the range, since angles are large at corners and small along the sides. The sum will vary according to whether the polygon is concave or convex.



Classification is summarised by the following table. The threshold criteria on range must distinguish between a smooth circle and, say, an octagon.

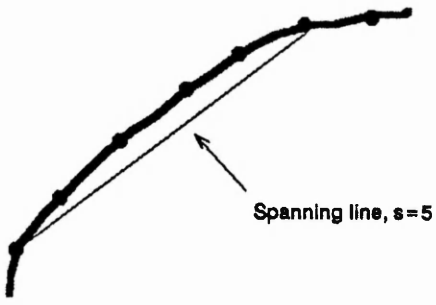
	Sum	Range
Straight line	Small	Small
Curve	Large	Small
Polygon	-	Large

A major disadvantage with this technique is that a geometric element to be recognised has to be drawn with a single pen stroke. The pen must be lifted between each line element in a symbol. A square, for example, cannot be drawn in one stroke of the pen, the user is constrained to lifting the pen between each side of the square. It can be seen from the samples of gate sketches in section B.4 that the natural tendency is to use flowing strokes

rendering several line elements. Assuming that freedom of expression is conducive to the thought process, and concentrating on the subject being sketched, it is desirable that the user is constrained as little as possible in rendering mental images, and is at liberty to sketch in their natural style.

Another technique [16], based on local angular variation, segments a stroke when the pen has travelled a significant distance and has deviated from its present path by a significant angle, producing a polygonal approximation. Further analysis is necessary to extract curves from the polygonal approximation. The technique retains corner points accurately, but intuitively it will likewise retain minor sharp deformations, an inherent weakness in techniques which analyse local angular variation. Transitions from straight to curved regions are reported to be detected late.

The most widely used segmentation techniques in pattern recognition monitor some characteristic over a region of the line pattern, usually in relation to a spanning line, which conceptually joins two points s points apart in the line pattern. The characteristic of interest is either maximised or aggregated over the spanned region of the line and compared with a threshold value. The objective is to classify the region as a whole and thus be tolerant of local deformations and palsy.



Various characteristics and their relative merits have been discussed [12, 15] such as area, perpendicular distance (often called error norm, or error distance), and length of line pattern in relation to span line, as well as angular variation [1]. There are two basic methods of operating a spanning line, known as 'hop-along' and 'scan-along'. With the hop-along method the spanning line spans consecutive hop sized segments. With the scan-along method the spanning line scans the pattern point by point continually reviewing classification.

The effectiveness of these span line methods depends on the segmentation criteria with which they are partnered. Pavlidis proposed a hop-along method, segmenting according to the ratio of line pattern length over span line length [13]. The algorithm starts with s points of the line pattern (the hop size) and evaluates the characterising ratio. If the ratio meets the segmentation criteria then the line is segmented to the hop size, and the next hop sized region is considered. Otherwise if the ratio fails to meet the segmentation criteria, then the spanned region is conceptually split into two, A and B. Region B will form part of the next hop sized region, but region A is re-evaluated and repeatedly split again, if necessary, until the segmentation criteria are satisfied. This technique has been used for fitting polygonal approximations. General disadvantages in the technique are apparent:-

i) the characterising length ratio is incapable of distinguishing a curved region from a polygonal region.

ii) a region may be re-evaluated several times before it is adequately segmented.

iii) frequent merging of collinear segments is necessary.

iv) if the length ratio is large it means that the line has significant curvature, but additional analysis is then necessary to distinguish between a curve and a wavy straight line.

Considering now the scan-along method, the scan-along method has been partnered with an error distance characteristic to segment hand drawings into lines and arcs [8]. However although the kind of drawings considered are simple and regular the technique fragments them to a large degree. Also it is doubtful that the technique would be adequate for the more difficult aspects of sketches such as overtracing [3].

The most attractive segmentation technique, from the point of view of the real-time dynamic objectives of this recognition system, is a scan-along analysis of angular variation as proposed by Freeman and Davis [9]. With this technique the change in angle of the latest pen movement is analysed and the pen path is classified as either moving straight or curved. This allows the pen path to be segmented and replaced with a geometric line type the moment that the pen turns a corner or changes classification. A section of the pen path is only analysed once.

Problems with the scan-along technique are:-

i) corners, and transitions in classification are detected late. By 'late' I mean when the scanning line has advanced a few points further than the corner or transition point.

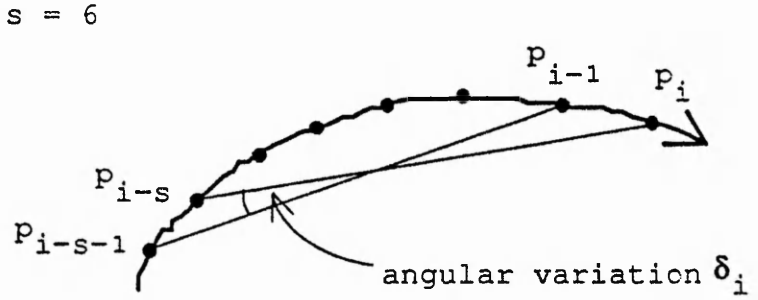
ii) because a spanning line cannot differentiate curve types, a curve which is *intended* to vary in curvature along its length (eg. a spiral) cannot be distinguished without further analysis.

No stroke segmentation technique could be found, that could classify variable curves without secondary analysis. A remedy was envisaged for the problem of late detection, a modification to the scan-along algorithm. Therefore, the scan-along technique has been chosen, for its potential efficiency and classifying power, as the most suitable for stroke segmentation.

4.1 Development of a Technique for Stroke Segmentation

Freeman used the scan-along technique for analysing curvature in chain coded pictures [4, 10], but it is equally applicable to any other data representation that has angular homogeneity. Consider a hand drawn line sampled at consecutive points $p_{i-n} \ldots p_{i-2}\ p_{i-1}\ p_i$, the scan line spans from the latest point p_i to a previous point p_{i-s}, s is the scan length. The scan line makes an angle q_i with some reference axis. When the next point p_{i+1} is received the scan line advances to it, illustrated in figure 4-1, then the angular variation associated with it d_{i+1} is the angular difference between the two scan lines, hence

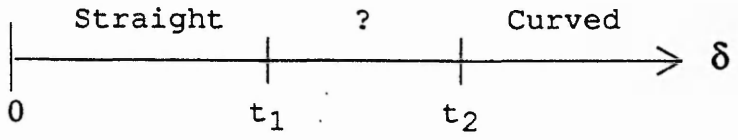
$d_i = abs(q_i - q_{i-1})$ for any point p_i, $i > 0$.



angular variation is the angular difference between the scan lines.

Figure 4-1 Scan-along analysis of angular variation

Angular variation between consecutive span lines provides a smoothed measure of curvature along the pen path, so a segmentation technique based on this will be far more tolerant of local deformations than one based on local angular variation. The segmentation algorithm monitors this smoothed angular variation d at each point to determine whether the pen is describing a straight line or a curve. If d is below a certain threshold t_1 the pen is describing a straight line, above another threshold t_2 it is describing a curve. In the region between t_1 and t_2 classification cannot be made without a further test. The further test is to compare the sum of angular variation accumulated in the stroke segment so far with threshold t_3. A straight line will have a sum below t_3, and for a curve the sum will be above t_3, as illustrated in figure 4-2.




```
if δ < t1 then
     straight
else
if δ < t2 then
     curved
else
if Σδ < t3 then
     straight
else
     curved
```

Figure 4-2 Classification tests

4.1.1 Fitting Geometric Lines

A stroke is segmented when a point is received which causes the classification so far to alter i.e. at a corner or at a transition between straight and curved. The hand drawn stroke segment is then fitted with a geometric line of the appropriate type and this replaces the hand drawn segment in the displayed sketch.

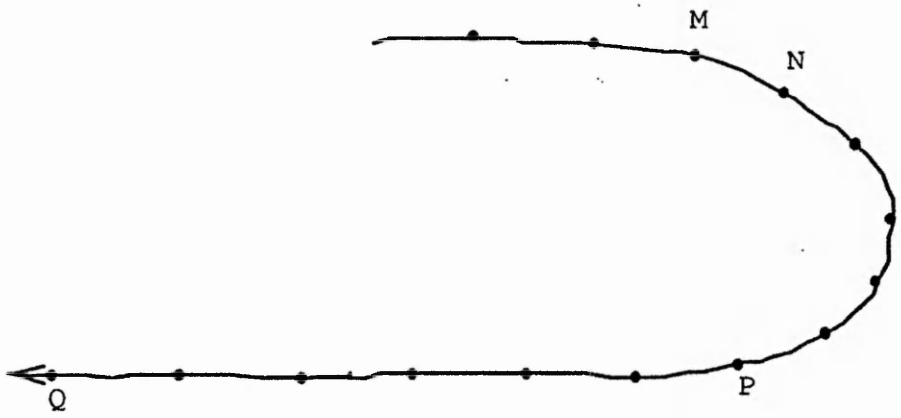
Wrong classification of strokes must be anticipated due to ambiguities in human sketching. However, a measure of confidence can be attained in classifying a stroke according to which interval the angular variation falls into (figure 4-2).

Below t_1 the stroke is certainly straight; above t_2 it is certainly curved; between them there is no certainty, curvature is shallow or undulating. A measure of confidence is accumulated on every iteration of the segmentation algorithm, 100% if angular variation falls below t_1 or above t_2, and 0% between. At the end of the stroke segment the average of the accumulated confidence is attributed to the fitted line. This confidence of low level classification might be employed at higher levels to resolve ambiguities.

4.1.2 Difficulties Encountered in Stroke Segmentation

Different curve types cannot be distinguished at this stage in the recognition process due the ambiguity in their rendition by the human hand. Therefore a piecewise fit of a single curve type is used to approximate a curved stroke. Circular arcs are used since they are commonly occurring in business graphics (in other classes of drawing a different basic type of curve may be more suitable). The correct curve type can only be determined at a higher level of recognition.

There is a weakness in the scan-along method in that a corner or a transition between straight and curved regions is not detected until some point after the bend (see figure 4-3). This weakness has been cited as a reason to use an alternative technique to scan-along [15], but it can very easily be compensated for simply by segmenting at a point previous to the one which showed the bend.



(a) bend point M is not detected until point N.
(b) bend point P is not detected until a scan length later at point Q when the angular variation of the scan line has dropped below the straight threshold.

Figure 4-3 Detection of bend points with the scan-along method

In the case of a corner or a transition from straight to curved, it is only necessary to segment one point previous to the one which showed the bend; however, in the case of a transition from curved to straight it is necessary to segment a whole scan length previous to the one which showed the bend. The unfortunate possibility exists that the stroke may end before the transition to straight is detected, i.e. the pen lifts before point Q in figure 4-3(b), then the curve fitted will be over full and encompass the straight region, although this may subsequently be rectified by shape matching. The thresholds and scan length might be tuned to alleviate this weakness, but the potential of this and its application was not investigated.

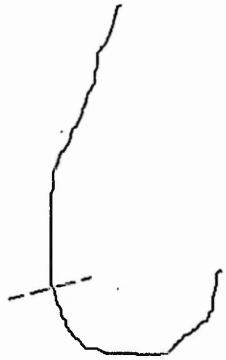


Figure 4-4 Different curves drawn with one pen stroke

A particular weakness is occasionally encountered when different curves are drawn with one pen stroke, as in figure 4-4 where one curve follows on from the other. Because the scan-along technique does not distinguish between regions of varying curvature, it does not recognise the stroke as distinct curves. It amalgamates them into one, resulting in a poor rendition of the stroke. This kind of stroke occurs infrequently (twice in 200 sketch samples), and so is tolerable in an interactive editing environment. It is not optimal, however, and some effort has been spent investigating a possible solution.

The following solution was tried. While a stroke is being classified as curved, the *average* curvature is monitored. When a significant increase is observed in the *rate of change* of average curvature, the stroke is segmented. Average curvature is the cumulative sum of angular variation in the spanning line averaged over the curve so far,

$$a_n = \frac{\sum_{1}^{n} d}{n} \qquad \text{at point n.}$$

(There is no performance penalty in this solution since angular variation is already being accumulated for the straight/curved test).

This solution was not successful, as the technique is much more sensitive to few points (small n early in the stroke) than to many points (large n).

Many people have a habit of 'overtracing' whilst sketching [3]: that is, reinforcing a line by redrawing over it several times without lifting the pen. It can be seen in some of the sketch samples, particularly the house of section B.3 In this kind of stroke the pen goes through a 180° change in direction which can present particular difficulties for a smoothed or aggregated segmentation technique. The scan-along method of analysing angular variation does not cope well with such inflexions, as shown in figure 4-4. When the pen path doubles back on itself the angle of the scan line does not deviate significantly. Not until the leading and trailing ends of the scan line are opposite each other, or have passed, does the scan line deviate sufficiently for the transition to be detected. The transition is detected very late resulting in the inflected stroke being truncated and the rendered line foreshortened.

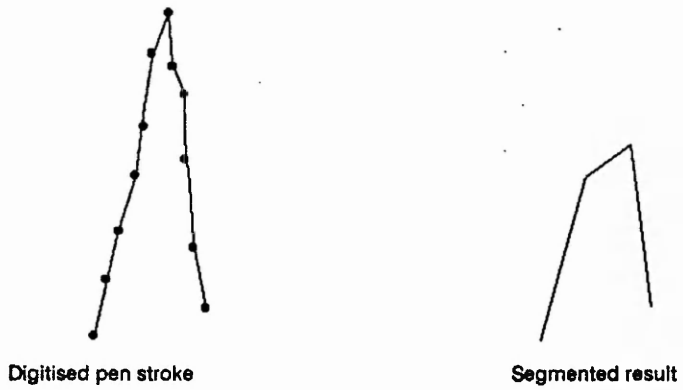


Figure 4-4 Truncation of an inflected stroke

It is necessary to monitor some supplementary characteristic in order to detect a point of inflexion. A characteristic which reveals a point of inflexion immediately is local angular variation, so an additional segmentation criterion prior to those in figure 4-2 is used which detects a large local deviation. Local angular variation is already calculated by the filtering algorithm, so its exploitation here is at no extra cost. In fact analysis of local deviation can assist in detecting corners in the first few millimetres of a pen where a scan line cannot be used effectively.

4.1.3 The Stroke Start-up Period

We shall now consider initialisation of the scan-along algorithm. Observing the first few millimetres of a pen stroke, it cannot be determined reliably whether the stroke is going to develop into a straight line or a curve. Only as the stroke progresses does ambiguity reduce to a level at which classification can be made with confidence. Empirical analysis of sketch samples has shown this period to

be on average about 20mm, see figure 4-5 and some of the sketches of pies in section B.2. Intuitively, this distance is also sufficient for the size of the scan line, since the scan line also needs to encompass a significant amount of curvature. Therefore classification begins when sufficient points have been sampled to comprise a scan line. This period is called the scan length, and covers, nominally, about 20mm of pen stroke.

Although classification is not reliable in this initial period it is still possible that the pen may lift, or a corner or inflexion may occur, thus forcing a classification. The total curvature of the stroke up to that point, enables a reasonable classification to be made. The total curvature is the sum of angular variation. The classification test applicable here, is the same as in the uncertain range of curvature between straight and curved in figure 4-2. That is, if the sum of angular variation is $> t_3$ the stroke segment is a curve, otherwise it is straight.

A weakness remaining in the modified scan-along algorithm has proved to be the initial classification after the stroke start-up period. Although the sum of angular variation determines whether a line starts off as being straight or bends, it will not tell you whether a bend is due to a corner or to a gradual curve, as in figure 4-5.

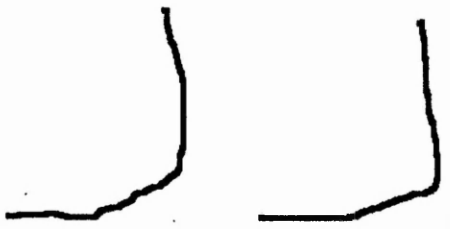


Figure 4-5 Curve vs. corner

Sharp corners are intercepted by the test for inflexions, but shallow corners are mis-classified as curves. This weakness has occasionally shown itself in the results.

A possible solution for this is to inspect the coefficient of variance, a statistical technique for making exactly this kind of distinction. I say 'possible' because it was not tried. It is computationally expensive, and an objective is to see the quality of results that can be achieved with a computationally cheap recognition process. Also, such mis-classifications ought to be recovered by shape matching.

The equation below gives the coefficient of variance of n values of a, the variable under study. In the stroke start-up period, a is the local angular variation. Coefficient of variance, V, is the square root of the result of dividing the variance by the square of the mean,

$$V = (\text{variance} / \text{mean}^2)^{1/2},$$

$$\text{variance} = \frac{\sum a_i^2}{n} - \left(\frac{\sum a_i}{n}\right)^2, \qquad \text{for } i = 1..n,$$

$$\text{mean} = (\sum a_i) / n.$$

Therefore,

$$V = \sqrt{\frac{n \sum a_i^2 - (\sum a_i)^2.}{(\ a_i)^2}}$$

The size of V indicates how much the angles between points vary. A large value of V indicates the presence of a corner. A small value of V indicates a smooth curve (zero V indicates a completely uniform curve).

4.1.4 Stroke Segmentation Algorithm

The complete segmentation algorithm iterates on each sampled point of a stroke segment, changing state as shown in the state transition diagram, figure 4-6. States refer to a sequence of sampled points: 'first of segment' is the first point in a segment; 'start of segment' is any point in the start-up period; 'mid segment' is when the scan-line is operational. The action of a state transition is either to segment the stroke, or nil (no action, in which case the point is in the current segment).

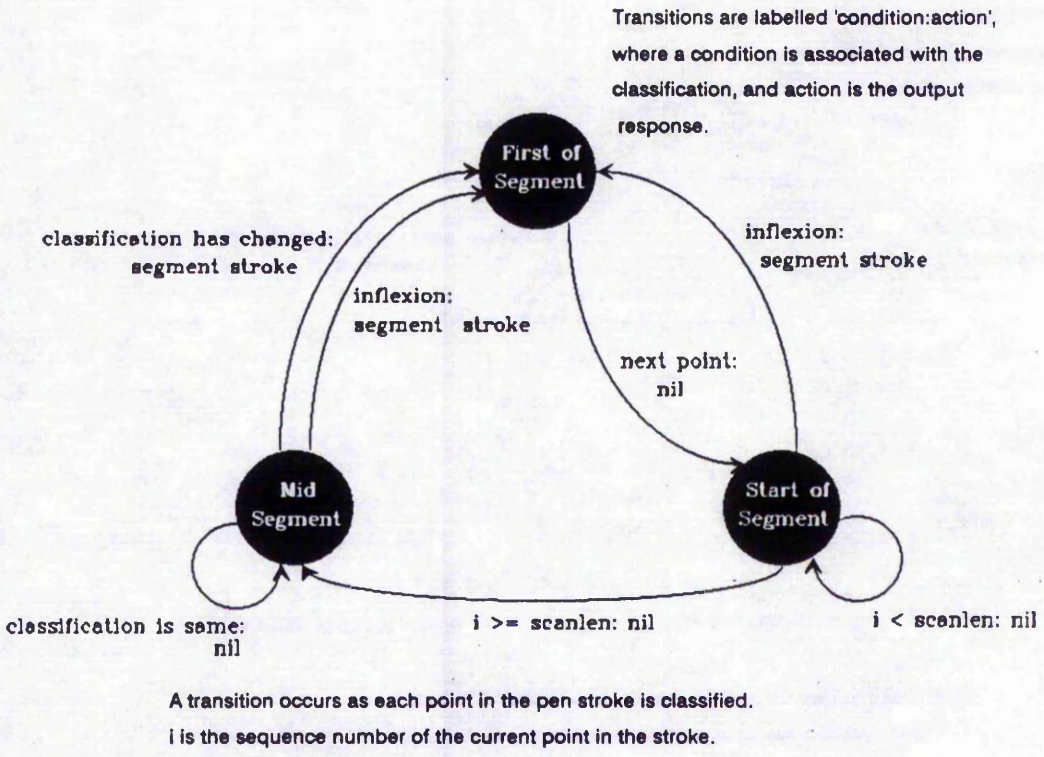


Figure 4-6 Segmentation state transition diagram

4.1.5 Scan Line Length

The length of the scan line is specified as a number of points, the number of points which objectively correspond to a distance of about 20mm, established earlier. Two factors affect the scan length. One is the sampling rate of the tablet, the other is pen speed. Pen speed is an external variable. Sampling rate is a constant, and so the distance between sampled points will vary in direct proportion to pen speed. Likewise, the distance spanned by a scan line of fixed length will also vary according to pen speed.

There are two possibilities for achieving a constant spanned distance of 20mm. One is to vary the length of the scan line, according to pen speed. The other is to use a fixed length scan line, and selectively filter out sampled points, so as to artificially maintain a *virtual* pen speed. The first, varying the scan length, increases complexity of the technique, whereas filtering increases efficiency. Therefore the fixed scan length option with filtering has been adopted.

To establish a suitable target for virtual pen speed a study of pen speed in drawings is necessary. Figures 4-7 to 4-9 show distributions of pen speed in sketch samples. The aggregated distribution of all sketch samples closely resembles a gamma distribution [17], peaking at zero, tailing off above 200 mm/sec. Although this aggregated distribution is mathematically regular individual sketch samples show a wide variation and are mainly irregular. For a given sketch subject different people will draw with widely differing pen speed distributions. Even different sketches drawn consecutively by a single person show a considerable variation in their pen speed distributions.

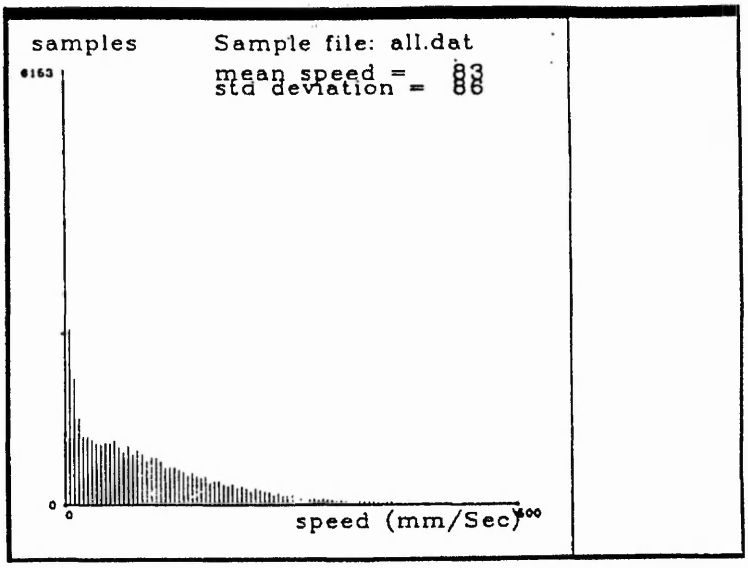


Figure 4-7 Aggregate Distribution

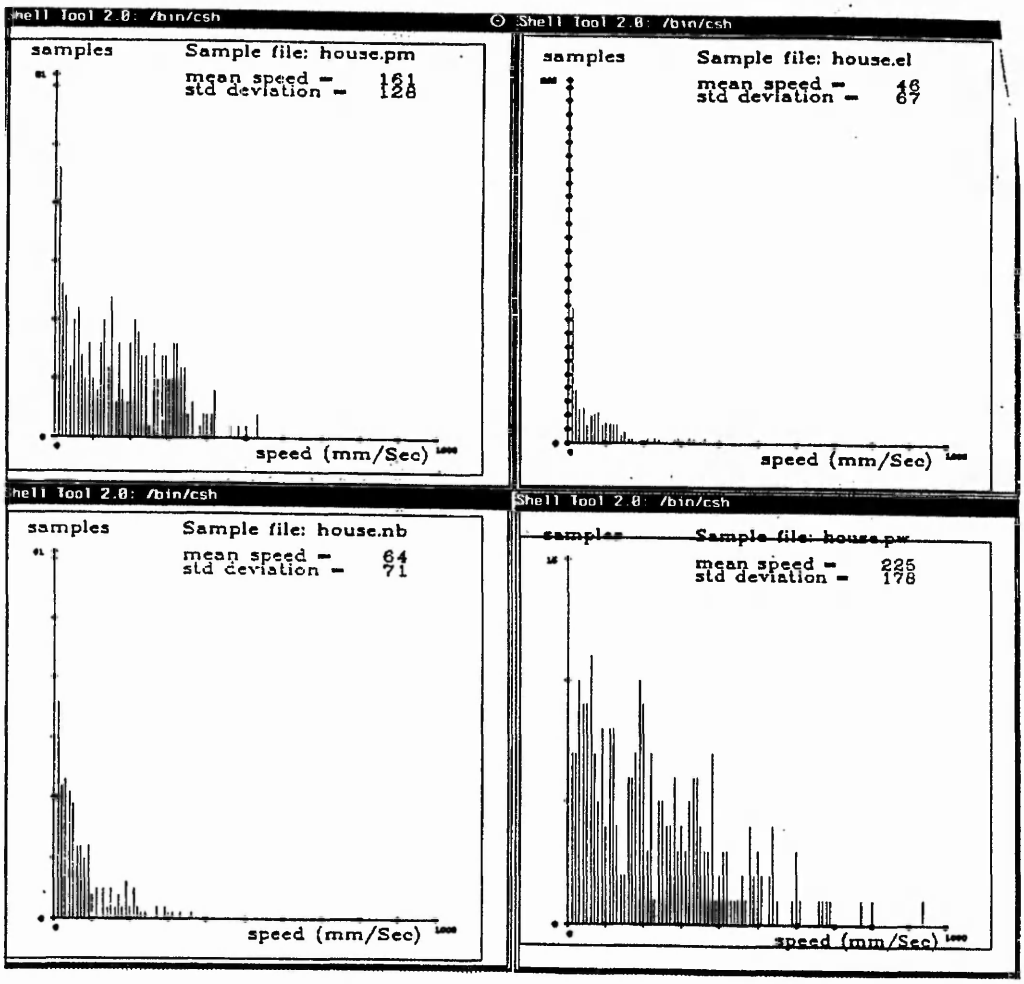


Figure 4-8 Distributions of different people.

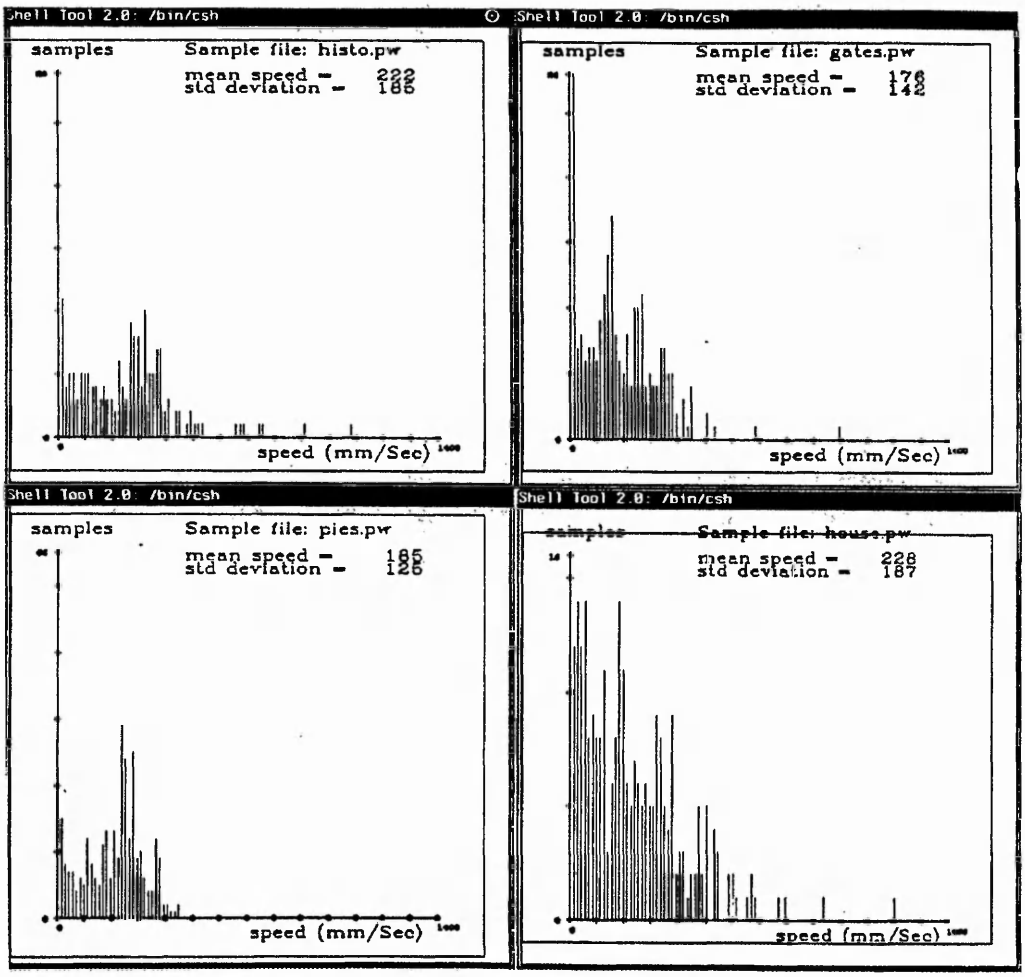


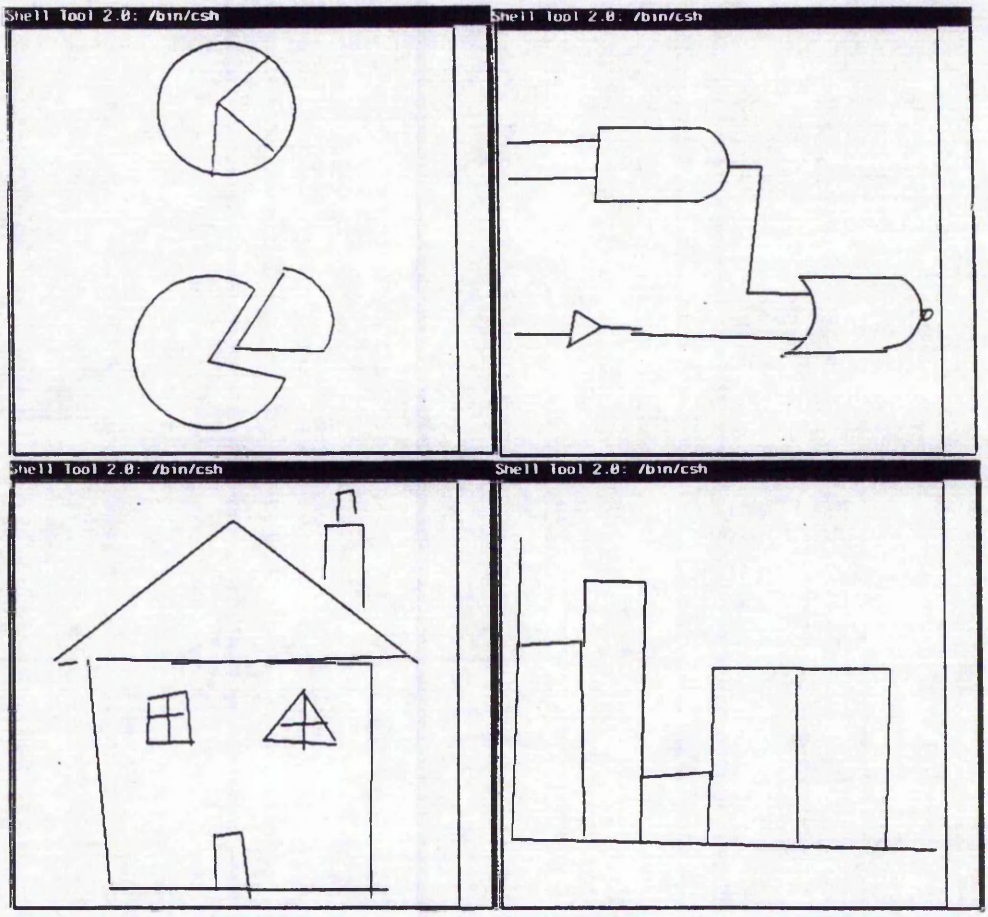
Figure 4-9 Distribution of same person.

The only useful observation that can be made from the pen speed distribution is the fact that filtering out low speed points will increase efficiency. By

experimentation, filtering to achieve a target pen speed of 150 mm/sec was found to give the best segmentation results for the majority of sketch samples. This parameter is a prime candidate for tuning to a particular user, although this was not investigated. The scan length to span 20 mm is calculated as follows:

$$\text{scanlen} = \text{ceil}(20 / \text{pen speed} * \text{sampling rate})$$

where pen speed = 150 mm/sec, which gives a scan length of 10, at sampling rate of 70 samples per second. The function ceil(x) calculates the smallest integer greater than or equal to x. Figure 4-10 shows some results of stroke segmentation, having replaced pen strokes completely with straight lines and circular arcs.



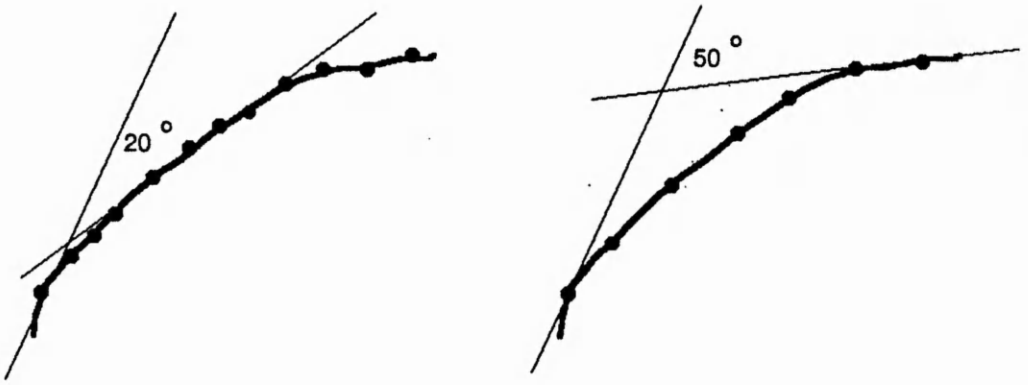
scanlen = 10 (about 20 mm)

Figure 4-10 Results of Stroke Segmentation.

5. FILTERING

The purpose of filtering is to help the segmentation technique work better and faster by removing sampled points that are not needed to classify the line. The type and amount of filtering required depends on the characteristics of the segmentation technique, and can only be determined after the segmentation technique has been chosen. Having selected a scan-along analysis of angular variation for stroke segmentation we can now consider its filtering requirements.

Not all sampled points of a stroke are needed by the scan-along algorithm for it to perform optimal segmentation. When the pen is moving slowly each successive sampled point lies close in proximity to the previous one, the local angular variation between them is small, and a fixed scan line is correspondingly smaller than if the pen was moving fast and the sampled points further apart. As a consequence the angular variation spanned by the scan line is less when the pen is moving slowly than when the pen is moving faster (see figure 5-1). This has the effect that the distinction between a straight line and a curve is less for slower pen speeds. At faster pen speeds the fixed scan-along algorithm is more sensitive to curvature and will recognise shallow curves more readily. Therefore filtering on a proximity basis gives a dual advantage that a) the pen speed is apparently increased with improved recognition of curves, and b) fewer points are passed to the segmentation stage, thus improving efficiency.



(a) slower pen

(b) faster pen

Figure 5-1 Slower pen speed yields closer sampled points, hence a fixed size scan line measures less curvature at slower pen speeds.

Referring back to the distribution of pen speed in figure 4-8, the concentrations of tall spikes at the left end of the distributions represent large proportions of slower pen speeds, so considerable efficiency could be gained from filtering slow pen speeds.

5.1 Filtering Criteria

There are two classes of sampled point which can be filtered out. The first is clustered points, that arise when the pen is stationary or not moving in any particular direction. The other, redundant points, arise by the pen moving slowly. Both classes may convey information about the user's intentions, such as deliberation and accuracy, but such information can be misconstrued by random events such as the user being distracted, and is, according to experience with

HUNCH [6], highly user dependent. In this work I have not attempted to establish reliable psychological metrics, and user independence is a goal. Therefore I discard clustered points and redundant points. A study into the psychological aspects of sketching was not found in the literature, but it might investigate whether clustered and redundant points are symptomatic of such indicators as hesitancy, deliberation, and calculation, that a recognition process might exploit.

Points are considered to be clustered if they lie within 0.5mm of the previous accepted point. Deciding whether a point is redundant is dependent on the segmentation technique in use, and is equivalent to deciding on an optimum pen speed.

Another method of filtering which is also computationally cheap is based on local angular variation between points [1]. This alone is not suitable for use with the scan-along algorithm, since it destabilises the virtual pen speed, and therefore the span of the scan line. Interpolative smoothing techniques have been used [7, 13] but are not computationally cheap. In the past digitising tablets often produced rogue points and the powerful smoothing properties of interpolative techniques were needed to cope with these. Modern tablets do not exhibit this problem and so the cheaper methods of proximity and angular filtering can be used.

Not filtering redundant points was tried and as intuitively predicted this gave poor sensitivity to curvature, resulting in curves being coerced to polygons. Filtering redundant points is done on a proximity basis, discarding a point if it lies close to the previous accepted point. This method has certain advantages:-

(i) it is computationally cheap, requiring only a simple arithmetic comparison on the latest point p_i,

$$\text{if } |p_i - p_{i-1}| <= \text{min\_travel then filter out } p_i$$

(ii) it implicitly filters clustered points as well as redundant points.

The proximity threshold 'min travel' is equivalent to the distance between sampled points when the pen is travelling at the minimum pen speed threshold of 150mm/sec, thus

coord spacing = pen speed/sampling rate
min travel = tablet resolution * coord spacing

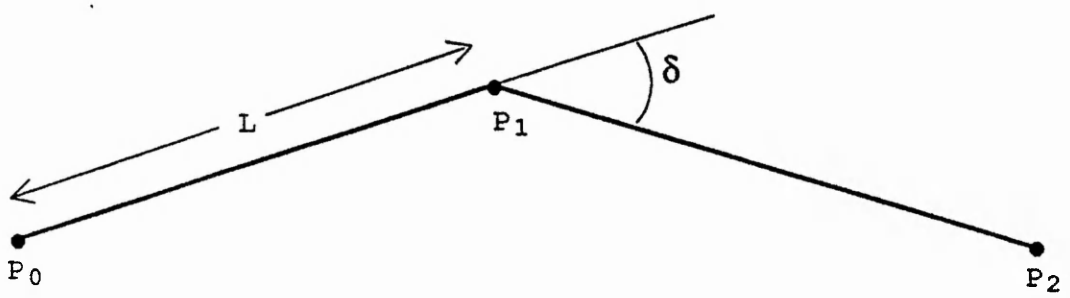
where the tablet resolution is in coordinates per millimetre, and the sampling rate is in coordinates per second, min travel will be in tablet coordinates.

A problem was found with using a proximity criterion alone. Critical points at corners and line end points were discarded resulting in truncated corners and foreshortened lines. This truncation can be as much as double the 'coord spacing'. For example, with the spacing calculated as above at a sampling rate of 70 samples per second the truncation of a corner being drawn at 140mm/s could be as much as 4mm, that is double the distance between sampled points.

The filtering technique adopted by Belaid and Masini [16] is similar and is also susceptible to loss of critical corner points. This is a more complex algorithm in

that is uses a filtering criterion based on local angular variation in conjunction with a proximity criterion. Consider a pen stroke with sampled points P_0, P_1, ... P_n, the filtering algorithm is as follows:

```
a <-- P0; b <-- P1; c <-- P2
for i <-- 2 to n do
        L <-- length( <a,b> )
        d <-- angle( <a,b>, <b,c> )
        if L > min travel and d > local threshold then
                pass b to the segmentation phase
                a <-- b
        else
                filter out b
        end if
        b <-- c
        c <-- Pi+1
end for
```



This is identical to the proximity filtering algorithm except for the addition of the local angle criterion which will filter out points while they lie roughly in a

straight line. This is the de-stabilising effect on pen speed which is characteristic of a local angle criterion and is therefore unsuitable for use with a scan-along segmentation technique. The reason that critical points are lost is because they are filtered out without knowing them to be critical. It cannot be known that a point is critical (such as point P in figure 5-2) until the succeeding point is received (showing the preceding point P to be a corner point). To retain critical points the latest filtered point must be remembered in case the next sample reveals it to be critical.

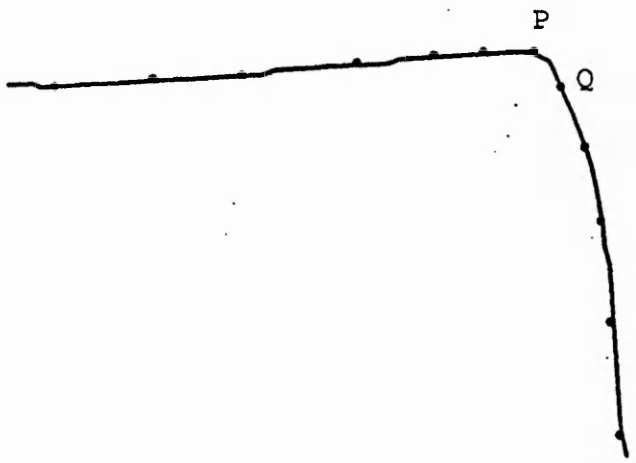


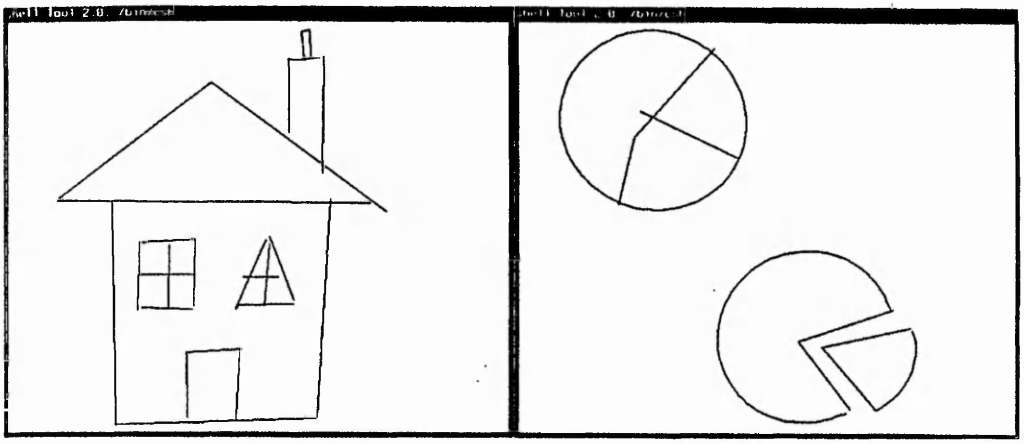
Figure 5-2 It cannot be known that P is critical until Q is received.

I have modified the basic proximity filtering algorithm to retain critical points by a combination of remembering the latest filtered point F and monitoring for large local angular deviations. The algorithm is as follows:

```
a <-- P0; b <-- P1; c <-- P2; F <-- void
for i <-- 2 to n do
        d <-- angle( <a,b>, <b,c> )
        if F not void and d > local threshold then
                pass F to the segmentation phase
                a <-- F
        end if
        L <-- length( <a,b> )
        if L > min travel then
                pass b to the segmentation phase
                a <-- b
                F <-- void
        else /* filter */
                F <-- b
        end if
        a <-- b
        b <-- c
        c <-- Pi+1
end for
```

F is the latest filtered point which is retained if the local angle shows a significant deviation. On smooth corners critical points were found with local deviations as low as 20 degrees, so this is the threshold value used. Such a low value has not been found to significantly reduce filtering frequency, while it ensures retention of all critical points in our library of sketch samples. P is always the previous sampled point, whether it was filtered out or not, so the local angle d is a measure of the true path of the pen. A consequence of

remembering filtered points for retention is that clustered points may also be retained. This is not desirable, therefore an extra test is needed to distinguish and discard clustered points. Figure 5-3 shows sketch samples with pen speed distributions and filtering statistics when the filtering algorithm is applied. Even in these apparently neat and clean sketches only a small proportion of points are needed for the segmentation phase: for the house 56%, plus a further 11% that were filtered but were retained because they were found to be critical turning points; for the pie only 28% (+ 2% retained clustered) were needed for segmentation. The average pen speed after filtering is always higher than the minimum of 150mm/sec, and thus is more suitable to the fixed span scan-along technique adopted for stroke segmentation.



Total points 342:-	Total points 647:-
21% clustered	28% clustered
12% redundant	42% redundant
11% retained	2% retained
56% passed	28% passed
100%	100%
Average speed 223mm/s	Average speed 202mm/s

Sketch samples before filtering, and statistics from the filtering phase, showing proportions of the total points:-

clustered	- discarded due to clustering
redundant	- fell below the minimum pen speed of 150 mm/sec
retained	- clustered or redundant but were found to be critical
passed	- not clustered nor redundant.

Also shown is the average apparent pen speed after filtering.

Figure 5-3 Results of Filtering

6. CURVE FITTING

A stroke segment, having been classified, is replaced with a geometric line. Hand drawn curves of different types are completely ambiguous when seen in isolation out of the context of the shape or subject which they constitute. For example a circular arc is indistinguishable from an elliptical or a sinusoidal arc, as can be seen in the gates and pies sketch samples of sections B.2 and B.4. Therefore it is convenient at this stage to fit every curved segment with a single geometric type. The most convenient type to fit is the circular arc since it is intuitively the most commonly occurring curve type in business documents, and it is easy to calculate, is often supported by graphics display hardware. The correct curve type can be fitted later when the drawing has progressed, by a contextually guided phase of recognition such as Shape Matching or Hierarchical Analysis. A close approximation to the hand drawn curve could be fitted such as a B-spline [3, 11, 14] but this is undesirable because a hand drawn curve is rarely an accurate rendition of what is intended, in the class of drawing considered.

A good fit of a circular arc is achieved by fitting it to three points on the stroke segment, the start, mid, and end points. As shown on figure 6-1 the centre is the point of intersection of two lines that perpendicularly bisect the chords SM ME. The radius of the hand drawn arc is estimated as half the linear distance between the start point S and the mid point M.

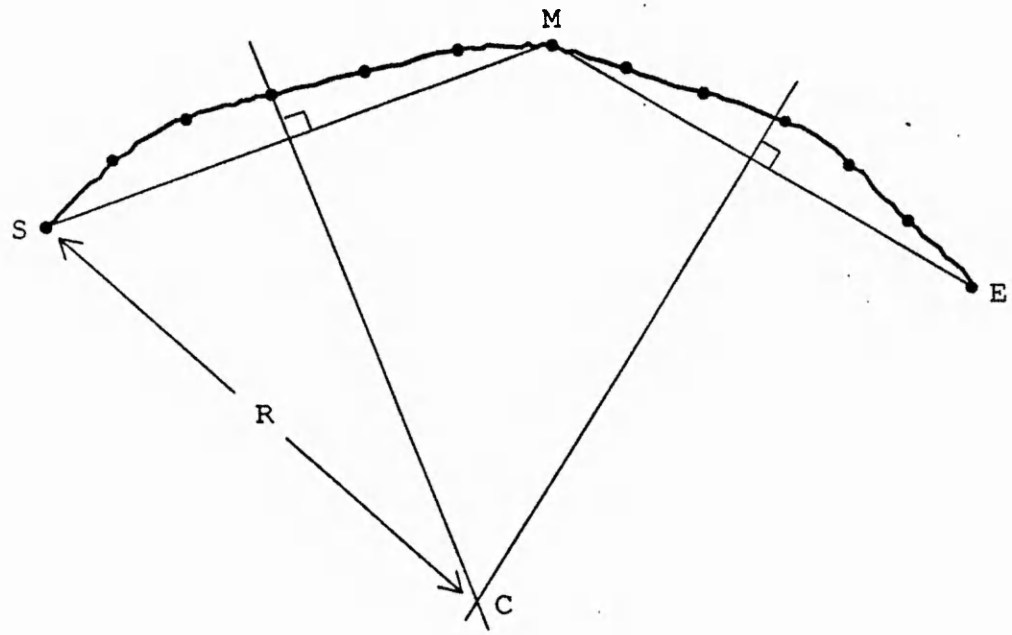


Figure 6-1 Fitting a circular arc

A special case of the circular arc is when it is intended to be closed, i.e. a circle. A circle is drawn as a curve, often highly elliptical, the end points of which rarely coincide but usually lie close. Some examples can be seen in figure 6-3(a). Occasionally an intended arc will be indistinguishable from a potential circle, and this is an ambiguity which can only be resolved by a higher level of recognition. However, in the absence of a higher level of recognition (which is likely in a cheap product) a means of recognising circles is needed, in which case a classification can be made based on the proximity of the arc's endpoint proximity relative to the radius. Figure 6-2 shows the proximity region (centred on the start point) in which the end point must lie for the arc to be recognised as a circle.

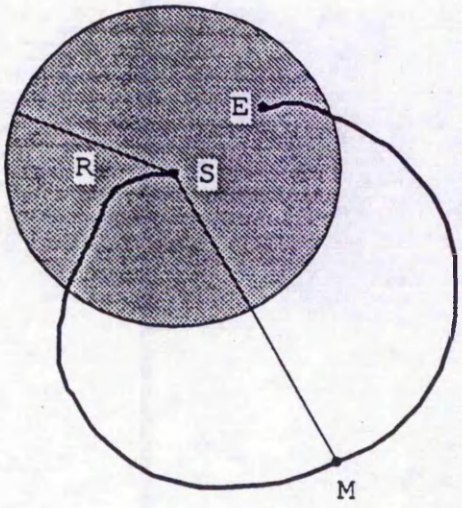
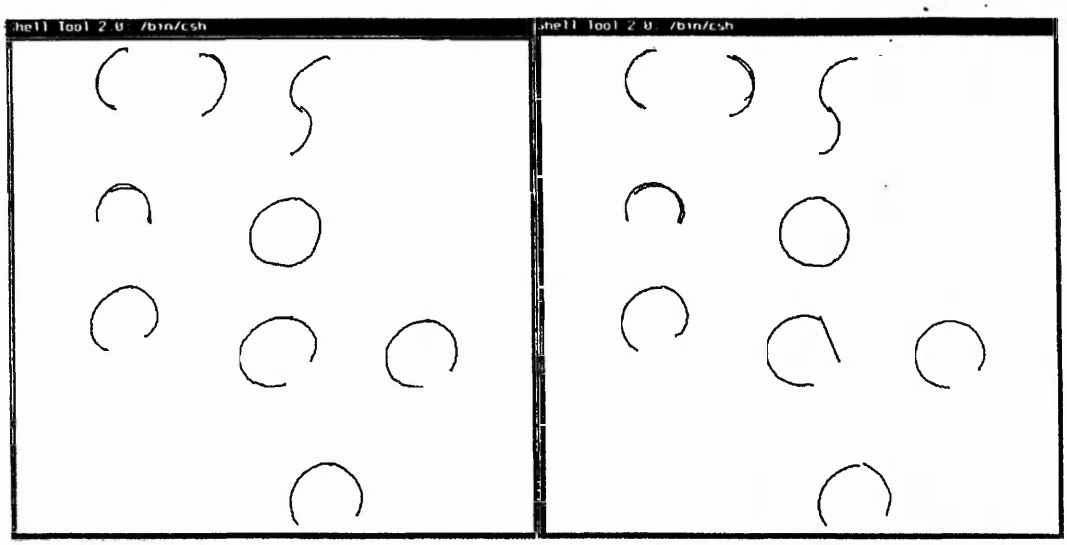


Figure 6-2 Proximity region for detecting arc closure

Best results were found with the radius of the proximity region set to the radius of the hand drawn arc. Some examples are shown in figure 6-3(b). The curve which has been mis-classified as straight illustrates the weakness of the segmentation algorithm when dealing with short strokes.



(a) hand drawn (b) fitted

The curve which has been mis-classified as straight illustrates the weakness of the segmentation algorithm when dealing with short strokes.

Figure 6-3 Recognition of circles.

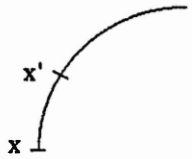
6.1 A Technique for Arc Display

Most graphics display devices do not have curve drawing facilities, curves have to be generated by the driving software. An efficient technique for scan-converting circular arcs has been developed in this project, based on the algorithm suggested by Newman and Sproull [18]. With the arc normalised with its centre at the origin, generation begins at the start point and iteratively calculates successive points on the circle until the end point is reached. A line

joining successive points is displayed at each iteration so as to approximate the curve by many facets. For a given point (x, y) on the circle the next point (x', y') is calculated by the iteration

$$x' = x + ey$$

$$y' = y - ex'$$



e is calculated so that facets are small enough to give the illusion of continuous curvature. Newman and Sproull suggest a value of $e = 2^{-n}$, where n is the number of binary digits in the circle's radius R, that is

$$2^{n-1} <= R < 2^{n}.$$

This value for e was found to generate far more facets than is necessary. A better value is $e = 2^{-n/4}$ which gives sufficient resolution, and results in faster display.

The algorithm must terminate when the circle generation reaches the end point. Termination tests are complicated by the fact that the curve generated by the algorithm is actually a spiral, therefore the end point of the circle is will not lie exactly on the curve. An effective test is to detect when the circle generation gets close to the end point. 'Close' suggests a region around the end point which must be large enough such that the iteration does not overshoot and small

enough so that circle generation does not terminate too far from the end point. The size, r, of the region corresponds to the maximum size of facets d_i, illustrated in figure 6-4.

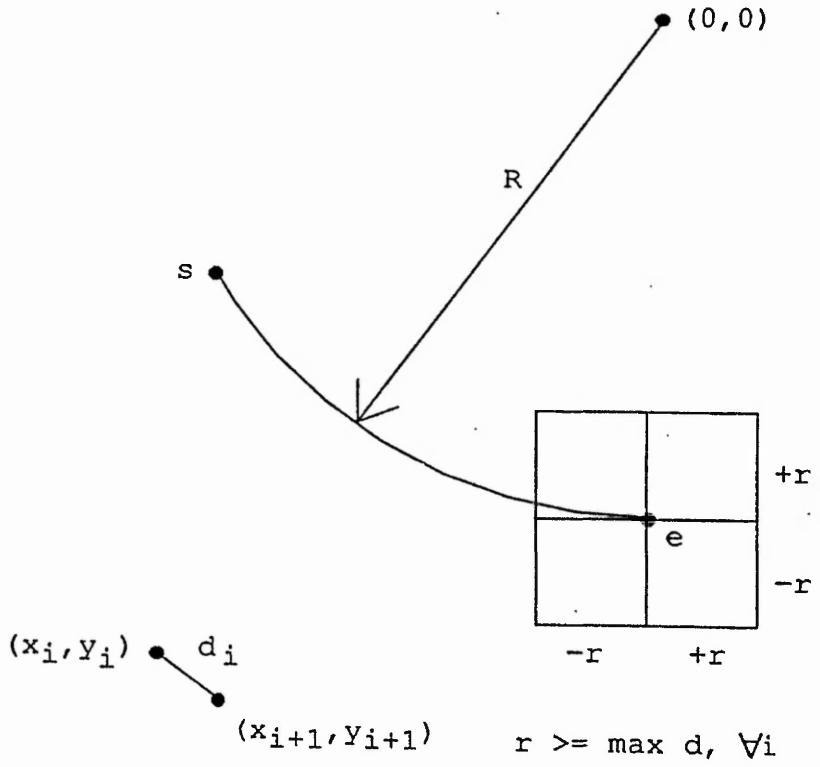
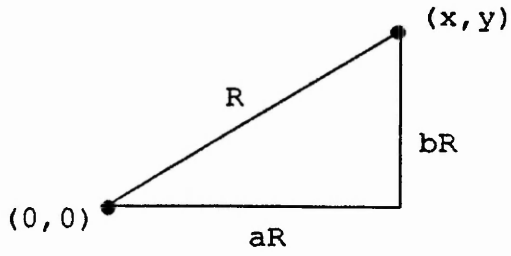
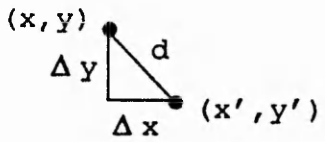


Figure 6-4 Region around an arc end point for terminating circle generation

In order to choose a suitable value for the size of the region r we need to know more about the facets d_i. In order to predict their maximum we now devise a general formula for the facet d.





$\Delta x = x' - x$
$\Delta y = y - y'$

$0 <= |a| <= 1$
$0 <= |b| <= 1$
$a^2 + b^2 = 1$

$x' = x + ey$

$y' = y - ex'$

In terms of a and b:

$x' = aR + ebR$

$y' = bR - e(aR + ebR)$

$= bR - eaR - e^2bR$

$Dx = x' - x = ebR$

$Dy = y - y' = eaR + e^2bR$

$d^2 = Dx^2 + Dy^2$

$= (ebR)^2 + (eaR + e^2bR)^2$

$= e^2R^2(1+e^2b^2+2eab)$

therefore,

$$d = eR(1+e^2b^2+2eab)^{1/2}$$

A suitable size for the termination region corresponds to the maximum sized facet. The range of d is $eR <= d <= eR(1+0.5e^2+e)^{1/2}$, corresponding to $a=1$ $b=0$, and $a=b=2^{-1/2}$ respectively. Since e is a small fraction the termination region can be approximated by

$$r = eR(1+e)^{1/2}.$$

Given a complete circle to generate, the algorithm was found to terminate after the first facet. This is because the first facet is generated inside the termination region, and does not escape before the termination test is applied. The explanation follows: generation begins at the start point; the termination region surrounds the end point; start and end points of a circle coincide, so the termination region encompasses the start point and thus the first facet generated. To avoid this, sufficient facets for the circle generation to get out of the termination region are initially needed before termination tests can be applied. The minimum facet is $d_{min} = eR$, therefore two iterations ensure escape from the termination region since $2d_{min} > r$.

The results of this circle generating algorithm can be seen in figure 6-3 and was used to display all the recognised curves in appendix B. The slight spiral that the algorithm generates is not noticeable.

7. STROKE ANALYSIS

7.1 Merging

A common habit in sketching is for a person to draw a line in several different parts. This can be seen occasionally in the sketch samples, particularly in sections B.1 and B.3, Histos and Houses. The parts can be drawn in any order and are sometimes not even drawn sequentially, or maybe they are added later as an afterthought. There is no consistency as to where such parts start and stop in relation to each other, there may be a gap between endpoints or they may overlap considerably. Overtraced lines overlap completely. It is desirable to merge such lines at some stage so that they become the single line that they are intended to represent. It is tempting to expect this merging to be accomplished implicitly by higher levels of recognition, but this would place considerable demands on higher level classifiers, increasing the likelihood of misclassification. It can be seen in the results that merging can be applied reliably at this low level, in the absence of contextual guidance, to drawings that do not contain close parallel lines. Close parallel lines are indistinguishable from overtracing, and merging will remove them.

Merge tests are applied to the latest fitted stroke segment to see if it can be merged into a previous line of the drawing. If a merge is executed then the modification may create another merge opportunity, so the process is recursive until all merges have finished. Only lines of the same type can be merged. Different merge criteria are applied depending on the line type. For two candidates of the same type the criteria are as follows:

straight lines,

i) they must be parallel (or anti-parallel)

ii) they must be close in proximity at some point

circular arcs,

i) they must be co-central

ii) they must be co-radial (similar radii)

iii) they must be close in proximity at some point

Techniques for detecting proximity have been chosen which make a compromise between effectiveness and computational cheapness. Techniques exist which will detect all possible overlap conditions, but are computationally expensive (such as calculating the linear distance between one candidate's endpoint and each interpolated point on the other line). Therefore I considered it sufficient to use cheaper techniques which will detect all but the more unlikely overlap conditions. Suitable techniques have been developed and are now described.

Straight lines are deemed to be parallel if their normalised slopes differ by no more than 10 degrees. They are deemed to be proximate if the longer line intersects a conceptual boundary surrounding the shorter line. This is illustrated in figure 7-1. The conceptual boundary, called the proximity box, lies a distance d from the shorter line, where d is a fraction of the average of the lines' summed lengths,

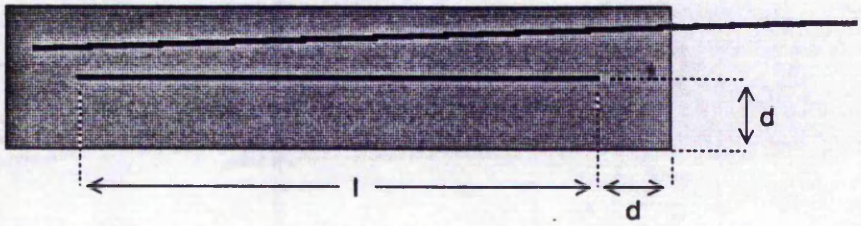
$$d = e(L+l)/2$$

where

$e < 1$ is the proximity coefficient

L is the length of the longer line

l is the length of the shorter line.



Straight lines are deemed to be proximate if the longer line intersects either the proximity box around the shorter line or the shorter line itself.

Figure 7-1 Proximity test on a straight line

Adequate merge detection in all sketch samples has been achieved with $e = 1/15$. The coordinates of the proximity box are derived by similar triangle principles from endpoint coordinates of the shorter line, depicted in figure 7-2. The hypotenuse of the small triangle is,

$$d = cl$$

By substitution c is derived from e,

$$c = \frac{e(L + l)}{2l}$$

By similar triangles,

$$a = cy$$

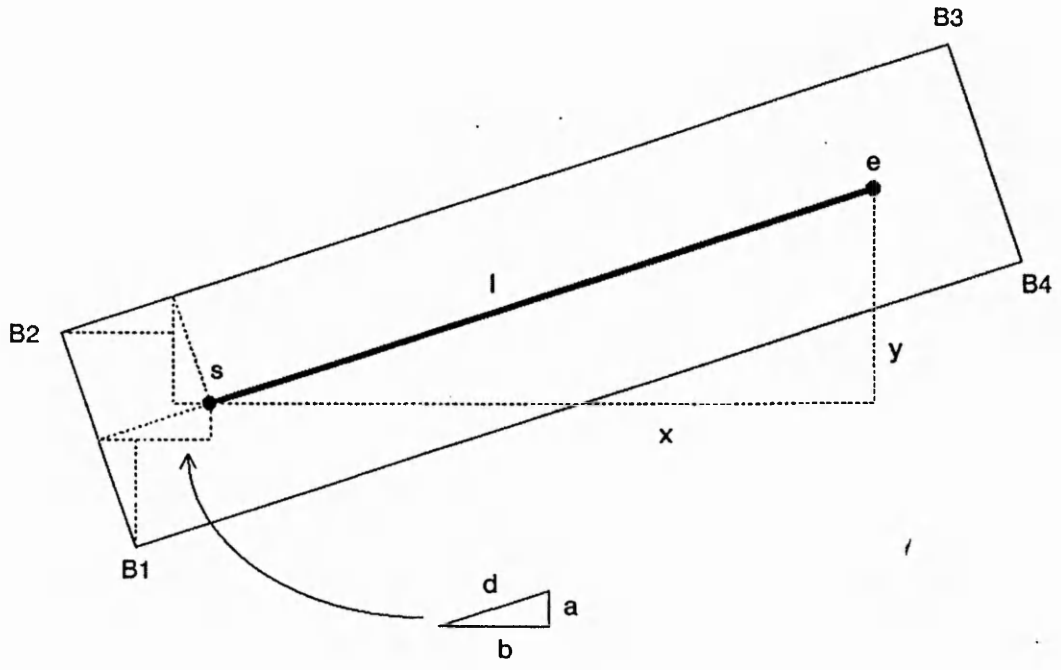
$$b = cx$$

x and y are the cartesian components of l, the line length. s.x and e.x denote the x-coordinates of the start and end points of the line.

$$x = e.x - s.x$$

$$y = e.y - s.y$$

This set of equations holds for any orientation of the line, and provide a general calculation for the coordinates of the proximity box.



B1.x = s.x - (b-a)
B1.y = s.y - (b+a)

similarly,
B3.x = e.x + (b-a)
B3.y = e.y + (b+a)

B2.x = s.x - (b+a)
B2.y = s.y + (b-a)

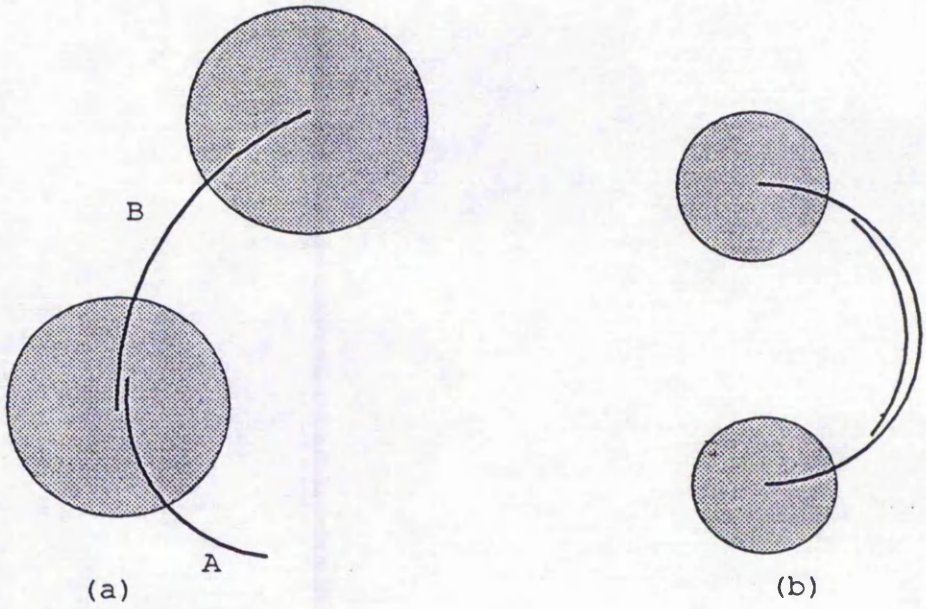
similarly,
B4.x = e.x + (b+a)
B4.y = e.y - (b-a)

Figure 7-2 Calculation of proximity box coordinates

Hand-drawn curves have been segmented into circular arcs. In the absence of higher level recognition merging can be used to tidy them. For two arcs to be merged they must be proximate, co-central, and co-radial. Circular arc segments are deemed to be co-central if the distance between their centres is less than half the larger radius, and they are deemed to be co-radial if the difference between their radii is also less than half the larger radius. These

proportions have been established by experiment, but could be adjusted for different types of drawing or user style.

Proximity tests are only applied to the endpoints of the arcs since these are the only known approximation points on the circumference. Proximity tests are applied using exactly the same technique as for detecting arc closure (see section 4). The arcs are deemed to be proximate, as in figure 7-3(a), if an endpoint of the smaller radius arc A lies inside either proximity region around each end of the larger radius arc B. Note that an arc such as that in figure 7-3(b) which is proximate but does not lie within a proximity region at its endpoints will not be merged. Such occurrences may be amended by hand.



(a) shows proximity regions around endpoints of larger radius arc B.

(b) although the small arc is proximate neither of its endpoints lie within a proximity region and therefore will not be detected as proximate.

Figure 7-3 Proximity detection for arcs

Assuming that two arcs have passed the merge detection tests we will now consider how they will be merged to create a new arc. The combination of phase and endpoint types determine whether the merge will result in a more convex arc, or is in response to overtracing. For two arcs there are eight possible combinations of phase and endpoint type pairs shown in figure 7-4.

One set of four result in a joined more convex arc, the other set of four constitute overtracing.

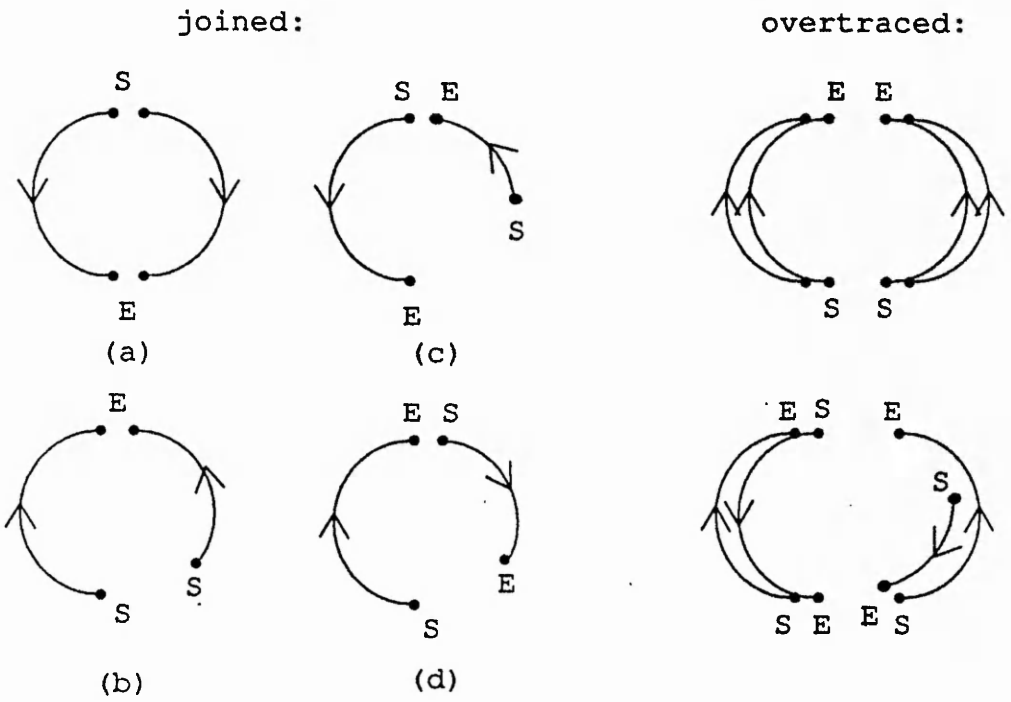
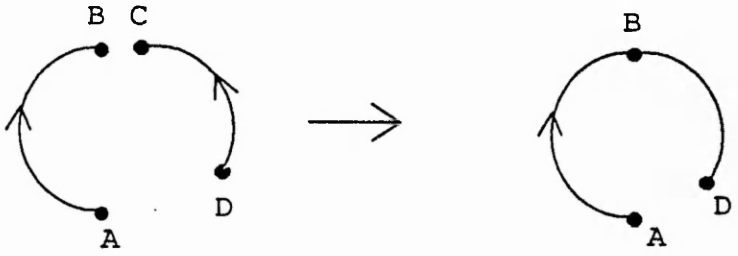


Figure 7-4 Arc merges, defined by two sets of possible combinations of phase and endpoint types.

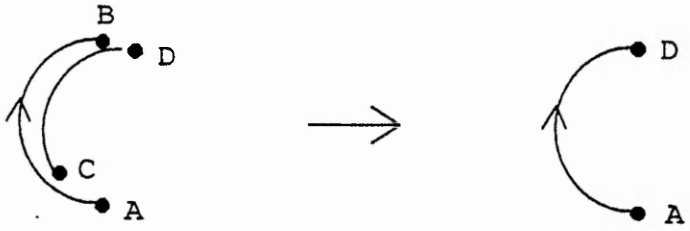
The two sets can be distinguished by the fact that for the joined set when the phase of the two arcs is the same the endpoint pairs will be different in type (c) and (d), and when the phases are opposite the endpoint pairs will be of the same type (a) and (b). For the overtraced set the reverse is true.

Referring to figure 7-5(a), in creating the new arc in the joined set one of the proximate endpoints B becomes the mid point. The other endpoint pair A, D

become the endpoints of the new arc. The phase of one of the merged arcs is retained and dictates which is the start point and end point A or D of the new arc. Referring now to figure 7-5(b), in creating the new arc for the overtraced set, the mid point and phase of one of the arcs is adopted, and also endpoints which are the furthest linear distance from the mid point. Finally, the new arc is tested for closure in case the merged arcs are intended to form a circle. None of the sketch samples had arcs that needed merging, but some users' styles may need it. The results of merging can be seen in appendix B.



(a) merging in the joined set.



(b) merging in the overtraced set.

Figure 7-5 Creating a merged arc

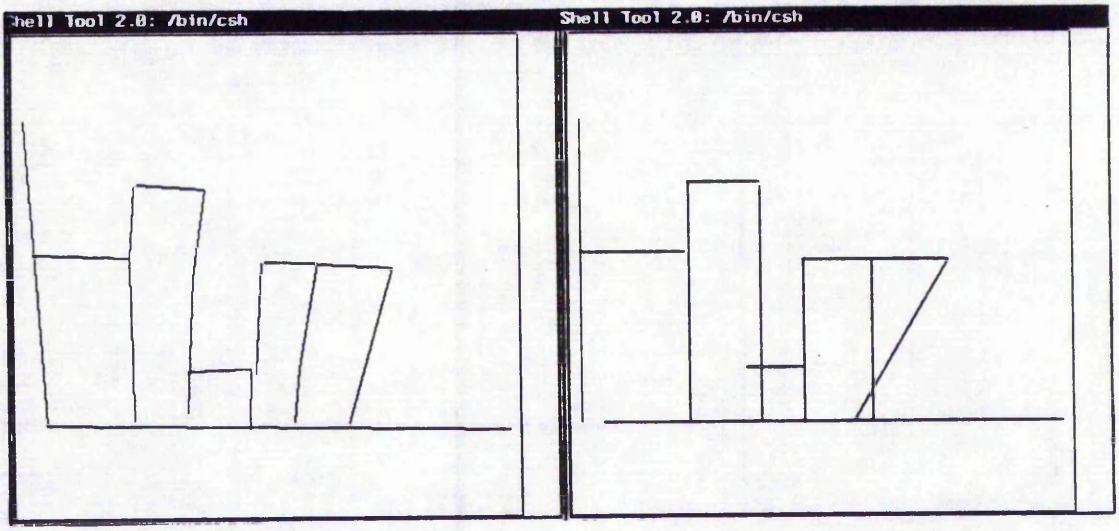
7.1.1 When To Perform Stroke Analysis

We will now consider when is the appropriate time in the recognition process to perform merging and other stroke analysis. Even the cheap merge tests described will take a significant amount of time to process when applied across the whole drawing, so consideration must be given to maintaining interactive response. While the user is drawing, response to pen movement needs to be instantaneous, but while the pen is lifted all processing power can be devoted to non real-time recognition activities. In a single processor machine highest priority must be given to monitoring pen movement, second highest priority to the initial rendition of the sketch, that is Stroke Segmentation, all other recognition activities are of less urgency. The experimental implementation, however, devised a scheduling strategy which allows for most merges to be performed immediately and cheaply. The majority of merges are the cheapest to detect because they arise in successive strokes. That is, the latest stroke either overtraces the immediately preceding one, or is an extension. These are the cheapest to detect because only one merge test is necessary, compared to searching through the whole drawing repeating the merge tests, as is necessary for non-successive strokes. Therefore while the pen is down merging is done on the latest pair of lines, and when the pen lifts the rest of the drawing is searched.

7.2 Alignment of Lines and Shapes

Due to human inaccuracy in sketching, lines that are intended to be horizontal or vertical (which is the vast majority of straight lines) are almost always skewed, they can never be assumed to lie exactly on the axis. Diagonal lines also commonly occur, at 45°, and in isometric views 30° and 60°.

Experiments have shown that alignment before Shape Matching can destroy shape information, as seen in the sketch sample of figure 7-6. Alignment of a recognised shape is therefore more desirable than aligning its component lines. The action of shape matching will implicitly perform alignment.



(a) segmented (b) aligned to isometric norms

Figure 7-6 Alignment can destroy shape information.

The same principle applies to elementary shapes which constitute a more complex shape, that alignment should be applied to the complex shape rather than to its constituents.

Alignment is essentially, therefore, a higher level function appropriate after all other phases of recognition have been applied. But unless the recognition

process is all encompassing, and leaves no lines unaccounted for, there will be a need to align individual lines. Alignment of individual lines is described here.

7.2.1 Aligning Individual lines

For general purpose alignment that is suitable to business documents, the isometric norms 0, 30, 45, 60 and 90 degrees were chosen.

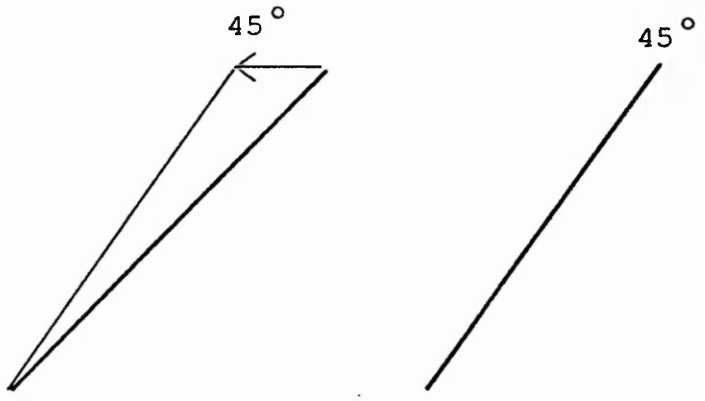
The operation of alignment is ideally a rotation about some pivot point. Considering straight lines first, the general method of rotating a straight line is first to normalise it so that the pivot point lies at the origin, then the other endpoint can be rotated by the following transformation,

$$x' = xCos(q) + ySin(q)$$
$$y' = yCos(q) - xSin(q)$$

equation 7-1.

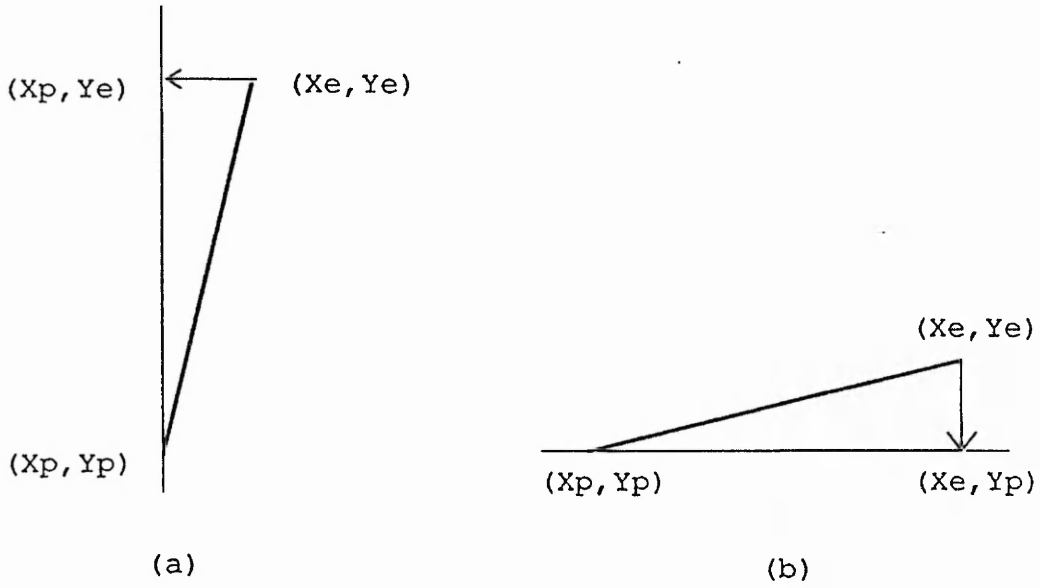
However, since alignment is performed frequently a computationally cheaper method which does not use trigonometric functions is desirable. An alternative method has been devised here which takes advantage of small angles of rotation. With alignment norms of 0, 30, 45, 60, 90 degrees the maximum angle of rotation possible is 15º. So rather than rotate an endpoint it can be projected onto the nearest alignment norm (figure 7-7). To do this the line is first normalised with the pivotal endpoint at the origin. To project the other endpoint, one coordinate of its (x,y) pair is retained, x say, while y is transformed to the alignment norm, using the general line formula y = xm. Which coordinate is retained depends on whether the line is closest to horizontal or vertical. If it is closest to vertical, y is retained, otherwise x. The

maximum shortening or lengthening effect is 12% , incurred by alignment to 45°. To 30 and 60° the effect is 10%, and to 0 and 90° it is 3.4%.



(a) projection onto 45° alignment norm.
(b) after alignment - 12% shorter.

Figure 7-7 Alignment by projection



(a) adopt x coordinate of pivot point.
(b) adopt y coordinate of pivot point.

Figure 7-8 Alignment to horizontal and vertical by projection

In the transformation formula y = xm, m is the gradient of the alignment norm from a set of constants (one for each norm). Negation then translates the constant into the required quadrant.

A considerable efficiency is gained from the projection method when alignment is to horizontal or vertical norms. This is the case for the vast majority of straight lines, when m would assume one of the values 0, 1 or -1. Here, applying the line formula is unnecessary. In fact normalising the line to the origin is also unnecessary. All that is needed is to adopt one coordinate from the pivot point,

and assign it to the other endpoint. For horizontal alignment adopt the y coordinate; for vertical alignment adopt the x coordinate, as shown in figure 7-8.

Alignment in these cases is optimally efficient.

Considering the problem of choosing a point on the line to use as a pivot, we should use the point with the largest number of connections in order to minimise the number of broken connections. The point with highest connectivity could lie anywhere on the line. To find it proximity tests would have to be applied at interpolated points all the way along the line, repeatedly scanning the drawing for connections. This would be an expensive process. I avoided interpolation by using a simple approach of only considering the two endpoints for the pivot, since most connections are at the endpoints, trading-off computation against the possibility of sub-optimal breakages. Examples of aligning by these methods are shown in section B.5 Results of aligning. Note the broken connections and, in some samples, the loss of shape.

We will now consider the method developed for alignment of circular arcs. For the sake of simplicity in the narrative we will talk in terms of using the start point for the pivot. An arc is aligned by:

1) aligning to a norm the chord which conceptually joins the arc's endpoints,
2) then rotating the arc's midpoint and end point to meet the new chord.

The chord is aligned to one of the alignment norms in exactly the same way as a straight line, i.e. by projection.

The arc itself may not be projected since there is no predefined gradient to project the mid point onto. Therefore the arc's mid point and end point are rotated by the general rotation formula of equation 7-1. The angle of rotation is the difference between the chord's original slope and the alignment norm. The centre of the arc is then re-calculated.

This experiment in aligning arcs was found to bring no improvement to the subjective rendition of the sketch, and in many instances made it worse. It showed that simply rotating a fitted arc is insufficient. What is really needed is to re-model the arc to particular characteristics determined by the object of which it is a part. This function is implicitly performed by Shape Matching and Hierarchical Analysis.

8. CONCLUSIONS

Techniques for use in interactive recognition of sketches by computer have been investigated and developed for use either in programs requiring simple hand-input of business graphics, or as preprocessing for higher level recognisers. Stroke segmentation, filtering, and merging are generally applicable, rendering a digitised sketch into a structure of geometric lines much more suitable for higher level recognition, or which can be subjected to latching and aligning on an individual line basis.

Modifications were developed to overcome weaknesses in the technique chosen for stroke segmentation, and were successful, but given a short pen stroke any segmentation technique without contextual guidance will be unreliable due to the inaccuracy of human sketching. Likewise, curves of different types are completely indistinguishable.

A technique was originated for detecting lines that require merging, that is lines that are drawn in separate parts intended to be one line. This was found to be successful for classes of drawing that do not contain close parallel lines, which might be mistaken for overtracing or multi-part lines, and its success is affected by the user's style. Latching and aligning are also subject to similar considerations. The decisions of whether and when to apply merging, latching, and aligning, and settings of the associated threshold values, might benefit from being under the user's control, since they are all sensitive to the type of drawing involved and the user's style, but this remains to be investigated.

A study into the psychological aspects of sketching would be a useful contribution, that might investigate whether clustered and redundant points are symptomatic of such indicators as hesitancy, deliberation, and calculation, that a recognition process might exploit.

A. Bibliography

[1] Morphological Features and Sequential Information in Real-time Handprinting Recognition.
M. Berthod and J.P. Maroy
Proc. Second Int. Joint Conf. on Pattern Recognition, Aug 1974.

[2] Interactive Hand-drawn Diagram Input System
O. Kato, H. Iwase, M. Yoshida, J. Tanahashi
Proc. of PRIP '82. IEEE Computer Soc. Conf. on Pattern Recognition and Image Processing, Jun 1982 Las Vegas.

[3] Graphical Input Through Machine Recognition of Sketches
C.F Herot
Computer Graphics (ACM) vol.10 no.2 pp.97-102 summer 1976.

[4] Use of Incremental Curvature For Describing and Analysing Two-dimensional Shape.
H. Freeman
Proc. Conf. Pattern Recognition and Image Processing 1979, IEEE.

[5] Sketching, An Informal Dialogue Between Designer and Computer
J. Taggart
Computer Aids to Design and Architecture, ed. N.Negroponte Petrocelli/Charter 1975.

[6] Recent Advances in Sketch Recognition
N. Negroponte
Proc. AFIPS 1973 pp.663-75.

[7] Piecewise Linear L1 Approximation of Plane Curves
N.M. Abdelmalek
Proc. 7th Int. Conf. Pattern Recognition, vol.1 pp.105-8 Canada 1984 IEEE.

[8] Approximation of Line Drawings by Straight Lines and Circular Arcs
M. Nagura
Trans. IECE Japan pp.839-45 September 1981 (Japanese).

[9] A Corner-finding Algorithm for Chain-coded Curves
H. Freeman and L.S. Davis
IEEE Trans. Comp. C-26 no.3 pp.297-303.

[10] Shape Description via the Use of Critical Points
H. Freeman
Pattern Recognition vol.10 no.3 pp.159-66, 1978.

[11] On-line Cursive Script Recognition
C. Higgins, R. Whitrow
Interact Conf. 1984 Elsevier Science Publishers.

[12] Structural Pattern Recognition
T. Pavlidis
Springer-Verlag, 1977.

[13] Curve Fitting as a Pattern Recognition Problem
T. Pavlidis
Proc. 6th Int. Conf. on Pattern Recognition pp.853-8 1982.

[14] The Numerical Evaluation of B-Splines
M.G. Cox
J. Inst. Mathamatical Applications pp.134-49 vol.10 1972.

[15] Robust Shape Description Based On Curve Fitting
J. Eklundh, J. Howako
Proc. 7th Int. Conf. Pattern Recog. Montreal 1984 pp.109-12.

[16] Segmentation of Line Drawings For Recognition and Interpretation
A. Belaid, G. Masini
Technology and Science of Informatics vol.1 no.2 1983 pp.121-33.

[17] Mathematical Statistics
J. Freund
Prentice/Hall International 2nd edition pp.112-14.

[18] Principles of Interactive Computer Graphics
W. Newman, R. Sproull
McGraw-Hill Kogakusha 2nd edition pp.27-8.

[19] MIRABELLE, A System For Structural Analysis of Drawings
G. Masini, R. Mohr
Pattern Recognition vol.6 no.4 pp.363-372 1983, Pergamon Press.

B. RESULTS

As described in chapter 3, sketch samples were donated by a variety of people who rendered target drawings in their own hand. In the following sections each page shows one sketch in four stages of recognition:-

top left

the raw sketch as it was drawn, captured by digitising;

top right

the result of stroke segmentation and fitting;

bottom left

the result after merging;

bottom right

the result after latching.

The configuration on each page of raw, segmented, merged, and latched has a key to remind the reader:-

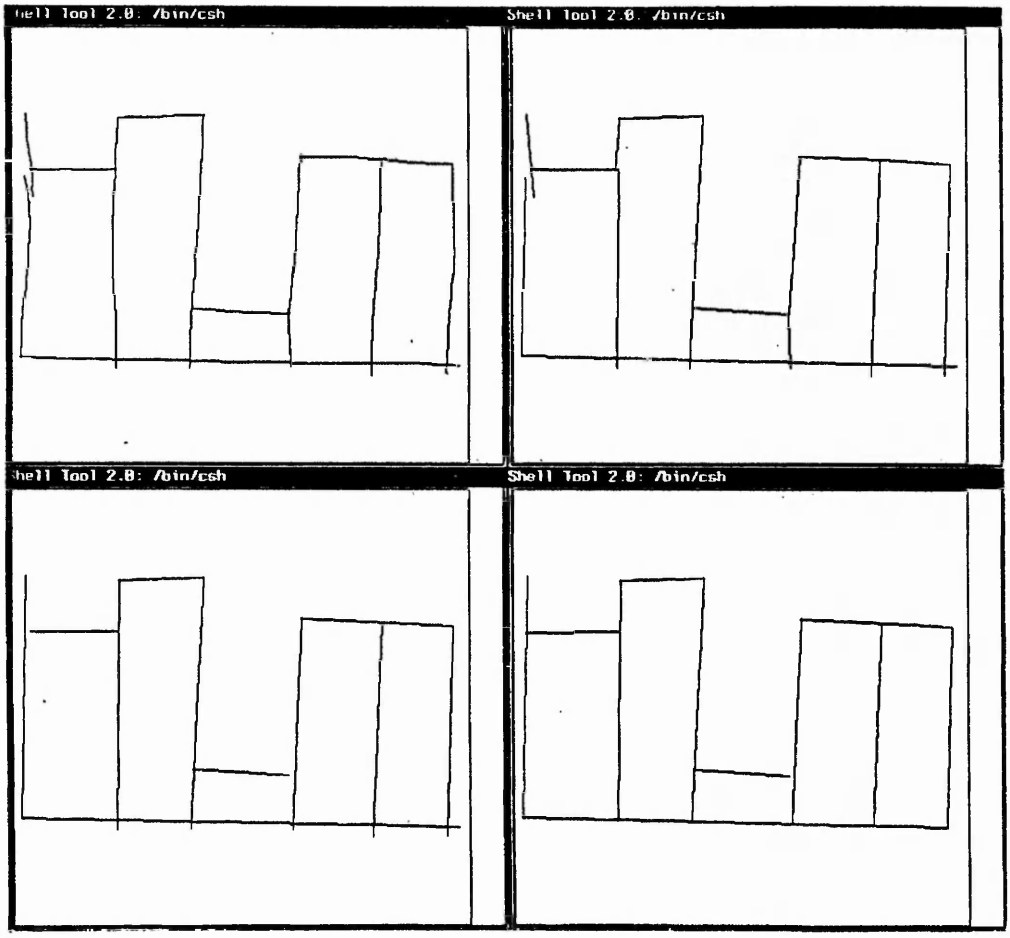
r s

m l

B.1 Histo Results

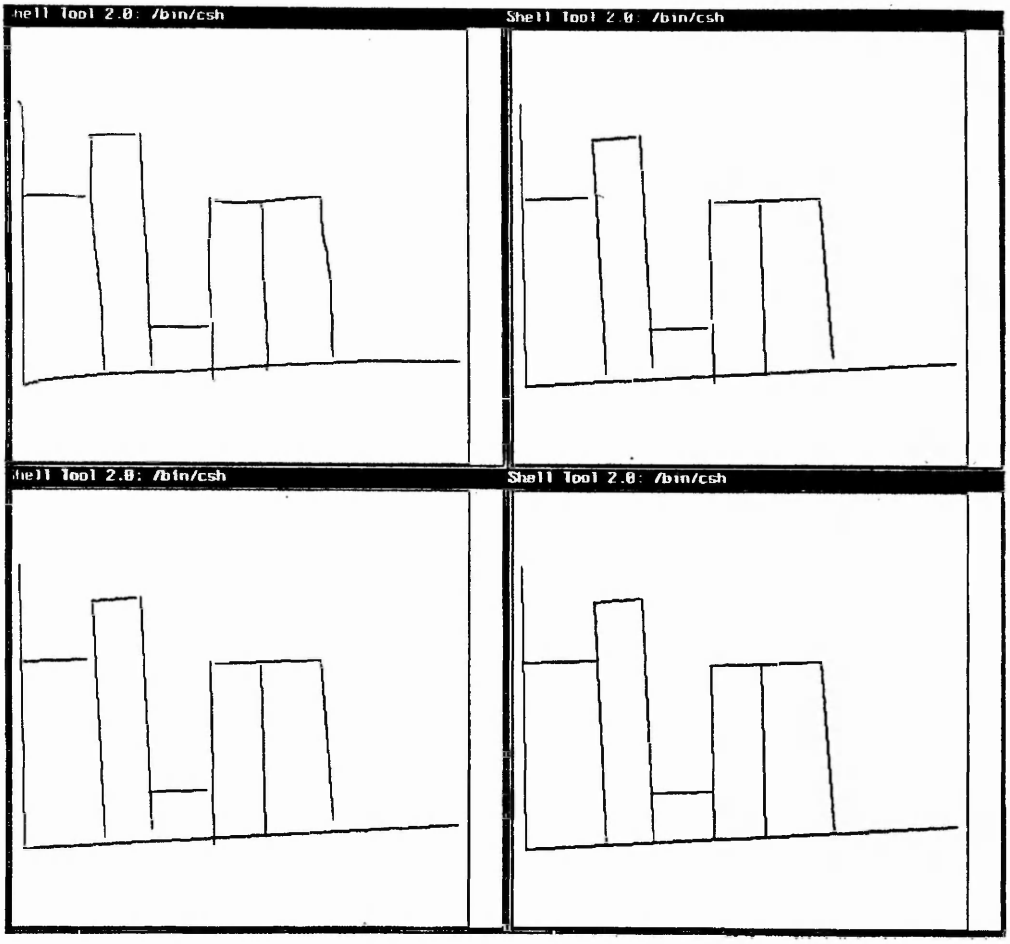
r s

m l



r s

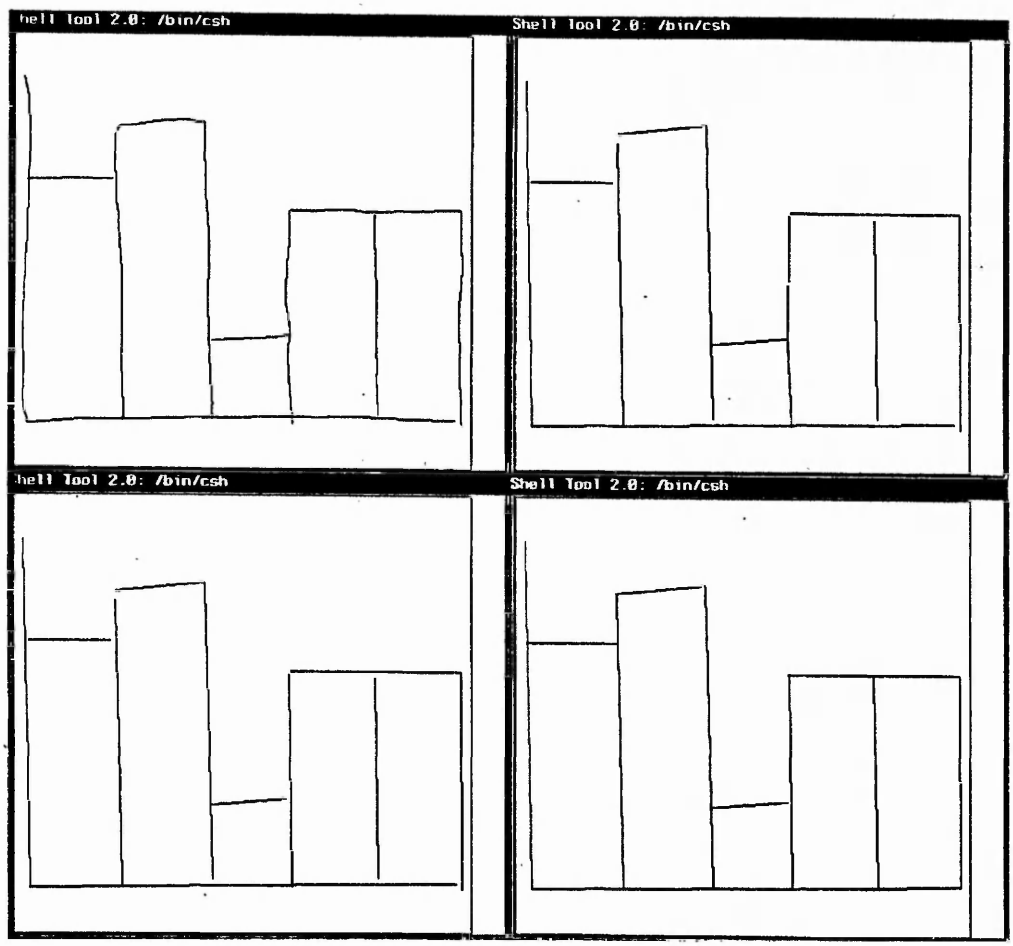
m l



B. Results 3

r s

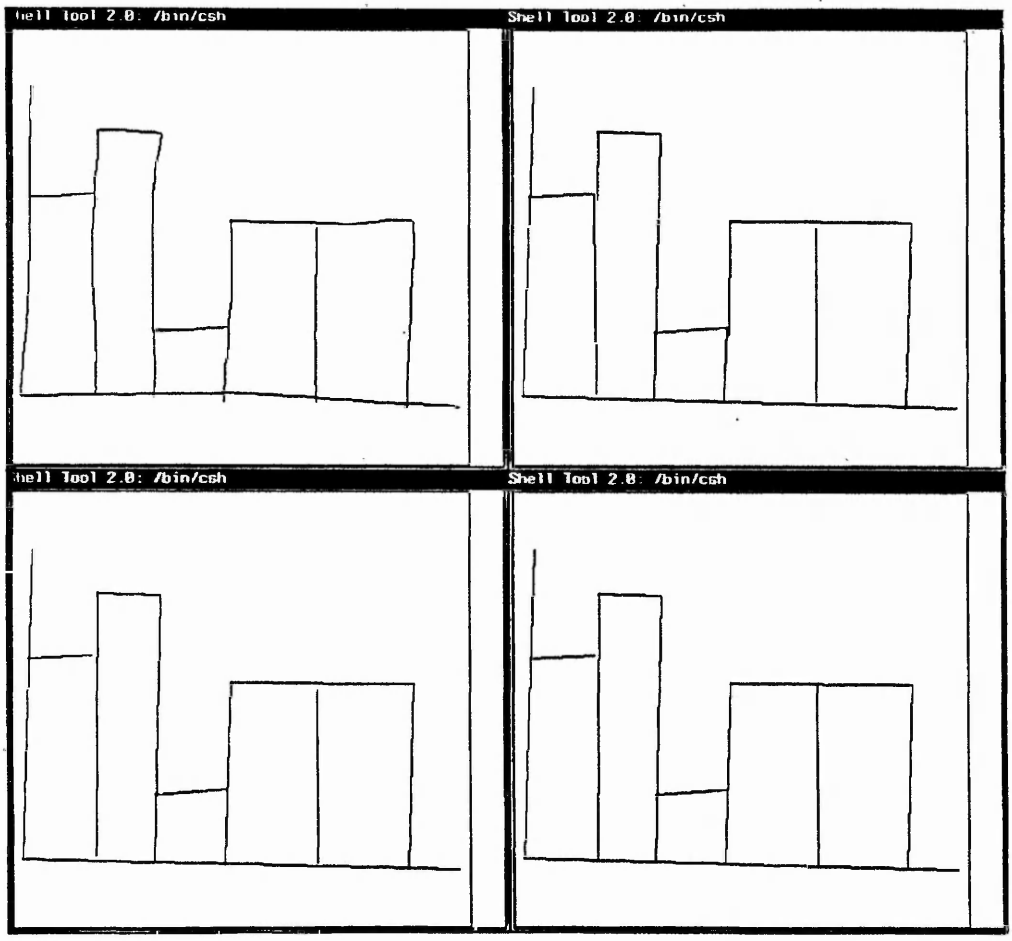
m l



B. Results 4

r s

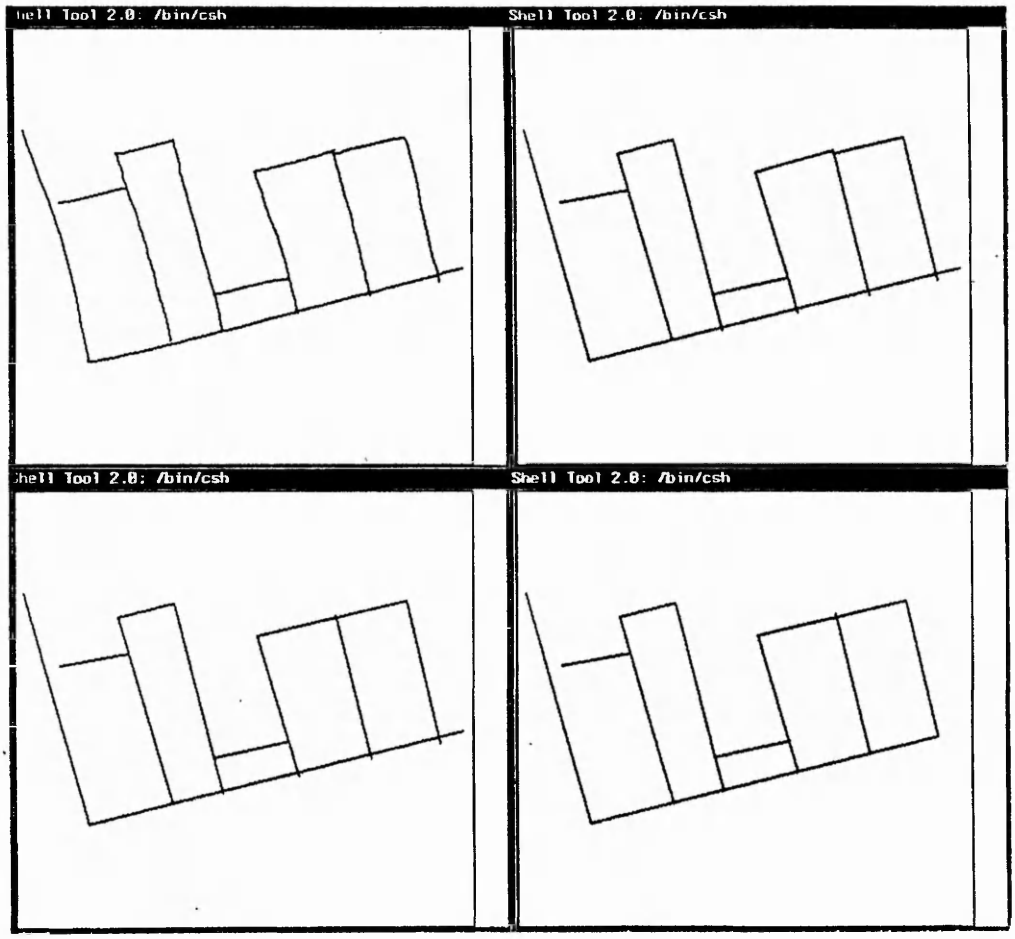
m l



B. Results 5

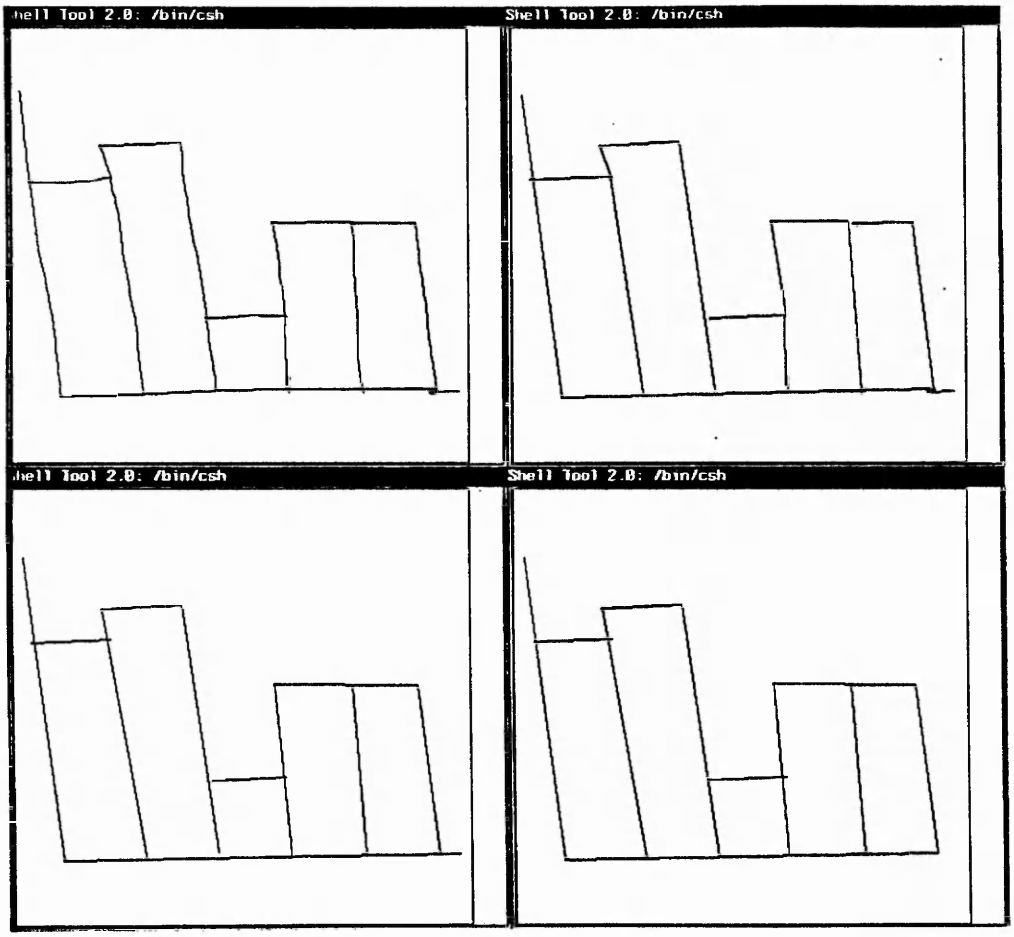
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

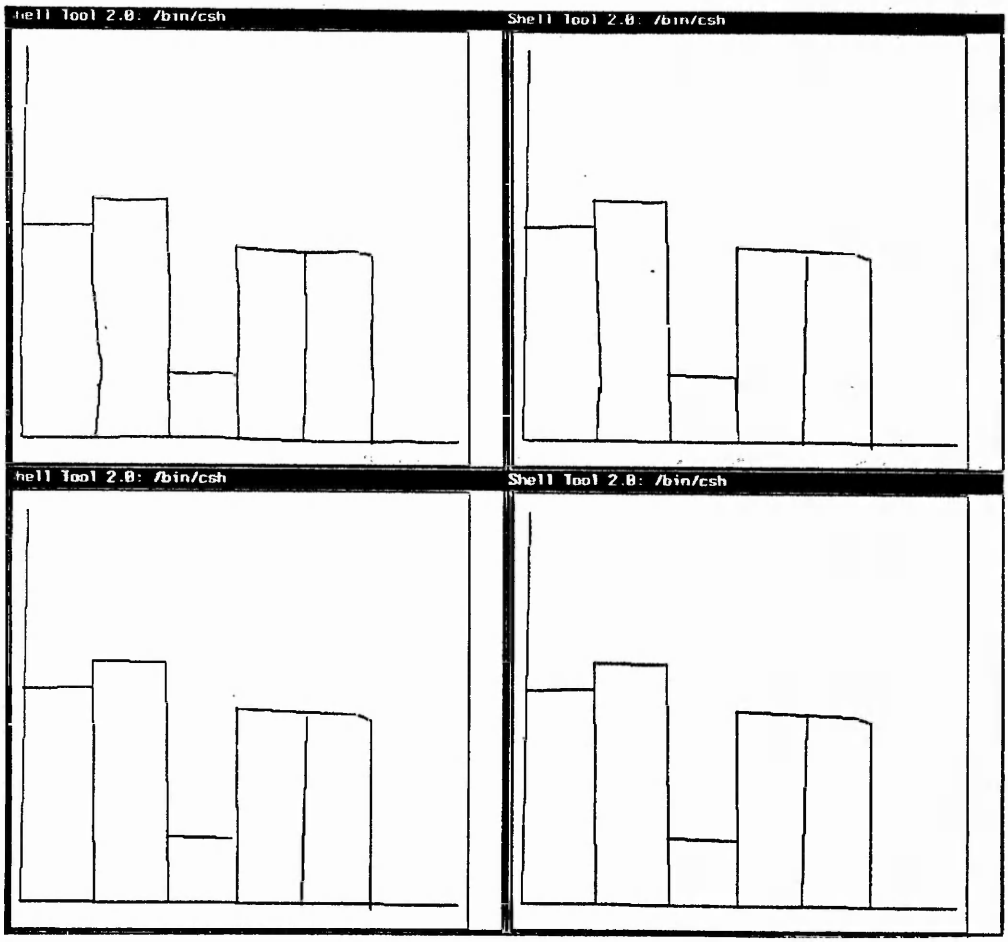
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

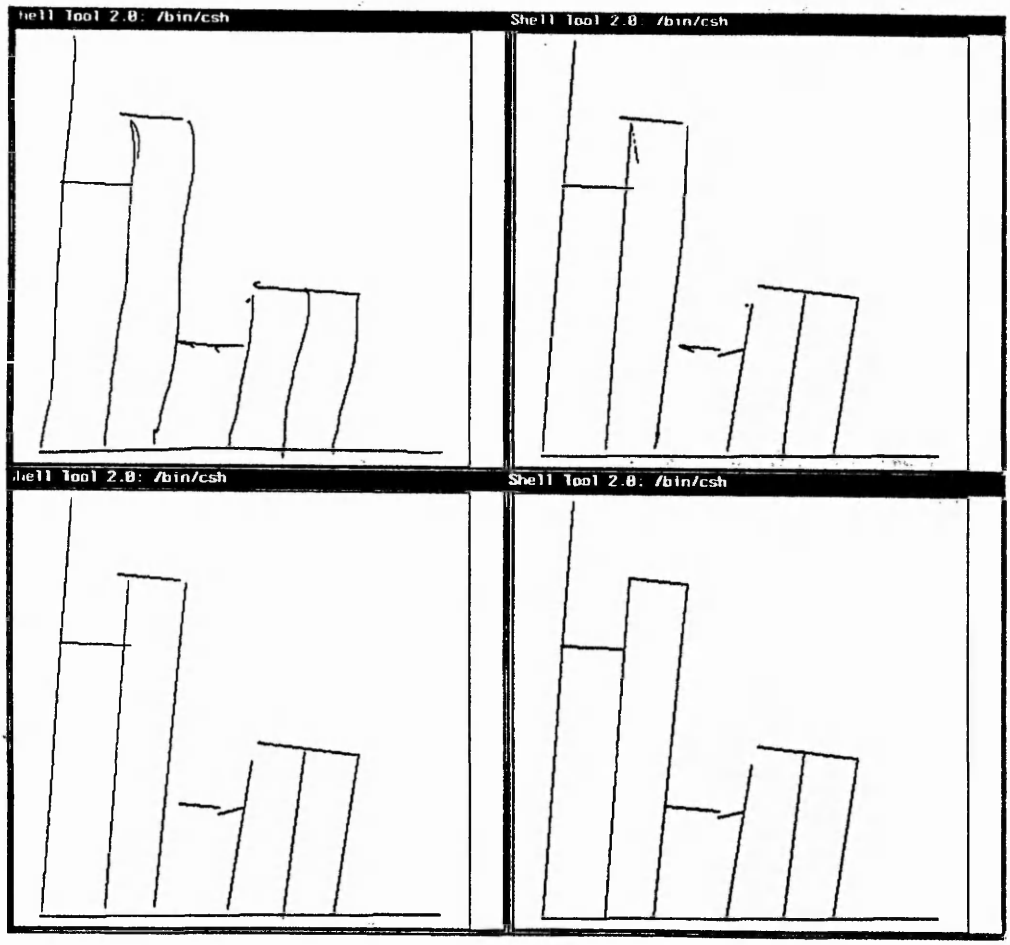
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

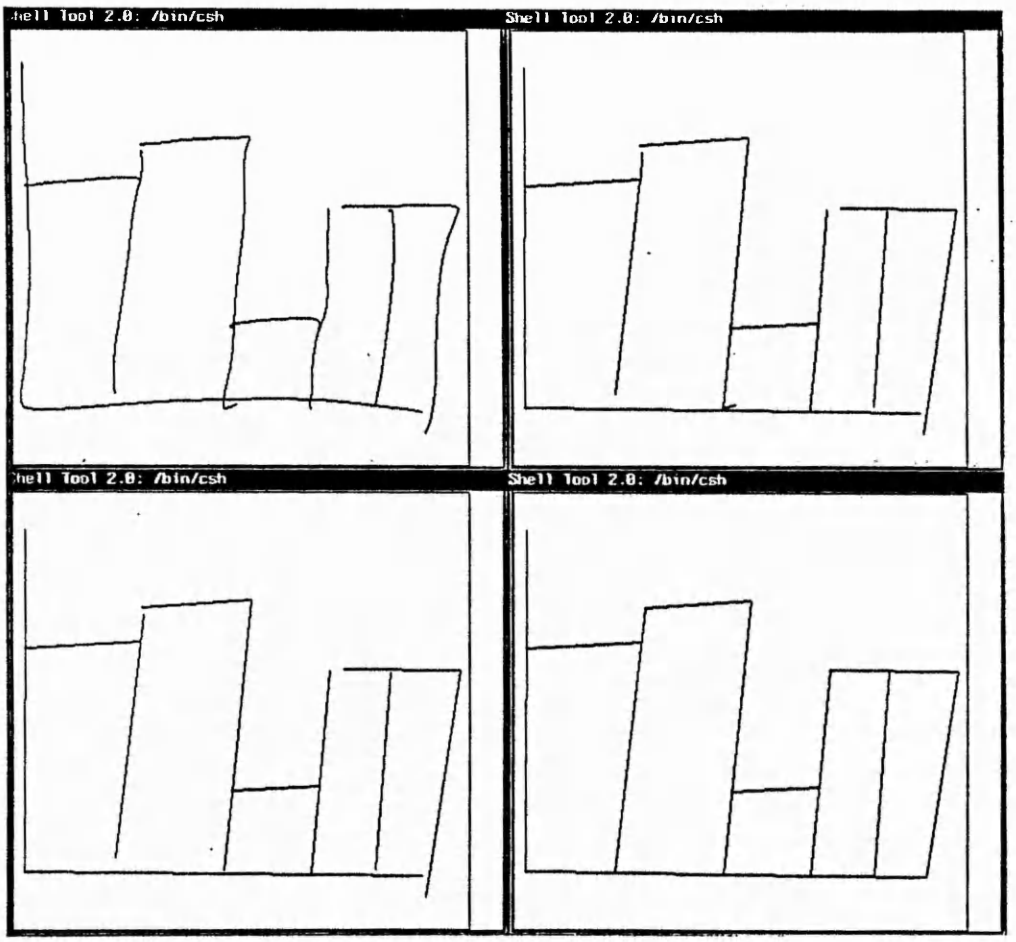
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

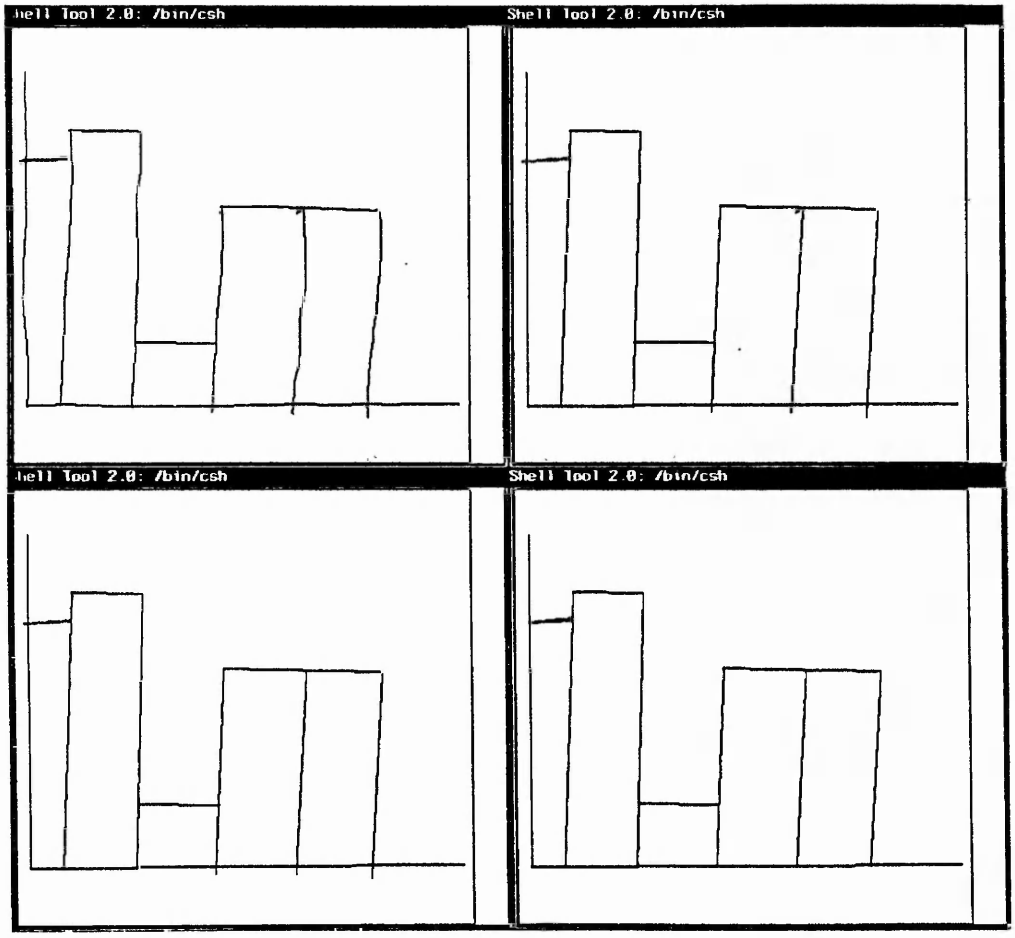
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

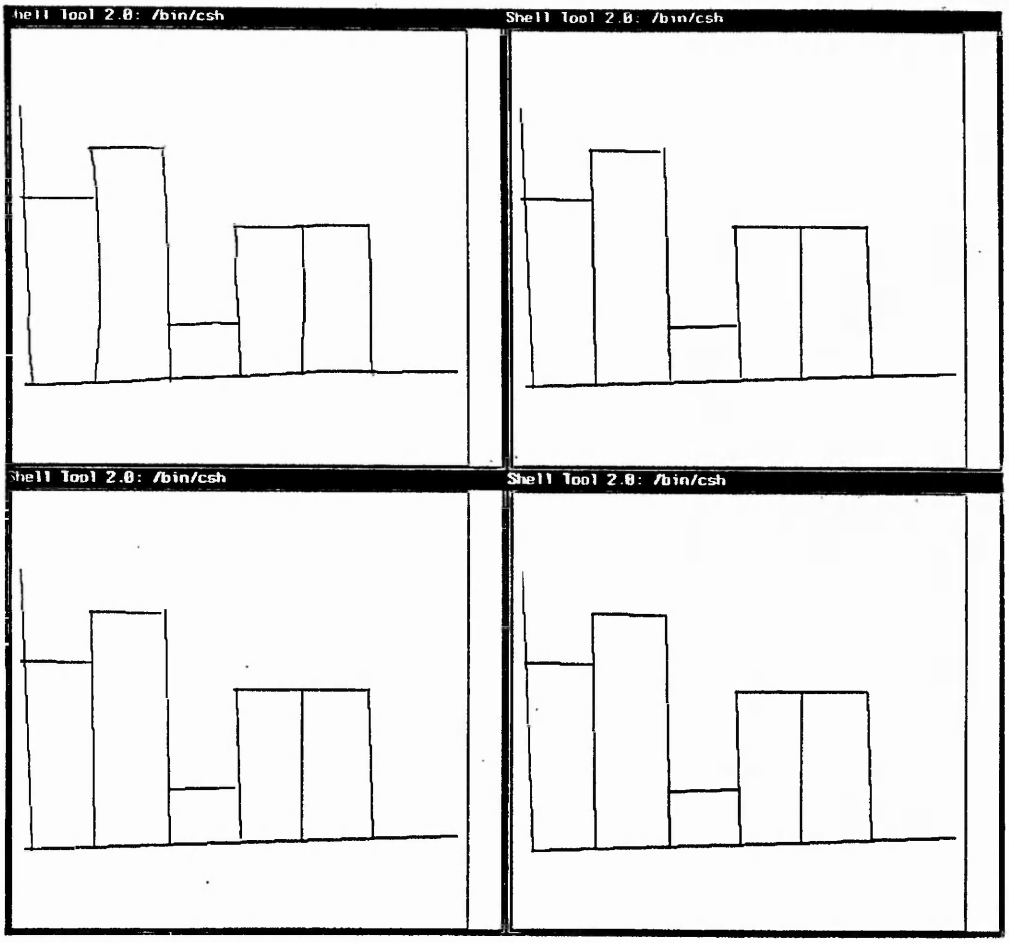
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

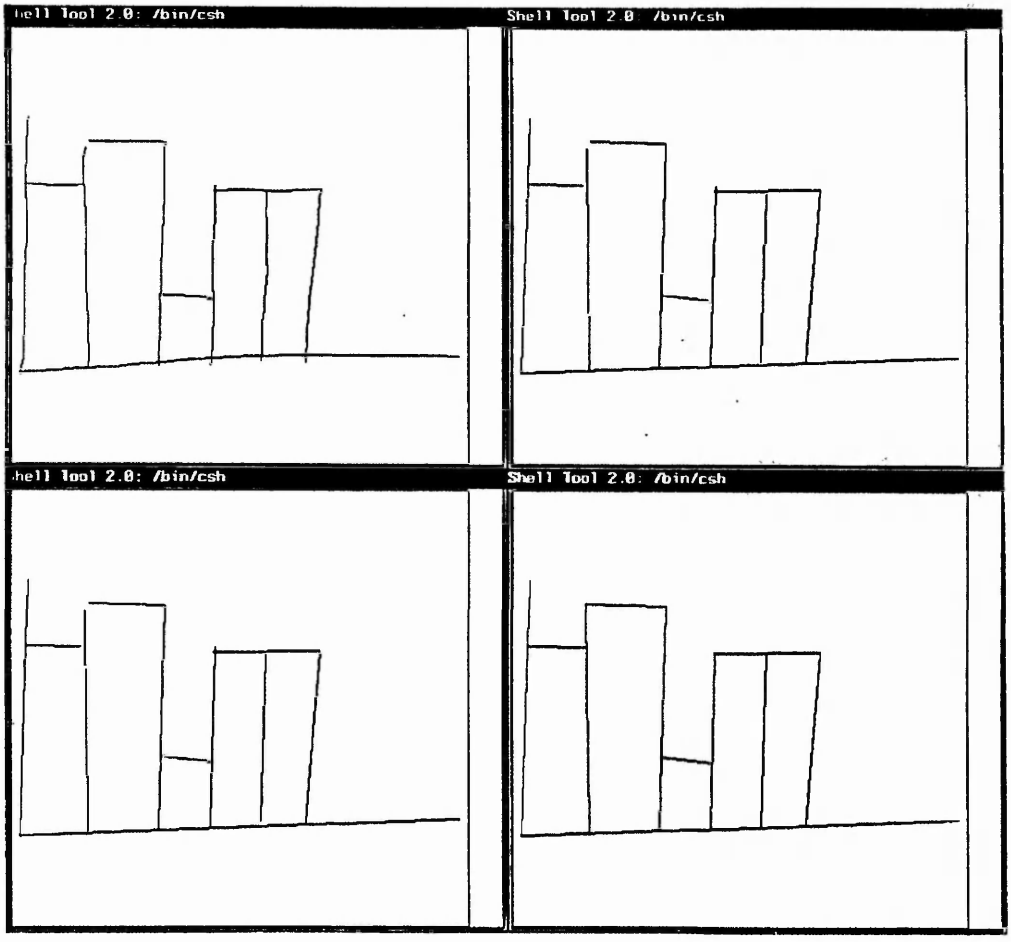
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

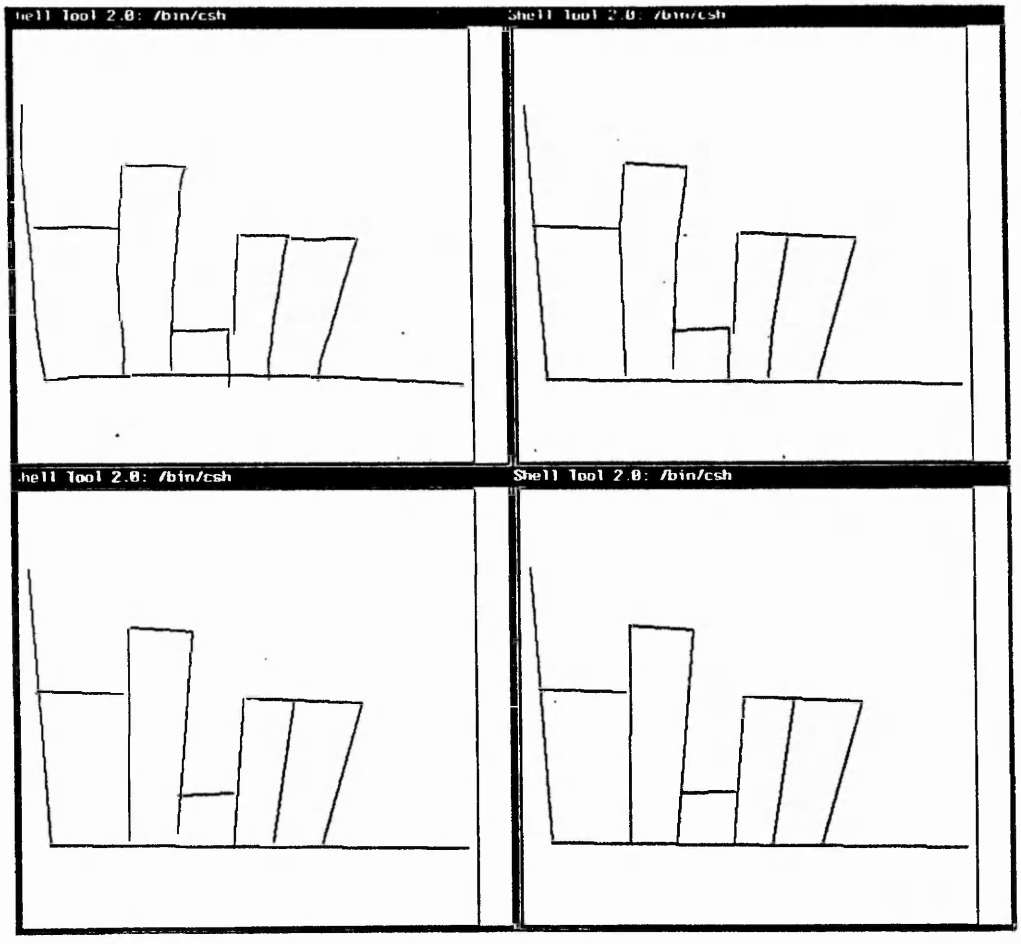
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

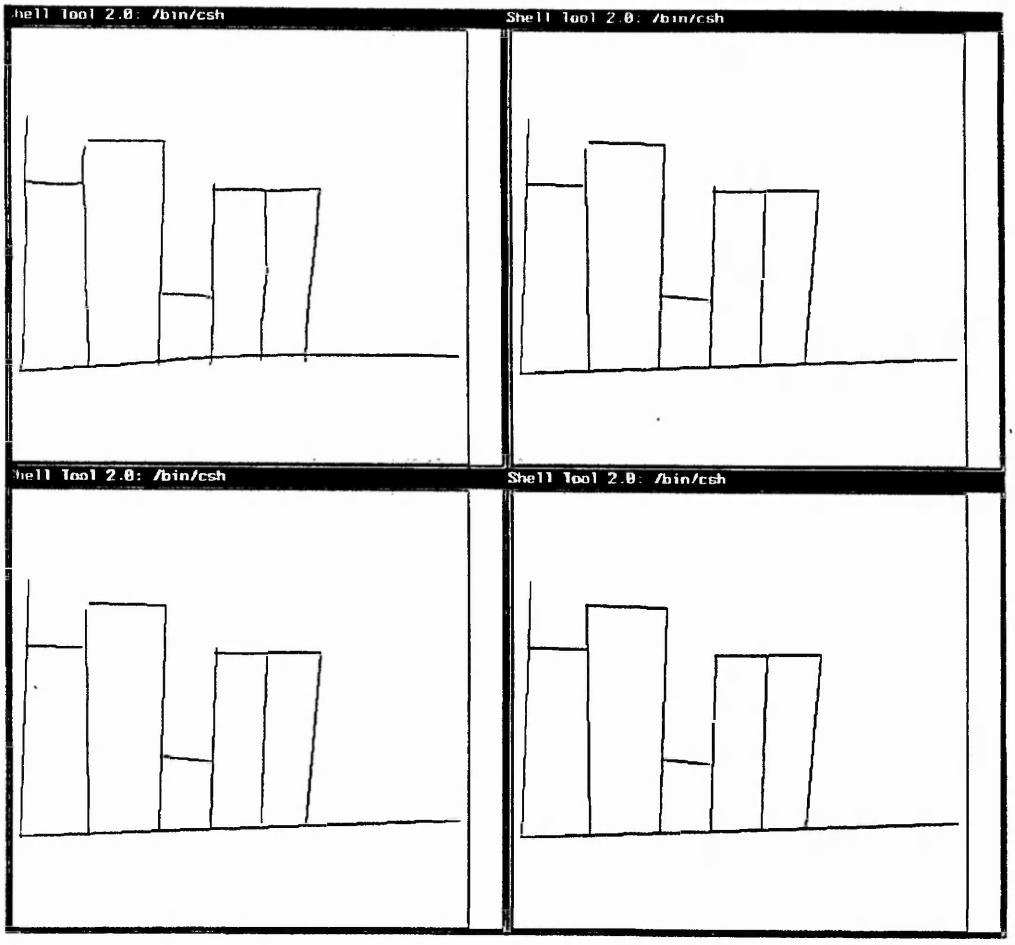
r s

m l


hell Tool 2.8: /bin/csh
Shell Tool 2.8: /bin/csh
hell Tool 2.8: /bin/csh
Shell Tool 2.8: /bin/csh

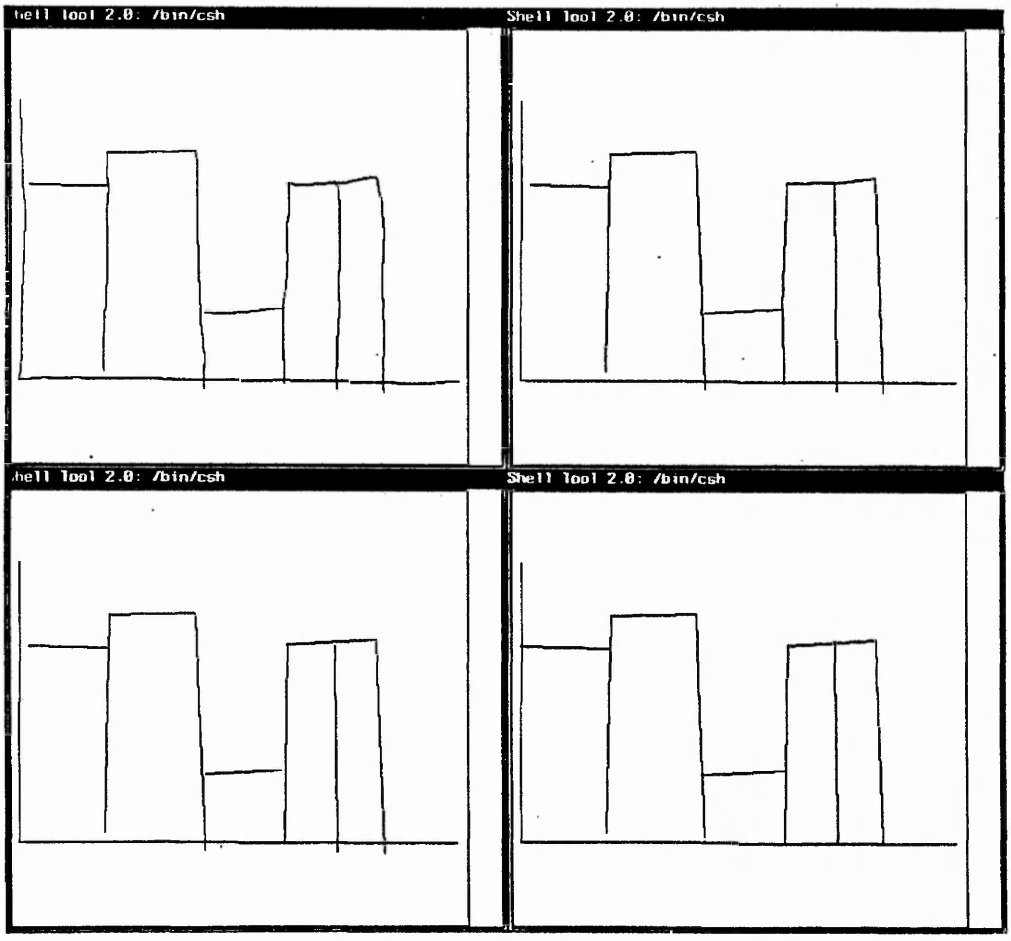
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

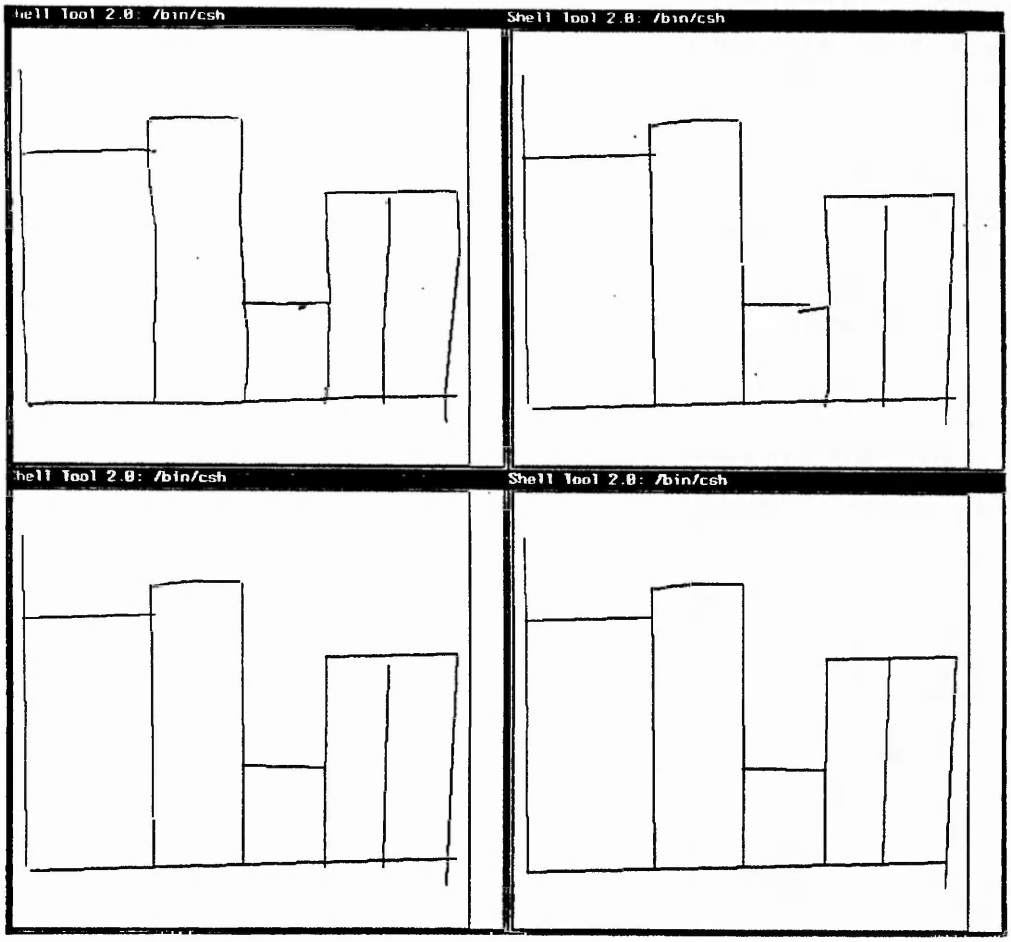
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

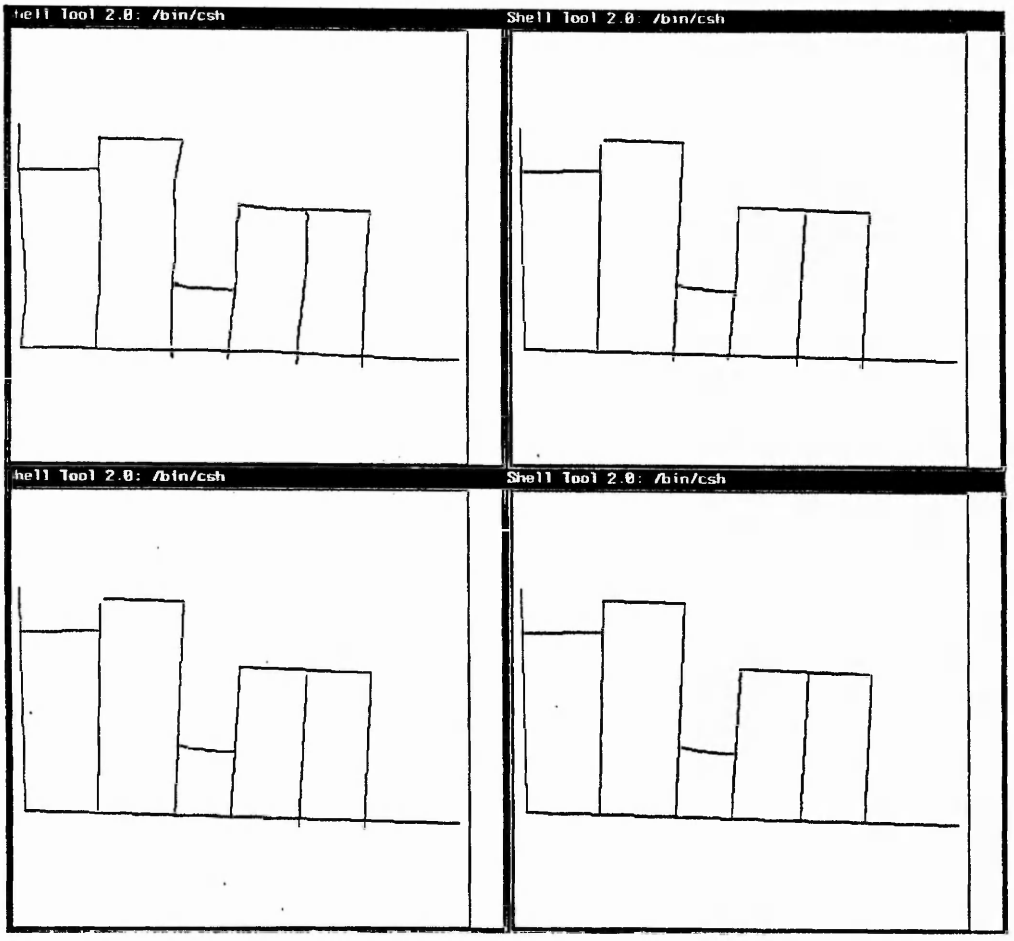
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

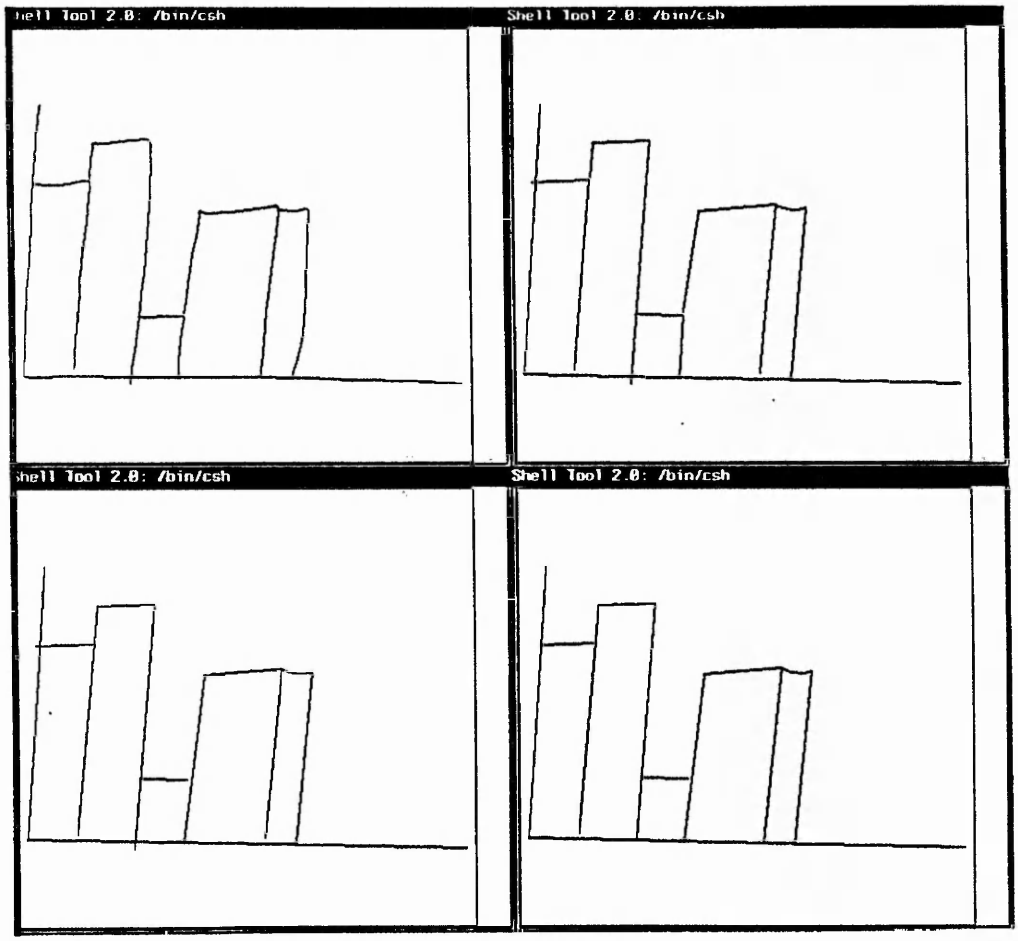
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

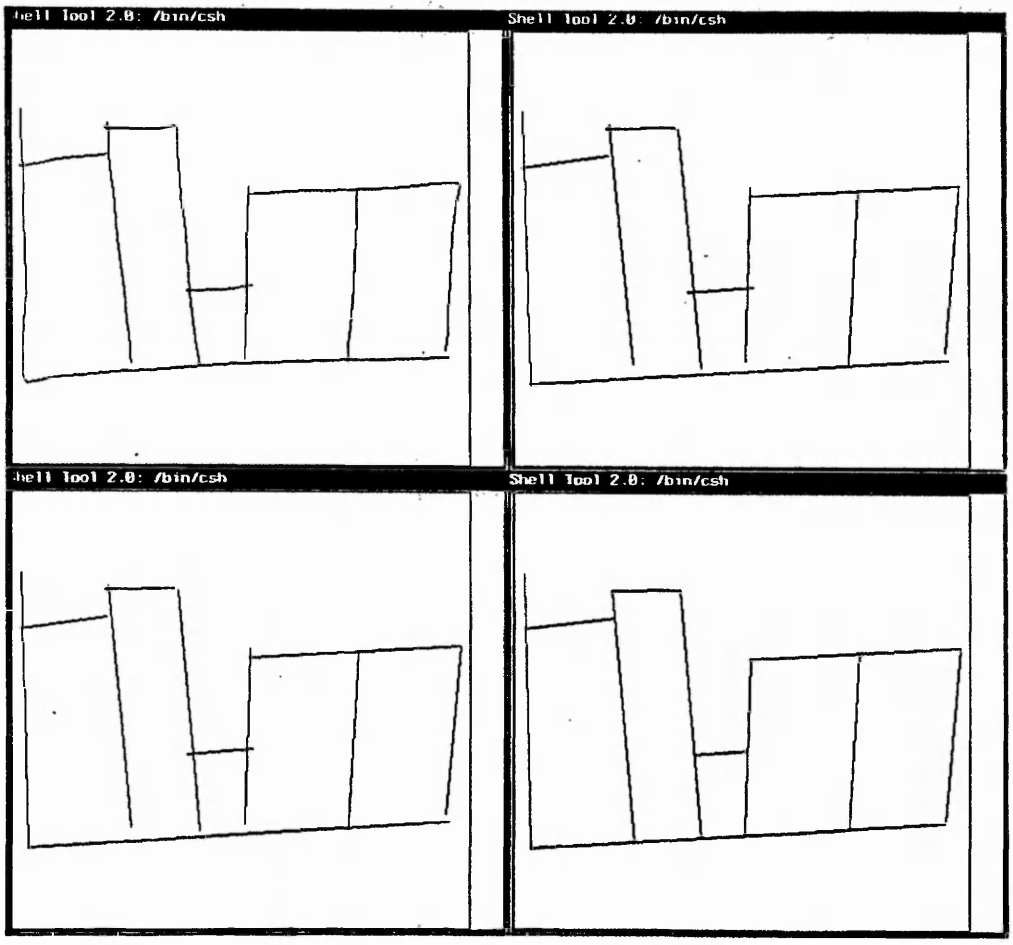
r s

m l


hell Tool 2.8: /bin/csh
Shell Tool 2.8: /bin/csh
shell Tool 2.8: /bin/csh
Shell Tool 2.8: /bin/csh

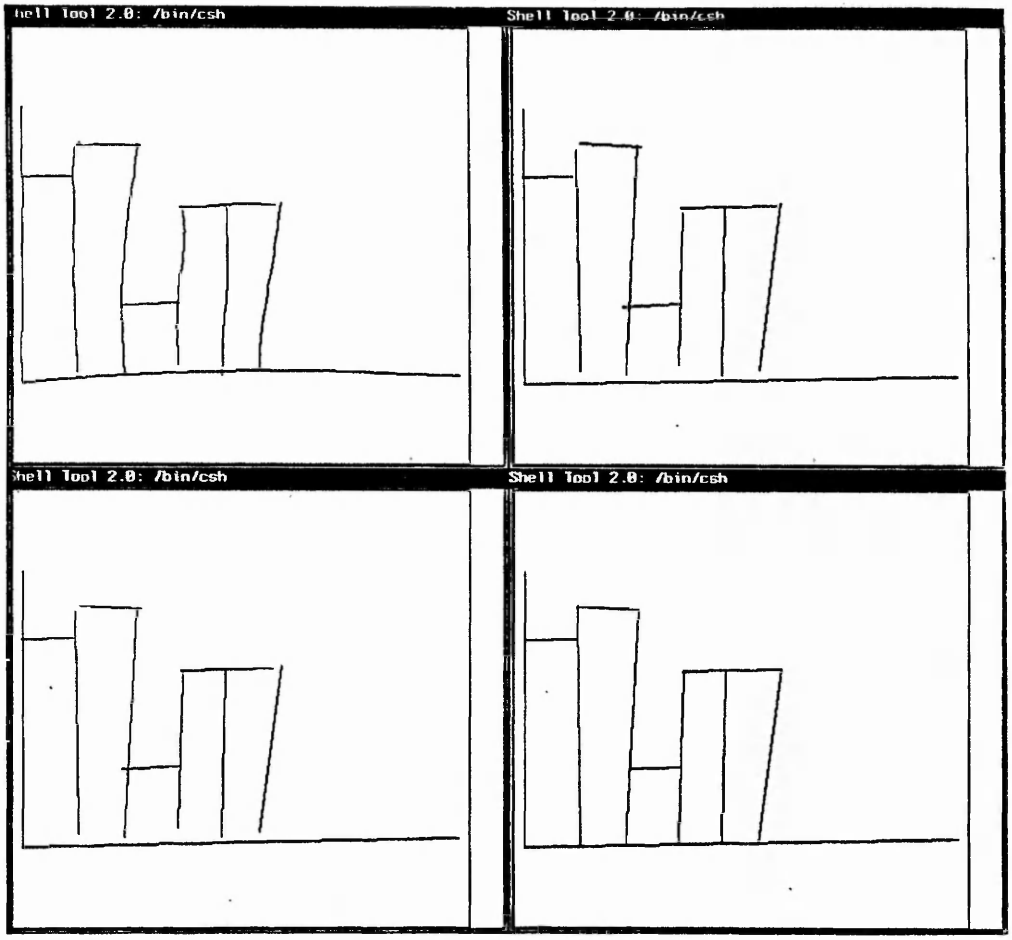
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

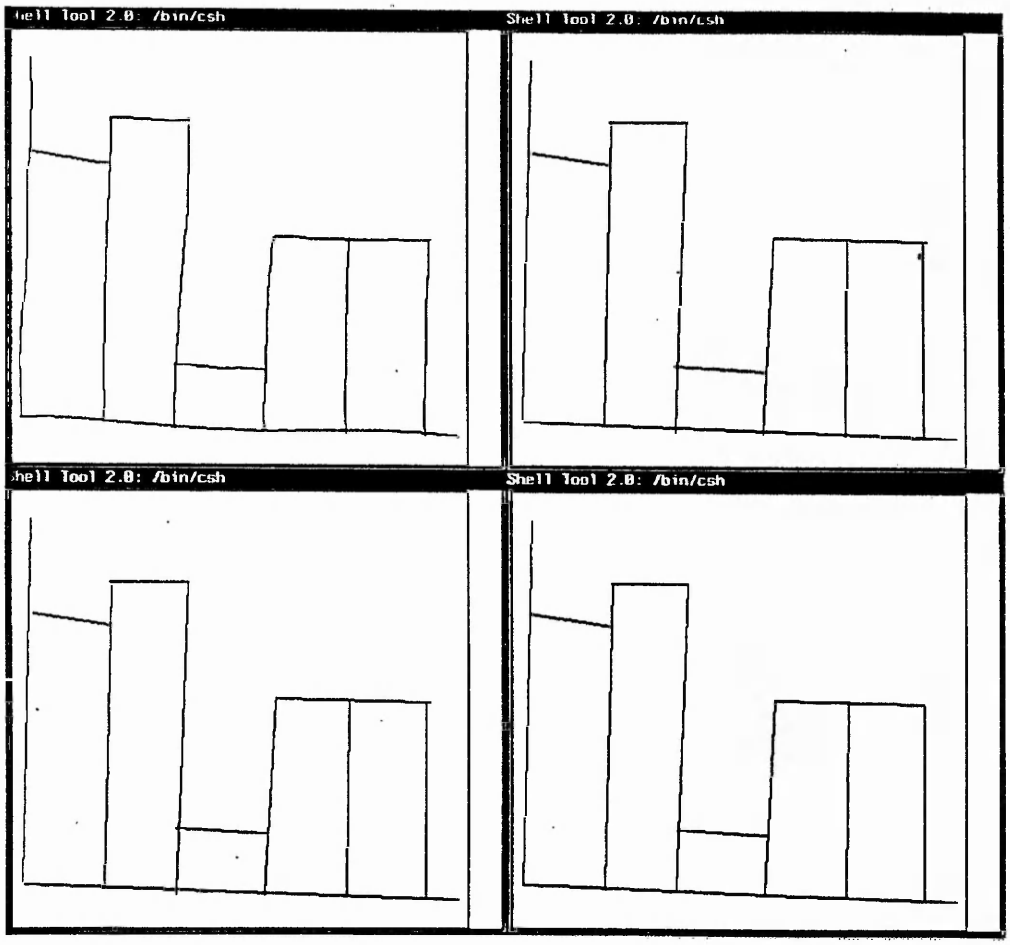
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

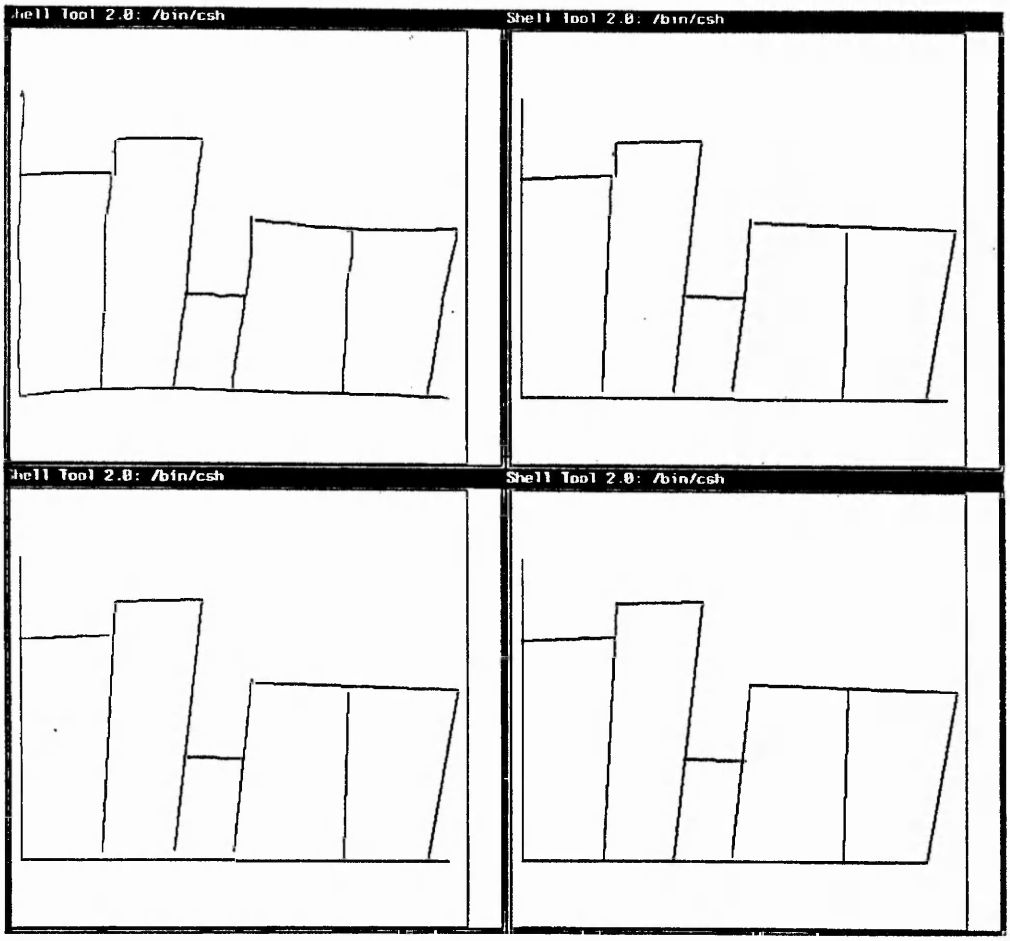
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

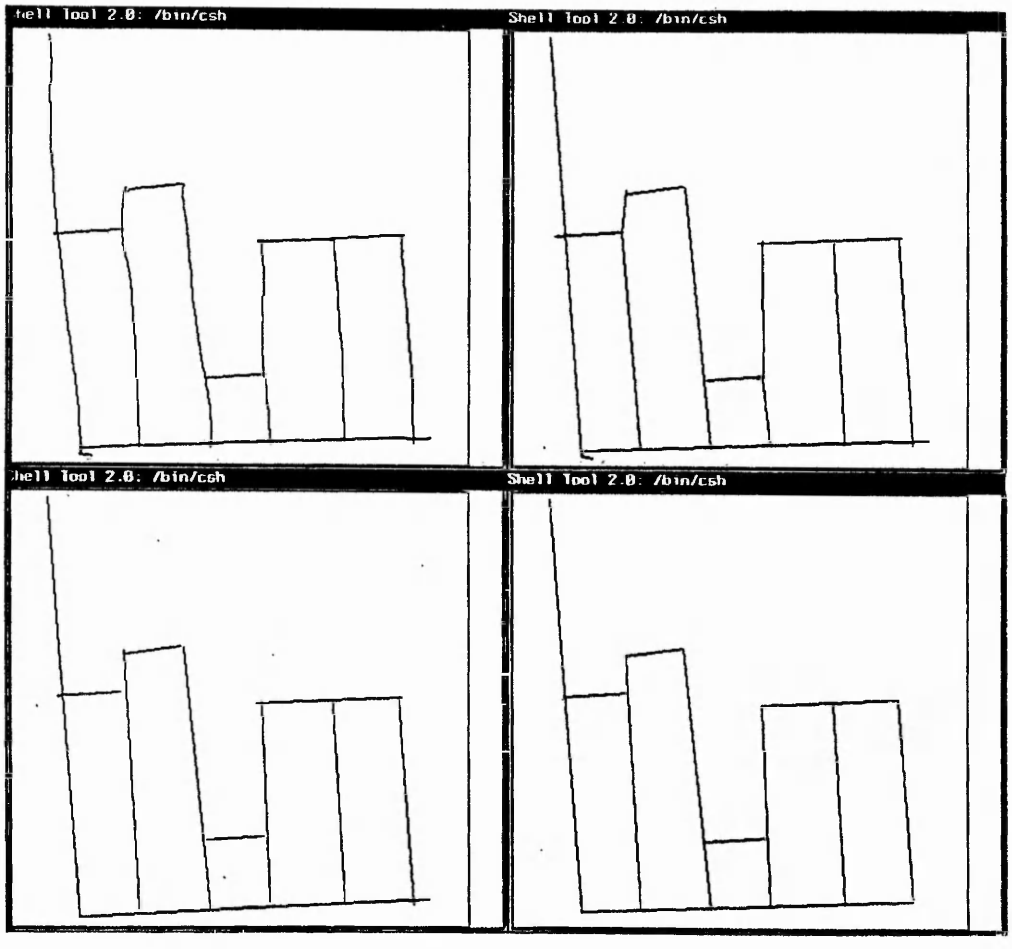
r s

m l.


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

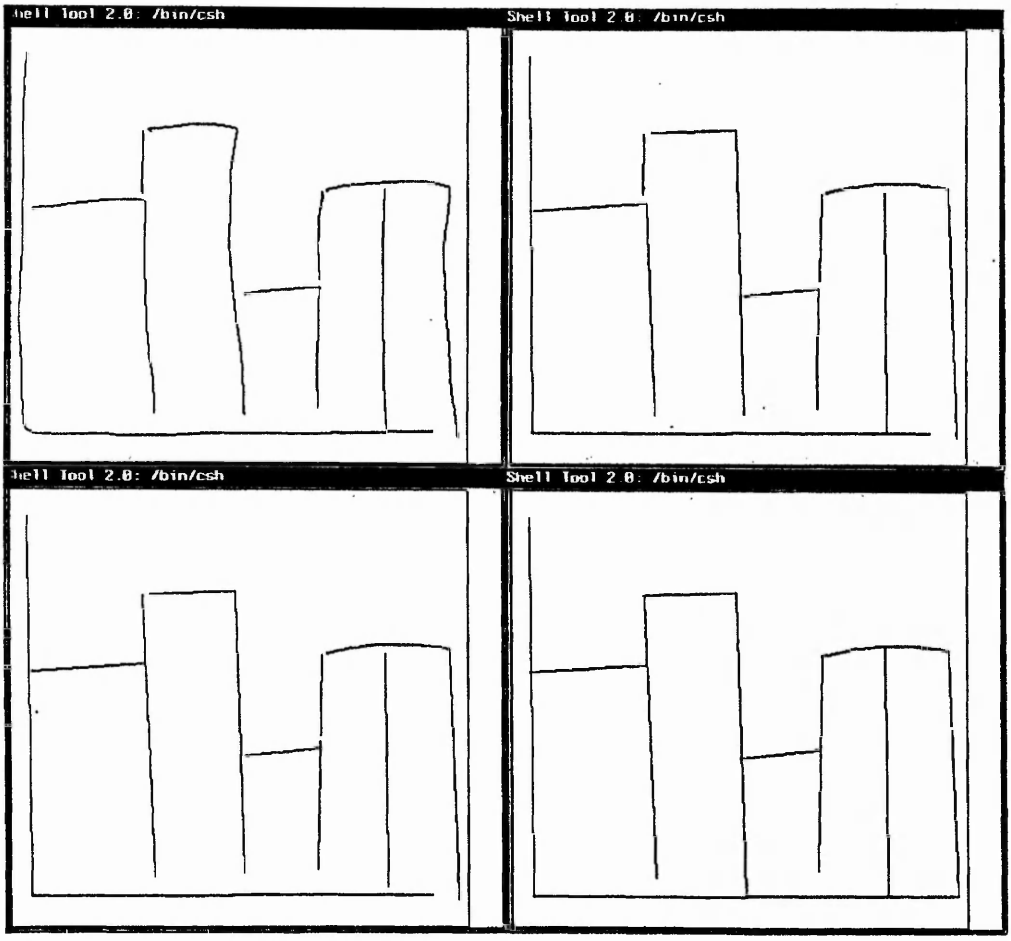
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

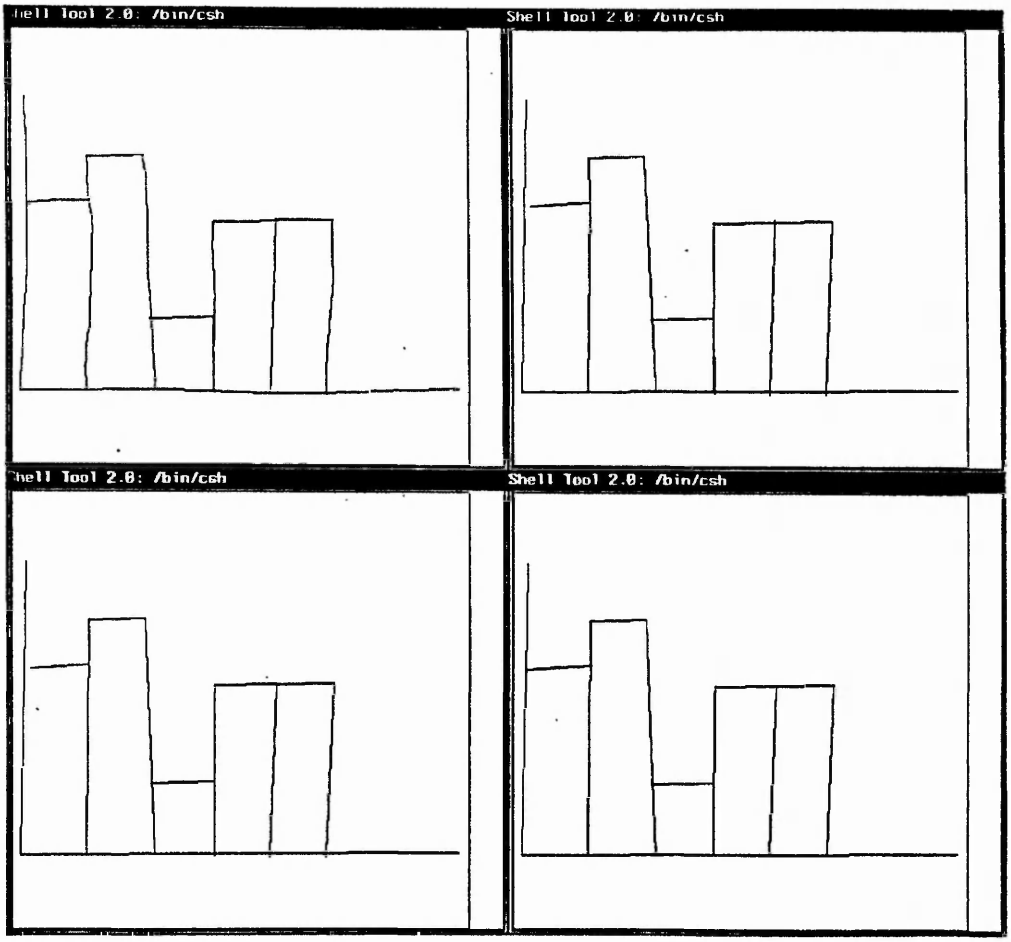
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

r s

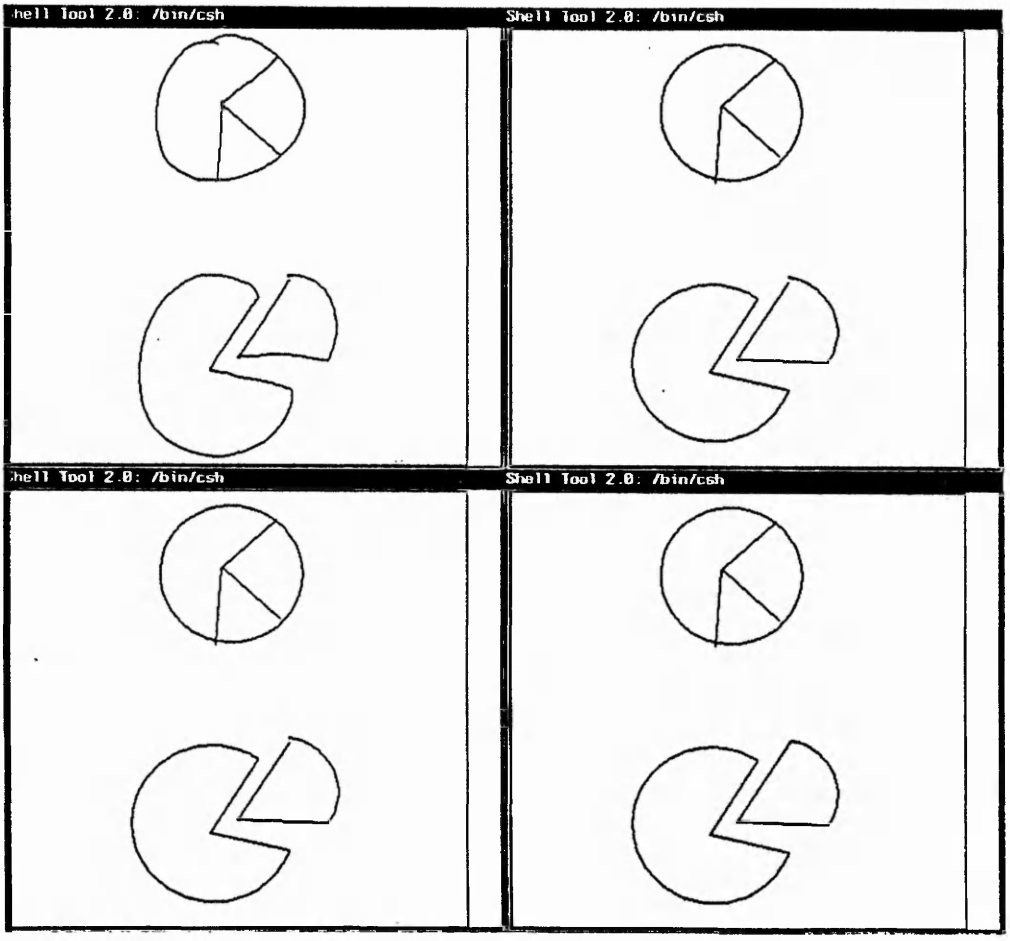
m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

B.2 Pie Results

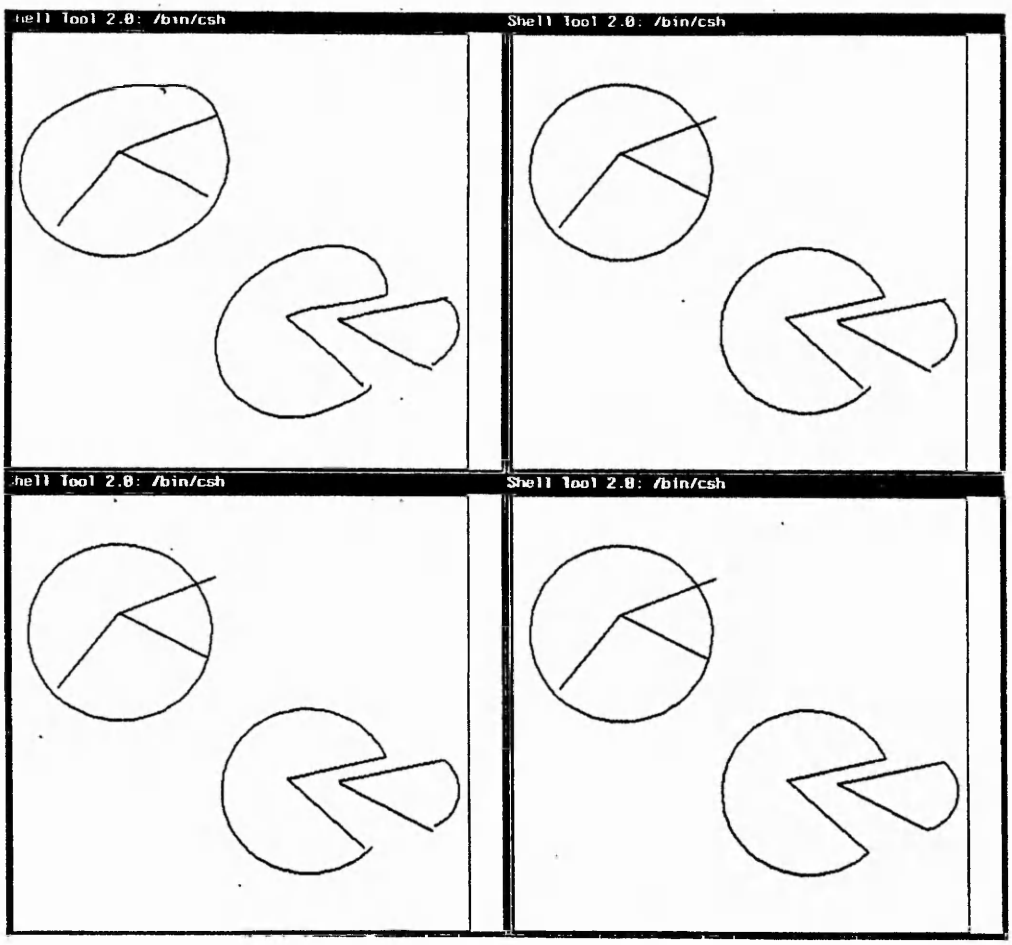
r s

m l



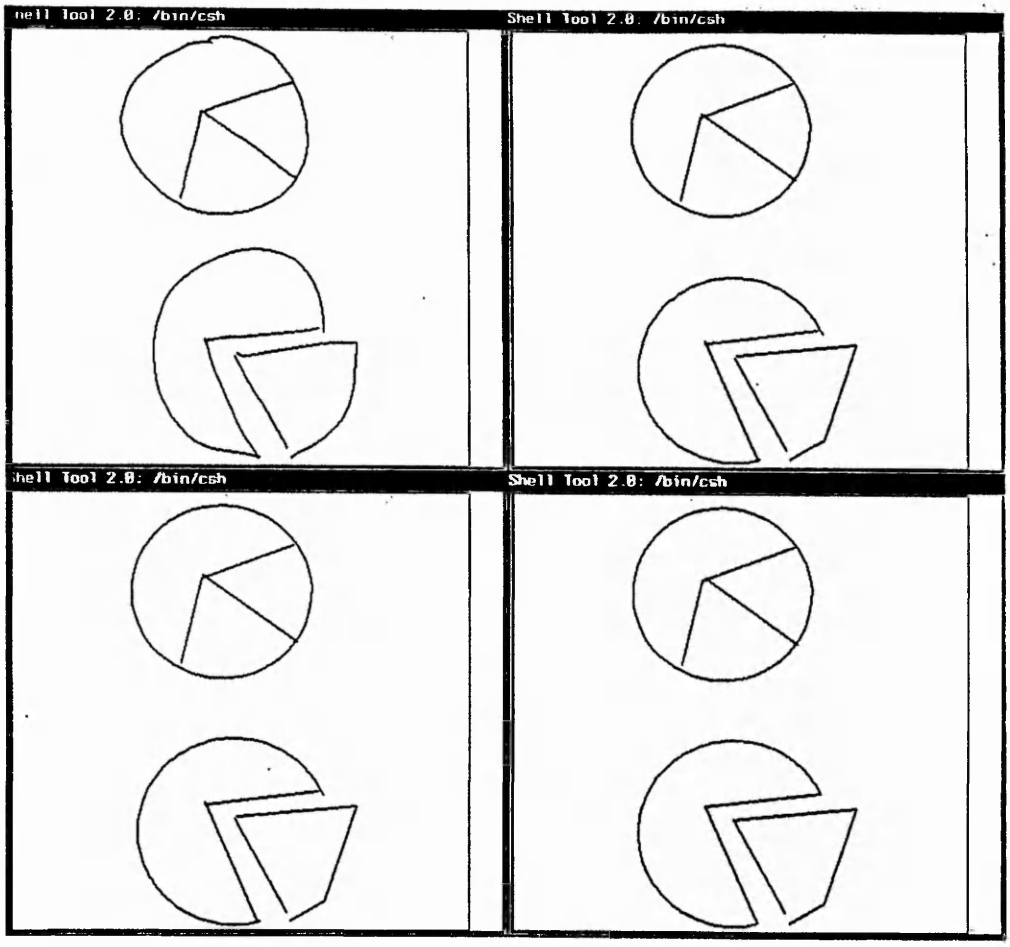
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

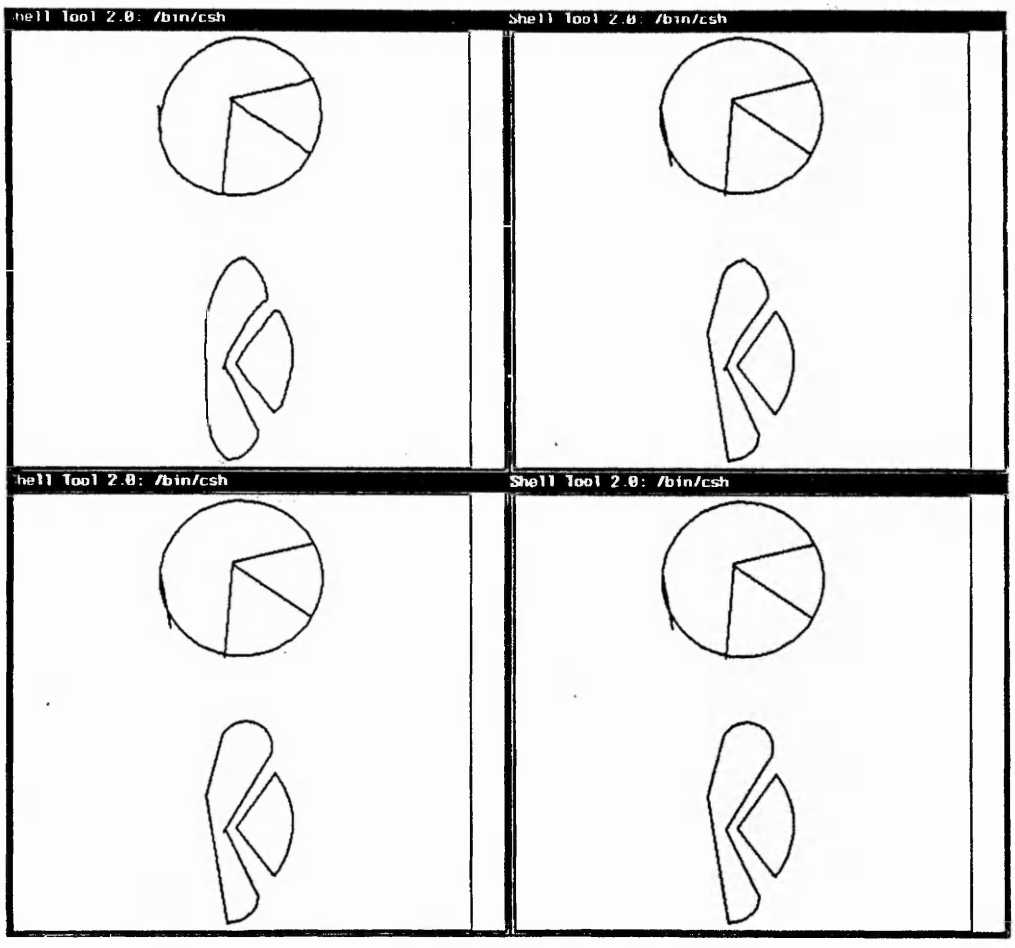
r s

m l


nell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

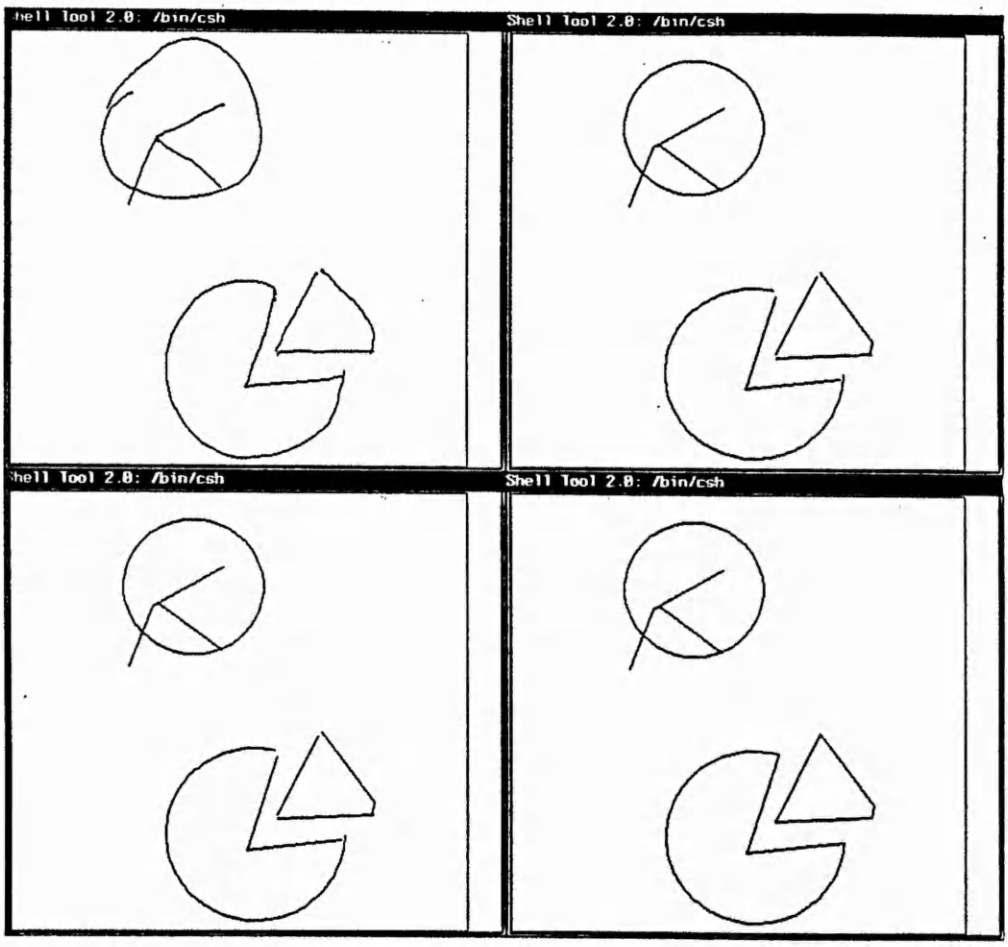
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

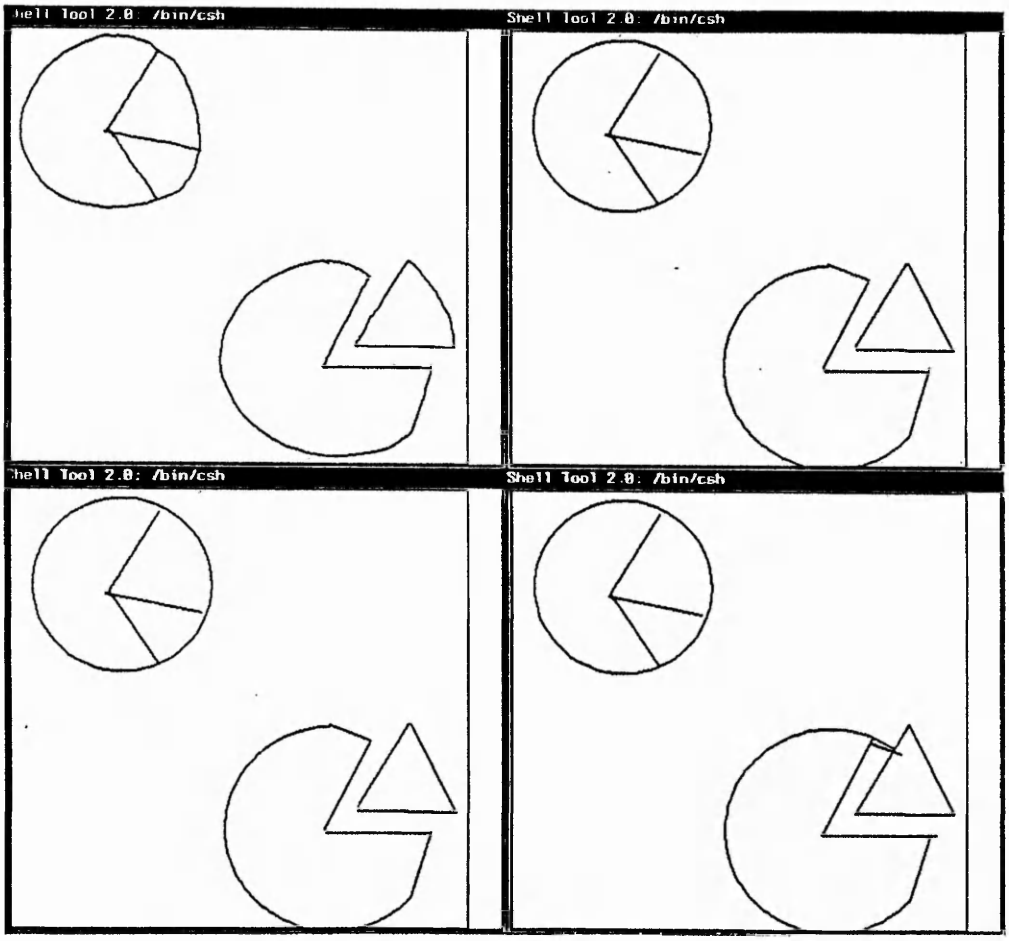
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

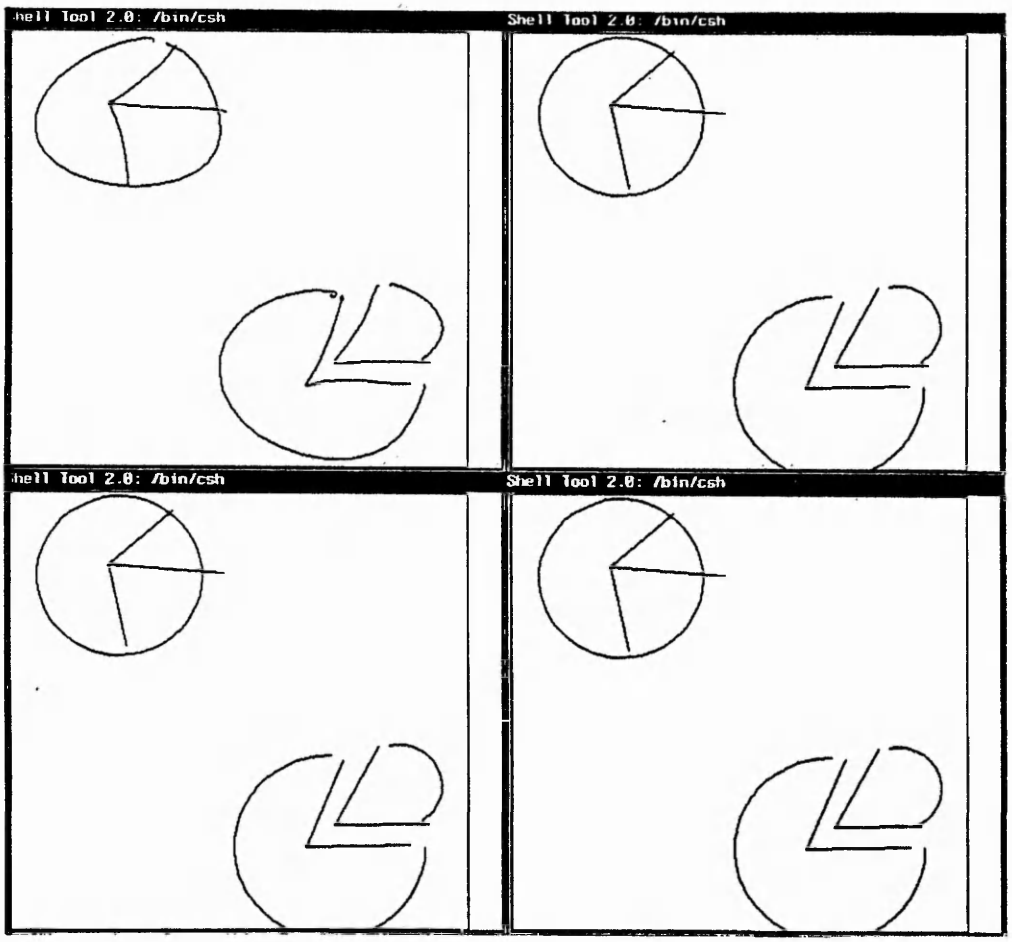
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

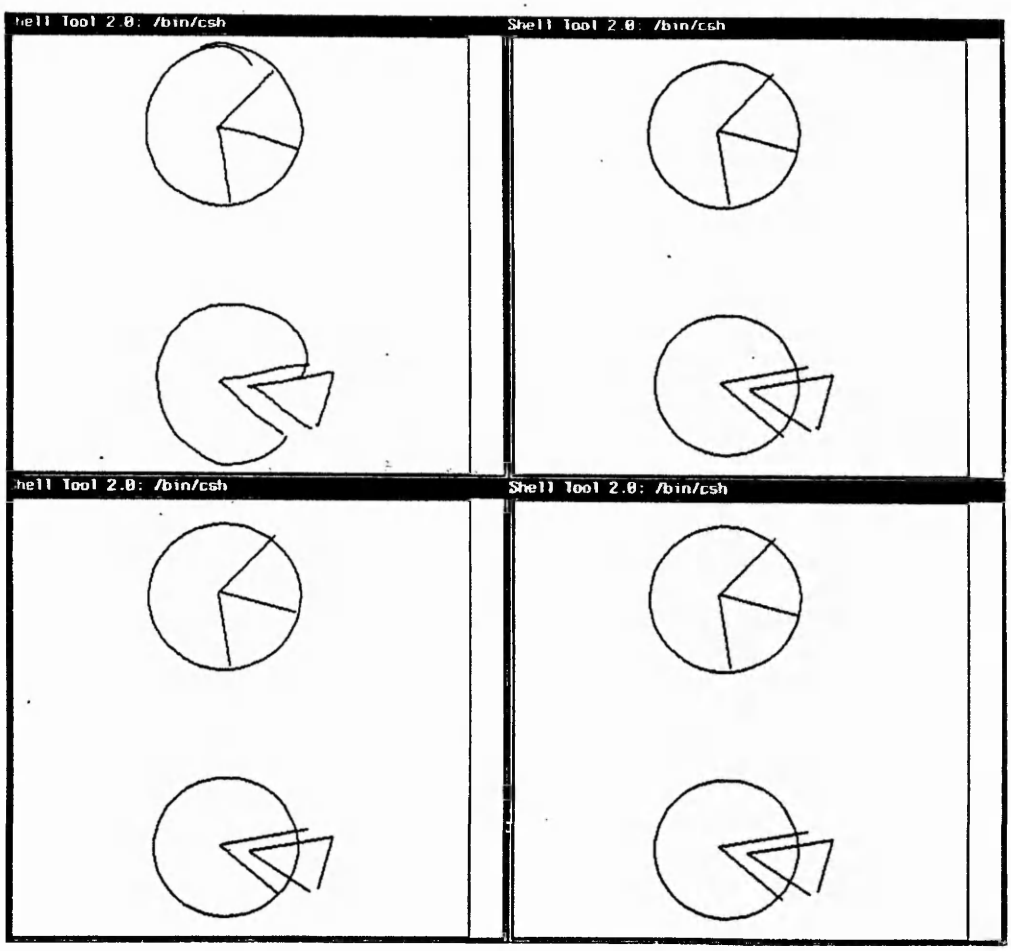
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

r s

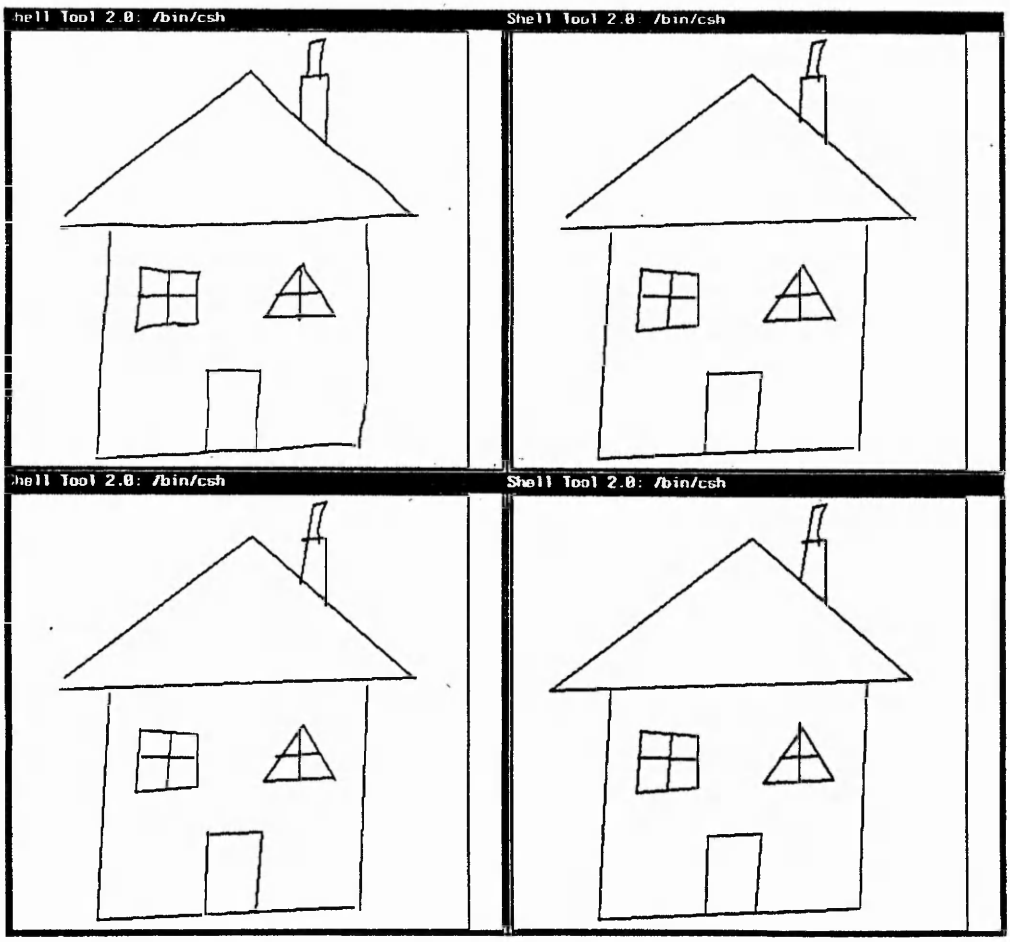
m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

B.3 House Results

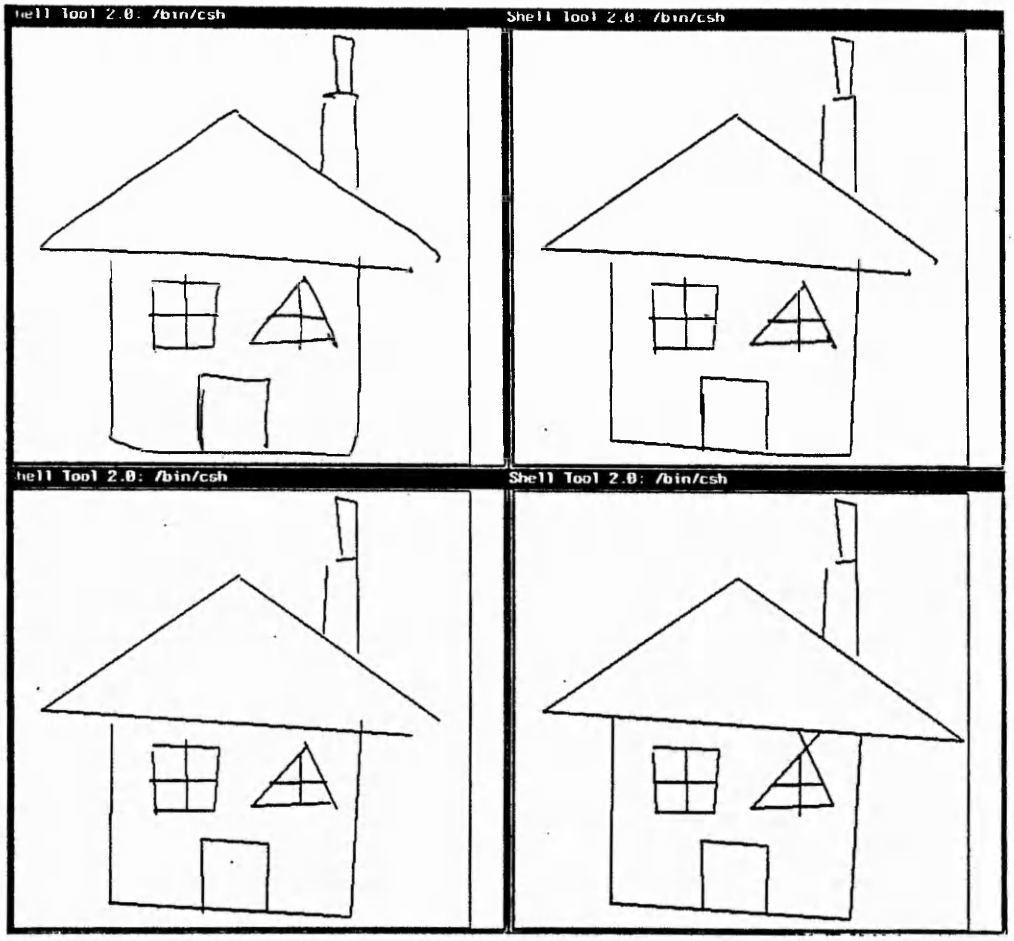
r s

m l



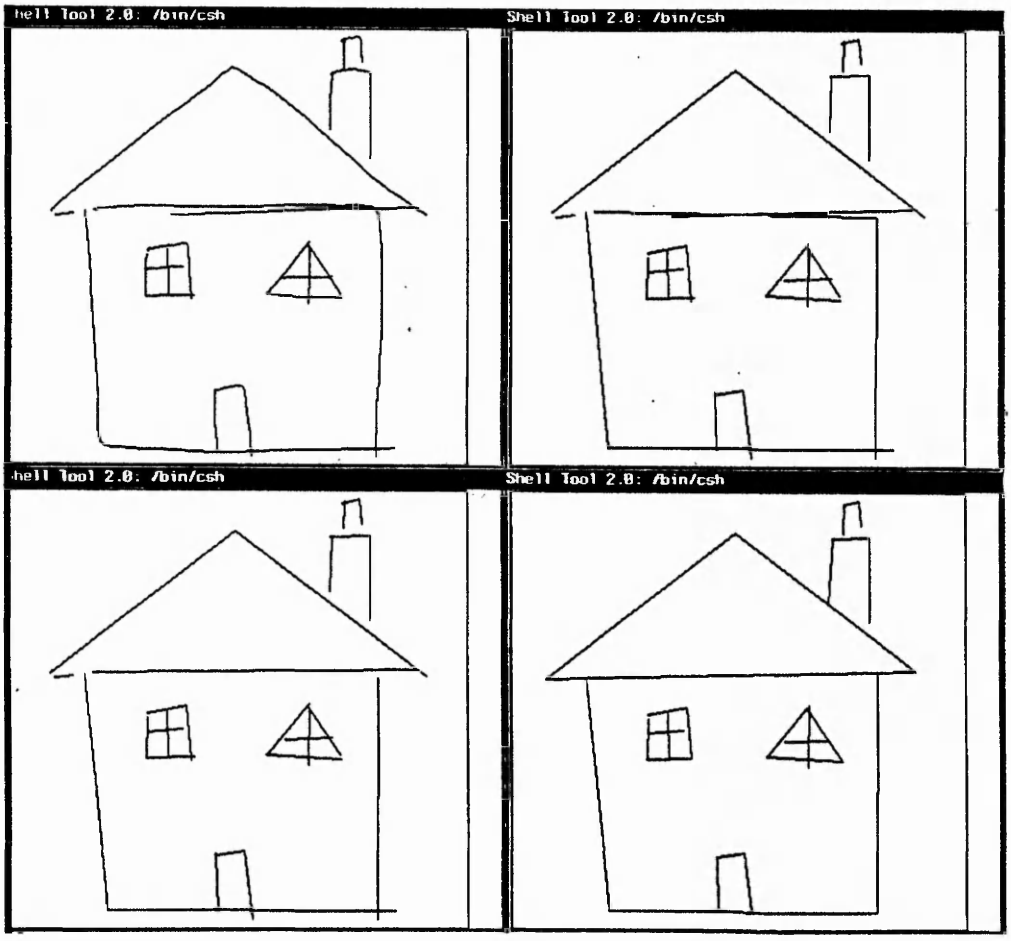
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

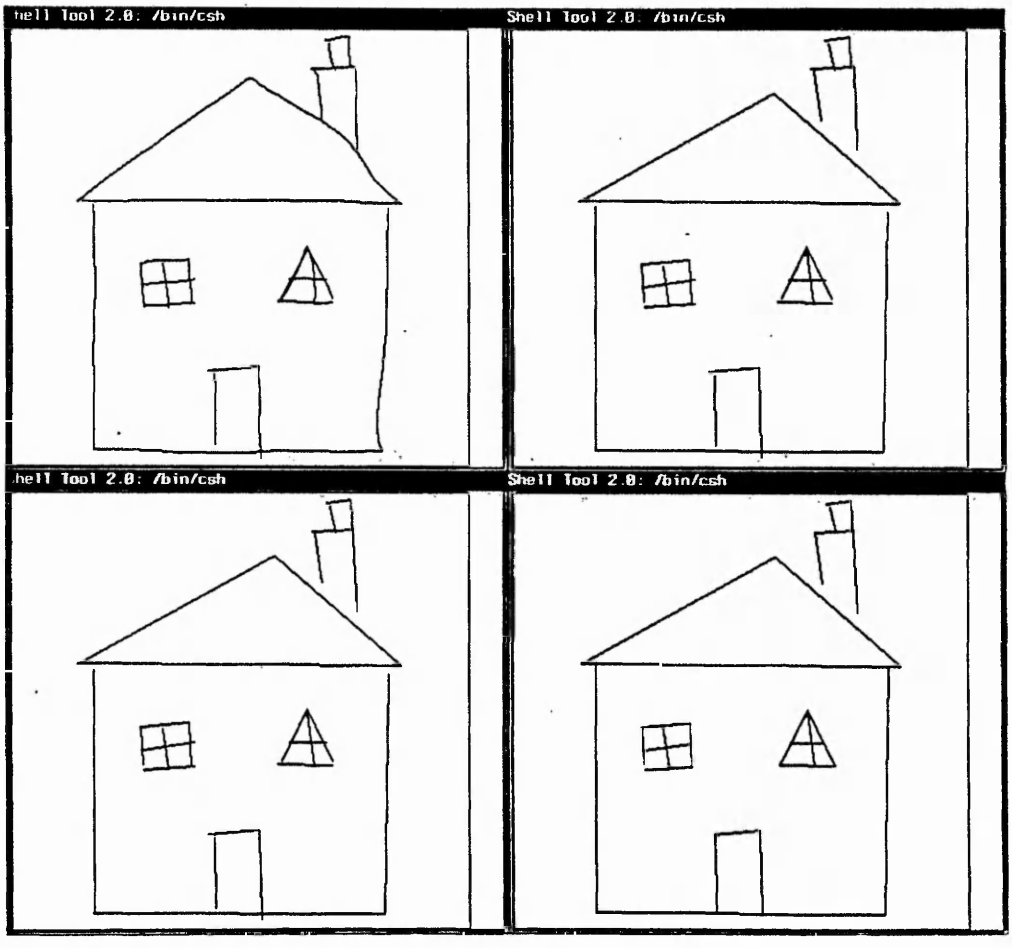
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

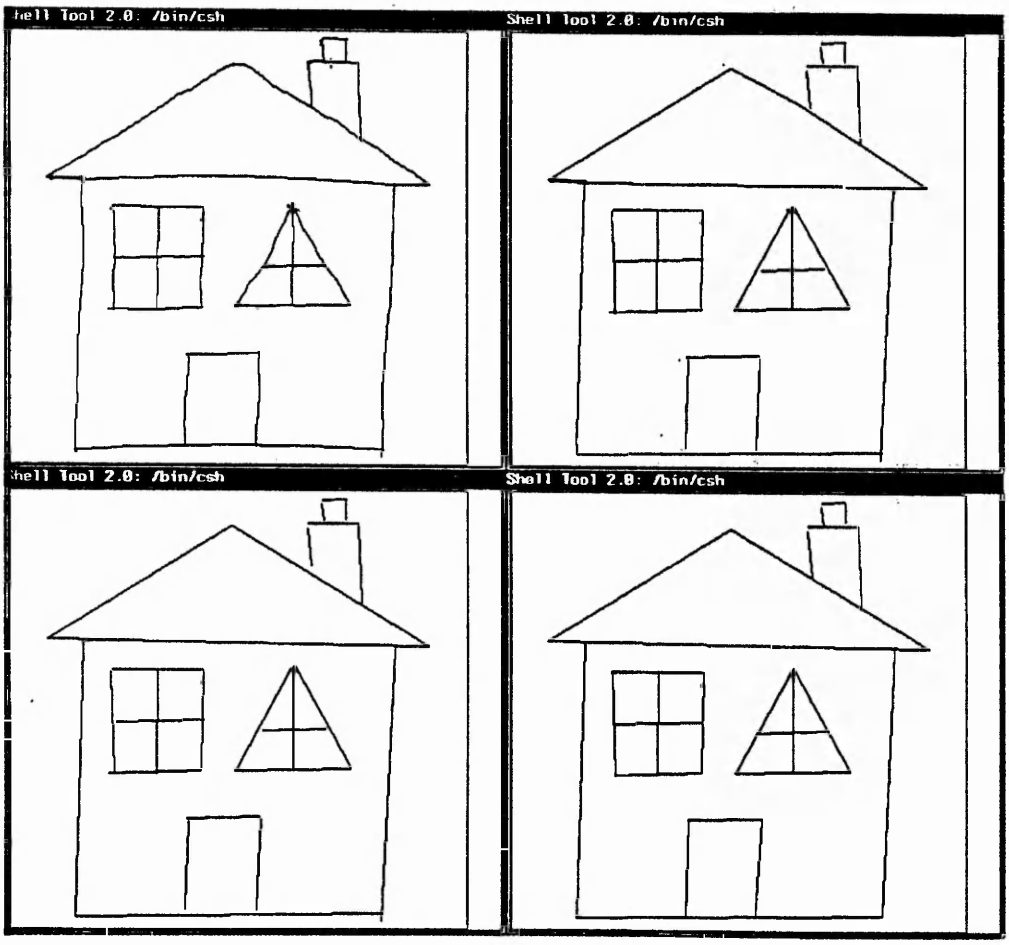
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

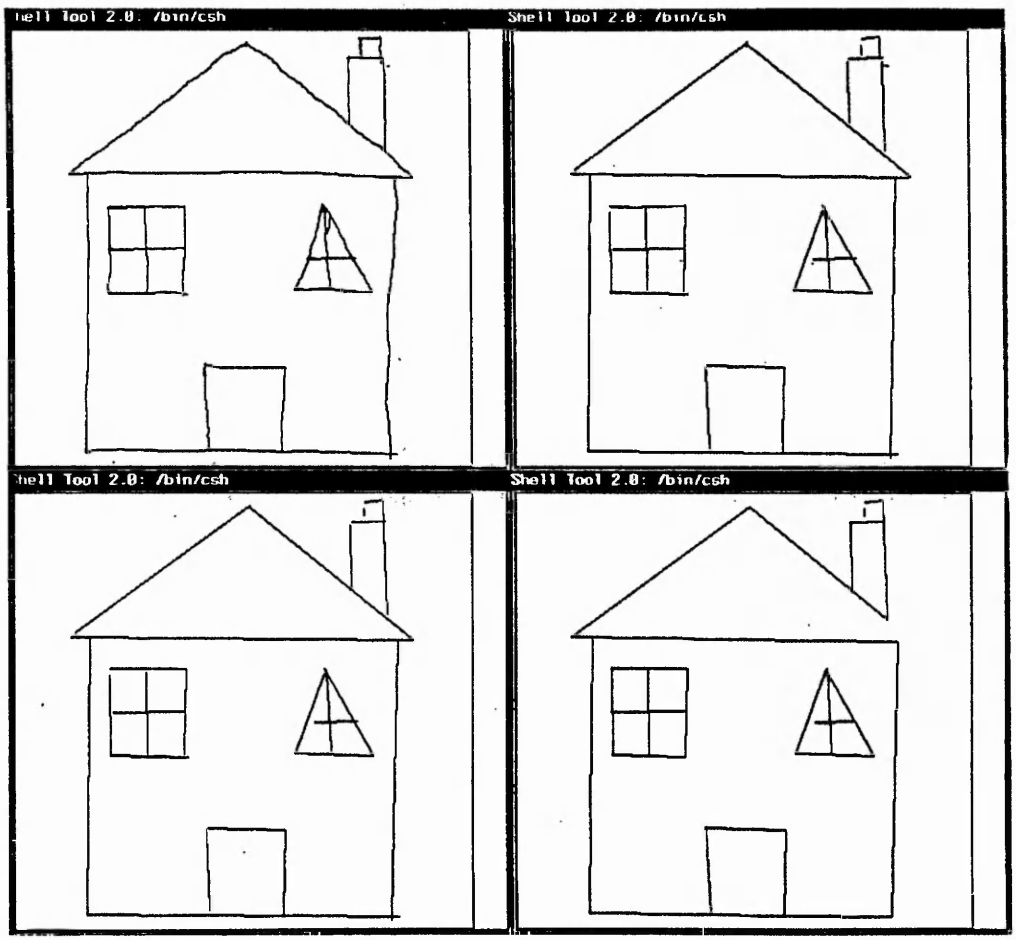
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

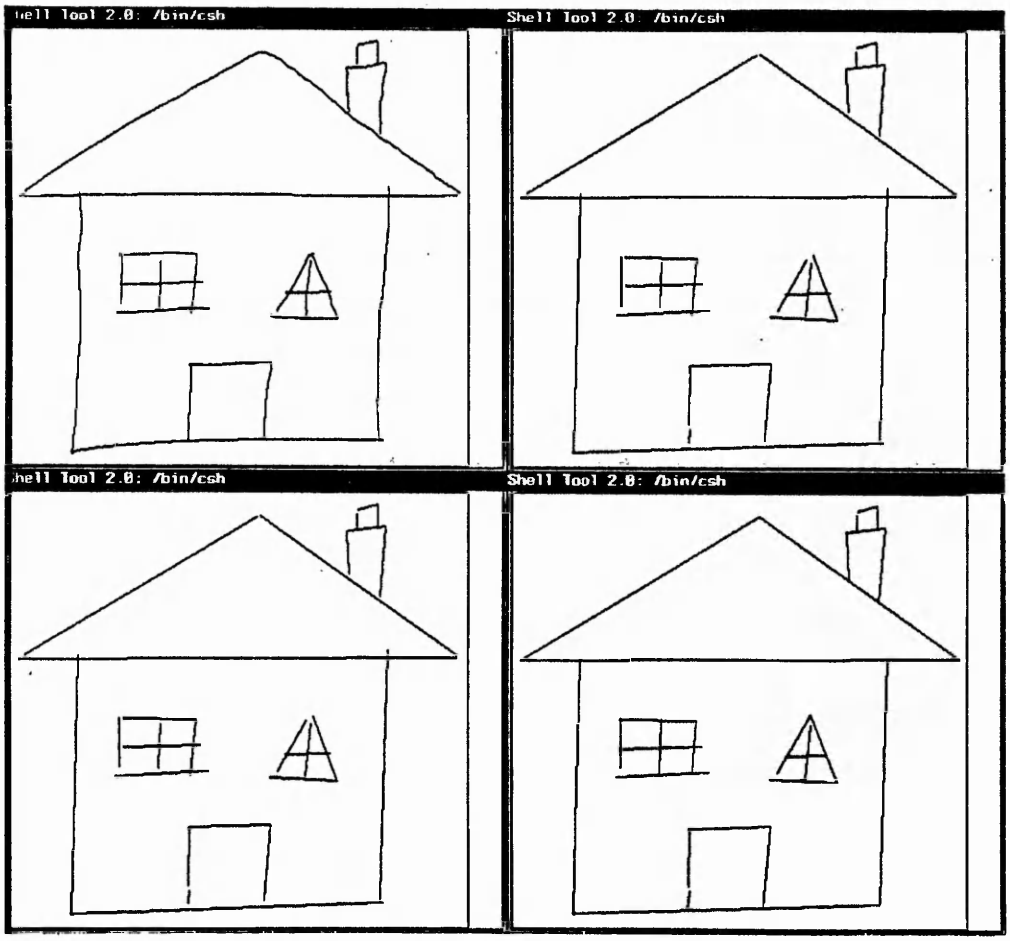
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

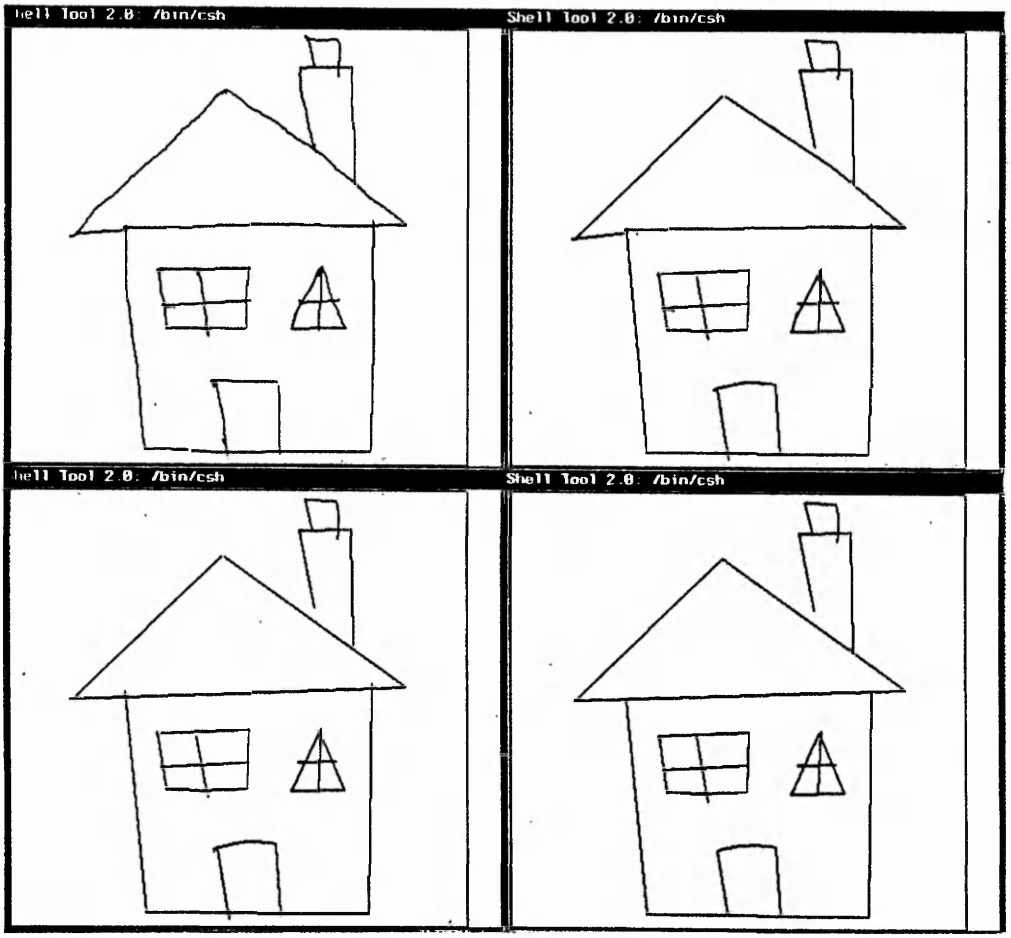
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

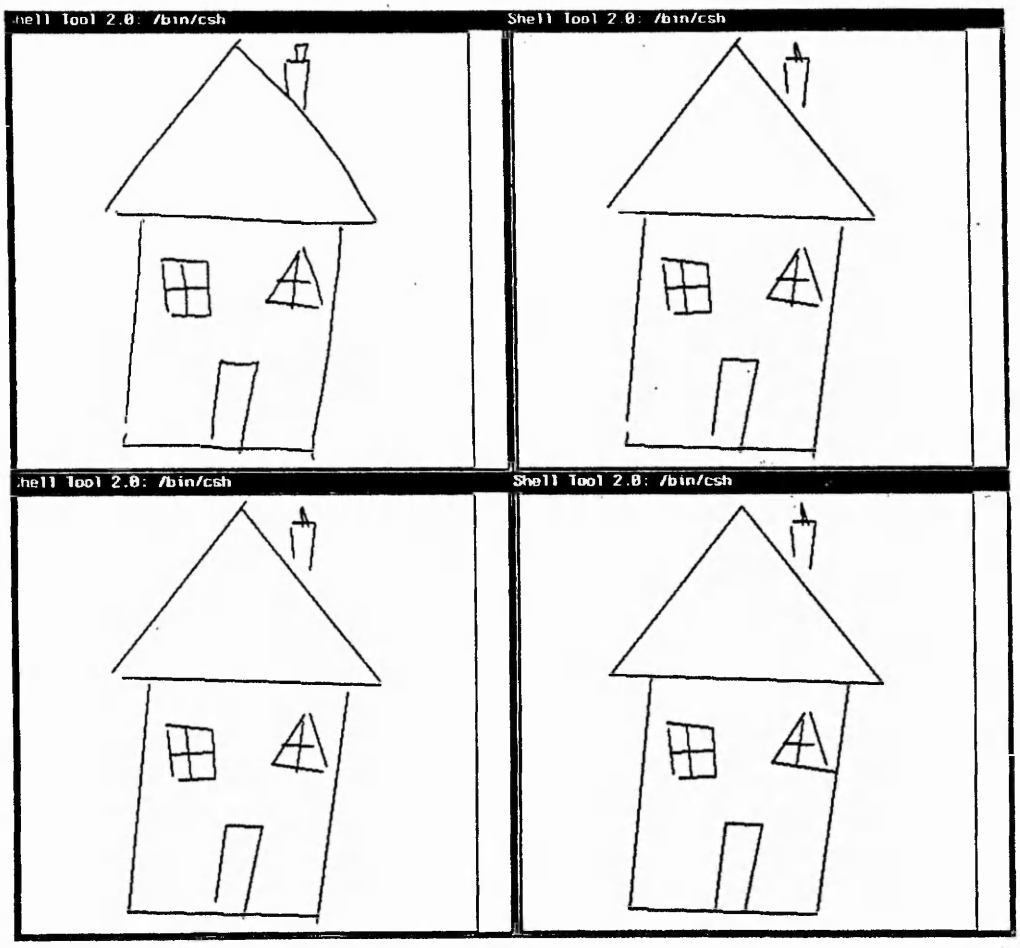
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

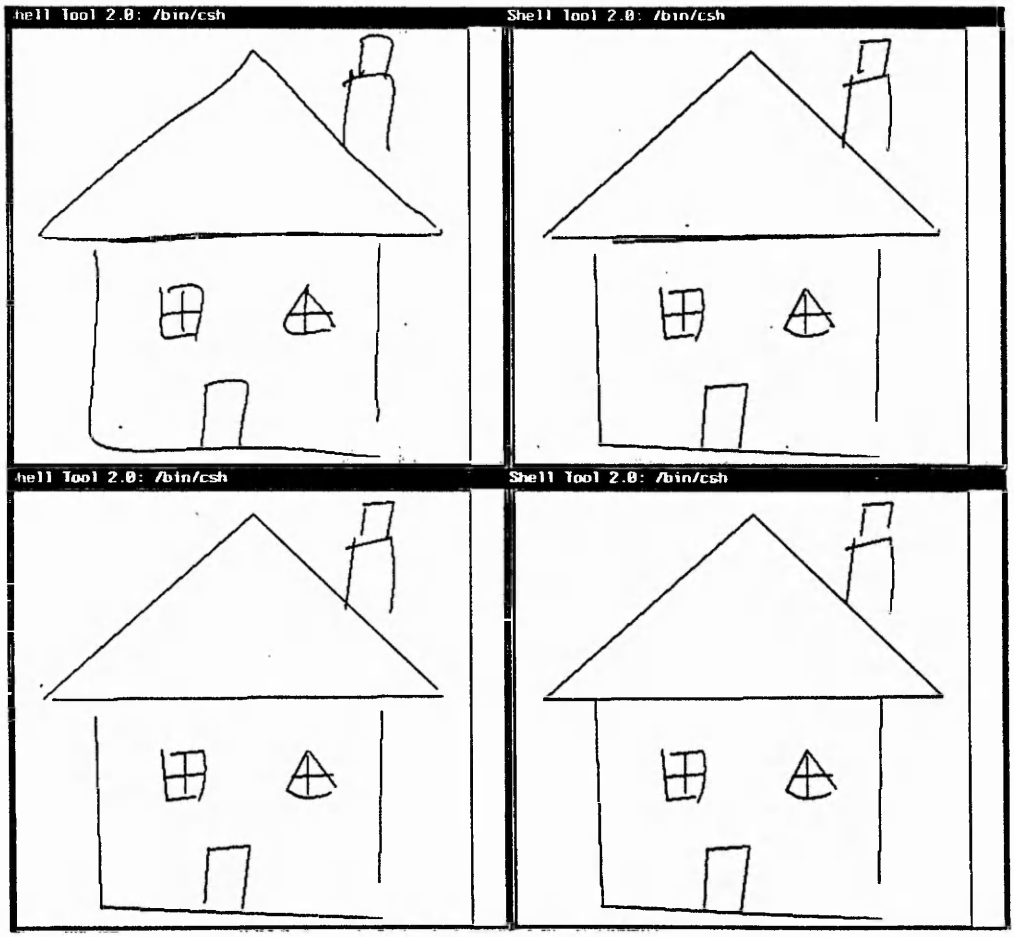
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

r s

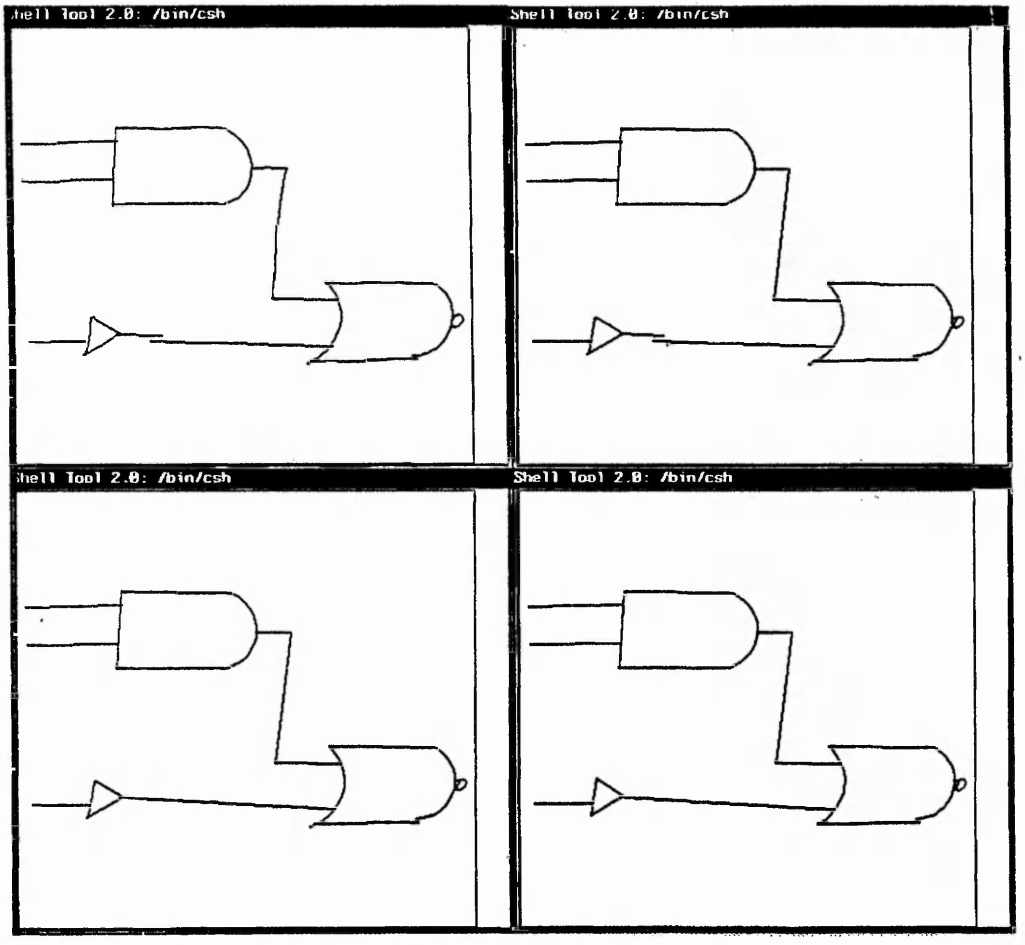
m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

B.4 Gate Results

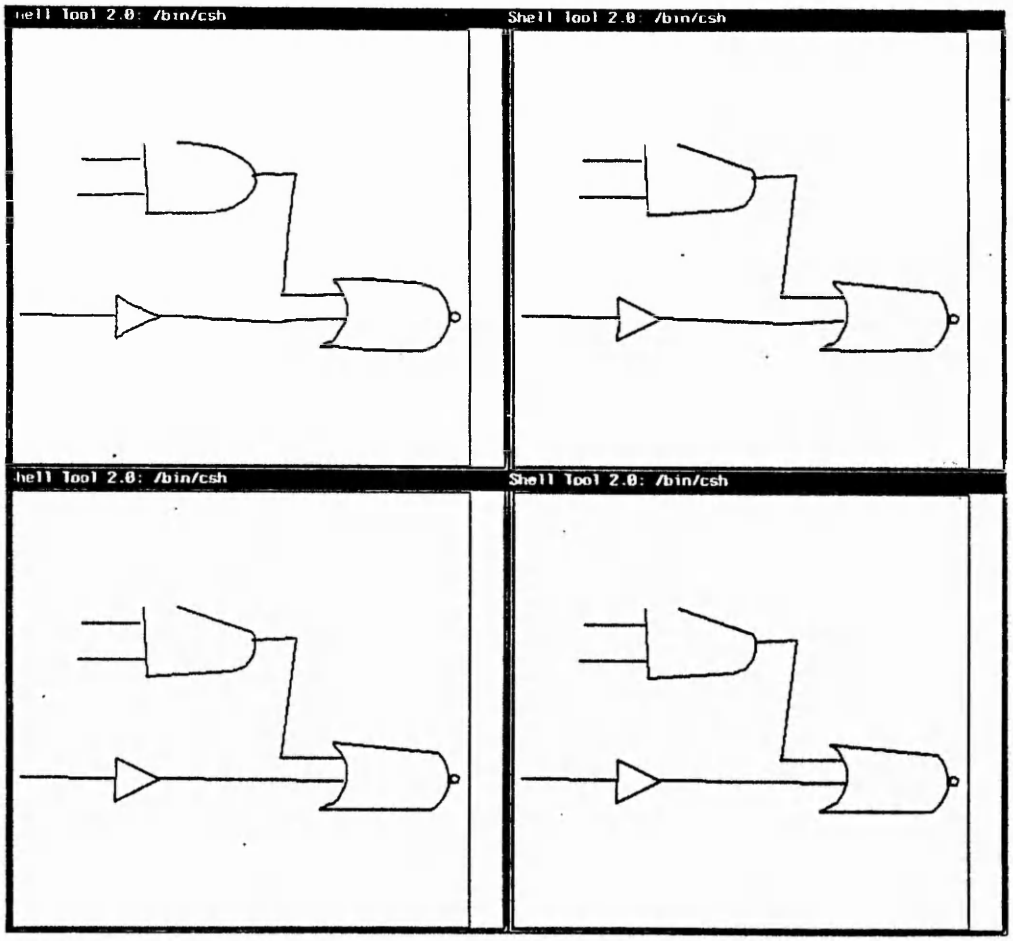
r s

m l



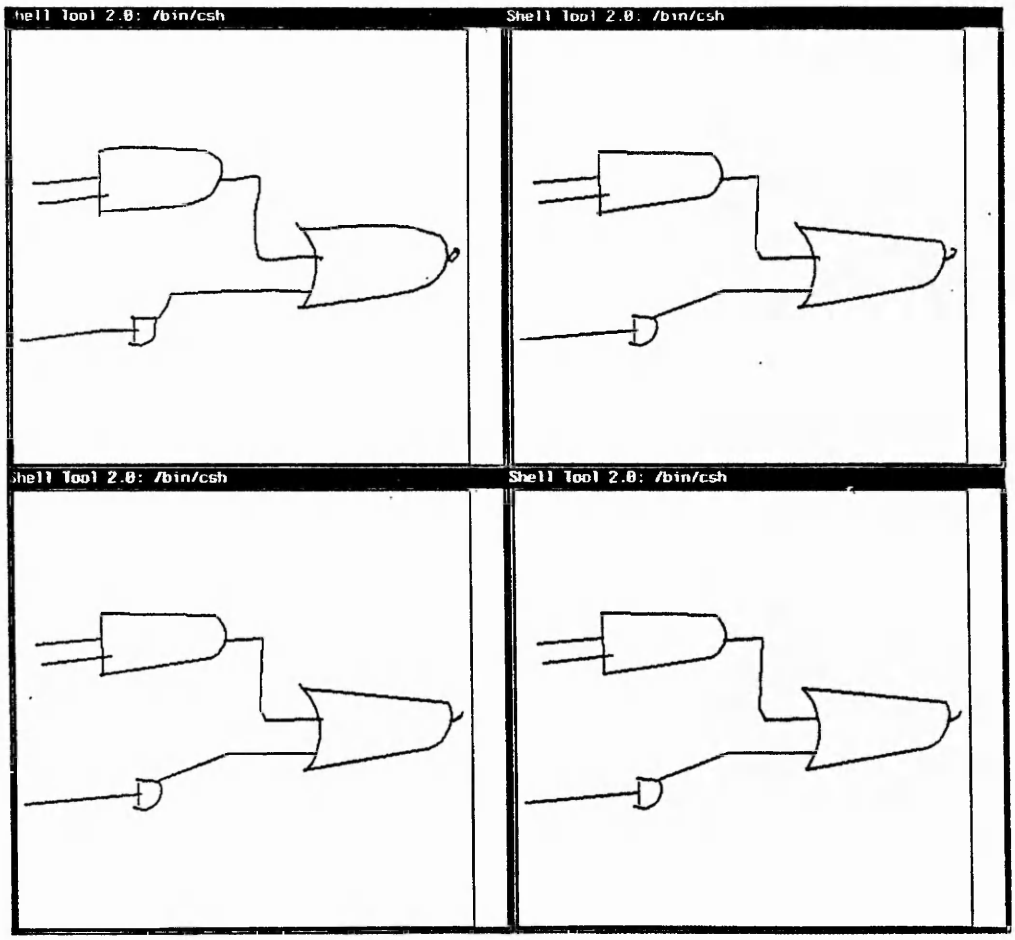
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

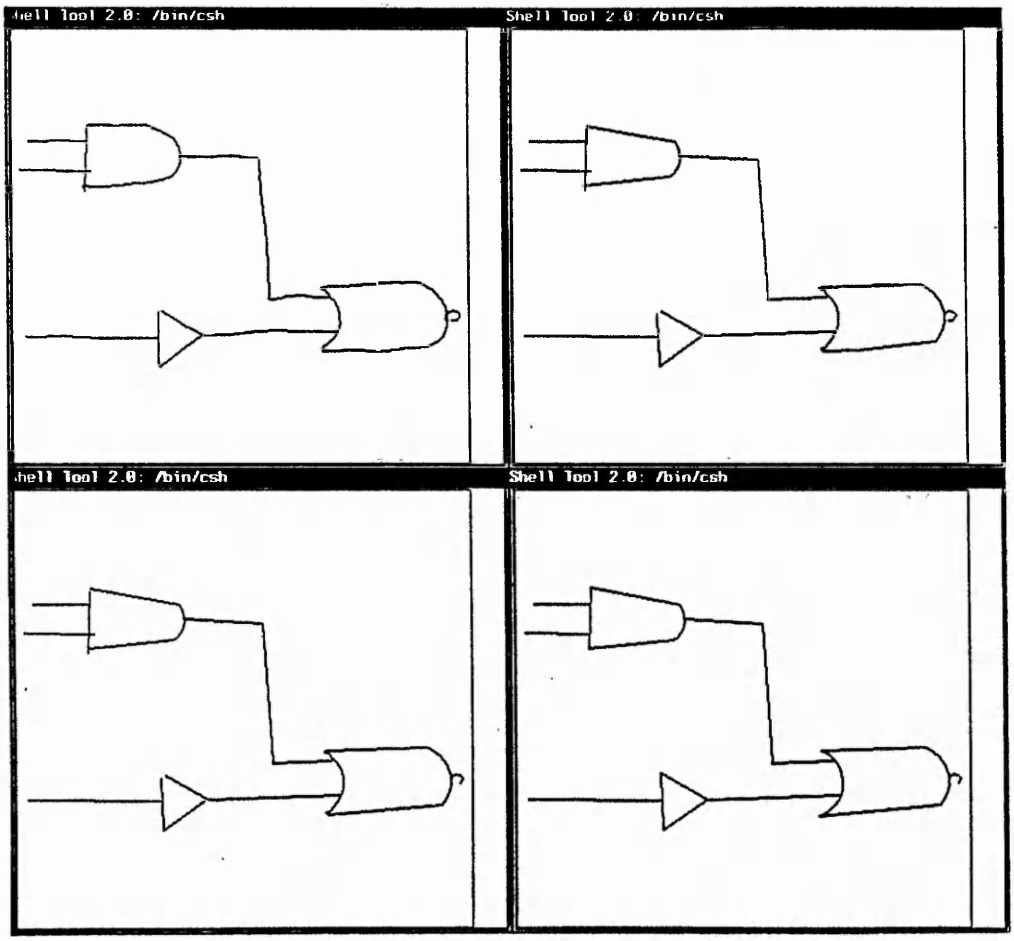
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

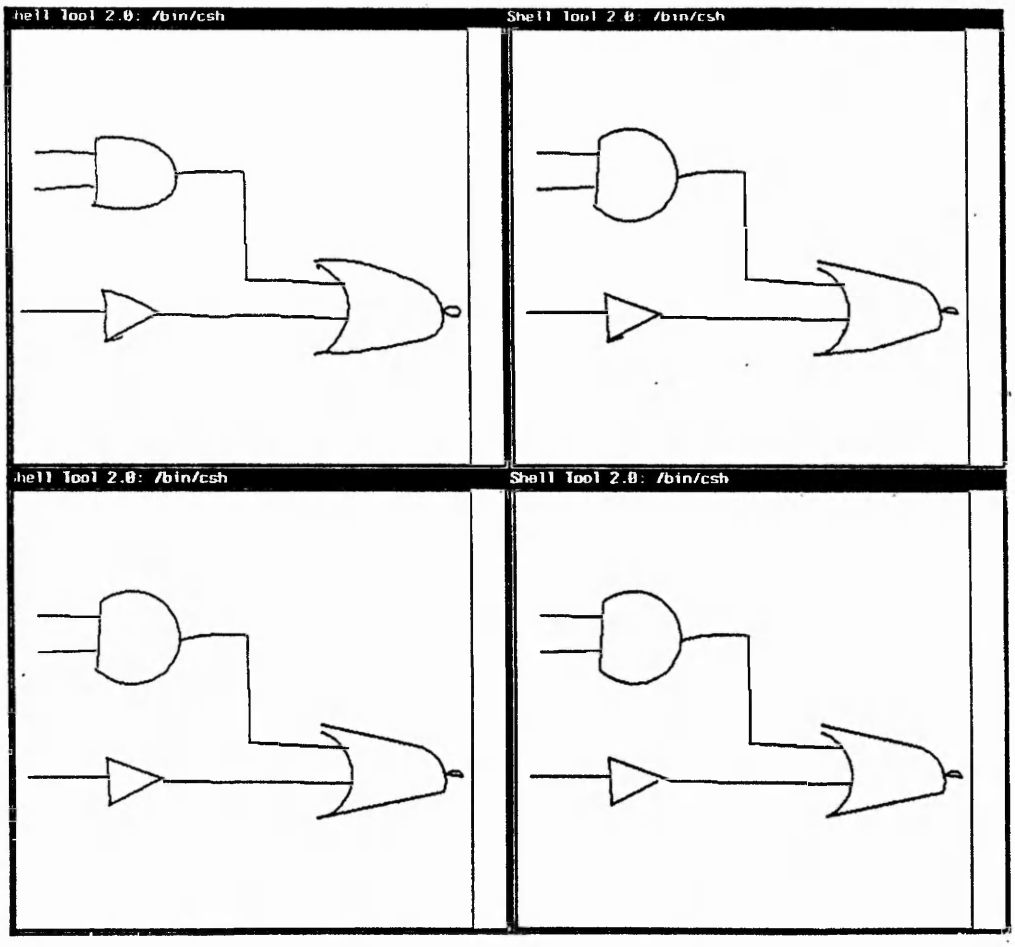
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

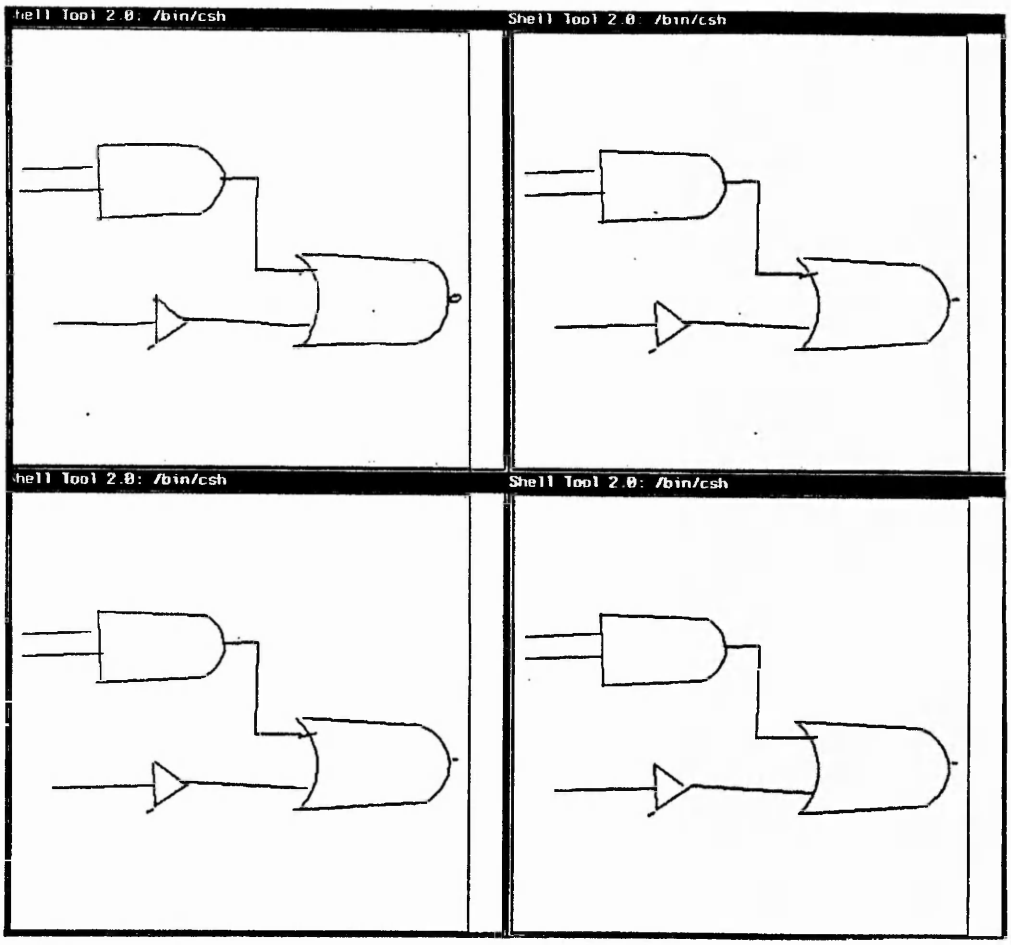
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

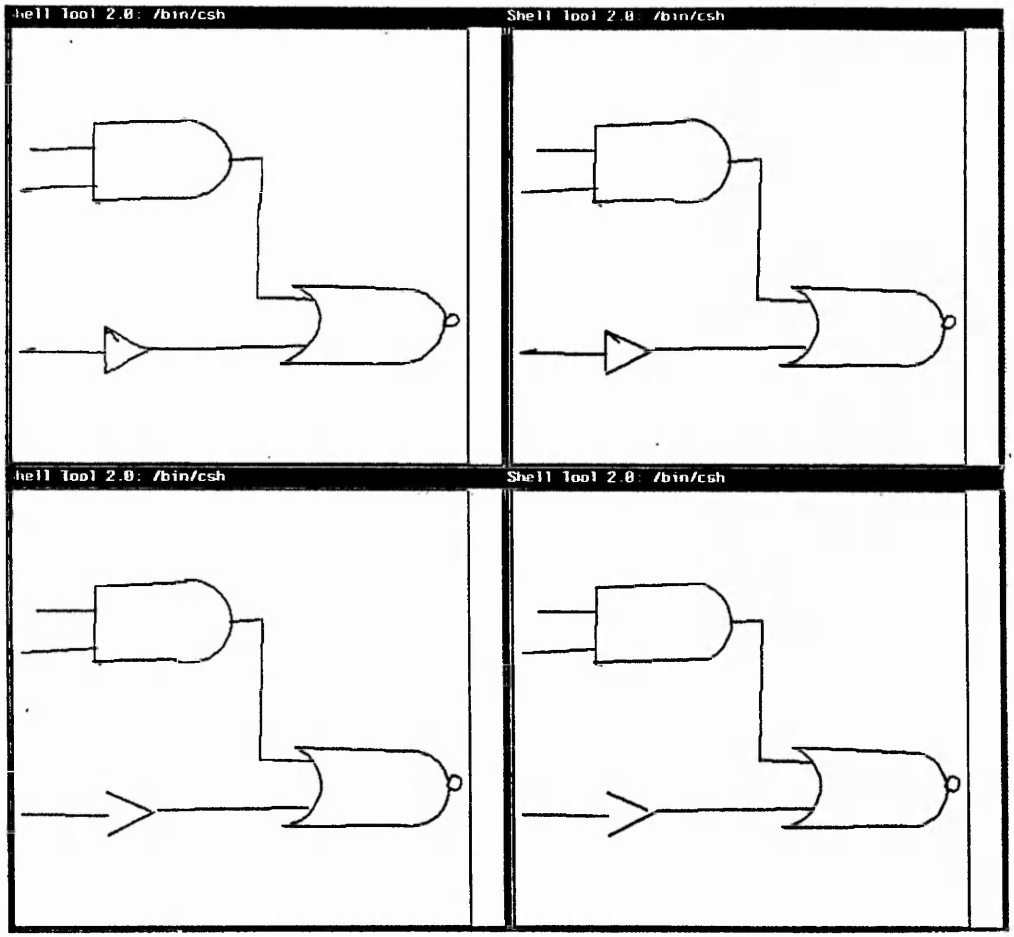
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

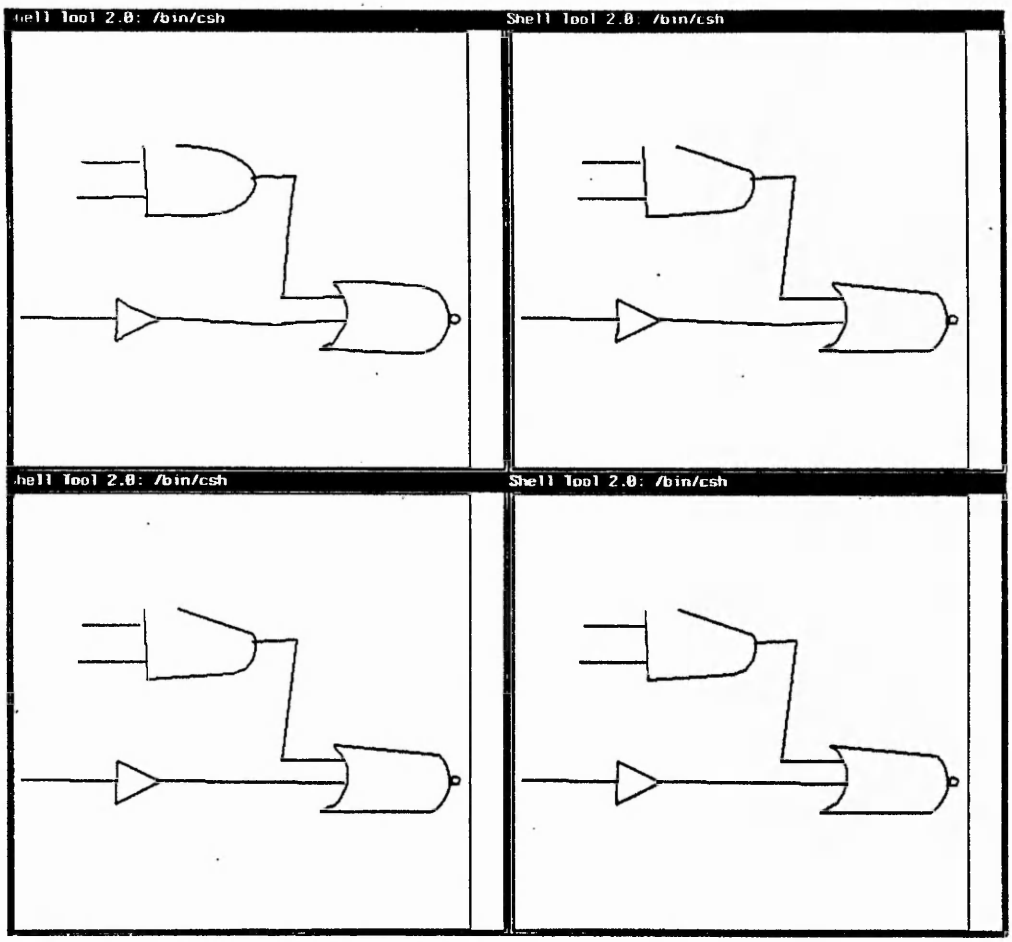
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

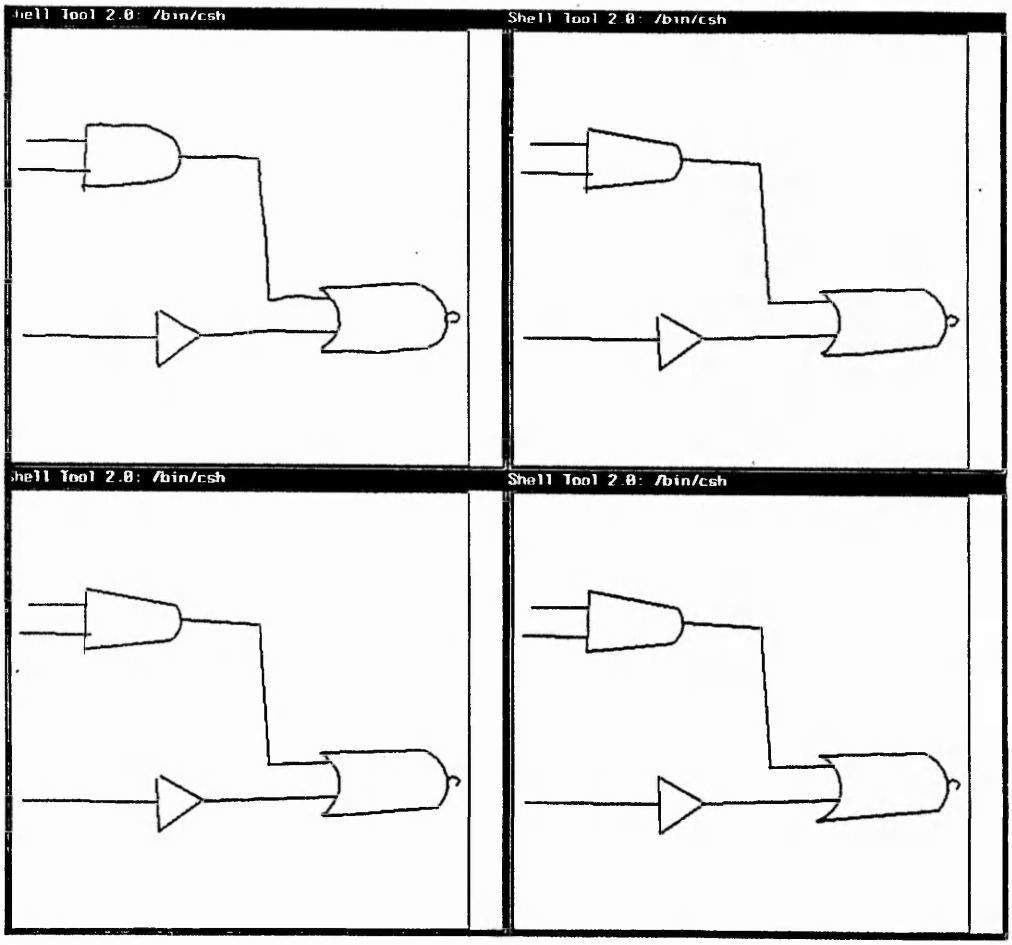
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

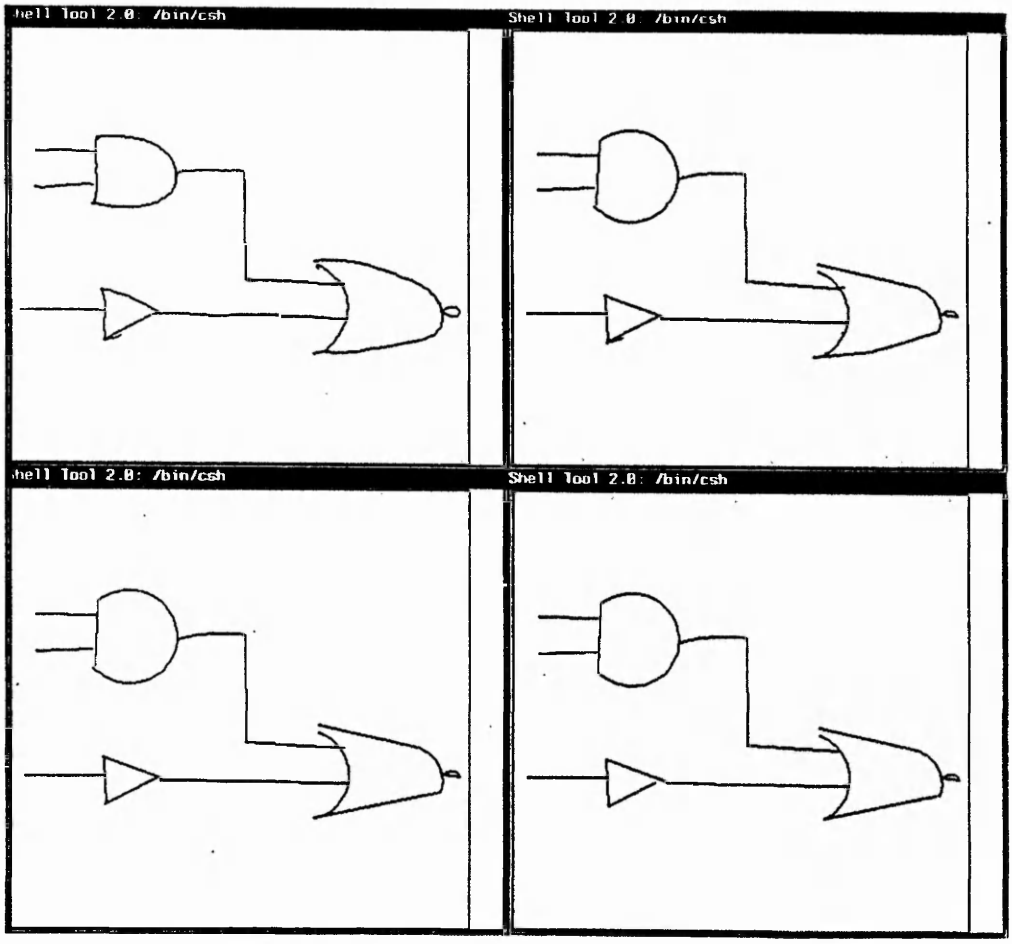
r s

m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

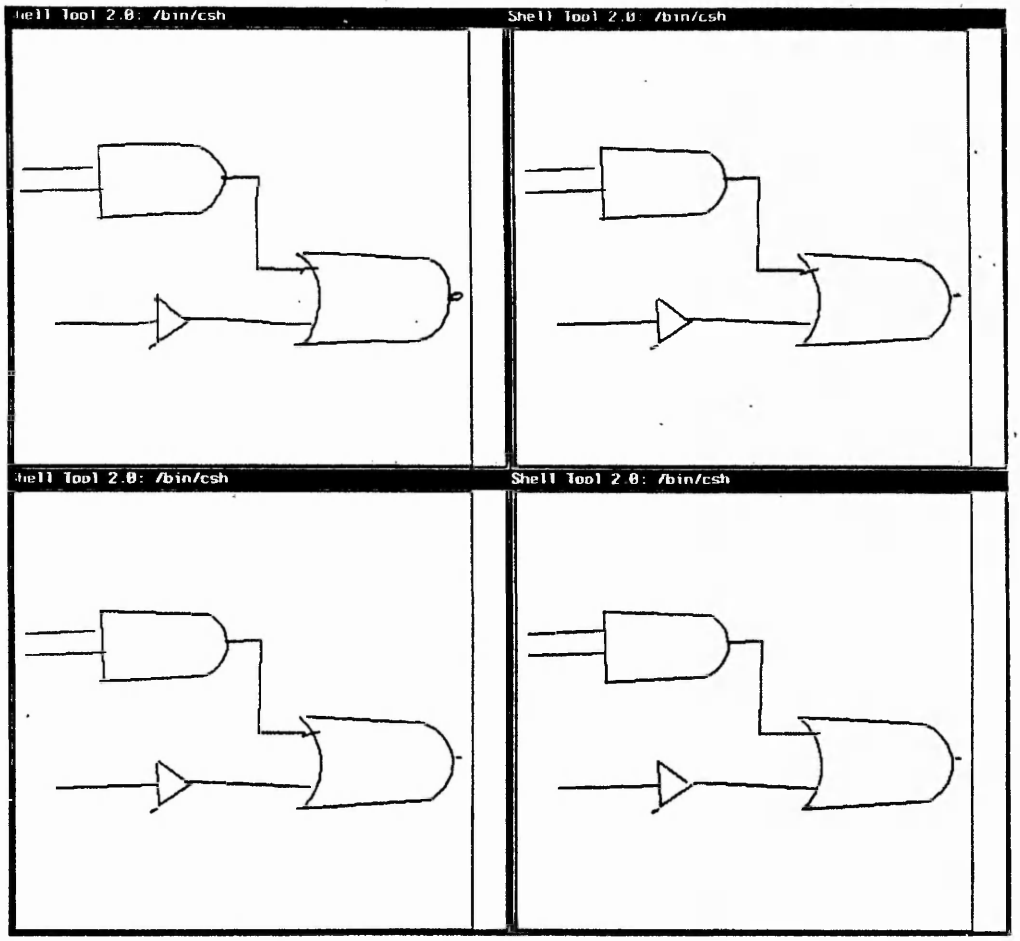
r s

m l


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

r s

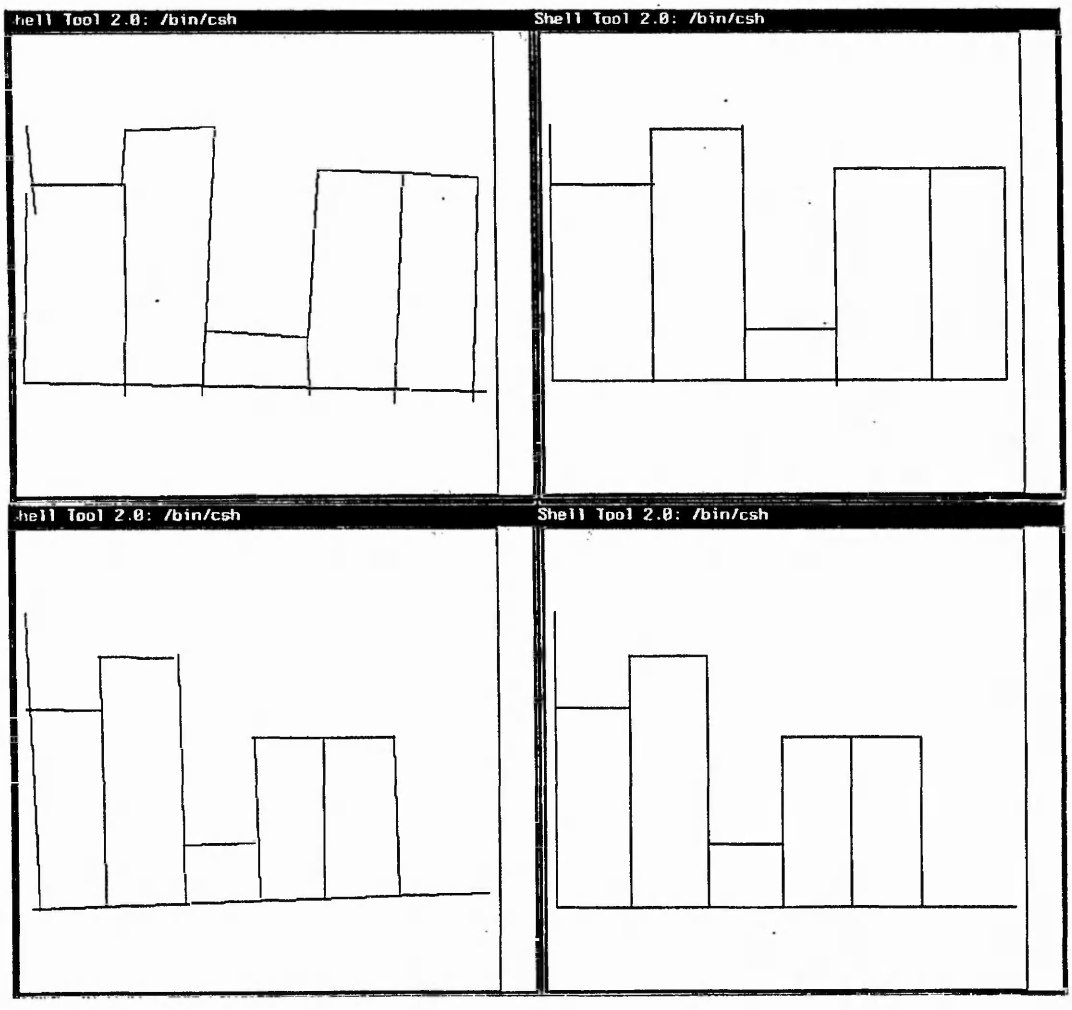
m l


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

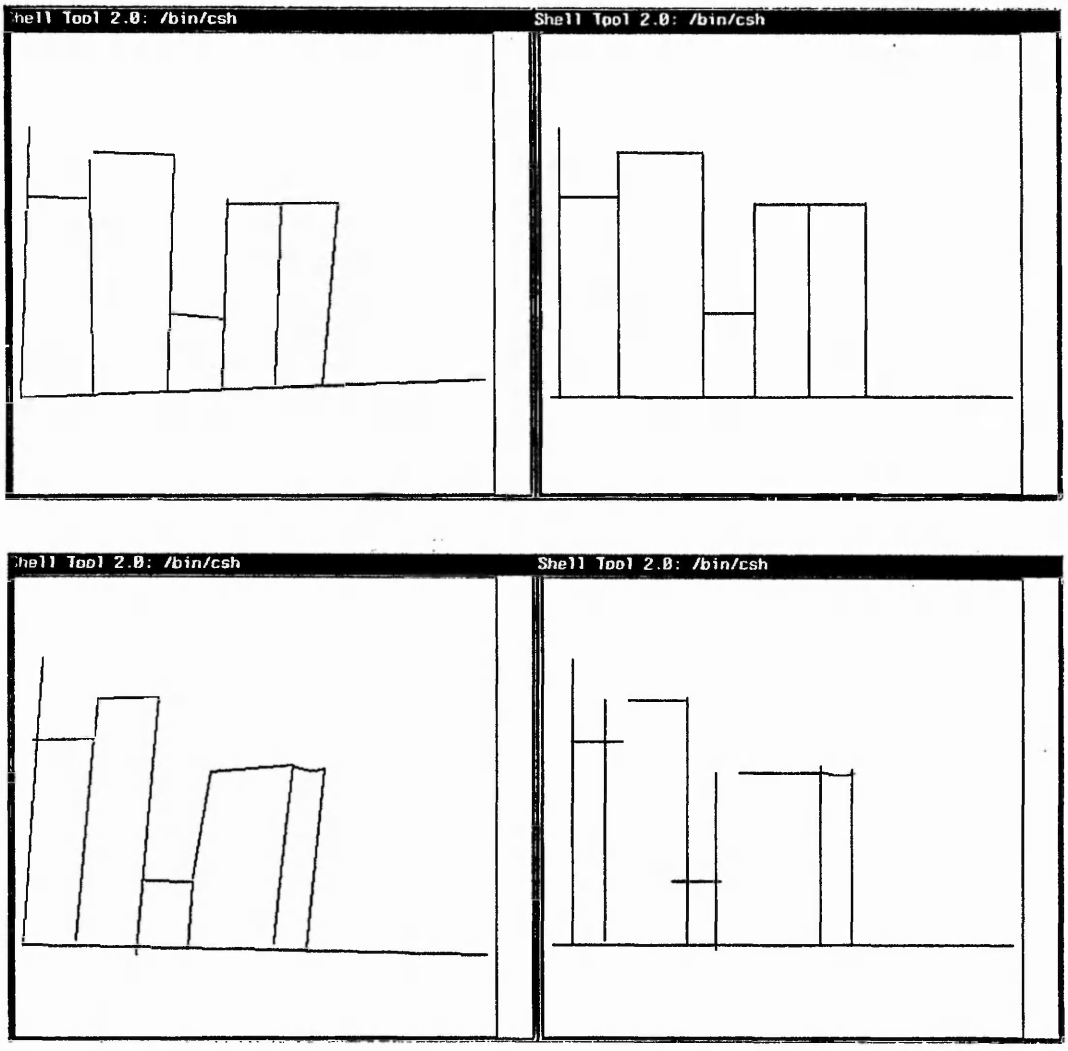
B.5 Results of Aligning

In the remaining pages sketches are shown in two stages of recognition, segmented on the left, then after merging, latching, and aligning on the right.

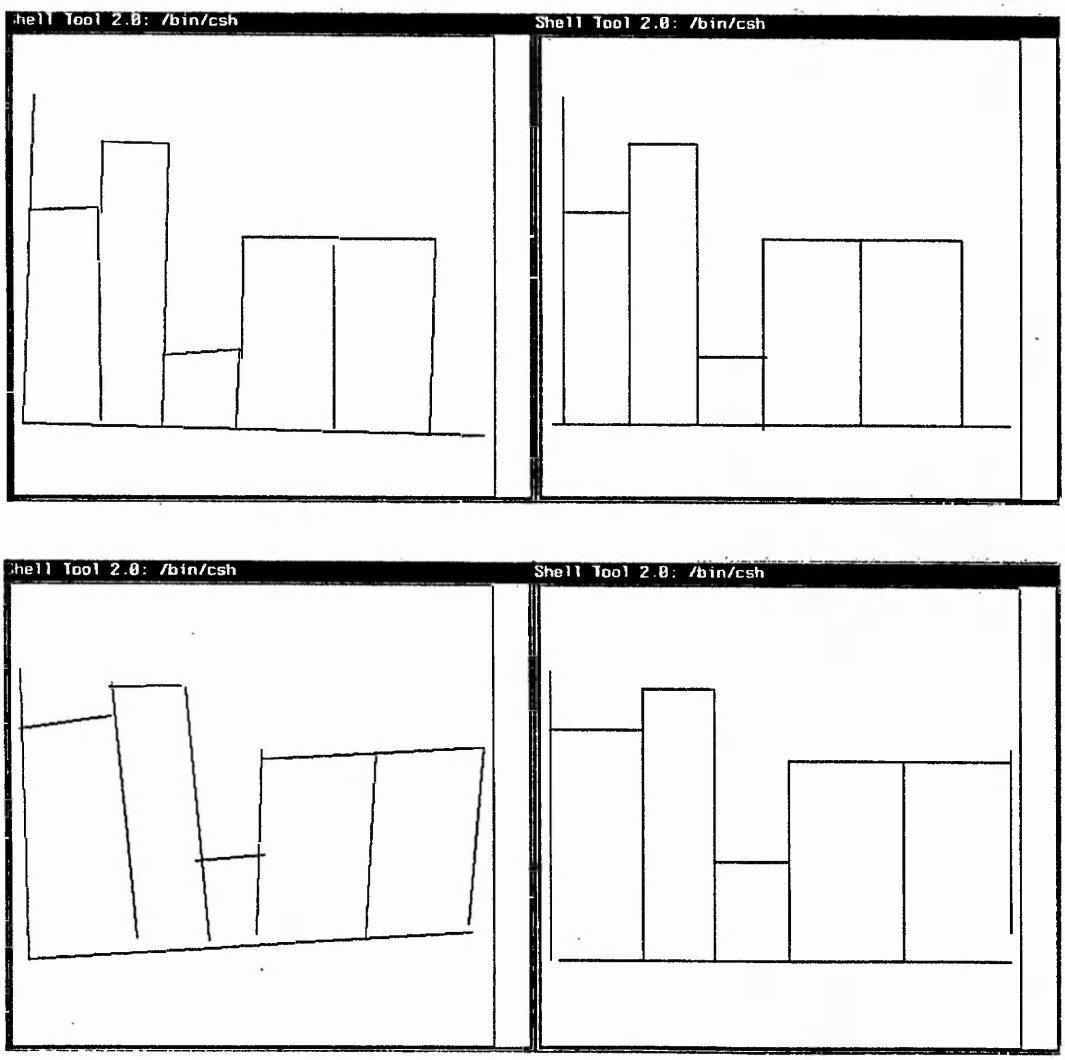
s a



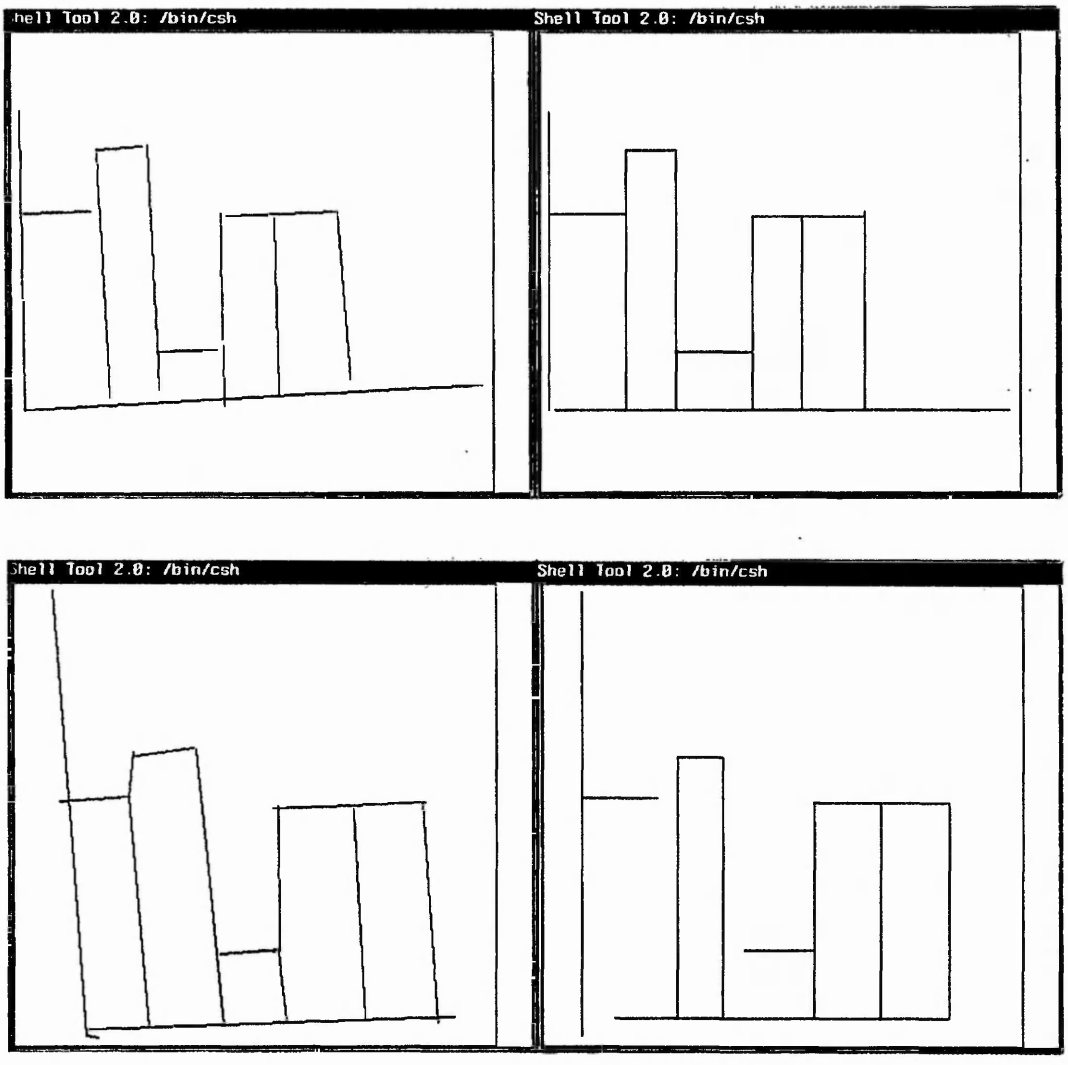
s a


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

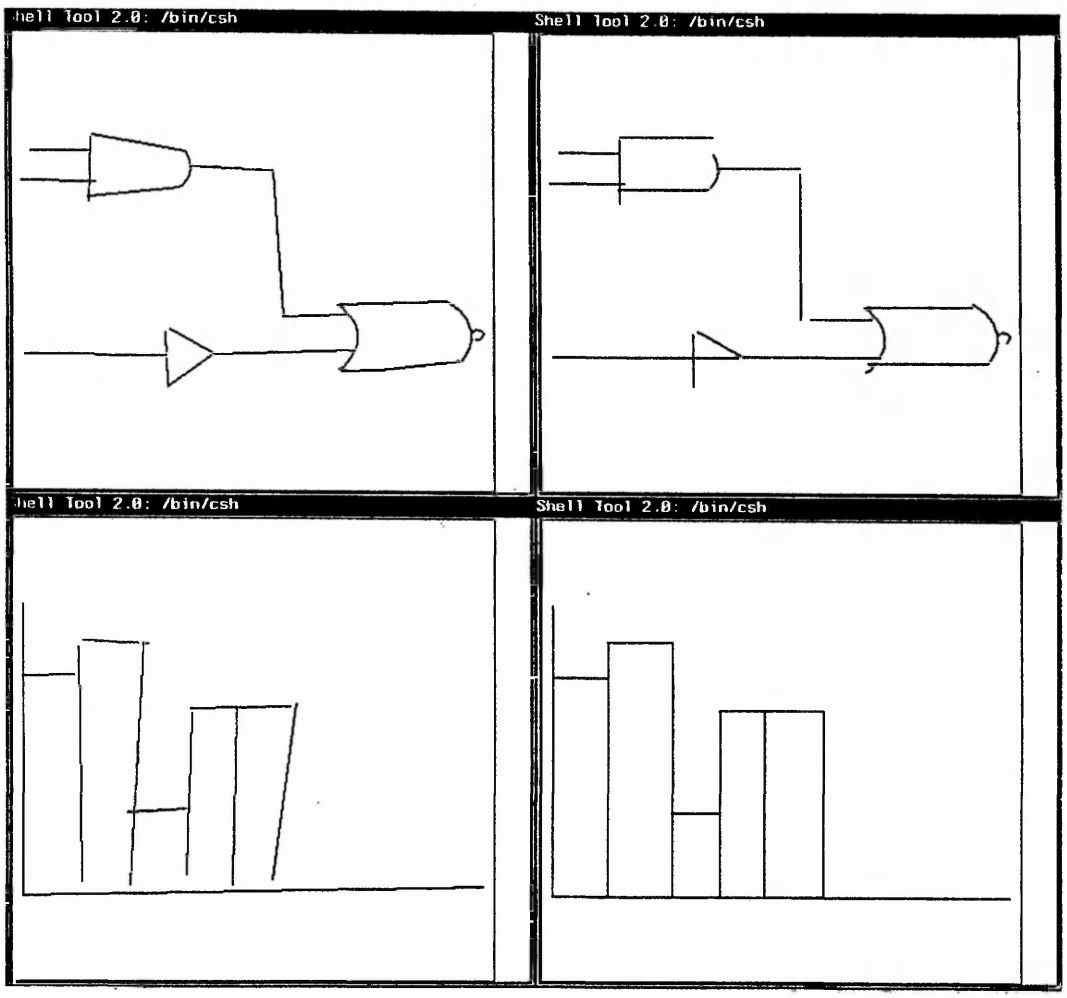
s a


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

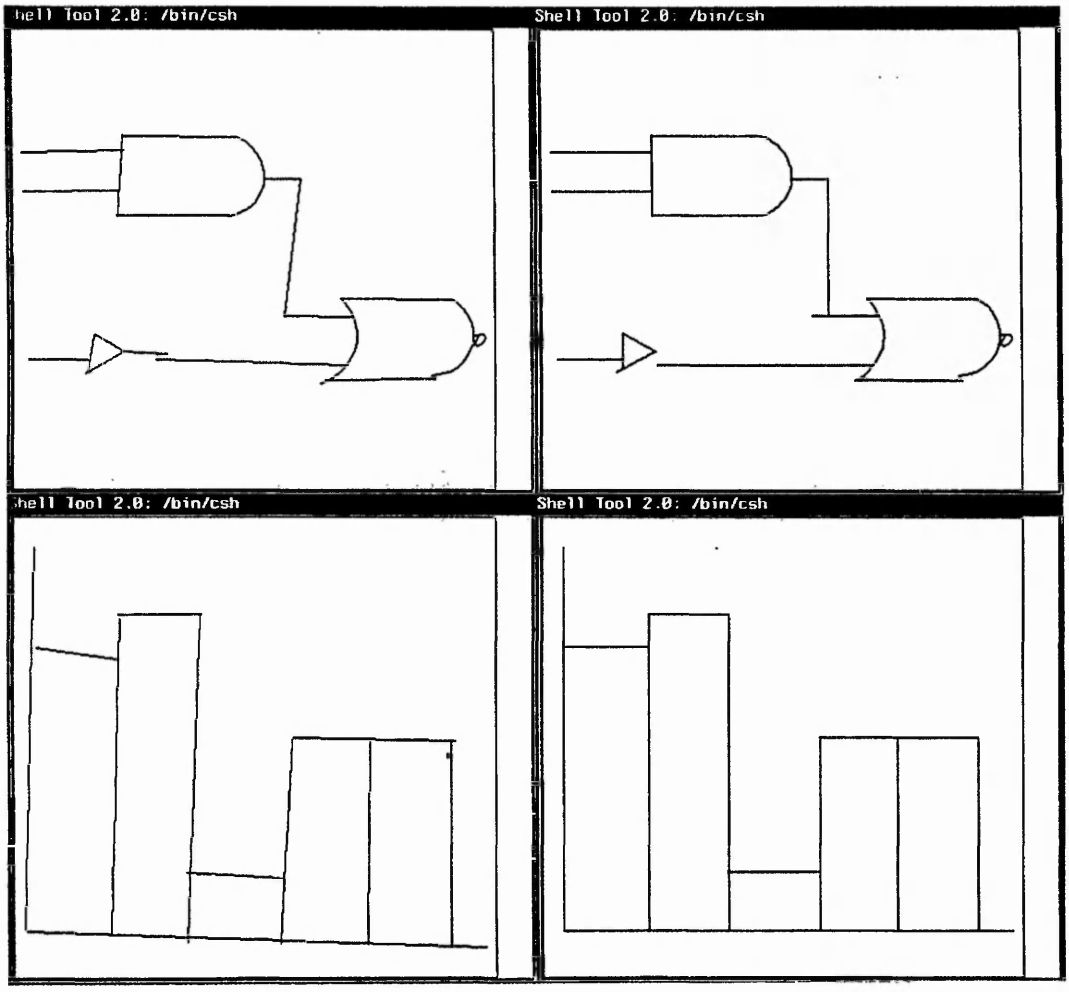
s a


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

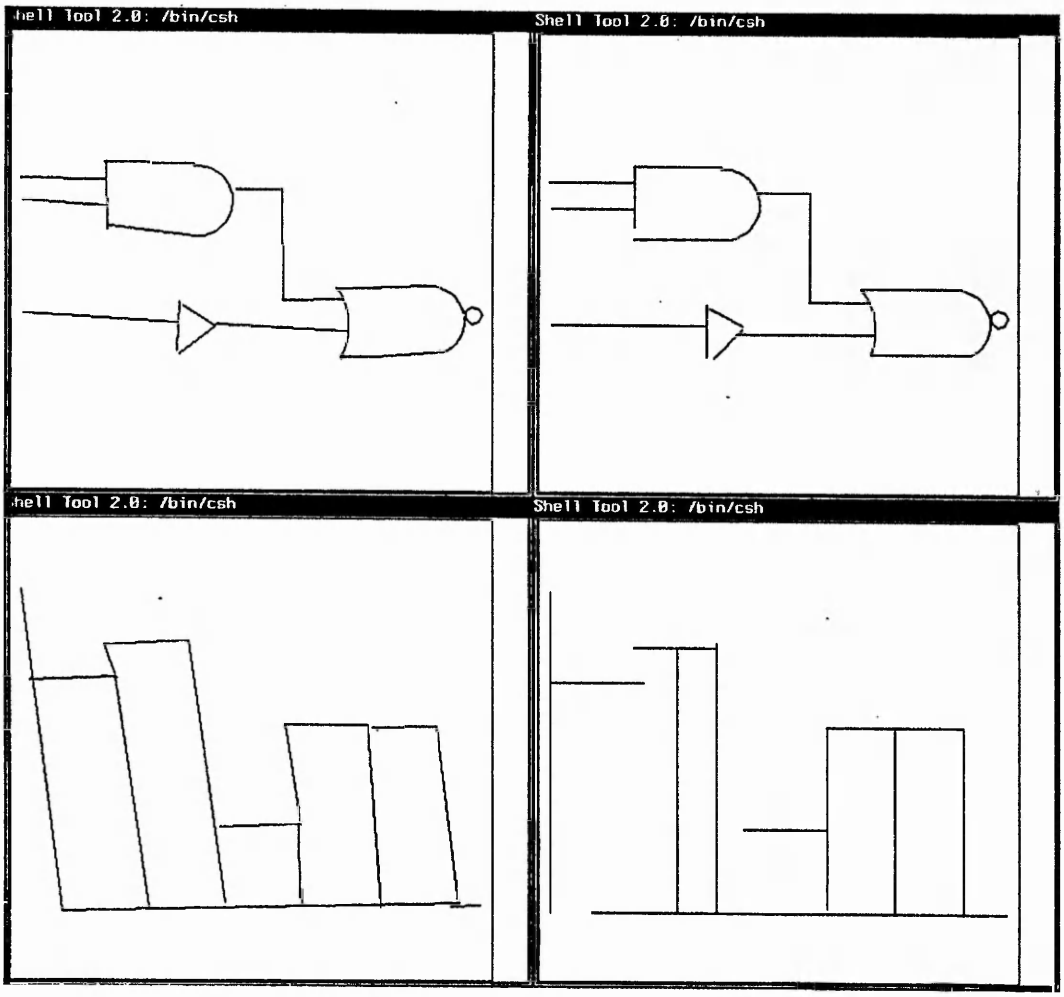
s a


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

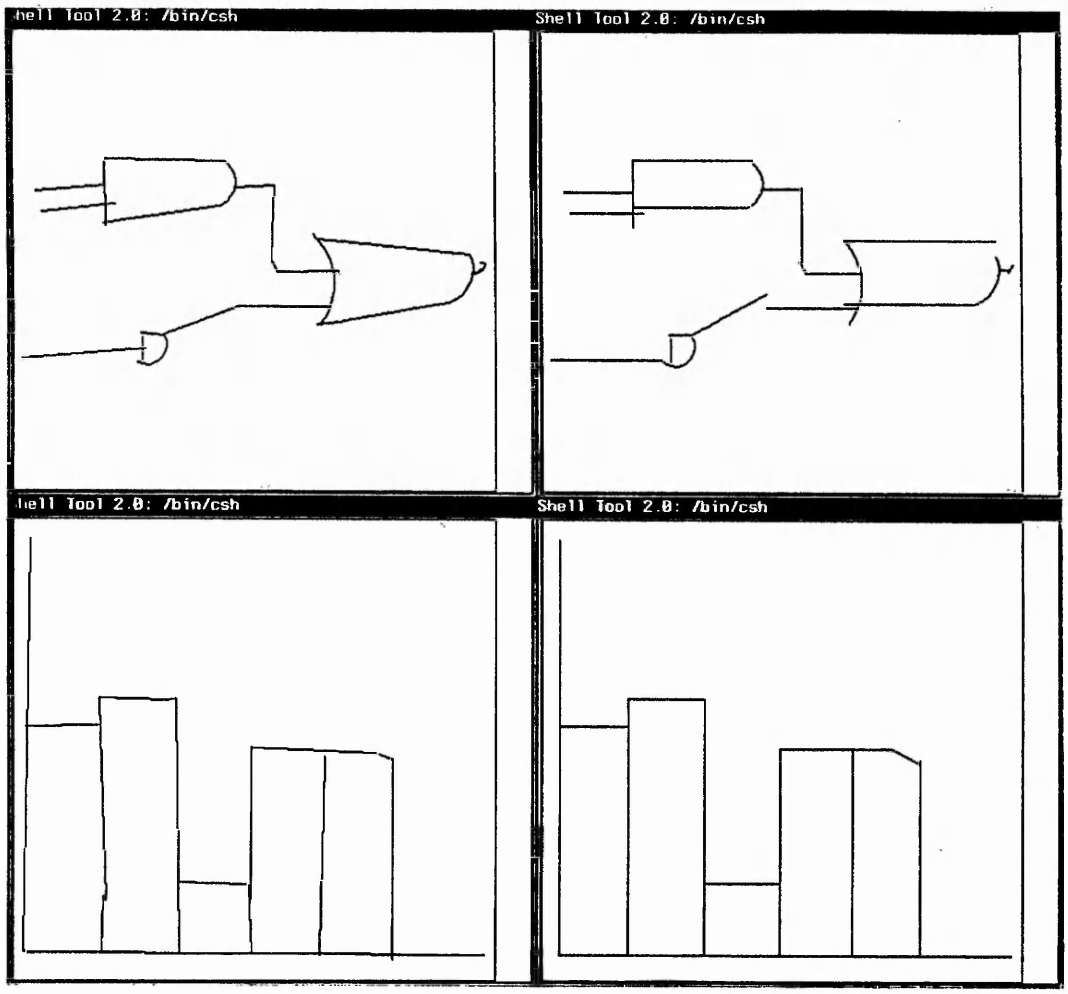
s a


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

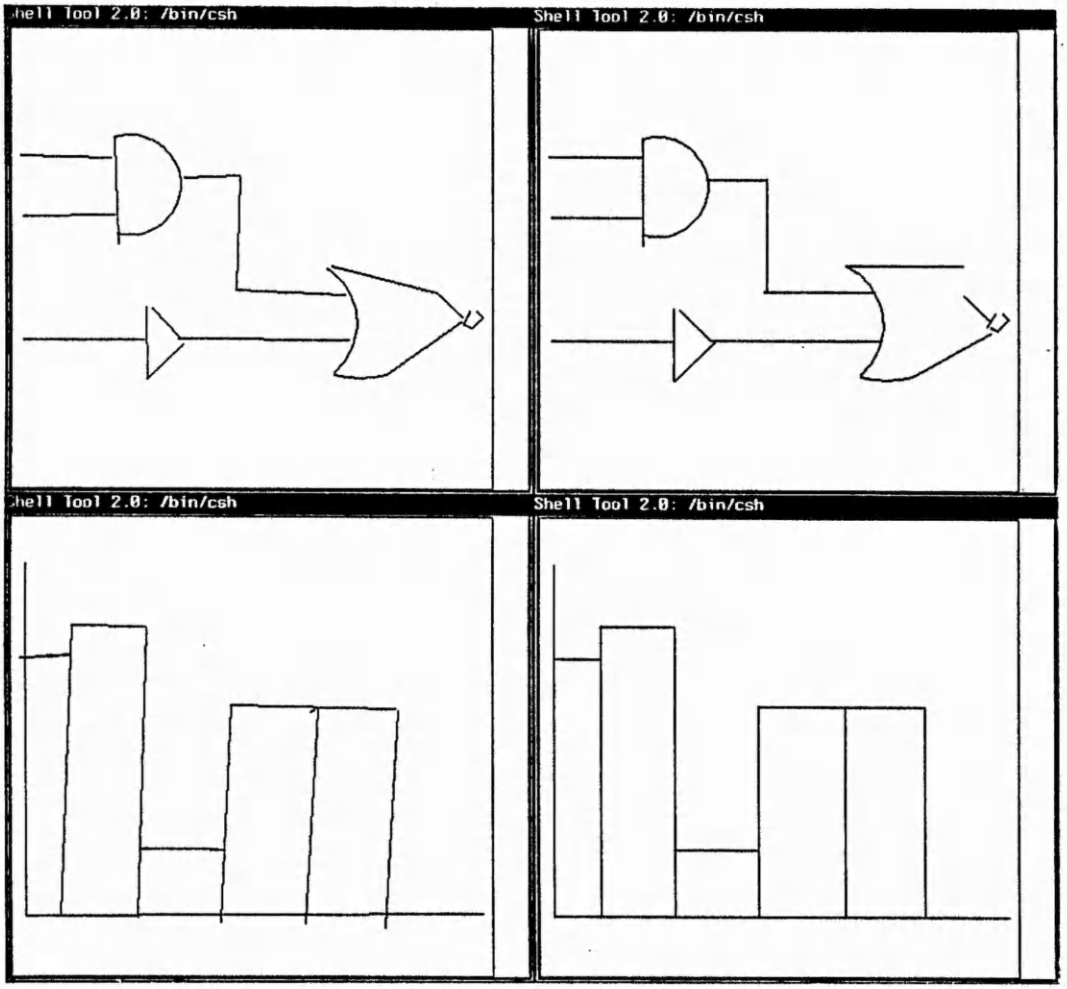
s a


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

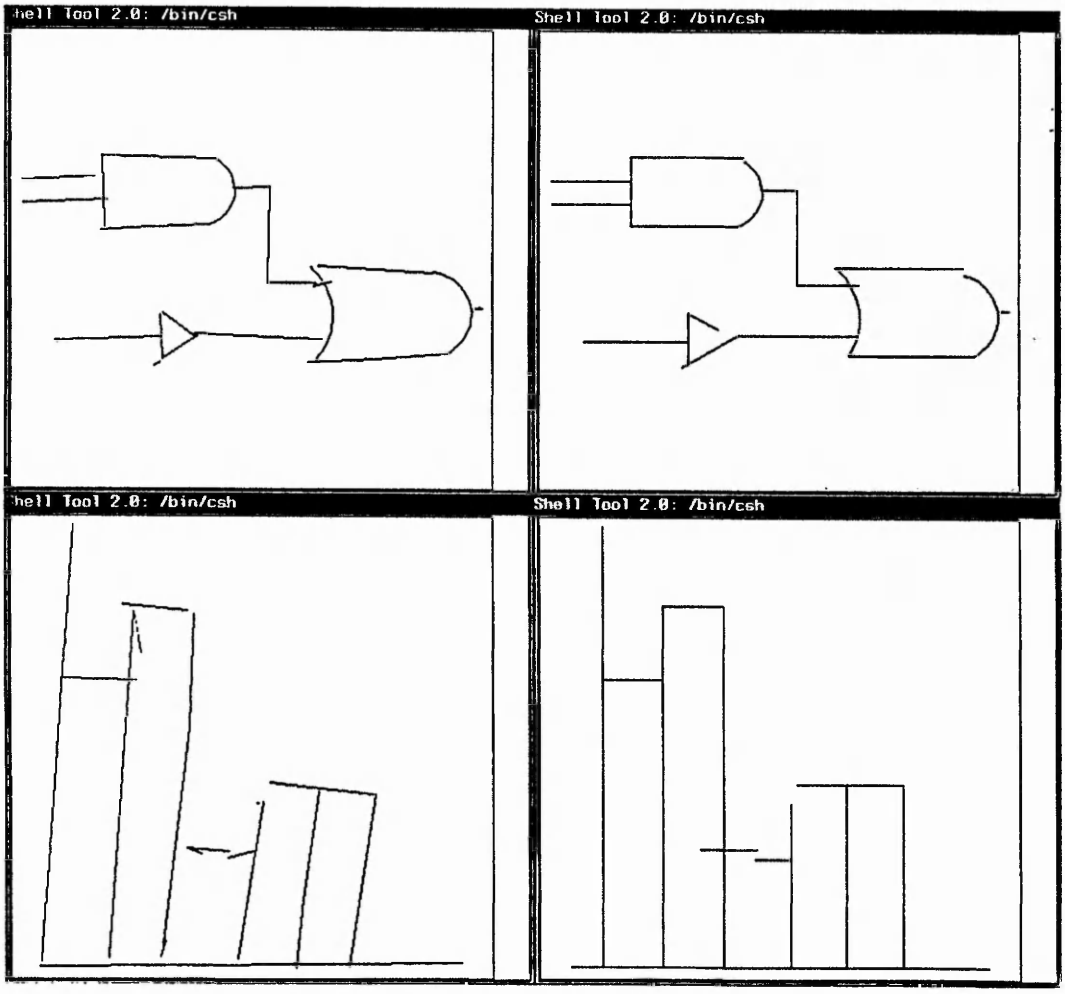
s a


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

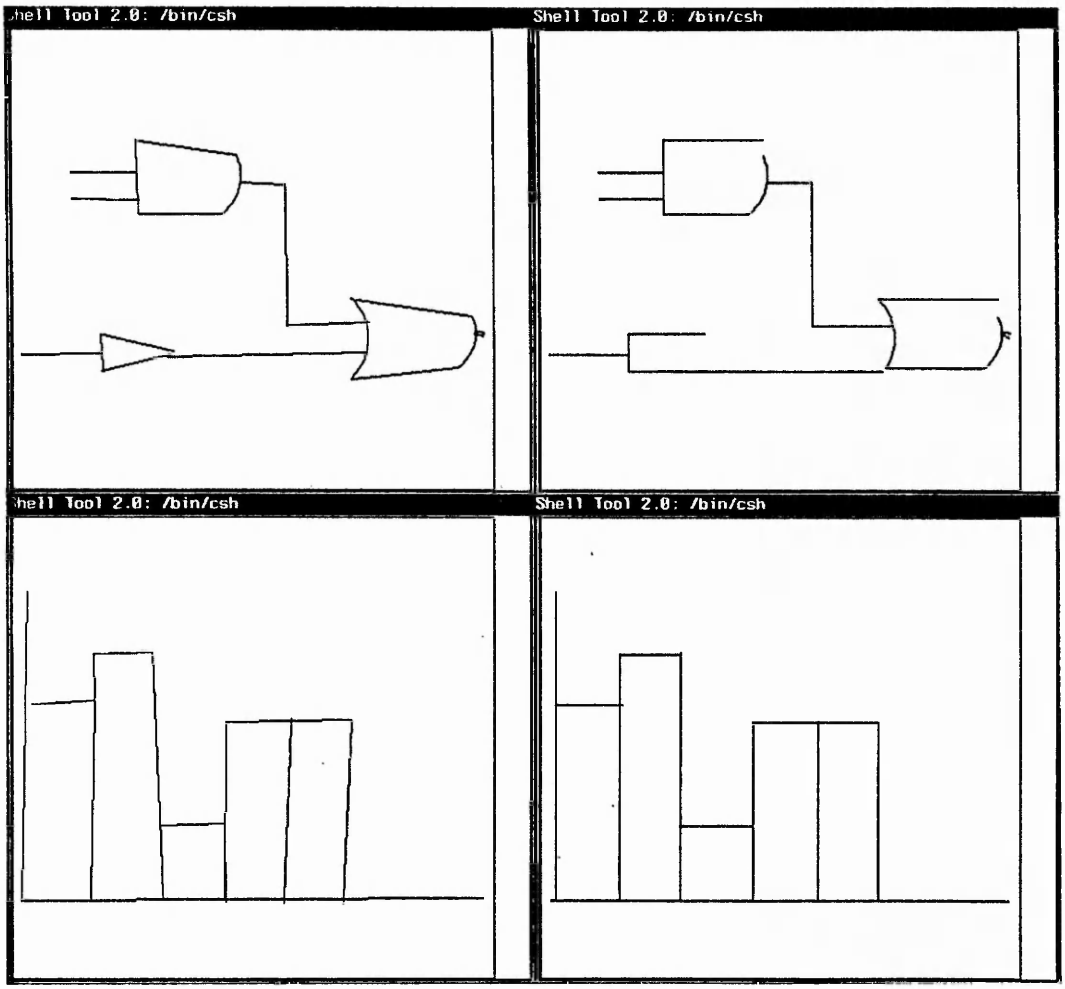
s a


hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
hell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

s a


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

s a.


Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh
Shell Tool 2.0: /bin/csh

s a