FINAL REPORT

NASA LANGLEY RESEARCH CENTER

An Investigation of Optimization Techniques for Drawing Computer Graphics Displays

NSG -
NASA GRANT NO. 1328

Frederick R. Stocker

MATERIALS RESEARCH LABORATORY

THE PENNSYLVANIA STATE UNIVERSITY

UNIVERSITY PARK, PENNSYLVANIA 16802

Frederick R. Stocker, PI
Applied Research Laboratory

Gerald G. Johnson, Jr.
Materials Research Laboratory

Herbert A. McKinstry
Materials Research Laboratory

The Pennsylvania State University
University Park, Pennsylvania 16802

TABLE OF CONTENTS

OVERVIEW

As specified in the first report for Grant #1328 dated May 1976, the remainder of this investigation has been devoted to:

a.) a review of the optimization levels or techniques developed to date and the application of these levels to typical user plot data from both NASA, Langley, and from the Pennsylvania State University Computation Center;

b.) developing some detailed statistics for one of the levels discussed relative to typical types of NASA plotting data sets currently produced in Penn State Computation Center production plotting environments;

c.) the results of the application of optimization levels to the National Bureau of Standards Occidental Character Data Sets;

d.) the results of using optimization levels to develop new techniques to make 3-D static or dynamic images more efficient;

e.) a review of the visual aspects and requirements of the testing methodology.

We will also discuss:

f.) additional areas in which optimizers could prove useful;

g.) some hardware benefits possible.

INTRODUCTION

This study has dealt with but one aspect of graphics data optimization, and that is the one concerned with the proliferation of on-line computer graphics plotting devices both in batch and interactive plotting environments in recent years, with a corresponding drop in cost. This has necessitated the study of how to speed up the output phase to cut the wait time for a plot to be output or, in the case of a refresh graphics terminal, the amount of flicker perceived in various pictures or images. This study has been concentrated on the output phase with particular attention being paid to vector rather than raster oriented devices. The study was further narrowed to concentrate on the optimization of plot data only after it has been formulated into a typical vector move/draw table defined with reference to an output space of definition.

Our previous report dealt with the typical vector structures encountered in a single picture being put out on such devices as flatbed or drum plotters. This report will deal with some of the possibilities for choice of tolerances in optimization based on our data related to real world plots put out by typical systems, with special emphasis on the plotting of data from various NASA Langley Research Center plotting systems and a brief discussion of how, when, and where the techniques might be applied is included.

In addition, discussion of how optimization could be applied to plots produced on real-time interactive display devices is discussed.

Several actual plot data sets both from NASA and Penn State which are typical will be used for illustrative support.

Due to the limited funds available, only the surface of optimization techniques could be developed, but even at that, significant results have already been obtained for typical vector oriented plotting devices. Some of the suggestions for optimization at the interactive level are a direct consequence of results obtained with slower static plotting devices.

The study emphasis was placed at the 2-D move/draw plotting output levels since at that point the user and particular hardware devices come together from an internal programming standpoint and characteristics of both could be taken into account in the development of techniques.

All results were developed relative to existing plotting packages and support hardware so that the results would have immediate utility in real world situations.

TECHNIQUE, OPTIMIZER PROGRAM TESTING AND VERIFICATION

In order to develop effective optimization techniques, several visual approaches to studying plot data were developed on the ADAGE Model 30 graphics computer. Each plot type to be studied had several representative plots selected and the move and draw structures were studied by outputting one or more of the plots on the Adage screen with variable time rates controlling the rate of plotting. As a result, as an optimization algorithm was developed, its effectiveness for a particular class of plots could be seen immediately and compared by plotting the output produced by algorithm modifications all concurrently and at human compatable rates. As a result algorithm development was greatly accelerated in time.

Other plotter hardware was characterizable by sitting at the Adage and running a plot onto the screen at the rates specified for the given devices. Concurrent with this, timing and cost statistics could be viewed simultaneously.

As discussed in the first report, the basic technique for optimizing a vector plot was to eliminate multiple moves which occur naturally in any plotting environment until handled specifically and to reorder the resulting sets of draws so as to significantly reduce the total length of the moves connecting each segment of a plot. Overhead in computer time to accomplish this was then brought into account for higher levels of optimization by setting a tolerance length, referred to here as epsilon, below which a move with length shorter than epsilon would be considered a draw and hence not involved in the move reduction process. In this the final report, our level 6 optimizer will be utilized in all detailed examples since this level has been thoroughly tested and is typical of the levels developed.*

A graphics visual technique was utilized for optimizer algorithm testing as far as the visual part of the graphics data was concerned. For all plot data, set pictures with no flicker, pre-optimized and post-optimized data were alternately flashed on the screen at an effective rate of 40 frames per second. Thus any change in the visible vector structure showed as a set of flashing vectors each of which was displayed at a rate of 20 frames per second. This technique also allowed the pre- and post-optimized move structures to be compared to study the reductions relative to the various algorithms developed. Thus the visual integrity of the original data was constantly maintained and the reliability of the optimizer algorithms checked continuously.

*All level six optimizers with source listings were delivered in machine readable form to the appropriate NASA, LANGLEY, personnel as designated by the NASA technical officer Robert E. Smith, Jr.

All timing statistics were worked up relative to vector plotting system techniques for plot production. Pre- and post-optimization timing statistics were maintained relative to a variable plotter pen control rate where the time for a given increment for both move and draw is set by the tester. Pen up/down time for the pen and ink type plotters was also accounted for in most of the implementations. All statistics were developed by comparing before and after plotting times for percentage savings. Hence, the statistics worked up are plot dependent and reasonably independent of the hardware device rates of the typical plotters used for testing at the Penn State Computation Center. Device switching time for going from a move to a draw or a draw to a move was considered as negligible. All plots were converted to a form plottable on a Calcomp 563 and an ADAGE model 30 for comparison purposes.

TYPICAL SAMPLE PLOT DATA SETS

Typical of the plot data produced by users is that illustrated in Figure 1 through Figure 13b.

Better than half of the plot data sets used for algorithm development and testing were contributed by users of the Computation Center's graphics programs. Several programs which produced plots were enhanced after optimization techniques were applied and thus some Penn State graphics users have already realized benefits from the techniques developed to date. As an example of where this occurred, when the EZGS Graphics System (see Fig. 5a through Fig. 5c for comparisons before and after the application of optimizer level 6 to the plot data) was being integrated into the Computation Center's graphics environments, the time required to plot a typical picture was found to be excessive whenever heavy overhead was encountered, optimization was applied to determine if there could be significant time and cost reductions. It was found that much time was saved and therefore EZGS algorithms would yield significant improvements in plotting time if they were reworked. This allowed EZGS to be effectively integrated into the Computation Center graphics environments. Note that Fig. 5c also represents the move structure resulting from our current implementation of EZGS once the rewrite was accomplished.

Art/Architecture data typically has a lot of pattern data with returns to origin of each pattern common (i.e., each template or subpattern is started from an origin and then a return to that origin is also present to conclude the template). Also, many times the individual components are spread all over the output space of definition but in the final copy appear coherent or well organized (see Fig. 6a through 7c for comparisons before and after application of level 6). Only by actively watching the plot is it clear that much jumping around on the surface is required to output the plot.

Graph output is usually much cleaner due to much previous programmer experience with such plots and hence such output typically yields less overhead to optimize. This is usually the result of using previously well developed packages for axes, character, and curve generation (see Fig. 4 as an example of such a plot used in our testing).

Surface data takes distinct advantage of optimization due to the use of sweep surfaces to represent them. The modification of a sweep typically from strictly left-to-right to one of left-to-right alternated with right-to-left is automatically performed by an optimizer when epsilon is chosen appropriately. (Reference Fig. 8 for a typical plot used in our testing.)

Finally, the type of data supplied by NASA, Langley, yielded well to the optimizers and it embodied most of the characteristics mentioned for the first four examples. Therefore, let us now discuss the NASA, Langley, data in more detail.

NASA LANGLEY DATA

In order to assess the possible needs and applicability of various optimization techniques to typical NASA type plotting data, several plot data sets were obtained from NASA Langley through Bob Smith. One such set, noted as JET3 data, was used to test various levels of optimization as they were being developed. The results of this testing on the final version of level 6 of the optimizers is shown in Figures 10a through 13b, and summarized in Figures 14a and 14b located at the end of this report. Note that the comparisons indicate quite clearly that significant savings in both time and money are possible with such optimizers. It can be seen that to simply minimize the total sum of the lengths of the moves can be quite costly in processing time if such a device as epsilon is not introduced. It can also be seen that by picking an appropriate epsilon the costs in time can be quickly reduced to reasonable levels. Hence the question becomes one of choosing epsilon appropriately.

It should be noted that the typical NASA, Langley, plot data set is quite similar in nature to the typical plot data sets found in the Penn State University plotting environments.

Hence, for the rest of this report, the NASA JET3 data will serve for illustrations.

Consider the jet data shown in Figure 1. For the sake of this discussion, all surrounding data to the jet was removed as shown in Figure 2a. Its corresponding move structure is shown in Figure 2b. Note the line density. Figure 2c shows the resulting move structure after the tolerance value is set greater than the longest move (value is 6000, which is equivalent to 30 inches on our system) thus effectively taking out only multiple moves. The density of moves in terms of length is still fairly high. Now check Figure 3a where the tolerance value has been set to 0. Here the length of the sum of the moves has been greatly reduced but, as can be seen from Figure 4 which shows the choice of tolerance value epsilon plotted against CPS time on an IBM 370/168 computer, this has been costly in terms of computer processing time. As can be seen from Figure 4, a more realistic epsilon would be 60 (three-tenths of an inch) as this keeps CPU time down and still produces a greatly reduced value for the sum of the move lengths. This reduction is visually demonstrated in Figure 3b which shows the resulting move structure.

The cost in time to plot the original data on a Calcomp plotter was 15 minutes and 6 seconds. The cost with epsilon set to 0 was 5 minutes and 33 seconds but the CPU time for data preparation was too high. With epsilon set to 60, the plotting time was 6 minutes and 54 seconds while the CPU time is 9 seconds in our computer environment. More work needs to be done in the area of automatically choosing epsilon for a given set of plot data.

In a static environment, epsilon was initially determined by considering the number of moves in the data involved and then testing in at one quarter to one half an inch. Later statistics indicated this was a good choice in the computer environments in which the plot was being produced. This value for epsilon is a percentage of the plots overall scale and hence eventually such a value should be chosen as a percentage of the total moves times the basic raster length of an output device.

HERSHEY CHARACTER DATA

Another major data base considered was the Hershey Occidental character data base made available by the government. By applying the optimizers to a typical subset of the character data (see Fig. 9) a decrease in plotting time of approximately 12% for typical plotted paragraphs of text was realized using the subset. The potential long-term savings with respect to the use of plotted characters is thus seen to be considerable.

We specifically tested against the following N devices because of their widely differing characteristics.

1.) Tektronix 4662 flatbed plotters as well as several tubes
2.) Calcomp 563/564 drum plotters
3.) Gencom 300Q and AGILE A1 Typewriter Terminals
4.) Anderson/Jacobson 832 Typewriter Terminals
5.) Adage Model 30 Interactive Graphics System

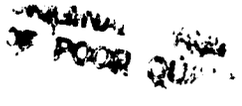
Special control parameters were assigned in some cases due to the nature of the devices chosen. For example, the GENCOM 300Q typewriter terminal caused us to minimize and re-order the long vertical moves between characters and also to typically start at the top of the paper, run down the paper and finish off the bottom in order to reduce paper slippage due to carriage roll. To accomplish this, each character was first optimized individually. On the other hand, as a second case, the Calcomp plotter picture of characters was treated globally due to the Calcomp plotter's high degree of repeatability. As a third case, the Tektronix 4662 characters were treated from two standpoints. One of line speeds available and the second with an eye to hardware character generator character substitution for software characters where appropriate. It should be noted that no general study has been done as yet on typical english text where some characters such as the letter "E" predominate. Such paragraphs might yield more-or-less savings and would certainly be worth looking at.

USE OF LEVELS TO UPDATE 3-D STATIC IMAGES ON SCREEN MORE EFFICIENTLY

Several methods were used to make images more efficient in terms of lower vector counts and less wasted move structure. The methods typically resulted in flicker reductions in images with high vector counts.

One method was to automatically take each 3-D sub-picture as it was being shown and while the original picture is on the screen, rebuild its corresponding 3-D table with one of the optimizers and then replace the original image with the new one. A second method was to take each sub-picture and treat it as a point and then optimize the interconnecting move structure thus reducing the potential for image tearout for images too far apart. A major disadvantage with our current optimizers occurs here for images which require different types of optimizers to be utilized depending on the image's overall orientation in three space. Which optimizer should be applied in this case is currently a result of visual inspection of the move/draw structure of a particular image combined with a choice of epsilon based on past experience with similar types of data.

As a result of the above, a third method developed allowed the user to momentarily view a plot's move structure in time and determine by visual inspection whether optimization is warranted and at what level.



WHERE USEFUL

The optimizers have been found to be useful in plotting algorithm development since they allow such development to proceed without initial concern for typical plotting inefficiencies. At the same time, before and after views of the user's algorithm output causes the user to realize where improvement is needed or even possible. This can save much time in the case where someone may be trying to improve an algorithm that doesn't need it from the plotting standpoint.

If it becomes necessary to put a cleaver algorithm modification in that may yield faster plotting times but make the program hard to update, then an optimizer can do the job and the original algorithm left alone. One simple example is seen when a surface is being output with a sweep from left to right for example. Rather than complicate the algorithm by building the sweep into it, just apply an optimizer and the sweep reversal will be accomplished automatically.

The optimizers automatically reduce the total output time by reducing the amount of data to be sent down a line for a plot.

The optimizers allow such packages as EZGS to be utilized even when their original plots' output would take too long to get on paper.

At the same time, the optimizers suggest to the programmer how a package should be reorganized and reprogrammed with minimum time wasted.

The optimizers automatically reduce flicker on vector systems by simplifying an image's components.

OPTIMIZING MANY-COLORED PLOTS

One problem area in which the optimizers are quite handy is that of outputting a mixed color plot to a device. First an N-colored plot is separated into N files of moves and draws, one for each color. Then an optimizer is applied to each of the N files as well as to all of the files collectively and the resulting files are plotted in the sequence indicated by the last optimizer application. This reduces the number of color changes, both human and mechanical. One significant result of this is a much freer use of color by a graphics programmer since constantly having to change pens, or the use of a plotter with less pens than you want, is virtually eliminated. (You must be willing to make the minimum number of pen changes necessary.)

HARDWARE BENEFITS

Several benefits have been immediate at the computer graphics hardware level. On the typical vector plotters such as the Tektronix 4662 flatbeds, Calcomp Plotters and the Adage model 30, the time to simply display or plot a typical image is often greatly reduced. This of course allows more image data to be processed in a given amount of time. On the Adage scope, images take less time to be displayed for a given frame time and hence the amount of time available in any given time period to a support program is increased. At the same time, if an image flickers on the screen, the flicker is reduced or eliminated by the compressors.

With the typical typerwiter terminal such as a Gencom 300Q, an Anderson Jacobson 832 or an AGILE Model A1 a significant reduction of paper shift or movement was realized whether time was saved or not. This allowed plots which would distort due to heavy vertical movement to be plotted on such devices with no apparent shifting or skewing.

The combining of two or more plots without the need for careful checking of order of combination is made possible by passing all of the plots through the optimizers and letting the optimizers do the work.

Finally, the optimizers take the typical plot data set and reduce all code counts and structures to a minimum for typical line transmission. This enables a typical plot to be output at optimum speed with respect to a particular plotting device and the corresponding transmission (baud) rate implemented. Again, significant time can be saved here.

SOME OF THE NEW TECHNIQUES

There were many new techniques tried in the optimizers, some of which have been discussed earlier in this report. In addition, it should be noted that the optimizers would be best installed at the output level with an on/off option so as to allow for normal preview when there are bugs in an algorithm. This would require a fair sized chunk of memory and a microprocessor, but such technology is getting cheaper. Such memory should be user allocatable so as to allow for partial or total plot data sets to be optimized as a function of user allocation.

Also, a character recognition routine could be installed at the micro-processor level to allow the substitution of hardware character generator characters for existing software characters in a plot thus saving additional transmission time.

Color sorting and separation could also be taken advantage of by putting a control program in at the micro-processor level thus minimizing the number of physical ink changes required as discussed earlier.

SUMMARY

The primary emphasis of this study was placed in the area of computer graphics data optimization techniques and their application to picture plotting times and costs as they affect the computer graphics user. All theory and techniques were tested on graphics plotter data which was output from existing plotting systems currently made available at the Penn State Computation Center. In addition, systems and device timing were all done using the variety of computer graphics hardware made available at the Computation Center. Graphics plotter data optimization has been seen to cover an extensive number of computer graphics software systems as well as a variety of plot output devices. Thus, we restricted ourselves to vector graphics at or near the plot output stage.

The fundamental idea employed for vector data was that of reducing plotting time by reducing the total sum of the lengths of the moves contained in the data for a plot before the picture was actually plotted on an output device. However, if all move endpoints were sorted on, the computer time necessary to prepare the data typically became excessive and thus made optimization costly. Hence a tolerance value was specified and whenever a move was encountered with length less than the tolerance value, it was treated as a draw and not utilized in the computer sort on the moves for optimization. This meant that the choice of the tolerance values determined the amount of time spent on the optimization sort of the moves as well as the time it took to plot or output the picture.

The optimization techniques have been used to produce plotting time reductions of as much as 90%. Our application of the optimizers by existing graphics character data bases which are used repetitively have produced percentage time savings in the range of ten to twenty percent for typical paragraphs of plotted text. Another significant application was that of graphics plotting algorithm review and improvement. As noted, an optimizer applied to early output indicates whether effort should be expended to improve the algorithm from a plotting data layout standpoint. This is certainly of benefit for algorithm development where concern over programmer data ordering for output can be forgotten until an algorithm for plotting has reached design completion since an optimizer can keep plot output test time to a reasonable level. One final note on scope or CRT systems: by optimizing

vector data for a vector scope, the resulting picture will typically flicker less on refresh CRTs. This provides enhanced visual perception for high vector count images.

PROJECT MEMBERS

Principal Investigator

Frederick R. Stocker, Applied Research Laboratory, PSU

Assisted by

Dr. Gerald G. Johnson, Jr., Materials Research Laboratory, PSU
Associate Professor of Computer Science

Dr. Herbert A. McKinstry, Materials Research Laboratory, PSU
Associate Professor of Solid State Technology

Student Support

Richard Lee Kulp, Graduate Student in Computer Science, PSU

James M. Allison, Undergraduate in Computer Science, PSU

Lawrence J. Jones, Undergraduate in Computer Science, PSU

Randall C. Quinn, Undergraduate in Computer Science, PSU

Special thanks for several plot data sets go to

Scott Cox, Computation Center

Larry Bunge, Computation Center

Ray Masters, Department of Arts and Architecture

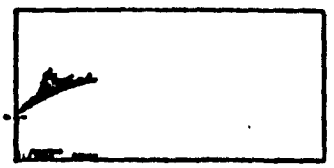
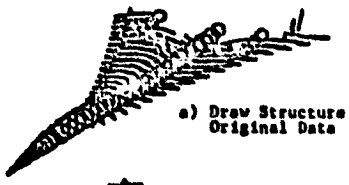
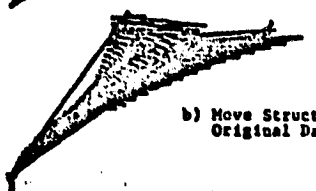


Fig. 1 - Original Data





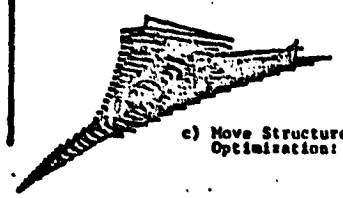


Fig. 2 - Windowed Data

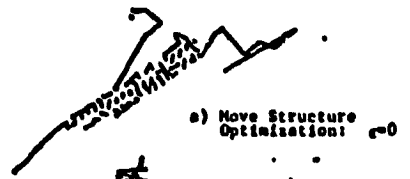


Fig. 3 - Windowed Data

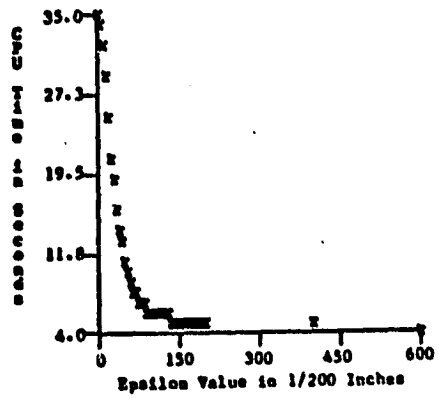


Fig. 4 - CPU Time as a Function of Epsilon

The four figures above were extracted from the review

Optimization Techniques for Drawing Computer Graphics Displays

presented at the Langley Basic Research Review in February of 1978 by Robert E. Smith, Jr.

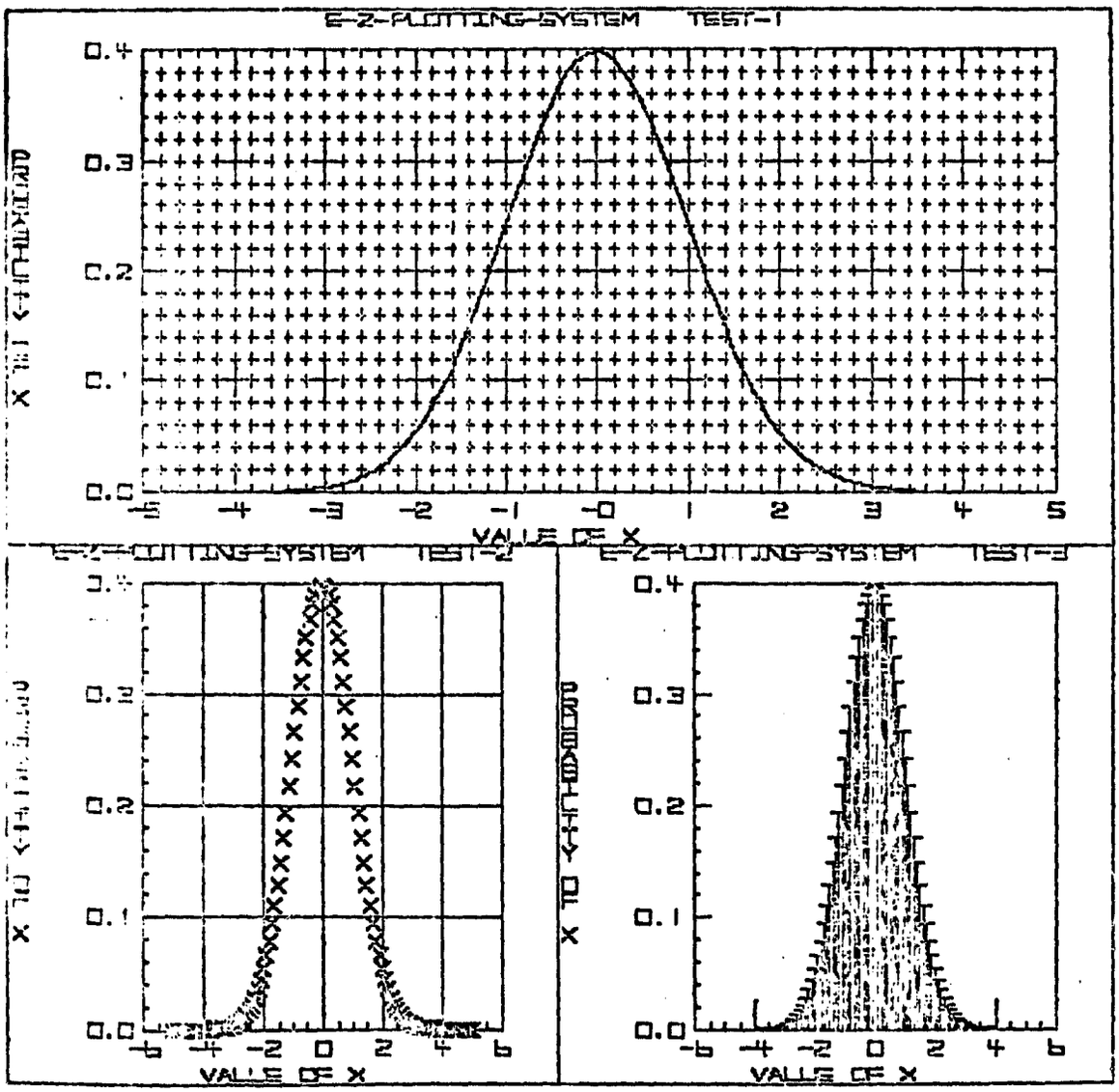


Fig. 5a - A typical set of plots output by EZGS

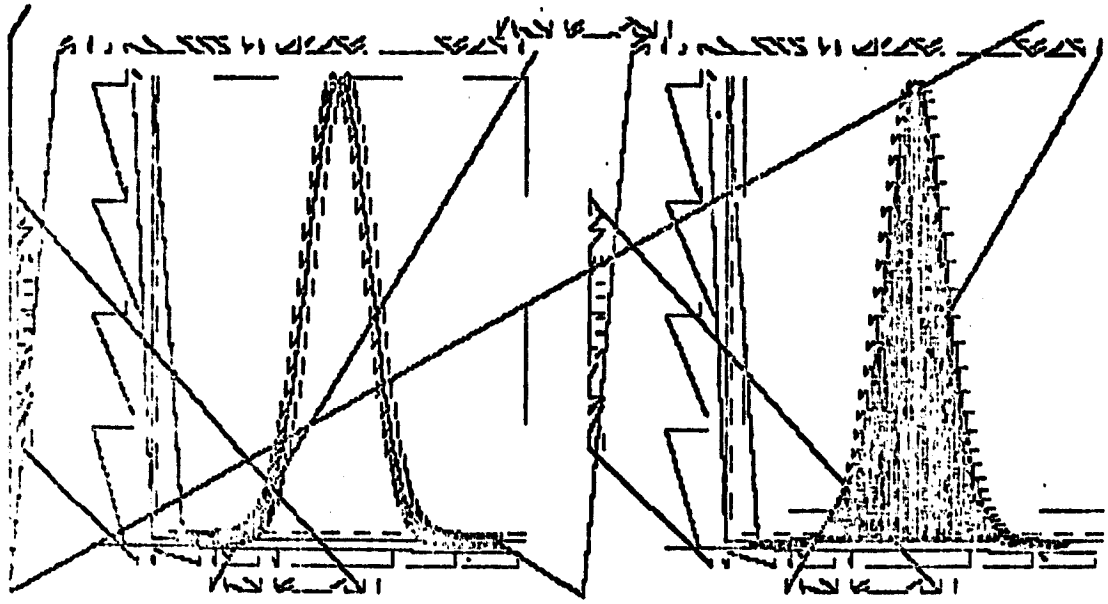


Fig. 5b - The corresponding move structure for the lower two plots

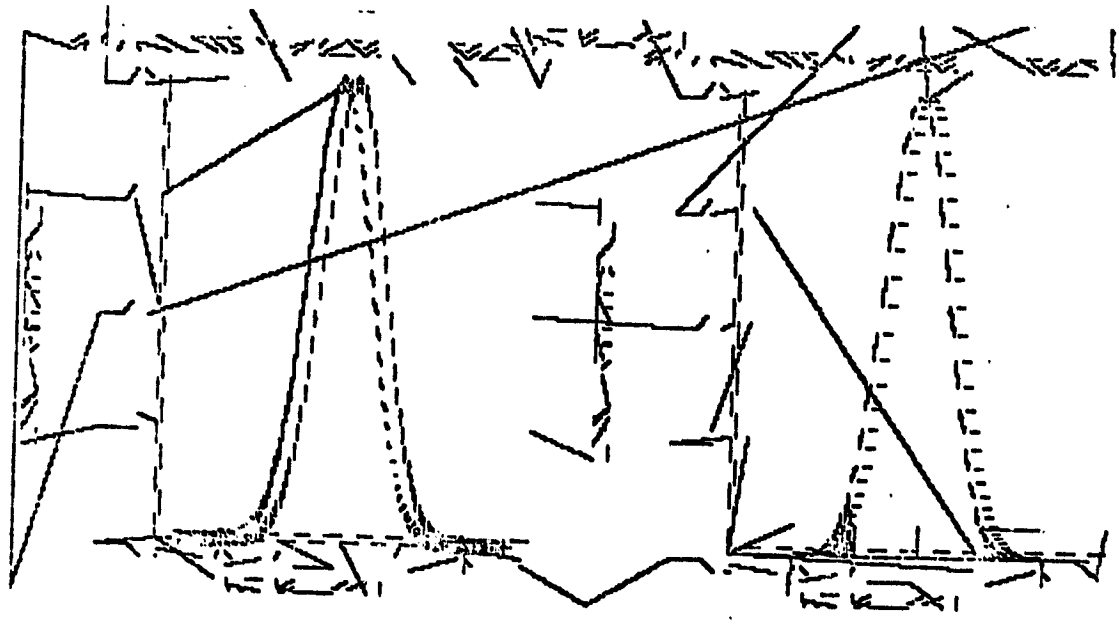


Fig. 5c - The corresponding move structure for the lower two plots after the level 6 optimizer has been applied to it

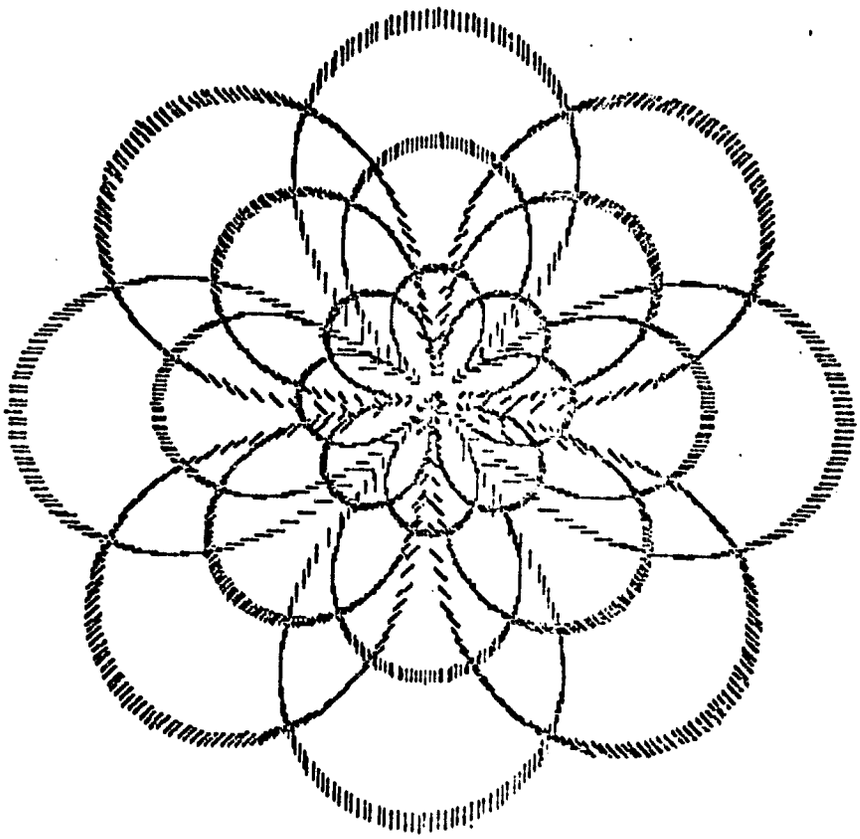


Fig. 6a - Typical Art data

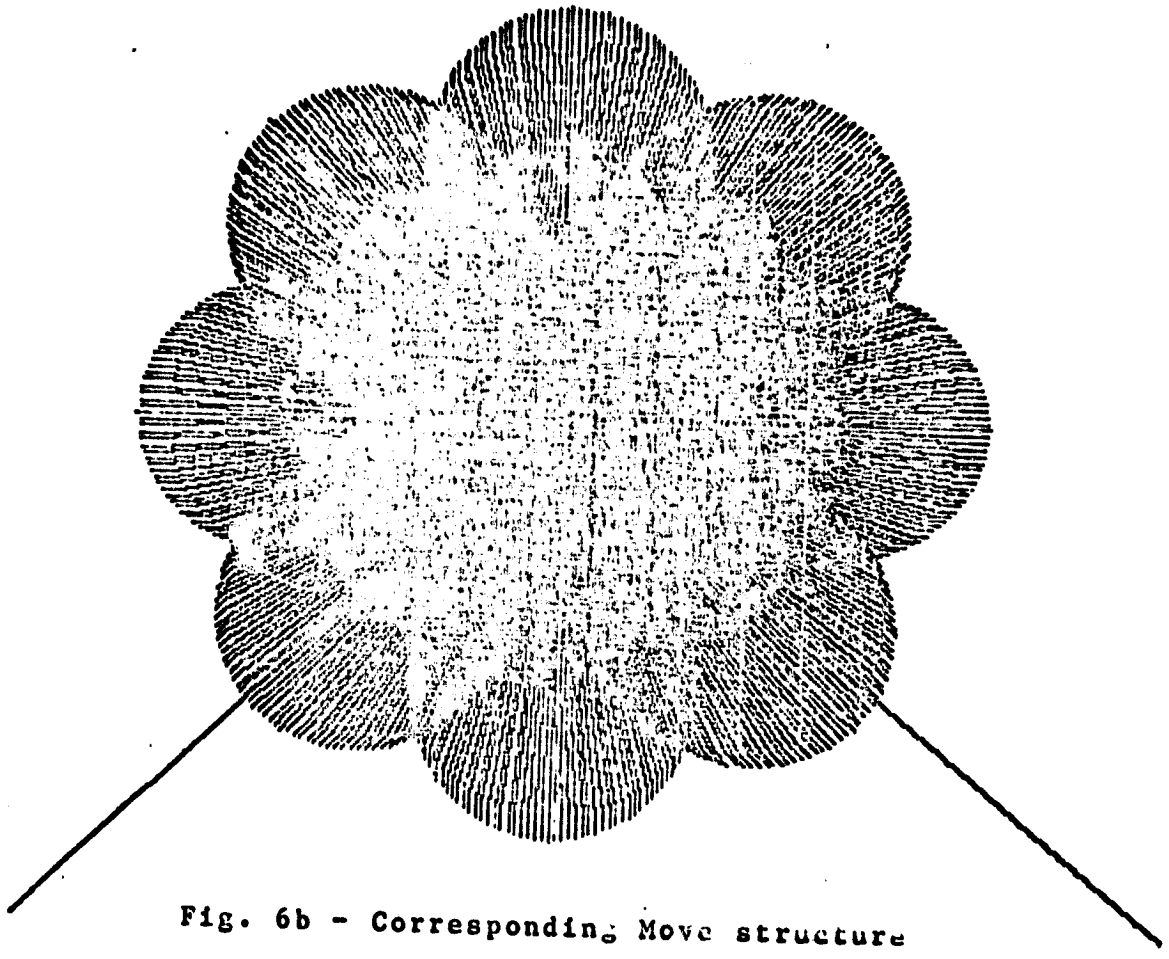


Fig. 6b - Corresponding Move structure

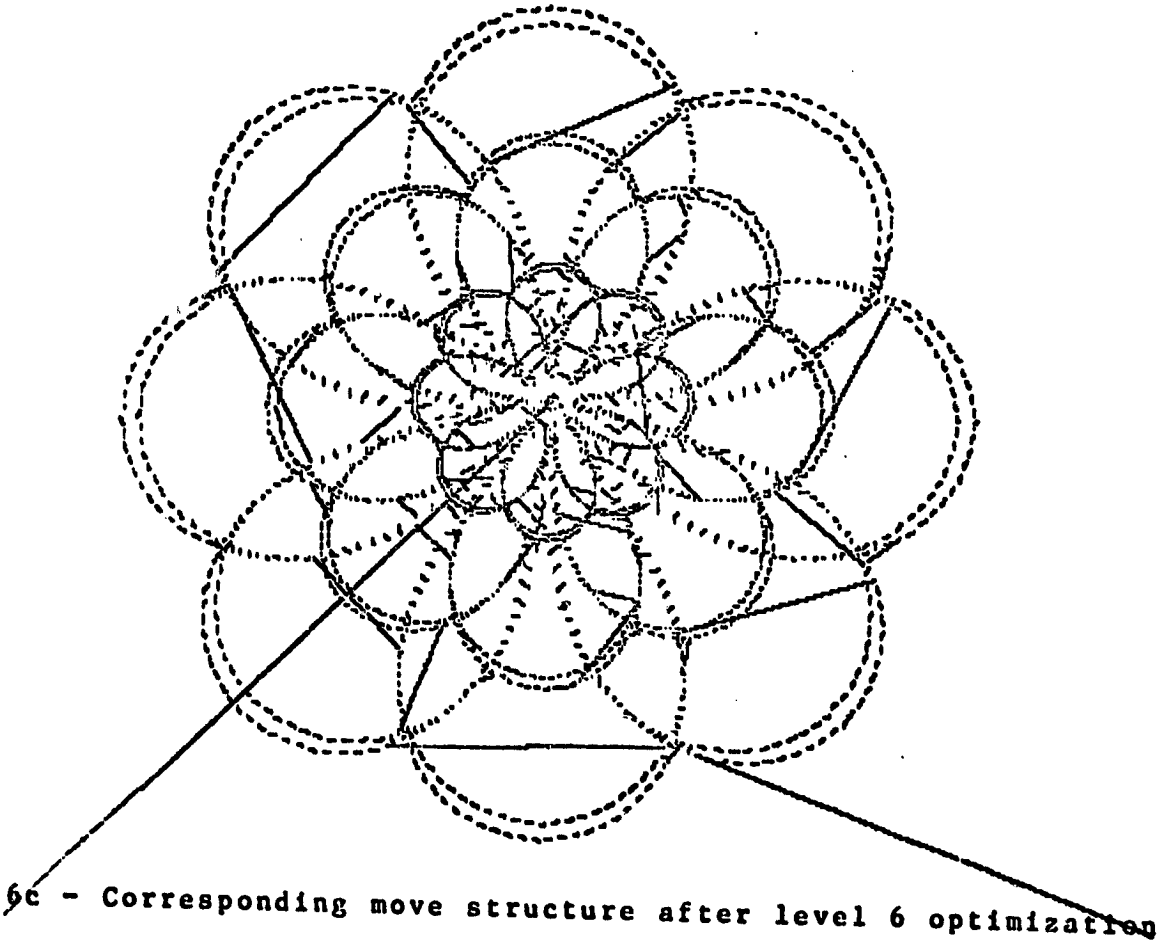


Fig. 6c - Corresponding move structure after level 6 optimization

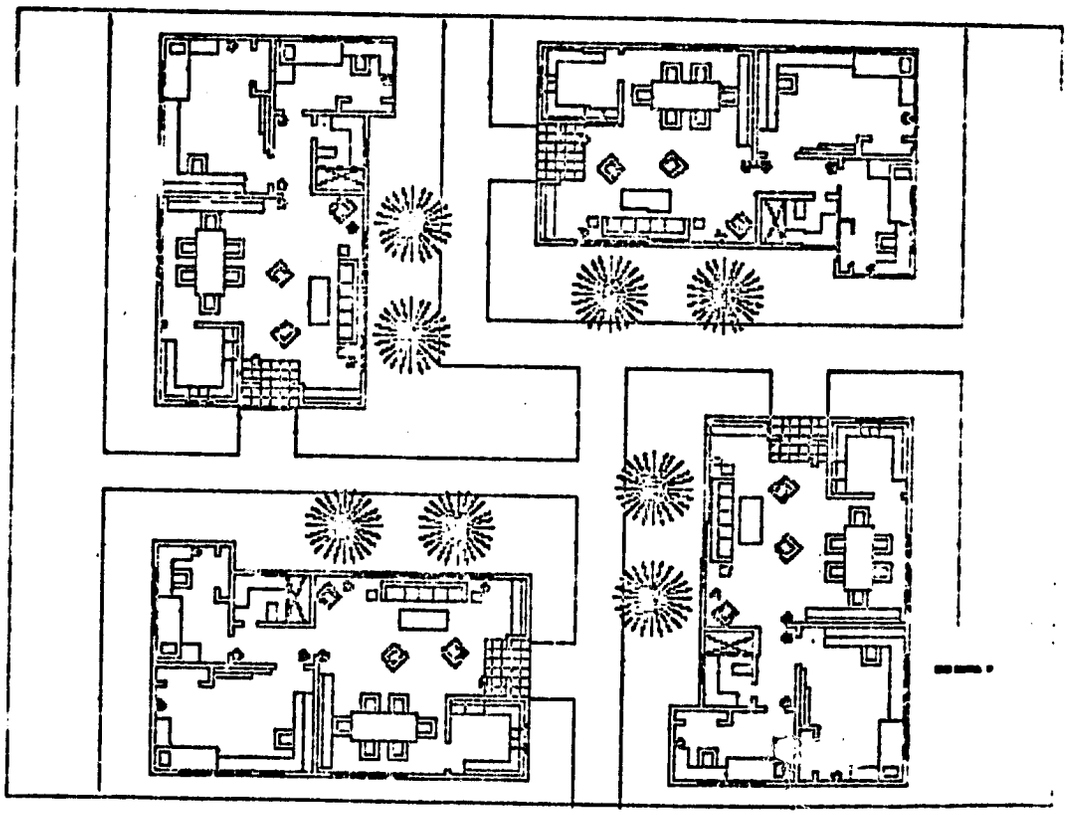


Fig. 7a.- Typical architecture data

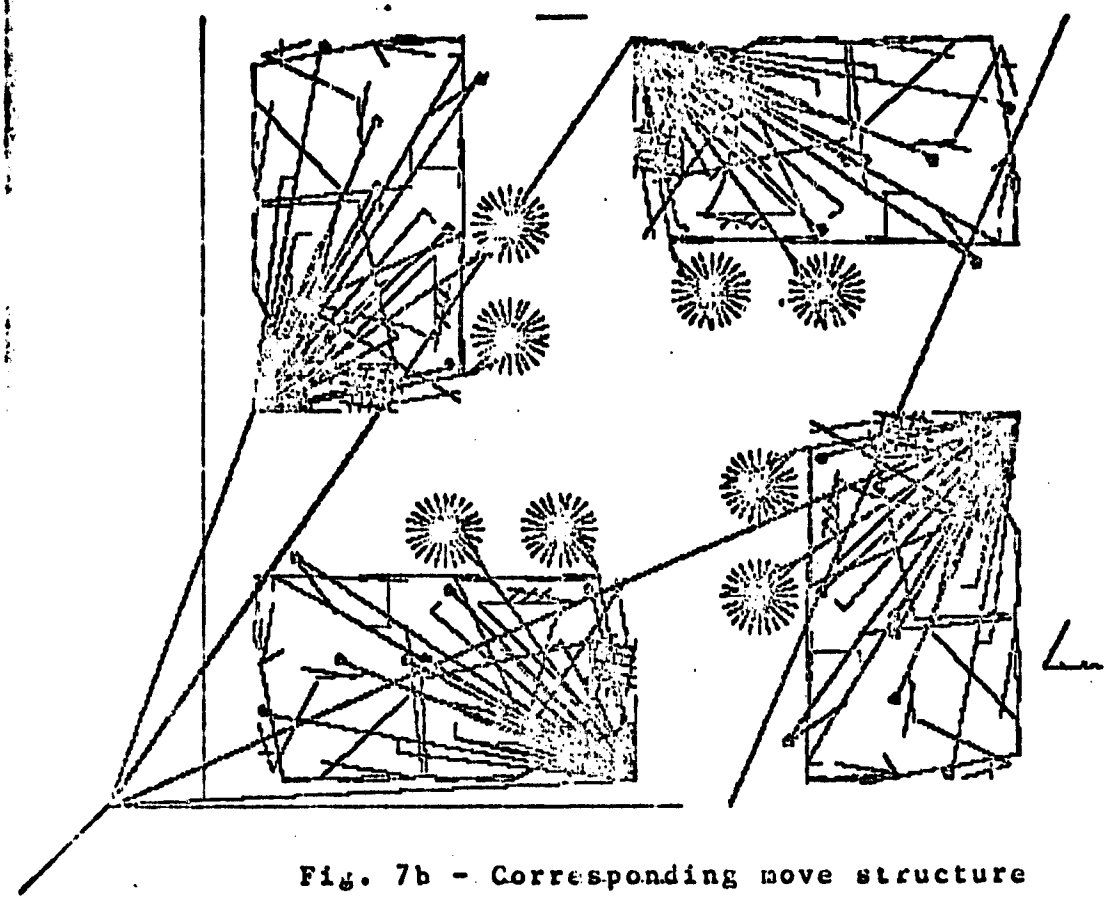


Fig. 7b - Corresponding move structure

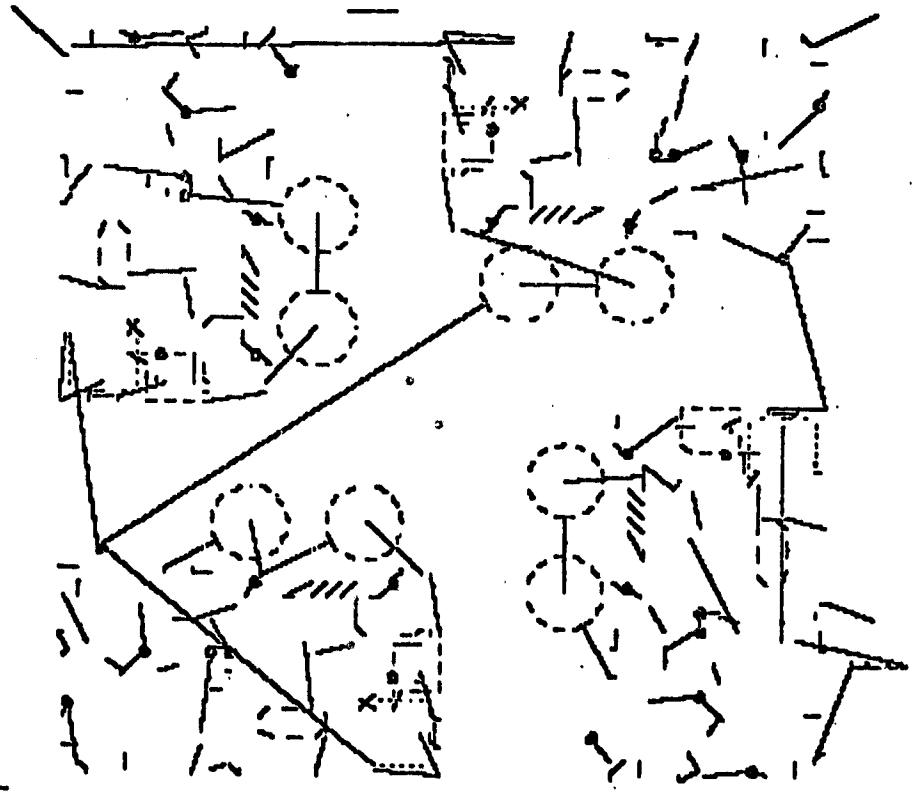


Fig. 7c - Corresponding move structure after level 6 optimization

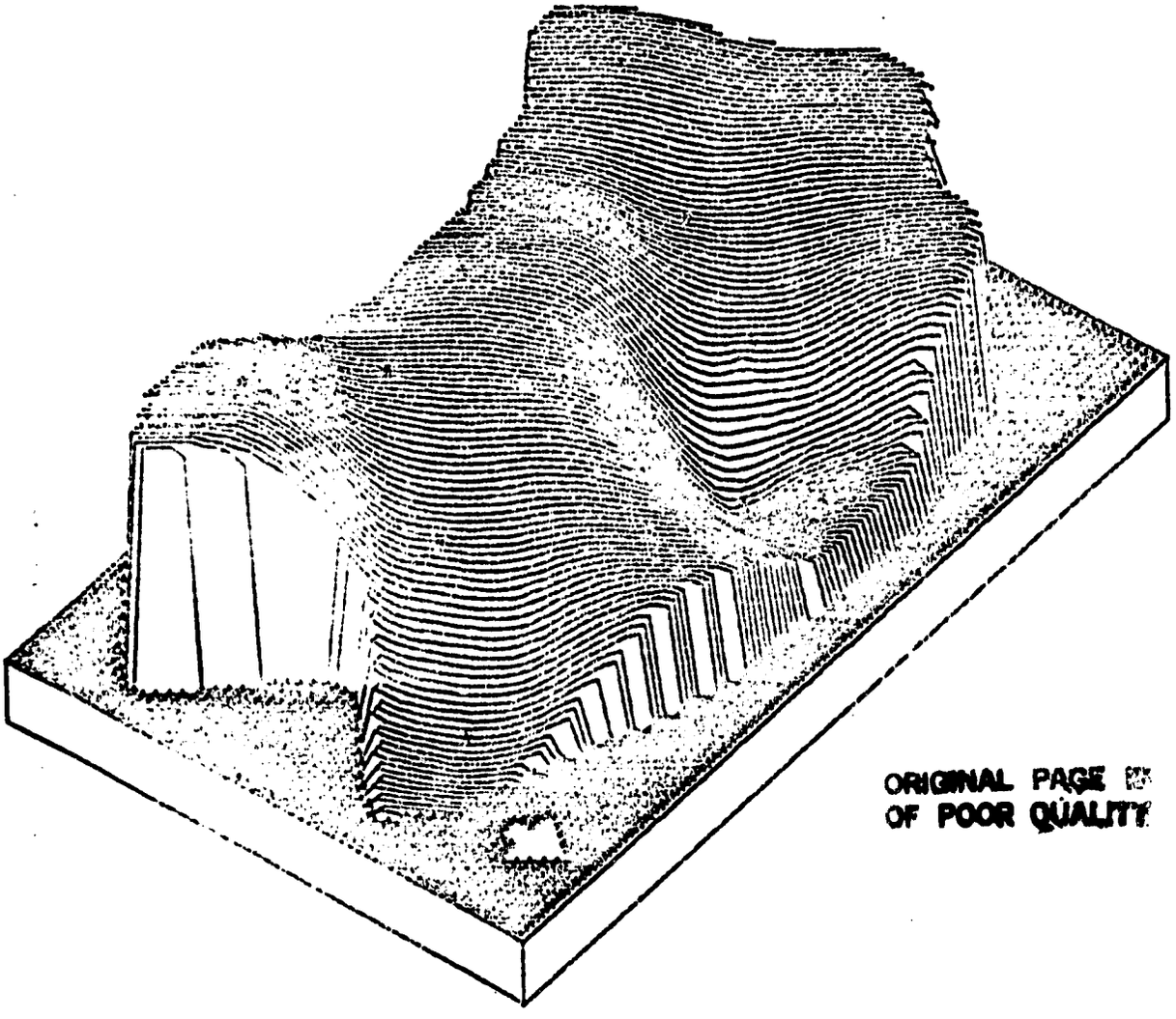


FIGURE A : MANTEGNA BAY - CONTOUR

AZIMUTH - 231 ALTITUDE - 45
WIDTH - 6.00 HEIGHT - 2.00

* BEFORE FORESHORTENING 11/02/77

Fig. 8 - Typical surface data

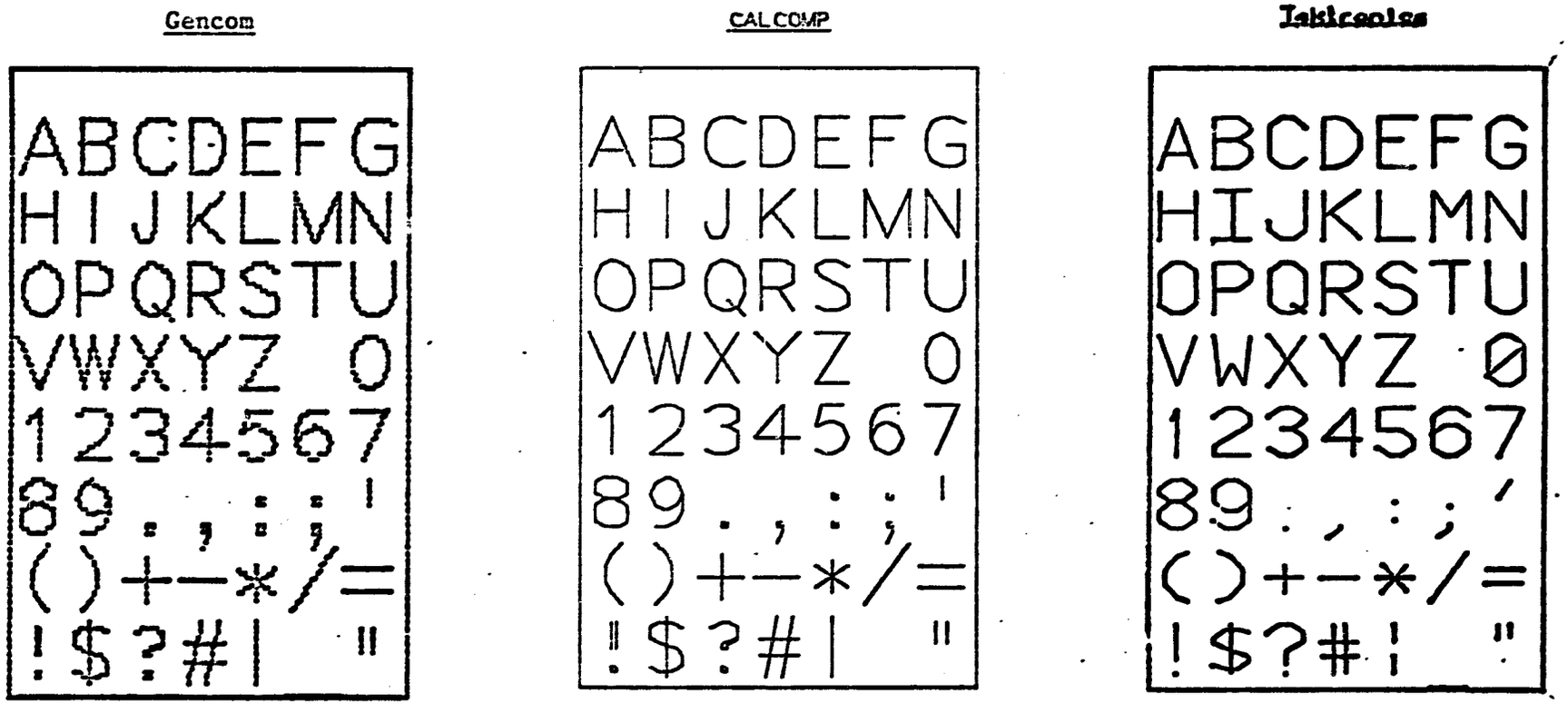
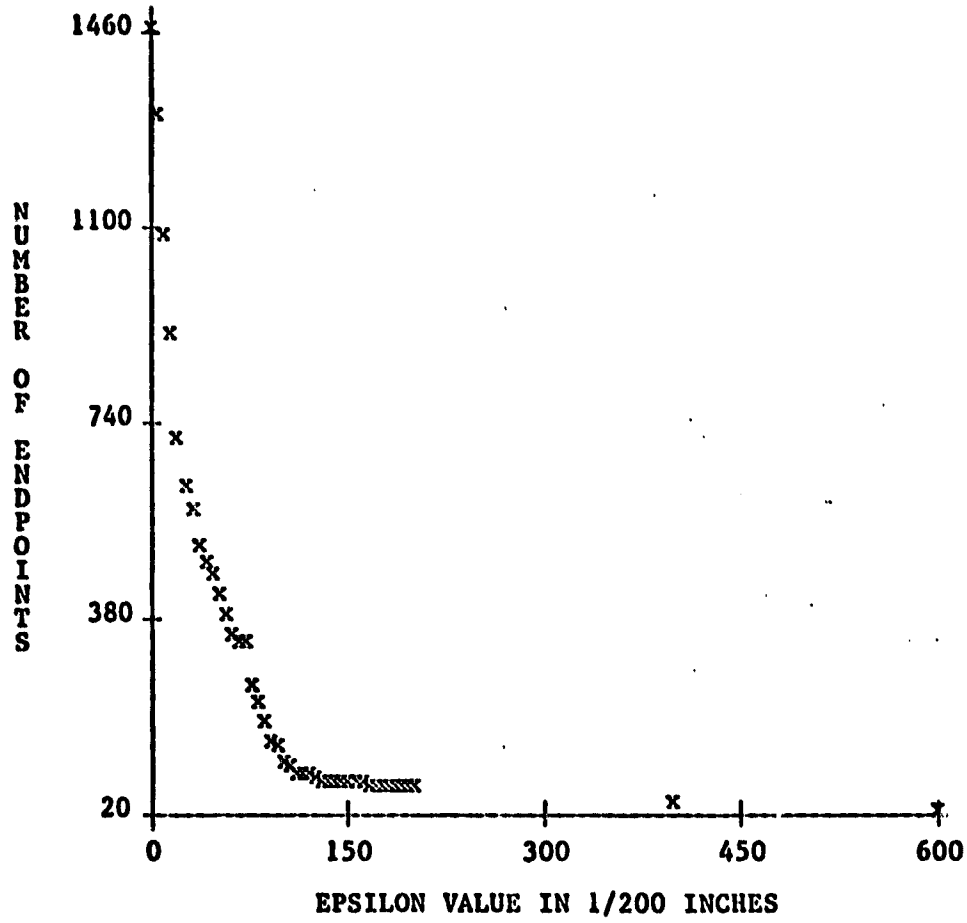


Fig. 9 - Portion of occidental character set used for testing on different devices

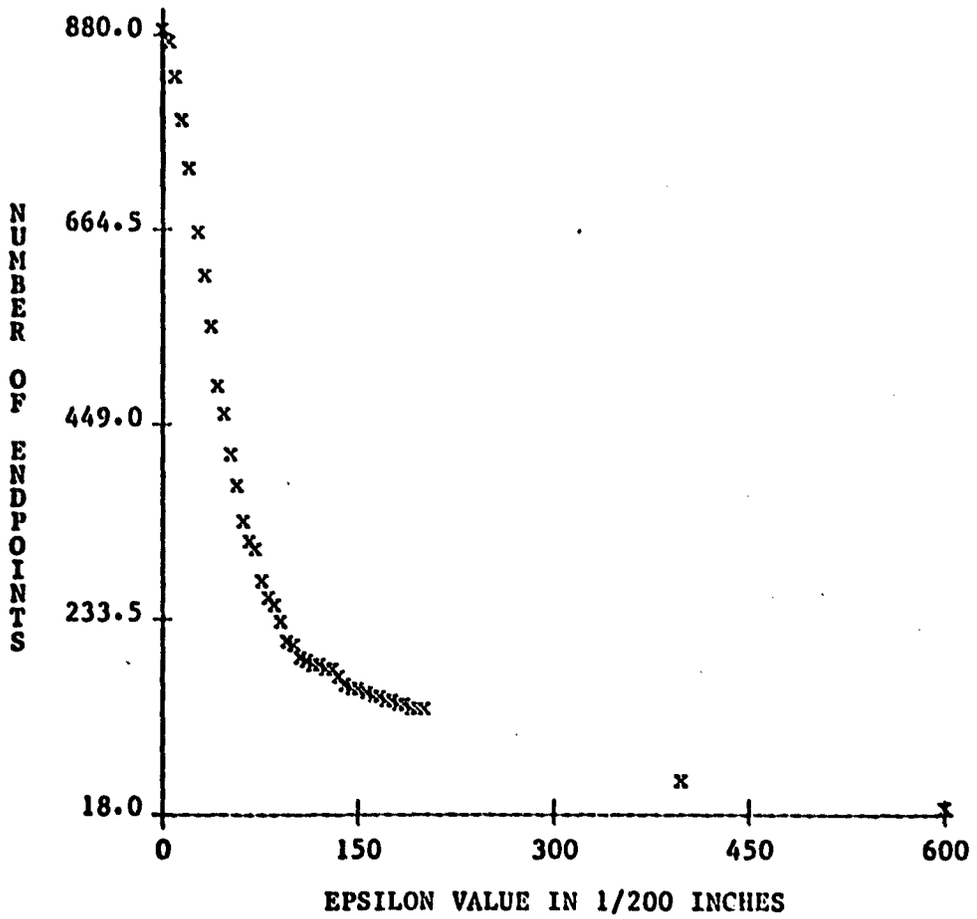
EPSILON VS NUMBER OF ENDPOINTS - JET 1



THIS GRAPH SHOWS THE RELATIONSHIP BETWEEN THE SELECTED VALUE OF EPSILON AND THE RESULTING NUMBER OF SUB-IMAGE ENDPOINTS PROCESSED.

Fig. 10a
Comparison plot for JET1

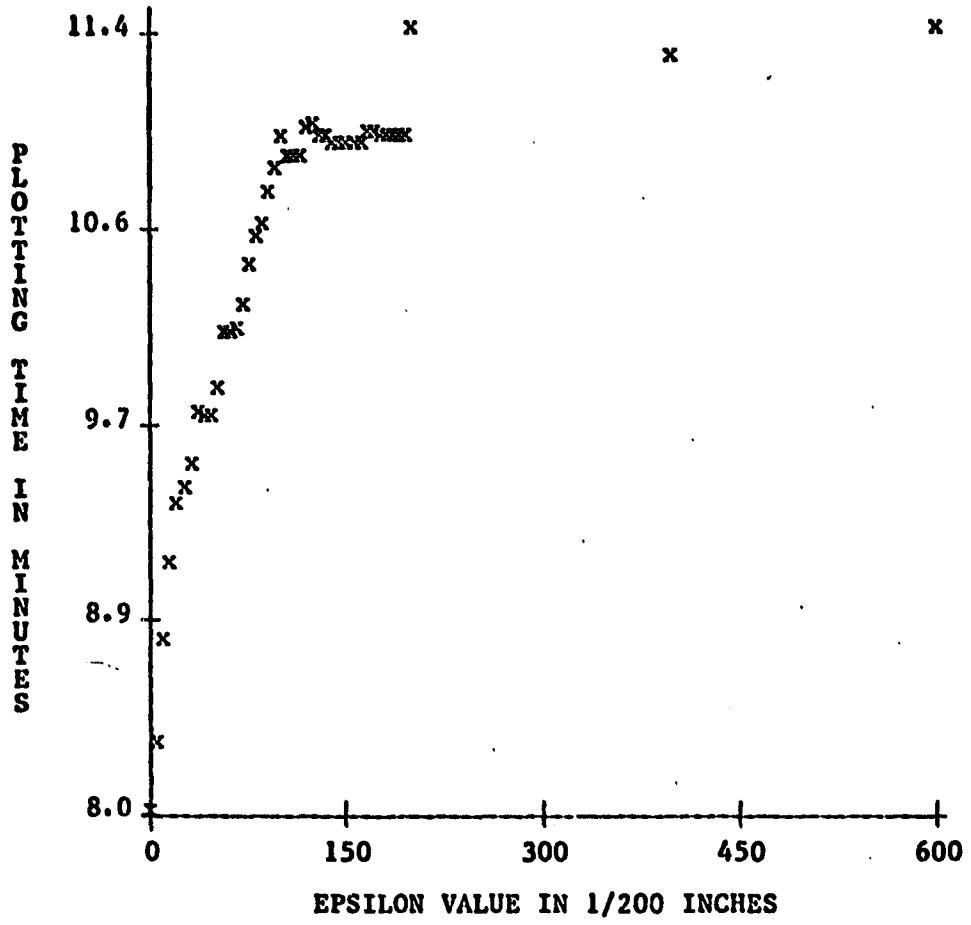
EPSILON VS NUMBER OF ENDPOINTS - JET 3



THIS GRAPH SHOWS THE RELATIONSHIP BETWEEN THE SELECTED VALUE OF EPSILON AND THE RESULTING NUMBER OF SUB-IMAGE ENDPOINTS PROCESSED.

Fig. 10b
Comparison plot for JET3

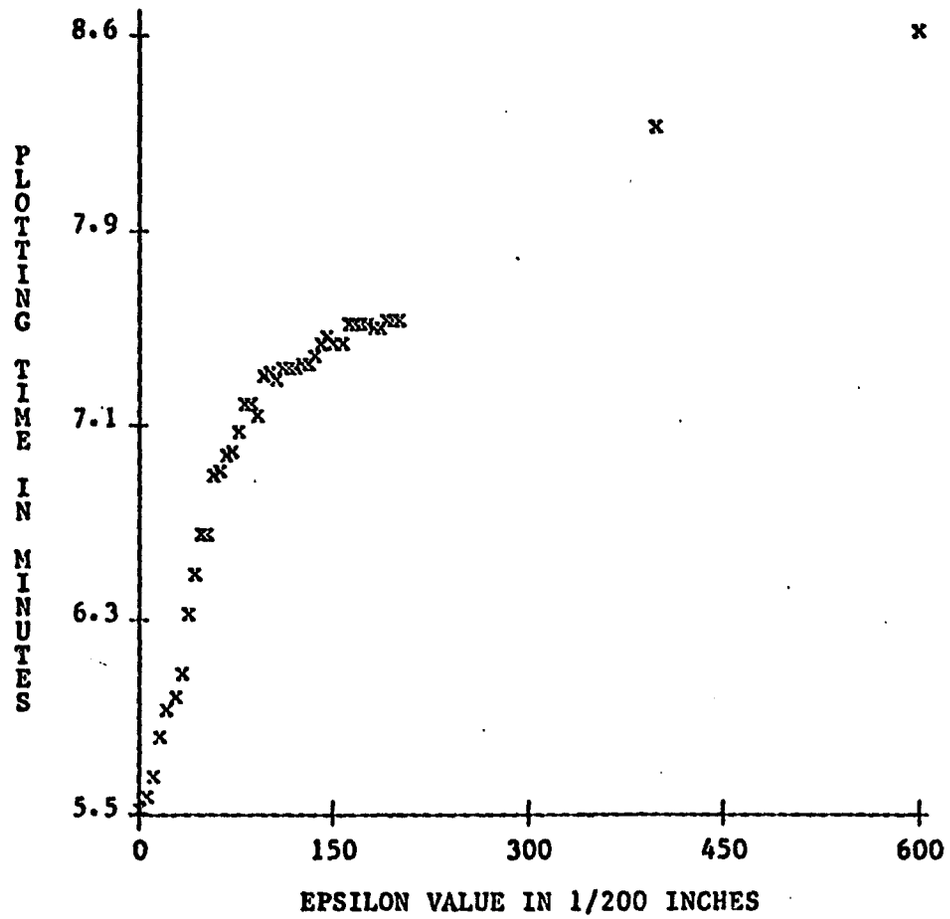
EPSILON VS PLOTTING TIME - JET 1



THIS GRAPH SHOWS THE RELATIONSHIP BETWEEN THE SELECTED VALUE OF EPSILON AND THE TIME REQUIRED TO PLOT THE RESULTING DATA FOR JET 1.

Fig. 11a
Comparison plot for JET1

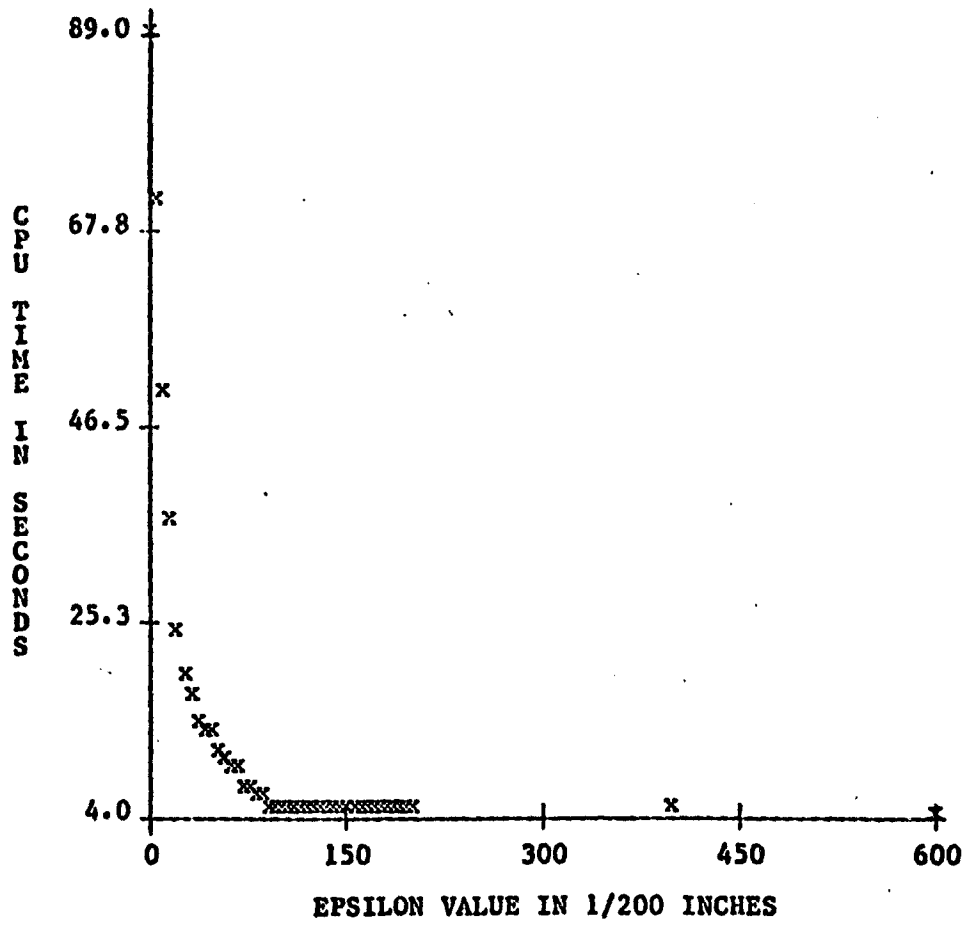
EPSILON VS PLOTTING TIME - JET 3



THIS GRAPH SHOWS THE RELATIONSHIP BETWEEN THE SELECTED VALUE OF EPSILON AND THE TIME REQUIRED TO PLOT THE RESULTING DATA FOR JET 3.

Fig. 11b
Comparison plot for JET3

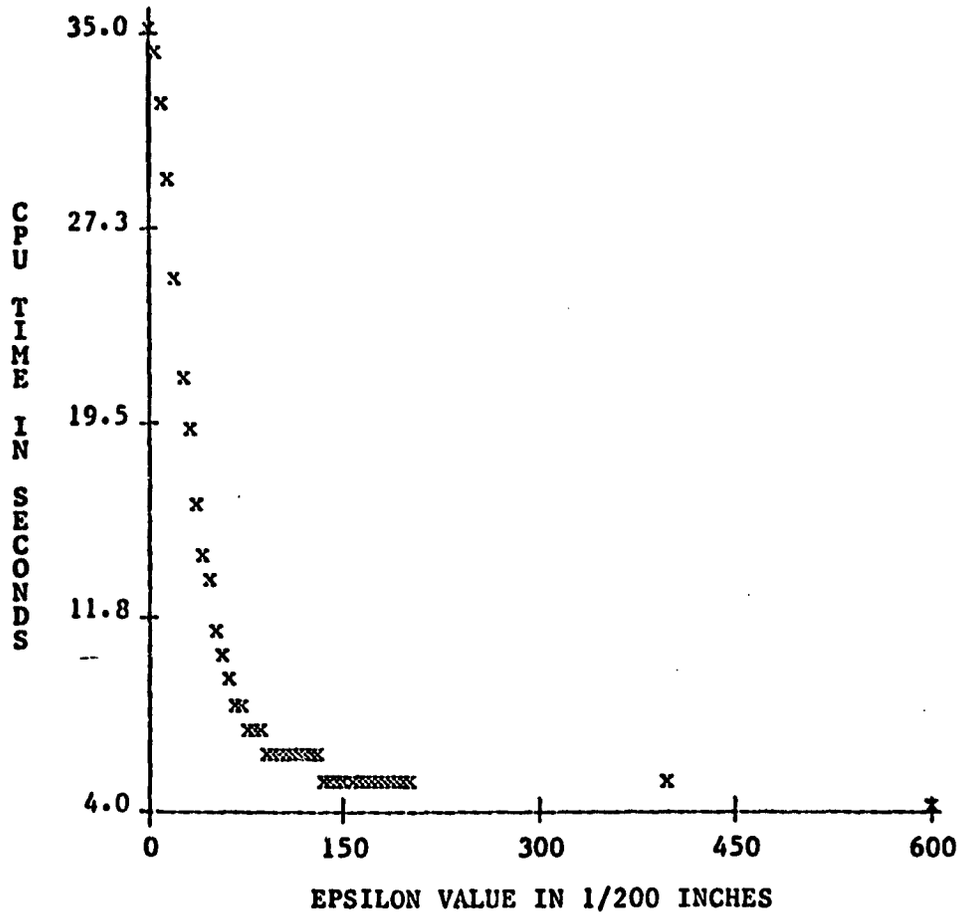
EPSILON VS CPU TIME - JET 1



THIS GRAPH SHOWS THE RELATIONSHIP BETWEEN THE SELECTED VALUE OF EPSILON AND THE AMOUNT OF CPU TIME REQUIRED TO PROCESS THE DATA FOR JET 1.

Fig. 12a
Comparison plot for JET1

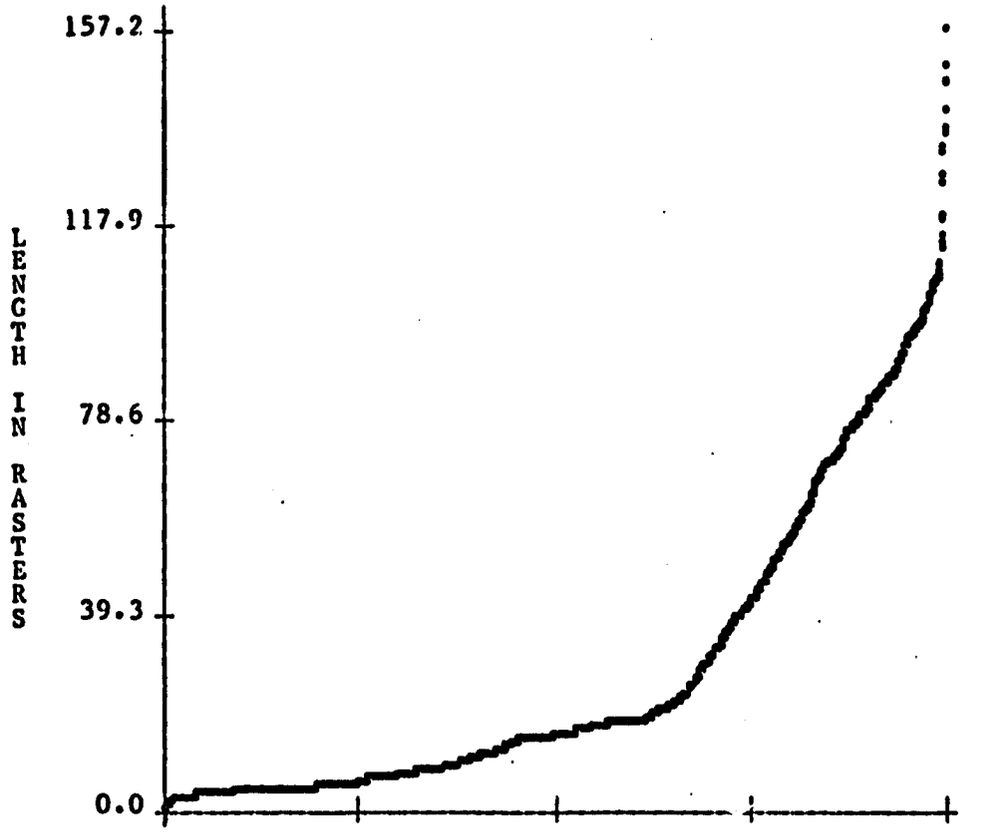
EPSILON VS CPU TIME - JET 3



THIS GRAPH SHOWS THE RELATIONSHIP BETWEEN THE SELECTED VALUE OF EPSILON AND THE AMOUNT OF CPU TIME REQUIRED TO PROCESS THE DATA FOR JET 3.

Fig. 12b
Comparison plot for JET3

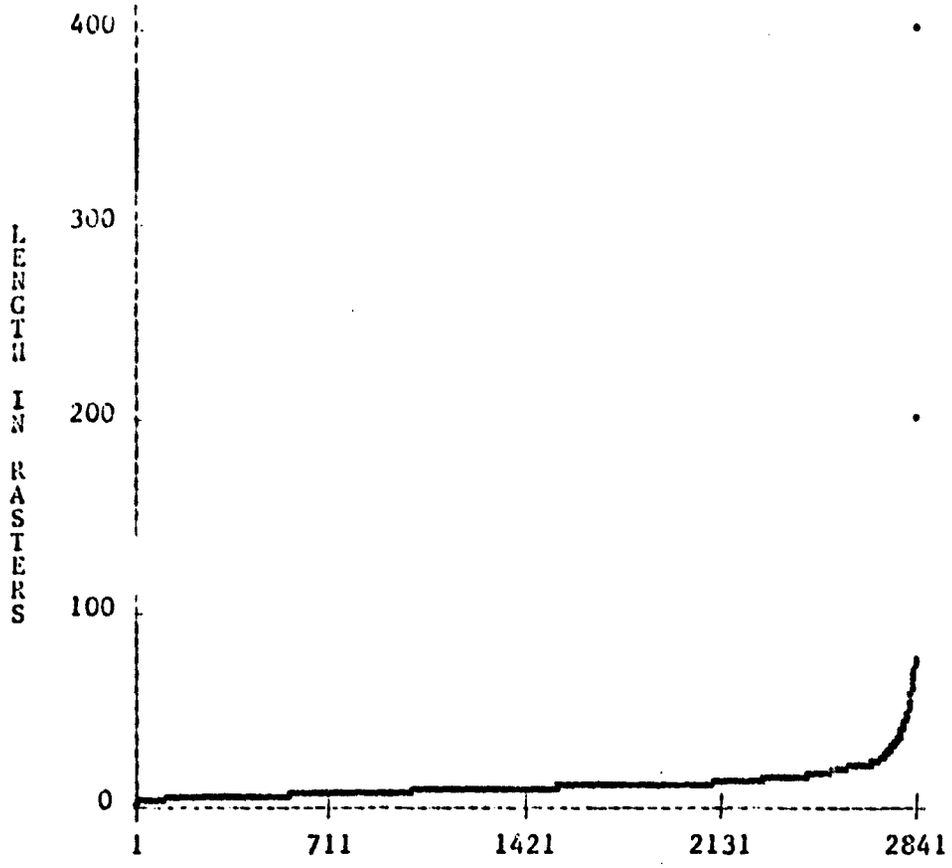
DRAW VECTOR LENGTH DISTRIBUTION FOR NASA JET 1



THIS PLOT REPRESENTS EACH DRAW BEING SET SIDE BY SIDE, IN ORDER OF LENGTH. THIS GIVES A GRAPHIC COMPARISON BETWEEN LONG AND SHORT VECTORS.

Fig. 13a
Comparison plot for JET 1

DRAW VECTOR LENGTH DISTRIBUTION FOR NASA JET 3



THIS PLOT REPRESENTS EACH DRAW BEING SET SIDE BY SIDE, IN ORDER OF LENGTH. THIS GIVES A GRAPHIC COMPARISON BETWEEN LONG AND SHORT VECTORS.

Fig. 13b
Comparison plot for JET3

JET 1 LEVEL 6 STUDY DATA

THE FOLLOWING ARE ALL LEVEL 6

EPSILON VALUE IN 1/200 INCH	END-POINT COUNT	IBM 370/168 CPU TIME IN SECONDS	PLOT TIME IN MINS	PLOT TIME IN SECS
0	1460	89	8	3
5	1300	71	8	20
10	1078	50	8	47
15	896	36	9	6
20	700	24	9	22
25	612	19	9	26
30	570	17	9	32
35	504	14	9	46
40	474	13	9	44
45	450	13	9	45
50	414	11	9	52
55	376	10	10	6
60	344	9	10	6
65	328	9	10	7
70	326	7	10	13
75	250	7	10	23
80	216	6	10	31
85	178	6	10	34
90	146	5	10	42
95	134	5	10	48
100	110	5	10	56
105	98	5	10	51
110	86	5	10	51
115	84	5	10	51
120	82	5	10	58
125	78	5	10	59
130	72	5	10	56
135	72	5	10	56
140	70	5	10	54
145	70	5	10	54
150	68	5	10	54
155	68	5	10	54
160	68	5	10	54
165	66	5	10	57
170	66	5	10	57
175	64	5	10	56
180	64	5	10	56
185	64	5	10	56
190	64	5	10	56
195	64	5	10	56
200	64	5	11	24
400	36	5	11	17
600	20	4	11	24
6000	4	5	11	46
LEVEL 1	1460		11	46
ORIGINAL DATA	1460		11	58

Fig. 14a

JET 3 LEVEL 6 STUDY DATA

THE FOLLOWING ARE ALL LEVEL 6

EPSILON VALUE IN 1/200 INCH	END-POINT COUNT	IBM 370/168 CPU TIME IN SECONDS	PLOT TIME IN MINS	SECS
0	880	35	5	33
5	868	34	5	36
10	828	32	5	0
15	780	29	5	50
20	724	25	5	57
25	654	21	5	59
30	606	19	6	5
35	548	16	6	19
40	486	14	6	29
45	454	13	6	38
50	408	11	6	38
55	372	10	6	53
60	334	9	6	54
65	314	8	6	57
70	304	8	6	58
75	270	7	7	3
80	250	7	7	10
85	244	7	7	10
90	224	6	7	7
95	204	6	7	16
100	196	6	7	17
105	186	6	7	15
110	180	6	7	18
115	174	6	7	18
120	174	6	7	18
125	170	6	7	19
130	170	6	7	19
135	164	5	7	21
140	156	5	7	24
145	150	5	7	26
150	148	5	7	24
155	146	5	7	24
160	140	5	7	29
165	140	5	7	29
170	138	5	7	29
175	136	5	7	29
180	132	5	7	28
185	132	5	7	28
190	126	5	7	30
195	126	5	7	30
200	126	5	7	30
400	48	5	8	16
600	18	4	8	39
6000	4	4	8	57
LEVEL 1	880		8	57
ORIGINAL DATA	880		15	6

Fig. 14b