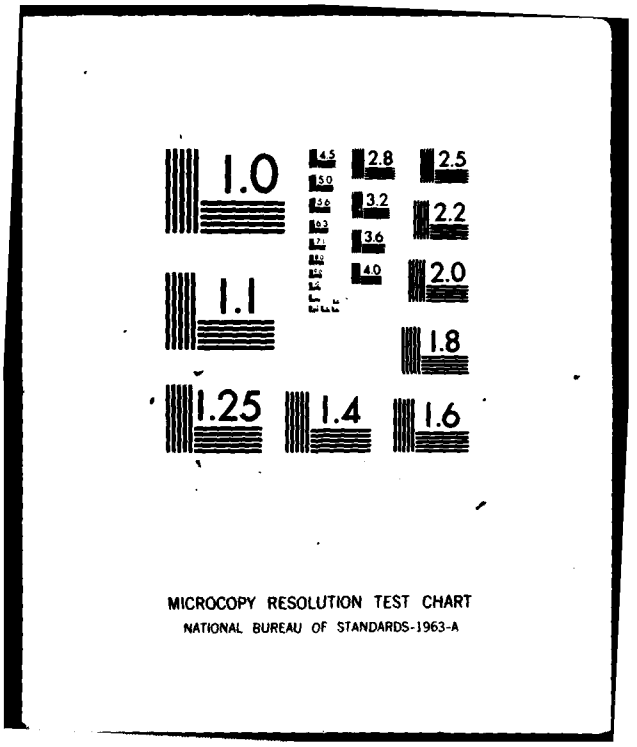
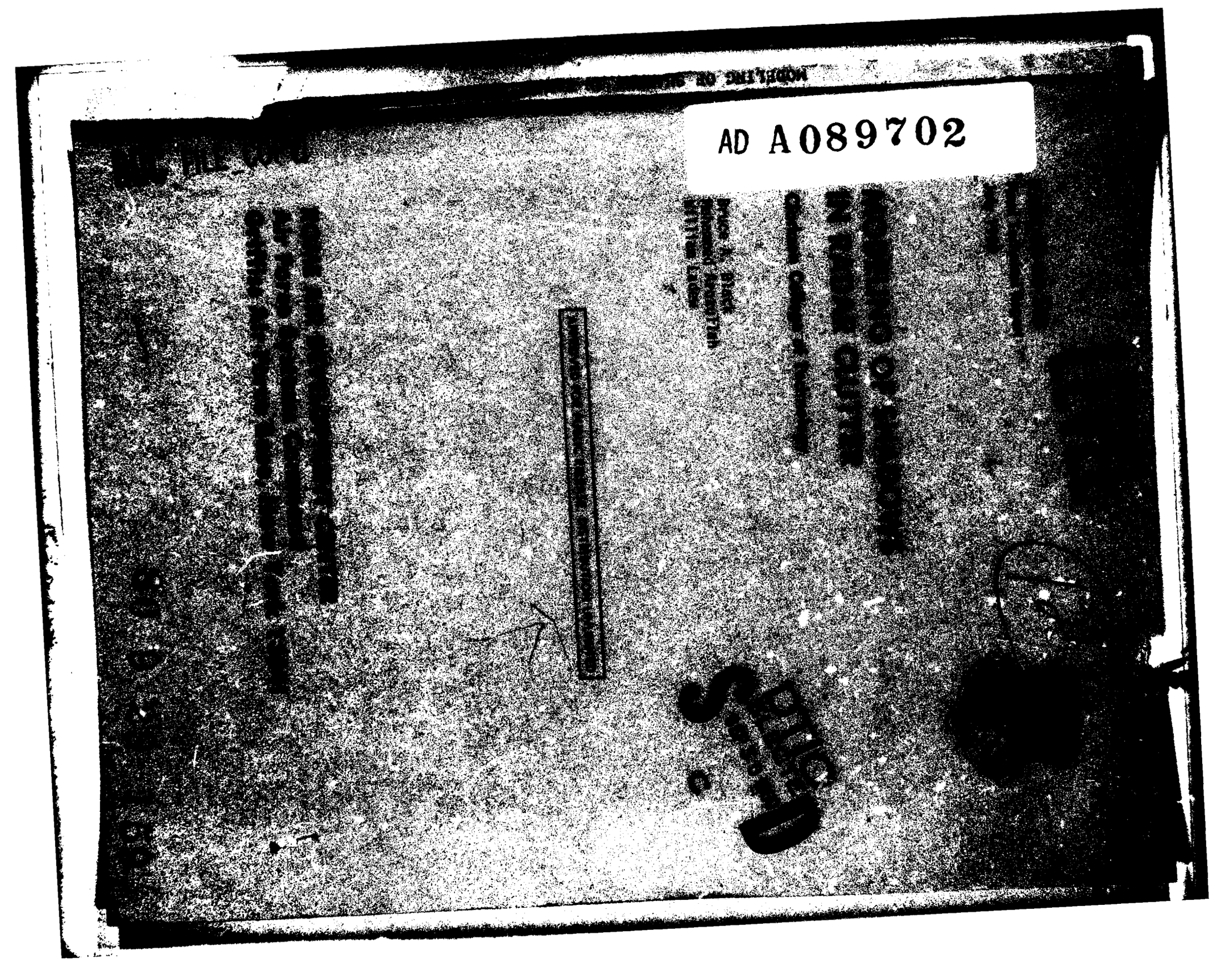
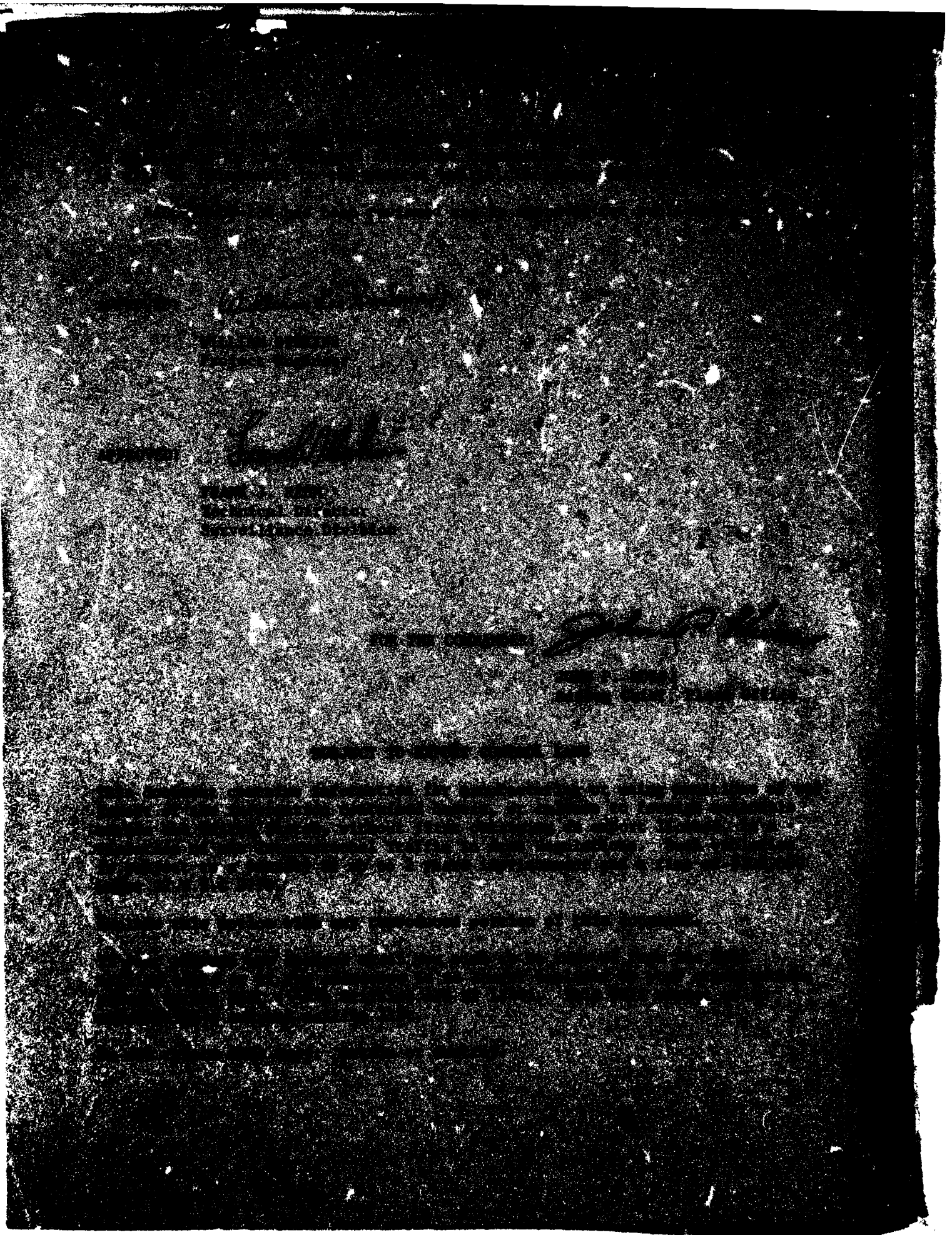
AD-A089 702 CLARKSON COLL OF TECHNOLOGY POTSDAM N Y DEPT OF ELEC--ETC F/G 17/9
MODELING OF SHADOWS IN RADAR CLUTTER,(U)
JUL 80 B A BLACK, M AROZULLAH, W LADEW F30602-78-C-0102
UNCLASSIFIED RADC-TR-80-238 NL


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A





UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
(18) 1. REPORT NUMBER RADC-TR-80-238	2. GOVT ACCESSION NO. AD-A089 702	3. RECIPIENT'S CATALOG NUMBER
(6) 4. TITLE (and Subtitle) MODELING OF SHADOWS IN RADAR CLUTTER		(9) 5. TYPE OF REPORT & PERIOD COVERED Final Technical Report, 1 Jun [illegible] - 31 Dec 79
		6. PERFORMING ORG. REPORT NUMBER N/A
(10) 7. AUTHOR(s) Bruce A. Black, Mohammed Arozullah, William Ladew		(15) 8. CONTRACT OR GRANT NUMBER(s) F30602-78-C-0102
9. PERFORMING ORGANIZATION NAME AND ADDRESS Clarkson College of Technology, Department of Electrical and Computer Engineering, Potsdam, NY 13676		(16) 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45061IPE (17) 11
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (OCTS) Griffiss AFB NY 13441		(11) 12. REPORT DATE Jul[illegible]80
		13. NUMBER OF PAGES 114
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same (12) 114		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

406944 →

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: William L. Simkins (OCTS)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Clutter
Ground Clutter
Shadowing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report presents the results of work done during June-December 1979 on investigating the applicability of the Forgy-Jancey and minimum spanning tree clustering algorithms to the problem of locating and characterizing shadowed regions in ground clutter. Certain preprocessing techniques were also investigated for reducing the amount of data prior to clustering. The two clustering algorithms were applied to both simulated and actual radar data, the latter provided by the Signal Processing —

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE UNCLASSIFIED (Cont'd)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

406944

Item 20 (Cont'd)

Laboratory at RADC. The algorithms were run on the IBM 360/65 at Clarkson College using programs either written at Clarkson or obtained from the pattern recognition package ARTHUR.

The Forgy-Jancey algorithm was found to be flexible and capable of handling large data sets at moderate cost, but was inconsistent in its ability to detect the "natural" clusters in unknown data. The minimum spanning tree algorithm was more reliable in this regard, but the ARTHUR implementation was found to be costly in terms of required storage. The report contains numerous figures illustrating the application of the two clustering algorithms to various configurations of data.

Clustering techniques appear to hold promise for examining the structure of unknown clutter data, but the results must be interpreted with care, since a natural clustering is not always found. The primary result of the work to date has been the installation of the programs on the Clarkson College computer and the familiarization of the researchers with their use and limitations. It now remains to apply these programs to a wider class of data.

CONTENTS

LIST OF FIGURES

LIST OF FIGURES, CONTINUED

EVALUATION

1. As future tactical and strategic sensors become more automated and unattended, their processing will increasingly treat the environmental scatter as information rather than "clutter" or interference. By inference from measureable quantities and statistics, the processor will recognize the existence of weather, chaff, discrete targets, homogeneous distributed areas, shadowing as opposed to specular reflection, and other environmental categories. This information will allow the system to adapt its waveform, energy budget, and detection/CFAR algorithms for optimum performance. This effort investigated algorithms which may be usable in such a recognition scheme.

2. Stress was placed on the strengths, weaknesses, and peculiarities of two promising algorithms when applied to the recognition of shadowed areas. The program has provided an excellent insight into the robustness and sensitivities to data format of each algorithm and the care required in the interpretation and use of algorithmic results. This report will be a useful reference in the development of clutter model and associated adaptive processing for future intelligent unattended systems.

William L. Simkins

WILLIAM L. SIMKINS
Project Engineer

MODELING OF SHADOWS IN RADAR CLUTTER

1.0 INTRODUCTION

This report presents the results of the work done during the period June-December 1979 on investigating the applicability of nonparametric clustering techniques to the extraction of features from radar clutter. The Forgy-Jancey [1] and minimum spanning tree [2] algorithms were investigated, with particular application to the problem of locating and characterizing shadowed regions in ground clutter. The significant result of the work is an exposition of the strengths and weaknesses of the two clustering techniques when applied to both simulated and actual radar returns. It is expected that the same strengths and weaknesses should appear in applications involving extraction of clutter features other than shadows.

The motivation for undertaking the present study of ground clutter is summarized in [3]. The need has existed since before the Second World War to extend the ability of radars to detect and identify targets of low cross-section in severe clutter. Initial efforts were concerned with increasing transmitter power, but for the past two decades the emphasis has been on development of the signal processing abilities of radar receivers [4]. The recent explosive developments in digital technology are expected to result in a new generation

of sophisticated signal-processing radars, able to take advantage of random-access memories and high-speed computational ability. The need to specify how such radars are to be designed and used provides a requirement for an improved understanding of the structure of backscatter clutter.

There are several indications that the present understanding of backscatter clutter is inadequate. First, clutter measurement efforts concerned with various terrain and cultural features have produced data of only limited consistency. In particular, it is not uncommon to find a 15 to 20 dB spread in measurements of intensity returns made at different times from supposedly identical terrain. Second, there has been only limited success in predicting backscatter from toptgraphical maps, even though the terrain relief, surface features, and ground cover are specified on the maps. Finally, radars employing signal processing systems, such as CFAR, that are based on particular clutter models [5] have frequently been found to perform more poorly than analysis based on the clutter models would suggest.

There are several possible reasons for the poor consistency of measurement data and the difficulty of predicting backscatter intensity from terrain descriptions. First, it is likely that backscatter is highly sensitive to surface parameters such as moisture, temperature, and wind

velocity, that are difficult to measure simultaneously with the radar return, and have not always been included as a part of the recorded "ground truth". Second is the possibility that the clutter returns vary with the parameters of the radar itself in ways that have not yet been clearly recognized.

Much of the radar signal processing effort to date has been based on the assumption that clutter intensity can be modeled by a random variable. The motivation to develop such a model is strong, because of the success of design methods using the Neyman-Pearson decision rule in connection with a gaussian or rayleigh model for thermal noise. Such decision-theoretic methods could be easily extended to detection of targets in clutter, if only the probability density function of the clutter intensity could be found. It is likely, however, that ground clutter is too heterogeneous to be represented by a single random variable, or possibly even by a single type of random variable. It is therefore necessary that an examination of the structure of radar clutter be carried out in a way that does not rely on knowledge of a probability distribution.

Methods for investigating the structure of unknown data have been developed in connection with research in taxonomy [1,6] and chemistry [7]. These methods are referred to as either <u>classification</u> or <u>clustering</u> methods, depending on whether or not a statistical structure is assumed for the unknown data. The classification methods assume that each data

entity belongs to one of several possible categories, each category characterized by a probability density function. Most classification algorithms require training on data whose classification by category is known. The training allows the algorithm to estimate the density functions corresponding to the various categories. The estimated density functions are then used in classifying future unknown data. In contrast, clustering methods do not assume that categories of data can be characterized by density functions. Instead, data entities are grouped into clusters by examining the distances between entities based on some suitable measure of distance. A detailed discussion and comparison of many available classification and clustering methods is presented in [6].

In the present research the view has been taken that ground clutter from arbitrary sources can not be reliably represented in terms of probability density functions [8]. Attention was therefore focussed on an investigation of the various clustering methods. The research has proceeded along three paths: first an investigation of the Forgy-Jancey clustering algorithm, second an investigation of the minimum spanning tree algorithm contained in the program package ARTHUR, and, finally, an investigation into some pre-processing techniques applicable to either of the clustering algorithms.

A description of the Forgy-Jancey algorithm along with the results obtained from applying it to a variety of simulated and actual clutter returns is presented in Section 2. Although the Forgy-Jancey algorithm is well-known, it was necessary to create our own programs, and to devise from scratch the necessary input-output and display routines. Investigation of the minimum spanning tree clustering method is described in detail in Section 3. This investigation began with acquisition of the ARTHUR program package [7] from the University of Georgia. The necessary job-control language was created, and the ARTHUR package compiled so that it is currently available on the Clarkson College computer. The minimum spanning tree algorithm was tested on some of the same simulated data as was devised for the Forgy-Jancey tests. Section 4 contains the results of some pre-processing efforts that were tried on actual clutter data. The pre-processing was motivated by the need to reduce the quantity of data to a managable amount for the ARTHUR package. The pre-processing is typical of the steps that would be taken in an actual clustering effort, and provided an indication of the amounts of CPU time, programming time, and costs that are involved in such steps. Section 5 contains an overall summary and the conclusions.

Sections 1, 2, and 5 were written by Bruce A. Black, Section 3 by William Ladew, and Section 4 by Mohammed Arozullah.

2.0 THE FORGY-JANCEY ALGORITHM

2.1 The Algorithm

This section discusses the operation of the Forgy-Jancey clustering algorithm, and the tests that were made on simulated and actual radar data to probe the strengths and weaknesses of the algorithm as a tool for identifying the structure of ground clutter from shadowed terrain. The computer program for the Forgy-Jancey algorithm was obtained from Fordon [9]. Several modifications to the algorithm were made to allow greater flexibility in manipulating, saving, and displaying the input and output data. The algorithm employed in all of the tests is due to Forgy [10]. Jancey's version [1] contains a heuristic to avoid local minima, but was not needed in the present application. The basic Forgy-Jancey program along with the modifications is presented in block-diagram form below. A listing of the important subroutines is presented in an Appendix.

Figure 2.1 is a flowchart showing the operation of the Forgy-Jancey algorithm. This algorithm forms clusters of data entities, where each data entity is a vector of NV components. The number NV of components in a data entity (number of variables), the number NE of data entities, the number NC of

clusters, and the coordinates of the cluster centroids must all be specified by the user. In this version of the algorithm the number NC of clusters cannot be modified by the program; the program can neither create nor delete clusters. In operation, the program will partition the NE data entities into NC clusters. A measure of the compactness of the clusters that is used as a figure of merit by the program is the total summed deviations. For a given partition of the data into clusters, this is the sum over all the entities of the distance between each entity and the centroid of the cluster to which it is assigned.

Forgy's algorithm begins with an initial set of NC cluster centroids. In the main loop of the program, each data entity is assigned in turn to the nearest cluster centroid. When all of the entities have been assigned, the cluster centroids are recomputed. Each iteration requires a complete pass through the data. It is easily shown that a pass through the main loop cannot increase the value of the total summed deviations, so that the algorithm must converge to at least a local minimum. In each pass through the main loop the program counts the number of entities whose cluster assignment is changed. When the number of changes falls below a user-supplied parameter MINREL, the program halts. Except where otherwise noted, programs were run with MINREL equal to zero, corresponding to complete convergence.

The distances between entities and cluster centroids required by the algorithm are computed by a user-supplied subroutine DIST. Because this routine is externally supplied, the distance function can be configured to meet the specific application. In particular, it is possible to base the distance calculation on a subset of the components of the data entities. Both euclidean and non-euclidean distance measures are allowed.

The program path for the Forgy-Jancey algorithm is shown in Figure 2.2. The program MAIN and the subroutines EXEC, USER, RESULT, KMEAN, and USROUT were all supplied or modified as a part of the present effort. The program supplies as outputs:

1. A raw membership list, listing cluster membership for each data entity.
2. A sorted membership list, giving for each cluster the sequence numbers of its members.
3. A sorted data file. The data entities themselves are sorted into groupings by cluster membership. This file can be saved on disk, thus allowing the entities forming a single cluster to be isolated for further processing.
4. The program prints the final centroid locations for each cluster, the number of data entities belonging to each cluster, and the value of the total summed

deviations.

2.2 Testing

For initial testing of the Forgy-Jancey algorithm, a set of 10,000 data entities representing a 100-point by 100-point "scene" was generated. Each data entity was a three-vector, (x,y,I), where x and y represent the position of the entity in rectangular coordinates and I represents the intensity of the radar return from position (x,y). Two rectangular regions were set aside in the 100x100 array to represent "shadows", one region occupying 651 points and the other occupying 121 points. The test scene is depicted in Figure 2.3.

For the initial test of the algorithm, the value 0 was given to the intensity component of data entities in the shadowed areas, while the value 1 was given to the intensity component of data entities in the unshadowed area. The algorithm was applied with the number of clusters NC set equal to two. The distance function DIST was set to compute the distance between two entities as the magnitude of the difference between their intensities. The output of the program is shown in Figure 2.4. The points marked 1 correspond to cluster 1, while the points marked with a dot correspond to cluster 2. Note that cluster 1 contains exactly the entities in the shadowed array.

A second clustering was carried out using only the entities in cluster 1 as data. This time the function DIST was

changed to compute the distance between two entities as the euclidean distance between their positions. The program was run with NC = 2. The output is shown in Figure 2.5. In this figure the points marked with a dot were not included in the input data. The points marked 1 and 2 represent the two clusters obtained by the algorithm. Note that these correspond exactly with the two shadows.

A second, more demanding, test of the Forgy-Jancey algorithm was made using random numbers as simulated intensity values. A program was written to produce a sequence of numbers having a rayleigh distribution. The distribution parameter sigma is entered as an input to the program. To establish the test data, 10,000 rayleigh-distributed numbers were generated with sigma = 1. These numbers were assigned as intensity values to the data entities in the 100 by 100 array. Next a sequence of rayleigh-distributed numbers was generated with sigma much greater than one. These numbers were added to the intensity values of all the data entities outside the two shadowed areas. Thus a set of test data was established to represent two shadows containing only rayleigh-distributed thermal noise in a larger region of brighter rayleigh-distributed clutter.

As in the previous test, the Forgy-Jancey algorithm was applied twice, once to separate the low-intensity shadows from the bright clutter background, and a second time to separate

the low-intensity cluster into individual shadows. The result the first clustering is shown in Figure 2.6. The bright clutter in this case has sigma set to 10,000. In the figure, the dark cluster is indicated by ones and the bright cluster by twos. Note that the dark cluster includes many points belonging to the bright region. In fact, the algorithm was unable in this case to locate the dark region; the program was instructed to find two clusters, and it did so by splitting the bright data half. The shadow data are simply included in the darker of the two clusters.

To provide a greater apparent separation between the dark and bright data, and to provide an example of the use of coordinate scaling, the program was modified to interpret intensity values in decibels. This was effected by employing the program USER to replace the intensity values I for each entity by their logarithms lnI as the entities were read into core. The result of a clustering done on the rayleigh data using this logarithmic transformation is shown in Figure 2.7. Perfect separation of the dark and bright regions was obtained. Figure 2.8 was obtained using only the dark cluster as data. The dark cluster was split on the basis of position into the two clusters shown by points marked 1 and 2.

The gap between the dark and bright intensity values can be reduced by changing the value of sigma used in generating

the bright rayleigh data. To create a more demanding exercise for the algorithm, the data set was recreated with sigma equal to 50. Logarithmic scaling of the intensity values was again used. Figure 2.9 shows the result of clustering for light and dark intensity regions. Points in the dark cluster are indicated by ones, and points in the bright cluster are indicated by twos. Note that a number of bright points have been misclassified by being included in the dark cluster. It would be expected that statistical variation among the intensity values of the non-shadow data would lead to some "bright" points with intensity values that are in fact as low as the values of the intensities of the shadow data. The number of misclassified points shown in the figure corresponds roughly with the number that would be expected on statistical grounds.

Problems with convergence of the Forgy-Jancey algorithm were encountered for the first time in generating the clusters shown in Figure 2.9. To generate this figure the algorithm was halted using MINREL = 100; i.e. on the first pass for which fewer than 100 data entities were reassigned. Figure 2.10 shows the result of running the algorithm all the way to convergence, a process which required nearly five minutes of CPU time. It is interesting, and somewhat surprising, to note that the dark cluster of Figure 2.10 includes even more misclassified "bright" points than does the dark cluster of

Figure 2.9.

The dark cluster of Figure 2.10 was used as the data base for the second clustering pass, in which individual shadows were separated. Figure 2.11 shows the result of clustering by position with NC equal to two. In this case the two shadows were correctly separated, but each cluster consists of a dense core, corresponding to the actual shadow, surrounded by a sparse "halo" of points. In an attempt to separate the shadows from their halos, several clustering passes were made on the data with NC greater than two. Figure 2.12 shows the result of clustering by position when NC equals three. The points marked zero are excluded from the data set. The three clusters are indicated by points marked 1, 2, and 3, respectively. (Cluster two is in the upper right-hand corner of the figure.) Note that the actual shadows still form the cores of separate clusters, but that both are still surrounded by halos. Figures 2.13, 2.14, and 2.15 are the results of clustering with NC respectively equal to four, five, and six. Figure 2.16 shows the case with NC equal to seven. One of the shadows has been split into two clusters, marked 1 and 7 in the figure. This is an undesirable result, from which two lessons can be learned: first that the Forgy-Jancey algorithm will find as many clusters as it is instructed to, whether or not these clusters are meaningful, and second, that increasing the number of

clusters is not an effective method for removing the halo from a dense core of points.

Identifying the value of NC that corresponds to the number of clusters actually present is difficult when dealing with unknown data, especially data of high dimensionality that cannot be readily plotted. Jancey [1] suggests beginning with NC = 1 and running the algorithm repeatedly with NC = 2, 3, etc. As the number of clusters increases, the value of the total summed deviations obtained at convergence will decrease. As "natural" clusterings are obtained, there should be significant drops in the value of the total summed deviations. An example of this effect is presented in [1]. Figure 2.17 shows the values obtained for total summed deviations in the above rayleigh example when NC = 1 through 7. Note that the most significant drop occurs when NC is raised from one to two.

2.3 Removal of Outliers

Fordon [9] suggests a modification of the Forgy-Jancey algorithm to allow rejection of data entities that are too far from any of the cluster centroids. A data entity can be isolated if it satisfies the relation

$$|D_{avg} - D_{min}| < (OUTLYR) \times D_{avg},$$

where D_{min} is the distance between the entity and the nearest cluster centroid and D_{avg} is the average of the distances between the data entity and each of the cluster centroids. The parameter OUTLYR is assigned a value between zero and one by the programmer. A value of zero results in no entities being isolated, while a value of one results in every entity being isolated. The main loop of the Forgy-Jancey algorithm was modified so that whenever an isolated entity is encountered it is designated as the centroid of a new cluster. This allows the program to increase the number of clusters. A provision was included to limit the number of clusters to a programmer-specified maximum, since CPU time was found to increase dramatically with the number of clusters. In interpreting the results of a clustering run, the programmer has the option of disregarding all newly-formed clusters as representing outliers; i.e. sets of anomalous entities that lie outside the known clusters, or of accepting all of the clusters in the expectation that the program has found a more natural

number of clusters than the programmer originally specified.

An example of a clustering run using the modified Forgy-Jancey algorithm is shown in Figure 2.18. The data set is the 100x100-point array depicted in Figure 2.3, with each entity having an intensity value of zero or one. Three outliers were created, having intensity values of zero, and lying outside the simulated shadows. The Forgy-Jancey algorithm was run twice, once to separate out the low-intensity points and once to cluster these points into individual shadows. Figure 2.18 is the result of the second run. To generate this figure, the program was started with NC = 2 and OUTLYR = 0.6. The program responded by creating two additional clusters, identified in the figure by the numbers 3 and 4. Although the program successfully located the three outlying points, it should be noted that the value used for the parameter OUTLYR is fairly critical. The program was unable to locate all of the outliers when run with OUTLYR = 0.4.

2.4 Testing of Actual Radar Data

A tape containing a segment of a PPI display was provided by the Signal Processing Laboratory at RADC, courtesy of William L. Simkins, Jr. The tape contains data in essentially the same format as the simulated data described above: each data entity is comprised of a set of three components representing x and y position and intensity. The data were created by quantizing a photograph of the original CRT display, and in consequence the intensity scale of the data is compressed by the responses of the CRT phosphor and the film. Figure 2.19 depicts one quadrant of the radar data. The grey scale used in the figure consists of the symbols (W,*,A,I,+,-,.,blank), in order from highest to lowest intensity. Figure 2.20 shows the low-intensity regions of the quadrant of data. The set of low-intensity entities was used as a data base for the Forgy-Jancey clustering algorithm. Numerous runs were made, using various values of input parameters. A typical output is shown in Figure 2.21. To generate this output twenty-one initial cluster centroids were chosen, corresponding roughly to the centers of the clusters visible in Figure 2.20. The outlier parameter was set to OUTLYR = 0.9, and the algorithm was stopped when fewer than 33 entities changed cluster membership (MINREL = 33). The program ran for nearly four minutes and generated a total of

twenty-five clusters.

Examination of Figure 2.21 reveals the twenty-five clusters, designated by the numbers zero through nine and the letters A through O. A close examination of this Figure reveals that many of the clusters found by the program do not correspond to the natural clusters evident in Figure 2.20. In particular, attention is called to cluster K, which is composed of the adjacent parts of four natural clusters, clusters C and O, which split a single natural cluster, cluster 2, which contains some entities that should properly belong to cluster 3, and cluster D, which encroaches on the territory of clusters C and O.

In processing the actual radar data, the Forgy-Jancey algorithm had a great deal of difficulty in locating the natural clusters. It is likely that this difficulty stems from two sources: the highly irregular shapes of the natural clusters, and the fact that the natural clusters vary greatly according to size. It is believed that the total-summed-deviations figure of merit used by the algorithm tends to emphasize the compactness of clusters. Thus whenever a cluster becomes too long and narrow, the algorithm will split it into two smaller clusters. The procedure used by the program for generating new clusters tends to result in clusters of about equal size. Hence the difficulty in dealing with a data set that contains clusters as small as cluster 7 and as

large as cluster 6.

2.5 Conclusions

The Forgy-Jancey clustering algorithm is a powerful tool whose principle advantages seem to be its generality and simplicity. The generality results from the ability of the algorithm to handle data sets of essentially any size, consisting of data entities of any dimension. The number of clusters that the algorithm can handle is also essentially unlimited, and the programmer is free to prescribe arbitrary coordinate scaling and an arbitrary distance measure. The simplicity of the algorithm is helpful when modifications are needed. In carrying out the present study it was necessary to add several input and output routines, and to modify the main loop of the algorithm to incorporate the ability to generate new clusters.

The drawbacks to using the Forgy-Jancey algorithm are clearly illustrated in Section 2.4, particularly in Figure 2.21. The Forgy-Jancey algorithm is known to be capable of finding clusters that bear no relation to any natural clustering [10], and must be used with extreme care on unknown data that can not be visually inspected. Some of the difficulty in locating the natural clusters in Figure 2.21 can be attributed to the procedure used for generating new clusters. Without this procedure, however, it is necessary to specify the number of clusters in advance. Jancey's procedure

for monitoring the total summed deviations offers one method by which the number of natural clusters might be determined.

3.0 MINIMAL SPANNING TREE ALGORITHM

3.1 Introduction

In this section we describe the minimal spanning tree algorithm [2] for clustering data. This graph-theoretic method differs greatly from the Forgy-Jancey method described in the previous section. We used an implementation of the MST (minimal spanning tree) algorithm in our investigation obtained in the ARTHUR subroutine package. ARTHUR is a general package of data manipulation and analysis subroutines that was obtained from the University of Georgia. The MST subroutine in the package is called TREE and requires use of a distance subroutine DIST, plus input and output utility subroutines also contained in the package. ARTHUR will be discussed further in this section, along with the description and discussion of TREE. Before we examine how TREE clusters our data, we must first discuss the MST algorithm and how it can be used for clustering.

3.2 Minimal Spanning Trees

A minimal spanning tree is an undirected graph that includes all the given data points, has no loops or cycles, and has a minimal distance over all the edges in the tree. The MST can use any distance metric to describe how close (or similar) the data points are, but the important point is that the resultant tree will have the smallest total distance of any of the possible spanning trees. The choice of the distance metric makes this method general for any type of data; however, in our study we only considered euclidean distance for x-y data. This is because we were only doing x-y clustering in this phase of our study. A different form of metric could be used for some other clustering, such as intensity clustering. The ARTHUR MST algorithm clusters data by finding "inconsistant" edges in the MST; the tree itself does not contain clusters, but the algorithm points out edges that when deleted would give a natural clustering. This method of forming clusters eliminates one of the major problems encountered in using the Forgy-Jancey algorithm, namely that we must tell it how many clusters to find, but the MST clustering algorithm then needs some form of information describing what an inconsistant edge is. The factors we use to determine whether an edge is inconsistant depend on the average length of the neighboring edges, where we can specify how deep along the tree a neighboring edge is. We

also need to know how many standard deviations from the average the edge is question is. The edge is cut based on these parameters, if the edge is more than the specified number of standard deviations from the average.

This method of forming clusters is general in as much as the distance metric is general and the clustering criterion is relative to distances in the tree. This method of choosing clusters based on inconsistant edges in the minimal spanning tree will work well on many types of data, as shown by Zahn [2]. After the tree is formed, the inconsistant edges are found and deleted and the connected groups that are left are the clusters. This means that we can form any number of clusters, not just a predetermined number or range of clusters. Also, we can find the tree just once, and try several different sets of clustering parameters on it, without the expense of finding the MST each time.

3.3 ARTHUR and the MST

We used the TREE routine in ARTHUR to cluster data using a minimal spanning tree. TREE can be given three parameters, the DEPTH of the neighbors, a FACTOR used in finding inconsistancies, and a SPREAD of the number of standard deviations allowed before an edge is inconsistant. An edge is inconsistant if its length is greater than FACTOR times the length of the neighboring edges, or if it is more than SPREAD standard deviations longer than the average of the neighboring edges, where neighboring edges are all edges within DEPTH edges from the edge being tested [7]. A copy of the ARTHUR documentation is contained as an appendix in Fordon [9]. In using TREE, the data must be input to ARTHUR using one of its utility routines. The distance matrix containing the distance between every pair of data points must be found and stored using the DIST utility routine, and then TREE is called. Part of a test run of ARTHUR and TREE is shown in Figure 3.1, showing how TREE outputs the MST as a list of nodes and neighbors. The resulting clusterings are shown in Figures 3.2, 3.3, and 3.4, as lists of entities (they call them patterns) belonging to the various clusters. These clusterings were all done on the same MST in the same run of ARTHUR; recalculation of clusters with different parameters does not require recalculation of the distance matrix or the MST. The main

limitation is that in examining N data entities, we require on the order of N^2 units of storage for the distance matrix, and of course the amount of time to compute all the distances. This is significant in that if we double our number of points, we need four times the storage and computer time to do the same processing. For example, a 10 by 10 picture has 100 points, and needs 10,000 words of storage for its distance matrix; if we increase the picture size to 100 by 100, we have 10,000 points and need 100,000,000 words to store the distance matrix. So to enlarge a picture by a factor of 10 requires a 10^4 increase in memory.

A temporary limitation of TREE is that it only prints a description of the MST in terms of the nodes and edges as in Figure 3.1, and clustering cannot be displayed other than in a list of cluster members. The cluster membership is not output in an ARTHUR format file so we cannot display the clustering using one of ARTHUR'S plotting routines.

3.4 Actual Clustering Using TREE

In order to test the clustering ability of TREE, we used data previously used in the Forgy-Jancey testing. We took part of one of our rayleigh data files, namely cluster number 2 from Figure 2.11, and converted it to ARTHUR data format. This picture consists of 218 data entities, shown as 1's in Figure 3.5. We want to isolate the obvious group of 121 entities from the surrounding outlier entities. We ran ARTHUR on this data, found the distance matrix using euclidean distance, and called TREE. TREE found the minimal spanning tree, shown in Figure 3.6, and clustered the data according to several different sets of parameters. The clustering formed by the default parameters of DEPTH=3, FACTOR=2 and SPREAD=0, as in Figure 3.2, did eliminate some of the outliers from the desired cluster, but left 207 entities in the cluster, which is not very close to our desired 121 entities in the cluster. These parameters did not form many extra clusters; only five clusters were found, seperating only 11 outlier entities from the main cluster.

By trying only a few different combinations of parameters, we made the clustering much better. Using DEPTH=1, FACTOR=1, and SPREAD=1, we found 33 clusters, shown in Figure 3.3. This clustering put 132 entities in the main cluster, eliminating almost all the outliers. It removed 86 of a possible 97 outlier entities from the main cluster. Using the same

parameters as before, but setting SPREAD=0.5, we obtained the best results as shown in Figure 3.4. We found 37 clusters, and the main cluster had only two outlier points included in it, for a total of 123 points in that cluster. We were therefore able to eliminate 95 of 97 outlier entities. This clustering is drawn out in Figure 3.7, shown as the remaining edges in the tree, with all of the clusters circled to identify them.

This clustering differs from possible clusterings by the Forgy-Jancey algorithm in several ways. First, it cannot split up dense clusters because it attempts to cut edges only where they are not dense. Also, since this is a single pass clustering, the decisions are made once, not iteratively updated. This means that there are no convergence problems with the resulting waste of computer time. Again, since we need not choose in advance how many clusters to find, we eliminate the problem of re-running the program with different numbers of clusters until an optimal number is found. Instead, the program will find the appropriate number of clusters, given the parameters for cutting inconsistant edges.

3.5 Conclusions

The TREE routine can create clusters from noisy data, as shown in Figure 3.7, without problems such as the splitting of desired clusters found with the Forgy-Jancey clustering. However, TREE in its present form is relatively expensive in both time and storage required. Part of this expense is due to the overhead involved in using the ARTHUR subroutine package, but using ARTHUR is worthwile because it gives us a powerful and flexible system of programs, rather than a stand-alone MST program. ARTHUR provides us with its flexible input routine, and has built-in routines to do scaling on the data, and even can orthagonalize data that is not orthagonal. It also has a histogram plotter to check the data distribution, three different data plotting routines, and several classification subroutines besides TREE, such as a bayesian classifier and a hierarchical classifier. A stand-alone MST program might be more efficient, but would take substantial programming effort and would sacrifice all those valuable subroutines. The space expense is due to the N^2 size of the distance matrix, which presently limits maximum size of the data set that can be analyzed. However, ARTHUR can store data in both memory and disk dynamically by swapping parts of the data set between core and disk, saving memory at the cost of more disk I/O operations.

TREE is more flexible than Forgy-Jancey because it does not have to told how many clusters to find. In addition, Zahn [2] suggests a variety of alternative schemes for dividing a minimum spanning tree into clusters, each of which might be useful for a different pattern of data entities. If some of these schemes were to be incorporated into the MST subroutine, it would significantly enhance the usefulness of the minimum spanning tree as a clustering technique.

4.0 PREPROCESSING TECHNIQUES

4.1 Introduction

In this section we describe a few preprocessing techniques used on the data before applying them to the clustering algorithms discussed in the preceding sections. The preprocessing techniques used are:

a) point-to-point local variance

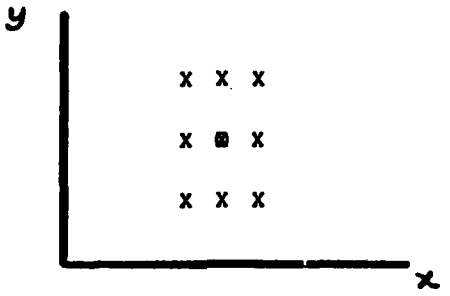
b) point-to-point local gradient

c) radial derivative

Of all these preprocessing techniques the radial derivative is the most promising one. These techniques were applied to the compressed digitized radar data provided to us by RADC and shown in Figure 2.19. In each case we applied the technique and examined the output data for clues for identification of bright and dark areas. Results obtained for each technique are discussed individually in the following. For each of these techniques we used half of the picture provided as otherwise the amount of data to be processed was too large.

4.2 Point-to-Point Local Variance

The idea behind this method is that probably the variation of the intensity of the radar return will be very small inside a shadow and it will be large in the peripheries of shadows. For each point in the picture, variance was calculated by considering nine neighboring points on the left-right and above and below this point as shown below



We first calculated the average intensity $\bar{I}$ for these nine points. Then we calculated the local variance by using the formula

$$\text{Variance} = \sum_{j=1}^{9} (I_j - \bar{I})^2/9$$

where I_j indicates intensity at the j-th point.

Then these variances were plotted on an eight level grey scale using the following symbols: (W,*,A,I,+,-,.,blank) in descending order of the value of variance (W represents the highest and blank represents the lowest values of the

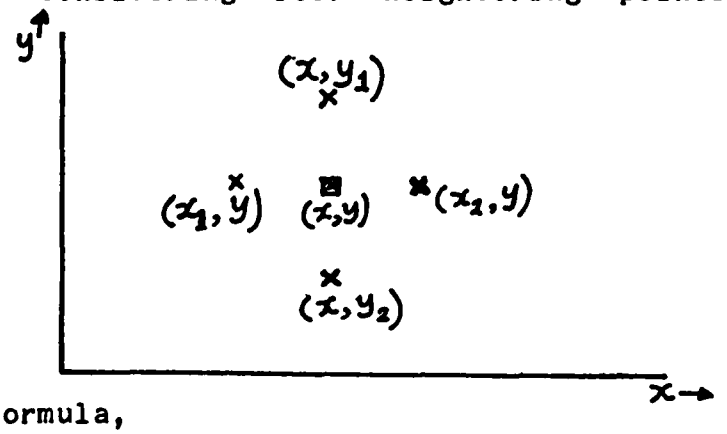
calculated variances). This plot of the variances is shown in Figure 4.1.

An examination of this figure reveals that the shadowy areas do have small variances and the edges have large variances. However; some areas with constant but high intensities also have small variances. Hence it appears that the local variance alone is not sufficient for clear cut idetification of shadows. Other information like intensity and edge information may have to be considered together with the local variance. Further, with this local variance method all directional information i.e. the variation of intensity with distance is lost.

For an 128 x 128 point data set computation of local variances needed 33 seconds of CPU time. The storage requirement was 2×128^2 bytes, 128^2 bytes for the original array and 128^2 bytes for the calculated variances. We used 200 kilobytes of storage and it was sufficient. Expense for one run was $2.22.

4.3 Point-to-Point Local Gradient

The idea behind using local gradient was that there should be higher variation of intensity on the edges of shadows than inside them. For each point of the array local gradient was calculated by considering four neighboring points as shown below.



By using the formula,

$$\text{local gradient} = \sqrt{(I_{(x_2,y)}-I_{(x_1,y)})^2+(I_{(x,y_2)}-I_{(x,y_1)})^2}$$

where $I_{(x,y)}$ represents the intensity at the point (x,y). The calculated local gradients were plotted over an eight level grey scale using the same symbols as in the case of local variance. This plot is shown in Figure 4.2.

An examination of Figure 4.2 shows that in the case of local gradient also shadows and high constant intensity areas have low gradients. The edges, however, showed larger gradient. Hence the local gradient above also can not be used to identify shadows conclusively. Other information like the

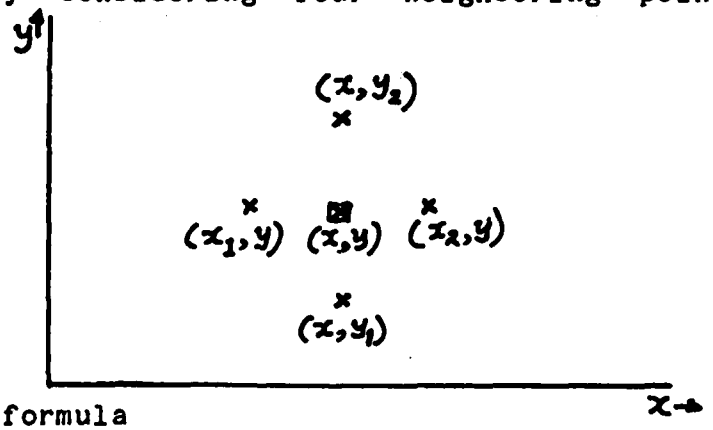
intensity, needs to be considered together with gradient.

For a 128 x 128 array of data points it took 25 seconds of CPU time to compute gradients. Storage requirement was also 2 x 128^2 bytes. An allocation of 200 kilobytes was sufficient. Expense for one run was $1.91.

4.4 Radial Derivative Method

In this method derivatives were calculated along radial lines along those of the radar beam starting from the location of the radar as shown by the X in Figures 4.3 and 4.4. The idea behind using radial derivatives is that as the radar beam travels in radial directions, the variation of intensity in the radial direction will be a better measure of the reflective nature of the objects illuminated by the radar. Also any object intercepting the radar beam will produce a shadow in the radial direction.

For each point of the data array radial derivative was calculated by considering four neighboring points as shown below

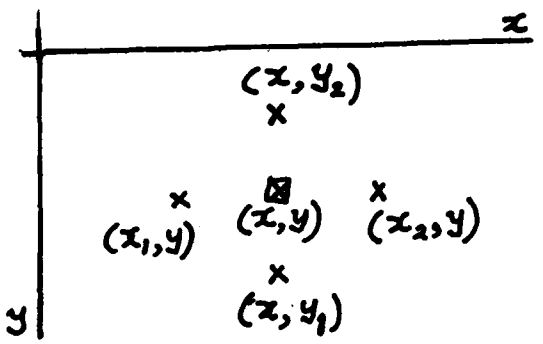


by using the formula

$$\text{Radial Derivative} = \frac{(I_{(x,y_2)} - I_{(x,y_1)})|y| + (I_{(x_2,y)} - I_{(x_1,y)})|x|}{\sqrt{2(x^2+y^2)}}$$

for the points in the first quadrant. This formula has to be

modified for other quadrants. For example for the points in the fourth quadrant as shown, the radial derivative is given by



$$\text{Radial Derivative} = \frac{(I_{(x,y_1)} - I_{(x,y_2)})|y| + (I_{(x_2,y)} - I_{(x_1,y)})|x|}{\sqrt{2(x^2+y^2)}}$$

These radial derivatives were plotted on a seven level grey scale using the following symbols.

```
        # < -30
-30 =< @ < -15
-15 =< - < - 5
 - 5 =< Blank < 5
   5 =< + < 15
  15 =< A < 30
  30 =< W
```

the plots are shown in Figures 4.5 and 4.6.

An examination of Fig. 4.5 reveals that:

1. All dark regions have low radial derivative

2. Range marker showed up.

3. Shadows were marked by the following characteristics: in front of the shadows we observed regions of high radial derivatives (cause of shadow) followed by regions of low radial derivatives (edges of shadow) further followed by regions of lower radial derivatives (shadow). It was also observed that shadows were followed by regions of higher (increasing radial derivative (elevated region beyond the shadow). We also observed regions of negative radial derivative before shadows and regions of positive radial derivatives after shadows.

4. For uniform, but high intensity regions (non-shadows) we observed regions of positive radial derivatives before them and regions of negative radial derivative after them. Of course, these uniform intensity areas themselves have low radial derivative like shadows. Thus the signs of radial derivative before and after a region of low radial derivative may be useful in distinguishing shadows from other low derivative regions.

5. However, we found some exceptions to these observations also. Radial derivative seems to be the most promising of all the preprocessing techniques test by us for detection and identifying shadows.

However, in order to put confidence into the use of radial derivative for detection and identification of shadows, more testing with data with a corresponding topographical map is

necessary. We did not have the topographical map to verify our conclusions. Also, the radial derivative makes use of radial changes and it does not take into account of side-to-side changes. Using both radial and local derivatives may improve the decision making.

Originally we plotted the radial derivative information on a seven point grey scale. However this plot looked too cluttered with symbols. Hence to have a clearer picture we replotted the radial derivatives on a five level grey scale. The derivative values represented by +, blank, and - were replaced by blanks. This plot is shown for the upper quadrant in Figure 4.7. The regions of low intensity depicted in Figure 2.20 are shown dotted in Figure 4.7. Using the plot we examined the regions of high but relatively constant intensity preceeding shadows to test them as precursors of shadows. We noticed that for these regions of high but constant intensity before shadows there were regions of positive derivative before them and regions of negative derivatives after them. This indicated that these regions are some sort of precursors of shadows. (These areas were selected by hand. This selection may be done by a computer program. However these programs may be quite expensive and it is easier to select them by the sight.) Figure 4.8 shows the same information as Figure 4.7, but with the precursors indicated by numbers 0-9 and letters

B-U.

Then we tried to distinguish these groups of bright regions to see if these belong to more than one type of regions. Our investigation did not bring out any significant differences among the various groups.

Next we attempted to correlate these precursor regions to the shadows. We did observe that there were shadows following many of these precursors. This gave confidence to our conclusions.

Finally, we used the spanning tree algorithm to cluster the dark areas (possible shadows). We did not obtain any accurate division of these areas into individual shadows.

For a 128 x 128 point picture it took 26 seconds of CPU time to calculate the radial derivatives. Storage requirement was 2×128^2 bytes. We were alloted 200 kilobytes and it was sufficient. Expense was $1.95 per run.

4.5 Conclusions

Out of the three techniques of preprocessing used, the radial derivative technique seems to be the most promising for detection and identification of shadows. However, combinations of these techniques may prove to be more useful under some conditions. It is necessary to have a topographical map corresponding to the data used to verify our conclusions.

4.6 Note

We, at Clarkson, wrote, debugged and tested the programs used for the grey-scale plots, calculation of local variances, local gradient and radial derivatives. These programs are specifically written for these types of data (type of numbers, data x-y, rather than radial format). These programs are now available on disk and also on tapes for further use on this type of project.

5.0 CONCLUSIONS

This report has presented the results of a comparison of nonparametric clustering techniques and their use in examining radar clutter from shadowed terrain. Successful techniques for extracting features from clutter will aid in understanding the underlying structure of the clutter and in turn will lead to the development of useful clutter models. The Forgy-Jancey and minimum spanning tree algorithms were tested on both simulated and actual radar data. Some preprocessing techniques were also tested on the actual radar data in connection with the minimun spanning tree algorithm. The Forgy-Jancey algorithm was found to be a very flexible tool, handling large multidimensional data sets with minimal cost in CPU time and storage. Unfortunately the performance of this algorithm was found to be inconsistant in terms of finding the "natural" number of clusters. In general, repeated applications of this algorithm are necessary in order to find the optimum number of clusters and parameter values. A more detailed discussion of the conclusions pertaining to the Forgy-Jancey algorithm can be found in Section 2.5.

The minimum spanning tree algorithm was found to be more robust in its ability to detect a natural clustering; it was

never observed to split a natural cluster as was the Forgy-Jancey algorithm. The ARTHUR implementation of the minimum spanning tree provides flexibility in clustering, since once the tree is found, the data can be clustered repeatedly without recomputing the tree. Finding the tree, however, turned out to be costly for large data sets, since the ARTHUR implementation requires calculation and storage of the distance between every pair of data entities. Alternative procedures for computing the tree are available, however, as are alternative procedures for clustering the data from the tree. A detailed discussion is presented in Section 3.5.

A discussion of the preprocessing techniques examined is presented in Section 4, with the conclusions in Section 4.5. The preprocessing was motivated by the need to reduce the amount of data provided to the minimum spanning tree algorithm, but was found to be a useful technique in its own right for extracting features from clutter. Computation of the radial derivative was the most revealing of the techniques studied for locating and defining the edges of shadows.

Clustering techniques appear to hold great promise for examination of the structure of clutter returns, provided that the techniques are applied with some care and common sense. A primary result of the work to date has been installation of the relevant programs on the Clarkson College computer, and the

familiarization of the researchers with their use. The next step would be a detailed application of these programs to actual radar data, including an examination of a wider variety of clutter properties.

APPENDIX: PROGRAM LISTINGS

This appendix contains the listings of the programs developed and used in this project. Figure A.1 is the comments section from subroutine EXEC, and describes the input format for Forgy-Jancey data. Figure A.2 is the input specification for the outlier Forgy-Jancey subroutine EXEC3. Figure A.3 is the subroutine EXEC to go along with the comments in Figure A.1. Figure A.4 is subroutine EXEC2 which allows outputting of the sorted data. Figure A.5 is subroutine EXEC3, the outlier calling subroutine. Figure A.6 (Parts 1-3) is subroutine KMEAN, the original Forgy-Jancey clustering routine. Figure A.7 (Parts 1-3) is outlier subroutine KMEAN2. Figure A.8 is the RESULT subroutine, which outputs the results of KMEAN. Figure A.9 is subroutine RSULT2, which allows output of the sorted data array. Figure A.10 shows the driver routine MAIN, function DIST, and subroutines USER and USROUT for a sample Forgy-Jancey clustering run. Figure A.11 shows MAIN and DIST for a sample outlier run, and also another version of DIST used in intensity clustering. Figure A.12 is subroutine RAY, our rayleigh random number generator. Figure A.13 is the program that created the rayleigh data set with sigma = 10,000. Figure A.14 is the program that created the rayleigh data set with sigma = 50. Figure A.15 is the plotting program that produced

all of our 100 x 100 and 128 x 128 point cluster plots. Figure A.16 is the grey scale plotting program used to generate all grey scale plots without overprinting. These programs were all either written at Clarkson, or installed and modified at Clarkson to suit the purposes of the project. For a description of ARTHUR, see Section 3, and the ARTHUR program documentation [7].

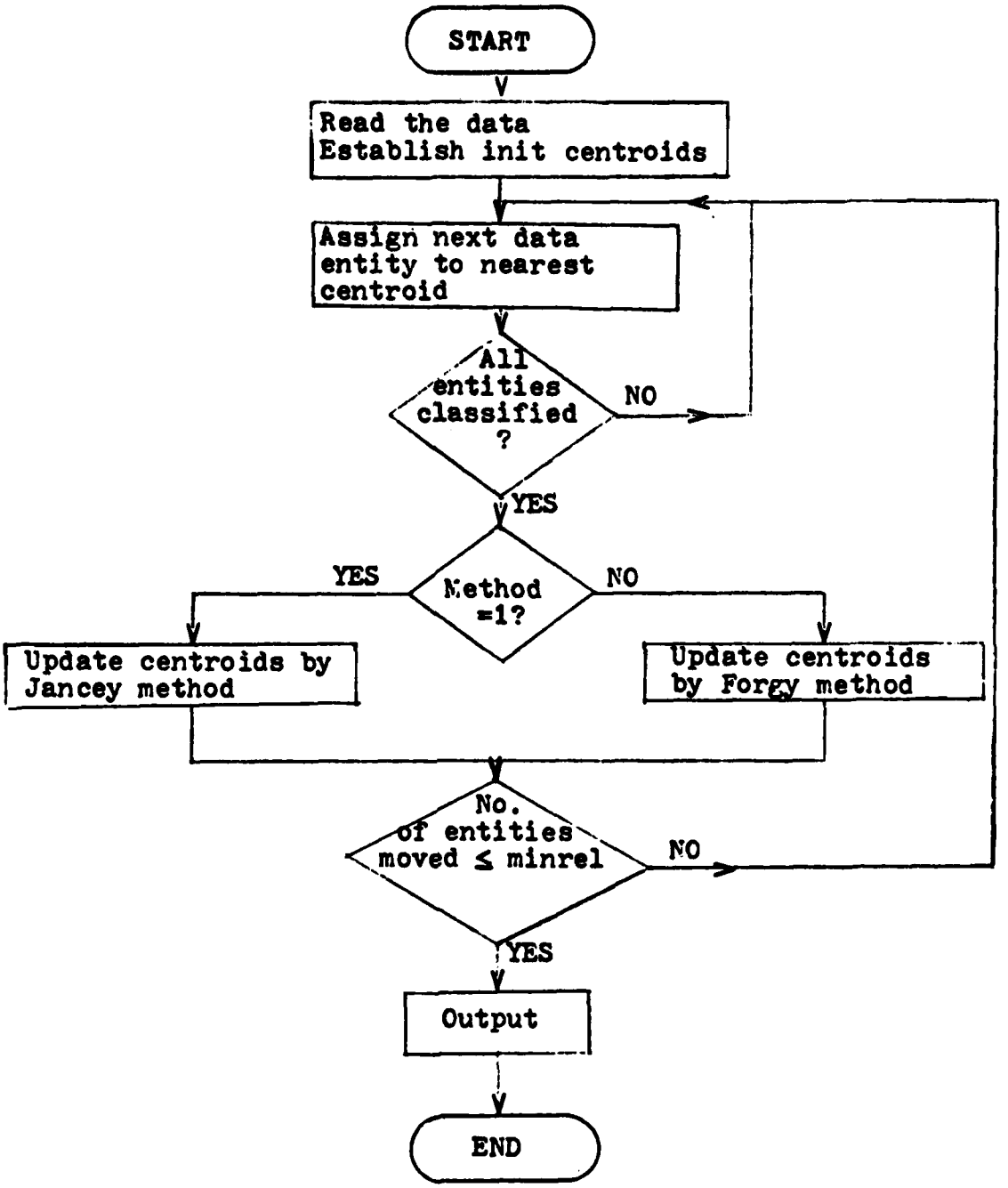


Figure 2.1: The Forgy-Jancey Algorithm

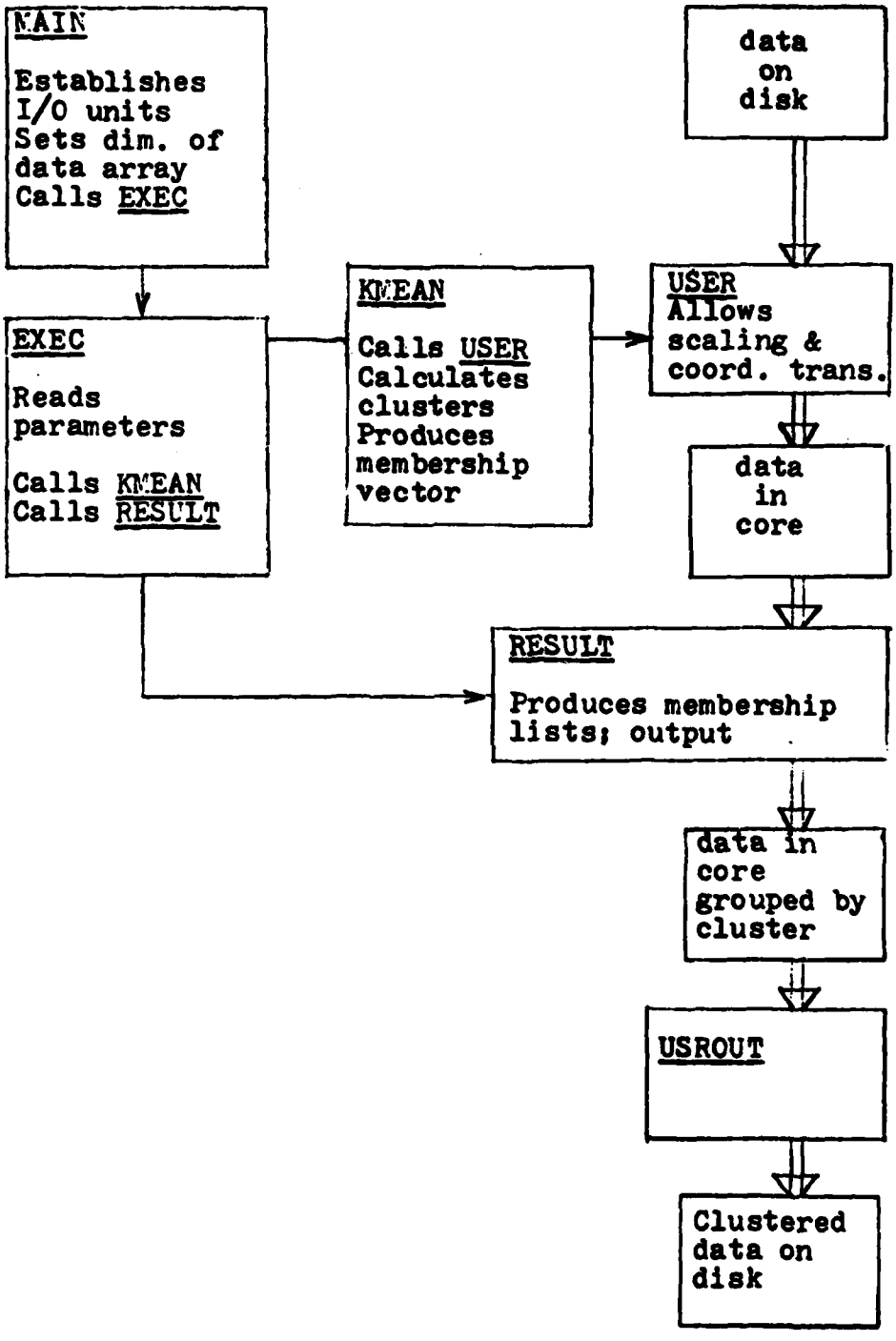


Figure 2.2: Forgy-Jancey Program Path

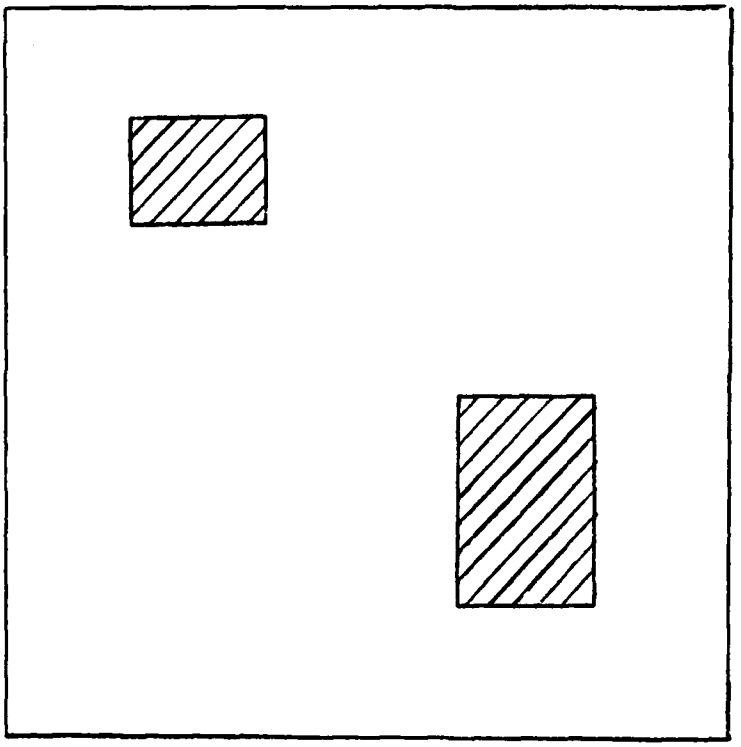


Figure 2.3: Test Scene for Clustering

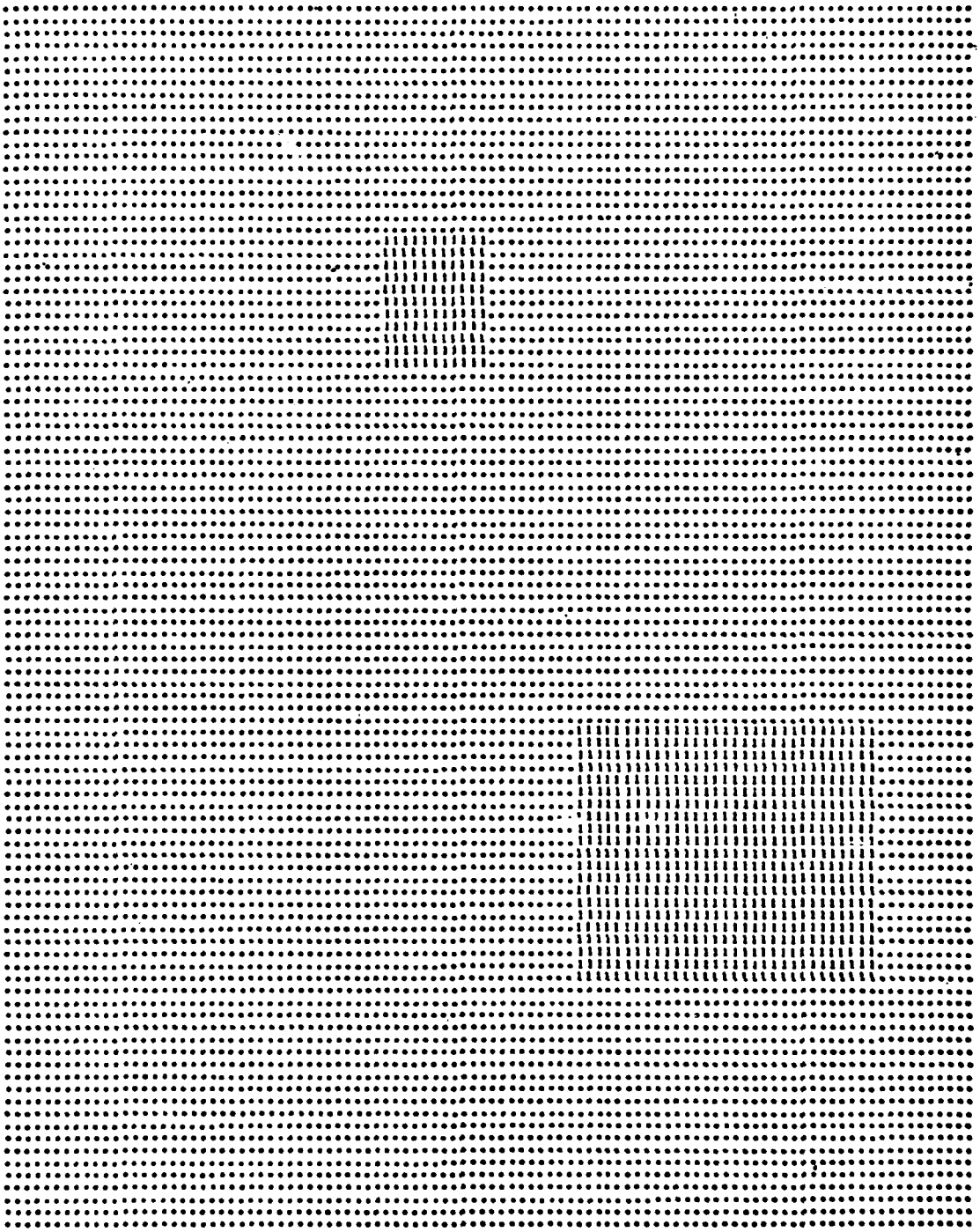


Figure 2.4: Intensity Clustering of Test Scene

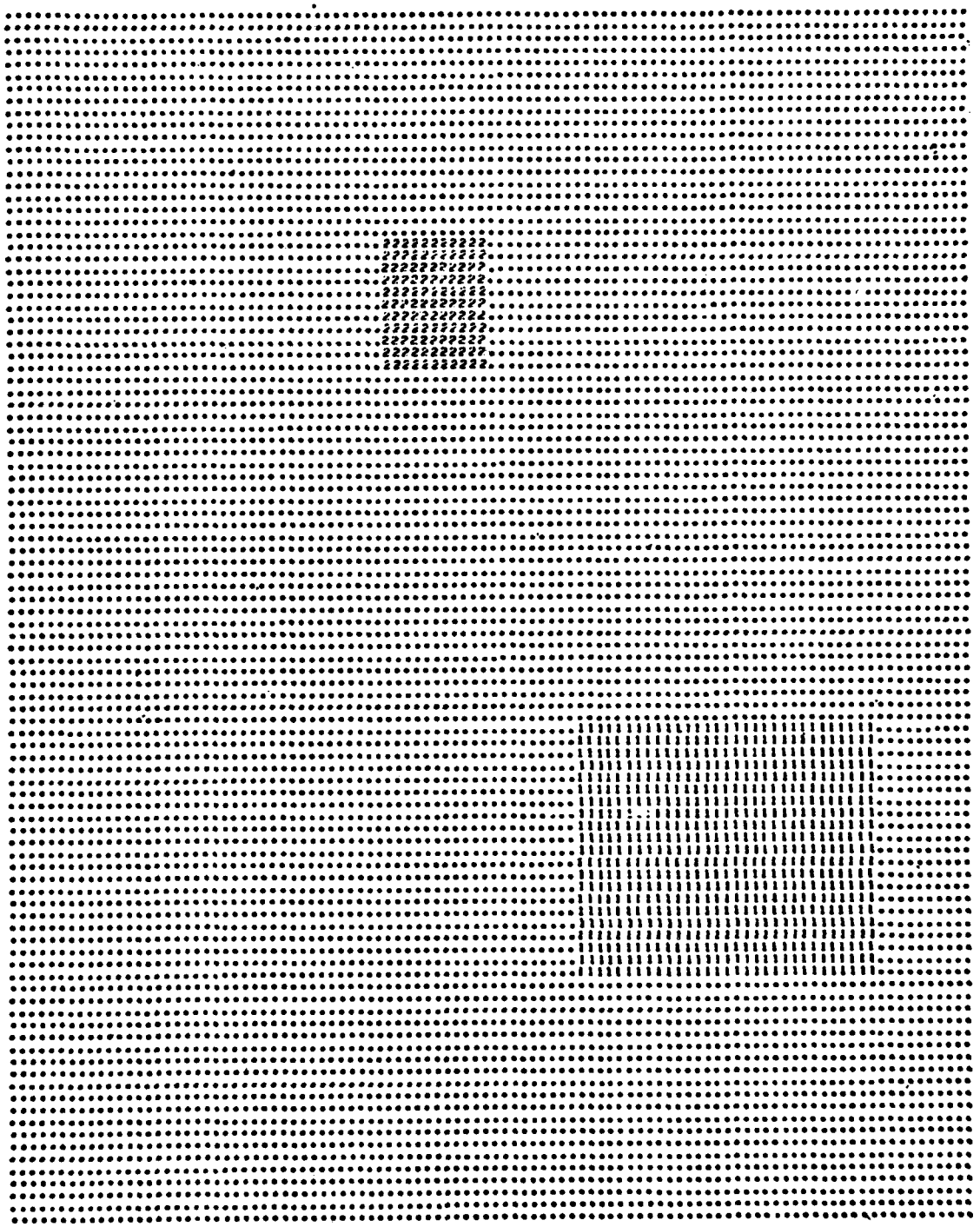


Figure 2.5: x-y Clustering of Test Scene

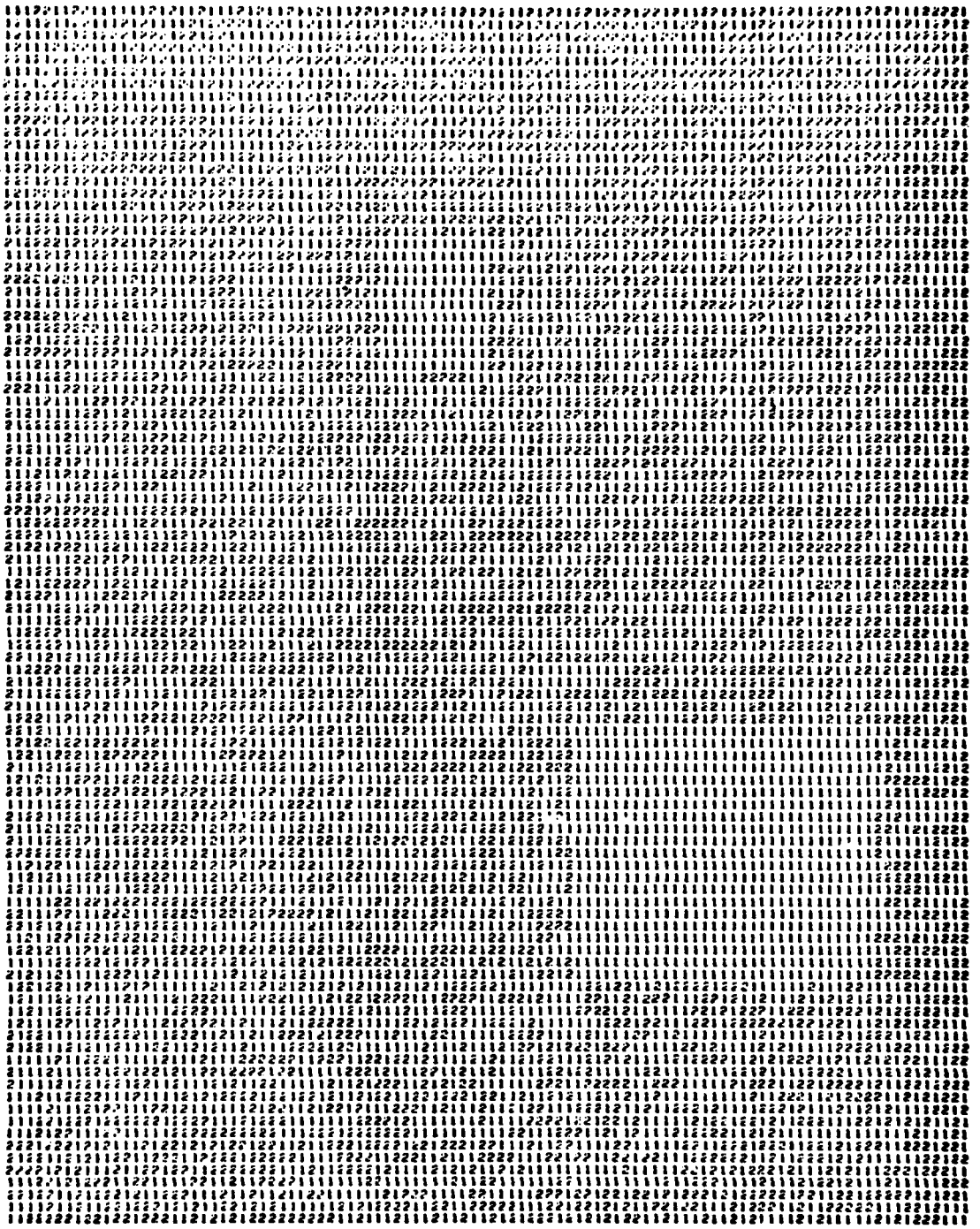


Figure 2.6: Intensity Clustering of Simulated Shadows

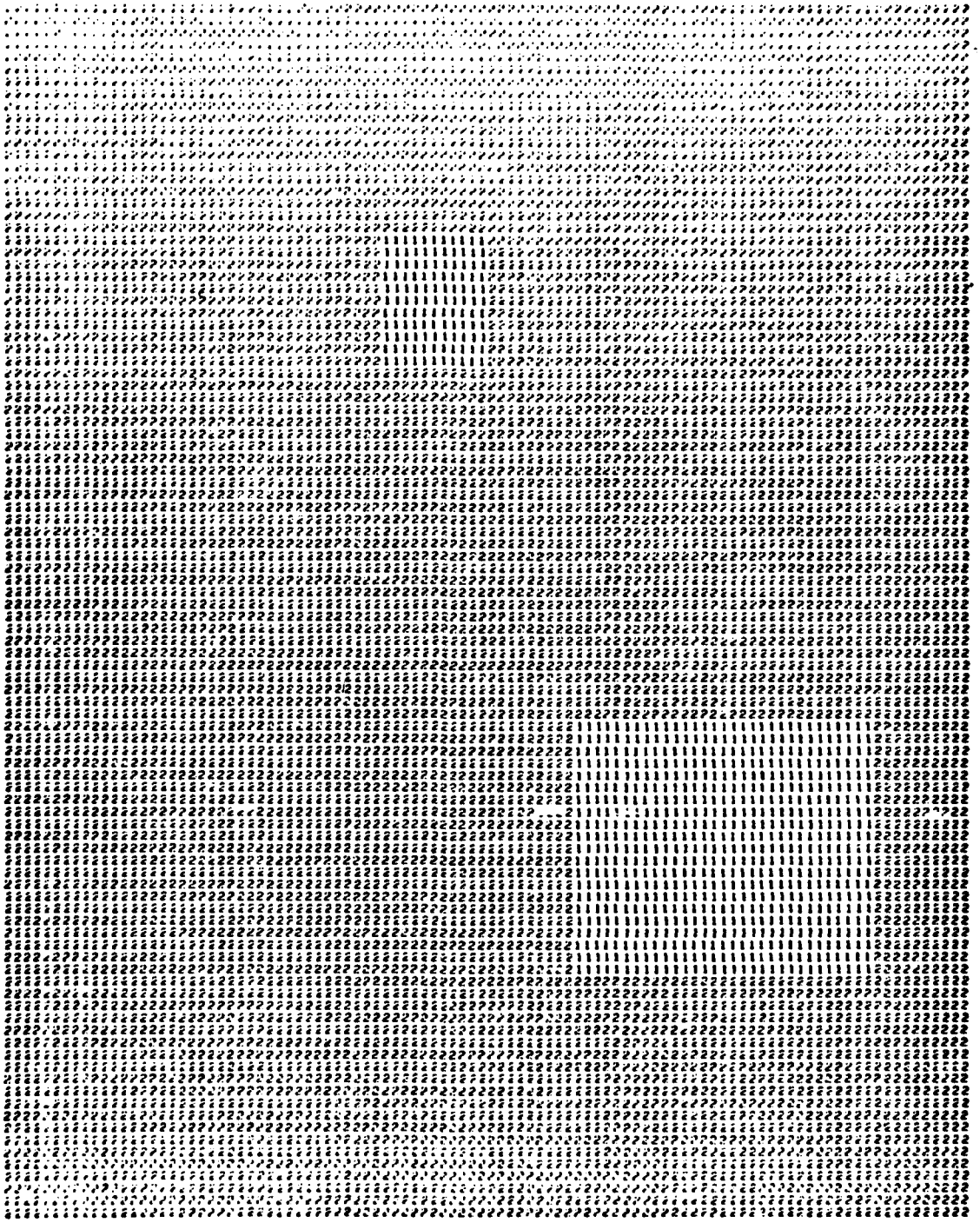


Figure 2.7: Intensity Clustering; Intensity in dB

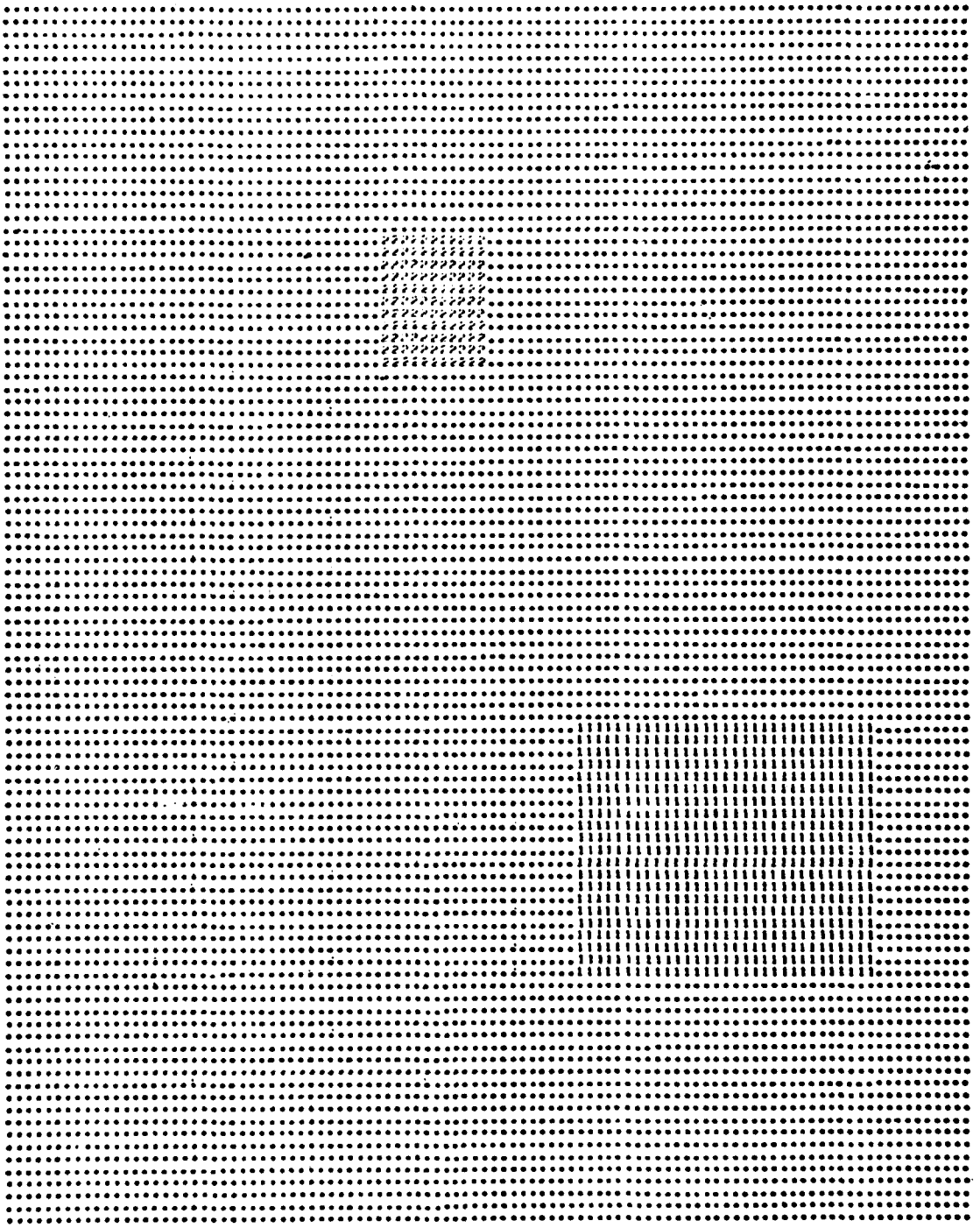


Figure 2.8: x-y Clustering of Simulated Shadows

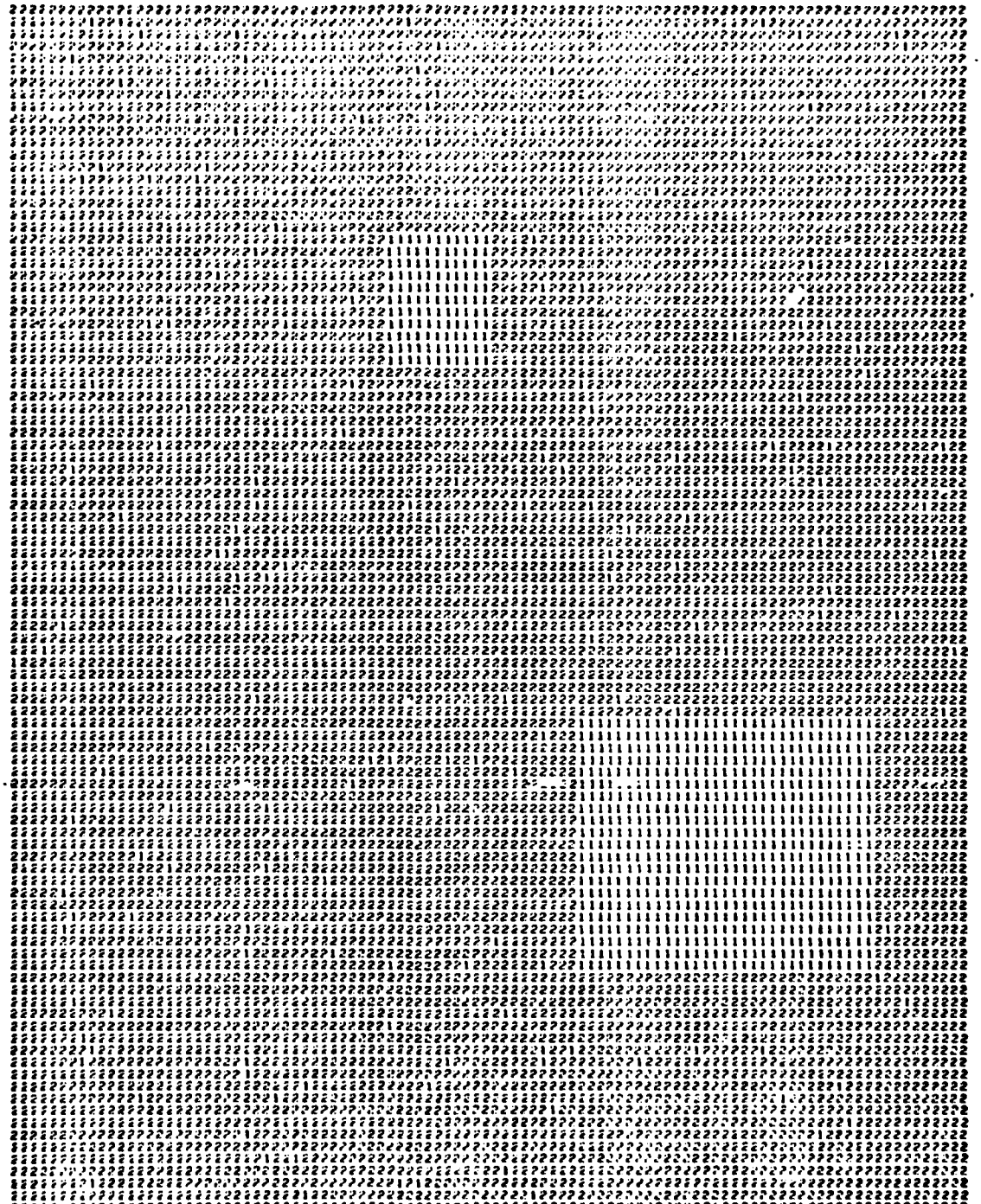


Figure 2.9: Intensity Clustering; Low Background Level

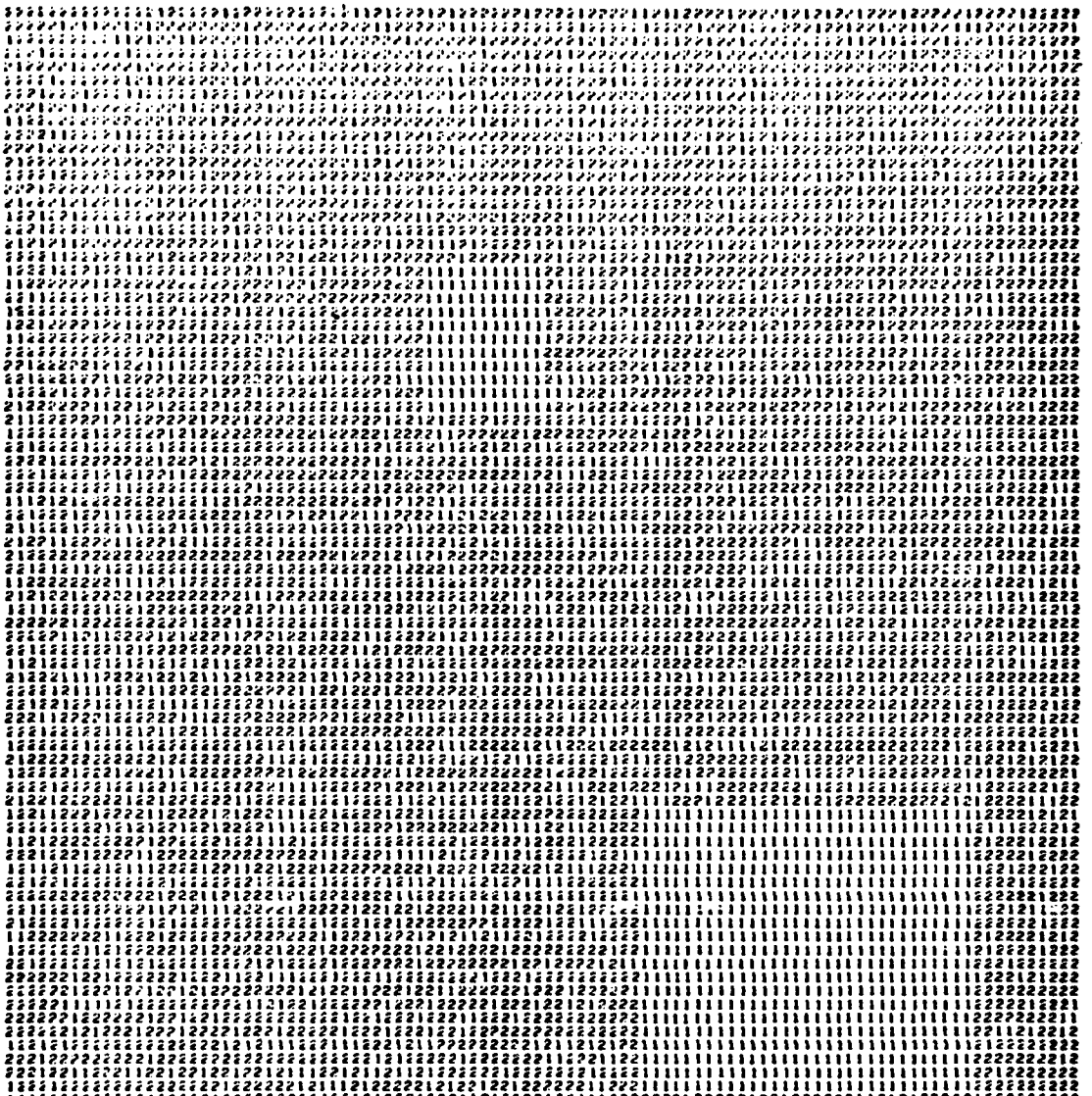


Figure 2.10: Low Background Level Run to Convergence

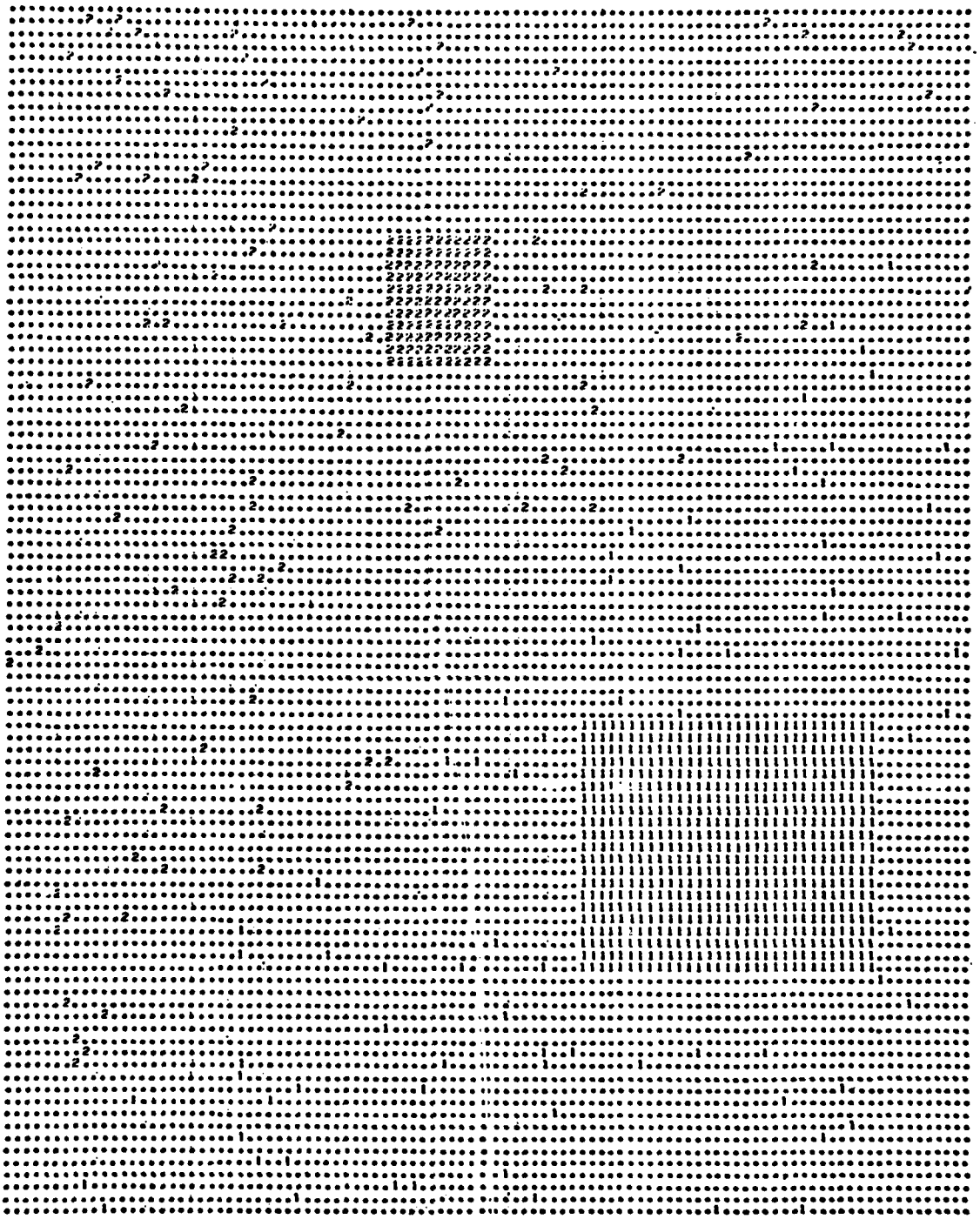


Figure 2.11: x-y Clustering, Low Background Level; NC=2

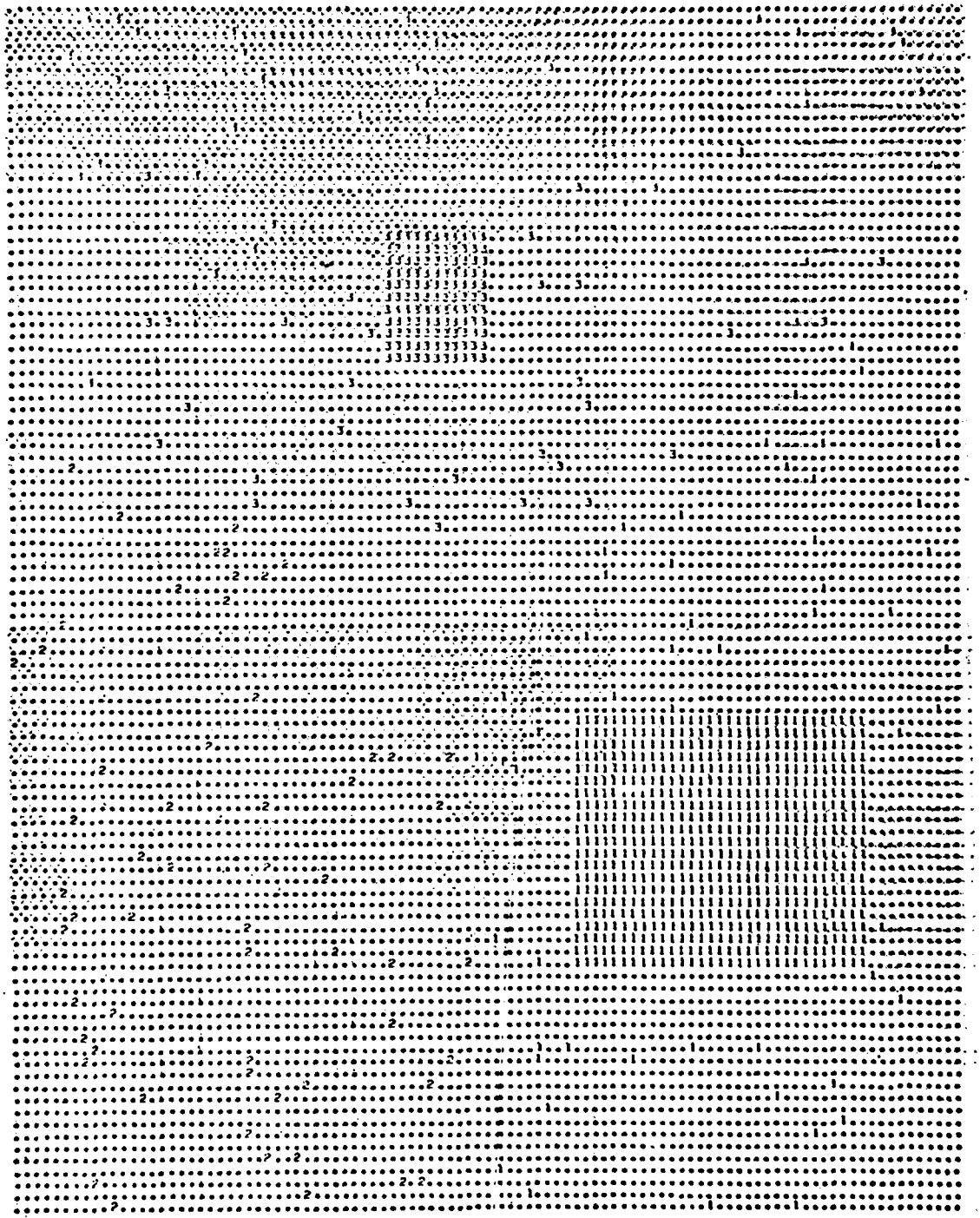


Figure 2.12: x-y Clustering, Low Background Level; NC=3

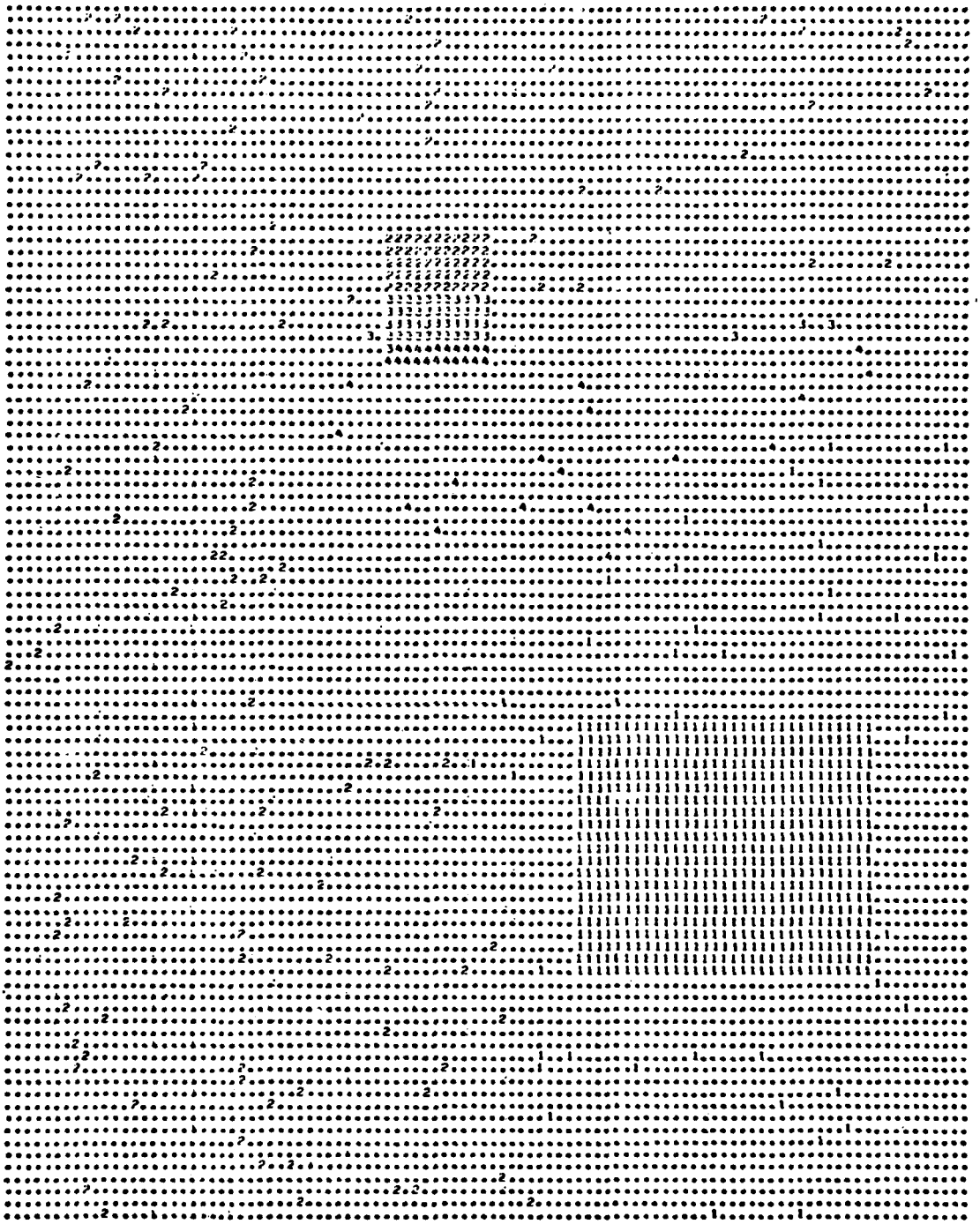


Figure 2.13: x-y Clustering, Low Background Level; NC=4

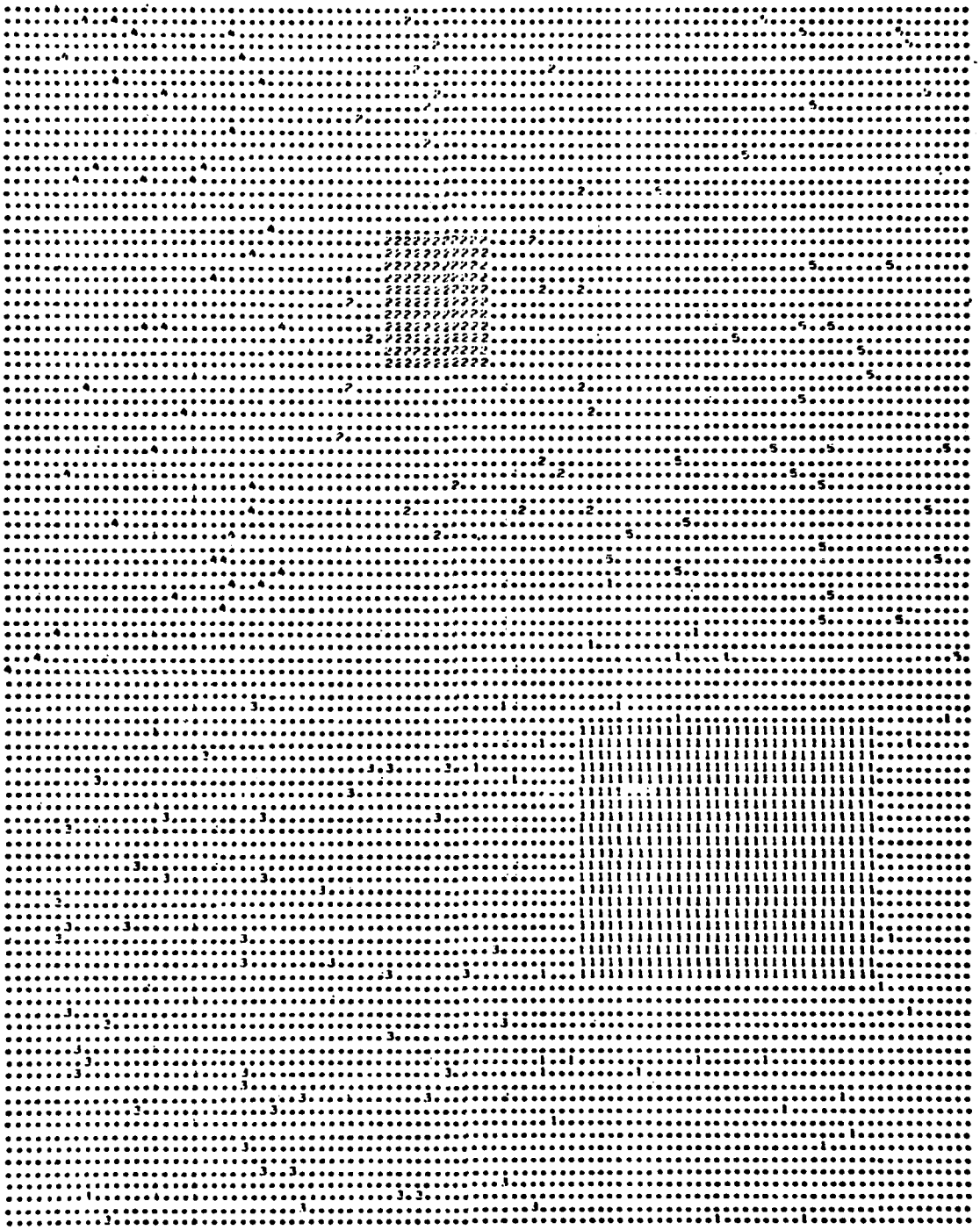


Figure 2.14: x-y Clustering, Low Background Level; NC=5

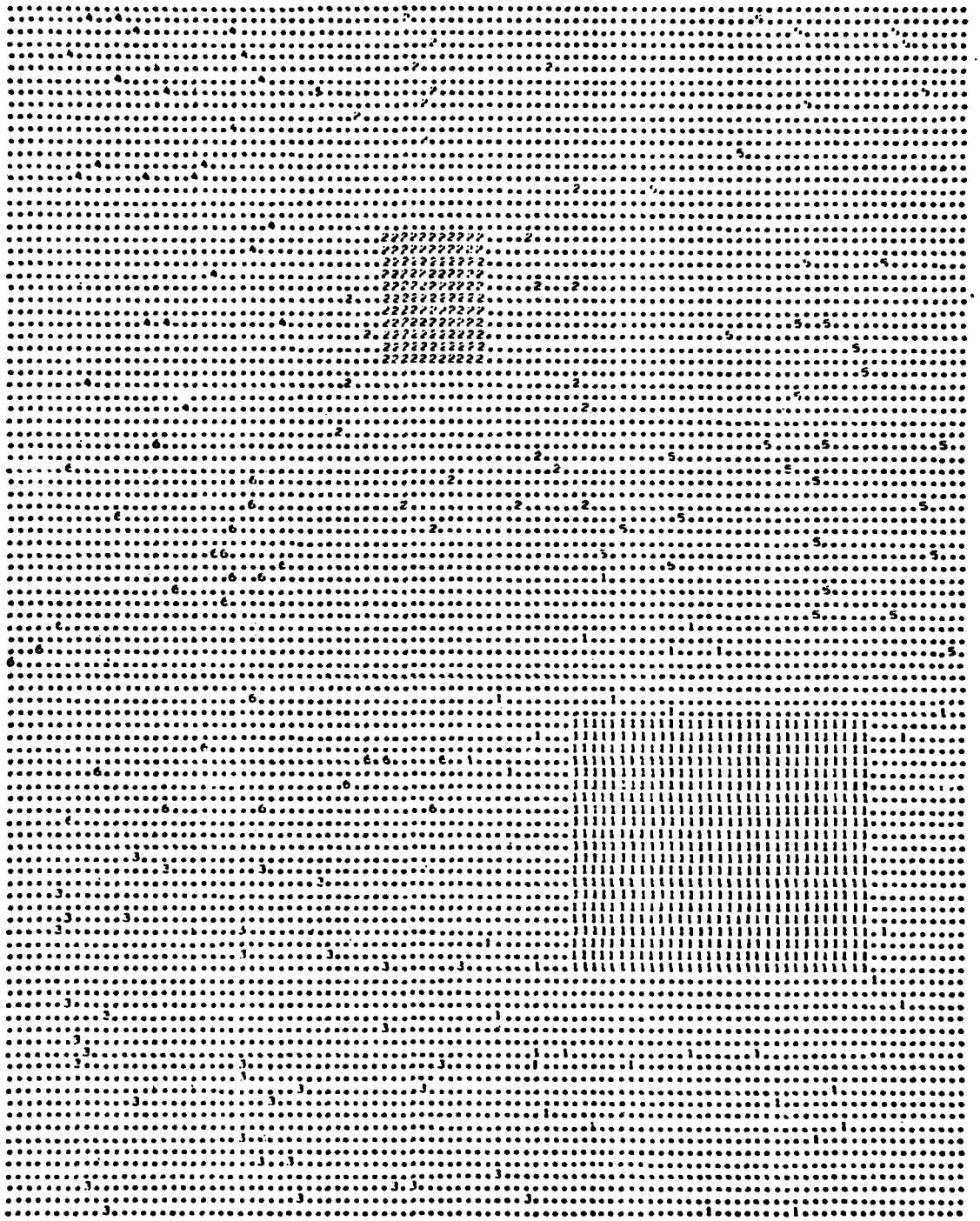


Figure 2.15: x-y Clustering, Low Background Level; NC=6

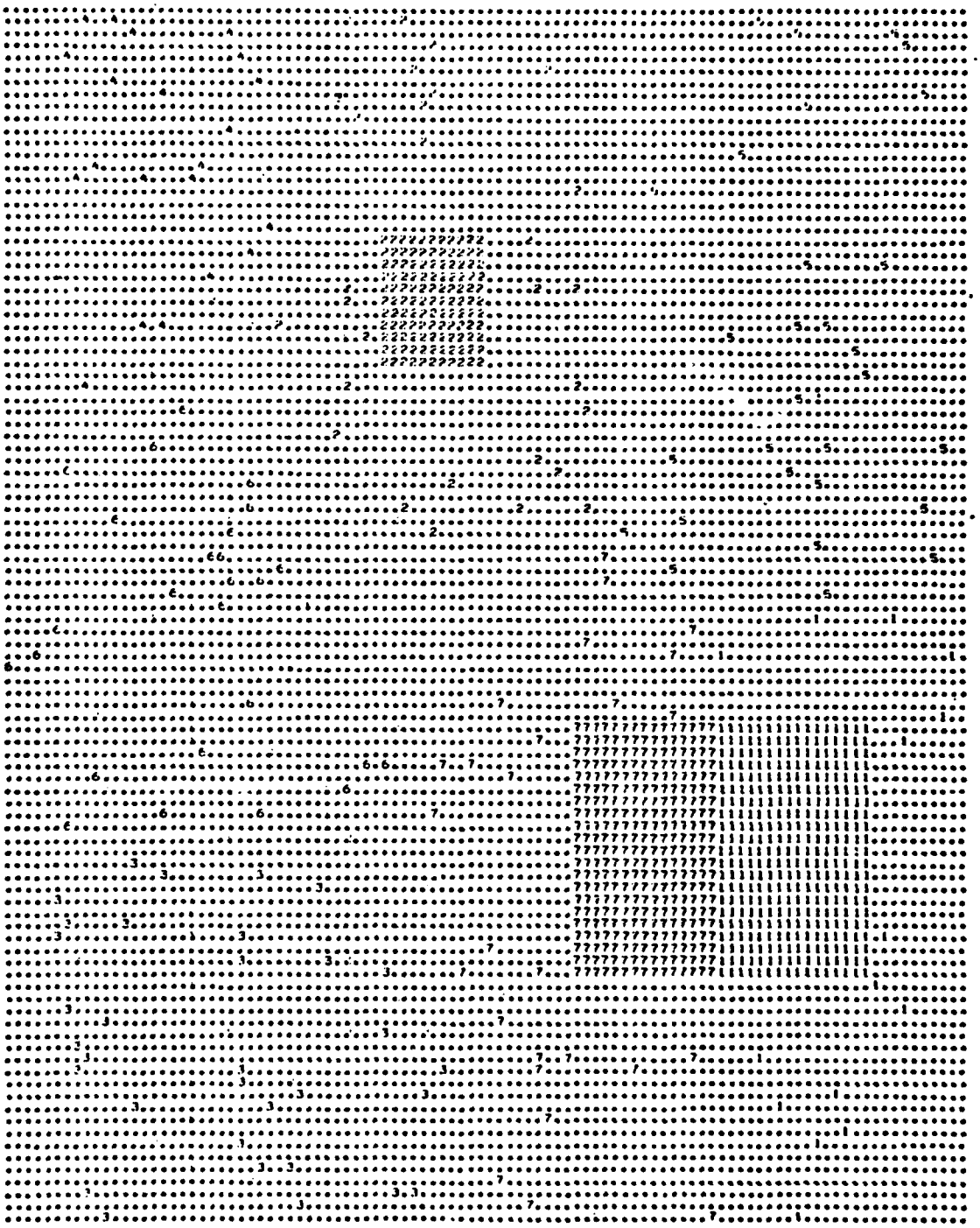


Figure 2.16: x-y Clustering, Low Background Level; NC=7

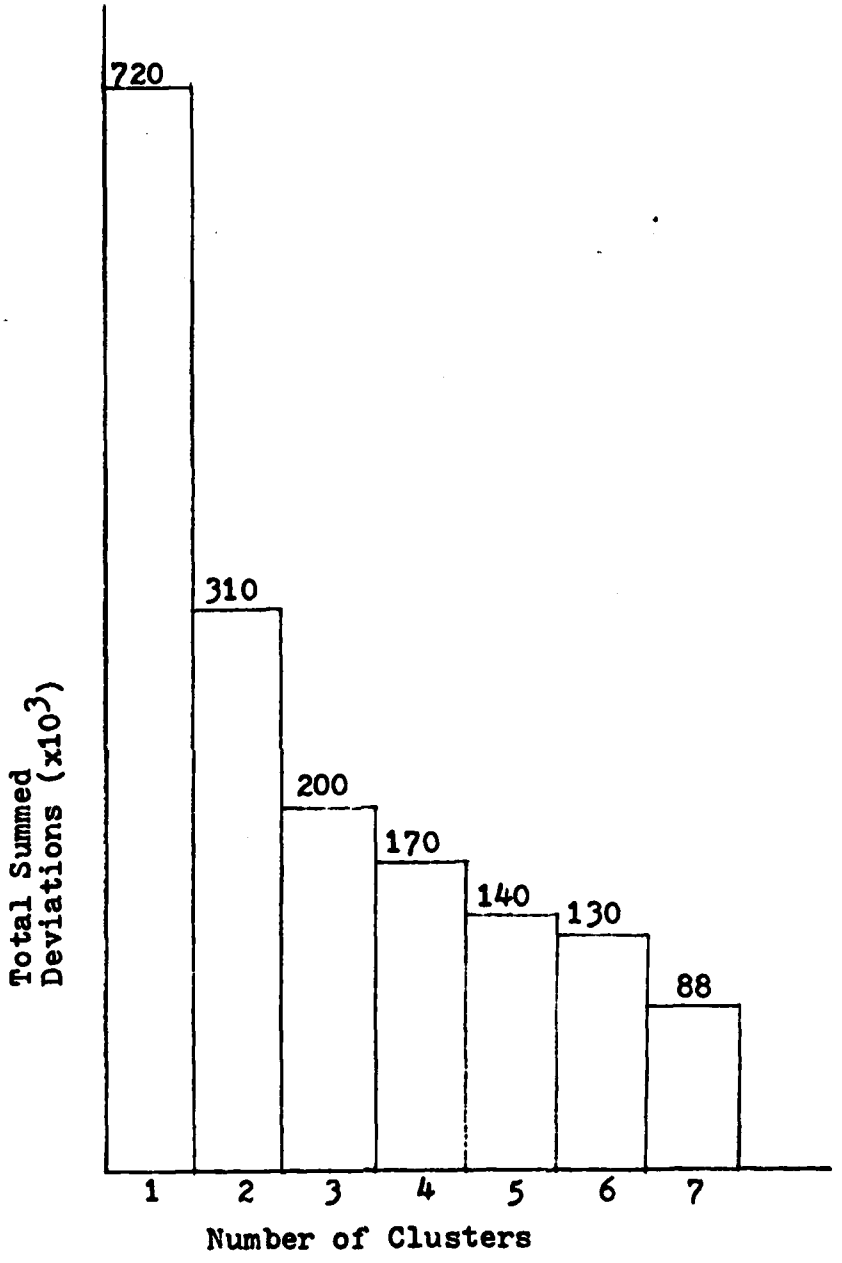


Figure 2.17: Total Summed Deviations

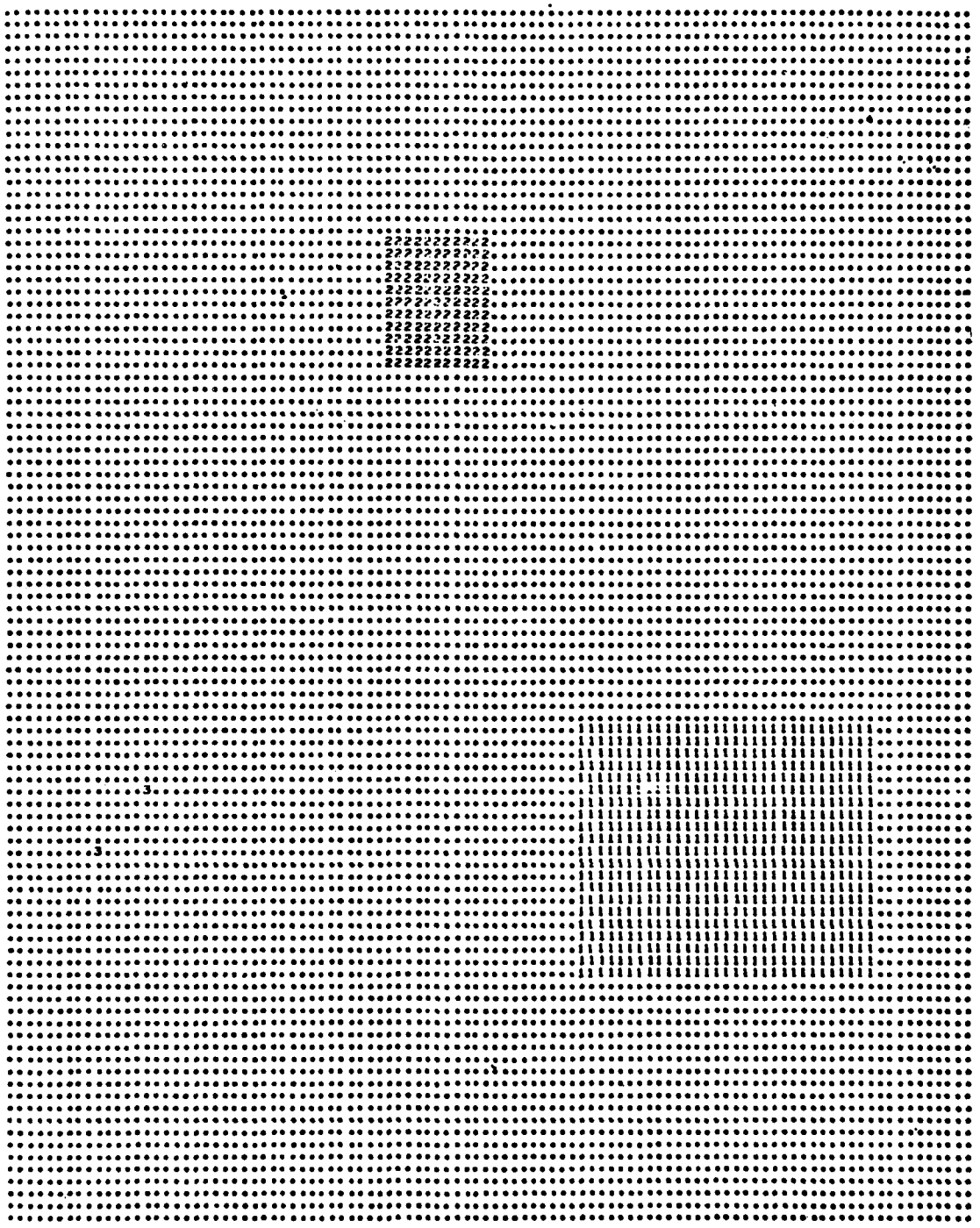


Figure 2.18: Test Scene With Outliers

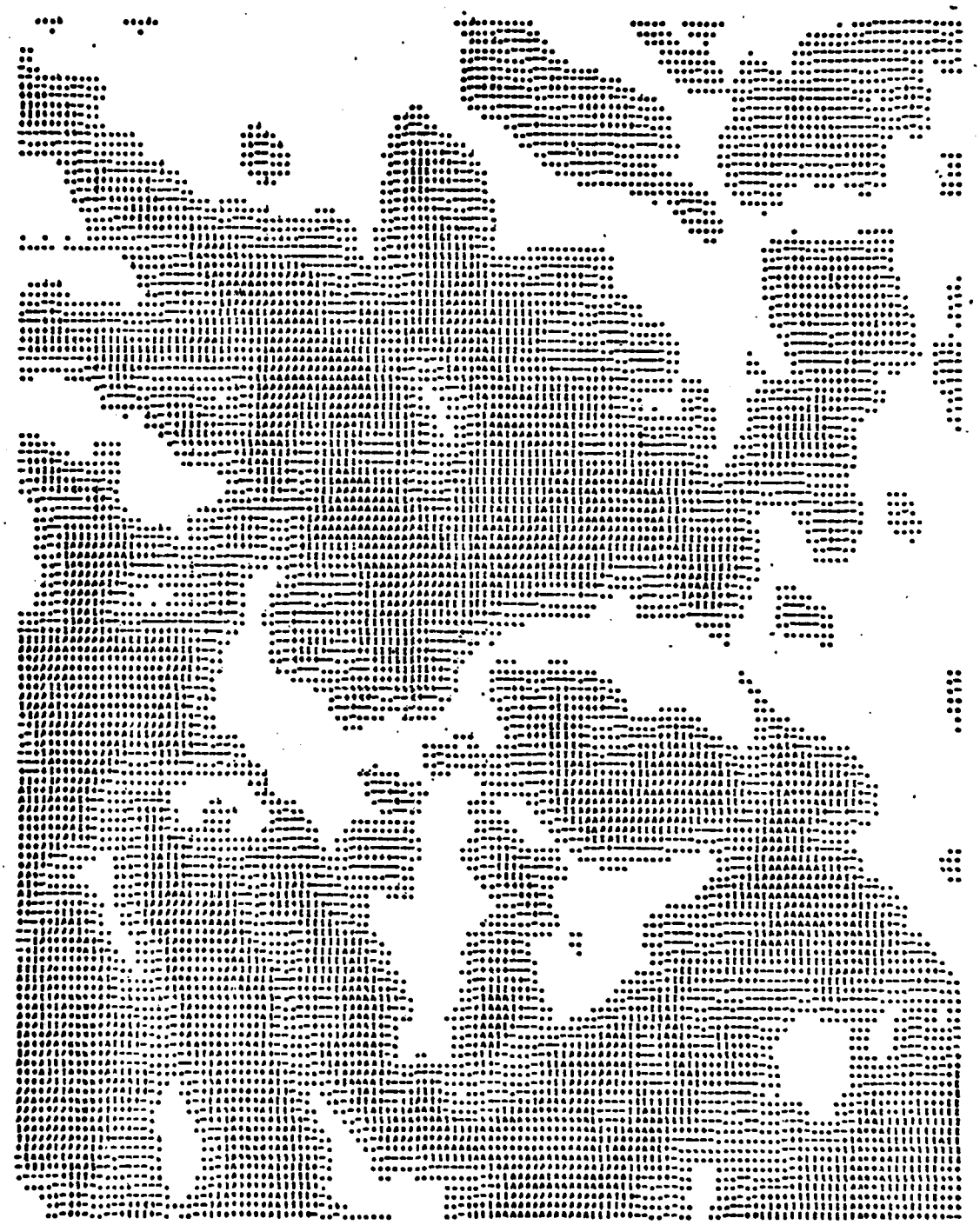


Figure 2.19: Radar Data

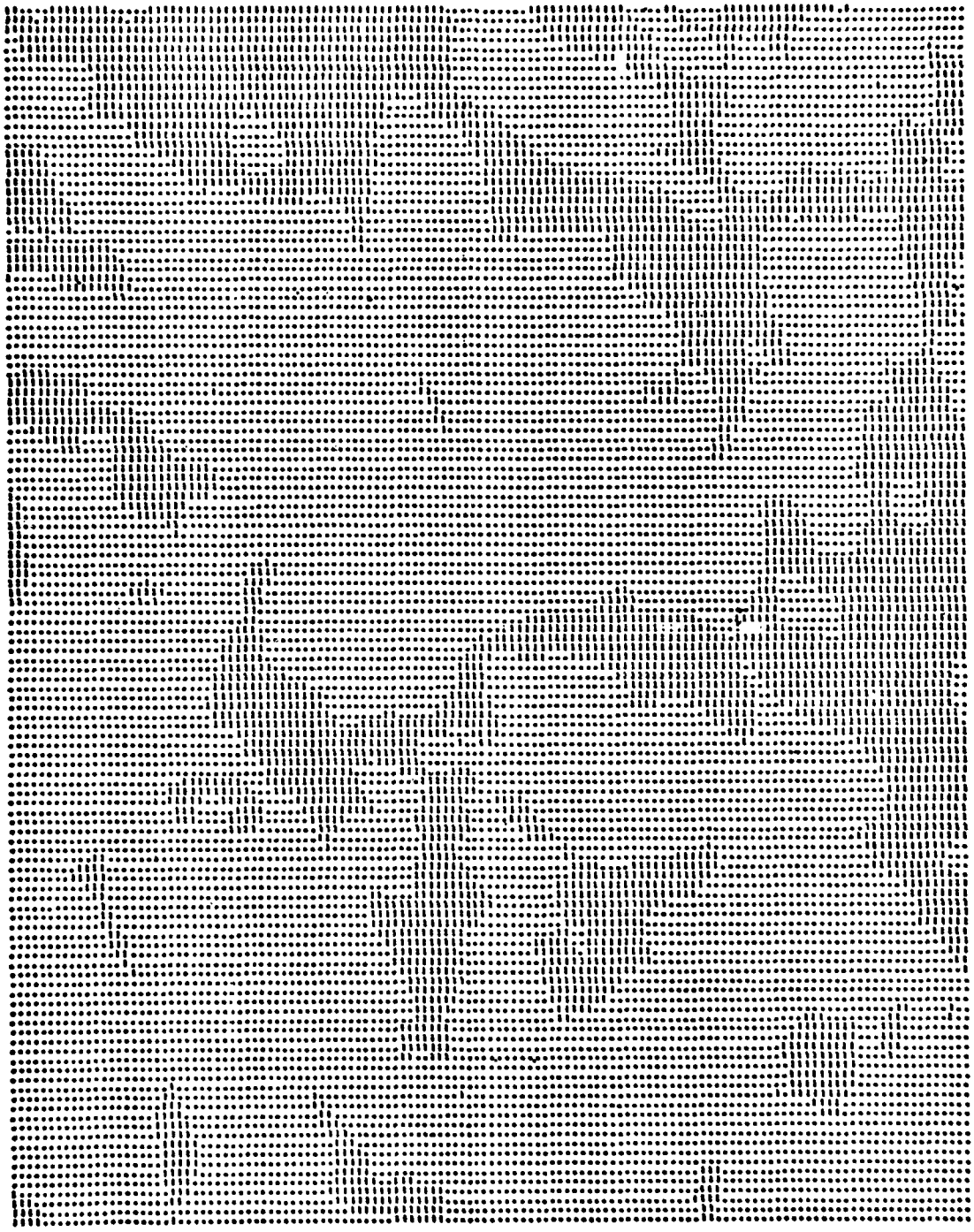


Figure 2.20: Low Intensity Regions

Figure 2.21: x-y Clustering of Low Intensity Regions

161	3	10.	44.	0.0	162	0	37.	4%.	0.0	163	0	20.	44.	0.0	164	0	39
165	3	13.	6%.	0.0	167	0	3%.	3%.	0.0	168	0	12.	9.	0.0	169	3	12
171	3	15.	1%.	0.0	172	0	3%.	6%.	0.0	173	0	36.	3%.	0.0	174	0	37
175	3	18.	70.	0.0	177	0	3%.	7.	0.0	178	0	2%.	58.	0.0	179	0	43
181	0	42.	2%.	0.0	182	0	42.	4%.	0.0	183	0	42.	54.	0.0	184	0	42
186	3	44.	2%.	0.0	187	0	41.	4%.	0.0	188	0	44.	22.	0.0	189	0	46
191	3	48.	2%.	0.0	192	3	41.	2%.	0.0	193	0	49.	18.	0.0	194	0	50
196	0	54.	4.	0.0	197	3	5%.	1.	0.0	198	0	58.	2%.	0.0	199	0	62
201	3	62.	40.	0.0	202	0	6%.	1%.	0.0	203	0	65.	1%.	0.0	204	0	67
206	0	6%.	7.	0.0	207	3	71.	1%.	0.0	208	0	72.	17.	0.0	209	0	72
211	0	7%.	7.	0.0	212	0	7%.	13.	0.0	213	0	77.	6.	0.0	214	0	83
216	0	86.	6.	0.0	217	0	87.	2.	0.0	218	0	88.	8.	0.0			

X-Y RAYLEIGH DATA

I-4 DIST. WHEN I GET THROUGH WITH IT, YOUR DATA IS GONNA BE WAYYYY OUT.

NIN 30 CREATED IN INPU NPAT 218 NTEST 0 NVAR 2 NCAT 1 IFER 0

NOUT 30

NTYPE 0 MAHALONOBIS DISTANCE OF ORDER (2)
(IE. EUCLIDEAN DISTANCE...)

X-Y RAYLEIGH DATA

I-4 TREE. I REALLY PUT YOUR DATA OUT ON A LIMB.

NIN 30 CREATED IN DIST NPAT 218 NTEST 0 NVAR 2 NCAT 1 IFER 0
DAVG 2.72346E 01 DMAX 1.2022E 02

NFNT 1

NIT 1 PRUNING PARAMETERS WILL BE READ FROM CARDS.

NODE				NEIGHBORS				DISTANCE
1	0	2.	9.	0.0	2	0	2. 12.	0.0	1.000E 00
					11	0	5. 7.	0.0	3.606E 00
2	0	2.	12.	0.0	5	0	3. 14.	0.0	2.236E 00
					1	0	2. 9.	0.0	3.000E 00
3	0	2.	42.	0.0	9	0	4. 45.	0.0	3.606E 00
4	0	2.	79.	0.0	7	0	3. 83.	0.0	4.123E 00
5	0	3.	14.	0.0	15	0	7. 12.	0.0	4.472E 00
					2	0	2. 12.	0.0	2.236E 00
6	0	3.	24.	0.0	12	0	5. 25.	0.0	2.236E 00
7	0	3.	83.	0.0	21	0	9. 84.	0.0	6.083E 00
					4	0	2. 79.	0.0	4.123E 00
					8	0	3. 93.	0.0	1.000E 01
8	0	3.	93.	0.0	7	0	3. 83.	0.0	1.000E 01
					10	0	4. 94.	0.0	1.414E 00

Figure 3.1: TREE Example Run

THE DATA HAS BEEN CLUSTERED SEARCHING FOR A CLUSTER OF 2
INCONSISTENT LINKS HAVE BEEN DETERMINED BY A FACTOR OF 2.00000E 00
AND A CUTOFF OF 0.0 STANDARD DEVIATIONS.

5 CLUSTERS DETECTED

CLUSTER	# CONSISTS OF	THE FOLLOWING	PDF PATTERNS.
1	2	2. 9.	2.0
2	0	8. 13.	0.0
11	0	5. 7.	0.0
5	0	3. 14.	0.0
15	0	7. 12.	0.0
17	0	4. 17.	0.0
27	0	14. 21.	0.0
30	0	15. 23.	0.0
23	0	11. 24.	0.0
70	0	23. 22.	0.0
39	0	15. 15.	0.0
16	0	7. 27.	0.0
119	0	27. 17.	0.0
90	0	21. 20.	0.0
26	0	14. 10.	0.0
12	0	5. 25.	0.0
171	0	34. 15.	0.0
114	0	27. 15.	0.0
33	0	10. 28.	0.0
120	0	27. 29.	0.0
28	0	15. 8.	0.0
6	0	3. 24.	0.0
174	0	37. 16.	0.0
95	0	25. 36.	0.0
185	0	43. 12.	0.0
133	0	24. 38.	0.0
163	0	49. 18.	0.0
177	0	39. 7.	0.0
134	0	28. 40.	0.0
169	0	32. 29.	0.0
148	0	46. 22.	0.0
168	0	32. 9.	0.0
146	0	25. 40.	0.0
135	0	28. 41.	0.0
131	0	27. 40.	0.0
173	0	36. 35.	0.0
109	0	46. 23.	0.0
157	0	30. 40.	0.0
147	0	29. 41.	0.0
122	0	27. 41.	0.0
107	0	26. 43.	0.0
191	0	48. 23.	0.0
186	0	44. 24.	0.0
158	0	39. 41.	0.0
108	0	26. 41.	0.0
98	0	25. 40.	0.0
194	0	50. 23.	0.0
192	0	48. 27.	0.0
181	0	42. 28.	0.0
159	0	30. 42.	0.0
97	0	25. 41.	0.0
82	0	21. 40.	0.0
188	0	38. 26.	0.0
190	0	47. 29.	0.0
179	0	40. 26.	0.0

Figure 3.2: Default Clustering

```
THE DATA WAS PERTURBED BY ADDING TO A RANDOM
INCREASE IN EACH DATA VALUE DETERMINED BY A FACTOR OF 1.00000.00
AND A SPREAD OF  1.00000.00 STANDARD DEVIATIONS.

   32 CLUSTERS DETECTED

CLUSTER  1 CONSISTS OF THE FOLLOWING     5 PATTERNS.
    1       0    2.    9.     0.0
    2       0    3.   12.     0.0
    5       0    3.   14.     0.0
   15       0    7.   12.     0.0
   17       0    8.   17.     0.0

CLUSTER  2 CONSISTS OF THE FOLLOWING     4 PATTERNS.
   27       0   14.   21.     0.0
   30       0   15.   20.     0.0
   23       0   11.   24.     0.0
   29       0   15.   15.     0.0

CLUSTER  3 CONSISTS OF THE FOLLOWING     2 PATTERNS.
   26       0   14.   13.     0.0
   28       0   15.    8.     0.0

CLUSTER  4 CONSISTS OF THE FOLLOWING     3 PATTERNS.
   10       0    7.   27.     0.0
   12       0    5.   25.     0.0
    6       0    3.   24.     0.0

CLUSTER  5 CONSISTS OF THE FOLLOWING     5 PATTERNS.
   70       0   23.   22.     0.0
  119       0   27.   17.     0.0
   46       0   21.   26.     0.0
  118       0   27.   15.     0.0
   33       0   19.   26.     0.0

CLUSTER  6 CONSISTS OF THE FOLLOWING     1 PATTERNS.
  120       0   27.   29.     0.0

CLUSTER  7 CONSISTS OF THE FOLLOWING   132 PATTERNS.
   95       0   25.   36.     0.0
  133       0   28.   38.     0.0
  134       0   28.   40.     0.0
  146       0   29.   40.     0.0
  135       0   28.   41.     0.0
  121       0   27.   40.     0.0
  157       0   30.   40.     0.0
  147       0   29.   41.     0.0
  122       0   27.   41.     0.0
  107       0   26.   40.     0.0
  158       0   30.   41.     0.0
  109       0   26.   41.     0.0
   96       0   25.   42.     0.0
  159       0   30.   42.     0.0
   97       0   25.   41.     0.0
   82       0   24.   40.     0.0
  160       0   33.   43.     0.0
  148       0   29.   42.     0.0
   83       0   24.   41.     0.0
   71       0   23.   40.     0.0
  161       0   30.   44.     0.0
  149       0   29.   43.     0.0
  136       0   28.   42.     0.0
```

Figure 3.3: Better Clustering

```
THE TREE HAS BEEN CLUSTERED SEARCHING TO A DEPTH OF  1
          INCONSISTENT EDGES HAVE BEEN DETERMINED BY A FACTOR OF 1.0000E 00
          AND A SPREAD OF  5.0000E-01 STANDARD DEVIATIONS.

   17 CLUSTERS DETECTED

CLUSTER  1 CONSISTS OF THE FOLLOWING     3 PATTERNS.
     1       0    2.    9.      0.0
     2       0    2.   12.      0.0
     5       0    3.   14.      0.0

CLUSTER  2 CONSISTS OF THE FOLLOWING     2 PATTERNS.
    15       0    7.   12.      0.0
    17       0    8.   17.      0.0

CLUSTER  3 CONSISTS OF THE FOLLOWING     4 PATTERNS.
    27       0   14.   21.      0.0
    33       0   15.   20.      0.0
    23       0   11.   24.      0.0
    29       0   15.   15.      0.0

CLUSTER  4 CONSISTS OF THE FOLLOWING     2 PATTERNS.
    26       0   14.   13.      0.0
    28       0   15.    8.      0.0

CLUSTER  5 CONSISTS OF THE FOLLOWING     3 PATTERNS.
    10       0    7.   27.      0.0
    12       0    5.   25.      0.0
     8       0    3.   24.      0.0

CLUSTER  6 CONSISTS OF THE FOLLOWING     5 PATTERNS.
    70       0   23.   22.      0.0
   119       0   27.   17.      0.0
    46       0   21.   26.      0.0
   118       0   27.   15.      0.0
    22       0   19.   28.      0.0

CLUSTER  7 CONSISTS OF THE FOLLOWING     1 PATTERNS.
   121       0   27.   29.      0.0

CLUSTER  8 CONSISTS OF THE FOLLOWING   121 PATTERNS.
    95       0   25.   36.      0.0
   113       0   28.   39.      0.0
   134       0   28.   40.      0.0
   136       0   34.   40.      0.0
   125       0   28.   41.      0.0
   121       0   27.   40.      0.0
   137       0   33.   40.      0.0
   127       0   29.   41.      0.0
   111       0   27.   41.      0.0
   107       0   26.   40.      0.0
   128       0   33.   41.      0.0
   108       0   26.   41.      0.0
    66       0   28.   40.      0.0
   129       0   33.   42.      0.0
    97       0   25.   41.      0.0
    85       0   24.   40.      0.0
   103       0   30.   43.      0.0
   131       0   32.   42.      0.0
    91       0   24.   41.      0.0
    71       0   21.   39.      0.0
   141       0   30.   44.      0.0
```

Figure 3.4: Best Clustering Found

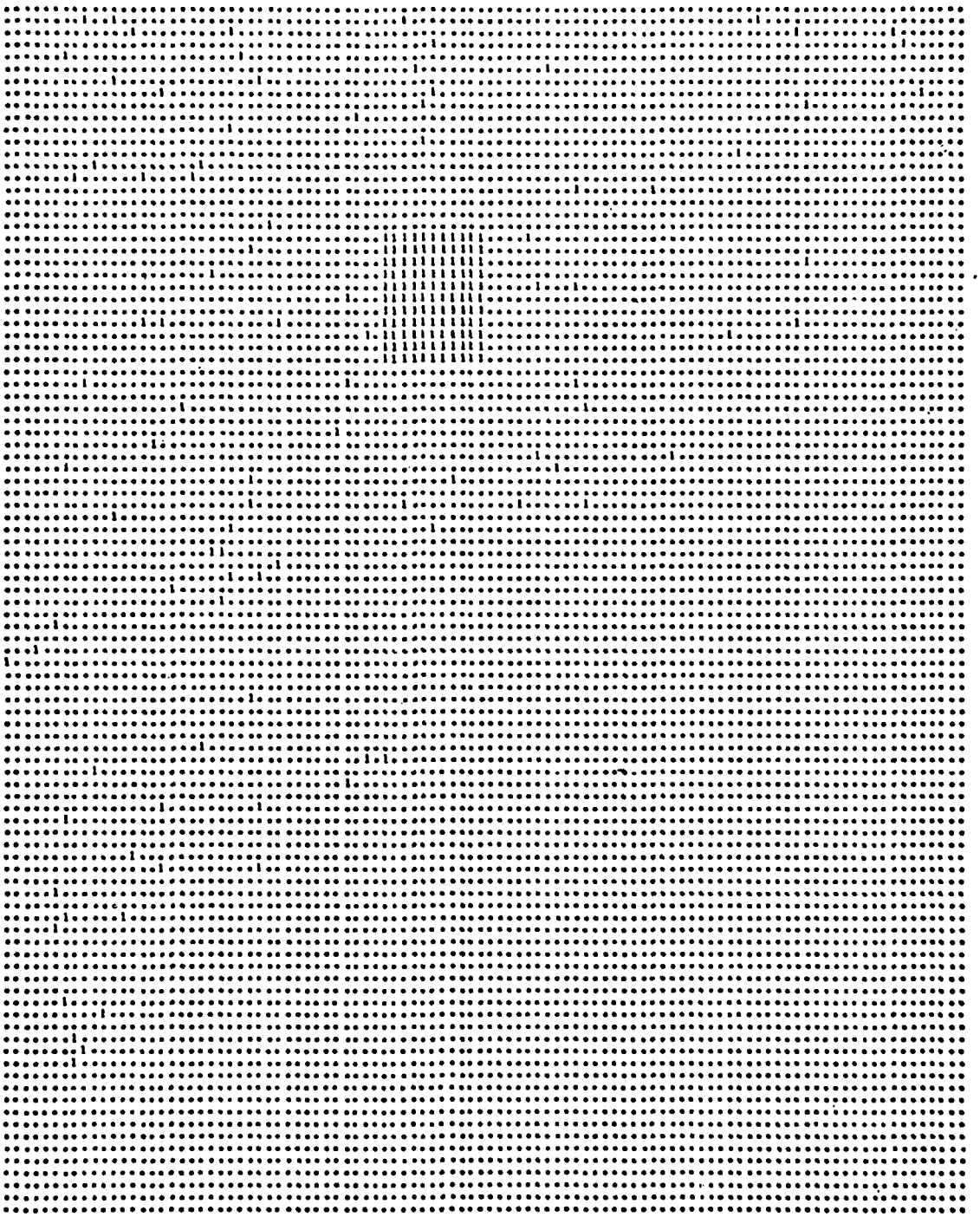


Figure 3.5: Data Set for Tree Example

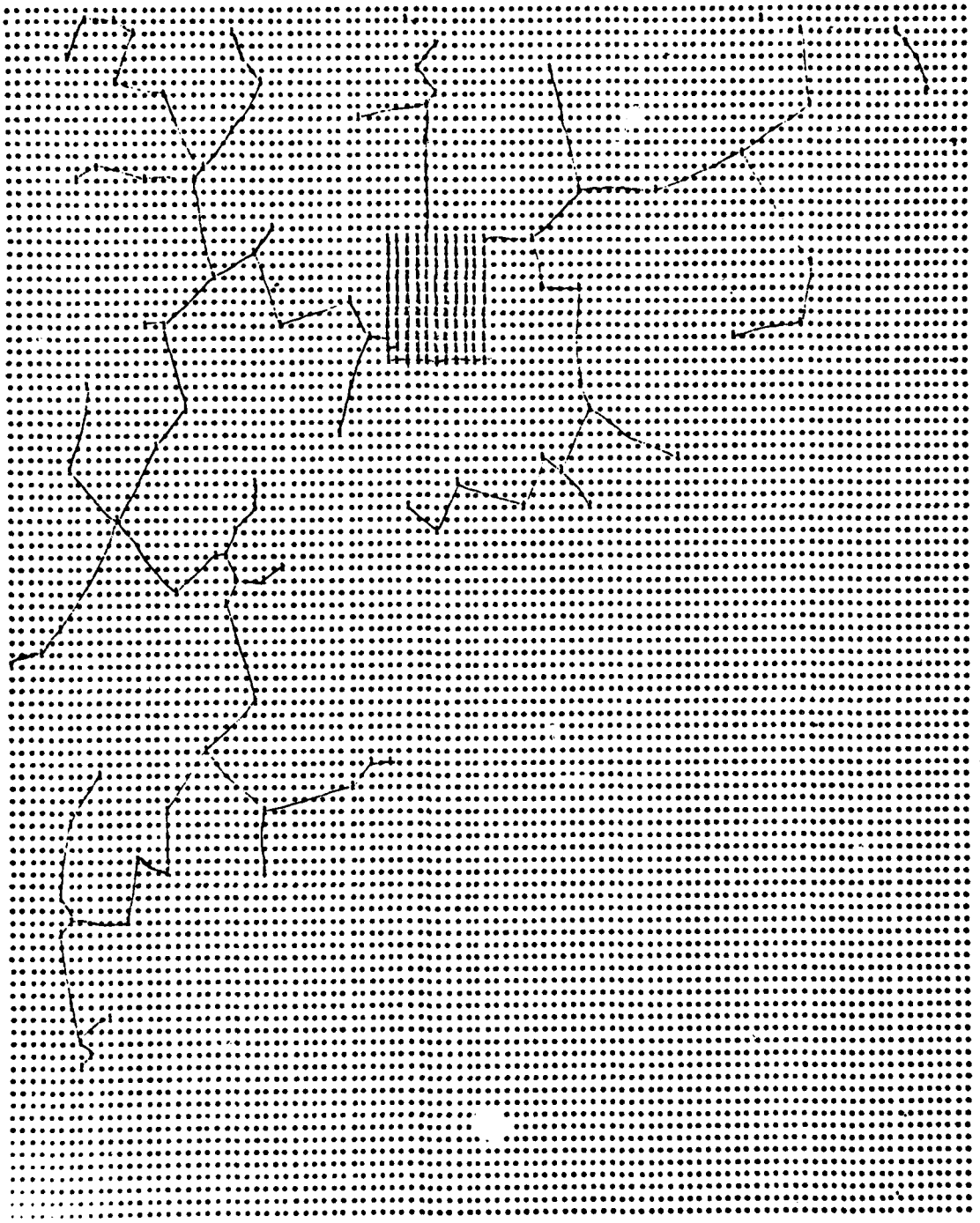


Figure 3.6: Minimal Spanning Tree

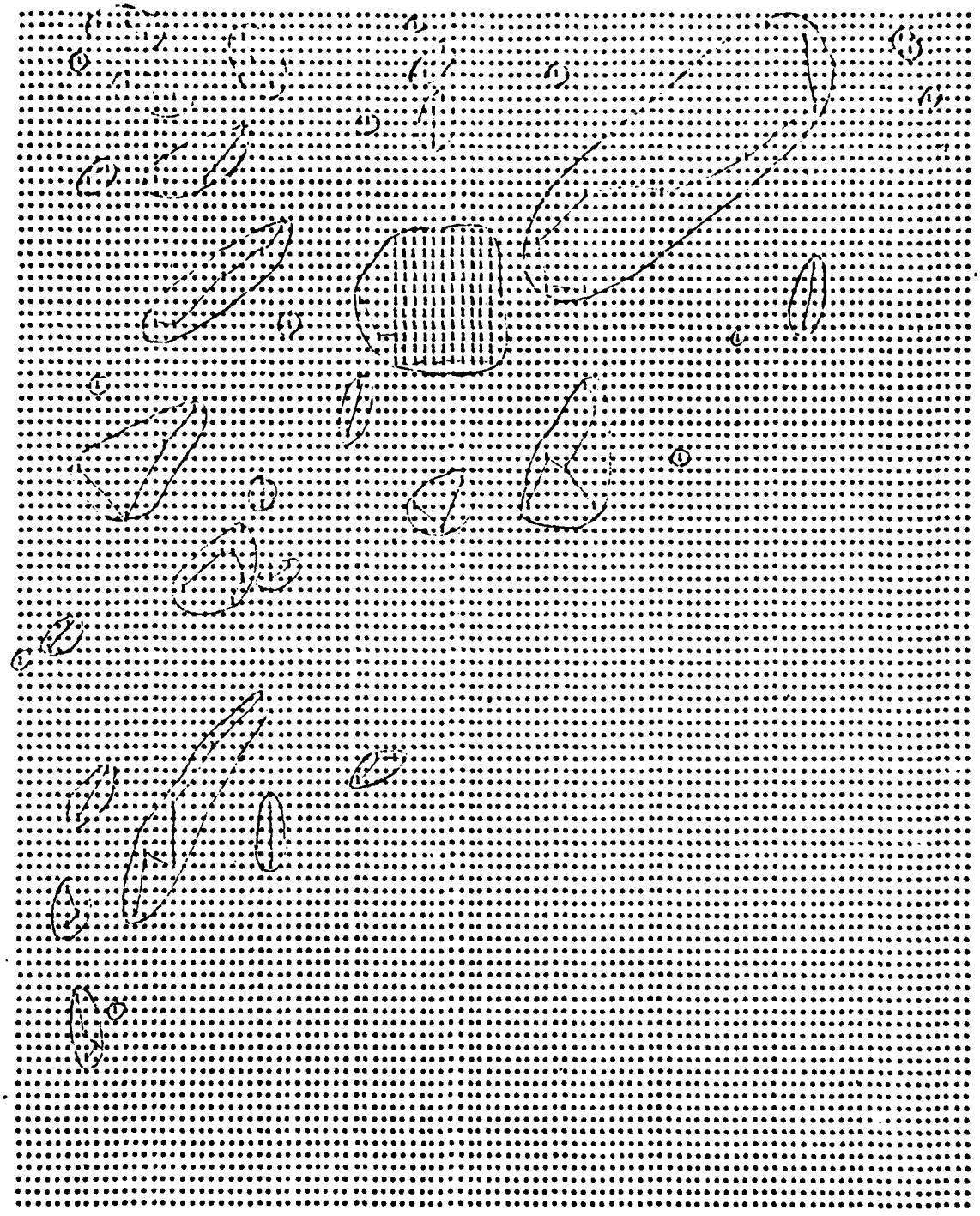


Figure 3.7: Best Clustering Found

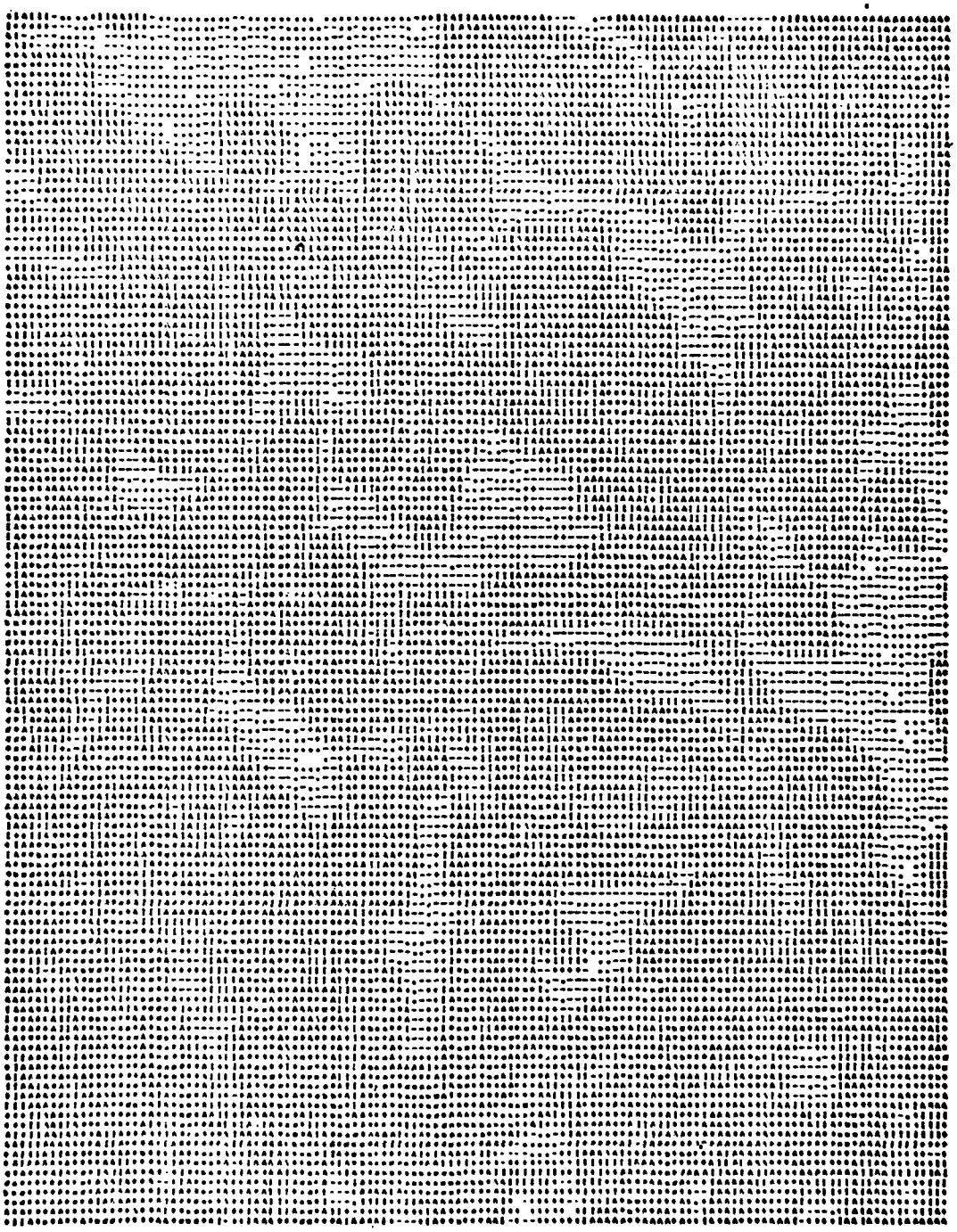


Figure 4.1: Point-to-Point Local Variance

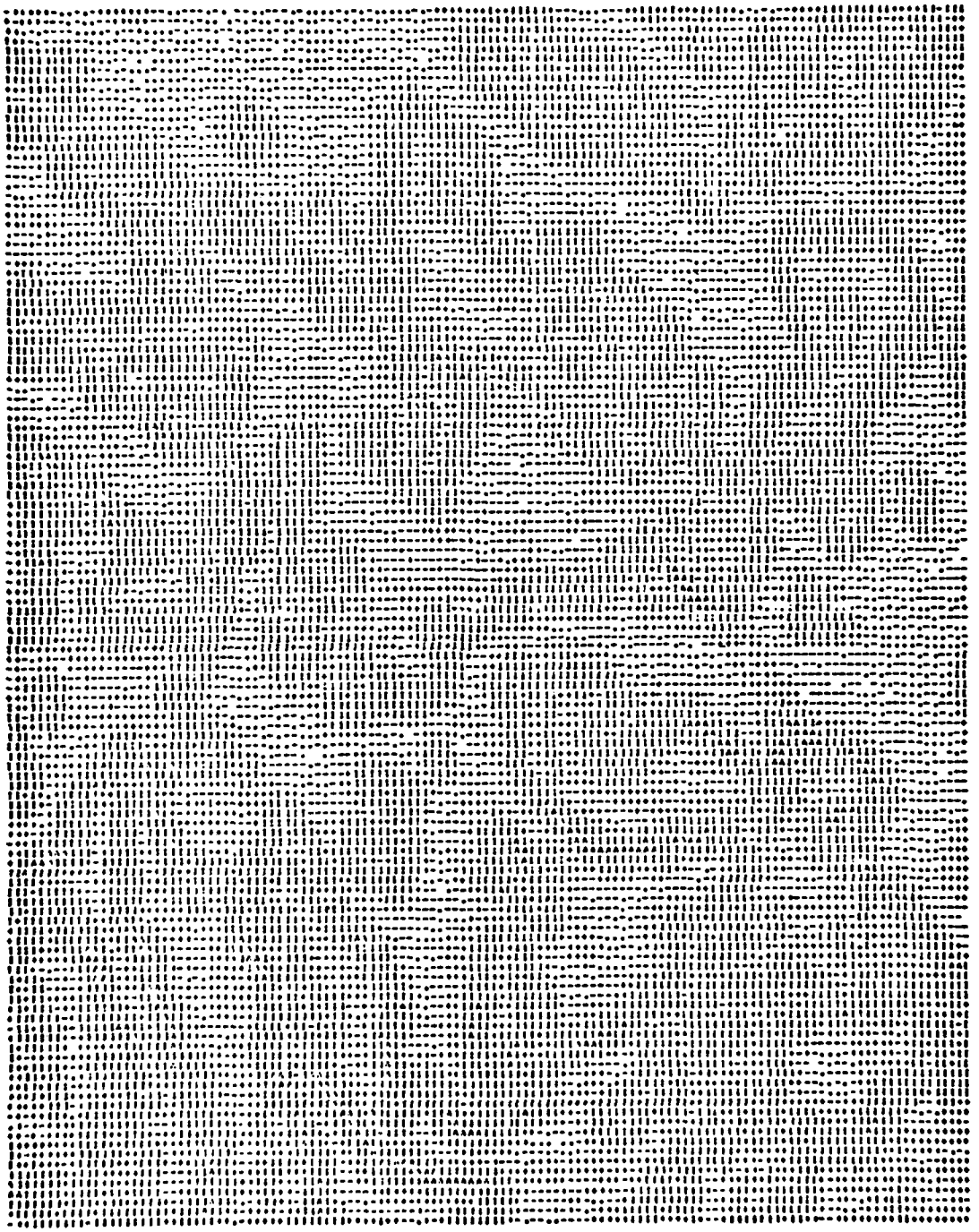


Figure 4.2: Point-to-Point Local Gradient

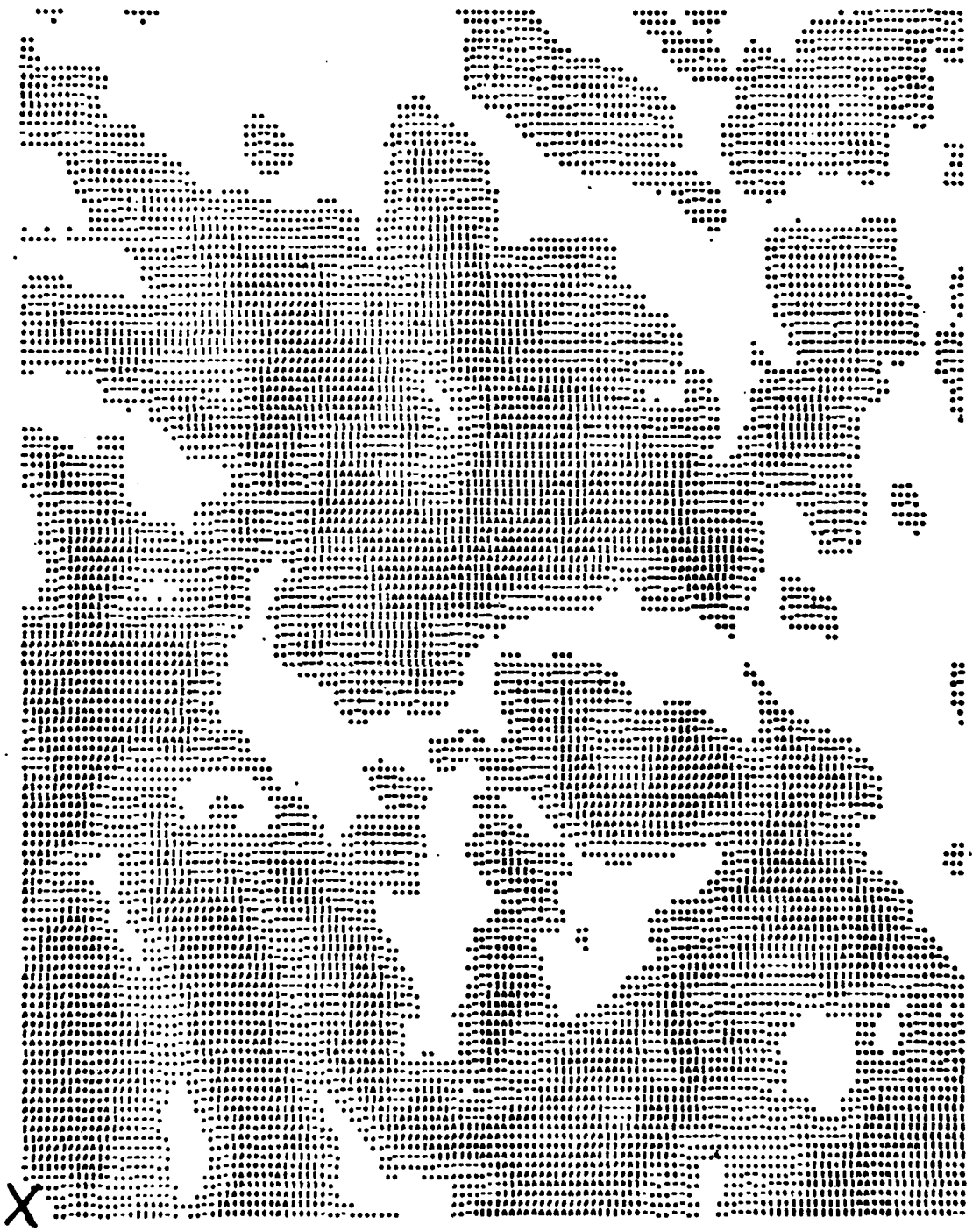


Figure 4.3: Location of Radar in Upper Quadrant

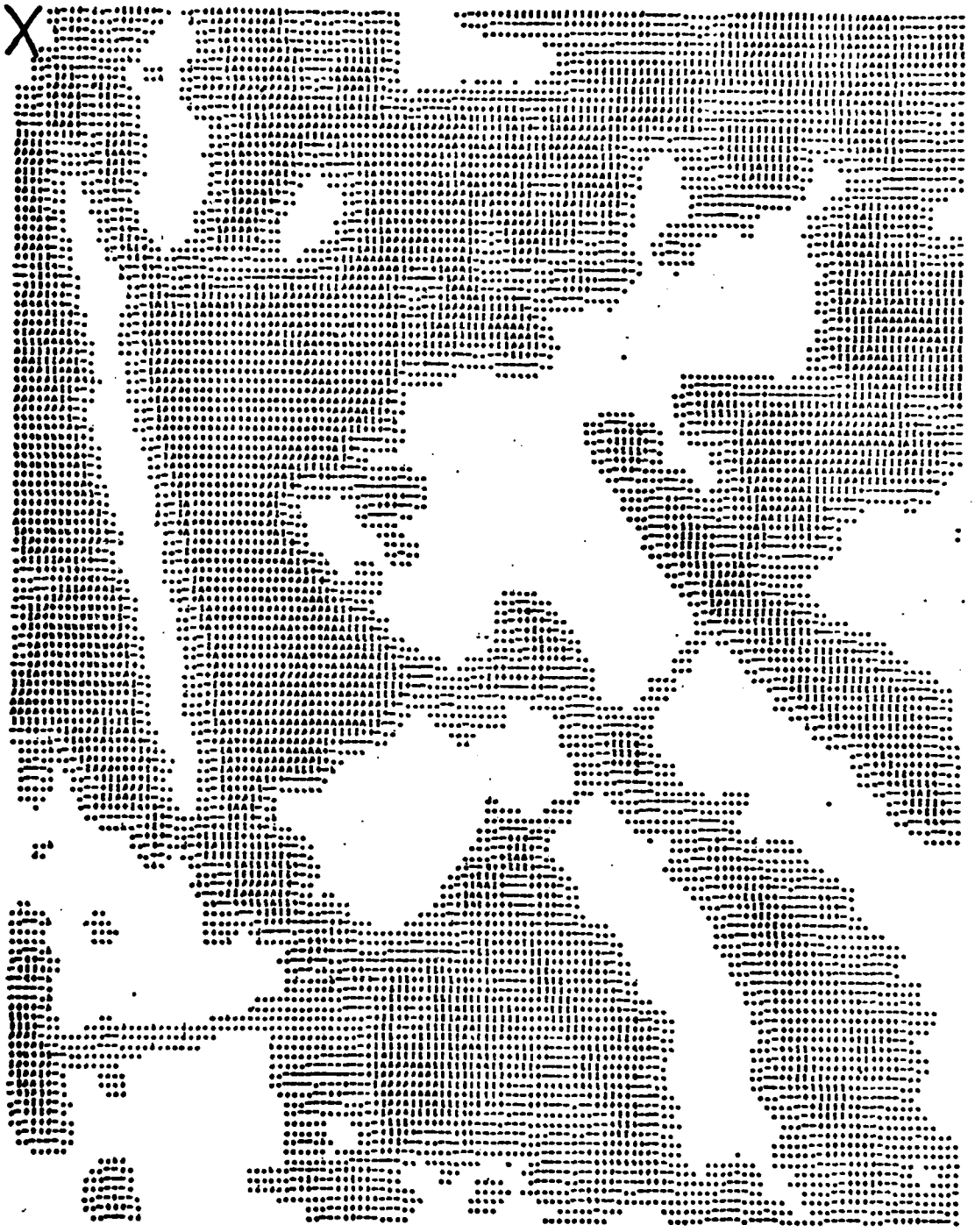


Figure 4.4: Location of Radar in Lower Quadrant

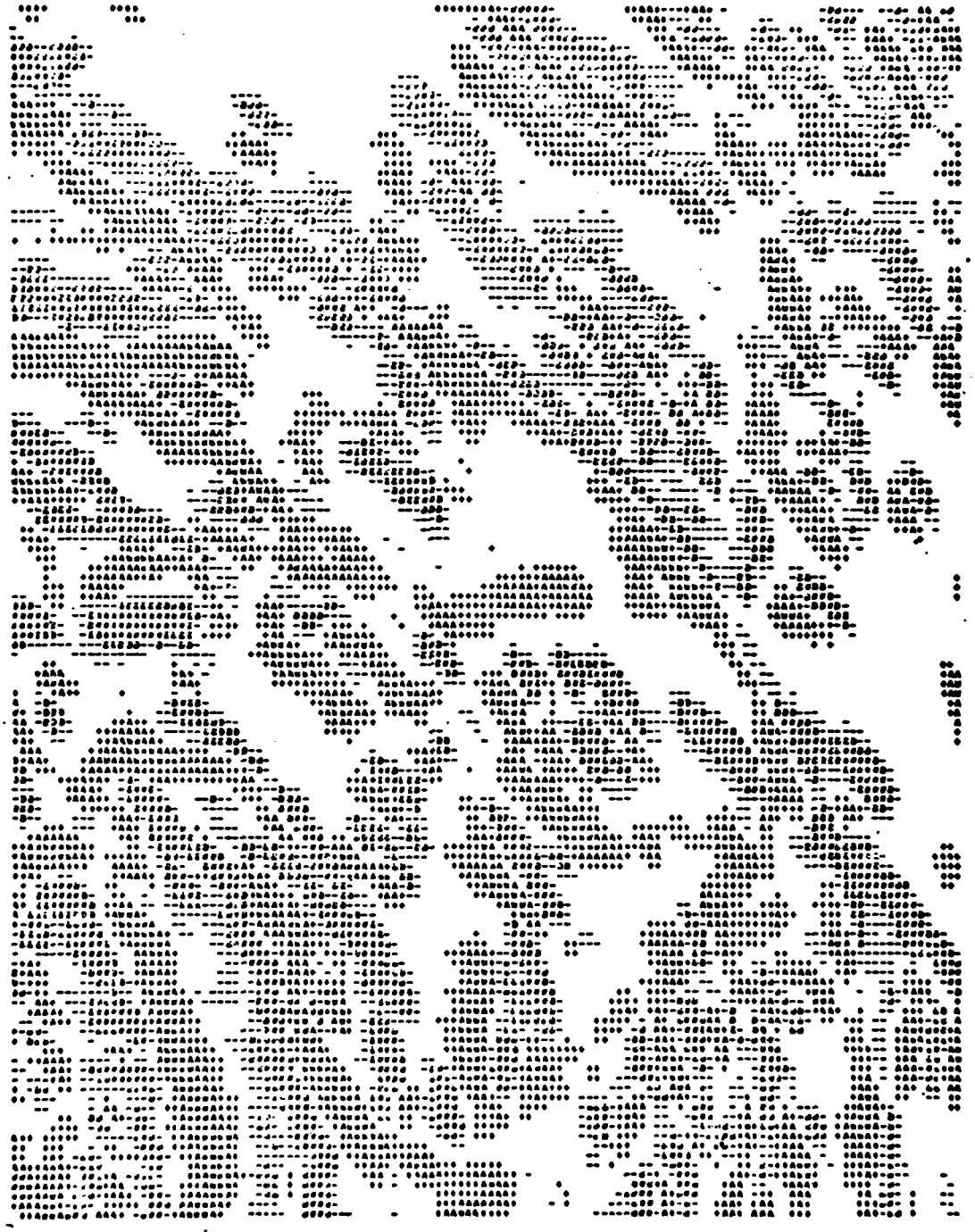


Figure 4.5: Radial Derivative, Upper Quadrant

Figure 4.6: Radial Derivative, Lower Quadrant

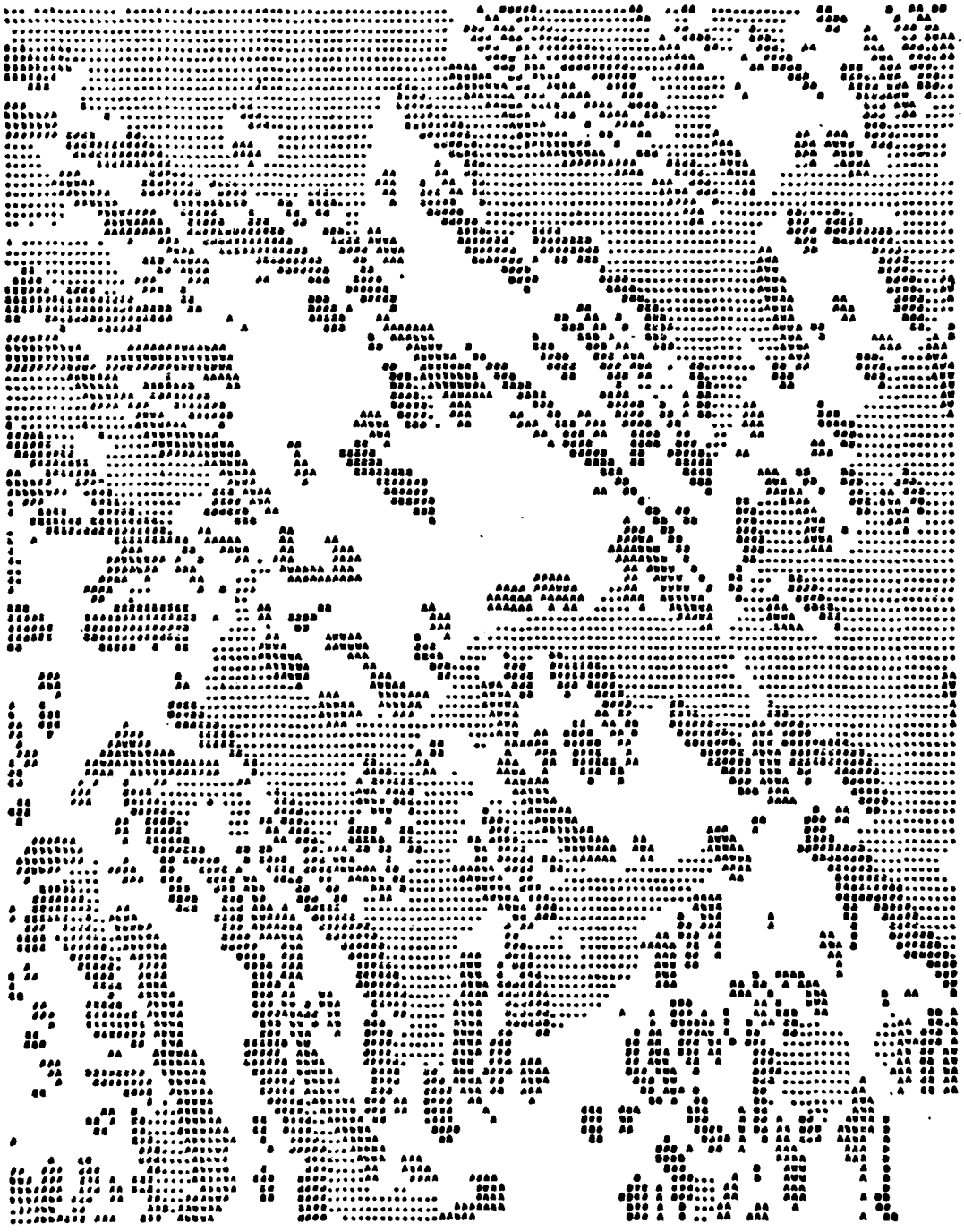


Figure 4.7: Radial Derivative and Shadows

Figure 4.8: Precursors of Shadows

```
C
C   THIS SUBROUTINE READS PARAMETERS, COMPUTES STORAGE AND CALLS MAJOR
C   PROGRAM SEGMENTS NEEDED FOR A NON-HIERARCHICAL CLUSTERING JOB USING
C   ONE OF THE METHODS PROGRAMMED AS A VERSION OF SUBROUTINE "KMEAN"
C
C   EVERY JOB REQUIRES THREE USER SUPPLIED DECK SEGMENTS.
C
C   1. PROGRAM "DRIVER" PERFORMS THE FOLLOWING TASKS.
C      A. ASSIGNS INPUT/OUTPUT UNITS.
C      B. ESTABLISHES THE DIMENSION OF THE "X" ARRAY AND SETS THIS
C         DIMENSION TO "LIMIT"
C      C. CALLS SUBROUTINE "EXEC2" (FORMERLY "EXEC")
C
C   2. SUBROUTINE "USER" IS EMPLOYED TO READ THE COMPLETE SET OF SCORES
C      ON THE VARIABLES FOR ONE DATA UNIT.
C
C   3. FUNCTION "DIST" COMPUTES THE DISTANCE BETWEEN TWO DATA UNITS OR
C      BETWEEN A DATA UNIT AND A CLUSTER CENTROID.  THE USER CAN SPECIFY
C      ANY DESIRED DISTANCE FUNCTION AND WEIGHT THE VARIABLES IN ANY
C      MANNER.
C
C   NOTE THAT SCALING AND TRANSFORMATION OF VARIABLES CAN BE
C   ACCOMPLISHED EITHER IN SUBROUTINE "USER" OR IN SUBROUTINE "DIST".
C
C**********************************************************************
C
C   NOTE:  MODIFIED SUBROUTINE "EXEC" IS CALLED "EXEC2", AND CALLS
C          "RSULT2".  "KMEAN" HAS NOT BEEN ALTERED.
C
C**********************************************************************
C
C----------------------------------------------------------------------
C   INPUT SPECIFICATIONS
C   CARD 1  TITLE
C   CARD 2  PARAMETER CARD
C     COLS  1- 5   NE=NUMBER OF ENTITIES (DATA UNITS)
C     COLS  9-13   NV=NUMBER OF VARIABLES
C     COLS 14-18   NC=NUMBER OF CLUSTERS
C     COLS 19-20   NTIN=INPUT UNIT FOR THE DATA SET
C                  NTIN=5, CARD READER
C                  NTIN=NE.GT.5, TAPE OR DISK FILE
C     COLS 21-22   NTOUT=OUTPUT UNIT FOR SAVING CLUSTER MEMBERSHIP LISTS
C                  NTOUT=7, CARD PUNCH
C                  NTOUT.LE.0, DO NOT SAVE MEMBERSHIP LISTS
C     COLS 23-27   MINREL=TERMINATION PARAMETER.  CLUSTERING ENDS WHEN A
C                        CYCLE THROUGH THE DATA SET RESULTS IN "MINREL"
C                        OR FEWER CHANGES IN CLUSTER MEMBERSHIPS
C                  MINREL.LE.0, ITERATE TO COMPLETE CONVERGENCE
C     COLS 28-29   IPART=INITIAL PARTITION PARAMETER
C                  IPART=1, SEED POINTS ARE SELECTED FROM THE DATA UNITS.
C                           READ THE SEQUENCE NUMBERS FOR THE CHOSEN DATA
C                           UNITS FORM CARD(S) 3 IN 20I4 FORMAT.  IF THE
C                           DATA SET IS NOT STORED IN CORE, THE LIST OF
C                           SEQUENCE NUMBERS MUST BE IN ASCENDING ORDER
C                  IPART=2, THE DATA UNITS ARE GROUPED INTO AN INITIAL
C                           PARTITION IN THE INPUT SEQUENCE WITH THE
C                           FIRST "NUMBR(1)" IN CLUSTER 1, THE NEXT
C                           "NUMBR(2)" IN CLUSTER 2 ETC.  READ THE
C                           "NUMBR" ARRAY FROM CARD(S) 3 IN 20I4 FORMAT.
C                  IPART=3, THE SCORE VECTORS FOR THE SEED POINTS ARE
C                           READ FROM CARD(S) 4 IN FORMAT "FMT" WHICH IS
C                           READ FROM CARD 3.
C     COLS 30-31   METHOD=PARAMETER FOR CHOOSING THE ALGORITHM IN ONE
C                        VERSION OF SUBROUTINE "KMEAN"
C                  METHOD=1, JANCEY ALGORITHM
C                  METHOD.NE.1, FORGY ALGORITHM
C
C*** CARDS 3 AND 4 ARE READ IN SUBROUTINE "KMEAN" ACCORDING TO THE
C*** PROCEDURE SPECIFIED BY THE CHOSEN VALUE OF "IPART".  NOTE THAT THE
C*** BASIC K-MEANS METHOD OF MACQUEEN SIMPLY USES THE FIRST "NC" DATA
C*** UNITS AS CLUSTER SEED POINTS AND THEREFORE IGNORES THE "IPART"
C*** PARAMETER.
C----------------------------------------------------------------------
C
C   STORAGE ALLOCATIONS IN THE "X" ARRAY
C   X(N1) TO X(N2-1)  NC*NV WORDS--STORAGE OF THE CENTR ARRAY
C   X(N2) TO X(N3-1)  NC WORDS--STORAGE OF THE NUMBR ARRAY
C   X(N3) TO X(N4-1)  NE WORDS--STORAGE OF THE MEMBR ARRAY
C   X(N4) TO X(N5-1)  NC*NV WORDS--STORAGE OF THE TOTAL ARRAY
C   X(N5) TO X(N6-1)  NV OR NE*NV WORDS--STORAGE OF THE DATA ARRAY
C   X(N6) TO X(N7)    NE WORDS--STORAGE OF THE LIST ARRAY IN "RESULT"
C
C   THE STORAGE OF THE "LIST" ARRAY AFTER THE "DATA" ARRAY NOW IMPLIES THAT
C   "MAX" MUST BE THE VALUE OF "N7".
C
```

Figure A.1: Forgy-Jancey Comments

```
C-----------------------------------------------------------------------
C  INPUT SPECIFICATIONS
C  CARD 1  TITLE
C  CARD 2  PARAMETER CARD
C    COLS  1- 5  NE=NUMBER OF ENTITIES (DATA UNITS)
C    COLS  6-10  NV=NUMBER OF VARIABLES
C    COLS 11-15  NC=NUMBER OF CLUSTERS
C    COLS 16-20  NEIN=INPUT UNIT FOR THE DATA SET
C                NEIN=5, CARD READER
C                NEIN.NE.5, TAPE OR DISK FILE
C    COLS 21-25  NTOUT=OUTPUT UNIT FOR SAVING CLUSTER MEMBERSHIP LISTS
C                NTOUT=7, CARD PUNCH
C                NTOUT.LE.0, DO NOT SAVE MEMBERSHIP LISTS
C    COLS 26-30  MINREL=TERMINATION PARAMETER.  CLUSTERING ENDS WHEN A
C                      CYCLE THROUGH THE DATA SET RESULTS IN "MINREL"
C                      OR FEWER CHANGES IN CLUSTER MEMBERSHIPS
C                MINREL.LE.0, ITERATE TO COMPLETE CONVERGENCE
C    COLS 31-35  IPART=INITIAL PARTITION PARAMETER
C                IPART=1, SEED POINTS ARE SELECTED FROM THE DATA UNITS.
C                         READ THE SEQUENCE NUMBERS FOR THE CHOSEN DATA
C                         UNITS FROM CARD(S) 2 IN 20I4 FORMAT.  IF THE
C                         DATA SET IS NOT STORED IN CORE, THE LIST OF
C                         SEQUENCE NUMBERS MUST BE IN ASCENDING ORDER
C                IPART=2, THE DATA UNITS ARE GROUPED INTO AN INITIAL
C                         PARTITION IN THE INPUT SEQUENCE WITH THE
C                         FIRST "NUMB(1)" IN CLUSTER 1, THE NEXT
C                         "NUMB(2)" IN CLUSTER 2 ETC.  READ THE
C                         "NUMBR" ARRAY FROM CARD(S) 3 IN 20I4 FORMAT.
C                IPART=3, THE SCORE VECTORS FOR THE SEED POINTS ARE
C                         READ FROM CARD(S) 4 IN FORMAT "FMT" WHICH IS
C                         READ FROM CARD 3.
C    COLS 36-40  METHOD=PARAMETER FOR CHOOSING THE ALGORITHM IN ONE
C                       VERSION OF SUBROUTINE "KMEAN"
C                METHOD=1, JANCEY ALGORITHM
C                METHOD.NE.1, FORGY ALGORITHM
C    COLS 41-48  NCMAX=MAX NUMBER OF CLUSTERS FOR OUTLIER ROUTINE
C                NCMAX=NC, DON'T DO OUTLIER ROUTINE
C                NCMAX .LE. 0, USE NE AS NCMAX (I.E. NO LIMIT ON CLUSTERS)
C    COLS 49-52  OUTLYR=COEFFICIENT (BETWEEN 0 AND 1 INCLUSIVE) FOR THE
C                       OUTLIER ROUTINE (NUMBER IN F4.2 FORMAT)
C                OUTLYR=1., MAX AMOUNT OF OUTLYING CLUSTERS
C                OUTLYR=0., NO OUTLYING CLUSTERS ALLOWED.

C*** CARDS 3 AND 4 ARE READ IN SUBROUTINE "KMEAN" ACCORDING TO THE
C*** PROCEDURE SPECIFIED BY THE CHOSEN VALUE OF "IPART".  NOTE THAT THE
C*** BASIC K-MEANS METHOD OF MACQUEEN SIMPLY USES THE FIRST "NC" DATA
C*** UNITS AS CLUSTER SEED POINTS AND THEREFORE IGNORES THE "IPART"
C*** PARAMETER.
C-----------------------------------------------------------------------

C  STORAGE ALLOCATIONS IN THE "X" ARRAY
C  X(N1) TO X(N2-1)  NC*NV WORDS--STORAGE OF THE CENTR ARRAY
C  X(N2) TO X(N3-1)  NC WORDS--STORAGE OF THE NUMBR ARRAY
C  X(N3) TO X(N4-1)  NE WORDS--STORAGE OF THE MEMBR ARRAY
C  X(N4) TO X(N5-1)  NC*NV WORDS--STORAGE OF THE TOTAL ARRAY
C  X(N5) TO X(N6-1)  NV OR NE*NV WORDS--STORAGE OF THE DATA ARRAY
C  X(N6) TO X(N7)    NE WORDS--STORAGE OF THE LIST ARRAY IN "RESULT"

C  THE STORAGE OF THE "LIST" ARRAY AFTER THE "DATA" ARRAY NOW IMPLIES THAT

C  "MAX" MUST BE THE VALUE OF "N7".
C
C**********************************************************************
C*                                                                    *
C* NOTE:                                                              *
C*    SINCE NC CAN BE INCREASED BY THE OUTLIER ROUTINE, THE           *
C*    ALLOCATIONS IN THE ARRAYS ARE BASED ON NCMAX, NOT NC.           *
C*    THIS CAN INCREASE STORAGE REQUIREMENTS FASTER THAN EXPECTED.    *
C*                                                                    *
C**********************************************************************
```

Figure A.2: Outlier Comments

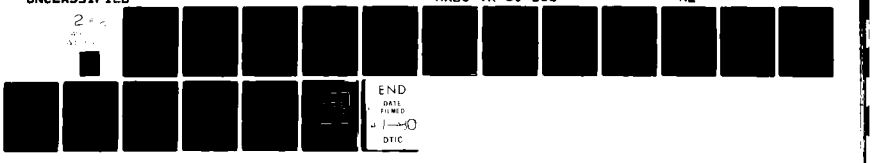
```
FORTRAN IV G LEVEL  21                  EXEC                DATE = 79365          09/56/58

0001                 SUBROUTINE EXEC(X,LIMIT)
                     C
                     C  THIS SUBROUTINE READS PARAMETERS, COMPUTES STORAGE AND CALL'S MAJOR
                     C  PROGRAM SEGMENTS NEEDED FOR A NON-HIERARCHIAL CLUSTERING JOB USING
                     C  ONE OF THE METHODS PROGRAMMED AS A VERSION OF SUBROUTINE "KMEAN"
                     C
                     C  MORE COMMENTS AND A FULL EXPLANATION CAN BE FOUND IN FILE "OD.FCOMMENT,
                     C  OR BY OBTAINING THE REPORT THAT WAS A SOURCE OF THIS PROGRAM.
                     C
0002                       DIMENSION X(1),TITLE(20)
0003                       READ (5,1000) TITLE
0004                       READ (5,1100) NE,NV,NC,NTIN,NTOUT,MINREL,IPART,METHOD
0005                       WRITE (6,2000) TITLE
0006                       WRITE (6,2100) NE,NV,NC,NTIN,NTOUT,MINREL,IPART,METHOD
0007                       N1=1
0008                       N2=N1+NC*NV
0009                       N3=N2+NC
0010                       N4=N3+NE
0011                       N5=N4+NC*NV
                     C  "N6" MAY (AND WILL) BE INCREASED IN "KMEAN"
0012                       N6=N5+NV-1
0013                       N7=N4+NE-1
0014                       MAX=N6
0015                       WRITE (6,2200) MAX,LIMIT
0016                       IF (MAX .GT. LIMIT) STOP
0017                       CALL KMEAN(X(N1),X(N2),X(N3),X(N4),X(N5),N5,NE,NV,NC,NTIN,MINREL,
                          * IPART,METHOD,LIMIT)
0018                       CALL RESULT(X(N1),X(N2),X(N3),X(N4),TITLE,NE,NV,NC,NTOUT,
                          * X(N6),X(N5))
0019                       RETURN
0020                1000   FORMAT(20A4)
0021                1100   FORMAT(8I5)
0022                2000   FORMAT('1',20A4)
0023                2100   FORMAT('0NE =',I5,/,' NV =',I8,/,' NC =',I8,/,' NTIN =',I6,/,
                          $ ' NTOJT =',I5,/,' MINREL =',I4,/,' IPART =',I5,/,' METHOD =',I4)
0024                2200   FORMAT('0REQUIRED STORAGE =',I8,' WORDS',/,
                          $       '0ALLOTTED STORAGE =',I8,' WORDS')
0025                       END
```

Figure A.3: Subroutine EXEC

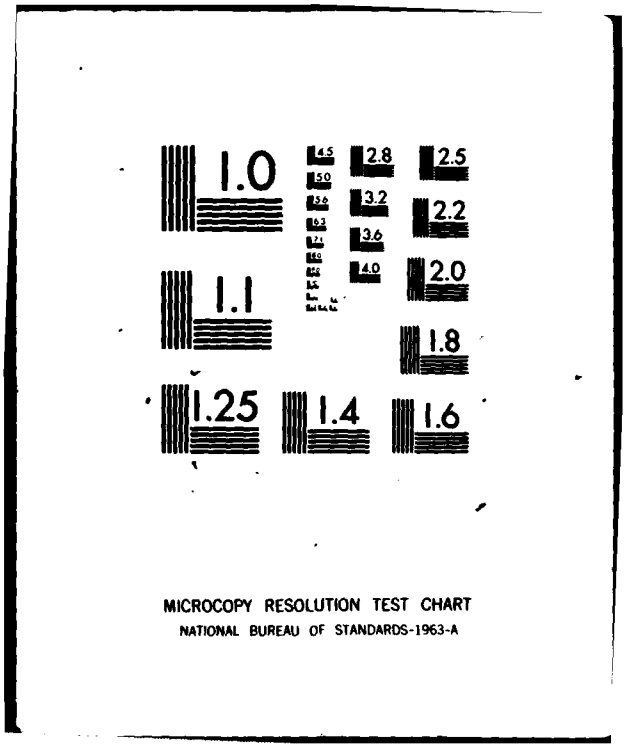
```
FORTRAN IV G LEVEL  21                  MAIN                DATE = 79365          10/02/46

0001          C  ALTERED SUBROUTINE EXEC--CALLS KMEAN, RSULT2
                 SUBROUTINE EXEC2(X,LIMIT)
              C
              C  THIS SUBROUTINE READS PARAMETERS, COMPUTES STORAGE AND CALL'S MAJOR
              C  PROGRAM SEGMENTS NEEDED FOR A NON-HIERARCHIAL CLUSTERING JOB USING
              C  ONE OF THE METHODS PROGRAMMED AS A VERSION OF SUBROUTINE "KMEAN"
              C
              C  MORE COMMENTS AND A FULL EXPLANATION CAN BE FOUND IN FILE "JJD.FCOMMENT",
              C  OR BY OBTAINING THE REPORT THAT WAS A SOURCE OF THIS PROGRAM.
              C  ---CHANGES MADE TO "N6", "N7", SUBROUTINE "RESULT" IS "RSULT2",
              C  ---CHANGED INPUT FORMATS (DOCUMENTED IN ROSCOE FILE "FCOMMENT")
              C
0002                  DIMENSION X(1),TITLE(20)
0003                  READ (5,1000) TITLE
0004                  READ (5,1100) NE,NV,NC,NTIN,NTOUT,MINREL,IPART,METHOD
0005                  WRITE (6,2000) TITLE
0006                  WRITE (6,2100) NE,NV,NC,NTIN,NTOUT,MINREL,IPART,METHOD
0007                  N1=1
0008                  N2=N1+NC*NV
0009                  N3=N2+NC
0010                  N4=N3+NE
0011                  N5=N4+NC*NV
              C  "N6" IS NOT THE SAME AS IN ORIGINAL PROGRAM, ALSO "N7" CHANGED, TOO
0012                  N6=N5+NV*NE
0013                  N7=N6+NE-1
0014                  MAX=N7
0015                  WRITE (6,2200) MAX,LIMIT
0016                  IF (MAX .GT. LIMIT) STOP
0017                  CALL KMEAN(X(N1),X(N2),X(N3),X(N4),X(N5),N5,NE,NV,NC,NTIN,MINREL,
                     * IPART,METHOD,,LIMIT)
0018                  CALL RSULT2(X(N1),X(N2),X(N3),X(N6),TITLE,NE,NV,NC,NTOJT,
                     * X(N4),X(N5))
0019                  RETURN
0020          1000    FORMAT(20A4)
0021          1100    FORMAT(I9,2I5,2I2,I5,2I2)
0022          2000    FORMAT(1H1,20A4)
0023          2100    FORMAT('ONE =',I9,/,' NV =',I9,/,' NC =',I8,/,' NTIN =',I8,/,
                     * ' NTOJT =',I5,/,' MINREL =',I6,/,' IPART =',I5,/,' METHOD =',I4)
0024          2200    FORMAT('0REQUIRED STORAGE =',I8,' WORDS',/,
                     *       '0ALLOTTED STORAGE =',I8,' WORDS')
0025                  END
```

Figure A.4: Subroutine EXEC2

AD-A089 702 CLARKSON COLL OF TECHNOLOGY POTSDAM N Y DEPT OF ELEC--ETC F/G 17/9
MODELING OF SHADOWS IN RADAR CLUTTER.(U)
JUL 80 B A BLACK, M AROZULLAH, W LADEW F30602-78-C-0102
UNCLASSIFIED RADC-TR-80-238 NL



END
DATE FILMED
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
FORTRAN IV G LEVEL 21                MAIN              DATE = 79365        10/21/29

                C  ALTERED SUBROUTINE EXEC--CALLS KMEAN2, RSULT2
 0001                 SUBROUTINE EXEC(X,LIMIT)
                C
                C  THIS SUBROUTINE READS PARAMETERS, COMPUTES STORAGE AND CALLS MAJOR
                C  PROGRAM SEGMENTS NEEDED FOR A NON-HIERARCHICAL CLUSTERING JOB USING
                C  ONE OF THE METHODS PROGRAMMED AS A VERSION OF SUBROUTINE "KMEAN2"
                C
                C  MORE COMMENTS AND A FULL EXPLANATION CAN BE FOUND IN FILE "OD.FCCOMMENT"
                C  OR BY OBTAINING THE REPORT THAT WAS A SOURCE OF THIS PROGRAM.
                C  ---CHANGES MADE TO "N6", "N7", SUBROUTINE "RESULT" IS "RSULT2".
                C  ---CHANGED INPUT FORMATS (DOCUMENTED IN ROSCOE FILE "FCOMMENT")
                C-----------------------------------------------------------------
                C  INPUT SPECIFICATIONS
                C  CARD 1  TITLE
                C  CARD 2  PARAMETER CARD
                C    COLS  1- 4   NE=NUMBER OF ENTITIES (DATA UNITS)
                C    COLS  5-11   NV=NUMBER OF VARIABLES
                C    COLS 14-16   NC=NUMBER OF CLUSTERS
                C    COLS 19-20   NTIN=INPUT UNIT FOR THE DATA SET
                C                 NTIN=5, CARD READER
                C                 NTIN=NF,S, TAPE OR DISK FILE
                C    COLS 21-22   NTOUT=OUTPUT UNIT FOR SAVING CLUSTER MEMBERSHIP LISTS
                C                 NTOUT=7, CARD PUNCH
                C                 NTOUT.LE.0, DO NOT SAVE MEMBERSHIP LISTS
                C    COLS 23-27   %(NREL=TERMINATION PARAMETER.  CLUSTERING ENDS WHEN A
                C                      CYCLE THROUGH THE DATA SET RESULTS IN "NINREL"
                C                      OR FEWER CHANGES IN CLUSTER MEMBERSHIPS
                C                 NINREL.LE.0, ITERATE TO COMPLETE CONVERGENCE
                C    COLS 28-29   IPART=INITIAL PARTITION PARAMETER
                C                 IPART=1, SEED POINTS ARE SELECTED FROM THE DATA UNITS.
                C                      READ THE SEQUENCE NUMBERS FOR THE CHOSEN DATA
                C                      UNITS FROM CARD(S) 3 IN 2016 FORMAT.  IF THE
**************01)  [EY033I COMMENTS  DELETED ********************************************
 0002                 DIMENSION X(1),TITLE(20)
 0003                 READ (5,1000) TITLE
 0004                 READ (5,1100) NE,NV,NC,NTIN,NTOUT,NINREL,IPART,METHOD,NCMAX,
                     * OUTLYR
 0005                 WRITE (6,2000) TITLE
 0006                 WRITE (6,2100) NE,NV,NC,NTIN,NTOUT,NINREL,IPART,METHOD,NCMAX,
                     * OUTLYR
 0007                 IF (NCMAX .LE. 0) NCMAX=NE
 0008                 N1=1
 0009                 N2=N1+NC*MAX*NV
 0010                 N3=N2+NC*MAX
 0011                 N4=N3+NE
 0012                 N5=N4+NC*MAX*NV
                C  "N6" IS NOT THE SAME AS IN ORIGINAL PROGRAM. ALSO "N7" CHANGED, TOO
 0013                 N6=N5+NV*NE
 0014                 N7=N6+NE-1
 0015                 MAX=N7
 0016                 WRITE (6,2200) MAX,LIMIT
 0017                 IF (MAX .GT. LIMIT) CALL SVC(15)
 0018                 CALL KMEAN2(X(N1),X(N2),X(N3),X(N4),X(N5),NS,NE,NV,NC,NTIN,NINREL,
```

```
FORTRAN IV G LEVEL 21                EXEC3             DATE = 79365        10/21/29

                     * IPART,METHOD,LIMIT,NCMAX,OUTLYR)
 0019                 CALL RSULT2(X(N1),X(N2),X(N3),X(N6),TITLE,NE,NV,NC,NTOUT,
                     * X(N4),X(N5))
 0020                 RETURN
 0021           1000  FORMAT(20A4)
 0022           1100  FORMAT(I4,2I5,2I2,I5,2I2,I8,F4.2)
 0023           2000  FORMAT(1H1,20A4)
 0024           2100  FORMAT('ONE =',I5,/,' NV =',I8,/,' NC =',I8,/,' NTIN =',I8,/,
                     * ' NTOUT =',I5,/,' NINREL =',I4,/,' IPART =',I5,/,' METHOD =',I4,
                     * /,' NCMAX =',I5,/,' OUTLYR =',F4.2)
 0025           2200  FORMAT('0REQUIRED STORAGE =',I8,' WORDS',/,
                     *       'ALLOTTED STORAGE =',I8,' WORDS')
 0026                 END
```

Figure A.5: Subroutine EXEC3

```
FORTRAN IV G LEVEL 21                KMEAN              DATE = 79365          09/51/38

0001            SUBROUTINE KMEAN(CENTR,NUMBR,MEMBR,TOTAL,DATA,NS,NE,NV,NC,NTIN,
               * MINMLL,IPART,METHOD,LIMIT)
                C
                C-----------------------------------------------------------------
                C  VERSION 1., THE DATA SET IS STORED IN CENTRAL MEMORY.
                C-----------------------------------------------------------------
                C
                C  THIS SUBROUTINE ITERATIVELY SORTS "NE" DATA UNITS INTO "NC" CLUSTERS
                C  USING THE ALGORITHM OF (METHOD .NE. 1)
                C
                C   FORGY, E.W., CLUSTER ANALYSIS OF MULTIVARIATE DATA, EFFICIENCY
                C   VERSUS INTERPRETABILITY OF CLASSIFICATIONS, PAPER(A ABSTRACT) AT THE
                C   BIOMETRIC SOCIETY (WNAR) MEETINGS, RIVERSIDE, CALIFORNIA, JUNE
                C   1965.  ABSTRACT IN BIOMETRICS, VOLUME 21, NUMBER 3, P 768.
                C
                C  OR THE ALGORITHM OF (METHOD = 1)
                C
                C   JANCEY, R.C., MULTIDIMENSIONAL GROUP ANALYSIS, AUSTRALIAN JOURNAL
                C   OF BOTANY, VOLUME 14, NUMBER 1, APRIL 1966, PP 127-130.
                C
                C  CENTR(NV*(J-1)+I)=SCORE ON THE I-TH VARIABLE FOR J-TH CLUSTER CENTROID
                C  TOTAL(NV*(J-1)+I)=TOTAL SCORE ON I-TH VARIABLE FOR DATA UNITS THUS
                C                   FAR ALLOCATED TO THE J-TH CLUSTER
                C  NUMBR(J)=NUMBER OF DATA UNITS THUS FAR ALLOCATED TO THE J-TH CLUSTER
                C  MEMBR(K)=CLUSTER TO WHICH THE K-TH DATA UNIT CURRENTLY BELONGS
                C  DATA(NV*(K-1)+I)=SCORE ON I-TH VARIABLE FOR K-TH DATA UNIT
                C
0002                DIMENSION CENTR(1),TOTAL(1),NUMBR(1),MEMBR(1),DATA(1),FMT(20),
                  * NAME(4)
0003                DATA NAME/'   F','ORGY',' JA','NCEY'/
0004                I=1
0005                IF (METHOD .EQ. 1) I=3
0006                WRITE(6,2000) NAME(I),NAME(I+1)
                C  CHECK FOR SUFFICIENT STORAGE
0007                NS=NS+NE*NV-1
0008                WRITE(6,2100) NS,LIMIT
0009                IF (NS .GT. LIMIT) STOP
                C  ESTABLISH INITIAL PARTITION
0010                IF (IPART .NE. 3) GO TO 20
                C  SEED POINTS ARE READ DIRECTLY FROM CARDS
0011                READ (5,1000) FMT
0012                WRITE(6,2200) FMT
0013                WRITE(6,2300)
0014                J1=0
0015                DO 10 J=1,NC
0016                READ(5,FMT) (CENTR(J1+I),I=1,NV)
0017                WRITE(6,2400) (CENTR(J1+I),I=1,NV)
0018           10   J1=J1+NV
0019                GO TO 30
                C  IPART=1 OR 2
0020           20   WRITE(6,2500) IPART
0021                READ(5,1100) (NUMBR(J),J=1,NC)
0022                WRITE(6,2600) (NUMBR(J),J=1,NC)
```

Figure A.6 (Part 1): Subroutine KMEAN

```
FORTRAN IV G LEVEL  21                KMEAN              DATE = 79365         09/51/30

      C  READ THE DATA SET INTO CENTRAL MEMORY
      30     K1=1
             DO 40 K=1,NE
             CALL USER (DATA(K1))
      40     K1=K1+NV
             IF (IPART .EQ. 3) GO TO 100
      C  IF "IPART" IS 1 OR 2 SET UP THE SEED POINTS
             IF (IPART .EQ. 2) GO TO 60
      C  IPART=1.  THE DATA UNIT WITH SEQUENCE NUMBER "NUMBR(J)" IS USED AS
      C  THE J-TH SEED POINT
             DO 50 J=1,NC
             NJ=(NUMBR(J)-1)*NV
             J1=(J-1)*NV
             DO 50 I=1,NV
             CENTR(J1+I)=DATA(NJ+I)
      50     CONTINUE
             GOTO 130
      C  IPART=2.  THE DATA UNITS ARE GROUPED INTO CLUSTERS WITH THE J-TH
      C  CLUSTER HAVING "NUMBR(J)" MEMBERS.
      60     K=0
             J1=-NV
      C  ACCUMULATE THE TOTAL SCORE ON EACH VARIABLE FOR EACH CLUSTER
             DO 80 J=1,NC
             NJ=NUMBR(J)
             J1=J1+NV
             DO 70 I=1,NV
      70     TOTAL(J1+I)=0.
             DO 80 KJ=1,NJ
             K=K+1
             MEMBR(K)=J
             K1=(K-1)*NV
             DO 80 I=1,NV
             J2=J1+I
             TOTAL(J2)=TOTAL(J2)+DATA(K1+I)
      80     CONTINUE
      C  COMPUTE THE CENTROIDS
             J1=0
             DO 90 J=1,NC
             DO 90 I=1,NV
             J1=J1+1
             CENTR(J1)=TOTAL(J1)/NUMBR(J)
      90     CONTINUE
             GO TO 115
      C  INITIALIZE ARRAYS
      100    DO 110 K=1,NE
      110    MEMBR(K)=0
      115    NPASS=1
      C  BEGINNING OF MAIN LOOP
      120    J1=0
             DO 130 J=1,NC
             NUMBR(J)=0
             DO 130 I=1,NV
             J1=J1+1
```

Figure A.6 (Part 2): Subroutine KMEAN

```
FORTRAN IV G LEVEL  21                KMEAN               DATE = 79365          09/51/30

0066      130   TOTAL(J1)=0.
0067            MOVES=0
0068            TDIST=0
          C  ALLOCATE EACH DATA UNIT TO THE NEAREST CLUSTER CENTROID
0069            K1=0
0070            DO 160 I=1,NF
0071            K2=K1+1
0072            J2=1
          C  COMPUTE DISTANCE TO FIRST CLUSTER CENTROID
0073            DREF=DIST(DATA(K2),CENTR(J2))
0074            JREF=1
          C  TEST DISTANCES TO REMAINING CLUSTER CENTROIDS
0075            DO 140 J=2,NC
0076            J2=J2+NV
0077            DTEST=DIST(DATA(K2),CENTR(J2))
0078            IF (DTEST .GE. DREF) GO TO 140
0079            DREF=DTEST
0080            JREF=J
0081      140   CONTINUE
          C  ALLOCATE DATA UNIT "K" TO CLUSTER "JREF"
0082            NUMBR(JREF)=NUMBR(JREF)+1
0083            TDIST=TDIST+DREF
0084            IF (JREF .EQ. MEMBR(K)) GO TO 150
          C  THE DATA UNIT CHANGES ITS MEMBERSHIP
0085            MOVES=MOVES+1
0086            MEMBR(K)=JREF
0087      150   J1=(JREF-1)*NV
0088            DO 160 I=1,NV
0089            J1=J1+1
0090            K1=K1+1
0091            TOTAL(J1)=TOTAL(J1)+DATA(K1)
0092      160   CONTINUE
          C  ALL DATA UNITS ALLOCATED.  TEST FOR CONVERGENCE
0093            WRITE(6,2700) MOVES,NPASS,TDIST
0094            NPASS=NPASS+1
0095            JREF=0
0096            IF (MOVES .GT. MINREL) GO TO 185
0097            IF (METHOD .NE. 1  .AND.  MOVES .EQ. 0) RETURN
0098            JREF=1
          C  COMPUTE TRUE CLUSTER CENTROIDS--FORGY UPDATE
0099      170   J1=0
0100            DO 180 J=1,NC
0101            DO 180 I=1,NV
0102            J1=J1+1
0103      180   CENTR(J1)=TOTAL(J1)/NUMBR(J)
0104            IF (JREF .EQ. 1) RETURN
0105            GO TO 120
          C
0106      185   IF (METHOD .NE. 1) GO TO 170
          C  JANCEY UPDATE
0107      190   J1=0
0108            DO 200 J=1,NC
0109            DO 200 I=1,NV

FORTRAN IV G LEVEL  21                KMEAN               DATE = 79365          09/51/30

0110            J1=J1+1
0111      200   CENTR(J1)=2.*TOTAL(J1)/NUMBR(J)-CENTR(J1)
0112            GO TO 120
0113      1000  FORMAT(20A4)
0114      1100  FORMAT(20I4)
0115      2000  FORMAT(1H0,2A4,
               * ' METHOD OF CLUSTER ANALYSIS.  DATA SET STORED IN CORE')
0116      2100  FORMAT('0REQUIRED STORAGE =',I5,' WORDS',/,
               *       ' ALLOTTED STORAGE =',I5,' WORDS')
0117      2200  FORMAT('0FORMAT',20A4)
0118      2300  FORMAT('1INITIAL CLUSTER CENTERS READ IN AS FOLLOWS',///)
0119      2400  FORMAT(1X,10F12.4)
0120      2500  FORMAT('1 IPART =',I2,',  NUMBR ARRAY READ AS FOLLOWS',///)
0121      2600  FORMAT(1X,10I7)
0122      2700  FORMAT(1H0,I5,' DATA UNITS MOVED ON ITERATION NUMBER',I3,/,
               * ' SUMMED DEVIATIONS ABOUT SEED POINTS =',E16.2)
0123            END
```

Figure A.6 (Part 3): Subroutine KMEAN

```
FORTRAN IV G LEVEL  21                KMEAN2          DATE = 79365        09/54/01

0001                  SUBROUTINE KMEANS(CENTR,TOTAL,NUMBR,MEMBR,DATA,NS,NF,NV,NC,NTIN,
                     * MINVEL,IPART,METHD,LIMIT,NCMAX,OUTLYR)
                C
                C  CENTR(NV*(J-1)+I)=SCORE ON THE I-TH VARIABLE FOR J-TH CLUSTER CENTROID
                C  TOTAL(NV*(J-1)+I)=TOTAL SCORE ON I-TH VARIABLE FOR DATA UNITS THUS
                C                    FAR ALLOCATED TO THE J-TH CLUSTER
                C  NUMBR(J)=NUMBER OF DATA UNITS THUS FAR ALLOCATED TO THE J-TH CLUSTER
                C  MEMBR(K)=CLUSTER TO WHICH THE K-TH DATA UNIT CURRENTLY BELONGS
                C  DATA(NV*(K-1)+I)=SCORE ON I-TH VARIABLE FOR K-TH DATA UNIT
                C
0002                  DIMENSION CENTR(1),TOTAL(1),NUMBR(1),MEMBR(1),DATA(1),FMT(20),
                     * NAME(4)
0003                  DATA NAME/'   F','ORGY','   M','ECEV'/
0004                  I=1
0005                  IF (METHD .EQ. 1) I=3
0006                  WRITE(6,2000) NAME(I),NAME(I+1)
                C  ESTABLISH INITIAL PARTITION
0007                  IF (IPART .NE. 3) GO TO 20
                C  SEED POINTS ARE READ DIRECTLY FROM CARDS
0008                  READ (5,1000) FMT
0009                  WRITE(6,2200) FMT
0010                  WRITE(6,2300)
0011                  JI=0
0012                  DO 10 J=1,NC
0013                  READ(5,FMT) (CENTR(JI+I),I=1,NV)
0014                  WRITE(6,2400) (CENTR(JI+I),I=1,NV)
0015             10   JI=JI+NV
0016                  GO TO 30
                C  IPART=1 OR 2
0017             20   WRITE(6,2500) IPART
0018                  READ(5,1100) (NUMBR(J),J=1,NC)
0019                  WRITE(6,2600) (NUMBR(J),J=1,NC)
                C  READ THE DATA SET INTO CENTRAL MEMORY
0020             30   K1=1
0021                  DO 40 K=1,NS
0022                  CALL USER (DATA(K1))
0023             40   K1=K1+NV
0024                  IF (IPART .EQ. 3) GO TO 100
                C  IF "IPART" IS 1 OR 2 SET UP THE SEED POINTS
0025                  IF (IPART .EQ. 2) GO TO 60
                C  IPART=1.  THE DATA UNIT WITH SEQUENCE NUMBER "NUMBR(J)" IS USED AS
                C  THE J-TH SEED POINT
0026                  DO 50 J=1,NC
0027                  NJ=(NUMBR(J)-1)*NV
0028                  JI=(J-1)*NV
0029                  DO 50 I=1,NV
0030                  CENTR(JI+I)=DATA(NJ+I)
0031             50   CONTINUE
0032                  GOTO 100
                C  IPART=2.  THE DATA UNITS ARE GROUPED INTO CLUSTERS WITH THE J-TH
                C  CLUSTER HAVING "NUMBR(J)" MEMBERS.
0033             60   K=0
0034                  JI=-NV
```

Figure A.7 (Part 1): Subroutine KMEAN2

```
FORTRAN IV G LEVEL  21                KMEAN2            DATE = 79368          09/54/01

      C  ACCUMULATE THE TOTAL SCORE ON EACH VARIABLE FOR EACH CLUSTER
            DO 80 J=1,NC
            NJ=NUMBR(J)
            J1=J1+NV
            DO 70 I=1,NV
      70    TOTAL(J1+I)=0.
            DO 80 KJ=1,NJ
            K=K+1
            MEMBR(K)=J
            K1=(K-1)*NV
            DO 80 I=1,NV
            J2=J1+I
            TOTAL(J2)=TOTAL(J2)+DATA(K1+I)
      80    CONTINUE
      C  COMPUTE THE CENTROIDS
            J1=0
            DO 90 J=1,NC
            DO 90 I=1,NV
            J1=J1+1
      C TEST FOR NUMBR(J)=0
            CENTR(J1)=TOTAL(J1)/NUMBR(J)
      90    CONTINUE
            GO TO 115
      C  INITIALIZE ARRAYS
      103   DO 110 K=1,NE
      110   MEMWN(K)=0
      115   NPASS=1
      C  BEGINNING OF MAIN LOOP
      120   J1=0
            DO 130 J=1,NCMAX
            NUMBR(J)=0
            DO 130 I=1,NV
            J1=J1+1
      130   TOTAL(J1)=0.
            MOVES=0
            TDIST=0
      C  ALLOCATE EACH DATA UNIT TO THE NEAREST CLUSTER CENTROID
            K1=0
            DO 160 K=1,NE
            K2=K1+1
            J2=1
      C  COMPUTE DISTANCE TO FIRST CLUSTER CENTROID
            DREF=DIST(DATA(K2),CENTR(J2))
            SUMDST=DREF
            JREF=1
      C  TEST DISTANCES TO REMAINING CLUSTER CENTROIDS
            DO 140 J=2,NC
            J2=J2+NV
            DTEST=DIST(DATA(K2),CENTR(J2))
            SUMDST=SUMDST+DTEST
            IF (DTEST .GE. DREF) GO TO 140
            DREF=DTEST
            JREF=J
```

Figure A.7 (Part 2): Subroutine KMEAN2

```
FORTRAN IV G LEVEL  21                 KMEAN2            DATE = 79365         09/54/01

        140    CONTINUE
               AVDIST=SUMDST/NC
        C  TEST FOR OUTLIERS, AND ALLOCATE NEW CLUSTERS IF ALLOWED.
               IF (.NOT. ((ABS(DREF-AVDIST) .LE. AVDIST*OUTLYR)
             *         .AND. (NC .LT. NCMAX))) GO TO 145
        C  NEED NEW CLUSTER
               NC=NC+1
               JREF=NC
               DO 145 L=1,NV
               CENTR(NV*(JREF-1)+L)=DATA(KI+L)
        145    CONTINUE
        C  ALLOCATE DATA UNIT "K" TO CLUSTER "JREF"
               NUMBR(JREF)=NUMBR(JREF)+1
               TDIST=TDIST+DREF
               IF (JREF .EQ. MEMBR(K)) GO TO 150
        C  THE DATA UNIT CHANGES ITS MEMBERSHIP
               MOVES=MOVES+1
               MEMBR(K)=JREF
        150    J1=(JREF-1)*NV
               DO 160 I=1,NV
               J1=J1+1
               KI=KI+1
               TOTAL(J1)=TOTAL(J1)+DATA(KI)
        160    CONTINUE
        C  ALL DATA UNITS ALLOCATED.  TEST FOR CONVERGENCE
               WRITE(6,2700) MOVES,NPASS,TDIST,NC
               NPASS=NPASS+1
               JREF=0
               IF (MOVES .GT. MINREL) GO TO 185
               IF (METHOD .NE. 1  .AND.  MOVES .EQ. 0) RETURN
               JREF=1
        C  COMPUTE TRUE CLUSTER CENTROIDS--FORGY UPDATE
        170    J1=0
               DO 180 J=1,NC
               DO 180 I=1,NV
               J1=J1+1
               IF (NUMBR(J) .LE.0) GO TO 180
               CENTR(J1)=TOTAL(J1)/NUMBR(J)
        180    CONTINUE
               IF (JREF .EQ. 1) RETURN
               GO TO 120
        C
        185    IF (METHOD .NE. 1) GO TO 170
        C  JANCEY UPDATE
        190    J1=0
               DO 200 J=1,NC
               DO 200 I=1,NV
               J1=J1+1
        C TEST FOR NUMBR=0 HERE, TOO...
        200    CENTR(J1)=2.*TOTAL(J1)/NUMBR(J)-CENTR(J1)
               GO TO 120
        1000   FORMAT(2X,A4)
        1100   FORMAT(20I4)
```

```
FORTRAN IV G LEVEL  21                 KMEAN2            DATE = 79365         09/54/01

        2000   FORMAT(1H0,2A4,
             * ' METHOD OF CLUSTER ANALYSIS.  DATA SET STORED IN CORE')
        2100   FORMAT('0REQUIRED STORAGE =',I5,' WORDS',/,
             *        '0ALLOTTED STORAGE =',I5,' WORDS')
        2200   FORMAT('0FORMAT',20A4)
        2300   FORMAT('1INITIAL CLUSTER CENTERS READ IN AS FOLLOWS',///)
        2400   FORMAT(1X,10F12.4)
        2500   FORMAT('1 (PART =',I2,'),  NUMBR ARRAY READ AS FOLLOWS',///)
        2600   FORMAT(1X,10I7)
        2700   FORMAT(1H0,I5,' DATA UNITS MOVED ON ITERATION NUMBER',I3,/,
             * ' SUMMED DEVIATIONS ABOUT SEED POINTS =',E16.2,10X,'NC =',I5)
               END
```

Figure A.7 (Part 3): Subroutine KMEAN2

```
FORTRAN IV G LEVEL  21                  RESULT              DATE = 79365        10/29/05

0001                  SUBROUTINE RESULT(CENTR,NUMBR,MEMBR,LIST,TITLE,NV,NC,NTOUT,
                     * TOTAL,DATA)
                C  THIS SUBROUTINE PRINTS THE RESULTS FROM A CLUSTERING JOB BASED
                C  ON ANY VERSION OF SUBROUTINE "KMEAN".
                C
0002                  DIMENSION CENTR(1),NUMBR(1),MEMBR(1),LIST(1),TITLE(20),TOTAL(1),
                     * DATA(1)
                C
                C  AS A CONTINGENCY PRECAUTION WRITE OUT THE RAW MEMBERSHIP LIST.
0003                  WRITE(6,2099) TITLE
0004                  WRITE(6,2100) (MEMBR(K),K=1,NE)
0005                  WRITE(6,2200) (NUMBR(J),J=1,NC)
                C  INVERT THE "MEMBR" ARRAY AND PUT THE RESULT IN THE "LIST" ARRAY.
                C  FIRST REVISE THE "NUMBR" ARRAY TO CONTAIN START POINTS IN THE
                C  "LIST" ARRAY FOR EACH CLUSTER
0006                  NUMBR(NC)=NE-NUMBR(NC)+1
0007                  JJ=NC
0008                  JJ1=JJ-1
0009                  DO 10 J=2,NC
0010                  NUMBR(JJ1)=NUMBR(JJ)-NUMBR(JJ1)
0011                  JJ=JJ1
0012                  JJ1=JJ-1
0013             10   CONTINUE
                C  BUILD "LIST" ARRAY
0014                  DO 20 K=1,NE
0015                  MEMBRK=MEMBR(K)
0016                  NJ=NUMBR(MEMBRK)
0017                  LIST(NJ)=K
0018                  NUMBR(MEMBRK)=NUMBR(MEMBRK)+1
0019             20   CONTINUE
                C  SAVE THE SORTED MEMBERSHIP LIST IF DESIRED
0020                  IF (NTOUT .LT. 0) GO TO 30
0021                  WRITE (NTOUT,3099) TITLE
0022                  WRITE (NTOUT,3100) (LIST(K),K=1,NE)
                C  RESTORE THE "NUMBR" ARRAY.
0023             30   JJ=NC
0024                  DO 40 J=2,NC
0025                  NUMBR(JJ)=NUMBR(JJ)-NUMBR(JJ-1)
0026                  JJ=JJ-1
0027             40   CONTINUE
0028                  NUMBR(1)=NUMBR(1)-1
                C  PRINT RESULTS FOR EACH CLUSTER
0029                  WRITE(6,2000) TITLE
0030                  K1=1
0031                  DO 50 J=1,NC
0032                  WRITE(6,2300) J,NUMBR(J)
0033                  J1=(J-1)*NV
0034                  WRITE(6,2400) (CENTR(J1+I),I=1,NV)
0035                  K2=K1+NUMBR(J)-1
0036                  WRITE(6,2500) (LIST(K),K=K1,K2)
0037                  K1=K2+1
0038             50   CONTINUE
                C  THIS DUMPS OUT "TOTAL" ARRAY

FORTRAN IV G LEVEL  21                  RESULT              DATE = 79365        10/29/05

0039                  DO 60 I=1,NC
0040                  WRITE(6,4000) I
0041             4000 FORMAT('-TOTALS FOR CLUSTER',I3)
0042                  WRITE(6,4100)(TOTAL(I+J-1),J=1,NV)
0043             4100 FORMAT(1X,E10.5)
0044             60   CONTINUE
0045                  RETURN
0046             2000 FORMAT(1H1,20A4)
0047             2100 FORMAT('0RAW MEMBERSHIP LIST',/,(1X,25I5))
0048             2200 FORMAT('0CLUSTER SIZES',/,(1X,25I5))
0049             2300 FORMAT('0CLUSTER',I3,' CONTAINS',I5,' DATA UNITS')
0050             2400 FORMAT('0CENTROID COORDINATES',/,(1X,10E12.4))
0051             2500 FORMAT('0MEMBERSHIP LIST',/,(1X,25I5))
0052             3099 FORMAT(20A4)
0053             3100 FORMAT(16I5)
0054                  END
```

Figure A.8: Subroutine RESULT

```
UNTRAN IV G LEVEL  21                    RSULT2              DATE = 79365          10/31/13

0001                  SUBROUTINE RSULT2(CENTR,NUMBR,MEMBR,LIST,TITLE,NP,NV,NC,NTOUT,
                     * TOTAL,DATA)
                C  THIS SUBROUTINE PRINTS THE RESULTS FROM A CLUSTERING JOB BASED
                C  ON ANY VERSION OF SUBROUTINE "KMEAN".
                C
                      DIMENSION CENTR(1),NUMBR(1),MEMBR(1),LIST(1),TITLE(20),TOTAL(1),
                     * DATA(1)
                C
                C  AS A CONTINGENCY PRECAUTION WRITE OUT THE RAW MEMBERSHIP LIST.
0003                  WRITE(6,2000) TITLE
0004                  WRITE(6,2100) (MEMBR(K),K=1,NP)
0005                  WRITE(6,2200) (NUMBR(J),J=1,NC)
                C  INVERT THE "NUMBR" ARRAY AND PUT THE RESULT IN THE "LIST" ARRAY.
                C  FIRST REVISE THE "NUMBR" ARRAY TO CONTAIN START POINTS IN THE
                C  "LIST" ARRAY FOR EACH CLUSTER
0006                  NUMBR(NC)=NP-NUMBR(NC)+1
0007                  JJ=NC
0008                  JJ1=JJ-1
0009                  DO 10 J=2,NC
0010                  NUMBR(JJ1)=NUMBR(JJ)-NUMBR(JJ1)
0011                  JJ=JJ1
0012                  JJ1=JJ-1
0013             10   CONTINUE
                C  BUILD "LIST" ARRAY
0014                  DO 20 K=1,NP
0015                  MEMBRK=MEMBR(K)
0016                  NJ=NUMBR(MEMBRK)
0017                  LIST(NJ)=K
0018                  NUMBR(MEMBRK)=NUMBR(MEMBRK)+1
0019             20   CONTINUE
                C  SAVE THE SORTED MEMBERSHIP LIST IF DESIRED
0020                  IF (NTOUT .LE. 0) GO TO 30
0021                  WRITE (NTOUT,3000) TITLE
0022                  WRITE (NTOUT,3100) (LIST(K),K=1,NP)
0023                  DO 22 K=1,NP
0024             22   CALL USRJUT(DATA((LIST(K)-1)*NV+1))
                C  RESTORE THE "NUMBR" ARRAY.
0025             30   JJ=NC
0026                  DO 40 J=2,NC
0027                  NUMBR(JJ)=NUMBR(JJ)-NUMBR(JJ-1)
0028                  JJ=JJ-1
0029             40   CONTINUE
0030                  NUMBR(1)=NUMBR(1)-1
                C  PRINT RESULTS FOR EACH CLUSTER
0031                  WRITE(6,2000) TITLE
0032                  K1=1
0033                  DO 50 J=1,NC
0034                  WRITE(6,2300) J,NUMBR(J)
0035                  J1=(J-1)*NV
0036                  WRITE(6,2400) (CENTR(J1+I),I=1,NV)
0037                  K2=K1+NUMBR(J)-1
0038                  WRITE(6,2500) (LIST(K),K=K1,K2)
0039                  K1=K2+1

FORTRAN IV G LEVEL  21                   RSULT2              DATE = 79365          10/31/13

0040             50   CONTINUE
                C  THIS DUMPS OUT "TOTAL" ARRAY
0041                  DO 60 I=1,NC
0042                  WRITE(6,4000) I
0043             4000 FORMAT('-TOTALS FOR CLUSTER',I3)
0044                  WRITE(6,4100)(TOTAL((I-1)*NV+J),J=1,NV)
0045             4100 FORMAT((1X,E10.5))
0046             60   CONTINUE
0047                  RETURN
0048             2000 FORMAT(1H1,20A4)
0049             2100 FORMAT('0RAW MEMBERSHIP LIST',/,(1X,25I5))
0050             2200 FORMAT('0CLUSTER SIZES',/,(1X,25I5))
0051             2300 FORMAT('0CLUSTER',I3,' CONTAINS',I6,' DATA UNITS')
0052             2400 FORMAT('0CENTROID COORDINATES',/,(1X,10E12.4))
0053             2500 FORMAT('0MEMBERSHIP LIST',/,(1X,25I5))
0054             3000 FORMAT(20A4)
0055             3100 FORMAT(16I5)
0056                  END
```

Figure A.9: Subroutine RSULT2

```
FORTRAN IV G LEVEL  21                  MAIN            DATE = 79365        11/19/01

C THIS IS THE DRIVING PROGRAM.  FURTHER DOCUMENTATION IN ROSCOE FILE *FCON
C
0001          DIMENSION X(4000)
0002          LIMIT=4000
0003          CALL EXEC2(X,LIMIT)
0004          END FILE 2
0005          STOP
0006          END
```

```
FORTRAN IV G LEVEL  21                  DIST            DATE = 79365        11/19/01

0001          FUNCTION DIST(X,Y)
0002          DIMENSION X(1),Y(1)
0003          DIST=0.
0004          DO 10 I=1,2
0005       10 DIST=DIST+(X(I)-Y(I))**2
0006          RETURN
0007          END
```

```
FORTRAN IV G LEVEL  21                  USER            DATE = 79365        11/19/01

0001          SUBROUTINE USER(X)
0002          DIMENSION X(1)
0003          READ(4,100)(X(I),I=1,3)
0004      100 FORMAT(2F4.0,E17.10)
0005          RETURN
0006          END
```

```
FORTRAN IV G LEVEL  21                  USROUT          DATE = 79365        11/19/01

0001          SUBROUTINE USROUT(X)
0002          DIMENSION X(1)
0003          WRITE(2,10)(X(I),I=1,3)
0004       10 FORMAT(2F4.0,F17.10)
0005          RETURN
0006          END
```

Figure A.10: MAIN; USER; USEROUT; Function DIST

```
FORTRAN IV G LEVEL  21                    MAIN                    DATE = 80111        06/59/08
                C DRIVER. FUNCTION DOC. IN FILE 'FCOMMENT'
                C
0001                 DIMENSION X(24000)
0002                 LIMIT=24000
0003                 CALL EXEC3(X,LIMIT)
0004                 END FILE 6
0005                 STOP
0006                 END
```

```
FORTRAN IV G LEVEL  21                    DIST                    DATE = 80111        06/59/08
 0001                 FUNCTION DIST(X,Y)
 0002                 DIMENSION X(1),Y(1)
                C  DISTANCE IS EUCLIDIAN (X-Y)
 0003                 DIST=0.0
 0004                 DO 10 I=1,2
 0005            10   DIST=DIST+(X(I)-Y(I))**2
 0006                 RETURN
 0007                 END
```

```
FORTRAN IV G LEVEL  21                    DIST                    DATE = 80114        04/42/30
 0001                 FUNCTION DIST(X,Y)
 0002                 DIMENSION X(1),Y(1)
                C  THIS DISTANCE FUNCTION ONLY USES THE THIRD VARIABLE--INTENSITY
 0003                 DIST=ABS(X(3)-Y(3))
 0004                 RETURN
 0005                 END
```

Figure A.11: MAIN; Two Versions of DIST

```
FORTRAN IV G LEVEL  21                  RAY                 DATE = 79365        10/34/09

0001                SUBROUTINE RAY(IX,IY,YFL,SIGMA)
0002                IY=IX*65539
0003                IF (IY .GE. 0) GOTO 10
0004                IY=IY+2147483647+1
0005          10    YFL=IY
0006                YFL=YFL*0.4656613E-9
0007                YFL=SQRT(-2.0*SIGMA**2*ALOG(1-YFL))
0008                IX=IY
0009                RETURN
0010                END
```

Figure A.12: Subroutine RAY

```
FORTRAN IV G LEVEL  21                  MAIN                    DATE = 79365        10/56/16

0001                  DIMENSION VALUE(100,100)
0002                  SIGMA1=1.
0003                  SIGMA2=10000.
0004                  IX=123456789
0005                  DO 100 I=1,100
0006                  DO 100 J=1,100
0007                  CALL RAY(IX,IY,YFL,SIGMA1)
0008            100   VALUE(I,J)=YFL
0009                  DO 1 I=1,19
0010                  DO 1 J=1,100
0011                  CALL RAY(IX,IY,YFL,SIGMA2)
0012            1     VALUE(I,J)=VALUE(I,J)+YFL
0013                  DO 2 I=20,30
0014                  DO 2 J=1,39
0015                  CALL RAY(IX,IY,YFL,SIGMA2)
0016            2     VALUE(I,J)=VALUE(I,J)+YFL
0017                  DO 3 I=20,30
0018                  DO 3 J=51,100
0019                  CALL RAY(IX,IY,YFL,SIGMA2)
0020            3     VALUE(I,J)=VALUE(I,J)+YFL
0021                  DO 4 I=31,59
0022                  DO 4 J=1,100
0023                  CALL RAY(IX,IY,YFL,SIGMA2)
0024            4     VALUE(I,J)=VALUE(I,J)+YFL
0025                  DO 5 I=60,80
0026                  DO 5 J=1,59
0027                  CALL RAY(IX,IY,YFL,SIGMA2)
0028            5     VALUE(I,J)=VALUE(I,J)+YFL
0029                  DO 6 I=60,80
0030                  DO 6 J=71,100
0031                  CALL RAY(IX,IY,YFL,SIGMA2)
0032            6     VALUE(I,J)=VALUE(I,J)+YFL
0033                  DO 7 I=81,100
0034                  DO 7 J=1,100
0035                  CALL RAY(IX,IY,YFL,SIGMA2)
0036            7     VALUE(I,J)=VALUE(I,J)+YFL
0037                  DO 300 I=1,100
0038                  DO 300 J=1,100
0039            300   WRITE(1,500) I,J,VALUE(I,J)
0040            500    FORMAT(2I4,F17.10)
0041                  END FILE 1
0042                  WRITE(6,1000) IY
0043            1000  FORMAT(IX,I11)
0044                  END
```

Figure A.13: Creates Rayleigh Data, SIGMA=10,000.

```
FORTRAN IV G LEVEL  21                  MAIN                    DATE = 79365        11/03/21

0001                  DIMENSION VALUE(100,100)
0002                  SIGMA1=1.
0003                  SIGMA2=50.
0004                  IX=492247045
0005                  DO 100 I=1,100
0006                  DO 100 J=1,100
0007                  CALL RAY(IX,IY,YFL,SIGMA1)
0008            100   VALUE(I,J)=YFL
0009                  DO 1 I=1,19
0010                  DO 1 J=1,100
0011                  CALL RAY(IX,IY,YFL,SIGMA2)
0012            1     VALUE(I,J)=VALUE(I,J)+YFL
0013                  DO 2 I=20,30
0014                  DO 2 J=1,39
0015                  CALL RAY(IX,IY,YFL,SIGMA2)
0016            2     VALUE(I,J)=VALUE(I,J)+YFL
0017                  DO 3 I=20,30
0018                  DO 3 J=51,100
0019                  CALL RAY(IX,IY,YFL,SIGMA2)
0020            3     VALUE(I,J)=VALUE(I,J)+YFL
0021                  DO 4 I=31,59
0022                  DO 4 J=1,100
0023                  CALL RAY(IX,IY,YFL,SIGMA2)
0024            4     VALUE(I,J)=VALUE(I,J)+YFL
0025                  DO 5 I=60,80
0026                  DO 5 J=1,59
0027                  CALL RAY(IX,IY,YFL,SIGMA2)
0028            5     VALUE(I,J)=VALUE(I,J)+YFL
0029                  DO 6 I=60,80
0030                  DO 6 J=91,100
0031                  CALL RAY(IX,IY,YFL,SIGMA2)
0032            6     VALUE(I,J)=VALUE(I,J)+YFL
0033                  DO 7 I=81,100
0034                  DO 7 J=1,100
0035                  CALL RAY(IX,IY,YFL,SIGMA2)
0036            7     VALUE(I,J)=VALUE(I,J)+YFL
0037                  DO 300 I=1,100
0038                  DO 300 J=1,100
0039            300     WRITE(1,500) I,J,VALUE(I,J)
0040            500     FORMAT(2I4,E17.8)
0041                  END FILE 1
0042                  WRITE(6,1000) IY
0043            1000  FORMAT(1X,I11)
0044                  END
```

Figure A.14: Creates Rayleigh Data, SIGMA=50.

```
FORTRAN IV G LEVEL  21                 PLOT                DATE = 79365           10/42/11

 0001                SUBROUTINE PLOT(IROW,JCOL)
 0002                LOGICAL*1 BOX(255,129),CHARS(50),DOT
 0003                DATA CHARS/'1','2','3','4','5','6','7','8','9','0','A','B',
                    * 'C','D','E','F','G','H','I','J','K','L','M','N','O',
                    * 'P','Q','R','S','T','U','V','W','X','Y','Z','+','<',
                    * '>','*','?','=','&','@','%','$','#','-','[','='/
 0004                DATA DOT/'.'/
 0005                DO 1 I=1,IROW
 0006                DO 1 J=1,JCOL
 0007        1       BOX(I,J)=DOT
 0008                DO 10 I=1,9999
 0009                READ (5,20,END=15)KOUNT
 0010        20      FORMAT(I5)
 0011                DO 30 NUMBER=1,KOUNT
 0012                READ(1,40,END=16) DATA1,DATA2
 0013        40      FORMAT(2F5.0)
 0014                INT1=INT(DATA1)
 0015                INT2=INT(DATA2)
 0016                BOX(INT1,INT2)=CHARS(I)
 0017        30      CONTINUE
 0018        10      CONTINUE
 0019        16      WRITE(6,55)
 0020        55      FORMAT(' *** ERROR ***  END OF DATA SET ON UNIT 01.')
 0021        15      DO 100 I=1,IROW
 0022        100     WRITE(10,200)(BOX(I,J),J=1,JCOL)
 0023        200     FORMAT(1X,129A1)
 0024                STOP
 0025                END
```

Figure A.15: Cluster Plot Program

```
FORTRAN IV G LEVEL  21                  MAIN                    DATE = 80318          00/35/46

            C  GRAY SCALE PROGRAM...
0001            IMPLICIT INTEGER*2 (I-N)
0002            REAL*4 XLINE(128),THRESH(8)
0003            LOGICAL*1 SCALE(8),PLINE(128)
            C READ IN THE GRAY SCALE SYMBOLS...
0004            READ(5,5)(SCALE(I),I=1,8)
0005        5   FORMAT(8A1)
            C
            C
            C  READ THE LOWEST VALUE FOR EACH GRAY LEVEL.
0006            DO 10 I=2,8
0007            READ(5,100) THRESH(I)
0008            II=I-1
0009            WRITE(6,131) SCALE(II),THRESH(I)
0010       10   CONTINUE
0011            WRITE(6,102) SCALE(1),THRESH(8)
            C  READ MAX HEIGHT AND VERT. AXIS VALUES. (<= 128; <= 256)
0012            READ (5,200) MAXV,MAXH
0013            WRITE(6,201) MAXV,MAXH
0014            DO 20 I=1,MAXV
0015            DO 30 J=1,MAXH
0016            READ (1,300) XLINE(J)
0017            GRAY=1
0018            DO 40 K=2,8
0019            IF (XLINE(J) .LE. THRESH(K)) GO TO 40
0020            GRAY=K
0021       40   CONTINUE
0022            PLINE(J)=SCALE(GRAY)
0023       30   CONTINUE
0024            WRITE(6,400)(PLINE(J),J=1,MAXH)
0025       20   CONTINUE
0026            STOP
0027       130  FORMAT(F15.7)
0028       101  FORMAT(' SYMBOL ',A1,' <',F15.7)
0029       102  FORMAT(' SYMBOL ',A1,' >',F15.7)
0030       201  FORMAT(' MAXV=',I4,' MAXH=',I4,/,'1')
0031       200  FORMAT(2I4)
0032       400  FORMAT(' ',128A1)
0033       300  FORMAT(8X,E12.7)
0034            END
```

FIGURE A.16: Grey Scale Program

REFERENCES

1. R.C. Jancey, "Multidimensional Group Analysis," Aust J Bot, vol 14, no 1, April 1966, pp 127-130.

2. C.T. Zahn, "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters," IEEE Trans on Computers, vol C-20, no 1, January 1971, pp 68-86.

3. Statement of Work for "Topographic Map/Backscatter Correlation," RFP Solicitation Number F30602-79-R-0205, RADC, Griffiss Air Force Base, New York, June 1979, pp 3,4.

4. F.E. Nathanson, Radar Design Principles. New York: McGraw-Hill Book Co, 1969.

5. D.K. Barton, Radars, vol 5: Radar Clutter. Dedham, Mass: Artech House, Inc, 1975.

6. R.M.Cormack, "A Review of Classification," J Royal Stat Soc, series A, vol 134, part 3, 1971, pp 321-367.

7. Documentation for ARTHUR, distributed by A.M. Harper, Laboratory for Chemometrics, Univ of Georgia, Athens, Ga, 30602.

8. B.A. Black and M. Arozullah, "Modeling of Shadowing in Radar Returns," Technical Report prepared under Postdoctoral Program Contract F30602-78-C-0102, Task 5, A-8-1142, RADC, Griffiss Air Force Base, New York, November 1978.

9. W.A. Fordon, "Statistical Pattern Recognition Techniques as Applied to Radar Returns," Technical Report prepared under Postdoctoral Program Contract F30602-78-C-0102, RADC, Griffiss Air Force Base, New York, 1978.

10. E.W. Forgy, "Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classifications," Biometrics Abstracts, vol 21, no 3, Sept 1965, pp 768-769.