AD-A076 119 KANSAS UNIV/CENTER FOR RESEARCH INC LAWRENCE REMOTE --ETC F/G 17/9
RADAR IMAGE SIMULATION OF SEASONALLY DEPENDENT REFERENCE SCENES--ETC(U
APR 79 J C HOLTZMAN , J E BARE , V H KAUPP DAAK70-78-C-0062
UNCLASSIFIED RSL-TR-370-2 ETL-0188 NL

1 OF 3
AD A076119






































































AD A076119

LEVEL II

12

ETL-0188

RADAR IMAGE SIMULATION OF SEASONALLY DEPENDENT REFERENCE SCENES

Principal Investigator J. C. Holtzman

RSL Technical Report 370-2

April 1979

Approved for public release; distribution unlimited

Prepared for:

U. S. Army Engineer Topographic Laboratories
Fort Belvoir, Virginia 22060

Contract DAAK-70-78-C-0062

DDC FILE COPY

DDC
RECEIVED
NOV 5 1979
A

THE UNIVERSITY OF KANSAS CENTER FOR RESEARCH, INC.
2291 Irving Hill Drive—Campus West
Lawrence, Kansas 66045

79 11 05 029

THE UNIVERSITY OF KANSAS CENTER FOR RESEARCH, INC.

2291 Irving Hill Drive—Campus West
Lawrence, Kansas 66045

(10) J. C. /Holtzman, J. E. /Bare,
V. H. /Kaupp, E. E. /Komp
J. A. /Stiles

(18) (19) ETL-0188

(6) RADAR IMAGE SIMULATION OF SEASONALLY DEPENDENT REFERENCE SCENES,

(9) Contract rept. 27 Mar-27 Sep 78

Principal Investigator J.C. Holtzman

(14) RSL-TR-

RSL Technical Report 370-2

(11) April 1979

(12) 238

Approved for public release; distribution unlimited

Prepared for:

U.S. ARMY ENGINEER TOPOGRAPHIC LABORATORIES
Fort Belvoir, Virginia 22060

(15)

CONTRACT DAAK-70-78-C-0062

406688

DESTROY THIS REPORT WHEN NO LONGER NEEDED.
DO NOT RETURN IT TO THE ORIGINATOR.

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

THE CITATION IN THIS REPORT OF TRADE NAMES OF COMMERICALLY AVAILABLE PRODUCTS DOES NOT CONSTITUTE OFFICIAL ENDORSEMENT OR APPROVAL OF THE USE OF SUCH PRODUCTS.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ETL-0188	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: RADAR IMAGE SIMULATION OF SEASONALLY DEPENDENT REFERENCE SCENES	5. TYPE OF REPORT & PERIOD COVERED: March 27, 1978- Sept. 27,1978 Contract Report
6. PERFORMING ORG. REPORT NUMBER: RSL Technical Report 370-2	
7. AUTHOR(s): J.C. Holtzman, V.H. Kaupp, J.A. Stiles, J.E. Bare, E.E. Komp, V.S. Frost	8. CONTRACT OR GRANT NUMBER(s): DAAK-70-78-C-0062
9. PERFORMING ORGANIZATION NAME AND ADDRESS: U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia 22060	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: April 1979
	13. NUMBER OF PAGES: 227
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

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

18. SUPPLEMENTARY NOTES

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

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

The results are reported from applying radar image simulation to produce simulated reference scenes of winter conditions for a missile guidance usage. A data base was constructed of the Watertown, New York, test site and simulated radar images were generated. The data base was prepared from historical data for an "average" winter at the test site. Simulated radar images were produced via the point scattering model and an empirical model was used for predicting the electromagnetic reflectance from the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20.

ground, its cover, and overlying snow. Copies of the simulated radar images are included.

The results reported were obtained for four (4) simulations corresponding to four specific altitudes in the terminal phases of the trajectory of a guided missile, each successively lower. The simulated images have been produced for testing against actual radar data of the same site via the Correlatron*. These tests have not been performed as the actual radar data have not yet been obtained.

---

*Correlatron is a two-dimensional cross-correlation measuring device manufactured by Goodyear Corporation and installed at ETL.

Accession For
NTIS GRA&I
DDC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or special

SUMMARY

The work reported in this document was performed under Contract DAAK 70-78-C-0062 with the Geographic Sciences Laboratory of the U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia. The contract monitor was Mr. Craig Baker. The U.S.A.E.T.L. has been instrumental in the support of research and applications of radar image simulation, radar backscatter measurements and theory. Additionally, interaction between E.T.L. (by way of Messrs. Bernie Schepps and Richard Hevenor) and the Remote Sensing Laboratory led to development of models and methodologies for high quality radar simulations, and more recently to a workshop on terrain and sea backscatter (also supported by the Army Research Office and the Naval Research Laboratory).

The purpose of the work reported herein was to produce radar reference scenes for a terminal guidance application. The target site selected for this work was the geographic region centered on the southwestern corner of the Freeman Bus Corporation garage in north-eastern Watertown, New York. The target site is the area ($\sim$ 4100 square kilometers) encompassed by a circle having an approximate radius of 36 km extending from the building corner (the center is located at 43° 58' 43.409" N by 75° 52' 31.043" W). The Watertown target site contains Lake Ontario in the northwestern region, Watertown in the central portion, and is characterized by a proliferation of small agricultural fields which are mostly fallow in the winter, and by forested regions. The site represents a complex mix of area extensive, distributed targets, and specular reflecting cultural targets.

Two bases were constructed for this target site. One was constructed for the high altitude, bands 3 and 4, reference scenes, and one for the low altitude, bands 1 and 2, reference scenes. The data bases were constructed to

represent the "average" winter conditions during the month of February as determined from historical weather records. Construction of these data bases is reported in Section 2.

After construction of the data bases was complete, they were digitized via human operators controlling a large table digitizer (the one located at ETL was used). The result of this task was computer-compatible tapes (CCT's) containing digital representations of all the boundaries of each data base arrayed in serial format. These data were converted into a completely-specified, two-dimensional array which became one of the two data bases. Each of these arrays contained all the information necessary for producing radar reference scenes. The information provided in each data base included the position and elevation of each point in a reference coordinate system, the microwave reflectance category of each point, and the snow depth of each point. Production of these digital arrays from the data base maps is related in Section 3.

For producing winter scene simulation, a model was required for the microwave reflectance properties of snow overlaying a ground category (i.e., snow overlaying bare ground). An empirical model was used. The model used is described in Section 4.

Two sequences of four (4) radar reference scenes were produced from the two (2) data bases; one (1) for winter conditions with snow present, and one (1) for winter conditions without snow. These results are presented in Section 5.

Conclusions and recommendations are presented in Section 6.

TABLE OF CONTENTS

PAGE

LIST OF FIGURES v

LIST OF TABLES vi

1.0 INTRODUCTION 1

1.1 Motivation and Purpose 1

1.2 Test Description 2

1.3 Synopsis of the Point Scattering Model 3

1.4 Theoretical Development of the PSM 3

1.5 PSM Simulation Implementation Philosophy 9

1.5.1 Simulation Parameters 10

1.5.2 Data Base 14

1.5.3 Reflectivity Data 19

1.6 Synopsis of Computer Programs 22

1.6.1 Introduction 22

1.6.2 PPI Software Realization of PSM 24

2.0 PRODUCTION OF SURFACE FEATURE OVERLAYS FOR USE IN THE SIMULATION OF WINTERSCENE RADAR IMAGERY OF THE WATERTOWN, NEW YORK, AREA . . 29

2.1 Preparing an Aseasonal Scene Data Base 33

2.2 Superimposing Winter Characteristics on the Aseasonal Scene 37

2.2.1 Isometric Addition of Snow to the Ground Throughout the Scene 40

2.2.2 Varying Snow Depth Surface Category Type 41

2.2.3 Varying Snow Depth with Exposure to Wind 42

2.2.4 Varying Snow Depth with Surface Type and Exposure to Wind 45

2.3 Watertown Data Base Specifications 45

3.0 CONSTRUCTION OF A DIGITAL DATA BASE FROM SURFACE FEATURE OVERLAYS 48

3.1 Definition of Work Performed 48

3.2 Construction Approach 48

3.3 Digitizing Surface Feature Overlays 50

3.4 Construction of a Digital Data Base Matrix 52

3.4.1 Introduction to the Approach 52

3.4.2 Summary of Computer Programs 54

3.5 Merging Different Category Matrices 54

PAGE

4.0 RADAR SIMULATION: BACKSCATTER DATA AND AN EMPIRICAL MODEL FOR SNOW 56

4.1 Backscatter Data for Winter Simulations 59

5.0 RESULTS 62

6.0 CONCLUSIONS AND RECOMMENDATIONS 69

6.1 Conclusions 69

6.2 Recommendations 71

6.2.1 Develop an Interactive Feature Extraction System . . 71

6.2.2 Develop Theoretical Scattering Models 73

6.2.3 Compile a Comprehensive Listing of Snow 74

6.2.4 Conduct an Empirical Backscatter Program 74

6.2.5 Perform a Sensitivity Analysis 74

APPENDIX A: Radar Reflectivity Data 76

APPENDIX B: Description of Computer Routines for Data Base Construction and Radar Image Simulation 102

Initfix 125

Areafix 143

Count 158

Sort 162

Build 167

Category and Cultural Merge 172

APPENDIX C: Specialization of the PSM PPI Implementation to the Radar System 182

Polar Create 197

Polar Array 203

Array Fix 207

Power 211

Graytone 217

Rectangular Create 222

Rectangular Array 224

LIST OF FIGURES

PAGE

Figure 1 PSM Simulation Implementation Philosophy 6

Figure 2 PPI Geometry 25

Figure 3 Flow of Data for Simulation of PPI Radar Data 27

Figure 4 Radar Image Simulations of Watertown, New York, in late fall to early winter 64

Figure 5 Radar Image Simulations of Watertown, New York in mid-winter 65

LIST OF TABLES

		PAGE
Table 1	Simulation Parameters	11
Table 2	Time Expenditures for Production of All Watertown Aseasonal and Winter Overlays	32
Table 3	Cover-Type Categories for 1 : 100,000 Scale Overlays (Watertown Site)	35
Table 4	Cover-Type Categories for 1 : 24,000 Scale Overlays (Watertown Site)	36
Table 5	Watertown Winter Scene Temperature and Snow Depth	39
Table 6	Winter Overlay Codes for Varying Snow Depth with Surface Cover Type Category	43
Table 7	Overlay Codes for Varying Snow Depth with Exposure to Wind	44
Table 8	Parameter Specifications 1 : 24,000 Scale Data Base (Watertown Site)	46
Table 9	Parameter Specifications 1 : 100,000 Scale Data Base (Watertown Site)	47
Table 10	Guidance Radar Parameters as Modeled for Simulation	63
Table 11	Simulation Characteristics	63

1.0 INTRODUCTION

1.1 Motivation and Purpose

Studies performed by Cosgriff et al.[1], Bush et al.[2], and Stiles et al.[3], have shown that the radar return amplitude from terrain can be affected dramatically by seasonal and meteorological changes. The radar reflectivity can be altered by many decibels for changes at a site such as snow and ice cover, ground moisture, and foliation or defoliation of trees. Therefore, it is imperative to study the effects of these changes on the radar image when it is being used as an "aseasonal" reference scene for guidance.

A secondary objective of the study was to determine the effect of seasonal variations in a target site on the guidance system. However, the test data to support this objective has not yet been collected by the U.S. Army.

The theoretical model and its implementation as a set of computer programs for radar image simulation was reported earlier.[4] More recently, high quality SLAR (side looking airborne radar) simulations were reported.[5]

---

[1]Cosgriff, R.L., W.H. Peake, and R.C. Taylor, "Terrain Scattering Properties for Sensor System Design," Engineering Experiment Station Bulletin, 181, Vol. 29, Ohio State University, Columbus, Ohio, May 1960.

[2]Bush, T., F. Ulaby, T. Metzler, and H. Stiles, "Seasonal Variations of the Microwave Scattering Properties of Deciduous Trees as Measured in the 1-18 GHz Spectral Range," Remote Sensing Laboratory Technical Report, RSL TR 177-60, University of Kansas Center for Research, Inc., Lawrence, Kansas, June 1976.

[3]Stiles, H., F. Ulaby, B. Hanson and L. Dellwig, "Snow Backscatter in the 1-8 GHz Region," Remote Sensing Laboratory Technical Report, RSL TR 177-61, University of Kansas Center for Research, Inc., Lawrence, Kansas June 1976.

[4]Martin, R.L., "SLAR Simulation and Applications," Master's Thesis, University of Kansas, September 1976.

[5]Holtzman, J.C., V.H. Kaupp, J.L. Abbott, V.S. Frost, E.E. Komp, and E.C. Davison, "Radar Image Simulation: Validation of the Point Scattering Model," Engineer Topographic Laboratories, Unites States Army, Fort Belvoir, Virginia, ETL-0117, June 1977.

1.2 Test Description

A set of four (4) radar scenes (called reference scenes) was produced of the Watertown test target site for testing against recorded radar data via a Correlatron*[6] test configuration.

The four reference scenes were produced via the PSM radar simulation model from a data base constructed of the Watertown test site. The data base was developed to support production of reference scenes of winter conditions at the target site. The approach taken was to model the target site in two separate stages. The first stage was the aseasonal one in which the characteristics of the ground and its cover which are invariable across seasons were modeled. The second stage was the seasonal one in which seasonal effects were modeled. The season specified for this work was winter. In particular, winter conditions for February at the target site as determined from historical perspectives were modeled.

Four (4) reference scenes were generated from this Watertown/winter data base. The four (4) simulations represent the reference scenes for different altitudes over the target, each successively lower in the terminal trajectory of a ballistic missile. The four (4) reference scenes were produced and stored on magnetic tapes which were shipped to ETL (Engineer Topographic Laboratories) for testing.

The plan was for the four (4) reference scenes to be tested against actual data via the Correlatron installed in a test configuration at ETL. The actual data were to have been obtained by the Army in a flight test program scheduled

---

*Correlatron is the name of a device manufactured by Goodyear Aerospace Corporation and which measured the two-dimensional cross-correlation between a "live" and a stored reference signal.

[6] Klass, P.J., "Guidance Device Set for Pershing Tests," Aviation Week and Space Technology, 12 May 1975.

for February 1978, or February 1979. Unfortunately, the flight test program was delayed and the subsequent data were unsatisfactory. Thus, no comparison data exist. The data obtained which are closest in time to the scheduled data are some acquired in late April or early May 1978. Conditions at the target site changed dramatically from February with its snow cover to May with its water cover. Thus, the most that can be accomplished with these reference scenes is a visual, subjective comparison. Flights are planned for the 1979-1980 winter; the results of these tests will be compared with simulated results if at all possible.

1.3 Synopsis of the Point Scattering Model

The PSM has been developed as a general approach to the problem of simulating the data from a radar. As operational radars are developed in different configurations, so has the PSM been developed to simulate the end-to-end response of different radar configurations.

The PSM is a general simulation model which is applicable to radars in both the PPI (Plan-Position Indicator) as well as the SLAR (Side-Looking Airborne Radar) configuration. One usage of the PSM which has been developed represents specialization of the PPI implementation for an application using simulated radar data for guidance of a ballistic missile in the terminal phase of descent. This application was for a missile which employed a guidance system incorporating a Correlatron.[6]

The Point Scattering Model (PSM) for simulating radar data has been developed and implemented on a digital computer. By simulating radar data is meant

---

[6]Klass, P.J., "Guidance Device Set for Pershing Tests," Aviation Week and Space Technology, 12 May 1975.

synthesis, via digital computer, of the data which would have been recorded by a radar flying the same ground track over the prescribed terrain sites. The radar senses the reradiated flux from the ground and its cover in the microwave portion of the electromagnetic spectrum and stores the results. The radar produces an output proportional to the reflectivity characteristics at a fixed wavelength of the terrain and when recorded in an image, displays the terrain in fine detail and with spectacular relief. So does the simulated radar image produced via the PSM. The PSM is not limited to simulating images, it is viable with suitable alterations for most recording formats.

The PSM represents the radar, the ground and its microwave response, and the stored data by a closed-system model. The model rests on firm, theoretical foundations and is mathematically rigorous. It incorporates all aspects of the radar problem starting with the transmitter, including such aspects as the antenna during transmission, the propagation path to the ground, the ground response, the return data, the antenna during reception, and the receiver and concluding with data storage and presentation. The PSM has been tested and validated for a specific class of radar targets, distributed targets.*

Just as all mathematical models are abstractions of reality, so is the PSM. It attempts to describe in closed form the processes of the system consisting of radar, ground, and data storage. The PSM is completely general and capable of synthesizing data having a desired accuracy if the cost is paid in time, complexity, and resources expended. But the true value of the PSM arises from the ease with which a specific implementation can be tailored or specialized,

---

*As used throughout, distributed targets are areas of natural terrain or ground cover such as wheat or corn fields, grassy expanses, forests, plowed ground, paved areas, etc., which are approximately homogeneous and large relative to a radar resolution element.

to take advantage of simplifications and approximations valid for a specific application, thereby making it efficient and cost-effective to use.

An approach has been developed regarding simulation that insures all the requisite data and information needed to simulate the response of a specific radar from a desired ground site (target) are obtained. This approach is illustrated conceptually in Figure 1 as the PSM simulation implementation philosophy.

The PSM implementation philosophy is the framework which identifies the various simulation requirements and relates these to one another. As can be seen from Figure 1, three (3) basic kinds of data and information interact with one another and serve as inputs to the simulation computer programs. These are: (1) data base, (2) reflectivity data, and (3) simulation parameters. Upon complete specification of these, the simulation computer programs can be run and radar data can be simulated for the system being modeled as if it, the radar, were flown over the data base (i.e., the ground site).

As the PSM has been implemented on a digital computer, its required input data must be in digital form. The first of these input data is a data base, a digital replica of the ground in the target area. This digital representation of the ground, called data base hereafter, models the ground and its cover in the desired site and contains a facsimile of both the dielectric categories present, called backscatter categories, as well as the geometric variations, called elevation data. The data base is, thus, a sampled replica of the backscatter categories present in a target simulation scene (both distributed and cultural* targets, or categories) and the elevation surface.

---

*As used here, cultural targets are targets, such as buildings, vehicles, etc., which are smooth relative to radar wavelengths and whose backscatter properties are essentially specular.

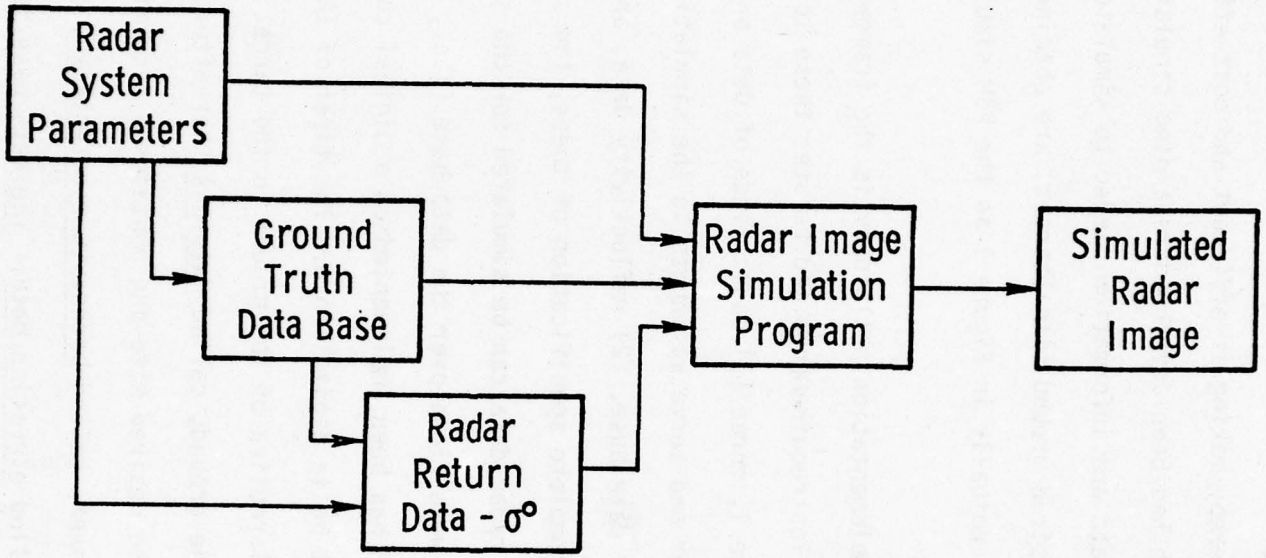


Figure 1. PSM Simulation Implementation Philosophy.

A data base typically consists of a digital matrix having at least four (4) dimensions: two (2) for the spatial location of each point, one (1) for its elevation, and at least one (1) for its backscatter category. The resolution built into a data base is determined by the sampling frequency and is highly dependent upon such factors as the resolution of the radar system being simulated, the application for which a particular data base is required, and available time and resources.

Within a target site numerous different types of vegetation and terrain cover exist. Some of these are distributed targets such as forests, wheat fields, grasslands, highways, runways, water bodies, etc., and some are cultural targets such as buildings and vehicles. Each different type of target is modeled as a different backscatter category in a data base. Backscatter category is, thus, the smallest spatial unit, or 'field' into which regions having homogeneous dielectric properties can be divided commensurate with the data base resolution, radar system, and application.

Also within a target site, the surface of the ground varies in relative height from point to point above a datum surface. This height variation from point to point creates unique patterns in radar data which must be modeled for simulation purposes. Elevation data are, as the name suggests, the value of the relative height of each point in a data base. The resolution of elevation data is described by the sampling frequency of the data base and the accuracy of the data from which the samples are obtained.

A ground response model is the second type of input data required by the PSM. The ground response model normally utilized in the PSM to reflect the properties of each of the different types of ground cover identified in the target site and included in the data base is backscatter, the differential scattering cross-section (σ^o). The PSM uses σ^o to simulate the interaction

between the ground and the electromagnetic energy incident from the radar, and to predict the percentage of reradiation back to the radar. For each different ground cover type specified in a data base (called backscatter category), the PSM requires a set of σ^{o} data, either experimental or theoretical, to be provided. The backscatter data used to make the sample results presented in Section 5 are discussed in Section 4.

The PSM requires input of system parameters, storage parameters, and ground track information. These data are required to modify the computational algorithms of the PSM so they reflect the desired system and storage parameters and, in addition, are required so that the computational algorithms can be simplified to increase efficiency and cost effectiveness. They are also used to specify key parameters pertinent to this data base and its construction, such as resolution, orientation, etc. Finally, these data are used to specify the flight-line parameters which allow determination of altitude, angles of incidence, and thus the geometrical aspects of the simulation problem.

Upon satisfaction of all input data requirements shown in Figure 1, the computational algorithms of the PSM are invoked to simulate the response of a desired radar flying a particular flight line over a specified portion of ground. The geometrical relationships between the location and orientation of the simulated radar and its platform and each point in the data base are solved with such effects as foreshortening, layover, shadow, look-direction, ect., being treated. Upon solution of this geometry, the PSM predicts the amount of power reradiated from each point in the data base back to the receiving antenna by finding the backscatter category of each point and solving the functional form of the appropriate backscatter data for the conditions at each point. The resulting calculations are further refined to account for reception, detection, and storage of the power from each point, and the data output from the PSM are stored on computer-compatible digital magnetic tapes.

1.4 Theoretical Development of the PSM

The theoretical development of the PSM has been reported previously[5,6,7] and, thus, it will not be repeated here.

1.5 PSM Simulation Implementation Philosophy

A philosophy has been developed regarding simulation that insures all the requisite data and information needed to simulate the response of a specific radar from a desired ground site (target) are obtained. This philosophy is illustrated conceptually in a previous figure (see Figure 1) as the PSM simulation implementation philosophy.

As can be seen from Figure 1, three (3) basic kinds of data and information interact and serve as inputs to the simulation computer programs. These are, starting with the one which should be specified first: (1) simulation parameters; (2) data base; and (3) reflectivity data. Important ramifications of each of these areas, interactions between them, and additional discussions explaining how some of the details can be solved are

---

[5]Holtzman, J.C., V.H. Kaupp, J.L. Abbott, V.S. Frost, E.E. Komp, and E.C. Davison, "Radar Image Simulation: Validation of the Point Scattering Model," Engineer Topographic Laboratories, United States Army, Fort Belvoir, Virginia, ETL-0117, June 1977.

[6]Holtzman, J.C., V.H. Kaupp, J.L. Abbott, V.S. Frost, E.E. Komp, and E.C. Davison, "Radar Image Simulation: Validation of the Point Scattering Method," Volume II, ETL-0118, Remote Sensing Laboratory Technical Report, RSL TR 319-28, University of Kansas Center for Research, Inc., Lawrence, Kansas, September 1977.

[7]Holtzman, J.C., J.L. Abbott, V.H. Kaupp, E.E. Komp, E.C. Davison, and V.S. Frost, "Radar Image Simulation: Validation of the Point Scattering Method," Addendum, ETL-0155, Remote Sensing Laboratory Technical Report, RSL TR 319-31, University of Kansas Center for Research, Inc., Lawrence, Kansas, June 1978.

presented in succeeding sections. In Section 1.5.1 simulation parameters are discussed. In Section 1.5.2 simulation data base and the underlying philosophy are related. In Section 1.5.3 reflectivity data for simulation are presented.

After all three kinds of input data are specified, the simulation computer programs can be invoked and radar data can be simulated for the system being modeled. The computer programs solve the geometrical relationships between the position of the simulated radar and the ground (i.e., each point in the data base) for determining such parameters as resolution and local angle of incidence. The computer programs are structured so as to model such propagation effects as layover, shadow, etc. Software is discussed in Section 1.6.

1.5.1 Simulation Parameters

Simulation parameters are all those parameters of the imaging process required for specializing the PSM for one radar, desired ground sites, and pertinent flight paths over those sites. Table 1 is a sample listing only, and is provided for reference to suggest what is meant by "simulation parameters."

As can be seen by reference to the table, simulation parameters are listed according to three (3) headings. The first of these, "Radar Systems Parameters," lists the various radar parameters which must be obtained so that the data base, σ^0 data, and computer programs can be specialized for a particular radar system. The second listing of simulation parameters, "Flight Path Parameters," is required so that the desired data base can be constructed, so that σ^0 data for seasonally varying ground cover types can be obtained, and so that simulated data can be produced having the correct

TABLE 1: SIMULATION PARAMETERS

SYMBOL	DESCRIPTION	POTENTIAL IMPACT
Radar System Parameters		
λ	Wavelength	1,2,3
-	Polarization	1,2
G	Antenna pattern factor in range direction	1,3
β	Antenna azimuth beamwidth	1,3
τ	Pulse length	1,3
-	Scan format (i.e., PPI or SLAR)	1,3
M	Receiver transfer function	3
Q	Analog-to-digital converter transfer function	3
S	Data processing effects	3
γ	Slope of the linear portion of film curve or density versus logarithm of exposure	3
K	Intercept point of line having slope, γ	3
N	Number of "independent samples"	3
L	Real antenna length	1,3
B_C	Total system bandwidth	3
B	Resolution bandwidth	3
θ_N	Near-range edge of swath angle-of-incidence	3
θ_F	Far-range edge of swath angle-of-incidence	3
Flight Path Parameters		
h	Altitude above a mean surface	1,3
-	Latitude and longitude of target area	1

1 Data base
2 Reflectivity data
3 Computer programs

TABLE 1: SIMULATION PARAMETERS (continued)

SYMBOL	DESCRIPTION	POTENTIAL IMPACT
-	Direction of flight	1,3
-	Season and meteorological conditions	1,2,3
$L_T L_R$	Atmospheric losses	3
Application and Simulation Parameters		
-	Intended use of simulations for determining degree of accuracy required in specification of various parameters and transfer functions	1,2,3
n	Number of bits in the output computer word	3
m	Signal dynamic range which is to be in the final simulated radar data	3
I_{MIN}, I_{QC}, I_C	Intensity calibration data	3
P_{RC}, G_{RC}	Density calibration data	3

1 Data base
2 Reflectivity data
3 Computer programs

orientation, scale and radar effects such as layover and shadow. The third listing, "Application and Simulation Parameters," is required for maximum specialization of the PSM to the application.

From Figure 1 and Table 1, it can be seen that the simulation parameters interact with the simulation phases labeled data base, σ^o data, and simulation computer programs. These data are normally specified first, and the other phases of simulation follow.

Upon specification of these parameters, construction can begin on the data base if a new one is required. The location on the Earth, orientation, and size of the data base will have been specified. The resolution for which the data base will be constructed will have been determined. Ground cover differentiation criteria will have been devised (i.e., criteria defining how to differentiate between important and unimportant ground cover types will have been developed which, for example, will allow constructing a data base having ground cover type boundaries appropriate, e.g., for a 16 GHz, HH polarization, 50 meter radar). Thus, important conditions for the data base will have been established and work can begin for constructing it.

Specification of these simulation parameters is necessary, also, for the σ^o data, the second phase of interaction shown in Figure 1. The specification of frequency, polarization, angular range, season, and meteorological conditions together with the ground cover type identified in the data base label which σ^o data are needed.

The PSM simulation computer programs can be specialized and modified upon specification of the simulation parameters. Simplifying approximations can be made for various aspects of the model such as for the receiver function, where appropriate. Appropriate transfer functions, the antenna

pattern, system parameters, etc., can be written into the simulation computer programs. Data handling simplifications can be designed either to reduce the cost of simulation, or to make simulation feasible, as from very large data bases.

The uses to which the simulation parameters are put have just been sketched out. This has not been an exhaustive discussion of how they interact for the details of interaction are system and application special. The intent here was to suggest what kinds of information are needed, how this information interacts with various phases of simulation, and how simplifications can result from prudent use of the information.

1.5.2 Data Base

A data base is a digital replica of the ground, modeling its topography and cover. A specific data base will contain a symbolic representation of the dielectric categories present as different ground cover types, or backscatter categories, as well as the elevation surface of a specific site. The data base is, thus, a sampled replica of the backscatter categories present in a target simulation scene and the elevation surface. Development of a data base for radar simulation of winter scenes is summarized in Section 2.

A data base typically consists of a digital matrix having at least four (4) dimensions: two (2) for the spatial location of each point, one (1) for its elevation, and at least one (1) for its microwave reflectivity category. More than four (4) dimensions will be required for a data base when seasonal and meteorological variations are to be simulated. The finest resolution which can be built into a data base is determined by the ground spot size each matrix element represents and is highly dependent upon such factors as the resolution of the radar system being simulated, the

application for which a particular data base is required, and available time and resources.

Accurate construction of a data base is crucial to the overall simulation effort. The final, simulated radar data can be no better than the data base and, frequently, it is a degraded form of it. At one extreme there is a one-to-one mapping of data base elements into radar resolution cells and on the other extreme is a many-to-one mapping. Most cases of radar simulation fall between these extremes with, perhaps, four (4) to twenty-five (25) data base elements mapped into a single radar resolution cell.

Regardless of how many data base elements map into a resolution cell, the crucial element is the inherent accuracy of the data base; the accuracy built into the data base. This question of accuracy extends both to modeling the spatial distribution of ground cover types, distributed targets, as well as to specifying the elevation surface, elevation data. Accuracy of modeling the spatial distribution of ground cover types is a dual problem. First, one must decide the smallest size of distributed target which will be uniquely identified as a homogeneous region in the data base. Second, one must correctly interpret the source intelligence data from which the data base is built for determination of what kind of ground cover exists within each distributed target. Accuracy of specifying the elevation surface is also a dual problem. First, one must find a Nyquist sampling interval from the maximum rate-of-exchange of elevation in the area of interest and then relate this sampling interval to that required by the radar and applications. Second, one needs to determine the underlying accuracy of the source elevation data which are to be used.

There are several sets of criteria which interact to establish the sampling frequency of the ground and topography and, thus, the ground spot size each data base element represents: (1) the matters of accuracy just discussed, (2) questions of resolution from the standpoints of both the radar system and the application (these data are determined from Table 1), and (3) economic considerations raised by the amounts of resources available to construct a data base and to produce simulations from it. The intersection of these sets of criteria probably represents the best choice which can be made for sampling frequency and accuracy in a data base. If the sets are non-intersecting, then the decision must be made on another basis.

After determination of the sampling frequency and accuracy for which a data base is to be constructed of a specific site, work on building it can begin. Data bases are typically built by hand from various sources of intelligence data such as high resolution aerial photographs, etc. A radar/photo-interpreter (PI) acquires the necessary intelligence data and employs manual cartographic feature extraction techniques to interpret the source data and to develop the data base.

The PI draws a map by hand on a stable-base drawing media. This map consists of boundary lines separating different features and ground cover types. Boundaries of major features either can be traced or transferred from the source data, or both. Boundaries of minor features are difficult to locate and are, therefore, obtained via subjective interpretation criteria employed by the PI. These interpretation criteria are normally developed through experience and established to meet the appropriate requirements levied in Table 1. The construction of this hand-drawn data base map is a major effort if the desired resolution and accuracy is modest or better

(less than 50 meters) for a target site of minimal size (i.e., even for one of approximately 50 square kilometers). The resolution of the map depends upon the judgment of the PI, his knowledge of the target site, and his familiarity with ground cover and feature types found in a site.

When the hand-drawn data base map has been finished, it is a symbolic line drawing of the boundaries separating distributed targets (such as forests and fields) and the locations of cultural targets (such as buildings and roads). For use on a computer, this line drawing must be digitized and converted into a completely specified matrix.

A large table digitizer* has been used in the past to digitize the boundary lines in the data base map and to store these digital data on computer-compatible magnetic tape[8]. A human operator traces each boundary with the cursor, and the computer interfaced to the table periodically samples and records the position of the cursor. After digitization--a long, time-consuming task subject to countless errors--a computer-compatible magnetic tape (or multiple such tapes) contain the sampled points stored consecutively, serially, of each boundary in the original data base map. These serial digital boundary data next must be expanded into a completely specified matrix.

Special software have been developed to convert the serial digital boundary data into a completely specified matrix. The task is to sort the boundary data by their X- and Y-values and to fill-in the matrix. If it would

---

*A large table digitizer is here meant to be a table having a top surface one (1) by one and one-half (1½) meters, or more and having an underlying find grid of wires (i.e., 75 per centimeter). A cursor is used to trace drawings on the top surface with electric fields identifying the intersecting pair of wires the cursor passes over.

[8]McNeil, M., V.H. Kaupp, and J.C. Holtzman, "Digitization of Pickwick Site Data Base," Remote Sensing Laboratory Technical Report, RSL TR 319-4, University of Kansas Center for Research, Inc., February 1977.

be possible to assume that the digitized boundary data are error-free this task would not be difficult. Unfortunately, the digitized boundary data are subject to many errors. Both the human operator and the table and computer interfaces are sources of errors. Human errors range from inaccuracy of following lines with the cursor to completely missing boundaries, from incorrectly identifying each boundary to failing to identify some boundaries, and from tracing boundaries in the wrong direction to incorrectly registering the map, setting-up the coordinate system, and specifying the scale. Computer-generated errors range the complete gamut from scrambling the data to failing to operate. The multitude, variety, and complexity of errors in the digitized boundary data means this task requires a lot of interaction between man and machine because it isn't feasible, normally, to develop software "smart" enough to check for every error and correct for them in a single pass through the data. The software package to facilitate this and produce, ultimately, a completely specified data base matrix is described in Section 3.

At the completion of this activity, the hand-drawn data base map has been converted into a digital matrix. The data stored in each cell of this matrix is the data base information concerning the backscatter category of each spot on the ground. Each cell implicitly represents the location (X- and Y-location) of a ground spot relative to the known corner points and represents the ground spot size via the sampling frequency by which the data base was built, and explicitly specifies the backscatter category of each point. If seasonal or meteorological data are to be included in the data base, they have been added in the past by drawing a separate seasonal data base map, digitizing it, and juxtaposing these data into the category data.

Last, digital elevation data of the target site must be obtained and merged with the category data base. To this point, digital elevation data have been provided by various sponsors, thus, development work on this task has not been undertaken. The elevation data provided to date have come from DMA (Defense Mapping Agency) and have been produced either from standard 7½' quadrangle USGS maps (United States Geologic Survey) via suitable digitizing and interpolation techniques[9] or from stereoscopic photo-pairs and the UNAMACE[10] system. In either case, the work performed here has been limited to merging elevation data from different computer-compatible magnetic tapes with the category digital data base.

Merging of the elevation data with the category data on a single computer-compatible magnetic tape completes the task of constructing a data base of a specific site. Assuming that all problems encountered have been either solved or safely skirted, this tape contains a data base of a specific site at a desired scale, resolution, and accuracy, and this data base is ready for input to the simulation computer programs.

1.5.3 Reflectivity Data

After specification of key simulation parameters in Table 1 (e.g., frequency and polarization) and after identification of the different backscatter categories during data base construction, then reflectivity data (σ^o, or backscatter data) can be obtained. Reflectivity data are used in

---

[9] McNeil, M., V.H. Kaupp, and J.C. Holtzman, "Digital Elevation Data Base Construction: Pickwick Site," Remote Sensing Laboratory Technical Report, RSL TR 319-3, University of Kansas Center for Research, Inc., Lawrence, Kansas, July 1976.

[10] Bertram, S., "The Universal Automatic Map Compilation Equipment," Photogrammetric Engineering and Remote Sensing, Vol. 31, No. 2, March 1965.

the PSM to model the radar and ground interaction. The reflectivity data normally used in the PSM are backscatter, the differential scattering cross-section (σ^0). The PSM uses either experimental[1,11,12] or theoretical[13,14] data to simulate the interaction between the ground and the electromagnetic energy incident from the radar, and to predict the percentage of reradiation back to the radar. For each different ground cover of feature type specified in a data base, the PSM requires a set of σ^0 versus angle-of-incidence (a function of frequency and polarization), either experimental or theoretical to be input. The different ground cover, or categories specified in a data base can be classified into three (3) sets: (1) distributed targets; (2) cultural targets; and (3) seasonal or meteorological targets. Reflectivity data used in simulating radar responses for winter scenes are discussed in Section 4.

---

[1] Cosgriff, R.L., W.H. Peake, and R.C. Taylor, "Terrain Scattering Properties for Sensor System Design," Engineering Experiment STation Bulletin, 181, Vol. 29, Ohio State University, Columbus, Ohio, May 1960.

[11] Bush, T.F. and F.T. Ulaby, "Fading Characteristics of Panchromatic Radar Backscatter from Selected Agricultural Targets," IEEE Transactions on Geoscience Electronics, Vol. GE-13, October 1976, pp. 149-157.

[12] Ulaby, F.T. et al., "Radar Response to Vegetation," IEEE Transactions on Antennas and Propagation, Vol. AP-23, No. 1, January 1975, pp. 36-45, and "Radar Response to Vegetation II: 8-18 GHz Band," IEEE Transactions on Antennas and Propagation, Vol. AP-23, September 1975, pp. 608-618.

[13] Hevenor, R.A., "Backscattering of Electromagnetic Waves from a Surface Composed of Two Types of Surface Roughness," TR ETL-TR-71-4, Engineering Topographic Laboratories, The United States Army, Fort Belvoir, Virginia, October 1971.

[14] Fung, A.K. and H.L. Chan, "Backscattering of Waves of Composite Rough Surfaces," IEEE Transactions on Antennas and Propagation, September 1969.

Distributed targets are those areas which can be characterized by a differential scattering coefficient; this implies that the scattered, returned energy is composed of many returns whose phases are independent.[15] Each homogeneous region must be of a size that will provide a large number of scattering centers which are randomly located within it. When these conditions are satisfied, an average value of the differential scattering cross-section (σ^o) can be used to model the radar return from distributed targets. Most ground cover types located in data bases made to date satisfy these criteria reasonably well, thus, σ^o data were used to model the radar and ground interaction.

Cultural targets are defined within the context of radar simulation to be man-made objects. Their radar returns are characterized by the high probability of specular reflection, which is obviously dependent upon the construction geometry, orientation with respect to the radar platform, antenna beamwidth, and system resolution. The fact that radar returns from cultural targets are so highly dependent upon orientation, and the fact that they do not ordinarily satisfy the criteria listed in the previous paragraph illustrate why σ^o data for cultural targets are not usually obtainable and why cultural targets are not readily applicable to digital simulation.

An alternate means of indicating the existence of a cultural target in simulated radar data via the PSM has been utilized in the past, symbolic representation. In this representation, the cultural target is modeled as an isotropic radiator of X decibels. Thus, for any flight path and data base orientation, the cultural target is modeled as behaving the same.

---

[15]Moore, R.K., in The Radar Handbook (M.I. Skolnik, Ed.), McGraw-Hill, New York, 1970.

If this symoblic representation is not good enough, directional dependence can be introduced by specifying one or more directions relative to true North in which the cultural target is specular. This can be accomplished with a minimal increase in simulation complexity and cost, minimal relative to complete, accurate specification of each building, its geometry, corners, etc.

Seasonal or meteorological targets are defined here to mean the changes introduced in a data base to account for the perturbations introduced by adding seasonal or weather effects. The normal category data base represents a snap-shot of the target site. The seasonal or meteorological data base, as discussed in Section 2, is one way of allowing the target site to mature with time. Normally, complex models can be developed or possibly found in the literature which predict the functional dependence of the underlying category in a desired altered state, altered by season or weather. Some simulations have been produced from a single data base which represent different times of the year. These are shown in Section 5.

Upon obtaining the requisite reflectivity data, the work preliminary to simulating radar data is finished. The computer programs have been specialized, the data base produced, and the reflectivity data obtained. It remains only to enter these data into a computer and produce desired simulations.

1.6 Synopsis of Computer Programs

1.6.1 Introduction

As shown previously in Figure 1, the PSM computer programs can be utilized to form simulated radar data only after satisfaction of the three (3) input requirements for simulation parameters, data base, and reflectivity

data. The final computational algorithms of the PSM simulation do not represent the complete model. These algorithms, or modifications of them are invoked to produce final, desired results only after all the geometrical relationships between radar and ground spot (resolution element) have been solved and after all the propagational phenomena have been properly treated. These geometrical and propagational considerations are treated in the data handling and radar effects modeling which have been designed into a specific software realization of the PSM.

In general, geometric effects such as radar resolution size, local angle of incidence, range, etc. and propagational phenomena such as shadow, layover, compression, etc. are treated explicitly by the software. They are developed from such considerations as the flight path parameters, and the ground topography modeled in the data base.

The general PSM computer software and all specializations incorporate the same data processing philosophy. This philosophy requires the data base to be stored on computer-compatible digital magnetic tape (CCT) as a matrix developed in a rectangular coordinate system. The first step for the PPI implementations is conversion of this data base from rectangular to polar coordinates (SLAR, of course, uses the rectangular coordinate system).

The simulation computer programs are structured into two separate phases. The first phase accepts the CCT containing the data base and calculates all the geometrical and propagational effects. This phase predicts the power for each point in the data base from the geometrical data and stores the power data on an interim CCT. The second phase incorporates the resolution aspects of the system being modeled and combines the predicted power for the appropriate numbers of data base points into each radar

resolution cell. Finally, it converts the power predicted for each radar resolution element into the appropriate grey-tone value for each pixel (picture element) in the output image, and stores the results on a CCT.

If the application being simulated is a PPI radar, then the next function performed is conversion from polar coordinates back into rectangular coordinates. In either case, the final results are stored on an output CCT in a raster scan format for evaluation on a visual system such as a VDI*.

This philosophy is developed in a little more detail in the following section for one PPI implementation.

1.6.2 PPI Software Realization of PSM

A PPI is a forward sector scanning, real-aperture radar employing a scanning antenna. PPI's have historically been used as "forward-looking" sensors for USAF (U.S. Air Force) aircraft in the general roles of terrain-avoidance, nagivation, guidance, and in addition, in specialized roles. The geometry of a PPI radar is illustrated in Figure 2.

As shown in Figure 2, the PPI radar is mounted so as to illuminate the ground in a sector forward of the aircraft. The antenna of a PPI radar rotates (or electronically scans) a vector of the ground in front of the aircraft. For example, starting at 45^{o} to the right of the flight path the antenna rotates while illuminating the ground across a 90^{o} sector to 45^{o} to the left of the flight path. Other sectors can be scanned. During the time the antenna is rotating across the desired sector of ground, the radar transmits pulses of electromagnetic energy to the ground at a high rate, or PRF (Pulse Repetition Frequency). Each pulse illuminates the ground from

---

*VDI is a Video Digital Interface System manufactured by Interpretation Systems, Incorporated, Lawrence, Kansas.

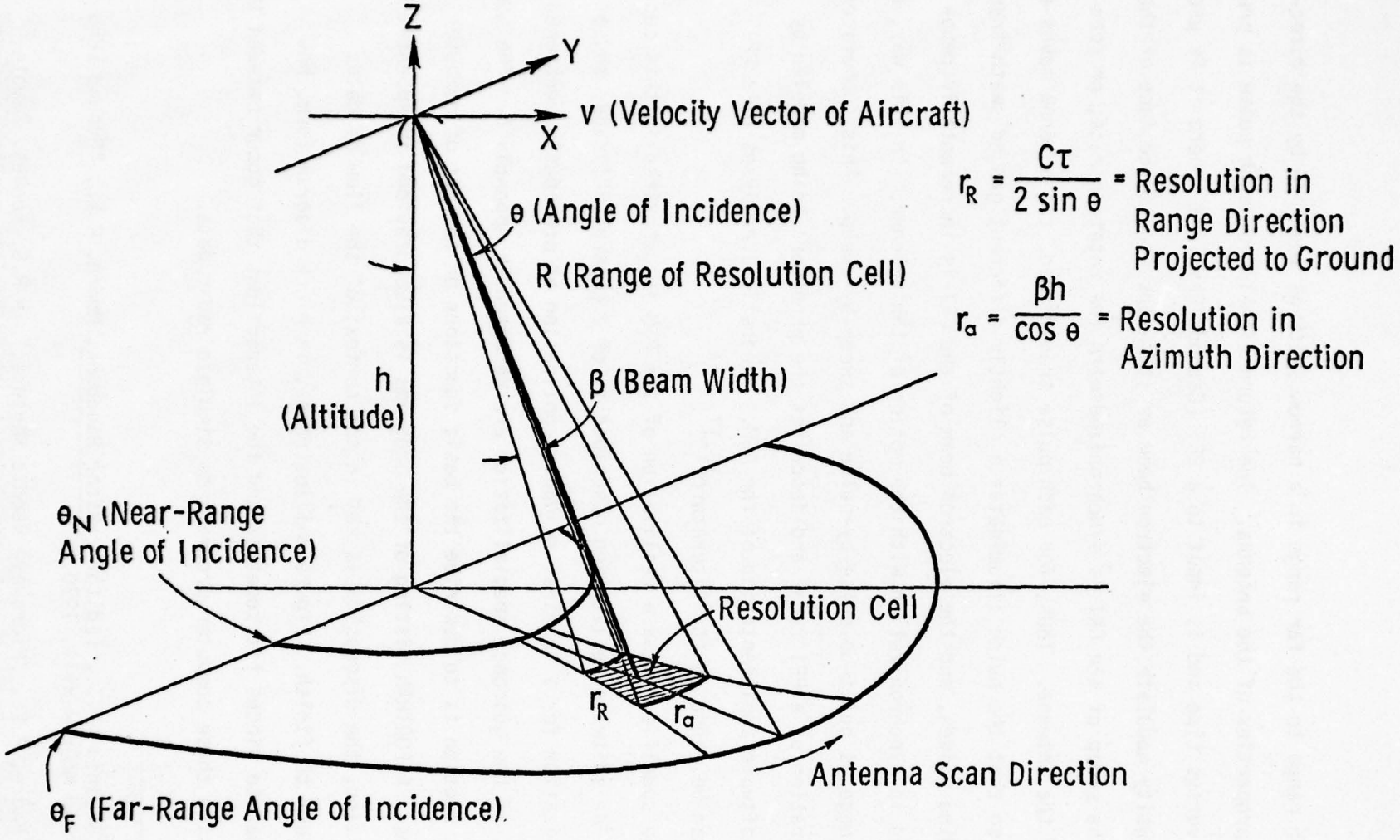


Figure 2. PPI Geometry.

the near range to the far range in a narrow swath as confined by the directional properties of the antenna. The return signal from each pulse is processed versus time and is input to a CRT (Cathode Ray Tube) where it is used to intensity modulate the electron beam as it sweeps across the face of the CRT. The sweep of the CRT is synchronized with the rotation rate, or scan rate of the antenna. Thus, for each pulse transmitted, the antenna moves a little so that the pulse illuminates a slightly different ground swath than preceeding pulses, and the electron beam of the CRT is incrementally repositioned in synchronization with the motion of the antenna. In this way, a radar image is built-up pulse-by-pulse and sweep-by-sweep. This summarizes the operation of a PPI radar and describes the phenomena being modeled by a PPI software implementation of the PSM. Theoretical analyses of a PPI radar can be found in the literature[16,17].

One specific software realization of the PSM is described in this section. The software realization described is of a general PPI radar and a specialization for a terminal guidance application as previously mentioned. Details of the guidance specialization are reported in Appendix C. The goal of this section is to describe the basic functions of the set of computer programs. A FORTRAN listing of the programs is also provided in Appendix C. In addition, the discussion is set in the context of the flow of data, from start to finish. Figure 3 illustrates, in block diagram form, how the separate pieces fit together and the interactions that occur between them when using these computer programs to simulate radar data.

---

[16]Skolnik, M.I., (Editor), Radar Handbook, Moore, R.K., "Ground Echo," New York: McGraw-Hill, 1970.

[17]Moore, R.K., "Microwave Remote Sensors," in R.G. Reeves, Remote Sensing Manual, Falls Church, Virginia, American Society of Photogrammetry, Chapter 9, 1975.

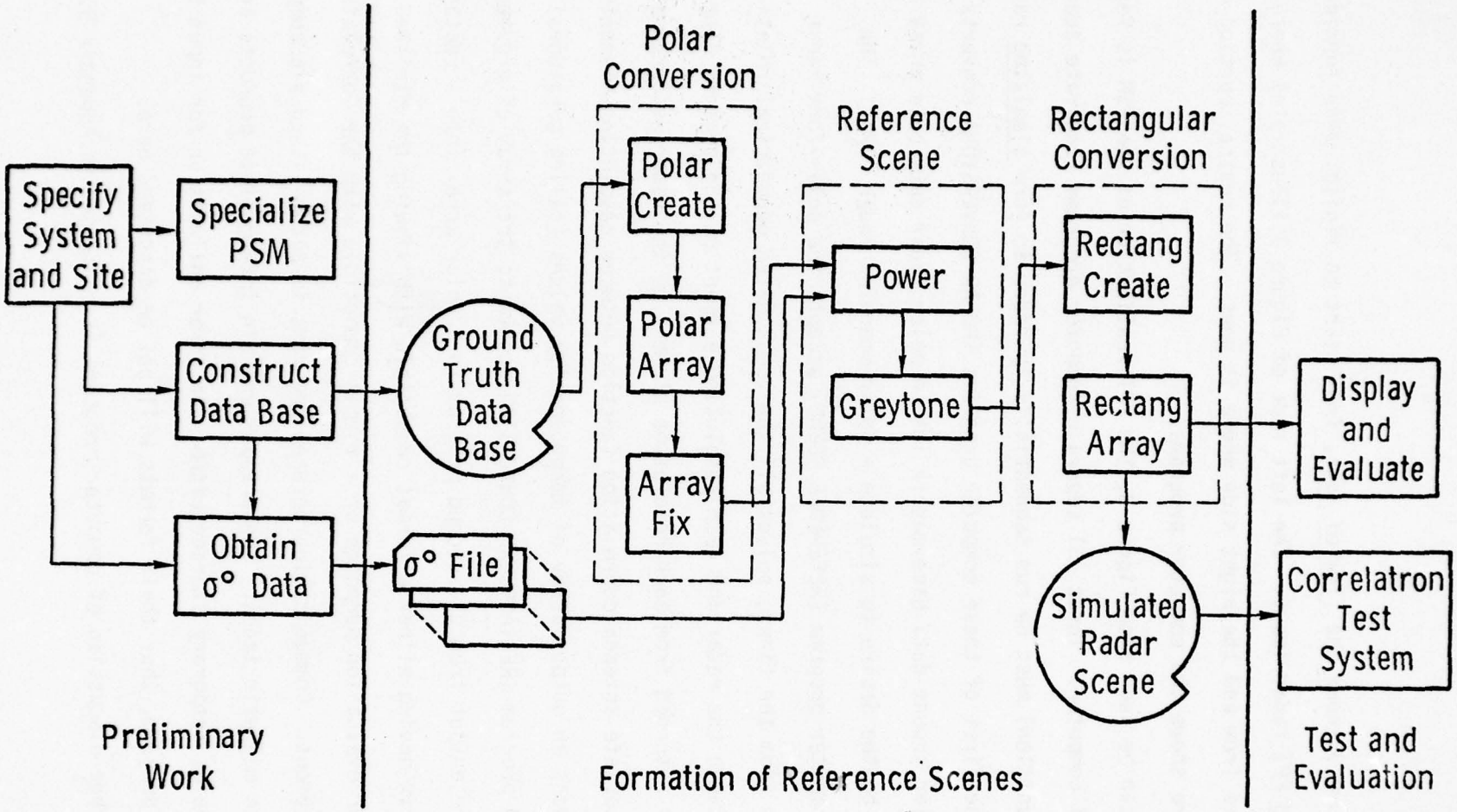


Figure 3. Flow of Data for Simulation of PPI Radar Data.

Figure 3 shows the flow of data, from start to finish, when forming a simulated PPI radar image. The left side of Figure 3 illustrates where the data comes from and the right side where it goes. The central portion of the figure shows the computer programs.

As can be seen from Figure 3, the PPI realization of the PSM is for a digital computer. Three (3) separate computer programs (separate according to function) must be run sequentially in order to form simulated radar data. The first of these computer programs (POLAR CONVERSION) converts a rectangular ground data base matrix into a polar radar data base array as dictated by the desire to simulate a polar-scanning radar (PPI). The second computer program (REFERENCE SCENE) accepts the polar-form radar data base from the first, solves all the complicated geometrical relationships between the radar and each resolution element on the ground, computes the power returned from each resolution element to the radar, incorporates the appropriate antenna correlation function between resolution elements, and produces an output array of image density values (called greytones). The third program (RECTANGULAR CONVERSION) converts the array of greytones which were output from the second program in a polar array into a rectangular grid matrix having either a format compatible with viewing the finished simulation for evaluation purposes or a format compatible with the Correlatron* test equipment. Communication between programs is accomplished via computer-compatible magnetic tapes. These tapes contain intermediate products and only serve as temporary storage either for error analysis or for input to the next program, thus their formats will not be discussed here.

Further discussion of computer programs is contained in Appendix B.

---

*Correlatron is the name of a two-dimensional cross-correlation measuring device manufactured by Goodyear Aerospace.

2.0 PRODUCTION OF SURFACE FEATURE OVERLAYS FOR USE IN THE SIMULATION OF WINTER SCENE RADAR IMAGERY OF THE WATERTOWN, NEW YORK, AREA

A data base was constructed for a target site centered on Watertown, New York, from which any season and, especially, winter scene radar images could be simulated. The scene initially was aseasonal, that is, it was lacking in any characteristics which would define it as belonging to a particular season of the year. Vegetation cover types, for example, were listed initially as a unique number within the set assigned for a general category such as "deciduous woodland" or "grassland," with no phenological descriptors such as "deciduous woodland, leaves "absent" or "grassland dormant." Since each distributed target had been assigned a unique number, appropriate modifiers to place the vegetation into a particular season such as the winter state and to cover the landscape with snow and the water with ice were added later.

This task was concerned primarily with the determination of the identity, areal distribution, and condition of the various surface cover types of the Earth's surface in the vicinity of Watertown, New York, and the preparation of cover category maps at scales of 1:100,000 and 1:24,000 to display these features for digitization and subsequent integration into the radar simulation process. The reasoning behind this approach and the materials and methods utilized are described in the following sections.

The 1:100,000 scale map was developed to support relatively high altitude radar simulations (> 4,500 m) and the 1:24,000 scale map was developed for low altitude simulations (> 1,000 m).

In the preparation of any map, regardless of its application, it is necessary to begin with an already available base map or to prepare a base map from aerial photographs and/or satellite imagery. To this base map,

then, additional information may be added as appropriate for the application. For the Watertown radar simulation data base a film positive orthophotomosaic (scale 1:100,000) prepared and supplied by the U.S. Army Engineer Topographic Laboratory served as source information for the location of land/water boundaries and cover-type boundaries for the 1:100,000 scale base map. For the preparation of the 1:24,000 scale base map, the center of the orthophotomosaic, covering an area with a radius of approximately eleven (11) kilometers around the city of Watertown, was photocopied and printed on paper at the enlarged scale. The maps were drawn on mylar film laid directly over the photographs.

For identification of surface cover types, it was decided to rely on stereo examination of the original 9 x 9-inch film positive air photos, acquired by Mark Hurd Aerial Surveys, from which the orthophotomosaic had been prepared. This source was supplemented with LANDSAT MSS (Multi-Spectral Scanner) imagery, color infrared aerial photographs of portions of the study site, maps of the Fort Drum Military Reservation supplied by USAETL[18] and the open technical literature[19,20,21,22,23]. The decision to make a versatile aseasonal base map was based on three factors.

---

[18]U.S. Army Terrain Analysis Laboratory, "Fort Drum, New York, Terrain Analysis," U.S. Army ETL, Fort Belvoir, Virginia, October 1977, 45 pages.

[19]Bowman, I., Forest Physiograph - Physiography of the United States and Principles of Soils in Relation to Forestry, John Wiley and Sons, New York, 1914.

[20]Gordon, R.B., "The Primeval Forest Types of Southwestern New York," New York State Museum Bulletin Number 321, 1940, pp. 1-102.

[21]Kuchler, A.W., "The Potential Natural Vegetation of the Conterminous United States," Map and Accompanying manual, Americal Geographical Society, New York, 1964.

[22]Roberts, E.A. and H.W. Reynolds, "The Role of Plant Life in the History of Dutchess County, New York," Dutchess County Planning Board, Poughkeepsie, New York, 1938, 44 pages plus map.

[23]Schepis, E.L., "Time Lapse Remote Sensing in Agriculture - An Application of Aerial Photographs," Unpublished Master's Thesis, Cornell University, 1968, 116 pages.

First, it might be desirous at some future date to simulate a radar image of the Watertown site for a different season.

Second, most of the surface category boundaries would be evident at all seasons, even with deep snow cover; only the condition of the surface cover within each bounded area would change with season, and little effort would be saved by eliminating those boundaries which would be obliterated by snow cover. By coding each bounded area by cover category type, it is possible to vary the radar backscatter input as necessary for each category to fit any season desired. The scheme is limited only by the availability, or lack thereof, of empirical backscatter data for the cover types and conditions to be simulated. Such a system has the added advantage of being easily modified to accommodate changes due to road construction, urban expansion, etc. This is especially true of the 1:24,000 scale base map on which each delimited zone was given a unique identification number within its category type, thus making it possible to access and alter, if desired, the computer file contents representing each individual zone (field, road, lake, etc.).

Third, it was desired to compare several different techniques for incorporating the scene characteristics representative of winter conditions. Some of these techniques involved adding snow cover either zone-by-individual-zone, or simultaneously to an entire cover category; others involved the blanket addition of snow of various depths, the estimation of which was based on knowledge of topography, tree heights, prevailing winds and average snowfalls for the months of December through March. Obviously, only an aseasonal base map would provide the versatility necessary for these comparisons.

TABLE 2

TIME EXPENDITURES FOR PRODUCTION OF ALL WATERTOWN ASEASONAL AND WINTER OVERLAYS

	DISTRIBUTED TARGET OVERLAYS	POINT TARGET OVERLAYS	SNOW DEPTH OVERLAYS	TOTAL
High altitude	71	15	20	106
Low altitude	216	40	20	276
Total	287	55	40	382

The details of the aseasonal base map construction and a discussion of several approaches for incorporating winter season descriptors into the Watertown scene are presented in the following sections.

2.1 Preparing an Aseasonal Scene Data Base

Two sets of base maps with different levels of resolution were created. For higher altitude simulations, base maps at a scale of 1:100,000 were prepared; for lower altitude simulations, a scale of 1:24,000 was selected. The maps were drawn with pencil onto .005" matte-finish mylar film laid directly over the source imagery. To reduce confusion due to overcrowding of the overlays, and subsequent errors in overlay production and digitization, the surface feature information was divided between two separate overlays. Boundaries of distributed targets such as water bodies, vegetation, and bare soil zones were traced onto one overlay. Point targets, including urban areas, small clusters of buildings, railroads, bridges and roads were traced onto the second. A third overlay was constructed for describing season features.

As with any vegetation map, positions of the category boundaries must be somewhat arbitrary when vegetation changes are gradual. However, where the cover types change abruptly, location of the boundaries is estimated to be accurate to within $\pm$ 37 m at a scale of 1:100,000 or $\pm$ 9 m at a scale of 1:24,000.

Total time required for production of the aseasonal scene overlays is estimated at 382 man-hours broken out as indicated in Table 2. Descriptions of the two data bases produced follow.

1:100,000 Scale (High Altitude) Overlays

For the high altitude aseasonal overlays, the Earth's surface was divided into 17 cover-type categories labeled with numerical codes as indicated in Table 3. All zones of coniferous forest were labeled as "10"; all zones of deciduous forest as "20", and so on.

Vegetated areas were distinguished primarily by their physiognomy, or gross canopy architecture, rather than by species composition since radar backscatter will depend more on the geometry and spatial distribution of plants and plant parts, the presence or absence of leaves, etc., than on floral characters and other visually minor features used for taxonomic separation of plant species. No attempt was made to assign crop names to cultivated fields since, during winter months, the fields would contain only plant residues or tilled soil. Although none of the water bodies were frozen in October when the aerial photographs were taken, boundaries were located in the water to permit sequential freezing of the water if desired for later season simulations. Locations of freezing boundaries are somewhat arbitrary but were based on prevailing wind direction during winter months and topography of the surrounding land surfaces. Individual road segments were given unique numbers to permit their removal or dimensional alteration if necessary, but roads were not subdivided by surface type or size. Cities were divided into sections which were dominated by trees, buildings, open ground, asphalt, or by some mixture thereof.

1:24,000 Scale (Low Altitude) Overlays

For the low altitude overlays, 30 cover-type categories were used (see Table 4). Within the numerical identification code system, each bounded area was given a unique number so that it could be easily accessed and

TABLE 3

COVER-TYPE CATEGORIES FOR 1:100,000 SCALE OVERLAYS
(WATERTOWN SITE)

	IDENTIFICATION CODE	CATEGORY
Distributed target overlay:	10	Coniferous forest
	20	Deciduous forest
	30	Shrubland or scrub
	40	Swamps and marshes
	50	Grass
	60	Cultivated fields
	70	Bare ground
	80	Water, never freezes
	81	Water, first to freeze
	82	Water, second to freeze
	83	Water, third to freeze
	95	Unidentified*
Point target overlay:	2	Urban area, mainly trees
	4	Urban area, mainly buildings
	5	Urban area, equal portions of buildings and trees
	7	Urban area, mainly open ground, with or without a few buildings and trees
	8	Urban area, mainly asphalt surfaces
	6	Urban area, unidentified*
	8100 - 8207	Roads
	8555	Railroads
	8255	Roads, unidentified*

*These numbers were for the use of the digitizing team only and were to be assigned to map features which had been drawn by the overlay team but had failed to receive an identification number.

TABLE 4

COVER-TYPE CATEGORIES FOR 1:24,000 SCALE OVERLAYS
(WATERTOWN SITE)

	IDENTIFICATION CODE	CATEGORY
Distributed target overlay:	1120	Black River
	1200 - 1206	Small rivers
	1300 - 1334	Small water bodies
	1400	Lake Ontario
	1600 - 1606	Swamps and marshes
	1700 - 1774	Bare ground
	1800 - 1812	Asphalt
	1900 - 1911	Gravel
	2000 - 2155	Coniferous forest
	2300 - 2480	Deciduous forest
	2524 - 2999	Mixed forest
	3000 - 3574	Grass
	4000 - 4487	Shrubland or scrub
	5000 - 7965	Cultivated fields
Point target overlay:	1000	Lines of trees
	8200	Heavy-duty roads, without trees
	8233	Heavy-duty roads, tree-lined
	8400	Medium-duty roads, without trees
	8455	Medium-duty roads, tree-lined
	8600	Light-duty roads, without trees
	8677	Light-duty roads, tree-lined
	8800	Unimproved dirt roads, without trees
	8899	Unimproved dirt roads, tree-lined
	8911	Railroads, without trees
	8988	Railroads, tree-lined
	9011 - 9246	Urban areas
	a	Mainly trees
	b	Mainly buildings
	c	Trees and buildings
	-	Open ground
	d	Asphalt
	9999	Large buildings

altered, if necessary, in the computerized data base. For example, all areas of bare ground were assigned a number in the 1700's, but each had a unique number (1701, 1702, 1703, etc.). Vegetation classes for the 1:24,000 scale overlay are identical to those for the smaller scale, except that the category of mixed coniferous/deciduous forest was added. No freezing boundaries were added to water bodies, since it appeared that all water surfaces within the smaller 1:24,000 scale test site would be frozen over during the target winter month. Roads, although not uniquely numbered, were identified as to type (heavy-duty, medium-duty, light-duty, or unimproved dirt). Narrow lines of trees, such as along roads or around fields, were delimited on the larger scale map, but not on the smaller scale map. They were drawn on the second overlay with roads, etc., since that overlay is less crowded and chances of overlooking them during the digitizing process were believed to be smaller. Urban areas were subdivided into areas dominated by trees, buildings, asphalt, or a mixture of trees and buildings. Because the four-digit identification numbers used would not fit into the small spaces delimited within the urban areas, each number was given an alphabet letter suffix (a = mainly trees; b = mainly buildings; c = equal amounts of trees and buildings; and d = mainly asphalt), and only the code letter was written in each bounded area within a city.

2.2 Superimposing Winter Characteristics on the Aseasonal Scene

As previously noted, the Watertown data base was constructed to satisfy objectives of a study performed for USAETL. USAETL specified that a data base be constructed for an "average" winter month. USAETL specified February as the target winter month. By "average" winter was meant one

which literally incorporated the seasonal aspects of the average of recent years. February 1975 was chosen as a model "average" winter. February was selected to satisfy study requirements, and 1975 was selected because climatic conditions that year approached the "average" for recent years. Table 5 lists temperature and snow depth in February for the years 1974-1976; 1977 was not considered "typical" for the Watertown area because of an extremely heavy snowfall and accumulation. Allowances for the seasonal changes must be incorporated into the data base. To change an aseasonal scene into a seasonal scene, information must be added to describe: (1) condition of the natural vegetation, (2) condition of the cultivated fields, (3) ice cover (if present) of water bodies, and (4) snow cover (if present) of all surfaces, for the particular date being simulated.

During February, most deciduous trees and shrubs would be completely leafless and in a dormant condition. Above-ground parts of grasses and other herbaceous species would be mostly dead and dry, if not altogether absent.

Cultivated fields would be devoid of crops, although some might be fallowed. Corn stubble might be left standing, but other crop residues would probably be turned under.

Ice thickness data for central New York state were difficult to obtain. However, data[24,25] suggest that most water surfaces would be completely frozen in February. Because average daily temperature maxima for 1974 and 1975 were below freezing, it seemed reasonable to assume, for purposes of this

---

[24]Bates, R.E., "Winter Thermal Structure and Ice Conditions on Lake Champlain, Vermont," CRREL Report 76-13, U.S. Army Corps. of Engineers Cold Regions Research and Engineering Laboratory, Hanover, New Hampshire, 1976.

[25]Bilello, M.A. and R.E. Bates, "Ice Thickness Observations, North American Arctic and Subarctic, 1962-63, 1963-64, Pt. 3," Special Report No. 43, U.S. Army Corps of Engineers Cold Regions Research and Engineering Laboratory, Hanover, New Hampshire.

TABLE 5

WATERTOWN WINTER SCENE TEMPERATURE AND SNOW DEPTH
[*U.S. Department of Commerce National Oceanic and Atmospheric Administration*, 1974-1976]

		TEMPERATURE			SNOW DEPTH (cm)	
FEBRUARY	STATION	AV. MAX. ^{o}C	AV. MIN. ^{o}C	AV. ^{o}C	TOTAL	MAX. DEPTH ON GROUND
1974	Wtwn[1]	-2.44	-14.40	-8.44	17.0	7.6
	WtFAA[2]	-3.11	-13.20	-8.16	14.7	10.2
1975	Wtwn	-0.05	- 8.88	-4.44	54.1	17.8
	WtFAA	-1.05	- 9.66	-5.33	49.0	25.4
1976	Wtwn	3.72	- 9.44	-2.83	39.3	30.4
	WtFAA	4.94	- 6.61	-0.83	13.2	6.4

[1]Wtwn = Watertown

[2]WtFAA = Watertown airport

data base, that the ice surfaces would remain frozen during the day, and no attempt was made to allow for standing water which might be present on top of partially thawed ice. Because ice thicknesses for February for the test site are generally considerably greater than the skin depth of the radar system being modeled, it was not deemed necessary to include ice thickness contours on the overlays. As was mentioned earlier, however, by digitizing ice/open water boundaries into the aseasonal scene, the option of changing the ice/open water ratio on Lake Ontario and its major tributaries was retained.

Snow depth data were obtained from the USDA NOAA* 1976-1976 records of the Watertown and Watertown airport stations[26]. All other New York reporting stations with recording gauges and detailed weather records were well beyond the borders of the test site. To simplify the task, it was assumed that temperatures would always remain below freezing and no free water would exist within the snow layer; no attempt was made to model incident solar radiation and resultant snow-melt patterns, since no supporting data on layering effects were available. The snow layer was assumed to be vertically homogeneous.

Four techniques were evaluated for satisfying the requirements of the sponsor of this work for estimating snow depth and incorporating that into the simulation algorithm. These are described below.

2.2.1 Isometric Addition of Snow to the Ground Throughout the Scene

The simplest approach to adding snow to the winter scene is simply to blanket the ground everywhere with an equal measure of snow regardless of

---

*U.S. Department of Commerce, National Oceanic and Atmospheric Administration.

[26]U.S. Department of Commerce National Oceanic and Atmospheric Administration, "Climatological Data: New York, Volumes 86-88," Environmental Data Service, National Oceanic and Atmospheric Administration, U.S. Department of Commerce, 1974-1976.

vegetation cover, topography, direction or degree of exposure or direction of the wind. This has the obvious advantage of requiring no time for the production of an overlay. It is versatile in that any depth of snow can be added. It does, however, ignore variations in snow depth which are known to be related to other scene variables such as those mentioned above and therefore offers a very rough approximation of any snow mantle which would actually be present. For the Watertown area, a snow base of 15 cm would fall within the range of values recorded for maximum depth on the ground on any given day.

2.2.2 Varying Snow Depth Surface Category Type

Snow accumulates to different depths in forests as compared to open fields, or even in deciduous forests as compared to needleleaf evergreen forests[27]. By taking advantage of this knowledge, it is possible to model the snow cover slightly more realistically than is possible by isometric addition of snow, while still avoiding the expenditure of time for manual production of a separate overlay. It is necessary only to apply different "snow compensation: coefficients according to surface cover type categories of the aseasonal scene overlays."

Snow depths to be used for the simulation were selected on the basis of ground truth data reported[27] for south-central New York. Their data were

---

[27] J.N. Spaeth and C.H. Diebold, "Some Interrelationships Between Soil Characteristics, Water Tables, Soil Temperature and Snow Cover in the Forest and Adjacent Open Areas in South-Central New York," New York Agricultural Experiment Station Memoirs, No. 213, Cornell University, Ithaca, New York, 76 pages.

standardized by transforming their absolute reported values into ratios and then using these ratios to adjust 1975 base values. Where cover categories differed from theirs (e.g., urban areas, railroads, etc.) "reasonable" estimates were used for snow cover accumulation. Table 6 lists snow depths used for this approach.

2.2.3 Varying Snow Depth with Exposure to Wind

To take wind direction into account in an effort to anticipate drifting and blowing patterns, one should rightfully have some indepth knowledge of the effects of shape, size, and location of surface protuberances (houses, hills, stands of trees, etc.) on wind speed and direction, on a local scale as well as a larger scale. Snowflake size and water equivalency may also affect snow accumulation patterns. However, exact conditions of wind speed and direction, snow water-equivalency, snow depth, etc. may vary significantly from one snowfall to another; hence, drift patterns can be expected to vary as well.

This approach to winter scene modeling was consistent with the sponsor's requirements and, thus, a third overlay was constructed to indicate snow depth contours on a coarse scale. Prevailing winds were assumed to blow from the southwest, depositing less snow on exposed hilltops and bluffs and more snow in protected sites on the leeward side of hills, bluffs, and wooded areas. Valleys more or less perpendicular to the prevailing wind direction were allowed to accumulate more snow than those more or less parallel to the wind direction. No variations in snow depth were allowed for different vegetation types. Table 7 indicates the snow depth categories selected. The base may be varied to fit local conditions for the date to be simulated. For this case, base is 15 cm. The same identification codes

TABLE 6

WINTER OVERLAY-CODES FOR VARYING SNOW DEPTH WITH SURFACE COVER TYPE CATEGORY

IDENTIFICATION CODE		COVER TYPE	SNOW DEPTH
1:100,000	1:24,000		
10	2000 - 2155, 2524 - 2670	Coniferous forest, mixed coniferous/deciduous, in part	base* x 5
20	2672 - 2999, 2300 - 2480	Deciduous forest, mixed coniferous/deciduous, in part	base x 2
30	4000 - 4487	Scrublands	base x 1.5
40	1600 - 1606	Frozen swamp	base x 1.5
50	3000 - 3574	Grassland	base
60	5000 - 7965	Agricultural	base
70	1700 - 1774	Bare ground	base
--	1900 - 1911	Gravel pit	base x 2
80	1400	Open water	no snow
81, 82, 83	1120, 1200 - 1206, 1300 - 1334	Frozen water	base
	9001 - 9246	Urban areas:	
2	a	Mainly trees	base x 2
4	b	Mainly buildings	no snow
5	c	Trees and buildings	base
7	-	Open ground	base
8	d	Asphalt	
	8200, 8233, 8400, 8455	Heavy- and medium-duty roads, with or without trees	no snow
8100 - 8207	8600, 8677	Light-duty roads, with or without trees	base x 0.5
	8800 - 8899	Unimproved dirt roads	base
8555	8911, 8988	Railroads	no snow
----	1800 - 1812	Asphalt	no snow

*Snow base for this simulation was set at 15 cm

TABLE 7

OVERLAY CODES FOR VARYING SNOW DEPTH WITH EXPOSURE TO WIND

IDENTIFICATION CODE (1:100,000 and 1:24,000 SCALE)	SNOW DEPTH
000	No snow
100	Base*
200	Base + 15 cm
300	Base + 30 cm
400	Base + 45 cm

*Snow base for this simulation was set at 15 cm

and depth categories were used for both the low and the high altitude overlays. Approximately 20 hours each were required to produce the high and low altitude snow depth overlays (see Table 2).

4. Varying Snow Depth with Surface Type and Exposure to Wind

In the real world, snow accumulation and drift patterns will be related not just to surface type, vegetation type or exposure, but to a combination of these factors plus wind speed, snow wetness, etc. Incorporation of snow characteristics into the simulation algorithm does not appear feasible at this time. However, computer combination of both the wind exposure overlay and the surface type variations in snow depth may be possible. This method was not tested but is recommended for future research.

2. Watertown Data Base Specifications

Three types of overlays were constructed for both the 1:24,000 and 1:100,000 scale data bases: (1) distributed target, (2) cultural target, and (3) snow depth. These overlays were digitized and merged with elevation data as discussed in Section 3.

Data Base Specifications -- Watertown

Specification of pertinent parameters for the 1:24,000 scale data base is presented in Table 8, and for the 1:100,000 scale data base in Table 9.

TABLE 8

PARAMETER SPECIFICATIONS: 1:24,000 SCALE DATA BASE
(WATERTOWN SITE)

Data base size:	Approximately 23.2 km x 23.7 km (14.4 x 14.7 miles)
Site location:	Center located at coordinates 43^{o} 58' 43"N by 75^{o} 52' 31"W
Spatial sample size:	6.25 m (20.5 feet) in both range and azimuth
Elevation data accuracy:	Estimated to be approximately $\pm$ 12.5 m ($\pm$ 41.0 feet)
Backscatter category resolution:	Estimated to be approximately 30.5 m (100 feet) in both range and azimuth
Number of elements in digital matrix:	14,105,600 (3712 records, each containing 3800 points) at 6.25 m per sample in both range and azimuth; N = 3712 records and M = 3800 points
Scale:	1:24,000
Source intelligence data used:	
Elevation data:	Provided by ETL from the output of the UNAMACE elevation data computer program
Category data:	Spatial geometry and detail was obtained from 1:100,000 scale orthophoto (rectified geometry) and distributed category boundaries interpreted from 1:100,000 high-resolution aerial photographs which also served as the source for the orthophoto. (Note: the geometry of the terrain in the orthophoto was rectified for a tangent plane approximation to the Earth centered on the target center).

TABLE 9

PARAMETER SPECIFICATIONS: 1:100,000 SCALE DATA BASE
(WATERTOWN SITE)

Data base size:	Approximately 80.8 km x 85.0 km (50.2 x 52.8 miles)
Site location:	Center located at coordinates 43^{o} 58' 43"N by 75^{o} 52' 31"W
Spatial sample size:	25 m (82.0 feet) in both range and azimuth
Elevation data accuracy:	Estimated to be approximately $\pm$ 12.5 m ($\pm$ 41.0 feet)
Backscatter category resolution:	Estimated to be approximately 100 m (328 feet) in both range and azimuth
Number of elements in digital matrix:	10,988,800 (3232 records, each containing 3400 points) at 25 m per sample in both range and azimuth; N = 3232 records and M = 3400 points
Scale:	1:100,000
Source intelligence data used:	
Elevation data:	Provided by ETL from the output of the UNAMACE elevation data computer program
Category data:	Spatial geometry and detail was obtained from 1:100,000 scale orthophoto (rectified geometry) and distributed category boundaries interpreted from 1:100,000 high-resolution aerial photographs which also served as the source for the orthophoto. (Note: the geometry of the terrain in the orthophoto was rectified for a tangent plane approximation to the Earth centered on the target center).

3.0 CONSTRUCTION OF A DIGITAL DATA BASE FROM SURFACE FEATURE OVERLAYS

3.1 Definition of Work Performed

A digital data matrix which comprehensively describes the target scene is required as input for the PSM radar simulation package. Each number in this input matrix is encoded to describe a fixed area of ground in the target scene. The area represented by each value in this matrix defines the resolution of the data base.

The information required at each point in the data base includes:

(1) the local elevation above sea level,

(2) a category assignment for the ground cover of that cell, and

(3) aseasonal category assignment.

This section describes the techniques used to transform hand-drawn maps into digital form and to combine the results into a single complete digital data matrix for input to the radar simulation programs (such as those listed in Appendix C).

3.2 Construction Approach

A single general format was carefully defined for the various category maps to be input to the data base construction package so that all of them could be transformed into digital format with the same software. Each map consists of a series of line boundaries that completely segment the target area into distinct fields. The category type of each field is identified by a category number written inside the field boundary. No other details are included in a map except for a series of registration marks which are used to define precisely the scale and orientation of the map so that it may be registered with other maps and elevation data. In the cultural

map which includes linear targets such as roads and railways, a special value was associated with these lines to indicate that they represent a category type themselves and not merely a boundary between two category types.

A digital representation of the information included in each of these maps was generated by tracing the border of each field in the map on a large-scale digitizing table. Then a series of computer programs were run to convert the digital data representing field boundaries into a complete two-dimensional matrix in which each cell contains the category value corresponding to the field in which it is found.

The software package which was used to create and assemble the final digital data base can be divided into three major segments. After the digital data representing field boundaries were acquired, it was necessary to identify each field boundary in the data stream and insure that the boundary was continuous (at the resolution of the final data base) and formed a closed region (i.e., the last point of a boundary returned to the initial point of the boundary). At this stage, end markers were placed in the data stream to separate distinct borders, and when discontinuities were found points were added to the data stream to form complete boundaries. In the next step, for each closed boundary, the points representing the border were converted to matrix cell locations and all those matrix cells lying within the enclosed region were assigned the corresponding category value. Finally, after each of the required digital matrices were completed (e.g., distributed target, cultural target, and seasonal map), they were merged together along with the elevation matrix to form a single output unit containing all of the required information. Each of these steps will be described in detail in the following sections.

3.3 Digitizing Surface Feature Overlays

At the beginning of the data base construction process, the only input is a collection of hand-drawn feature maps which must be transformed into a digital data base. Since the first program in the series requires a digital input, it is necessary to somehow represent these maps digitally in a form which the programs can then manipulate. This is accomplished by digitizing the maps on a digitizing table. A Bendix digitizing table interfaced to a dedicated minicomputer, which in turn is connected to a seven-track magnetic tape drive for output, was used. For verification of a day's work, there was also a Calcomp plotter which, with some software support, was able to read the tape and plot out the areas which had already been digitized. This enabled the operators to check for areas which were digitized twice, or that were missed. The digitized boundary data were stored serially on computer-compatible magnetic tape for the remainder of the digital processing.

For ease of construction, it was decided that each area representing a given field would be digitized by tracing its entire boundary. This implies that there would be line segments which were digitized twice, but the loss of time was outweighed by the fact that the category number for a field could be followed by all of the coordinates defining the boundary of the field. The other major a priori decision was that each field would be digitized in a clockwise direction. This was required because one of the later programs (AREAFIX, see Appendix B) requires that it know the direction of travel of the line.

Before the start of each digitizing session, reference points were entered via the table. This was necessary for two reasons. First, the reference points defined the scale and orientation of the map. Since the completed data base must coincide with the elevation data base, these

reference points were chosen to correctly orient the digitized data in the same scale and direction as the elevation data. Next, since the maps were too detailed to be completely digitized in a single session, the reference points insured that the data from different sessions were correctly scaled and oriented with respect to each other, thus insuring geometric fidelity between the output of different sessions. For these reasons, great care was taken in both the determination and registration of the reference points.

The actual digitized data for each field consisted of four pieces of data. First, a symbol was output to indicate the beginning of a new field. This was optionally followed by a category number for the field to be digitized, if it differed from the category of the field previously digitized. Next, there were the actual x-y-coordinates of the field boundary. Finally, when all of the points for a single field were recorded, a special end-of-boundary symbol was written to the output tape.

The digitizing system supported two modes of operation for tracing the field--continuous and point modes. In the continuous mode, the digitizing system sampled the cursor position at regular (10 milli-second) time intervals and outputted an x-y-coordinate pair reflecting this position in the coordinate system defined by the reference points. With this mode, the individual simply traced the border with the cursor and the system automatically recorded data. Care had to be taken, however, not to stray from the line being traced since data were being collected continuously and it was not possible to erase errors. If an error were made during the digitization of a field boundary, all of the data for that boundary had to be deleted and the entire boundary redigitized. This mode was used for irregular boundaries. For rectangular fields, the point mode was used in which the operator needed only to record the corner points. Software was used

at a later time to process these data (INITFIX, see Appendix B) for connecting between the points. This mode greatly reduced the volume of data collected and stored, and also eliminated unwanted operator irregularities created when tracing a straight line.

After each day's work, the output tape was input to the plotting device to produce a visual record of the data collected. Closely examining these plots allowed the operators to detect errors, such as fields that were not digitized or that were incorrectly digitized so that these errors could be corrected in the following digitizing session.

3.4 Construction of a Digital Data Base Matrix

3.4.1 Introduction to the Approach

The transformation of digital data representing field boundaries into a two-dimensional digital matrix is conceptually quite simple. The points representing a field boundary are placed in the appropriate cells in the matrix and then an algorithm is executed to fill in all those cells enclosed by the closed line segment with the appropriate category number. The size of the data base to be produced, however, introduced some significant problems for implementation of this conceptual approach. The size of the data base matrices to be produced was approximately 3200 lines by 3200 columns, or about 10,000,000 cells per data base. This number is about 100 times larger than the amount of main memory available at the computer facilities, so it was clearly infeasible to store the entire data matrix in memory.

To overcome computer memory space limitations, an approach was developed to construct the two-dimensional matrix a single column at a time. The basic premise being that a column of the data base can be constructed if the bottom and top points are known for each field in the data base that the given column intersects.

The first major step to be performed in the data base construction sequence was to correct known errors in the original input data and to modify the format of the data to one which was more convenient for the remaining programs in the sequence. The errors in the input data were of two general types. The first type of error was discontinuities in line segments. There cannot be any errors in a field boundary for the construction process to execute correctly. Since the input data were sequential, gaps were easily detected by calculating the distance between adjacent input points. If a gap were found, a simple connect routine calculated the points required to fill in the gap. The second type of error was the result of errors occurring during the original digitizing process. Errors of this type include assignment of the wrong category to a field boundary, digitizing the same field more than once, or only partially digitizing a field. These errors had to be identified either by the operators themselves or through program checks at various points in the sequence. These errors were corrected by eliminating bad data points and changing incorrect category assignments, etc., in the data system.

The next step in the sequence was to label each point of a line boundary as a bottom, top, or interior point for that boundary. Interior points occur within vertical line segments and were deleted since only the bottom and top points of a boundary were needed. All boundaries were digitized in a clockwise manner to facilitate this labeling of the points. Given this information and the direction of the line segment before and after a point in the line, determine whether a given point is a bottom, top, or interior point. Direction of the line segment was easy to determine since the data remained in sequential order.

Having labeled each point as a bottom or a top, the points were then sorted according to the column in which they belonged so that each column could be constructed independently. At this stage, the sequential order of the data was lost since that information was no longer required.

Finally, each column of the completed data base was constructed independently from the list of bottom and top points falling in a given column. First the points in a column were sorted by their line number from smallest to largest. Then the column was expanded to its full length by assigning the appropriate category number to all those cells between the bottom and top points of a particular field.

3.4.2 Summary of Computer Programs

A description of all computer programs used to form the digital data base from the digitization information is contained in Appendix B.

3.5 Merging Different Category Matrices

Typically, three (3) sets of hand-drawn data base maps were constructed and passed through the digitization process described in the preceding sections. One (1) for distributed targets such as forests, fields, lakes, etc., one (1) for cultural features such as buildings, roads, runways, railways, etc., and one (1) for seasonal data such as snow depth, etc. At this point it is necessary to combine these three (3) matrices as output from the computer programs described in Section 3.4 with each other and with an elevation data matrix into a single meaningful data base containing all the information. This merging of data bases must be done carefully to insure that correct alignment between the data bases is accomplished. Further, while the snow and elevation information must be packed in with the cate-

gory information, the cultural information, i.e., cities, roads, railroads, and houses, actually replaces the category information.

With these requirements in mind, a three-program package was developed to accomplish the merging of these various data bases. The input to this sequence of programs is the complete category, snow, and elevation data bases, as well as the SORT output for the linear and point targets of the cultural map, and the BUILD output of the city areas. (For a description of SORT and BUILD, see Appendix B). Output from this sequence is the full rectangular data base, containing accurate category, snow, and elevation data for each resolution cell. The programs which produce this are:

(1) CATEGORY AND CULTURAL MERGE - which combines the category and cultural data bases

(2) ADD SNOW - which overlays the snow data base onto the output of (1), and also reassigned the categories of the data base

(3) ADD ELEVATION - which adds the elevation information to the data base produced in (2), the output of this program is the full rectangular data base

These programs are described in Appendix B.

4.0 RADAR SIMULATION: BACKSCATTER DATA AND AN EMPIRICAL MODEL FOR SNOW

The purpose of this chapter is to describe the radar reflectivity data utilized in radar simulation of the Watertown, New York, area "winter scenes". Two goals of the overall winter scene study, as set forth in the Introduction, are to artifically generate representative radar imagery (i.e., without the use of actual radar images) for a typical Watertown winter situation, and secondly, to evaluate the imagery. Of particular interest to the military community is an evaluation of the simulated imagery for terminal guidance. This imagery should be evaluated in the future to determine whether drastic seasonal differences occur between a representative winter scene and, for example, a "summer scene". Great disparity between the scenes implies a need for at least two, stored, reference images for year-round effectiveness in terminal guidance. However, if negligible discrepancies between seasonally different scenes are found to be consistently occurring, then the need for multiple reference scenes per target site is not well established. Nevertheless, it will be important in the future for the sake of profitably exploiting the potential of radar, both as a terminal guidance sensor and a means of in-flight guidance updating, to understand the seasonal variations of the radar images of terrain. Radar simulation is an excellent means of attaining this type of knowledge.

The backscatter data in radar image simulation serves as a link in the chain of events described in Chapter 1-3. Since real radar imagery is not used in the production of surface feature overlays or data bases, we need backscatter information in combination with photographic sources of intelligence to predict what the radar will "see" or what target types

the radar can distinguish, and how well it can do this. To avoid being cryptic, let us use an example. Photographic information shows us boundaries, field-to-field contrast, and so forth. What we would like to know, or predict with radar image simulation, is whether the radar can see similar boundaries, similar or reversed field-to-field contrast, etc. One situation in which photography (or any other intelligence source) fails to serve the needs of radar image simulation occurs when the camera fails to distinguish between two objects on the basis of tonal differences while the radar could have distinguished between them, given two equal resolution systems. Such occurrences cause "errors" in the data base, and are unavoidable in some instances without further knowledge to complement the photography. Radar backscatter data, together with photographic images, have served well the purpose of radar simulation and the images produced have been very similar to the actual radar imagery[5].

We have applied backscatter data to characterize "distributed" targets in simulations and have symbolically represented cultural targets as film saturations. Alternate approaches are being sought for cultural targets, as the concept of a backscatter coefficient for such targets is not felt to be valid. On the other hand, the backscatter coefficient σ^{o} is the only factor in the "radar range equation" which is determined by the terrain properties alone. Subsection 1.3.1 discusses the use of the radar equation for simulation, and mentions constraints on its use based on resolution cell size, the distribution and number of scatterers, and so on.

The backscatter coefficient is a measure of the reradiative ability of the terrain. Since σ^{o} is related to the power ratio of the received

---

[5]Holtzman, J.C., V.H. Kaupp, J.L. Abbott, V.S. Frost, E.E. Komp, and E.C. Davison, "Radar Image Simulation: Validation of the Point Scattering Model," Engineer Topographic Laboratories, United States Army, Fort Belvoir, Virginia, ETL-0117, June 1977.

signal to the transmitted signal, it contains no explicit phase information. However, the amplitude information in a plot of σ^o versus angle of incidence for a particular distributed target relays a good deal about the target. Several features of these plots are notable: the σ^o value at nadir, the shape of the curve, the rate of drop-off at large angles, obvious break points in the curve, and the total change in σ^o between zero degrees and near ninety degrees. These features of a σ^o curve have significance when obtained by a precise scatterometer. For example, such a plot can often tell us whether a surface scattering terrain under investigation is smooth or rough relative to the illuminating wavelength. In another situation it might indicate at what angular range a layered medium is making a transition between looking (to the radar) like a volume scatterer and a surface scatterer. Another interpretation of two σ^o curves plotted on the same scale might be an observation of difference in target moisture content, dielectric value, etc. Thus, the backscatter data plots contain information on the micro-scale (i.e., the order of wavelengths) compared to the terrain elevation and category sampling distances described in Chapters 2 and 3.

We can turn these observations about backscatter around to use them as guidelines for analysis of newly gathered σ^o data or for synthesis of backscatter data for currently unmeasured targets we would like to simulate. Other techniques which have to be employed to fill the gaps in the σ^o data catalog include frequency extrapolation or interpolation. Many such gaps exist in the σ^o data catalog when we begin to deal with the backscatter from snow, from snow over vegetation, and other, similar layered structures. However, one empirical model being developed at the Remote Sensing Laboratory

by W.H. Stiles and F.T. Ulaby for snow over another medium characterized by a known backscatter function (of σ^o versus θ) has been investigated and employed in the winter scene simulations. The following subsection briefly describes recent empirical developments.

4.1 Backscatter Data for Winter Simulations

In the winter of 1975 the Remote Sensing Laboratory began to conduct experiments on snow backscatter measurement. Since that time the number of frequency/polarization combinations has been greatly increased to yield a fairly large quantity of backscatter measurements. Several documenting reports and publications have since followed[28,29,30]. In addition to the work being done at the University of Kansas, researchers at the Georgia Institute of Technology, the National Bureau of Standards and the Massachusetts Institute of Technology have investigated the behavior of active and

---

[28]Stiles, W.H., F.T. Ulaby, B.C. Hanson, and L.F. Dellwig, "Snow Backscatter in the 1-8 GHz Region," RSL Technical Report 177-61, University of Kansas Center for Research, Inc., Lawrence, Kansas 1976.

[29]Stiles, W.H., B.C. Hanson and F.T. Ulaby, "Microwave Remote Sensing of Snow: Experiment Description and Preliminary Results," RSL Technical Report 340-1, University of Kansas Center for Research, Inc., Lawrence, Kansas, June 1977.

[30]Ulaby, F.T., A.K. Fung, and W.H. Stiles, "Backscatter and Emission of Snow: Literature Review and Recommendations for Future Investigations," RSL Technical Report 369-1, University of Kansas Center for Research, Inc., Lawrence, Kansas, June 1978.

passive microwave systems in response to snow as the target[31,32,33,34,35,36].

Preliminary observations of snow backscatter data indicate that snow can behave as either a volume or surface scatterer, that it can appear transparent or opaque or as an attenuator to microwaves, and that these types of behavior are causally related to the "free water content"[37] of the snow, its temperature profile, depth, density profile, ionic impurities, the frequency, polarization, and angle of incidence of the sensor, etc. Signal fading has been observed[38] and it requires appropriate averaging of the data to be able to arrive at a mean σ^{o} value.

---

[31]Currie, N.C., F.B. Dyer and G.W. Ewell, "Radar Millimeter Backscatter Measurements from Snow," Final Report, Engineering Experiment Station, Georgia Tech., Atlanta, Georgia, January 1977.

[32]Tsang, L. and J.A. Kong, "The Brightness Temperature of a Half-Space Random Medium with Non-Uniform Temperature Profile," Radio Science 10(12): 1025-1033, December 1975.

[33]Tsang, L. and J.A. Kong, "Microwave Remote Sensing of a Two-Layer Random Medium, IEEE Trans. Ant. and Prop., 24(3):283-287, May 1967.

[34]Tsang, L. and J.A. Kong, "Theory for Thermal Microwave Emission from a Bounded Medium Containing Spherical Scatterers," J. Appl. Phys., 48(8):3593-3599, August 1976.

[35]Tsang, L. and J.A. Kong, "Wave Theory for Microwave Remote Sensing of a Half-Space Random Medium with Three-Dimensional Variations," Radio Science (to be published), 1978.

[36]Tsang, L. and J.A. Kong, "Radiative Transfer Theory for Active Remote Sensing of Half-Space Random Media," Radio Science (to be published).

[37]Ulaby, F.T. and W.H. Stiles, "Backscatter and Emissivity of Snow," Proceedings MIcrowave Remote Sensing Symposium, Houston, Texas, December 1977.

[38]Stiles, W.H. and F.T. Ulaby, "The Active and Passive Microwave Response to Snow Parameters, Part I: Wetness," RSL Technical Report 340-2, University of Kansas Center for Research, Inc., Lawrence, Kansas, October 1978.

Analysis of the Kansas data by W.H. Stiles for the situation of a layer of snow over an underlying frozen ground has generated an exponential empirical model for the backscatter coefficient. (Both the air-snow and the snow-ground interfaces were fairly smooth in the actual experiments). This model is reported in Stiles' dissertation and as Remote Sensing Laboratory TR 340-3. It will not be reported here as it was unpublished at the time of the writing of this report. All data resulting from the application of the empirical model are reported in Appendix A.

5.0 RESULTS

The Watertown simulation site is centered on the southwestern corner of the Freeman Bus Company garage on the northeastern side of Watertown, New York, and is characterized both by several city cultural complexes as well as by forested regions and agricultural field pattern areas. This geographic region represents a mix of cultural and distributed targets. Construction of the data base for this site is reported in Section 2 and simulation parameters for it are summarized there. Reflectivity data used in making simulations from this data base are presented in Appendix A.

Sample simulated PPI results have been produced for the Watertown site. These results have been generated via a digital computer from the guidance specialization discussed earlier (see Section 3 and Appendix C), and have been photographed from the image display of the VDI. Table 10 presents the parameters of the guidance radar as modeled. A sequence of simulated scenes representing four different altitudes of the PPI radar over each of the data base sites were produced and are presented here.

Figures 4 and 5 present sequences of four (4) simulated radar images produced from the Watertown target site. In the figures, the scenes labeled Band 1 represent the lowest altitude and Band 4 the highest. The Band 1 scene portrays a simulated radar image of an area consisting of approximately 250 square kilometers, and the Band 4 scene portrays an area of approximately 4,000 square kilometers. Table 11 summarizes the simulation characteristic of each scene.

In both figures, a sequence of images starting with Band 1 and ending with Band 4 is presented. Figure 4 presents radar data for the Watertown site in late fall to early winter, and Figure 5 in mid-winter with a significant accumulation of snow on the ground. These have been produced by

TABLE 10

GUIDANCE RADAR PARAMETERS AS MODELED FOR SIMULATION

Transmitting frequency: X-band

Polarization (transmit-receive): HH (horizontal-horizontal)

Resolution:

Band Number	Range Resolution	Azimuth Resolution
1	30.5 m (100 feet)	$1/2^{o}$
2	30.5 m (100 feet)	$1/2^{o}$
3	100 m (328 feet)	$1/2^{o}$
4	100 m (328 feet)	$1/2^{o}$

TABLE 11

SIMULATION CHARACTERISTICS

Transmitting frequency: X-band

Polarization (transmit-receive): HH (horizontal-horizontal)

Resolution:

Band Number	Range Resolution	Azimuth Resolution	Number of Resolution Cells
1	30.5 m (100 feet)	$1/2^{o}$	105, 120
2	30.5 m (100 feet)	$1/2^{o}$	210, 240
3	100 m (328 feet)	$1/2^{o}$	128, 160
4	100 m (328 feet)	$1/2^{o}$	255, 600

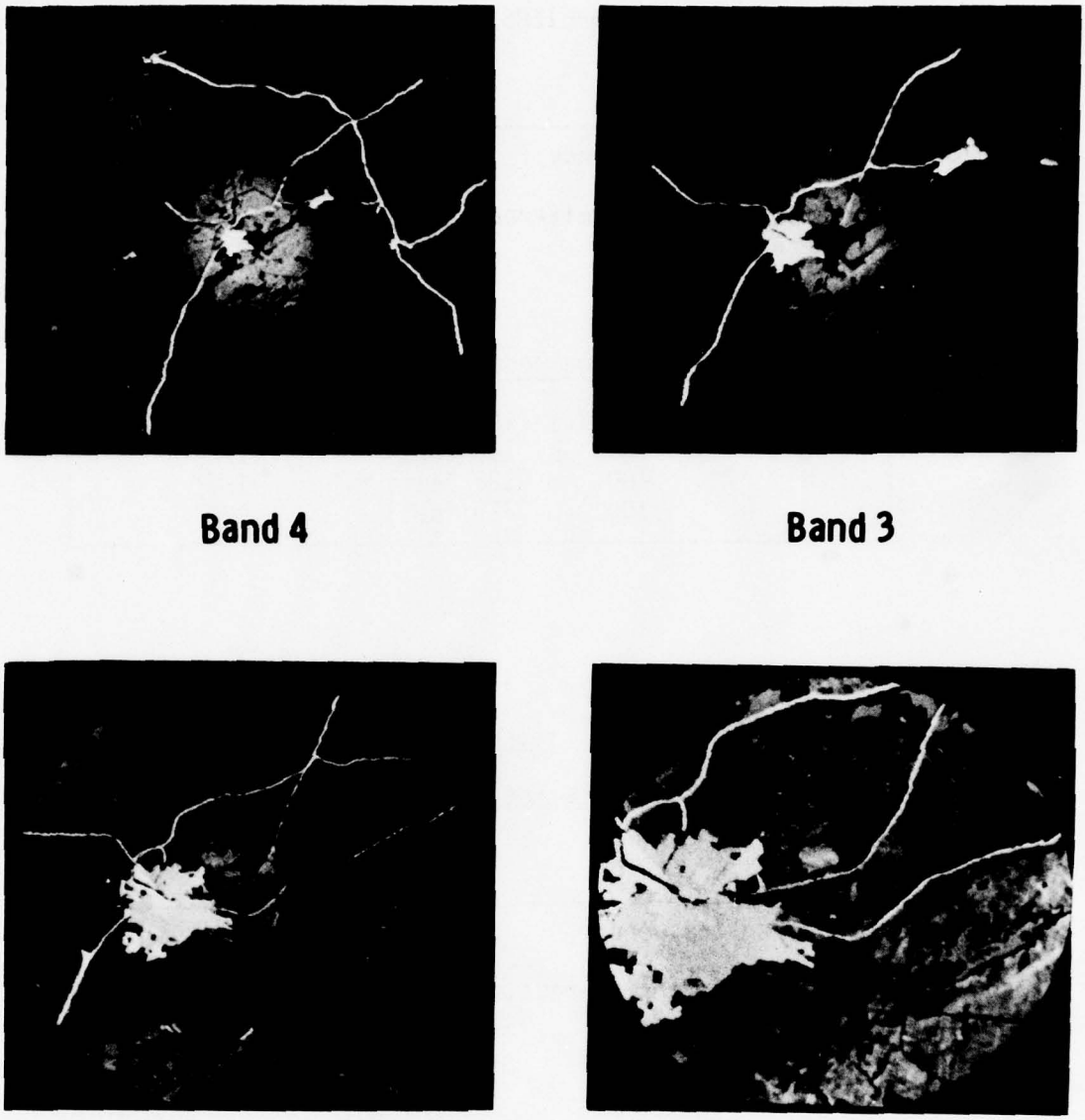


Figure 4. Radar Image Simulations of Watertown, New York, in late fall to early winter.

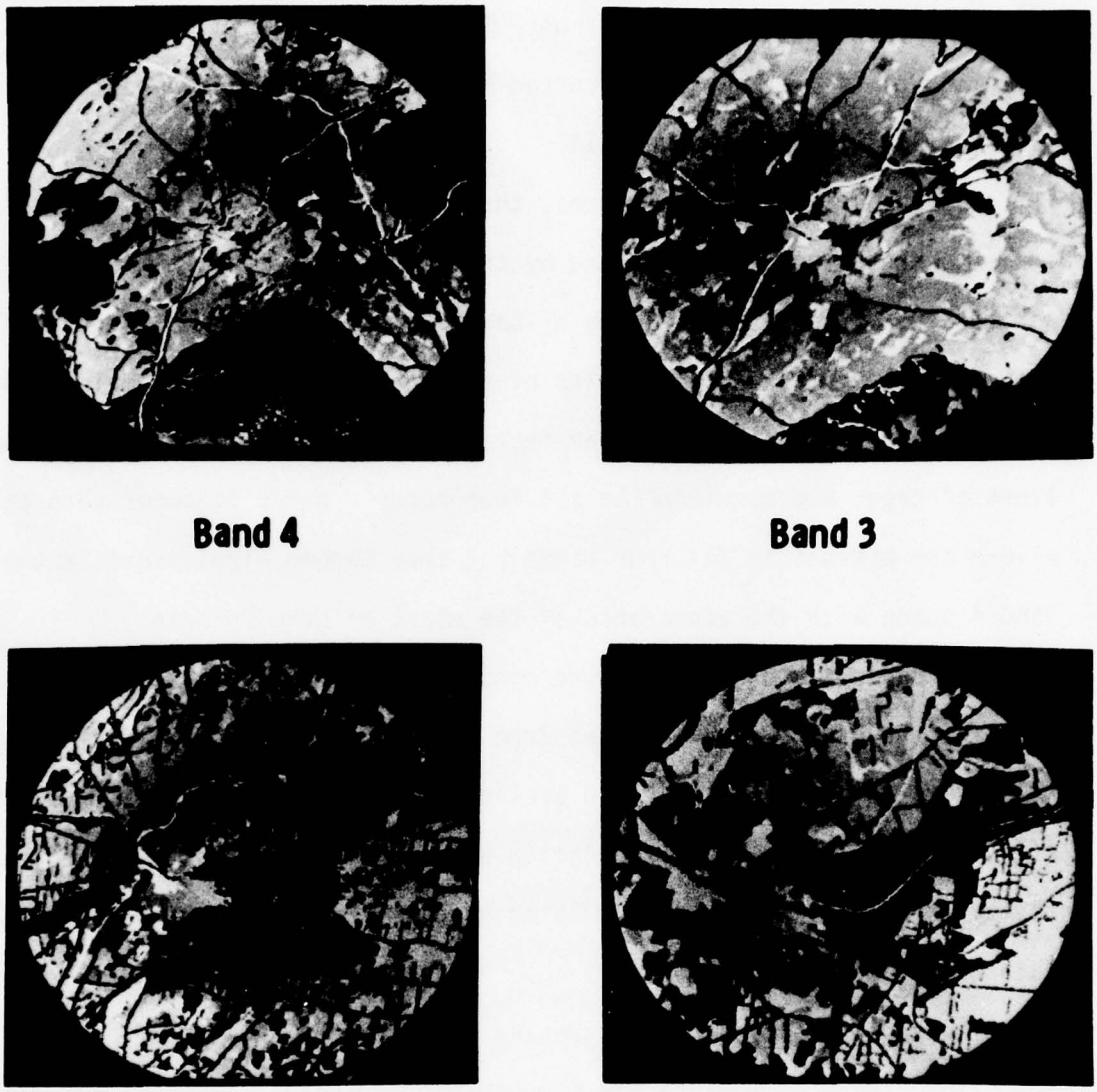


Figure 5. Radar Image Simulations of Watertown, New York in mid-winter

the computer software configuration for simulating the ballistic missile guidance system and the Correlatron. Thus, the geometry and parameters of these scenes is the same as discussed in Section 1.6 and Appendix C, and as summarized in Tables 10 and 11.

As can be seen in the figures, the characterization of the simulated images changes from one dominated by the city of Watertown with reasonable percentages of forest and fields of bare ground in the Band 1 image, to one dominated by forest and fields of bare ground with city complexes present in the Band 4 image. Linear features such as roads, railroads, and lines of trees are prominent in all four bands. Water features such as rivers are present in all four bands but they become significant in the Band 4 scene with the appearance of the coast of Lake Ontario.

The Bands 1 and 2 images were constructed from 1 1:24,000 scale data base and the Bands 3 and 4 scenes from a 1:100,000 one. Construction of these data bases is discussed in Section 2 and their characteristics and attributes for radar image simulation are summarized there. The reflectivity data used for making simulated guidance radar images from these data bases are listed in Appendix B.

The sequences of scenes presented in Figures 4 and 5 were prepared to test radar image simulation via the PSM for a more complex site and for a different season than earlier work[5,6,7]. Unfortunately, the flight test

---

[5]Holtzman, J.C., V.H. Kaupp, J.L. Abbott, V.S. Frost, E.E. Komp, and E.C. Davison, "Radar Image Simulation: Validation of the Point Scattering Model," Engineer Topographic Laboratories, United States Army, Fort Belvoir, Virginia, ETL-0117, June 1977.

[6]Holtzman, J.C., V.H. Kaupp, J.L. Abbott, V.S. Frost, E.E. Komp, and E.C. Davison, "Radar Image Simulation: Validation of the Point Scattering Method," Volume II, ETL-0118, Remote Sensing Laboratory Technical Report, RSL TR 319-28, University of Kansas Center for Research, Inc., Lawrence, Kansas, September 1977.

[7]Holtzman, J.C., J.L. Abbott, V.H. Kaupp, E.E. Komp, E.C. Davison, and V.S. Frost, "Radar Image Simulation: Validation of the Point Scattering Method," Addendum, ETL-0155, Remote Sensing Laboratory Technical Report, RSL TR 319-31, University of Kansas Center for Research, Inc., Lawrence, Kansas, June 1978.

program, scheduled to be flown in February 1978, has not yet been conducted and, thus, the actual data for comparison analyses have not yet been collected. The simulated radar images therefore can only be shown in the figures for information as correlation results have not been produced, though they may be produced some time after the flight tests planned for the 1979-1980 winter.

The winter scene sequence shown in Figure 5 was developed utilizing a unique data base construction philosophy discussed in Section 2, and an empirically derived model for the effects of snow covering an underlying reflectivity category [to be published in RSL TR 340-3, by Stiles, et al. 1979]. As can be seen by comparing the scenes having snow to those without snow, the major effect from this model is an average brightening. But also, new boundaries are created and some old ones are diminished, effects important in the correlation process to be used (i.e., the Correlatron).

On a qualitative basis, the sequences portrayed in both figures 4 and 5 "look" appropriate for the scenes and seasons they represent; fall and winter, respectively. Two major artifacts are immediately obvious in all three figures: (1) lack of fading and (2) simulation of cultural targets. Because these sequences were produced for testing via a Correlatron, fading was not introduced into the simulated images as fading would have been an additional source of "noise" thereby lowering the correlation peak and broadening it. Because these sequences were to be as omni-directional as possible, cultural targets were simulated symbolically. The simulation process for cultural targets was one of marking the presence, shape, size, and location of each individual target or complexes of them. They were all given a maximum return value with no attempt to actually predict their return with the exception of cities. Cities were subdivided into four (4) kinds of categories: (1) mostly structures, (2) mostly trees, (3) mixed

structures and trees, (4) open land. The first three (3) of these categories were treated symbolically and the fourth was simulated as a normal distributed target.

6.0 CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

It is believed that development of the PSM model has resulted in a very advanced simulation system which has expanded the frontiers in radar simulation. The guidance radar simulation work reported here represents one "real-world" application of the model. The following items represent the major accomplishments of this simulation project:

(A) The generation of winter situation reference scenes and development of methods for constructing data bases.

  (1) The necessary techniques have been developed to extract feature information from various kinds of source intelligence for constructing data bases.

  (2) Specialized computer programs have been developed to process hand-drawn feature maps into data bases.

  (3) An approach has been developed for adding a dimension to data bases whereby the radar images corresponding to different seasons can be simulated.

  (4) Data bases have been developed from manual photo-interpretation techniques. Interactive feature extraction techniques would speed-up construction of data bases and is judged to represent a potential reduction in the costs of making data bases.

(B) The need for empirical backscatter models and backscatter data has been identified as the radar scenes show seasonal changes for geographic locations receiving considerable, medium to high free water content snow.

(1) Too little backscatter data are available for the kinds of reflectivity categories in the terrestrial envelope of scenes for which radar simulations are potentially desired.

(2) The numbers of variations in conditions in the ground (i.e., seasonal and meteorological variations) with variations in radar parameters (i.e., frequency, polarization, etc.) are too many for measuring all permutations. Theories must be developed for extending and extrapolating empirical backscatter data for conditions not measured.

(C) The PSM has been demonstrated to be a high-quality general methodology for the simulation of radar imagery. This judgment is based upon the qualitative analysis of the simulations and similar actual imagery.

(1) The necessary mathematical models, software implementations, and techniques have been developed to simulate radar images representative either of PPI or SLAR (real or synthetic aperture).

(2) Radar propagational phenomena such as layover and shadow as well as geometrical relationships between radar and ground are accurately treated.

(3) The general PSM implementations are capable of simulating the radar return from complex terrain having significant relief.

(4) The general PSM implementations are capable of simulating the radar return from a scene for a radar having a specified resolution and averaging of independent samples (within the limitation imposed by the data base).

(5) The general PSM implementations are capable of simulating the radar return as processed via any desired antenna to receiver to image system.

(6) The general PSM implementations are capable of producing a simulated radar images portraying a desired subset of the dynamic range of a scene.

(7) The general PSM simulation implementations should be specialized as much as possible for each unique application to increase efficiency and to reduce costs.

6.2 Recommendations

The radar simulation project concluded and reported here represents a significant advance in simulation technology for radars. Although the work completed has resulted in an advanced simulation methodology, additional work is contemplated for refining various aspects. Several recommendations follow for this additional work and refinements.

6.2.1 Develop an Interactive Feature Extraction System

A major obstacle to increased use of radar image simulation is the construction of data bases which are suitable for desired applications. The chief problem encountered in constructing data bases is in feature extraction. Feature extraction is the process of identifying the geometry and category (i.e., electromagnetic reflectance) properties of the scene and transferring them to the data base. It is recommended that an interactive feature extraction system be developed. Potential benefits which might accrue from such a system span many scientific disciplines. Obviously, radar image simulation would be served. Not so obvious are the sciences such as geology and geography which rely upon manual interpretation of imagery for results. In addition, the general field of image processing would be aided by development of an interactive capability such as the system recommended.

Classical techniques for feature extraction are manual techniques. Typically, a photo-interpreter scans the intelligence data and draws upon his interpretation experience to decide what information to transfer manually to the data base under construction. These decisions are made with as few digital computer image enhancement techniques as possible. This reticence to use available enhancement routines is caused, in part, by the very nature of the automated routines. They are not generally applicable to any but specific, well-structured, test cases. In addition, use of these techniques requires that the interpreter also be a computer expert. Moreover, the interpreter loses control and visibility of what he is trying to accomplish when he enters the computer world of automated land-use classification, or pattern recognition, or region definition, or etc. These reasons have serious ramifications for feature extraction, and, consequently, data base construction; they cost money. They cost money in the sense that it takes a much longer time to extract the features for a data base than might otherwise be necessary; data are manipulated by hand and the best information may not be obtained.

Clearly, a tremendous improvement of the product developed, resources expended, and time required could be obtained if a workable marriage between computer and interpreter could be arranged. The computer is very good at manipulating vast amounts of data in short periods of time; the human is not. The human is beyond comparison when it comes to drawing upon learning experience to make decisions. The computer excels at clearly defined repetitive tasks, at statistical analyses, at image enhancements. A cooperative approach in which the human is used to make decisions and guide the processing direction of the software, and the computer is used to manipulate the

data rapidly and easily and to remove the drudge from the human would be optimal--optimal in the sense of maximizing the return for resources expanded and minimizing the time and effort. This cooperative approach is called interactive feature extraction.

Interactive feature extraction requirements have been surveyed and a design philosophy has been developed for constructing an interactive feature extraction system. In this design philosophy, the computer is used to display, enhance, manipulate, and otherwise aid the interpreter as he performs his function. And the human is used to make decisions and to guide the computer in real-time as the programs run. Depending upon the level of sophistication of the interactive software, and the computer and display complex, tremendous savings of resources and improvements in efficiency and quality of the finished product are visualized.

6.2.2 Develop Theoretical Scattering Models

The recommendation is made for conducting theoretical electromagnetic scattering studies aimed at providing solutions to specific problems.

The increasing applications for radar image simulation require ever larger catalogues of backscatter data. All radar image simulations must use some model for the reflectivity properties (backscatter) of the objects in the scene. These data are required by the simulation model to produce the greyscale data in the simulated image. It is not reasonable to expect to measure and record the backscatter data for all possible permutations and combinations of the variables: frequency, polarization, categories, seasonal changes, and etc. Theoretical models must be developed both to extend and extrapolate the measured data to cases which have not been measured, as well as to predict the scattering response theoretically.

6.2.3 Compile a Comprehensive Listing of Snow and Ice Backscatter Data

A key component in transforming the data base into a radar image has been shown to be the backscatter data, the microwave reflectivity model.

It is recommended that a comprehensive listing of snow and ice backscatter data be compiled. The result of this effort should include both empirical backscatter data and theoretical models. This study would require a brief literature search to gather available empirical data and theoretical backscatter models. The theoretical models should be evaluated to determine their applicability to the radar image simulation problem and they should be examined to determine techniques to extend and extrapolate available empirical data across frequencies, polarizations, and depression angles.

6.2.4 Conduct an Empirical Backscatter Program

The question of whether an aseasonal radar simulation of target sites such as Watertown, New York, can be applied to a summer guidance situation has been addressed implicitly. The answer to this question is no, probably, for the choice radar frequency, polarization, etc. simulated in this project. This question could alternately be answered by appeal to backscatter data measurements. The Army Engineer Topographic Laboratories is currently supporting the collection of snow backscatter data through the University of Kansas Remote Sensing Laboratory. Also being investigated are snow covered grass, concrete, asphalt, and trees.

6.2.5 Perform a Sensitivity Analysis

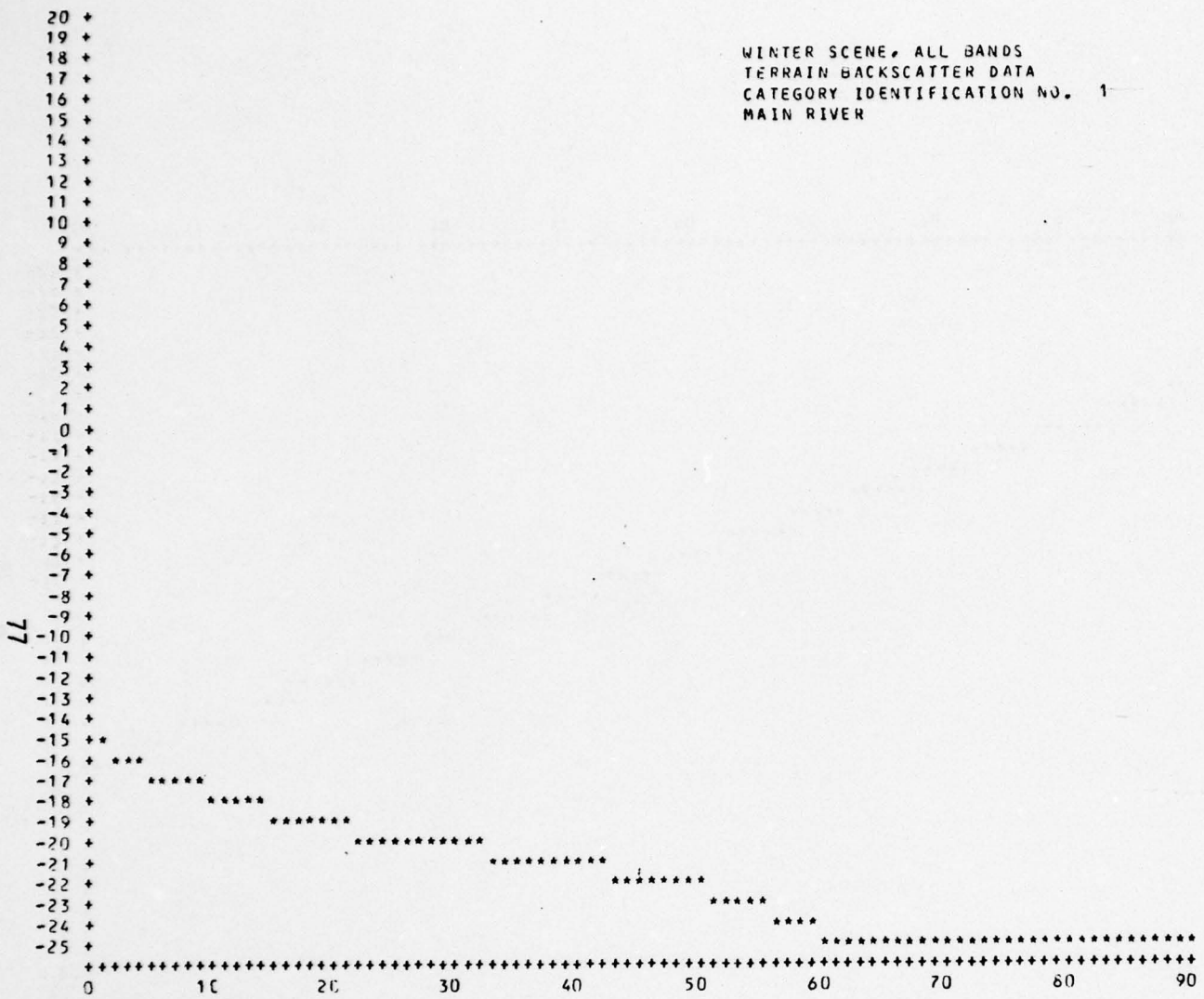
The utility and versatility of radar image simulation can be improved and the cost reduced if the minimum level of detail required to be in the data base for specific applications of radar image simulation can be deter-

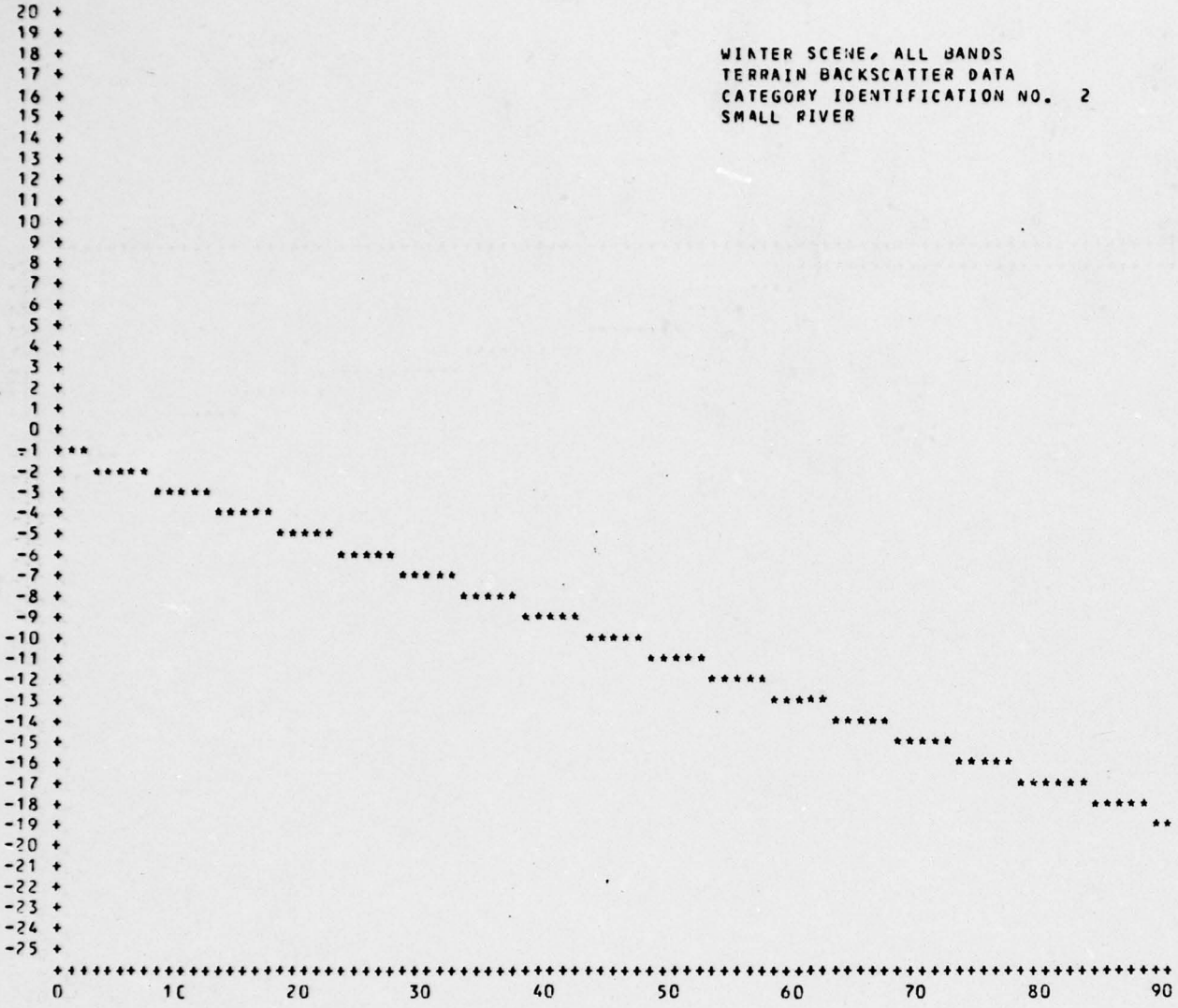
mined. As previously noted, the most expensive part of the radar simulation process is the building of the digital data base. If it can be determined that, for a specific application, the level of detail in the data base can be reduced, this translates directly into savings of time and money. It is recommended that such an analysis be conducted for the applications of radar image simulation most often used, or for any projected high use application.

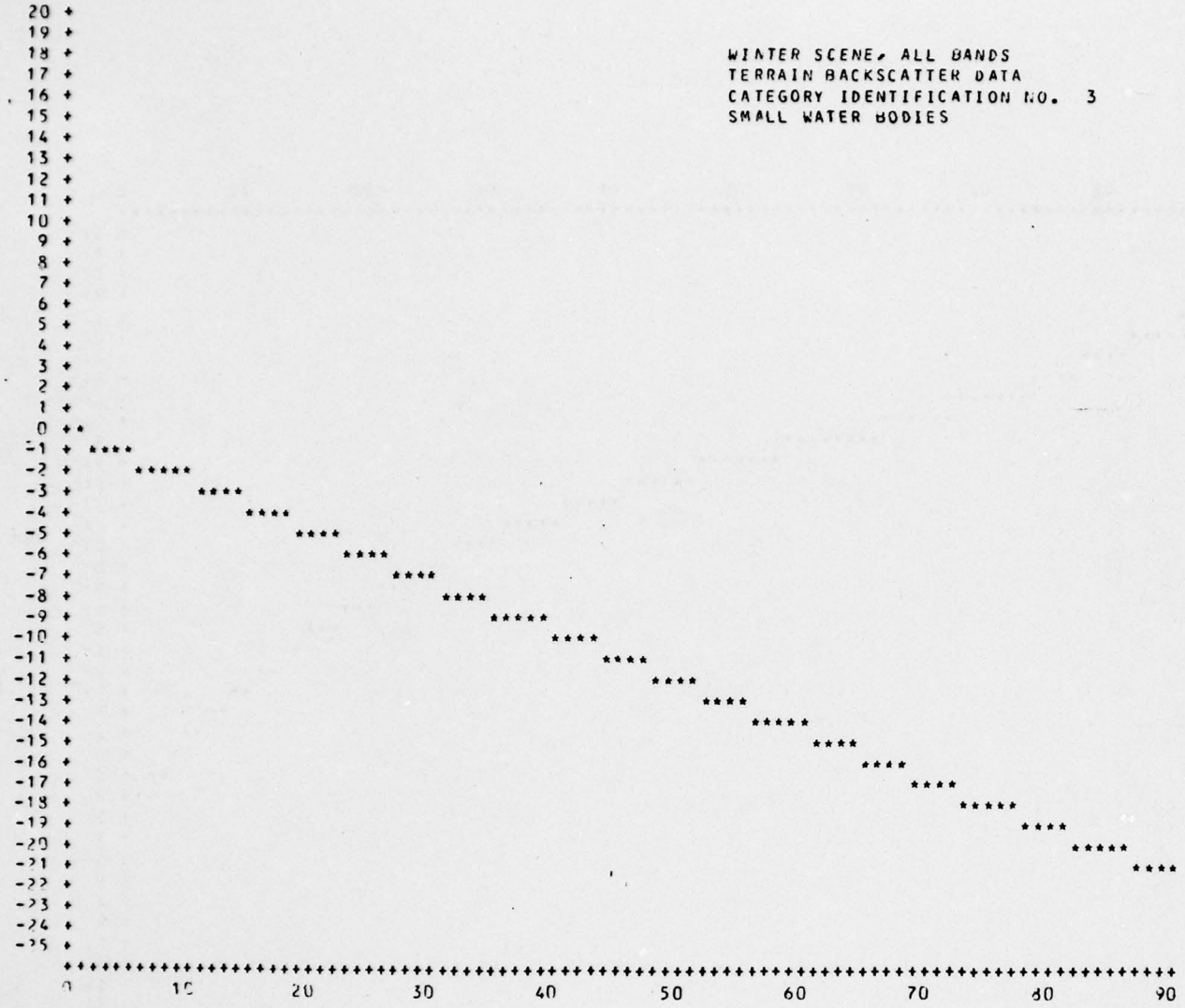
The radar image quality metrics being developed under contract to the Army Research Ovvice represent a potential quantitative a-proach for accomplishing this objective. They present a way to relate the "goodness" criteria of an application to the simulation data base via predictive, mathematical expressions. A possible study format would be to produce a sequence of simulated radar images having step-wise degraded parameters and evaluating them as to their "goodness" for satisfying the application. This predictive expression could then be used for defining the level of detail required to be built into a data base for that application.

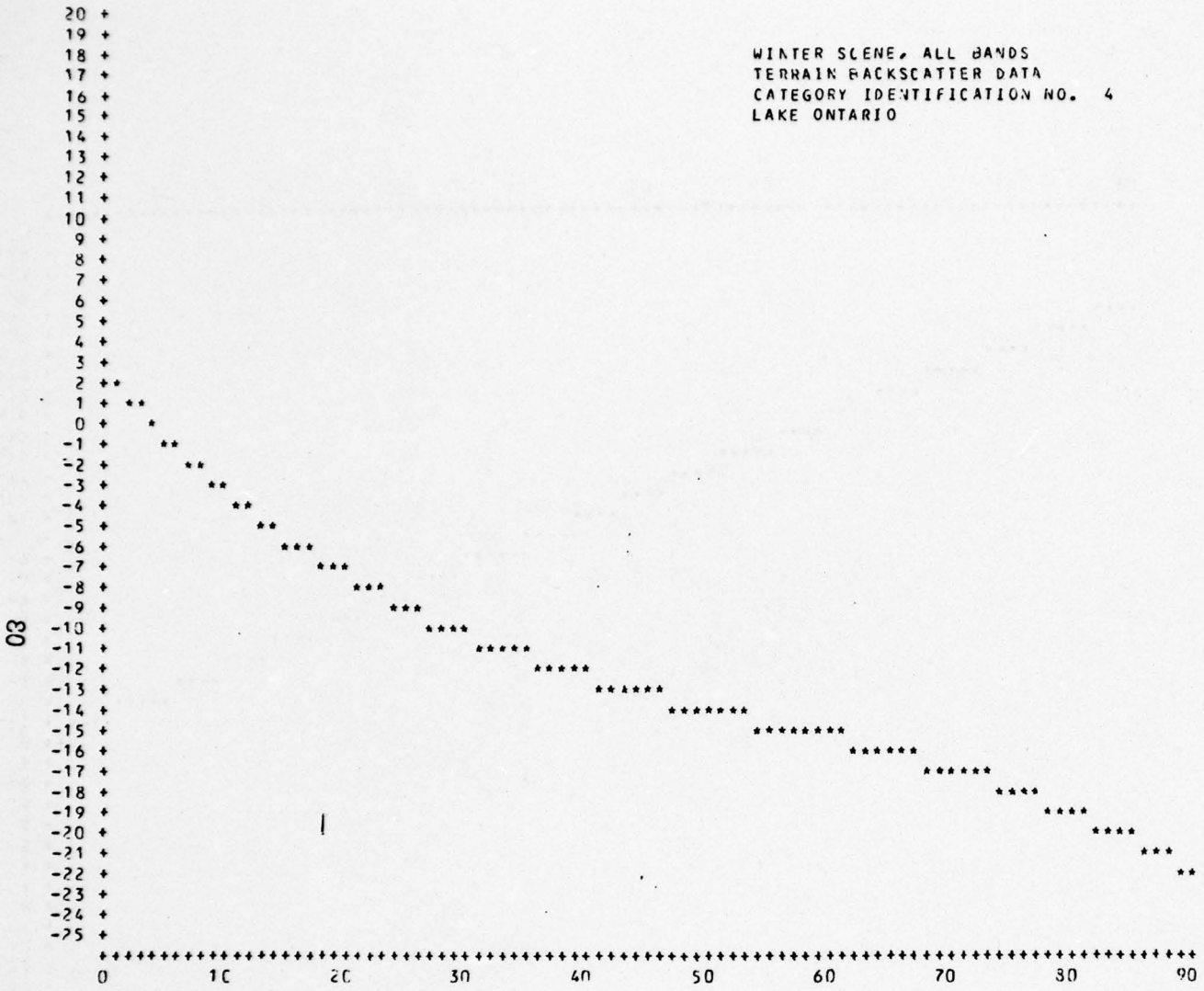
APPENDIX A

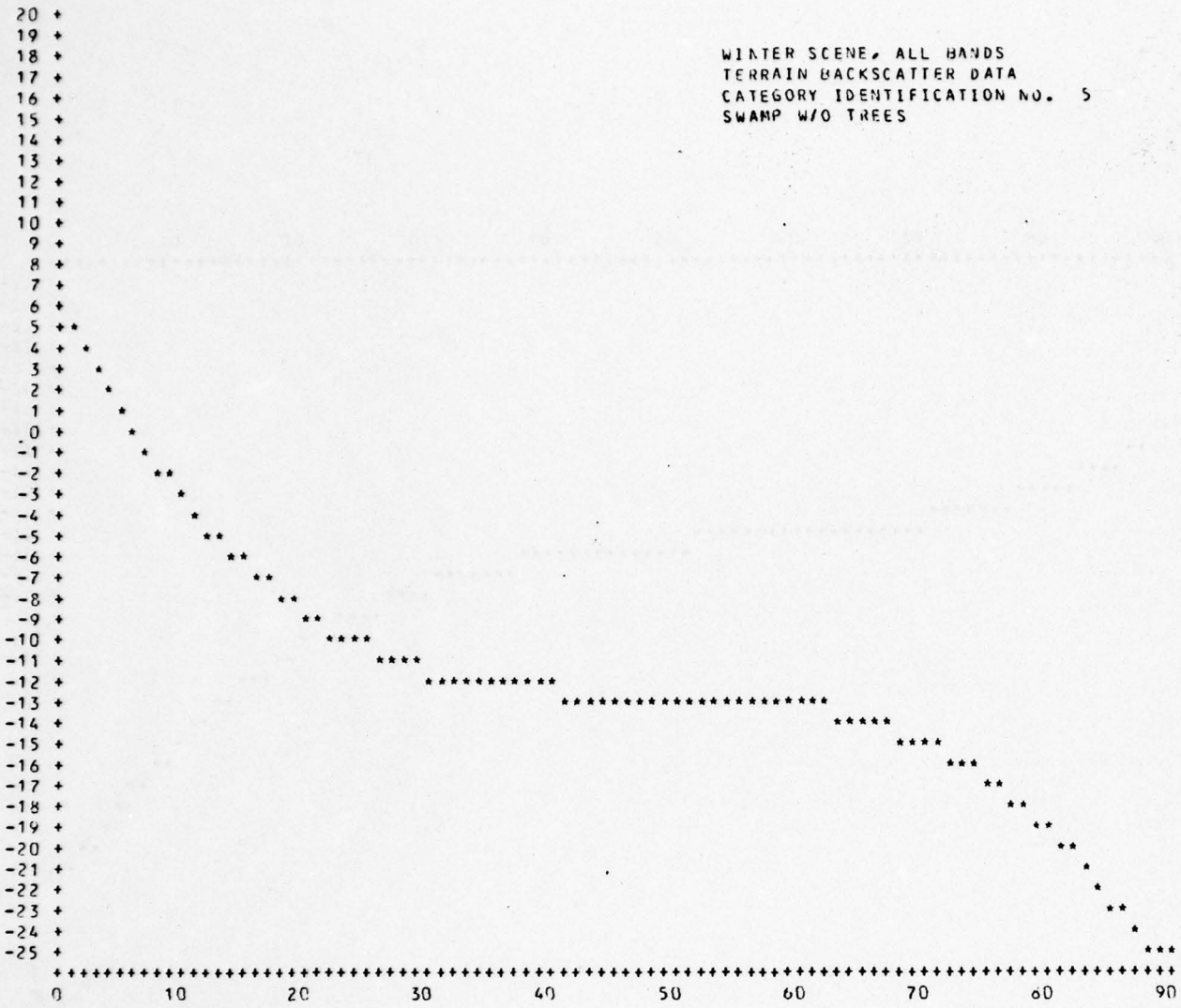
RADAR REFLECTIVITY DATA

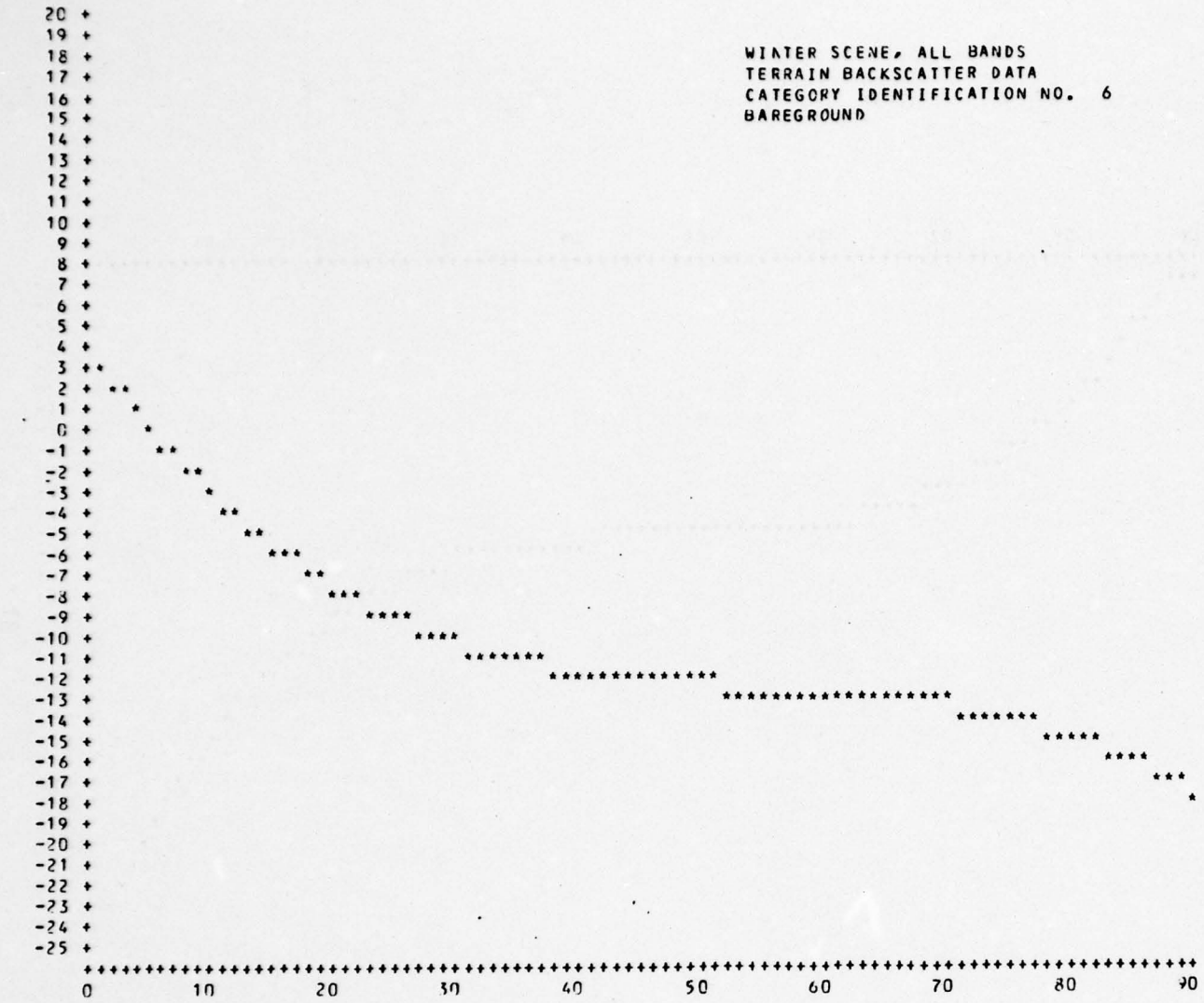

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 1
MAIN RIVER
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

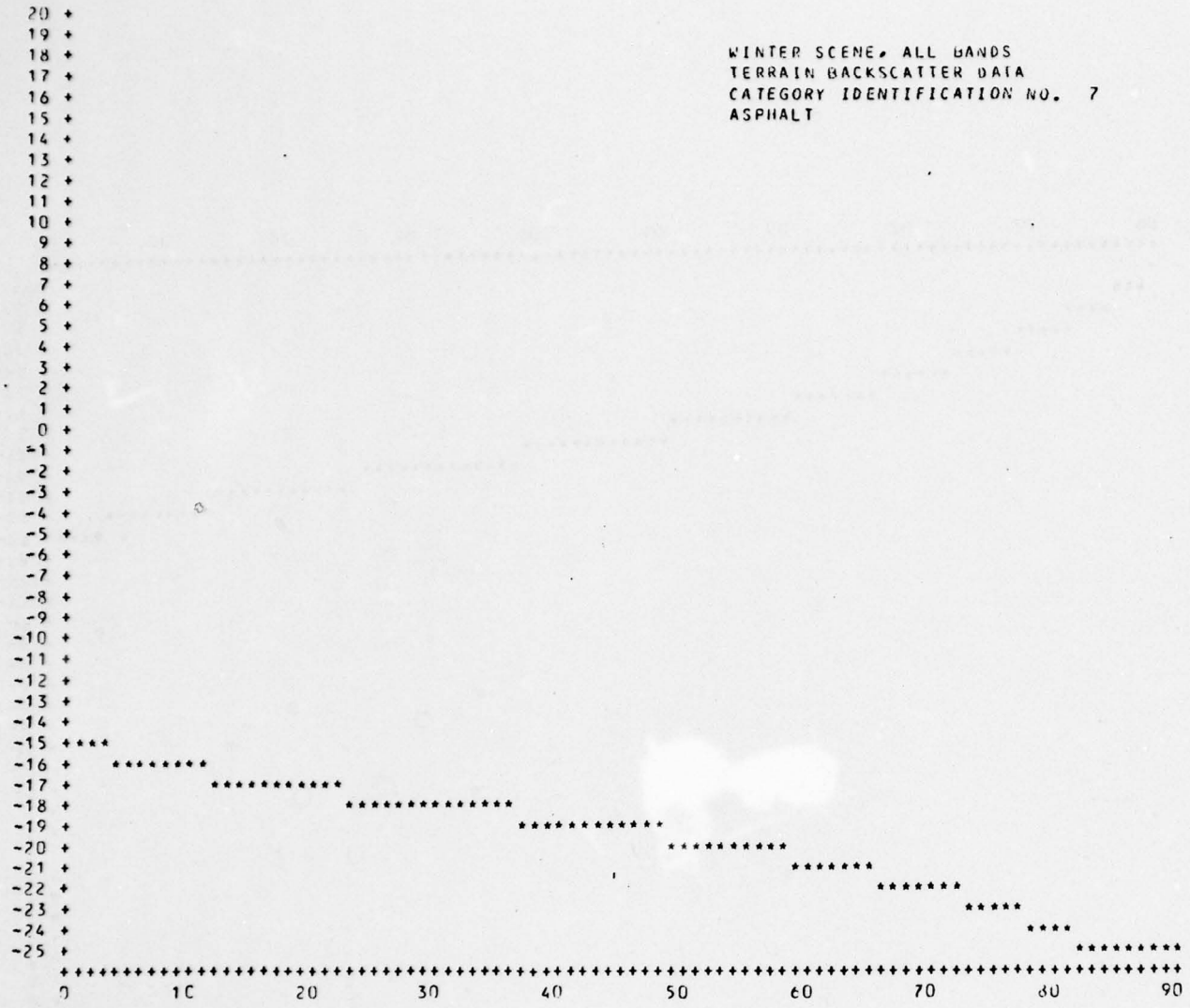

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 2
SMALL RIVER
20 +
19 +
18 +
17 +
16 +
15 +
14 +
13 +
12 +
11 +
10 +
9 +
8 +
7 +
6 +
5 +
4 +
3 +
2 +
1 +
0 +
-1 +**
-2 + *****
-3 + *****
-4 + *****
-5 + *****
-6 + *****
-7 + *****
-8 + *****
-9 + *****
-10 + *****
-11 + *****
-12 + *****
-13 + *****
-14 + *****
-15 + *****
-16 + *****
-17 + ******
-18 + *****
-19 + **
-20 +
-21 +
-22 +
-23 +
-24 +
-25 +
0
10
20
30
40
50
60
70
80
90


WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 3
SMALL WATER BODIES
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

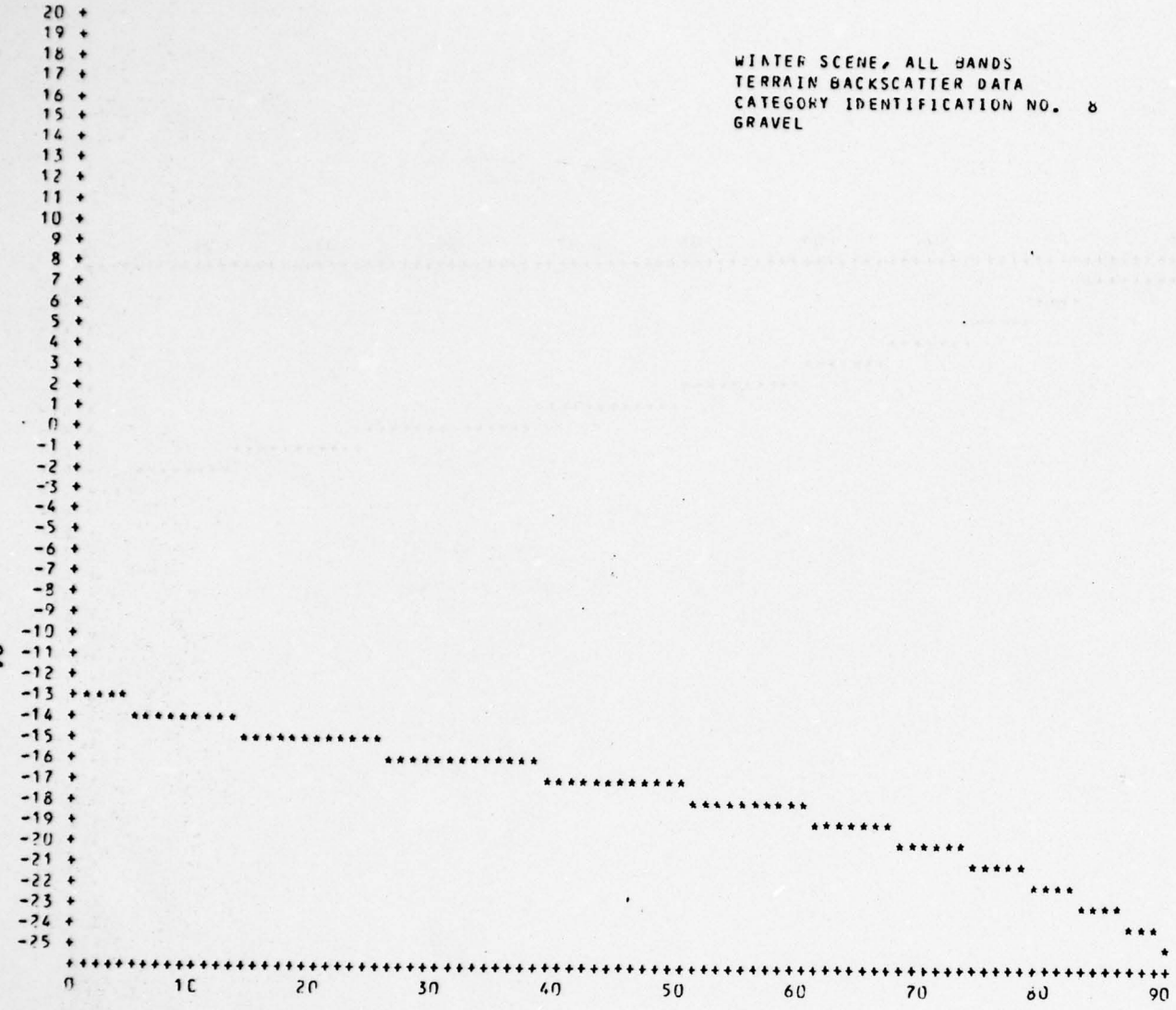

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 4
LAKE ONTARIO
dB
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

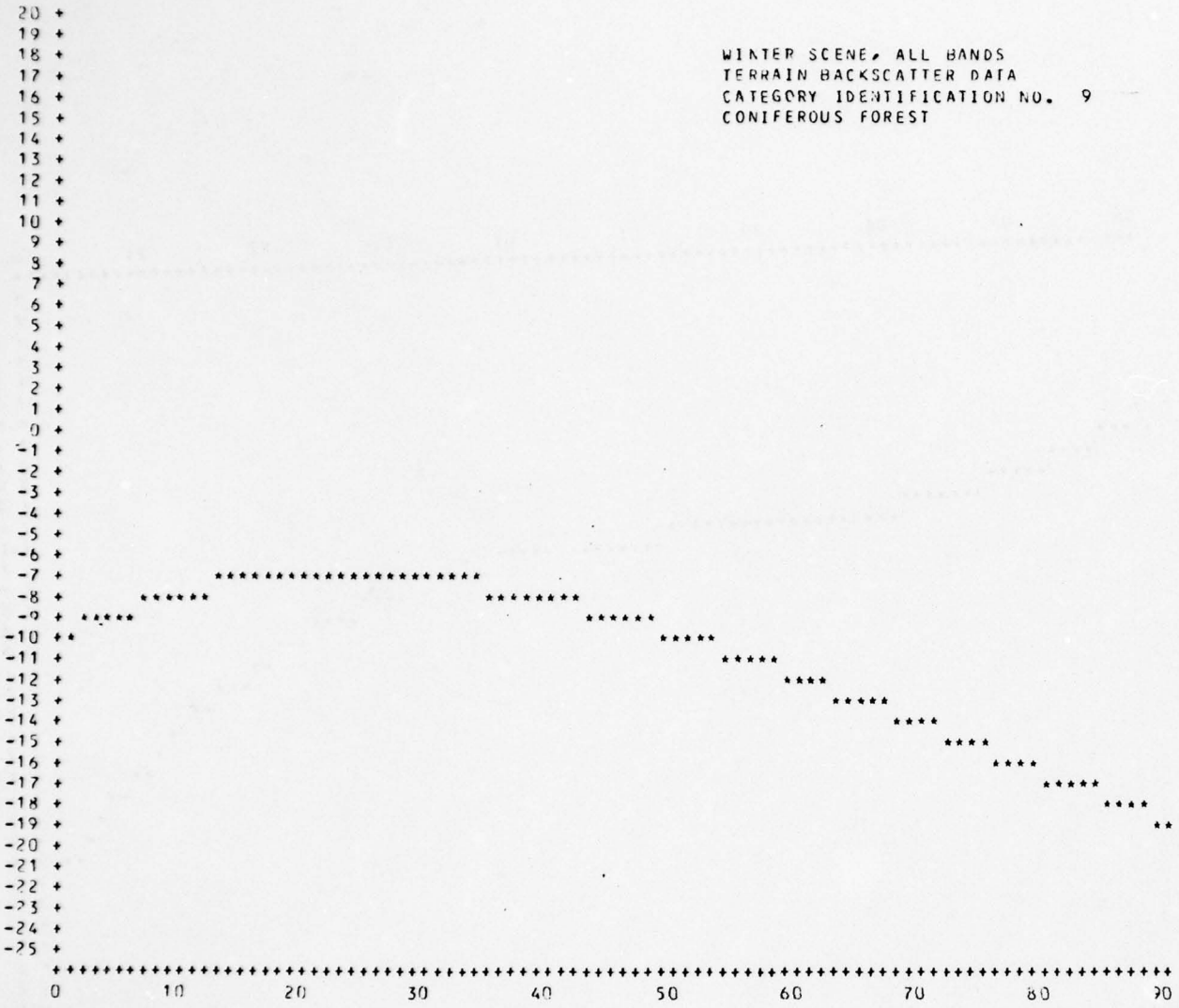

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 5
SWAMP W/O TREES
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90


WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 6
BAREGROUND
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90


WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 7
ASPHALT
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

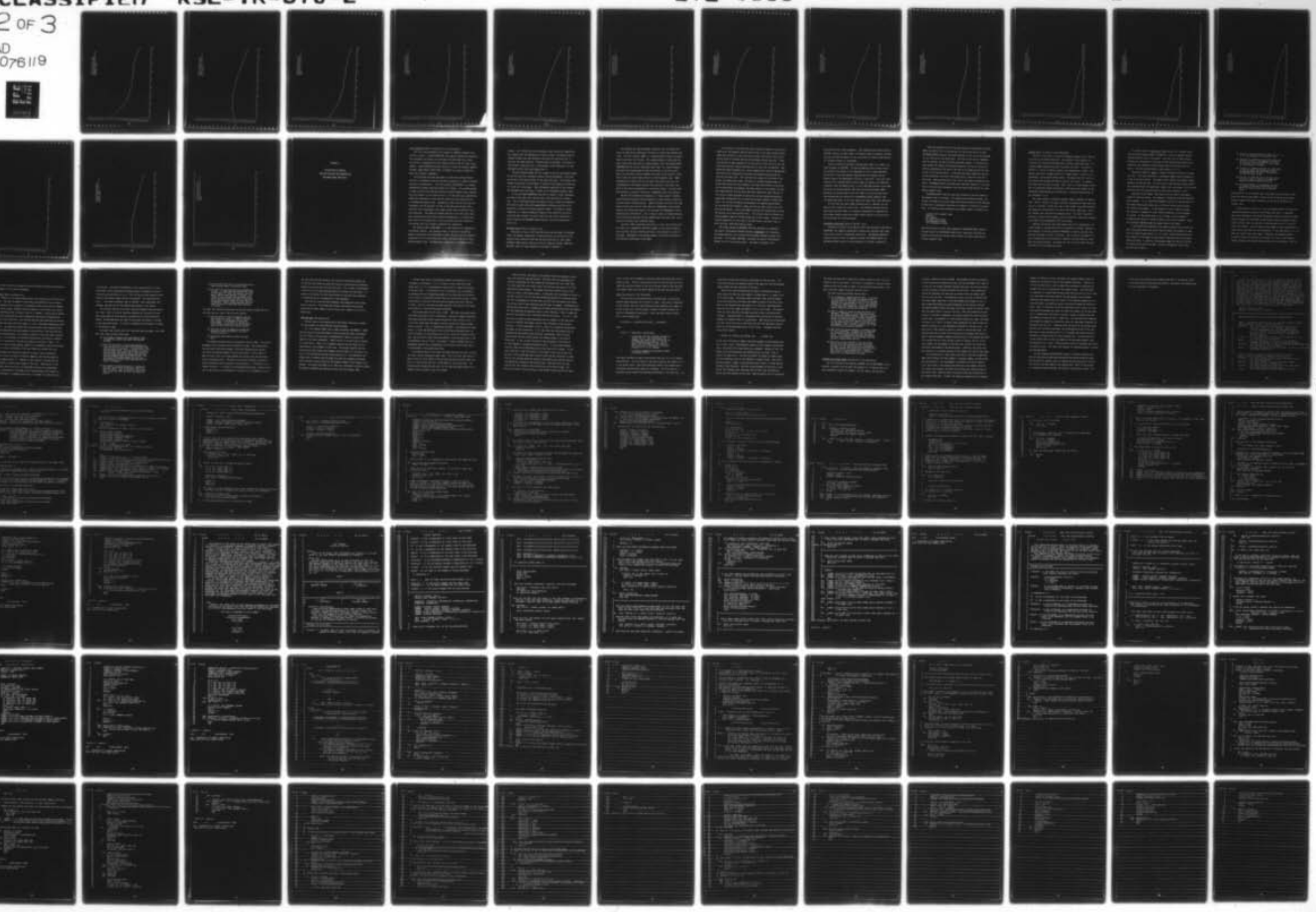
WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 8
GRAVEL

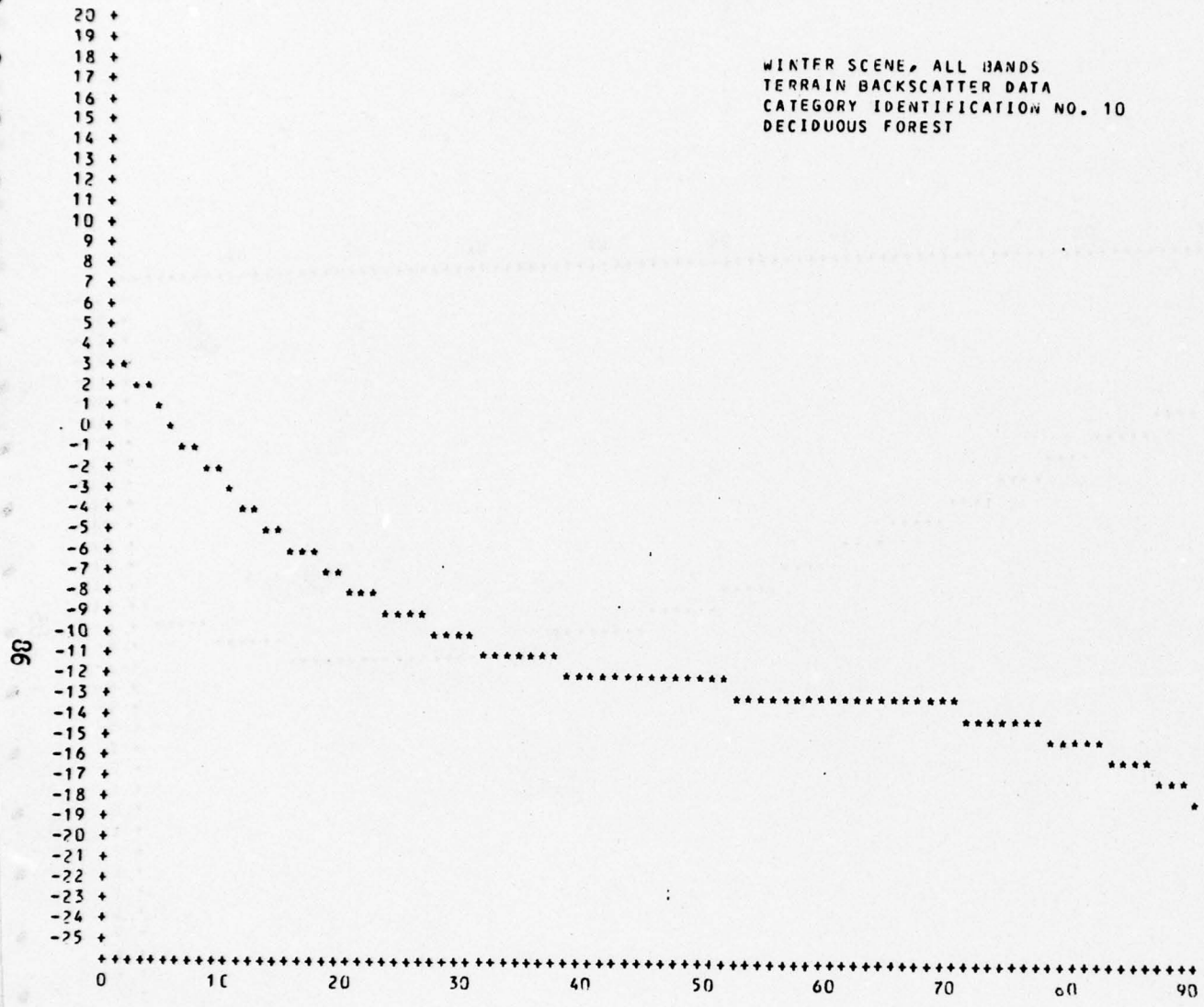


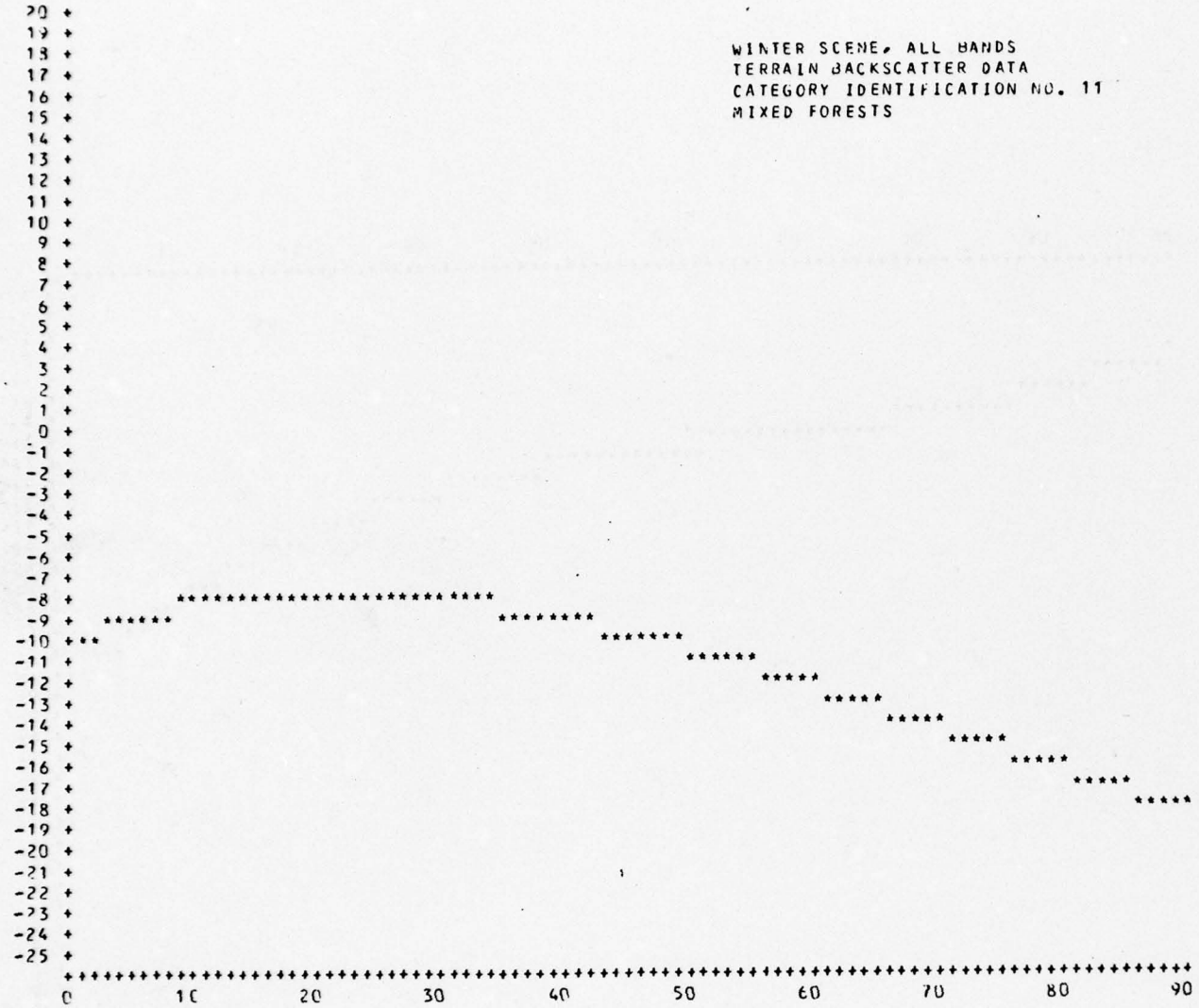

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 9
CONIFEROUS FOREST
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15 -16 -17 -18 -19 -20 -21 -22 -23 -24 -25
0 10 20 30 40 50 60 70 80 90

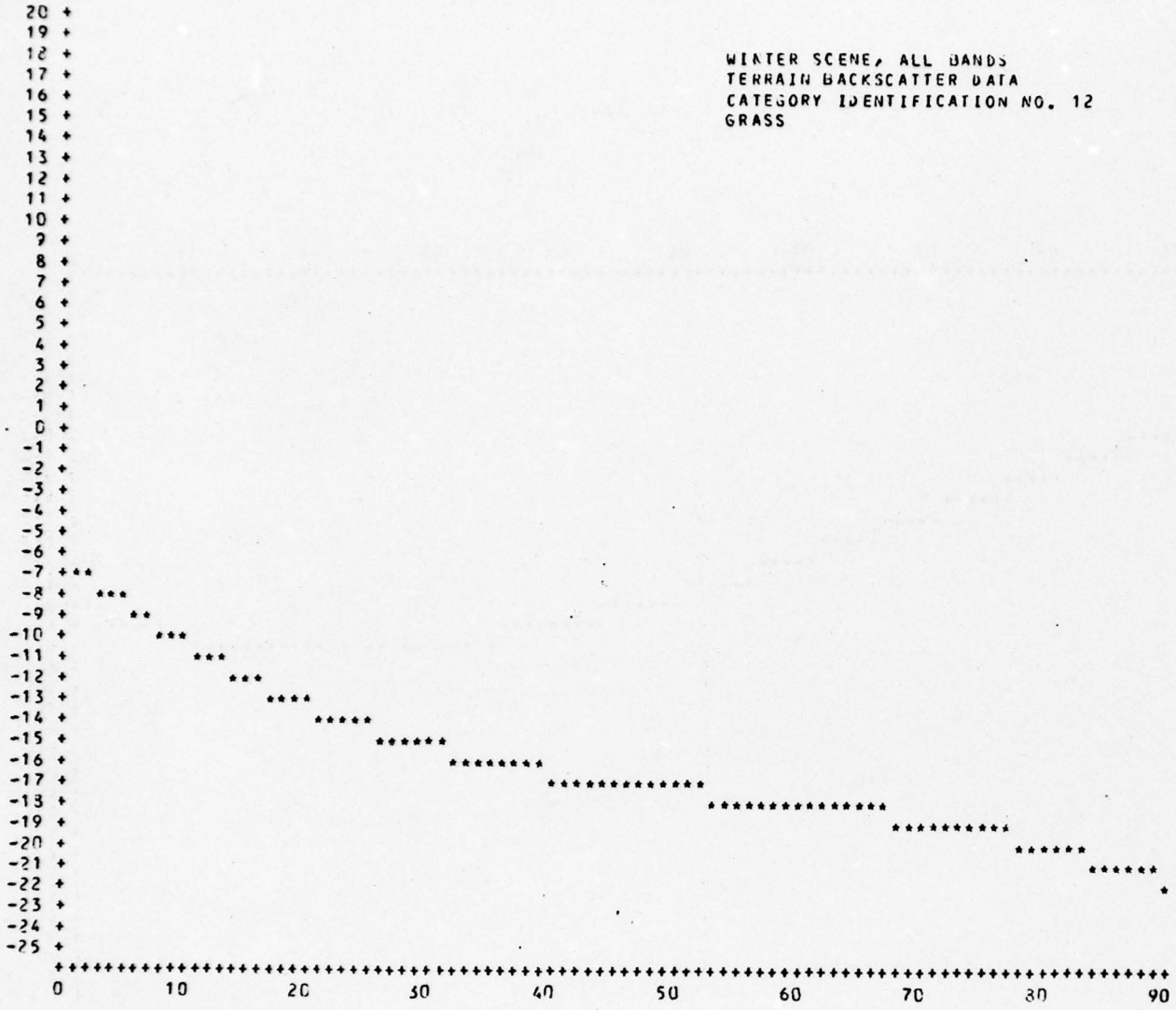
AD-A076 119 KANSAS UNIV/CENTER FOR RESEARCH INC LAWRENCE REMOTE --ETC F/G 17/9
RADAR IMAGE SIMULATION OF SEASONALLY DEPENDENT REFERENCE SCENES--ETC(U
APR 79 J C HOLTZMAN , J E BARE , V H KAUPP DAAK70-78-C-0062
UNCLASSIFIED RSL-TR-370-2 ETL-0188 NL

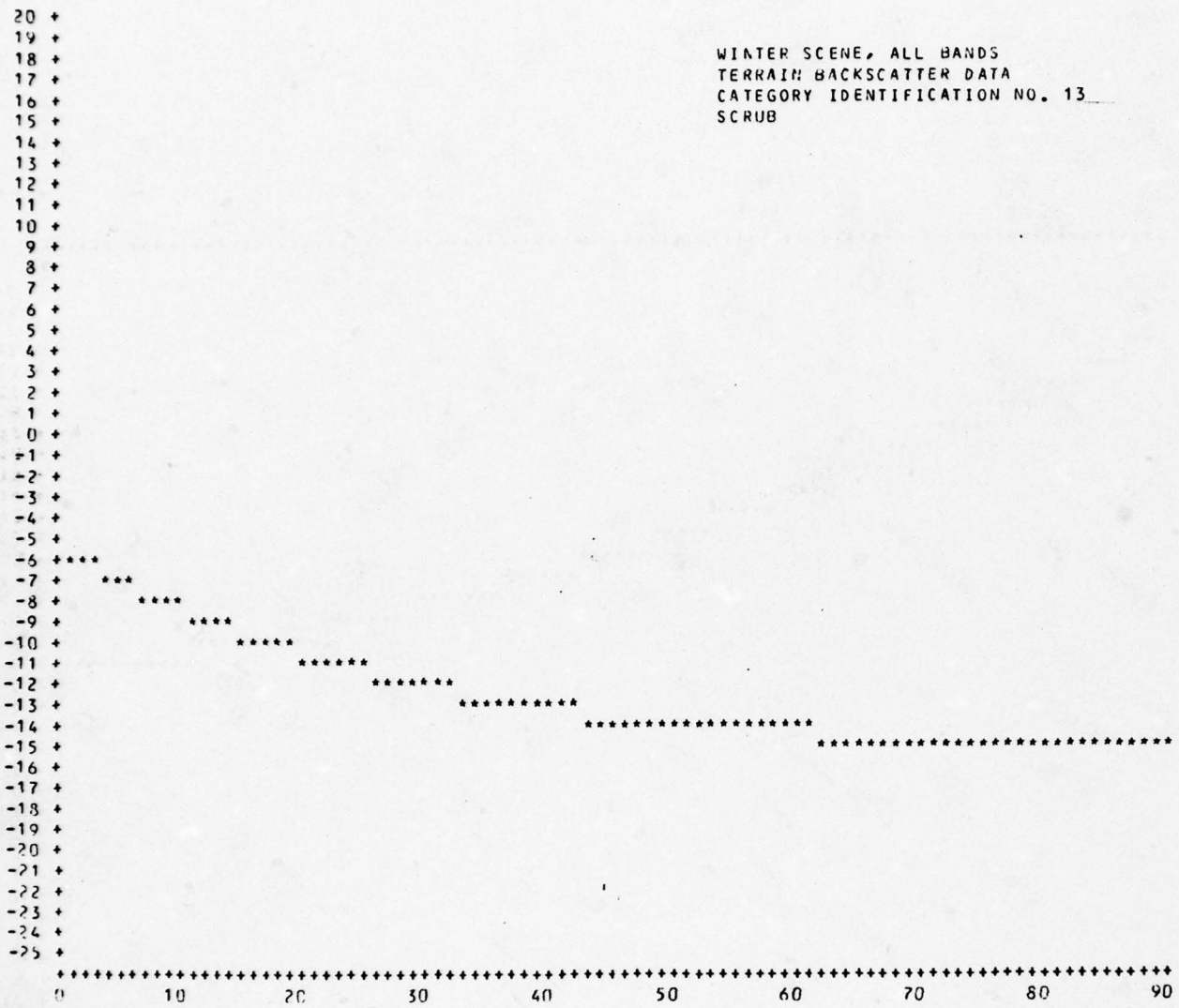
2 OF 3
AD
A076119

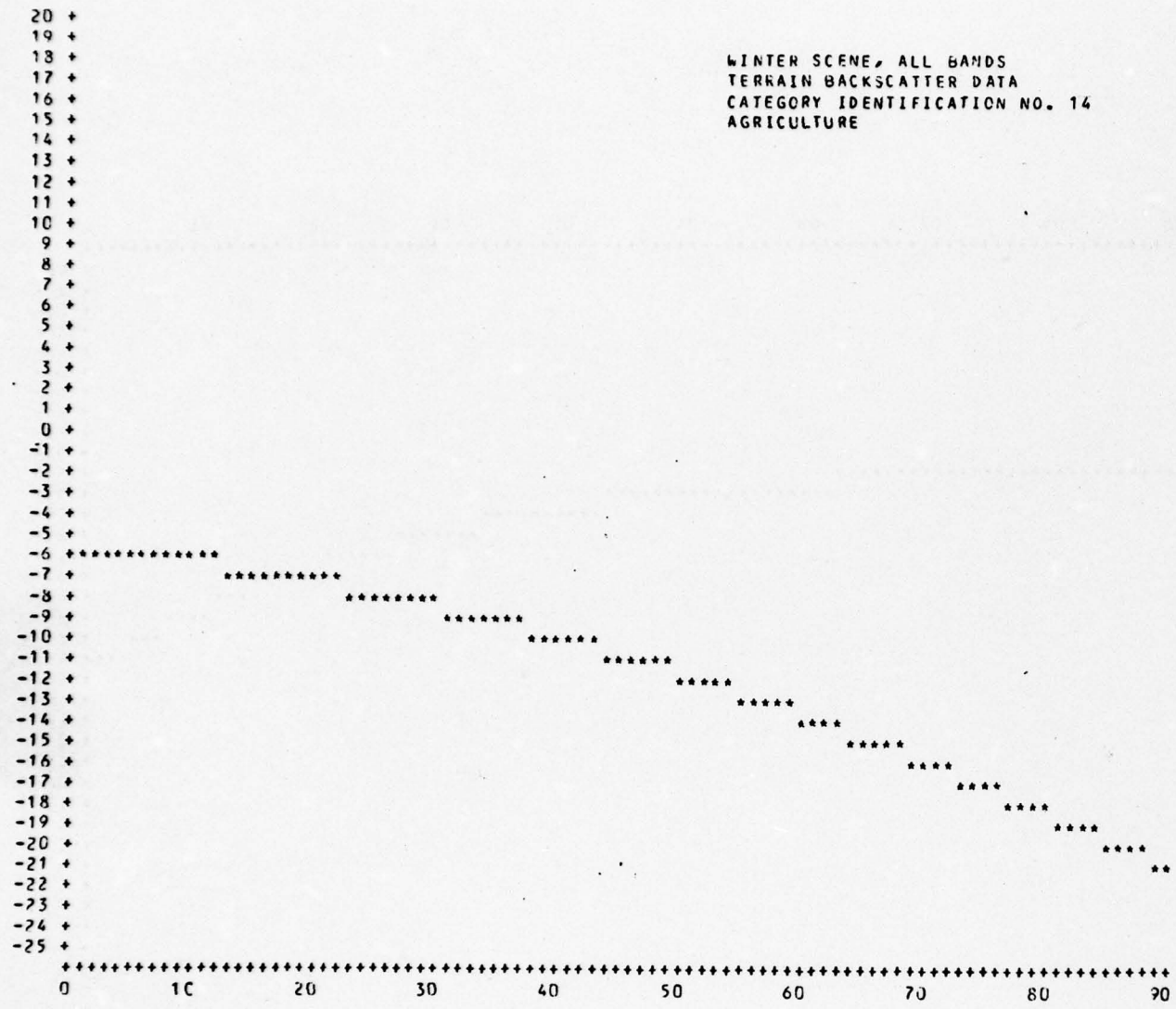


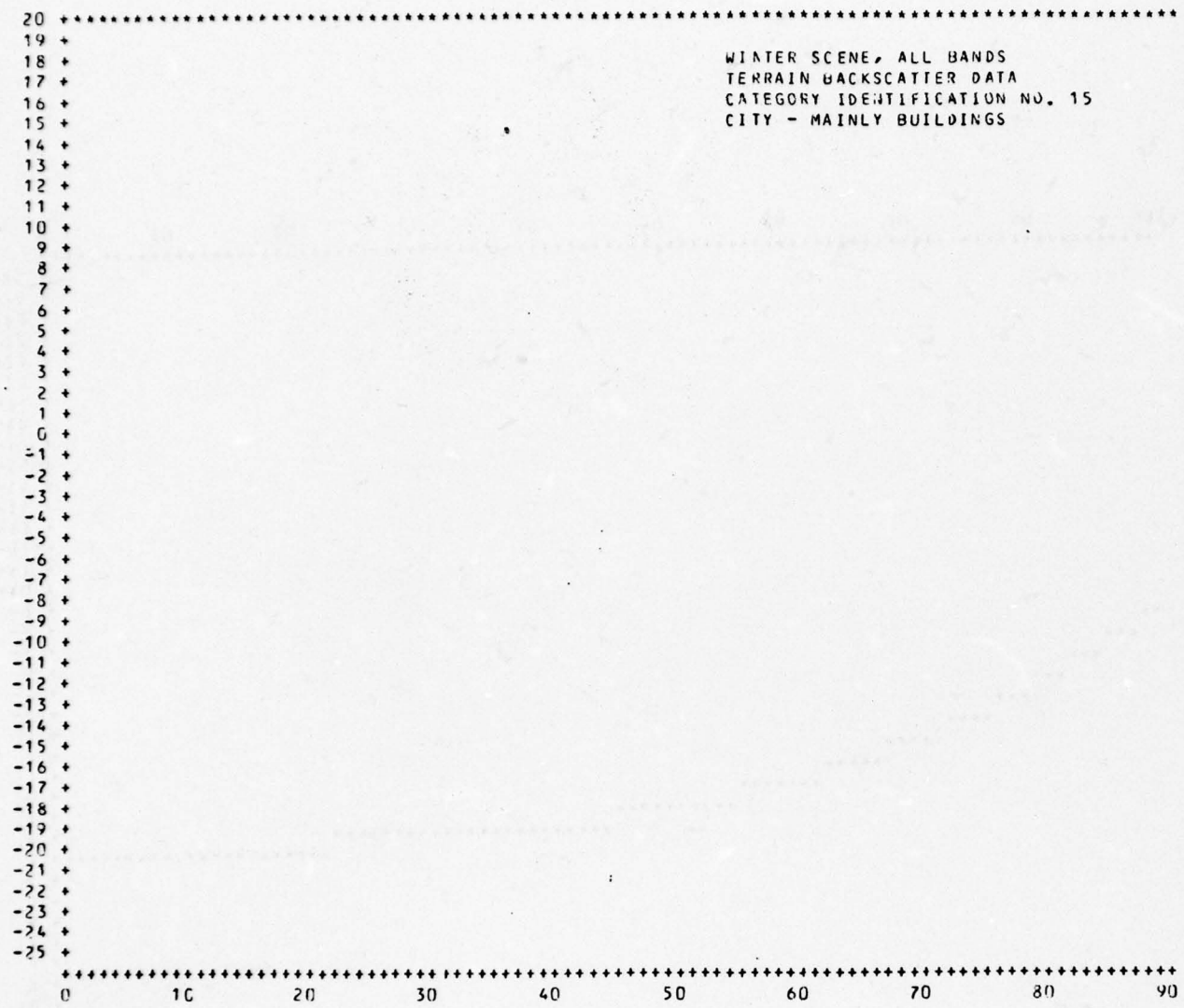

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 10
DECIDUOUS FOREST
dB
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

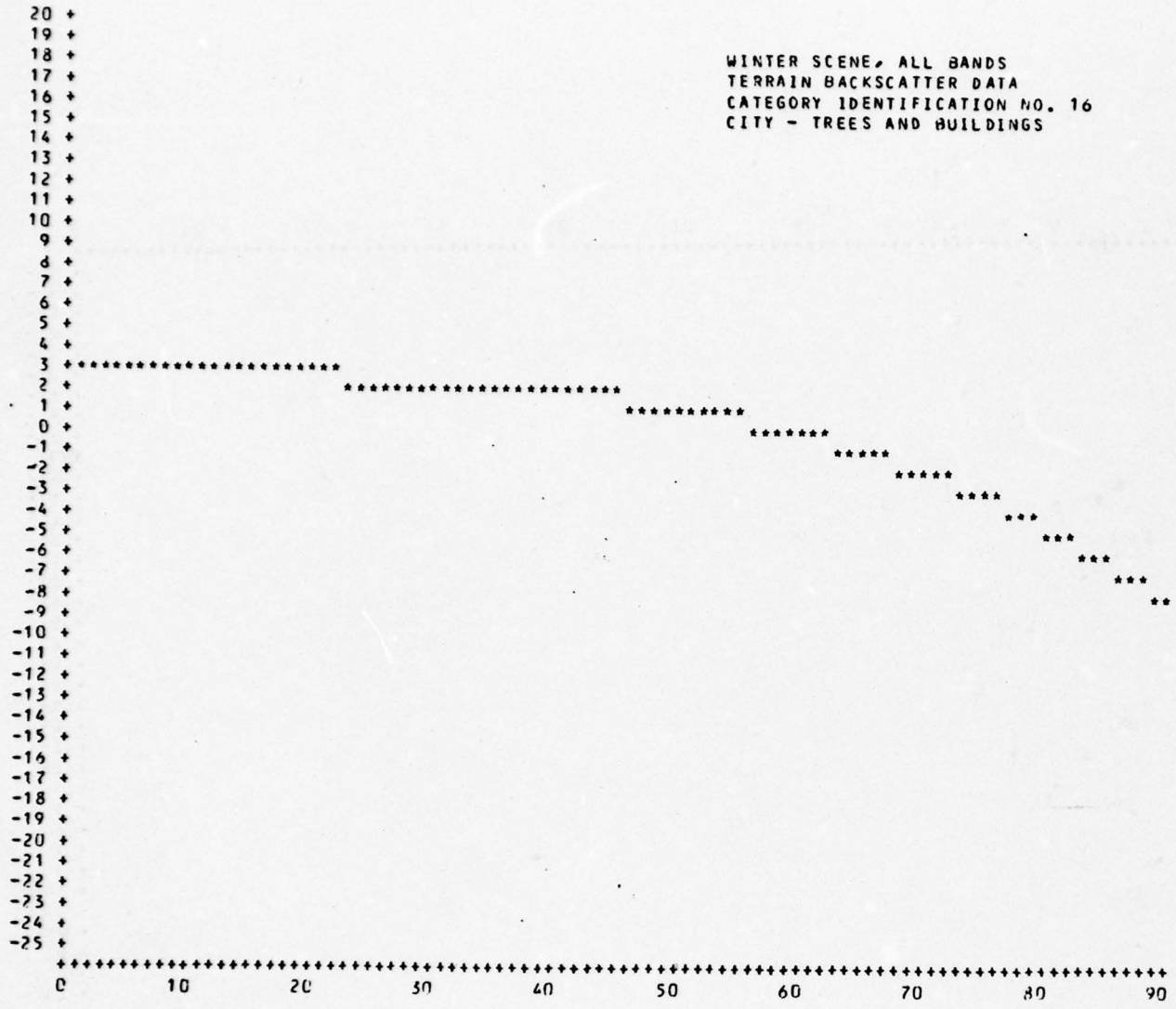

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 11
MIXED FORESTS
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

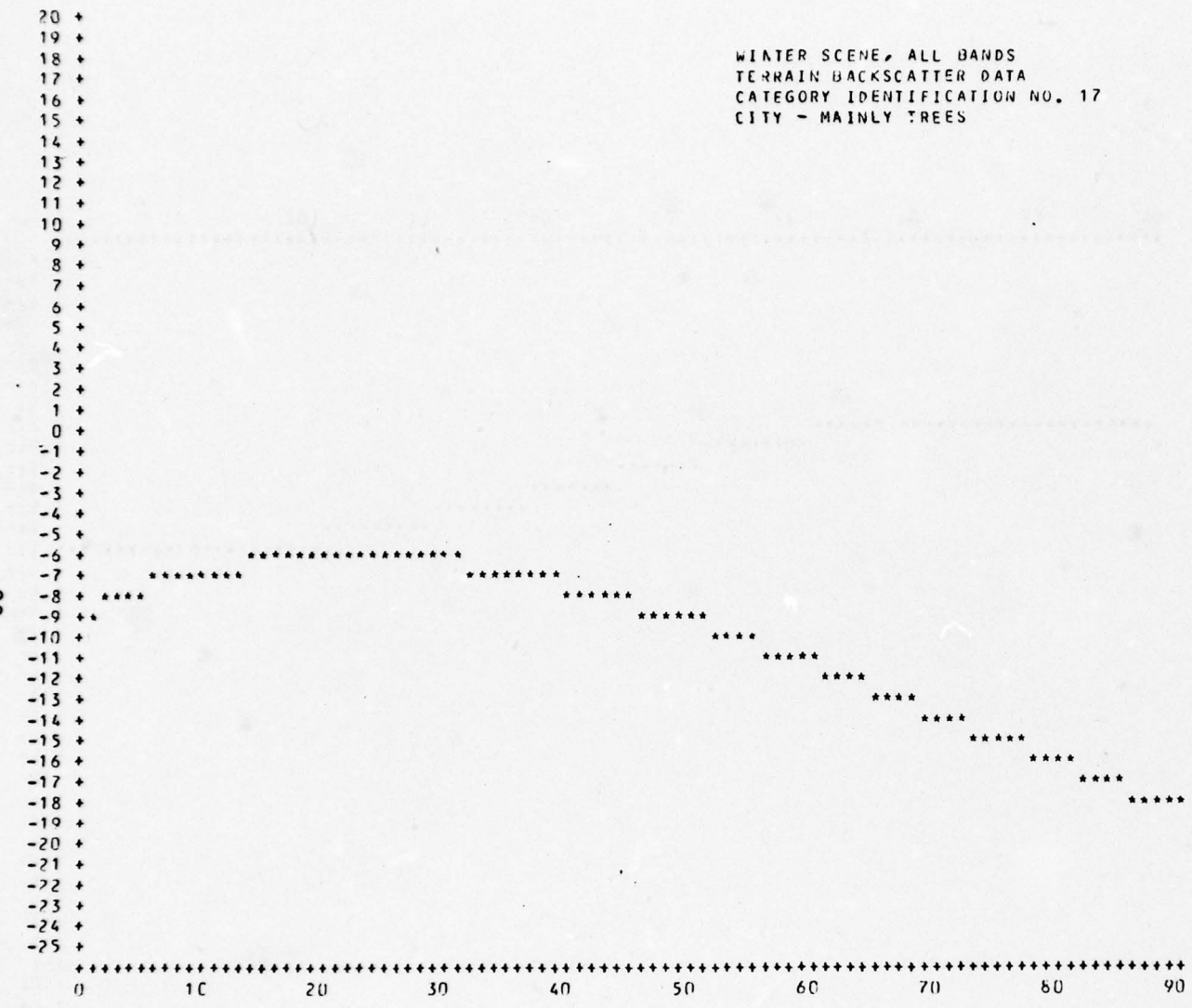

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 12
GRASS
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

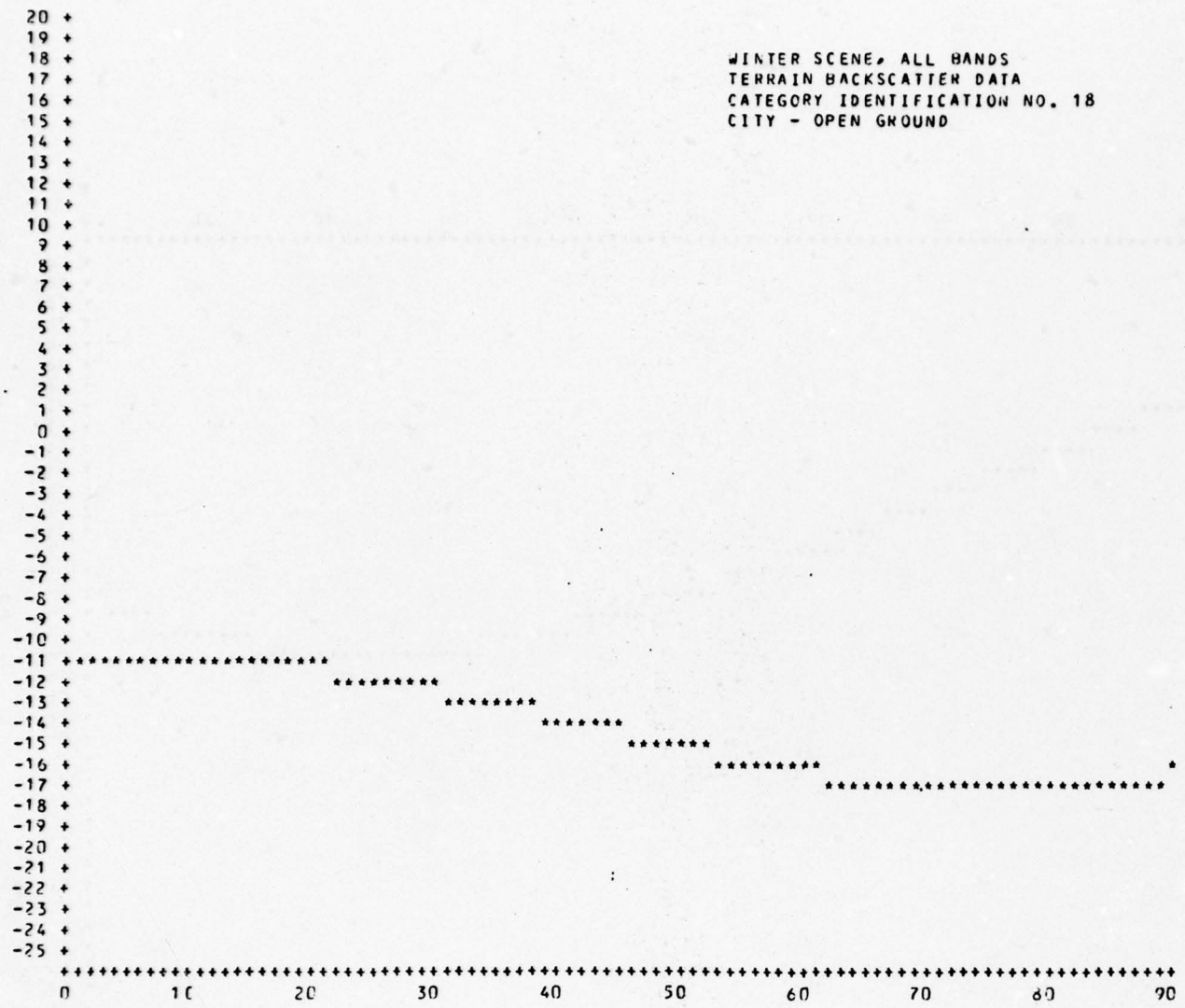

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 13
SCRUB
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

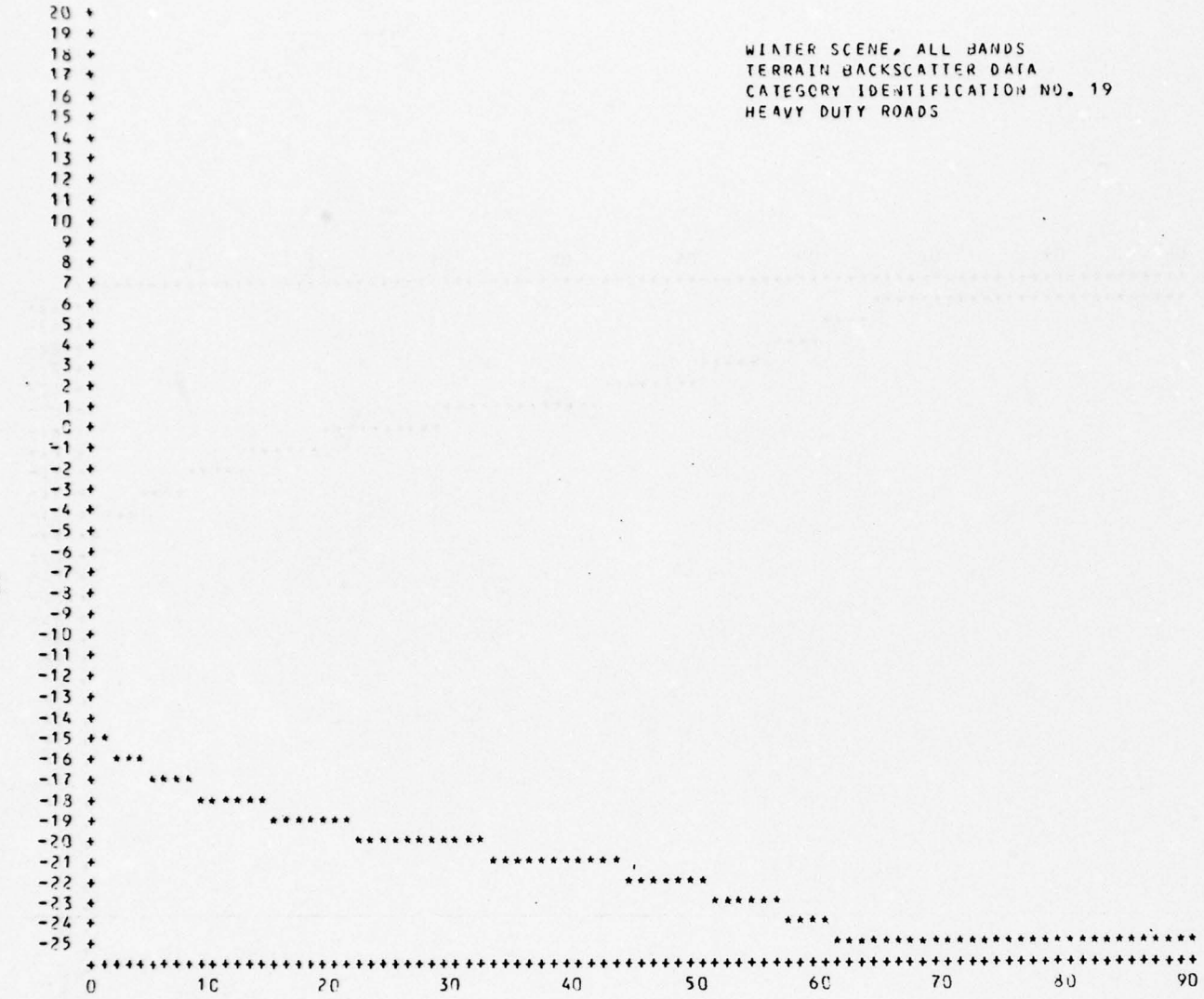

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 14
AGRICULTURE
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

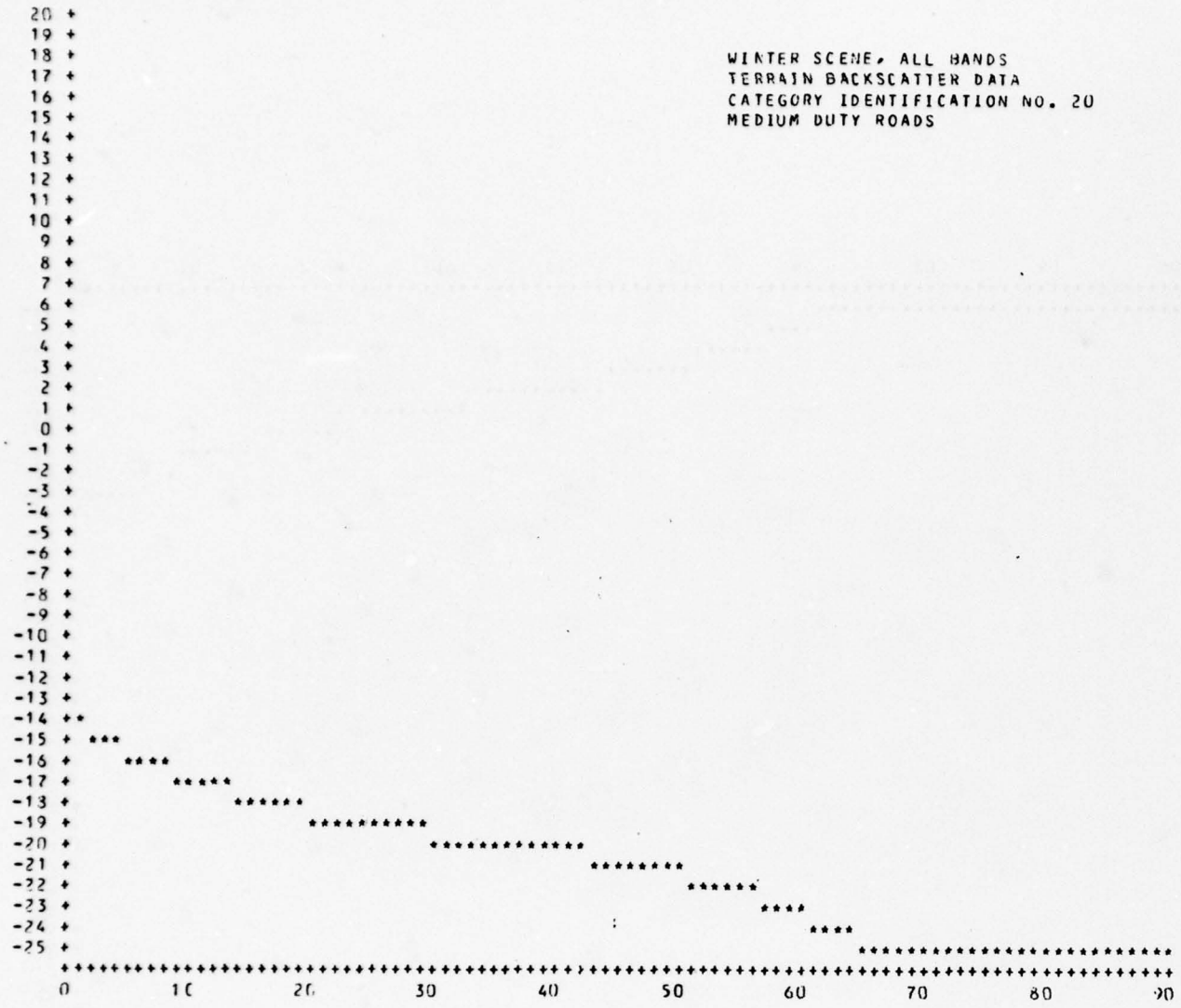

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 15
CITY - MAINLY BUILDINGS
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

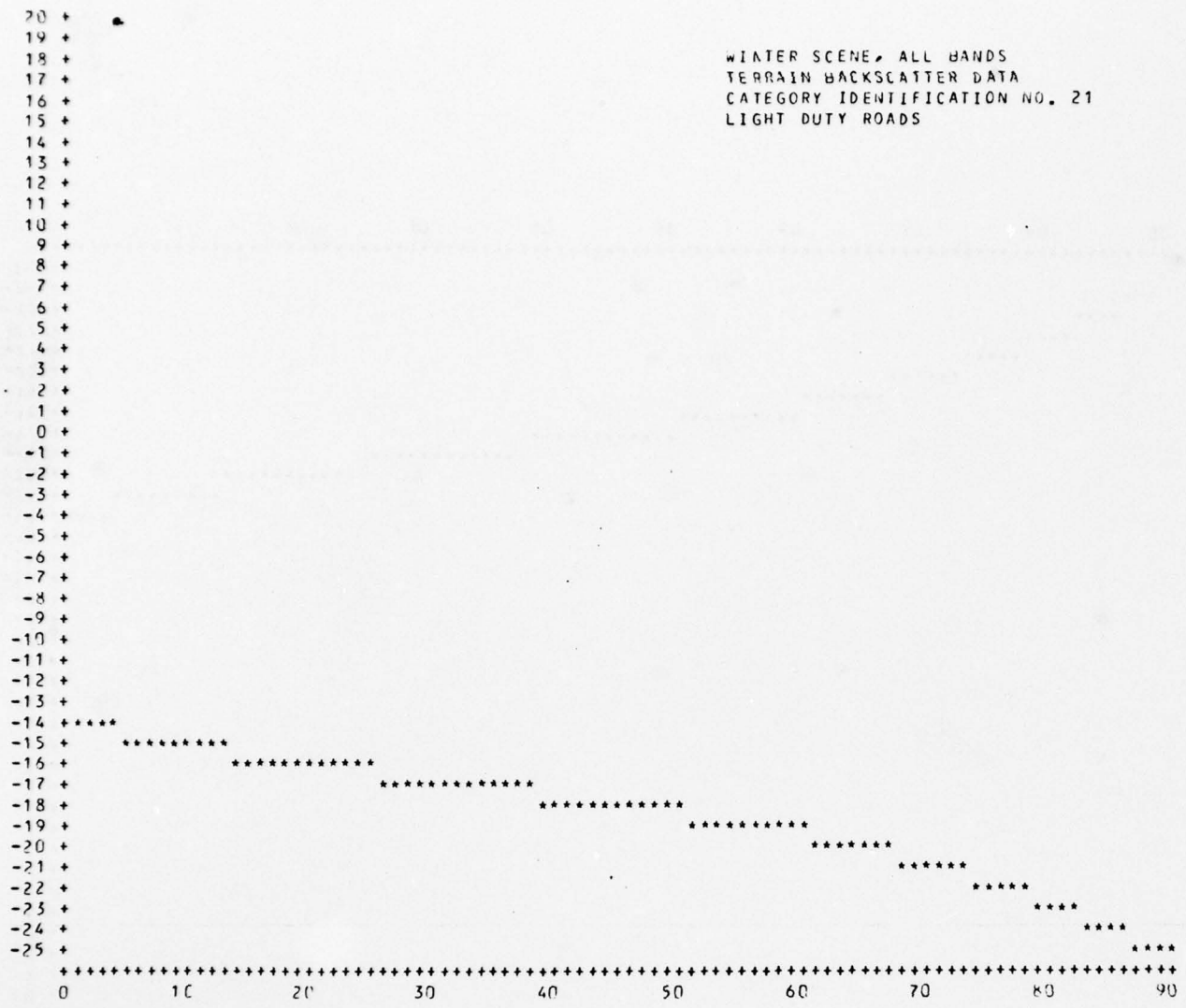

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 16
CITY - TREES AND BUILDINGS
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

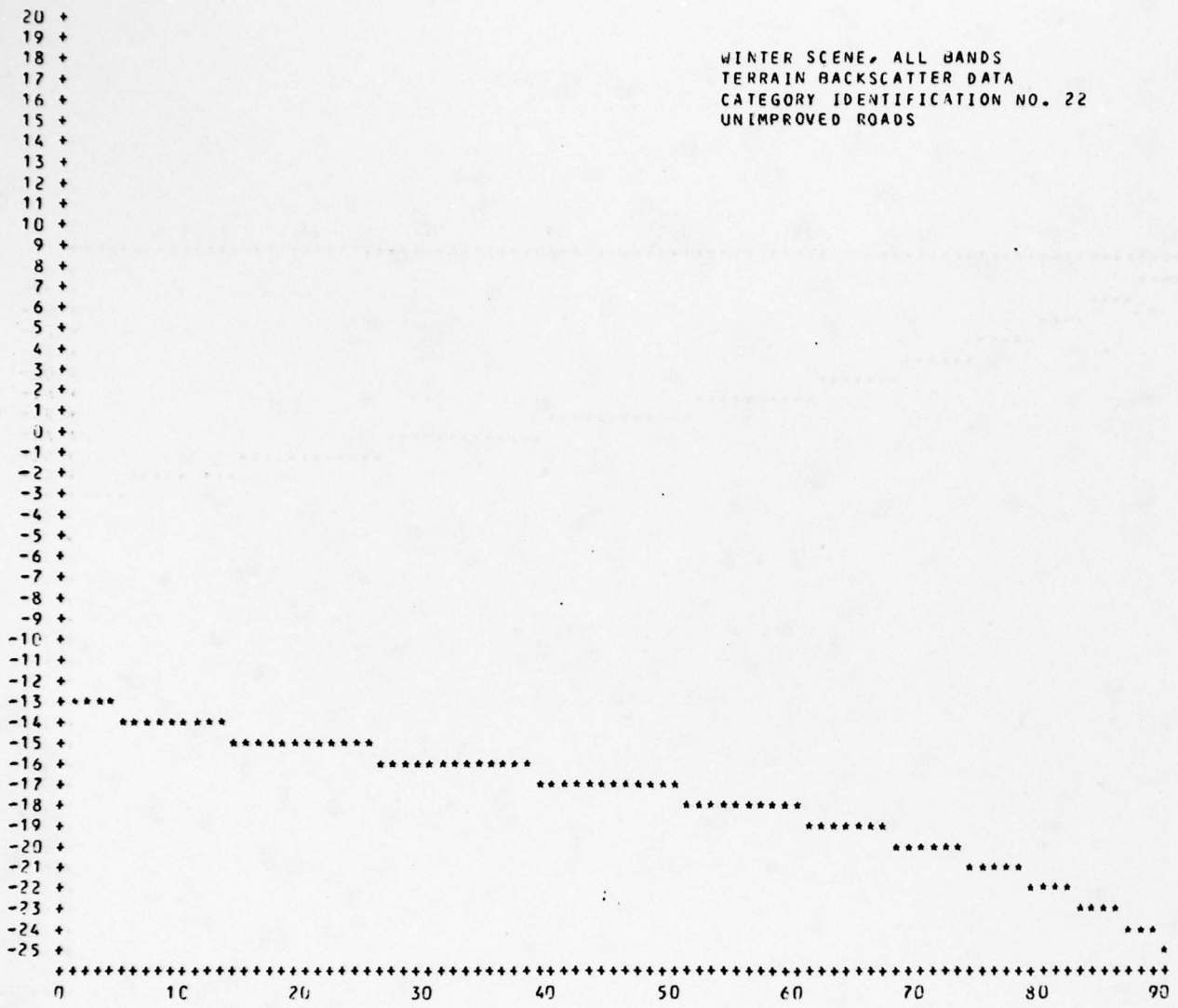

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 17
CITY - MAINLY TREES
dB
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

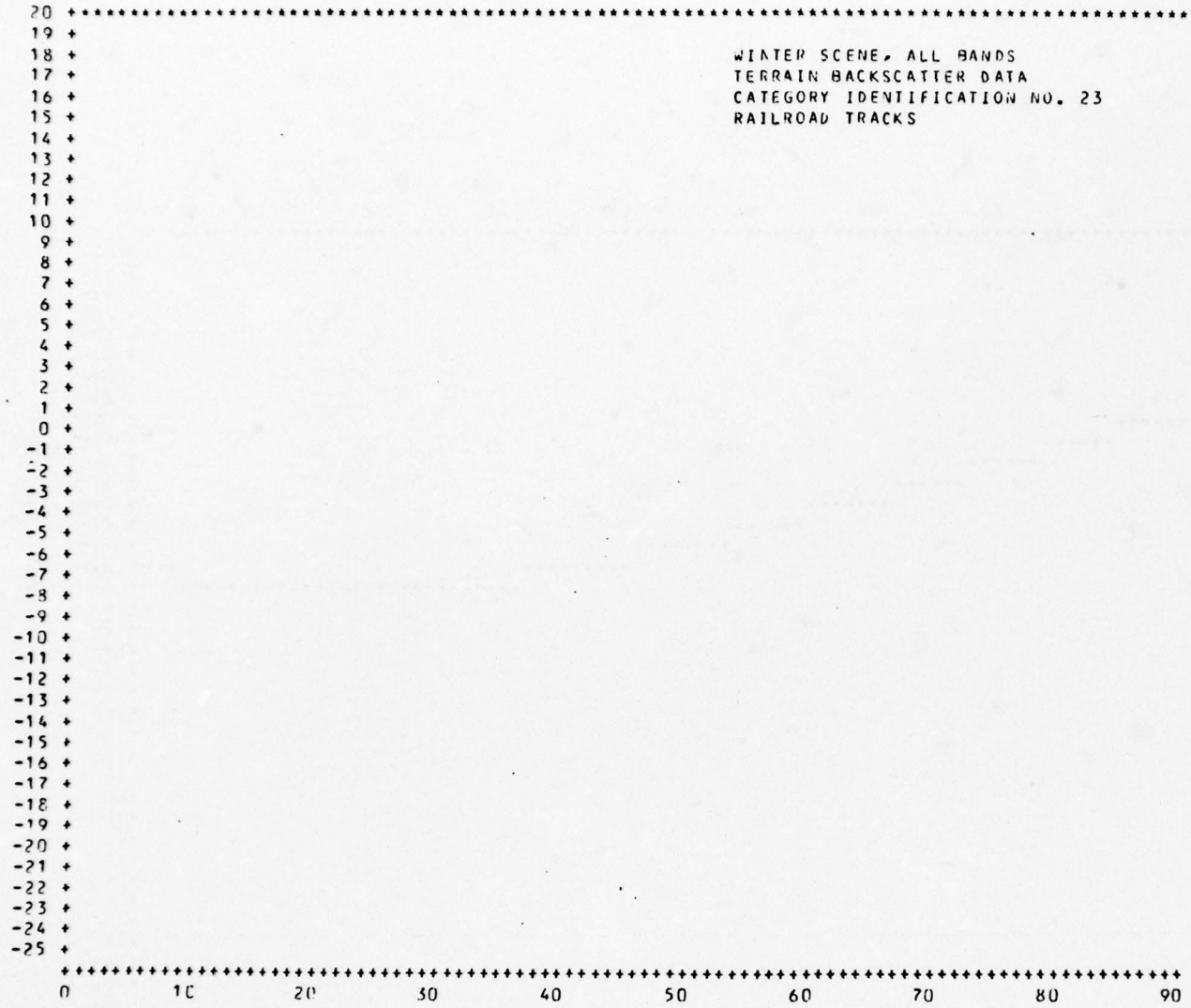

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 18
CITY - OPEN GROUND
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

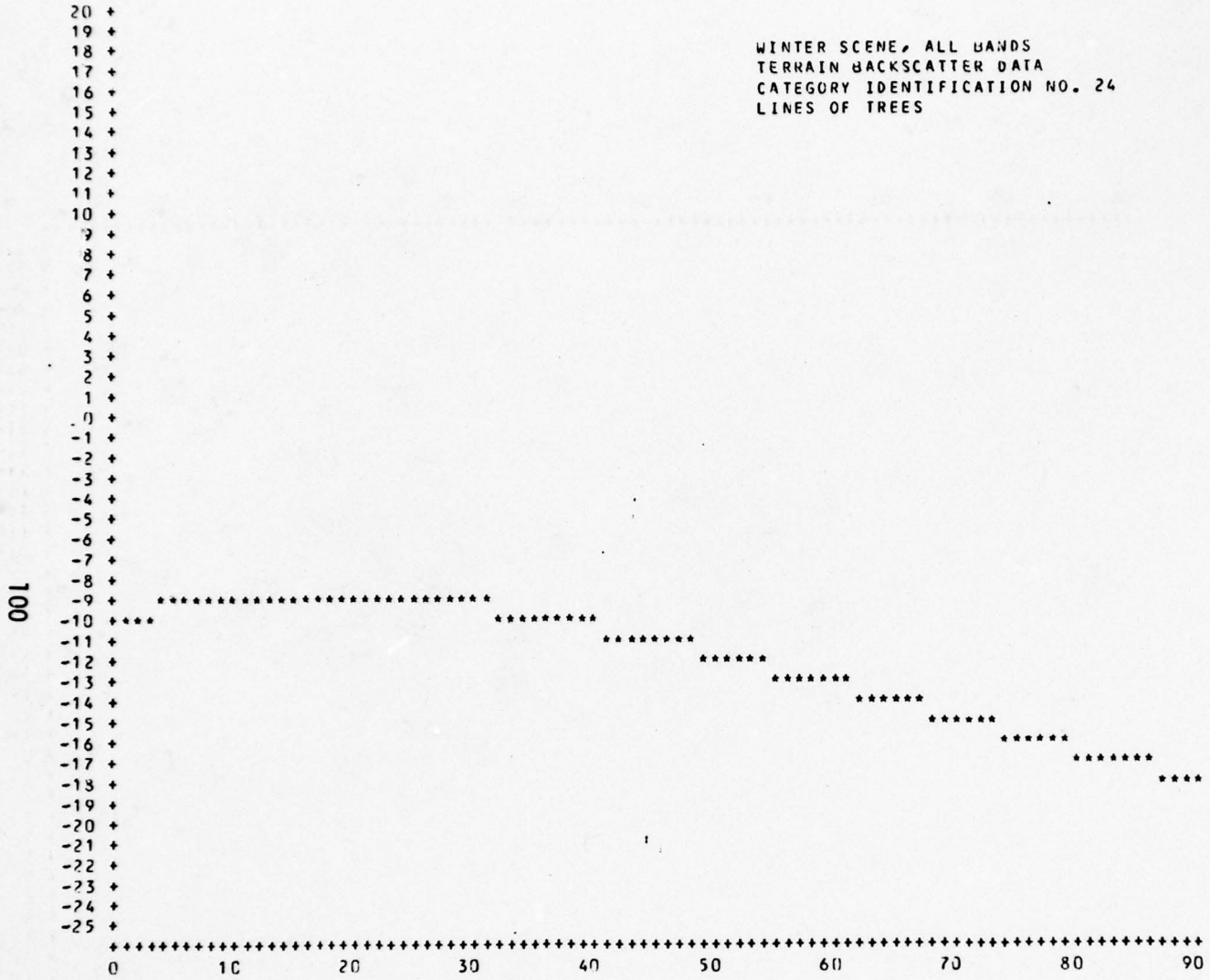

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 19
HEAVY DUTY ROADS
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

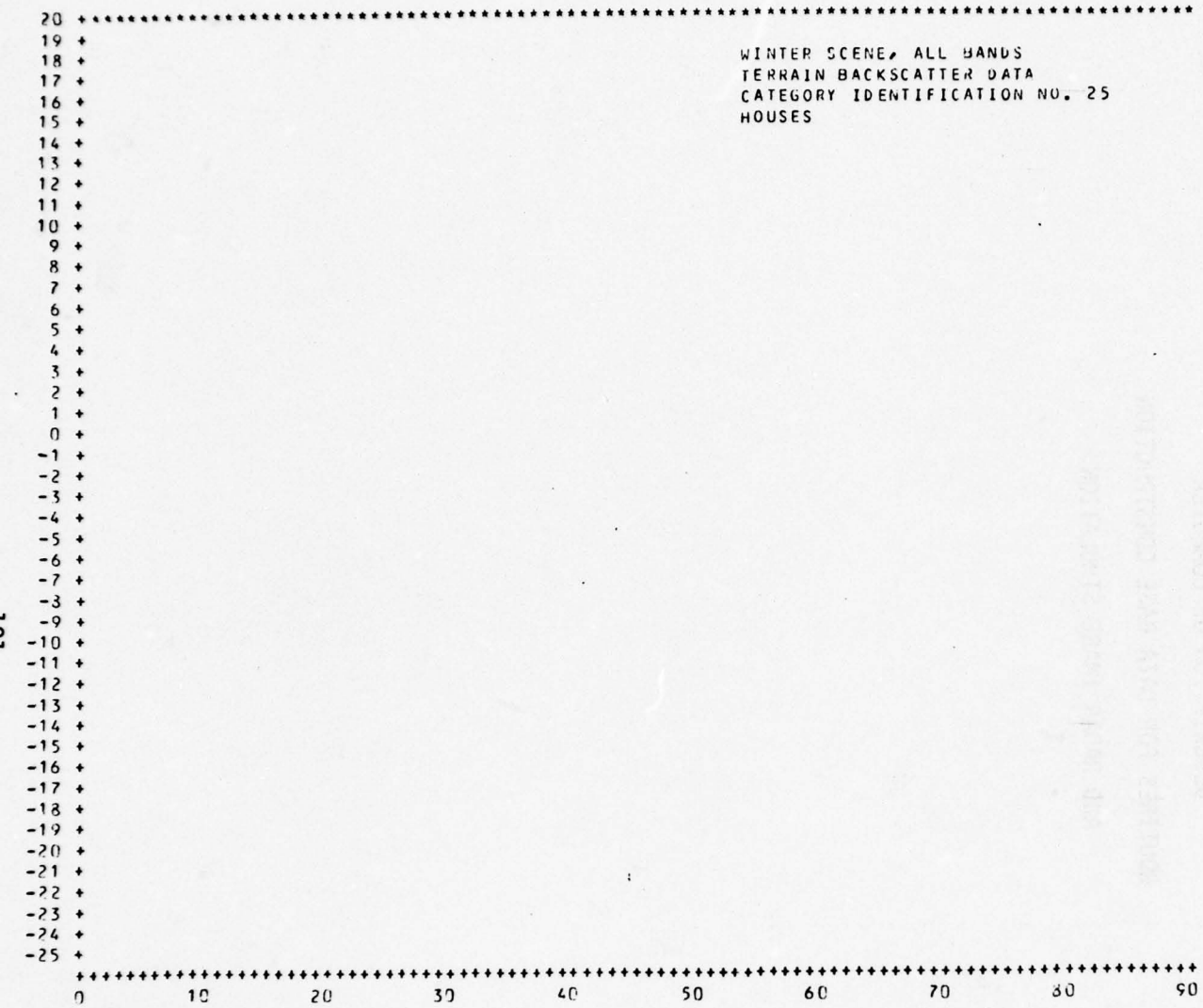

WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 20
MEDIUM DUTY ROADS
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90


WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 21
LIGHT DUTY ROADS
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90


WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 22
UNIMPROVED ROADS
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90


WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 23
RAILROAD TRACKS
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90


WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 24
LINES OF TREES
100
20 +
19 +
18 +
17 +
16 +
15 +
14 +
13 +
12 +
11 +
10 +
9 +
8 +
7 +
6 +
5 +
4 +
3 +
2 +
1 +
0 +
-1 +
-2 +
-3 +
-4 +
-5 +
-6 +
-7 +
-8 +
-9 +
-10 +
-11 +
-12 +
-13 +
-14 +
-15 +
-16 +
-17 +
-18 +
-19 +
-20 +
-21 +
-22 +
-23 +
-24 +
-25 +
0
10
20
30
40
50
60
70
80
90


WINTER SCENE, ALL BANDS
TERRAIN BACKSCATTER DATA
CATEGORY IDENTIFICATION NO. 25
HOUSES
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
0
10
20
30
40
50
60
70
80
90

APPENDIX B

DESCRIPTION OF COMPUTER ROUTINES FOR DATA BASE CONSTRUCTION AND RADAR IMAGE SIMULATION

Polar Conversion (Refer to Section 1.6.2 for motivation)

The first step performed by the sequence of computer programs illustrated in Figure 3 is POLAR CONVERSION, the computer program for converting the rectangular grid matrix of the ground truth data base into a polar array which is compatible with the scanning format of the PPI radar being modeled. POLAR CONVERSION actually consists of three (3) distinct computer programs, POLAR CREATE, POLAR ARRAY, and ARRAY FIX; copies of these programs are provided in Appendix C.

Three programs have been developed to perform the straight-forward function of rectangular-to-polar coordinates conversion in order to minimize the costs of performing it. The first program, POLAR CREATE, is a highly computational program requiring minimal core storage. This program accepts as input a ground truth data base stored on digital magnetic tape, computes the polar address (r,θ) of each point as it is read from tape, performs one-dimensional compression on the data and stores the data sequentially on an intermediate magnetic tape. Compression arises from the fact that the rectangular version (original version) of the data base contains finer resolution and, consequently, more sample points than are desired in the polar data base. The number of elements desired in the final polar data base (calculated from the radar resolution parameters) is a control parameter input to the program and is used to quantize the polar conversion calculations and, thus, produce a mechanism for compression.

The second program, POLAR ARRAY, is a core-intensive but computationally-minimal program. This program accepts as input the intermediate magnetic tape output from POLAR CREATE, orders the data from the sequential rectangular file to the correct polar array, and stores this on an intermediate magnetic tape. Two-dimensional compression is performed by this

program. This follows the same philosophy as does the earlier compression. The category data are compressed on a priority basis; the highest priority category brought from the rectangular data base for each polar cell is retained, the others deleted. Elevation data are averaged. The elevation value stored in any polar cell is the average of the elevation of all the rectangular points which were mapped into it.

The third program, ARRAY FIX, was created to rectify the problem created by the fact that the rectangular-to-polar-conversion mapping is less than one-to-one in the center portion of the polar data base. This program interrogates the nearest filled neighbor to determine the category and elevation of a polar cell found empty. Upon satisfactory operation of this program, the completed polar data base is stored on an intermediate magnetic tape.

In this way POLAR CONVERSION functions to convert a large rectangular grid matrix data base into a smaller polar array data base for minimal cost. Cost is minimized because the amount of core (a high-cost component) which must be used during the computationally-intensive portions of the operation is reduced to the bare-essential amount. The output of POLAR CONVERSION is a computer-compatible magnetic tape containing the ground truth data base arrayed in a polar format with the correct resolution to support directly the REFERENCE SCENE programs, the next programs which must be run.

Reference Scene (Refer to Section 1.6.2)

Reference to Figure 3 will show the second step performed is REFERENCE SCENE, the computer program which actually performs the simulation of the guidance radar system and forms the desired simulations of radar images. REFERENCE SCENE actually consists of two (2) computer programs, POWER and GREYTONE; copies of these programs are provided in Appendix C.

Two programs have been developed, instead of one, to minimize the costs and improve the operational efficiency of running the programs many times. The first program, POWER, is a computationally-intensive program requiring minimal core storage. This program accepts as input both the polar ground truth data base on digital magnetic tape from program POLAR CONVERSION and the terrain backscatter data input via a data statement, calculates the average power exiting the receiver on a pixel-by-pixel basis for each pixel (picture element, previously called "point") in the final scene, and stores these data on an interim magnetic tape. POWER recognizes each radial record in the polar data base as the scan line corresponding to the energy returned from one pulse of the radar, each point in the record corresponds to a resolution element. The polar data only contains a record for successive, independent scan lines (pulses).

The PRF (Pulse Repetition Frequency) of a radar system is normally quite high with successive pulses producing a return having a large overlap with several preceding pulses. POWER calculates a new scan line (new pulse) of data only for scan lines which are independent of one another (they do not overlap each other), and calculates the dependency of overlapping pulses statistically. This is done in order to minimize both the size of the polar data base and the computational load (and, thus, cost) required to produce radar reference scenes.

Similarly, POWER recognizes each point in a scan line from the polar data base as an independent resolution element in the radial direction, and calculates the data relating each point on the ground to a pixel in the image. Dependency of overlapping samples in the radial direction is statistically incorporated in the model.

For each point in the data base (each resolution element on the ground), POWER solves the geometry relating the position of the radar platform (three-dimensional position) to the point and calculates the slope of the terrain, and the angle of incidence and the range between platform and point. These calculations are made sequentially for each point of each record as the tape containing the polar ground-truth data base is read into the computer. Upon determining these parameters for a point, POWER enters the main computational algorithm of the program which calculates the average power exiting the receiver from that point. This calculation of power uses the slope of the ground (two-dimensional slope), the angle of incidence between radar and ground (both normal and local angles), the range from platform to the point on the ground being interrogated, the power pattern of the antenna, the category identification from the polar data base, backscatter data from the σ^0 file (such as those discussed in Section 4), and the transmitter/receiver/image model incorporated for the radar system whose response is being simulated. All of these variables and parameters are combined appropriately for calculating the estimate of power existing the radar receiver for each point on the ground. In this way POWER calculates the average power exiting the receiver on a pixel-by-pixel basis. The resultant data are stored on an interim magnetic tape for further processing in later stages. The data are ordered sequentially on this tape in the same form as the polar array in which the polar radar data base was input.

The second program of REFERENCE SCENE was developed to incorporate the spatial relationships between adjacent, <u>independent</u> cells decreed by the antenna pattern, and to convert the resultant estimates of power into greytones. This program, GREYTONE, calculates the spatial relationships between cells via an autocorrelation. The shape and length of the

autocorrelation are input parameters. Upon completing the autocorrelation, GREYTONE converts the data, power, into density values, greytones, quantizes them into the desired number of bits, and biases the range to that desired for ultimate storage in a photograph.

The bias required to display a desired power range in an image is an input parameter to GREYTONE. The desired mapping ratio of power exiting the receiver into density in the photograph is also an input parameter; the portion of the radar dynamic range desired to be mapped into the dynamic range of the photograph (17-20 dB) is specified. Thus, upon specification of an autocorrelation function shape and length, and quantizing parameters (bias and mapping ratio, or gain) GREYTONE operates on the power map input via digital magnetic tape from the previous program, POWER, producing the greytone map of the final image on a pixel-by-pixel basis. The greytone data are stored on an intermediate digital magnetic tape. The stored data order is still the same as the input polar radar data base.

At this point the radar image simulation work is complete, but the data are still stored in a polar array. The following program converts these data from polar back to rectangular coordinates for compatibility with either standard raster-scan format display devices for evaluation or the Correlatron for testing purposes.

Rectangular Conversion (Refer to Section 1.6.2)

Reference to Figure 3 will show the third step performed is RECTANGULAR CONVERISON, the computer program which converts the simulated radar image from a polar array to a rectangular grid matrix. RECTANGULAR CONVERSION actually consists of two (2) computer programs, RECTANGULAR CREATE and RECTANGULAR ARRAY; copies of these programs are provided in Appendix C.

These two programs exist for the same purpose as POLAR CREATE and POLAR ARRAY (see Section 1.6.2.1) and perform the inverse operations of them. RECTANGULAR CREATE requires input specification of the size of the rectangular grid desired for the output data. The size of this grid is dependent upon the purposes for which the data have been created. If the data have been created for display and evaluation purposes, the size of the grid is entirely determined by the size of the area to be viewed and the size limitations of the display device. If the data have been created for testing on the Correlatron, the size of the output grid is specified to be 921 x 921 pixels. Upon specification of the size of the output rectangular grid matrix, RECTANGULAR CREATE calculates again the polar address (r,θ) of each point in the specified rectangular array and stores these data on an intermediate magnetic tape.

RECTANGULAR ARRAY requires input specification of the data format of the magnetic tape to be output. For display and evaluation purposes, the format is defined to be raster format with the word length for the greytones of each pixel, and whether a positive or a negative is desired to be specified. For testing on the Correlatron, the output format is defined to be:

9 track digital magnetic tape;
1600 bpi;
921 records;
921 pixels per record;
0 corresponds to white;
255 corresponds to black.

Upon specification of these input parameters, RECTANGULAR ARRAY converts the data output from RECTANGULAR CREATE into a completely specified rectangular grid having the desired output format, and stores these data on a digital magnetic tape.

INITFIX (Refer to Section 3.4 for motivation)

INITFIX is the first in a series of computer programs used in the construction of the digital data base. The purpose of this program is to "clean up" the input from the digitizers, eliminate duplicate points, fill in gaps between consecutive points, and verify the input. The input to INITFIX is a series of logical records, each representing a target. Each logical record for a target consists of a header record, a series of x,y coordinates, and an end-of-target marker. The header record for each target contains the category and feature code for the target. The feature code identifies the type of target: point, line, or closed boundary (area). The x,y coordinates define the location of the target with respect to the coordinate system defined by the digitizers. The end-of-target marker is the coordinate (-1,-1).

Each input target is assigned a boundary number regardless of feature code. INITFIX consists of a major loop which processes one input tape each time it is executed. Inside this loop is the "target" loop which processes one target per iteration until the end of the current input tape. Inside the "target" look, there are two separate loops: the "point" loop which processes a group of point targets, and the "area-line" loop which processes area targets (closed boundaries) and line targets.

Whenever the current input tape reaches end-of-file unexpectedly, the current target is finished up and an error message is printed. Whenever an error message is printed, subroutine UPDATERR is called to save the error type and the target in which it occurred. This is done so a summary of all errors encountered can be printed to a separate file code after all input tapes have been processed. The reason for this is to facilitate the human verification of the program execution.

The output format for points and lines consists of a one-word record containing the boundary number, followed by a series of two-word records, each describing a point in the target. The first word in these records contains the x-coordinate of the point and the second word contains the y-coordinate, the category, and the feature code (1 for points and 3 for lines). The last two-word record in each series is (-1,-1).

The output format for areas is slightly different from that for points and lines. The format consists of a three-word header record followed by a series of two-word records. The header record contains the boundary number, category, and feature code. The first word in each two-word record contains the x-coordinate and the second word contains the y-coordinate. As for points and lines, the last two-word record in each series contains (-1,-1).

Execution of the "target" loop begins with the input of the header record for the next target. The category is tested for being the reference point category. If so, subroutine SKIPDUMP is called to skip the points in the target (these points were used to initialize the coordinate system for the digitizing table). If the category is out of range or assigned the "catch-all" category for areas without categories, an error message is printed. If the feature code is out of range, an error message is printed and the target is treated as being a set of point targets. Execution continues depending upon the feature code of the target.

If the feature code identifies the target as being a set of point targets, the program enters the "point" loop to process one target at a time until the end-of-target record (-1,-1) is read. Before entering the loop, the counters for the number of input points, duplicate points, and output points are reset. In addition, the boundary number for the set of points is written to the point-line tape. Each pass through the "point" loop performs the following:

(1) A point is read and tested for being (-1,-1). If it is, the program exits the "point" loop.

(2) The point is converted to the proper scale and the input counter is incremented. The range of the point is verified and if out of range, an error message is printed.

(3) The point is compared with the last input point. If they are equal, the duplicate counter is incremented while the "point" loop iterates for the next point target.

(4) The point is saved for use in the next iteration and is used to update the minima and maxima variables for points and lines.

(5) The output counter is incremented, the point is written in the point-line format to the point-line tape, and the loop is executed again.

Once the point (-1,-1) has been read, the program exits the "point" loop, writes the end-of-target marker to tape, prints the values of the three counters, updates the overall counters, and returns to the top of the "target" loop.

If the target is an area or a line, the program enters the "area-line" loop which processes one point of the target each time it is executed. Before the loop begins, the minima and maxima variables for areas are reset and the counters for the number of input points, duplicate points, output points, points filled in, and calls to the subroutine to fill in missing points are cleared. The first point of the target is read, converted to the proper scale, and saved for later processing. If the input point is (-1,-1), then the "target" loop starts again. Otherwise, the boundary number is written to the point-line tape if the target is a line, or the boundary number, category, and feature code are written to the area tape if the tar-

get is an area. Next the "area-line" loop begins and these actions are taken:

(1) The next point is read and a test is made for end-of-target. If it is the end of the target, then the loop is exited; otherwise the point is scaled and the input counter is incremented.

(2) The range of the point is verified and compared with the previous input point. If the point is out of range, an error message is printed. If the point is a duplicate, the duplicate counter is incremented and control returns to the top of the "area-line" loop for the next point.

(3) The previous point is written to the appropriate tape and the output counter is incremented. If the previous point is not adjacent to the current point, then subroutine CONNECT is called to fill in the missing points.

(4) The point is saved for the next time through and used to update the area minima and maxima variables if the target is an area, or the point-line minima and maxima variables if the target is a line. Control then returns to the top of the loop to process the next point in the target.

Once the end-of-target has been read, the loop is exited and if the first and last points are not identical, the last point is written to the appropriate tape. If the first and last points are not adjacent and the target is an area, then CONNECT is called to fill in the missing points. Next the end-of-target record is written to tape and the overall minima and maxima variables are updated. Before returning to the top of the "target" loop, the counters and minima and maxima variables are printed and the overall counters are incremented.

The target loop continues until the end of the current input tape. If there are more input tapes, the "tape" loop begins again and the next input tape is processed. Once all of the input tapes have been processed,

the overall totals are printed along with the overall minima and maxima and a summary of all errors encountered.

AREAFIX (Refer to Section 3.4)

AREAFIX is the second program used in the construction of digital data bases. The input to AREAFIX is the output from INITFIX for area targets. The purpose of this program is to eliminate single points that are tangent to an area target, eliminate vertical segments in the target, and classify all remaining points as being a "top" or a "bottom". A point may be labeled as such by examining the points immediately preceding and following the points to be classified. Four decision matrices were developed to correctly classify the point using the change in x and y coordinates between the three points. The algorithm assumes that the area targets were digitized clockwise and generates the exact opposite classification if the area were digitized counterclockwise. For example, consider an area that forms a perfect square. All the points in the bottom row would be classified as "bottoms" while all the points in the top row would be classified as "tops". All points in the two side columns, except for the endpoints, would be discarded. Thus, for each x-coordinate of the area, there are "bottom" and "top" y-coordinates that define the range of the area in that x-coordinate. Clearly, there is no need for vertical segments since the two endpoints of the segment define it. Inherent in this concept is the need for the elimination of single points tangent to the area. These points, if not discarded, would result in an x-coordinate with only one y-coordinate. This is undesirable as the range of the closed boundary, with respect to an x-coordinate, is dependent upon pairs of "bottoms" and "tops" for that

x-coordinate. The output from AREAFIX for one target consists of a one word header record containing the boundary number followed by a series of two-word records trailed by (-1,-1). The first word of each two-word record contains the boundary number and the x-coordinate. The second word contains: 1 bit used to indicate "top" or "bottom", the y-coordinate, the category, and two bits for the feature code (always a 2 for areas).

The mainline consists of an outer loop that processes one area target at a time. There is an inner loop that processes all the points in one area, other than the first and last, by making calls to the sub-routine ASSIGN. ASSIGN determines the label for one point and outputs it to tape in the prescribed format. If a point is labeled to be discarded, it simply is not written to tape.

The outer loop iterates until all areas have been processed. This loop, the "area" loop, performs these tasks:

(1) The category, feature code, and boundary number (assigned by INITFIX) are input from tape and printed.

(2) The first two points of the closed boundary are input and the first is saved to be processed after the last point in the area. If either point is the end-of-target marker, an error message is printed and the "area" loop starts again. Both points are used to update the minimum and maximum variables and the boundary number is written to tape. The positional differences between the first and second points are saved to be used when the first point is processed.

(3) The counter for duplicate points is cleared and the inner "point" loop is executed. Upon exit from the "point" loop, all points of the current area, except the first and last, have been processed.

(4) The last and first points are processed and the end-of-target marker is written to tape.

(5) The results of the processing are examined and printed. If fewer than four points were input, or the number of "tops" and "bottoms" were not equal, then an error message is printed. The number of tops, bottoms, and discarded points are printed along with the number of duplicate points and the total number of output points is incremented. Before looping back for the next area, the array to count the classifications is cleared.

The inner loop classifies each point of the area target (except the first and last) one at a time in the following fashion:

(1) The next point is read and compared with the most recent point read. If equal, the duplicate counter is incremented and the inner loop repeats. If the point is the end-of-target marker, then control exits the "point" loop to classify the first and last points.

(2) The point is used to update the minimum and maximum variables and ASSIGN is called to classify the point.

(3) The program loops back to process the next point.

The ASSIGN subroutine executes sequentially with no loops. The variable parameters that are passed to the subroutine include: the current point, the positional differences between it and the preceding point, and the positional differences between it and the next point. First a check is made to insure there are no gaps between the current point and the preceding and following points. The point is classified according to the positional differences that are passed to the subroutine. If the point is not labeled "discard", it is output in the aforementioned format. Before returning to the mainline, the coordinates of the point are incremented to yield

the next point and the variables for positional differences between the current and last points are given the values of the positional differences between the current point and the next point. This is done in anticipation of the next call to the subroutine which will process the next point. In addition, the counter for the classification of the point just processed is incremented prior to termination of the subroutine.

Once the last area has been processed, the program exits the "area" loop to finish up. The minimum and maximum x-coordinates are printed along with the total number of points output and a summary of all errors encountered.

COUNT AND SORT (See Section 3.4)

The next step in the construction of the digital data base is broken into two programs for more efficient resource useage.

The first program, COUNT, has as input the output from AREAFIX. COUNT uses one array to count the number of points having the same x-coordinate. For this reason, the array must be dimensioned at least as large as the domain of the input coordinates. COUNT consists of one main loop to process all input points. There is an isomorphic mapping from the domain of the input points to the elements of the array. The program requires that the array have at least twenty more elements than the domain in case there is an error in the minimum and maximum values passed to it. The program has the capabilities of deleting entire targets (as some are mistakenly digitized twice) and has an array containing the boundary numbers of the targets to be deleted. The output from COUNT is all done via a subroutine in order to buffer the output to decrease the execution time of the next program, SORT.

The main loop reads in the boundary number and determines if it is a boundary to be deleted. If so, it enters an inner loop which reads points until (-1,-1) is encountered at which point it returns to the top of the main loop. If the target is not to be deleted, the x-coordinate is unpacked and the corresponding element of the counting array (referred to as x-bins) is incremented and the input pair are written to tape. Execution returns to the top of the loop and the program loops until the end of input. Once all the input has been processed, the results of the count are written to a file and the program terminates.

The second program in this step is SORT. SORT sorts the output points from COUNT according to their x-coordinates. Because of memory limitations, the program must make several passes through the input to sort it all. It is for this reason that the number of points in each x-bin must be known prior to sorting. (This is accomplished by COUNT). The program uses three arrays: a large two-dimensional array to hold the sorted data pairs, an array of pointers to the sorting array for all x-bins to be sorted in this pass, and an array of flags reflecting whether or not the next point in the corresponding x-bin will start a new target. Similar to COUNT there is a monomorphic mapping from x-coordinates (to be sorted in the current pass) to the elements of the second two arrays.

The program consists of an outer loop which performs one "pass" through the input, and two consecutive inner loops. The first inner loop determines which x-bins are to be sorted in the current pass and the second inner loop sorts all input points that fall in the domain of the current pass. The program passes through the data until all the input has been sorted or the number of passes exceeds a user set variable.

Before the outer loop begins, the expected minimum and maximum x-coordinates are read from tape and printed. The outer loop first determines the starting x-coordinate for the current pass and enters the first inner loop. This loop reads the x-bin counts from a file and uses the counts to fill the second array with pointers. The second array then contains indices to the start of x-bins in the sorting array. It continues until the number of points in the next x-bin will not fit into the unallocated portion of the sorting array. The file is backspaced so the next pass can start with the proper x-bin count. The sorting array is then zeroed out and the input is rewound. Information concerning the current pass is printed out to paper and the second inner loop begins. This loop processes one two-word record in each iteration. First it reads the data pair and extracts the x-coordinate from the first word. If it is -1, then the end of a target has been encountered and all elements of the flag array are set to indicate "new target" and the loop continues to read the next data pair. If the x-coordinate is not -1, then it is tested for being within the domain of the current pass. If so, the index into the sorting array is obtained from the pointer array using the x-coordinate. Before storing the two words into the sorting array, the indexed element is verified as being zero. If not, an error occurred in the allocation of the array so an error message is printed and execution halts. If the current contents are zero, then the corresponding entry in the flag array is tested. If non-zero, then this is the first point of the target in this x-bin so the last two bits in the second data word are set to zero and the flag entry is reset. Finally, the two data words are stored in the sorting array and the index is incremented and restored to the pointer array. Upon exit from this loop (end-of-file) the contents of the sorted array are written to tape. For each x-bin sorted in the current pass, one variable sized record of all data

pairs in that x-bin preceded by a one-word record containing the x-bin is output to tape. Finally, the starting values for the next pass are calculated and execution returns to the top of the main loop for the next pass. The program loops until the entire domain of the input has been sorted.

BUILD (See Section 3.4 for motivation)

At this point in the construction of the data base, the digitized data has been converted into a series of x-bins. Each x-bin is comprised of two records, the first record being one word containing the number of points in the x-bin, and the second, variable in length, contains two words of information for each point. Thus, the input x-bins to BUILD are of the following form:

n1:(y1,b1),...,(yn1,bn1):n2,(y1,b1),...,(yn2,bn2);...

where

colon (:) = represents a record mark;

y = is one word with the following information packed into it: the y-coordinate, the category, the feature code, and one bit for the top/bottom classification. The bit used is the left-most bit, the sign bit, and is set (=1) if the point is a top and cleared (=0) if it is a bottom.

b = is one word containing the boundary number assigned by INITFIX.

The output from BUILD is again a two-record x-bin, but it is in a compact, ordered format. The first record is one word containing the number of words in the second record. Each three consecutive words in the second record constitutes one descriptor group for a boundary. The first word in a descriptor group contains the starting y-coordinate for the next area. The

second word contains the starting y-coordinate for the next area. The third word contains the category packed in the upper half and the boundary number packed into the lower half of the word.

BUILD first reads the beginning and ending x-coordinates from the input tape and writes the total number of columns (end is -1, begin is +1) to the output tape. The program enters the major loop, which processes one bin per iteration, and inputs the number of points in the next x-bin. Next all the two-word point descriptors are read into a two-dimensional array and the x-bin number and number of points are printed. The program enters an inner loop which processes one area in the x-bin during each iteration. Subroutine SORT is called which sorts the points of the next boundary in ascending order of y-coordinates. SORT stores the sorted points into the Y array and fills in the TYPE array with the top/bottom classification of the corresponding points in the Y array. Presumably the TYPE array should contain:

bottom, top, bottom, top, bottom, top, . . ., bottom, top

so the inner loop enters a deeper loop which attempts to match bottoms and tops in the Y array. Unfortunately, because of the slightest digitization error, the points don't always match up as expected. Because of this, the deepest loop (which processes one bottom-top pair per pass), tries to account for some errors. The user can control the amount of leeway given in error conditions, but an error message is always printed. Each time a bottom-top match is made, a three-word descriptor group is created and the innermost loop reiterates. Once the current boundary of the x-bin is processed, the innermost loop terminates and the inner loop reiterates to process the next boundary in the x-bin. Once the entire x-bin is processed,

the inner loop ends and all descriptor records created for this x-bin are output to tape as one record. The record is preceded by a one word record containing the total number of words in the descriptor-group record. The major loop then repeats to process the next x-bin.

The SORT subroutine performs several tasks:

(1) It is capable of deleting entire areas. If an area is to be deleted, SORT loops through the points in the area until it finds the first point in the next boundary not to be deleted or until it has searched through all remaining points. If the next boundary is not found, the subroutine returns with the parameters altered to reflect this condition.

(2) SORT is capable of reversing tops and bottoms in a boundary. Sometimes it is the case that an area was digitized counterclockwise. When this happens, all of the top-bottom classifications assigned by AREAFIX are exactly opposite. If the current area in the x-bin is one to be reversed, the values for top and bottom which are stored in the TYPE array are swapped. This, in effect, reverses all of the classifications assigned by AREAFIX.

(3) SORT sorts the current boundary in ascending order of its y-coordinates. A double sort is used with the sorted array being restored into the input array. If two points have the same y-coordinate, and they have different classifications, then the bottom is stored before the top.

(4) SORT fills up the Y and TYPE arrays by looping through the sorted boundary in the input array. Each pass of this loop extracts the y-coordinate and type from the input array and stores them into the Y and TYPE arrays respectively. Finally, SORT returns to the mainline with the parameters updated for the processing of the area.

CATEGORY AND CULTURAL MERGE (See Section 3.5 for motivation)

The first program of the sequence, CATEGORY and CULTURAL MERGE, differs markedly in purpose from the other two programs for it requires that category information actually be changed. For this reason, the program is

actually a modified version of MERGE. While MERGE started the processing of each record with an empty work array, the modified MERGE first reads in a record of the already completed category data base into this work array, and then proceeds to process the data for the cities. This effectively causes the completed city data base to overwrite the old category data with the new city data. The processing of the city data is altered from the normal MERGE algorithm so that where MERGE would attempt to fill in gaps between digitized areas whose boundaries do not exactly coincide, the modified MERGE ignores these gaps, and hence, leaves the original category data where there is not any city data. After the processing of each record of city data is completed, the program checks the linear and point target input, to determine if a record should contain any roads, railroads, or houses in it, and if it should, these categories are used to overwrite whatever category is at the designated coordinate in the work array. The work array is then output to tape as a record of the new data base. The cultural data base is given full priority over the category data base, so that it replaces whatever category might be found in the category data base. This insures that the linear targets contain no breaks, that all of the point targets are represented in the data base, and that the cities are accurately represented in the new data base. Therefore, the output from this program is an accurate merging of the category and cultural data bases.

This new data base is then used as the input to the next program ADD SNOW, which converts the field numbers to category numbers, and which adds the snow data base. The reassignment is necessary for several reasons. First, the fields in the category data base were given numbers which not only reflect the category, but which also allowed easy identification of individual fields, enabling easier correction during the construction of the category data base. Further, the simulation package uses the category

numbers as indices on arrays, and hence, the category numbers should be sequential, starting from one for the greatest efficiency. Finally, as will be explained later, small integers were preferred as indicative of category for ease of packing data into computer words. The translation of categories is accomplished by taking each point and using it as an index on a conversion array which contains the new category to be assigned. Therefore, the program reads in a record of the combined category and cultural data base, and sends each point in the array through the conversion array. Now it is necessary to add the snow data in such a way as to allow the simulation package easy access to either the category or the snow data. This is accomplished, along with a savings in space, by packing the snow and category information into a single word. The snow data base categories can uniquely be packed into three bits, while the reassigned categories can be packed into five bits. Therefore after the record of the category data base has had its points reassigned, the corresponding record of the snow data base is read, and each point in it is assigned the correct three bit code which is actually the old value divided by ten. The new snow value is then shifted left six bits and added to the reassigned category value, and restored in the record. After the whole record is processed, the record is written to tape. This tape, then contains the combined category, cultural, and snow data bases.

The last program, ADD ELEVATION, requires a complete elevation data base for the scene and data base produced by the second program be input. For each point of category data, the appropriate elevation data is shifted left 18 bits, and then added to the category data. After all the points in a record have been processed, the record is then output to tape. This

is the final tape containing the complete data base in rectangular coordinates with each word containing the elevation, snow depth, and category type for the ground spot it represents.

```
-79   20.765        I N I T F I X                                          LA

  1       C INIT     I N I T F I X
  2       C
  3       C      INITFIX IS THE FIRST IN A SERIES OF PROGRAMS USED TO CREATE
  4       C    CATEGORY DATA BASES. INPUT TO THIS PROGRAM IS A SERIES OF TWO
  5       C    WORD RECORDS, DIVIDED INTO LOGICAL GROUPS (VARIABLE IN SIZE).
  6       C    EACH LOGICAL GROUP CAN DESCRIBE A CLOSED BOUNDARY, A LINE, OR
  7       C    A SET OF SINGLE POINTS. THE FIRST RECORD OF EACH LOGICAL GROUP
  8       C    CONTAINS THE CATEGORY (FIRST WORD IN RECORD) AND THE FEATURE
  9       C    CODE (SECOND WORD IN RECORD). THE FEATURE CODE IDENTIFIES THE
 10       C    LOGICAL GROUP AS BEING A BOUNDARY, LINE, OR SET OF POINTS. THE
 11       C    CATEGORY IS VALUE ASSIGNED BY THE DIGITIZERS (OPTIONALLY UNIQUE)
 12       C    TO IDENTIFY THE BOUNDARY, LINE, OR SET OF POINTS. THE LAST RECORD
 13       C    IN EACH LOGICAL GROUP CONTAINS TWO WORDS, EACH WITH THE VALUE OF
 14       C    NEGATIVE ONE (-1). BETWEEN THE FIRST AND LAST RECORD IN EACH
 15       C    LOGICAL GROUP IS A SERIES OF TWO WORD RECORDS EACH REPRESENTING
 16       C    A SINGLE POINT. THE FIRST WORD OF THE RECORD IS THE X-COORDINATE
 17       C    AND THE SECOND WORD OF THE RECORD IS THE Y-COORDINATE.
 18       C
 19              IMPLICIT INTEGER  (A-Y)
 20              LOGICAL  DUMP ,NODUMP, SWITCHXY
 21       C
 22       C*****************************************************************
 23       C
 24       C      COMMON DECLARATIONS, DEFINITIONS, AND DESCRIPTIONS
 25       C
 26       C
 27              COMMON  /USERVARS/ SWITCHXY, ZSCALE, CATMAX, REFPTCAT, NOCAT
 28              DATA  SWITCHXY, ZSCALE / .TRUE., 10.333 /
 29       C      SWITCHXY - THIS IS A LOGICAL VARIABLE THAT MUST BE SET BY
 30       C                 THE USER. IF IT IS SET (.TRUE.), THEN THE
 31       C                 X AND Y COORDINATES ARE SWITCHED AFTER INPUT.
 32       C                 E.G. IF (100,35) IS THE POINT THAT IS READ AND
 33       C                 SWITCHXY IS SET, THEN THE POINT IS PROCESED AS
 34       C                 BEING (35,100).
 35       C      ZSCALE   - THIS VARIABLE MUST BE SET BY THE USER AND IS USED
 36       C                 TO SCALE ALL X,Y COORDINATES.
 37       C      CATMAX   - THE MAXIMUM VALUE A CATEGORY CAN BE WITHOUT
 38       C                 CAUSING THE PROGRAM TO ISSUE A WARNING MESSAGE.
 39       C      REFPTCAT - CATEGORY DESIGNATED BY THE DIGITIZERS FOR REFERENCE POI
 40       C                 WHEN SETTING UP THE DIGITIZER.
 41       C      NOCAT    - CATEGORY ASSIGNED BY THE DIGITIZERS TO TARGETS THAT WER
 42       C                 NOT GIVEN A CATEGORY
 43       C
 44       C
 45              COMMON  /TOTALS/ SINGLPTS, DUPLPTS, PTSREAD, PTSOUT
 46              DATA  SINGLPTS, DUPLPTS, PTSREAD, PTSOUT / 4*0 /
 47       C      SINGLPTS - THE TOTAL NUMBER SINGLE POINTS (FEATURE CODE IS
 48       C                 PT) THAT ARE OUTPUT.
 49       C      DUPLPTS  - THE TOTAL NUMBER OF DUPLICATE POINTS READ,
 50       C                 REGARDLESS OF THE FEARURE CODE.
 51       C      PTSREAD  - THE TOTAL NUMBER OF POINTS THAT HAVE BEEN READ,
 52       C                 REGARDLESS OF THE FEATURE CODE.
 53       C      PTSOUT   - THE TOTAL NUMBER OF POINTS WRITTEN TO TAPE
```

```
'4        C
'5        C
'6              COMMON  /STATS/ XMAX, XMIN, YMAX, YMIN
'7              DATA  XMAX, YMAX, XMIN, YMIN / 2*0, 2*100000 /
'8        C     XMAX - THE MAXIMUM X-COORDINATE OF FEATURE CODES AREA AND LINE
'9        C     XMIN - THE MINIMUM X-COORDINATE OF FEATURE CODES AREA AND LINE
'0        C     YMAX - THE MAXIMUM Y-COORDINATE OF FEATURE CODES AREA AND LINE
'1        C     YMIN - THE MINIMUM Y-COORDINATE OF FEATURE CODES AREA AND LINE
'2        C
'3        C
'4              COMMON  /PTSTATS/ XMAXPT, XMINPT, YMAXPT, YMINPT
'5              DATA  XMAXPT, YMAXPT, XMINPT, YMINPT / 2*0, 2*100000/
'6        C     XMAXPT - THE MAXIMUM X-COORDINATE OF FEATURE CODE PT.
'7        C     XMINPT - THE MINIMUM X-COORDINATE OF FEATURE CODE PT.
'8        C     YMAXPT - THE MAXIMUM Y-COORDINATE OF FEATURE CODE PT.
'9        C     YMINPT - THE MINIMUM Y-COORDINATE OF FEATURE CODE PT.
'0        C
'1        C
'2              COMMON  /CNCTSTAT/ TCONCTR,TCONPTS,MAXCONCT,CONPT,CONCTR
'3              DATA  TCONCTR, TCONPTS, MAXCONCT / 2*0, 200 /
'4        C     TCONCTR - THE TOTAL NUMBER OF CALLS TO SUBROUTINE "CONNECT"
'5        C     TCONPTS   - THE TOTAL NUMBER OF POINTS GENERATED BY SUBROUTINE
'6        C                 "CONNECT"
'7        C     MAXCONCT - THEIS VARIABLE MUST BE SET BY THE USER AND IS USED
'8        C                AS THE MAXIMUM NUMBER OF POINTS THAT CAN BE GENERATED
'9        C                BY ANY ONE CALL TO SUBROUTINE CONNECT WITHOUT ISSUING
'0        C                AN ERROR MESSAGE.
'1        C
'2        C
'3              COMMON  /IN/  INPUT, TOTINPUT, ENDMARK
'4              DATA INPUT, TOTINPUT, ENDMARK / 7, 1, -1 /
'5        C     INPUT     - FILE DESIGNATOR OF CURRENT INPUT FILE
'6        C                 THE FILE DESIGNATOR OF THE FIRST INPUT FILE MUST = 7
'7        C     TOTINPUT - TOTAL NUMBER OF INPUT FILES. MUST BE SET BY USER!
'8        C     ENDMARK  - VALUE USED TO MARK THE END OF A BOUNDARY ( USUALLY -1)
'9        C
'0        C
'1              COMMON  /ERRORS/ ERRLIST (4,19), ERRCOUNT
'2              DATA (ERRLIST(1,I), I=1,4) / 4*3 /
'3        C     ERRLIST - EACH OF THE FOUR ROWS OF THIS TWO-DIMENSIONAL ARRAY IS
'4        C               IS USED TO KEEP TRACK OF A DIFFERENT TYPE OF ERROR.
'5        C               COLUMNS 2-19 OF EACH ROW CONTAIN THE "UNIQUE" NUMBERS
'6        C               OF TARGETS IN WHICH AN ERROR WAS ENCOUNTERED. COLUMN 1
'7        C               OF EACH ROW CONTAINS AN INDEX TO THE NEXT AVAILABLE WORD
'8        C               IN THAT ROW. ONCE THE ROW IS FILLED, IT IS WRITTEN TO
'9        C               REPORT CODE 52 (CIA "PRINT") ALONG WITH AN APPROPRIATE
'0        C               MESSAGE. HERE ARE THE 4 TYPES OF ERRORS REMEMBERED:
'1        C                 ROW 1: SUBROUTINE "CONNECT" GENERATED MORE THAN
'2        C                        "MAXCONCT" POINTS WHEN CONNECTING TWO POINTS.
'3        C                 ROW 2: CATEGORY ERRORS
'4        C                 ROW 3: FEATURE CODE ERRORS
'5        C                 ROW 4: ERRORS WITH X,Y COORDINATES
'6        C
```

```
      C
      C
            DATA  LINE, AREA, PT, AREATAPE, PTLINTAP/ 1, 2, 3, 1, 2/
      C     AREA - FEATURE CODE FOR CLOSED BOUNDARIES
      C     LINE - FEATURE CODE FOR LINE TARGETS
      C     PT   - FEATURE CODE FOR POINT TARGETS
      C     AREATAPE - OUTPUT FILE DESIGNATOR FOR AREA TERGETS
      C     PTLINTAP - OUTPUT FILE DESIGNATOR FOR POINT AND LINE TARGETS
      C
      C
            DATA STRTUNIQ, DUMP, NODUMP / 1, .TRUE., .FALSE. /
      C     STRTUNIQ - THIS VARIABLE MUST BE SET BY THE USER! TO AID IN
      C                LATER PROCESSING, EACH LOGICAL GROUP IS ASSIGNED A
      C                A UNIQUE NUMBER ("UNIQUE"). "STRTUNIQ" DEFINES THE
      C                INITIAL VALUE OF "UNIQUE" FOR THIS RUN OF THE
      C                PROGRAM. THIS IS NECESSARY TO INSURE THE UNIQUENESS
      C                OF THE "UNIQUE"S WHEN THIS PROGRAM IS RUN MORE
      C                THAN ONCE FOR A SINGLE DATA BASE.
      C
            WRITE(6,1) 'R E S U L T S'
            PRINT 1, 'E R R O R S '
            CALL WRITER (AREATAPE,2,SWITCHXY,ENDMARK)
            IF (.NOT. SWITCHXY) GOTO 10
              WRITE(6,3)
              PRINT 3
      C
      10    UNIQUE = STRTUNIQ - 1
      C
      C
      C  THIS IS THE TOP OF THE LOOP THAT PROCESSES ALL OF THE INPUT TAPES
      C
      100   UNIQUE=UNIQUE + 1
      C
      C    GET CATEGORY AND FEATURE CODE. "GETCAT" RETURNS CONTROL TO STATEMENT
      C    LABELLED "300" UPON EOF OF ALL INPUT FILES
      C
              CALL  GETCAT (UNIQUE,CATGORY,FTCD,$300)
      C
      C       READ THE FIRST (X,Y) PAIR. IF THERE ARE NO POINTS FOR THIS BOUNDAR
      C     (I.E. (-1,-1) WAS READ) OR EOF WAS ENCOUNTERED, CONTROL IS RETURNED
      C     FROM "GETXY" TO THE STATEMENT LABELLED "200" TO OUTPUT A WARNING.
      C
              CALL GETXY (X,Y,ZSCALE,$200,$200)
      C
      C   WRITE BOUNDARY HEADER RECORD TO TAPE
              IF(FTCD .EQ. AREA) CALL WRITER (AREATAPE,3,UNIQUE,CATGORY,FTCD)
              IF(FTCD .NE. AREA) CALL WRITER (PTLINTAP,3,UNIQUE,CATGORY,FTCD)
      C
              IF (FTCD .EQ. PT)
             &      CALL POINT ( UNIQUE,X,Y,ZSCALE,PTLINTAP,$300)
              IF(FTCD .EQ. LINE)
             &    CALL AREALINE(UNIQUE,X,Y,ZSCALE,PTLINTAP,$300)
              IF(FTCD .EQ. AREA)
```

```
-79   20.768        I N I T F I X                                          LABE

160             &    CALL AREALINE(UNIQUE,X,Y,ZSCALE,AREATAPE,$300)
161              GOTO 100
162       C
163       C
164       C    COME HERE WHEN THE ENDING DELIMITER (-1,-1) IS READ AS THE FIRST
165       C   POINT OF ANY KIND OF BOUNDARY
166       C
167       200    WRITE(6,201)
168              CALL UPDATERR (4, UNIQUE, NODUMP)
169              GOTO 100
170       C    WHEN ALL PROCESSING IS DONE , WRITE OUT THE TOTALS.
171       C
172       300    WRITE (6,302) PTSOUT
173              WRITE(6,303) DUPLPTS
 74              WRITE(6,304) XMAX,XMIN,YMAX,YMIN
 75              WRITE(6,305) TCONCTR, TCONPTS
 76              WRITE(6,306) SINGLPTS
 77              WRITE(6,307) XMINPT, XMAXPT, YMINPT, YMAXPT
 78              DO 310 I = 1,4
 79       310      CALL  UPDATERR (I, 0, DUMP)
LOOP INDEX I MAY NOT BE REDEFINED IN CALL OR ABNORMAL FUNCTION
 80              WRITE(6,308) ERRCOUNT
 81              IF (ERRCOUNT .EQ. 0) PRINT 309
 82       C
 83       C
 84              STOP
 85       1      FORMAT ('1'///59X,'I N I T F I X'/59X,A13///)
 86       3      FORMAT (1X,40('*'),' NOTE: X & Y VALUES WILL BE SWITCHED',40('*'))
 87       201    FORMAT (' ***** THIS BOUNDARY HAD NO POINTS *****')
 88       302    FORMAT('TOTAL # OF PTS WRITTEN TO TAPE =',I5)
 89       303    FORMAT('TOTAL # OF DUPLICATE PTS =',I5)
 90       304    FORMAT('MAX AND MIN PTS OF ALL PTS WRITTEN TO TAPE =',4(I10,1X))
 91       305    FORMAT('TOTAL # OF CONNECTS =',I10,' &',I10,' TOTAL # PTS CNTED')
 92       306    FORMAT('TOTAL # OF INDIVIDUAL POINT TARGETS =',I10)
 93       307    FORMAT (15X,'LINE AND  POINT TARGETS',10X,'XMIN    XMAX    ',
 94              &         'YMIN    YMAX'/45X,4I8)
 95       308    FORMAT (///1X,I10,' ERRORS WERE ENCOUNTERED IN THIS RUN')
 96       309    FORMAT (1X,'***** NO ERRORS DISCOVERED *****')
 97              END
```

```
0-79   20.769        * * * * * * * *  POINT TARGET PROCESSING  * * * * * * * * *  LA

    1       C POINT     * * * * * * * *  POINT TARGET PROCESSING  * * * * * * * * *
    2       C
    3             SUBROUTINE  POINT (UNIQUE,TEMPX,TEMPY,ZSCALE,TAPEOUT,*)
    4       C
    5             IMPLICIT INTEGER  (A-Z)
    6       C
    7             COMMON  /IN / INPUT,TOTINPUT,ENDMARK
    8             COMMON  /TOTALS/ SINGLPTS, DUPLPTS, PTSREAD, PTSOUT
    9             COMMON  /PTSTAT/ XMAX, XMIN, YMAX, YMIN
   10       C
   11       C    POINT TARGET INITIALIZATIONS
   12             SINGLCTR = 0
   13             DUPCTR = 0
   14             READPT = 1
   15       C
   16       C
   17       C    THIS IS THE TOP OF THE LOOP WHICH PROCESSES POINT TARGETS
   18       C   CONTROL CAN EXIT THIS LOOP ONLY VIA AN ALTERNATE RETURN FROM THE
   19       C   SUBROUTINE "GETXY". UPON END-OF-FILE, CONTROL IS RETURNED TO THE
   20       C   STATEMENT LABELLED "340". WHEN A (-1,-1) IS READ BY "GETXY",
   21       C   CONTROL IS RETURNED TO THE STATEMENT LABELLED "350".
   22       300    CALL  GETXY ( X, Y, ZSCALE,  $340, $350)
   23             READPT = READPT + 1
   24       C
   25       C   OMIT DUPLICATE POINTS.
   26             IF (TEMPX .NE. X  .OR.  TEMPY .NE. Y) GOTO 330
   27               DUPCTR = DUPCTR + 1
   28               GO TO 300
   29       C
   30       C
   31       C    UPDATE MIN AND MAX X VALUES FOR SINGLE POINTS
   32       C
   33       330    IF (X .GT. XMAX) XMAX = X
   34              IF (Y .GT. YMAX) YMAX = Y
   35              IF (X .LT. XMIN) XMIN = X
   36              IF (Y .LT. YMIN) YMIN = Y
   37       C
   38              SINGLCTR = SINGLCTR + 1
   39              CALL WRITER (TAPEOUT,2,TEMPX,TEMPY)
   40       C
   41              TEMPX = X
   42              TEMPY = Y
   43              GOTO 300
   44       C
   45       C    THE INPUT FILE HAS REACHED THE END (EOF) BEFORE THE ENDING DELIMITER
   46       C      WAS READ. OUTPUT A MESSAGE AND CLOSE UP THIS LIST OF POINTS.
   47       C
   48       340    WRITE(6,341)TEMPX,TEMPY
   49        341   FORMAT(//,'<*><*><*> EOF BEFORE DELIMITER ON POINTS')
   50              CALL UPDATERR(4,UNIQUE,NODUMP)
   51       C
   52       C    THE ENDING DELIMETER -1,-1 IS WRITTEN TO TAPE.
   53       C
```

```
79   20.769          * * * * * * * * *  POINT TARGET PROCESSING  * * * * * * * * * *  LAB

54         350   CALL WRITER ( TAPEOUT,2,TEMPX,TEMPY)
55               CALL WRITER ( TAPEOUT,2,ENDMARK,ENDMARK)
56         C
57               SINGLPTS = SINGLPTS + SINGLCTR
58               PTSOUT = PTSOUT + SINGLCTR
59               PTSREAD = PTSREAD + READPT
60         C
61               WRITE(6,351)SINGLCTR,DUPCTR
62         351   FORMAT(/,10X,I6,' SINGLE POINTS ',I6,' DUPLICATES')
63               RETURN
64               END
```

```
      C
      C
      C
      C  * * * * * * *  PROCESSING FOR LINE AND AREA TARGETS  * * * * * * * *
      CAREALINE     A R E A  L I N E      PROCESS AREAS AND LINES
      C
            SUBROUTINE AREALINE (UNIQUE,TEMPX,TEMPY,ZSCALE,FILECD,*)
            IMPLICIT INTEGER (A-Y)
            COMMON / IN / INPUT,TOTINPUT,ENDMARK
            COMMON /TOTALS/ SINGLPTS,DUPLPTS,PTSREAD,PTSOUT
            COMMON /STATS/ TXMAX,TXMIN,TYMAX,TYMIN
            COMMON  /CNCTSTAT/ TCONCTR,TCONPTS,MAXCONCT,CONPT,CONCTR
            DATA PTLINE / 2/
            READPT = 1
            TAPECTR = 0
            DUPCTR = 0
            CONPT = 0
            CONCTR = 0
            CONAREA = TCONPTS
            XMAX = 0
            XMIN = 100000
            YMAX = 0
            YMIN = 100000
      C
      C     SAVE THE STARTING POINT
              XSTR =TEMPX
              YSTR = TEMPY
      C
      C     LOOP TO READ IN X,Y COORDINATES AND PROCESS THIS AREA OR LINE
      C
       510    CALL GETXY (X,Y,ZSCALE,$550,$540)
              READPT = READPT + 1
      C
      C     CHECK TO SEE IF DUPLICATE POINTS.  IF SO SKIP IT AND READ
      C     IN ANOTHER POINT.
      C
              IF(TEMPX .NE. X .OR. TEMPY .NE. Y)GO TO 530
              DUPCTR = DUPCTR + 1
              GO TO 510
      C
      C     CHECK THE ABSOLUTE DIFFERENCE BETWEEN X AND Y TO SEE IF
      C     A GAP IS PRESENT.  IF SO CALL CONNECT SUB WHICH FIXES THE
      C     AREA AS LONG AS THE GAP IS NOT OVER 10 PTS.  IF OVER 10 PTS.
      C     AN ERROR MESSAGE IS WRITTEN OUT.  THIS HELPS DECIDE WHETHER
      C     THE END DELIMETERS WERE FORGOTTEN.
      C
       530    CALL WRITER (FILECD,2,TEMPX,TEMPY)
              TAPECTR = TAPECTR +1
              IF(IABS(X-TEMPX) .GT. 1 .OR. IABS(Y-TEMPY) .GT. 1)CALL
             &   CONNECT(TEMPX,TEMPY,X,Y,FILECD)
              TEMPX = X
              TEMPY = Y
      C
```

```
      C
      C    CALCULATES THE MINUMUN AND MAXIMUM PTS. OF X AND Y.
      C
            IF(TEMPX .GT. XMAX)XMAX = TEMPX
            IF(TEMPX .LT. XMIN)XMIN = TEMPX
            IF(TEMPY .GT. YMAX)YMAX = TEMPY
            IF(TEMPY .LT. YMIN)YMIN = TEMPY
            GOTO 510
      C   END OF FILE WAS ENCOUNTERED ON THE INPUT TAPE BEFORE THE ENDING
      C     DELIMITER (-1,-1) WAS READ. OUTPUT A MESSAGE AND CLOSE UP THE
      C     OUPUT STRING
      C
      540   WRITE(6,541) TEMPX,TEMPY
       541  FORMAT(//,'<*><*><*> EOF BEFORE DELIMITER ON AREA')
            CALL UPDATERR(4,BNDCTR,NODUMP)
            RETURN 1
      C
      C
      C    THE TRAILER POINT (-1,-1) HAS BEEN READ. EXECUTION CONTINUES HERE
      C      TO FINISH UP THIS AREA OR LINE
      C
      550   IF (XSTR .NE. TEMPX  .OR.  YSTR .NE. TEMPY) GOTO 560
              DUPCTR = DUPCTR + 1
              GOTO 580
      C
      C    THE FINAL POINT AND THE STARTING POINT ARE DIFFERENT SO WRITE THE
      C      FINAL POINT (TEMPX, TEMPY) TO TAPE
      C
      560     CALL WRITER(FILECD,2,TEMPX,TEMPY)
              TAPECTR = TAPECTR + 1
              IF (FILECD .EQ. PTLINE) GOTO 580
              IF (IABS(TEMPX-XSTR) .LE. 1  .AND.  IABS(TEMPY-YSTR) .LE. 1)
             &  GOTO 580
              CALL CONNECT (TEMPX,TEMPY,XSTR,YSTR,FILECD)
            IF(IABS(TEMPX-XSTR).GT.MAXCONCT.OR.IABS(TEMPY-YSTR).GT.MAXCONCT)
           &    WRITE(6,561)IABS(TEMPX-XSTR),IABS(TEMPY-YSTR)
       561  FORMAT('<*><*><*>  START AND END POINTS ARE FAR APART',2I8)
      C
      C    THIS AREA (OR LINE) IS FINISHED. WRITE THE TRAILER X,Y (-1,-1) TO
      C      TAPE AND UPDATE OUTPUT TOTALS FOR THIS AREA OR LINE
      580   CALL  WRITER (FILECD,2,ENDMARK,ENDMARK)
            CONAREA = TCONPTS - CONAREA
            TAPECTR = TAPECTR + CONAREA
      C
      C    WRITE OUT ALL THE STATISTICS FOR EACH RECORD.
      C
            IF (TAPECTR .GT. 2) GOTO 590
              WRITE(6,581) TAPECTR
      581   FORMAT('<*><*><*>   ONLY TWO INPUT POINTS FOR THIS AREA')
              CALL UPDATERR(4,UNIQUE,NODUMP)
      590   WRITE(6,591)XSTR,YSTR,TEMPX,TEMPY
       591  FORMAT(25X,'START (X,Y) PT =',2I8,10X,'END (X,Y) PT =',2I8)
            WRITE(6,592)READPT,TAPECTR
```

```
:0-79   20.76?                                                          L

  107        592  FORMAT(25X,'# OF PTS READ IN =',I6,10X,
  108            &    '# OF PTS WRITTEN TO TAPE =',I8)
  109             WRITE(6,593) DUPCTR,CONCTR,CONAREA
  110        593  FORMAT(25X,'# OF DUPLICATES =',I6,10X,'CALLS TO CONNECT =',
  111            &    I5,' WITH ',I6,' POINTS CONNECTED')
  112             WRITE(6,594)XMIN,XMAX,YMIN,YMAX
  113       594   FORMAT(25X,' MIN AND MAX FOR X =',2I8,10X,'FOR Y =',2I8)
  114       C
  115       C    CALCULATING THE TOTAL'S OF ALL STATICISTICS.
  116       C
  117             PTSOUT = PTSOUT + TAPECTR
  118             DUPLPTS = DUPLPTS + DUPCTR
  119             TCONCTR = TCONCTR + CONCTR
  120             PTSREAD = PTSREAD + READPT
  121             IF (FILECD .EQ. PTLINE) RETURN
  122             IF(XMAX .GT.TXMAX)TXMAX = XMAX
  123             IF(XMIN .LT. TXMIN)TXMIN = XMIN
  124             IF(YMAX .GT. TYMAX)TYMAX = YMAX
  125             IF(YMIN .LT. TYMIN)TYMIN = YMIN
  126             RETURN
  127             END
```

```
      CONCT     C O N N E C T
            SUBROUTINE CONNECT (OX,OY,NX,NY,FC)
      C
            IMPLICIT INTEGER (A-Z)
            REAL X,Y,XDIRCTN,YDIRCTN,XSCALE,YSCALE
      C
            COMMON  /CNCTSTAT/ TCONCTR,TCONPTS,MAXCONCT,CONPT,CONCTR
      C
            CONPT = 0
            CONCTR=CONCTR+1
      C
            AX = IABS(OX-NX)
            AY = IABS(OY-NY)
            IF(AX .EQ. 0  .OR.  AY .EQ. 0) GOTO 100
            XDIRCTN = 1.
            IF (OX .GT. NX) XDIRCTN = -1.
            YDIRCTN = 1.
            IF (OY .GT. NY) YDIRCTN = -1.
      C
      C  SET "XSCALE", "YSCALE", "COUNT" FOR THE UPCOMING LOOP
      C
            IF (AX-AY) 10,100,20
      10      COUNT  = AX
              XSCALE = XDIRCTN
              YSCALE = YDIRCTN * FLOAT(AY) / FLOAT(AX)
              GOTO 30
      20      COUNT  = AY
              YSCALE = YDIRCTN
              XSCALE = XDIRCTN * FLOAT(AX) / FLOAT(AY)
      C
      C  LOOP TO CONNECT THE TWO POINTS USING SUBROUTINE "PATCH"
      C
      30    STARTX = OX
            STARTY = OY
            X = FLOAT(OX)
            Y = FLOAT(OY)
            DO 40 I = 1,COUNT-1
              X = X + XSCALE
              Y = Y + YSCALE
            IF(I .EQ. 1) GO TO 31
              CALL WRITER (FC,2,STARTX,STARTY)
              CONPT =   CONPT + 1
      C
      31      CALL PATCH(STARTX,STARTY,IFIX(X),IFIX(Y),FC)
              STARTX = IFIX(X)
              STARTY = IFIX(Y)
      40      CONTINUE
      C
            IF(STARTX .EQ. NX .AND. STARTY .EQ. NY)GO TO 41
              CALL WRITER (FC,2,STARTX,STARTY)
              CONPT =   CONPT + 1
      41    CALL PATCH(STARTX,STARTY,NX,NY,FC)
            GOTO 200
```

```
20-79   20.770      C O N N E C T

  54        C
  55        100   CALL PATCH(OX,OY,NX,NY,FC)
  56        C
  57        200   TCONPTS = TCONPTS + CONPT
  58                IF(CONPT .LT. MAXCONCT) RETURN
  59                WRITE(6,203) OX, NX, OY, NY, FC, CONPT
  60                CALL UPDATERR (1, BNDRY, NODUMP)
  61                RETURN
  62        C
  63        203   FORMAT ('***** TOO MANY CONNECTS   STARTX  STOPX   STARTY  ',
  64               &        ' STOPY  FILECD  CONNECTS'/24X,618//)
  65              END
```

```
20-79   20.770      S K I P  &  D U M P   SKIP AND DUMP OUT CATEGORY 1000          L

   1        C SKIP   S K I P  &  D U M P   SKIP AND DUMP OUT CATEGORY 1000
   2              SUBROUTINE  SKIPDUMP(BOUNDRY,INPUT,DUMP,CATGORY)
   3        C
   4              IMPLICIT INTEGER  (A-Z)
   5              LOGICAL  DUMP
   6              COMMON /JSERVARS/ SWITCHXY,ZSCALE
   7        C
   8        C
   9              PRINT 1000, BOUNDRY, CATGORY
  10              IF (DUMP) PRINT 1001
  11         10   CALL GETXY(X,Y,ZSCALE,$101,$101)
  12              IF (DUMP) PRINT 1002, X, Y
  13              IF (X .NE. -1) GOTO 10
  14         101  RETURN
  15        C
  16        1000  FORMAT (' **** BOUNDARY',I8,' WAS IGNORED. CATEGORY WAS',I8)
  17        1001  FORMAT (5X,'THE FOLLOWING PAIRS OF POINTS WERE IGNORED')
  18        1002  FORMAT (5X,I8,',',I8)
  19              END
```

```
20-79  20.770        P A T C H       FILL IN GAPS BETWEEN 2 POINTS                    L
    1       CPATCH      P A T C H       FILL IN GAPS BETWEEN 2 POINTS
    2       C
    3             SUBROUTINE  PATCH (X2,Y2,X,Y,FILECD)
    4       C
    5             IMPLICIT INTEGER(A-Y)
    6             COMMON  /CNCTSTAT/ TCONCTR,TCONPTS,MAXCONCT,CONPT,CONCTR
    7       C
    8       C   "CONCTR" IS A COUNTER THAT COUNTS THE NUMBER OF CALLS TO CONNECT.
    9       C   "CONPT" IS A COUNTER THAT COUNTS THE NUMBER OF POINTS GENERATED
   10       C   AND OUTPUT BY EACH CALL TO CONNECT.
   11       C   "MAXCONCT" IS A LIMIT VARIABLE THAT CONTROLS HOW LARGE "CONPT"
   12       C   CAN GROW UNTIL A WARNING MESSAGE IS OUTPUT.
   13       C   "TCONPTS" IS A COUNTER THAT COUNTS THE TOTAL NUMBER OF POINTS
   14       C   OUTPUT FROM AL CALLS TO CONNECT.
   15       C
   16       C
   17       C   COMPUTE THE ABSOLUTE DIFFERENCE BETWEEN THE TWO X AND Y VALUES.
   18       C
   19             AX=IABS(X2-X)
   20             AY=IABS(Y2-Y)
   21             IF(AX .LT. 2 .AND. AY .LT. 2) RETURN
   22             IF(X .GT. X2)XCHANGE=1
   23             IF( X .LT. X2)XCHANGE=-1
   24             IF( Y .GT. Y2)YCHANGE=1
   25             IF(Y .LT. Y2)YCHANGE=-1
   26             GOTO 110
   27       C
   28       C   CONNECT WORKS BY MODIFYING POINT (X2,Y2).  FIRST IT MAKES
   29       C   AX=AY (IE, IT MAKES THE ABSOLUTEE DIFFERENCE BETWEEN THE
   30       C   POINTS IN THE X- AND Y-DIRECTION THE SAME), AND THEN IT
   31       C   TRAVERSES THE DIAGONAL (IE, DECREMENTS BOTH X2 AND Y2 BY
   32       C   1 UNTIL (X2,Y2) = (X,Y)).
   33       C
   34       100   CALL WRITER (FILECD,2,X2,Y2)
   35             CONPT = CONPT + 1
   36       C
   37       C   RECOMPUTE "AX" AND "AY".
   38       C
   39       110   AX = IABS(X2-X)
   40             AY = IABS(Y2-Y)
   41       C
   42             IF(AX.LE.1 .AND. AY.LE.1)GOTO 250
   43       C
   44       C
   45             IF(AX-AY)100,222,150
   46       C
   47       C   THE CHANGE IN X IS GREATER THAN IN Y
   48       C   SO MODIFY ONLY X POSITION
   49       C
   50       150   X2 = X2  + XCHANGE
   51             GOTO 100
   52       C
   53       C   CHANGE IN Y GREATER THAN IN X
```

```
20-79   20.770        P A T C H       FILL IN GAPS BETWEEN 2 POINTS

   54       C  SO MODIFY ONLY Y POSITION
   55       C
   56       160    Y2 = Y2  + YCHANGE
   57              GOTO 100
   58       C
   59       C
   60       C
   61       C  DIFFERENCE IS SAME IN X AND Y, CHANGE BOTH DIRECTIONS.
   62       C  I.E. TRAVERSE THE DIAGONAL.
   63       C
   64       222    X2 = X2  + XCHANGE
   65              Y2 = Y2 + YCHANGE
   66              IF(X2 .EQ. X)GOTO 250
   67              CALL WRITER(FILECD,2,X2,Y2)
   68              CONPT  = CONPT  + 1
   69              GO TO 222
   70       C
   71       C  COME HERE WHEN DONE CONNECTING TWO POINTS.
   72       C
   73       250    RETURN
   74              END
```

```
20-79   20.771                                                          L

    1             SUBROUTINE  UPDATERR (TYPE, BNDRY, DUMP)
    2             IMPLICIT INTEGER  (A-Z)
    3             LOGICAL  DUMP
    4             COMMON  /ERRORS/ ERRLIST(4,19), ERRCOUNT
    5             DATA  CNCT,CATS,FTCD,PNTS/ 1,2,3,4 /
    6       C
    7       C
    8       C    UPDATE THE ERROR ARRAY AND ITS POINTER. IF THE ARRAY IS FULL DUMP
    9       C      OUT TO FILE CODE 52
   10       C
   11       C    IF A DUMP IS ALL THAT IS DESIRED, SKIP THIS SECTION
   12       C
   13             PTR = ERRLIST(TYPE,1)
   14             IF (DUMP) GOTO 200
   15             ERRCOUNT = ERRCOUNT + 1
   16       C
   17       C      MAKE SURE EACH PROBLEM BOUNDRY IS SAVED ONLY ONCE
   18       C
   19             IF (ERRLIST(TYPE,PTR-1) .EQ. BNDRY) RETURN
   20             ERRLIST(TYPE,PTR) = BNDRY
   21             ERRLIST(TYPE,1) = ERRLIST(TYPE,1) + 1
   22             IF (PTR .NE. 19) RETURN
   23       C
   24       C    THE ERROR ARRAY IS FULL, SO DUMP IT OUT
   25       C
   26       200   IF (PTR .EQ. 2) RETURN
   27               IF (TYPE .EQ. CNCT) PRINT 900
   28               IF (TYPE .EQ. CATS) PRINT 901
   29               IF (TYPE .EQ. FTCD) PRINT 902
   30               IF (TYPE .EQ. PNTS) PRINT 903
   31               STOP = 19
   32               IF (DUMP) STOP = PTR - 1
   33               PRINT 10, (ERRLIST(TYPE,I), I = 2,STOP)
   34               ERRLIST(TYPE,1) = 2
   35               RETURN
   36       C
   37       10    FORMAT ( 13 (1X,I6) )
   38       900   FORMAT (1X,'***** TOO MANY CONNECTS OCCURRED IN THESE BOUNDRIES')
   39       901   FORMAT (1X,'***** CATEGORY ERRORS OCCURRED IN THESE BOUNDRIES')
   40       902   FORMAT(1X,'***** FEATURE CODE ERRORS OCCURRED IN THESE BOUNDRIES')
   41       903   FORMAT(1X,'***** BAD X,Y VALUES WERE DETECTED IN THESE BOUNDRIES')
   42             END
```

```
:0-79   20.771   T  C A T    READ THE NEXT CATEGORY AND FEATURE CODE                L

      1       C  G E T  C A T    READ THE NEXT CATEGORY AND FEATURE CODE
      2       C
      3       C
      4       C      GETCAT READS THE CATEGORY "CATGORY" AND THE FEATURE CODE "FTCD"
      5       C    FROM "INPUT" FOR BOUNDARY NUMBER "UNIQUE". UPON REACHING END-OF-FILE
      6       C    ON "INPUT", CONTROL IS RETURNED VIA THE ALTERNATE RETURN.
      7       C
      8             SUBROUTINE  GETCAT (UNIQUE,CATGORY,FTCD,*)
      9             IMPLICIT INTEGER (A-Y)
     10             LOGICAL DUMP,NODUMP
     11             COMMON / IN / INPUT,TOTINPUT,ENDMARK
     12             DATA UNKNOWN,CATMAX,REFPTCAT / 9999, 9999, 1000 /
     13             DATA DUMP,NODUMP / .TRUE., .FALSE. /
     14       100   CALL GETXY(CATGORY,FTCD,1.,$800,$200)
     15             IF (CATGORY .NE. REFPTCAT) GOTO 110
     16               CALL SKIPDUMP(UNIQUE,INPUT,NODUMP,CATGORY)
     17               UNIQUE = UNIQUE + 1
     18               GOTO 100
     19       110   CONTINUE
     20       C
     21       C    THIS SECTION IS FOR CHANGING CATEGORIES
     22       C      E.G.
     23       C          IF (CATGORY .EQ. 9999) CATGORY = 1234
     24       C
     25             WRITE(6,111) CATGORY,UNIQUE,FTCD
     26       C
     27       C    "UNKNOWN" IS A CATCH-ALL CATEGORY ASSIGNED BY THE DIGITIZERS WHEN
     28       C   A BOUNDARY NOT GIVEN A CATEGORY IS ENCOUNTERED
     29             IF(CATGORY .NE. UNKNOWN) GOTO 120
     30             WRITE(6,112) CATGORY
     31             CALL UPDATERR (2, UNIQUE, NODUMP)
     32       C
     33       C      "CATMAX" IS THE MAXIMUM VALUE A CATEGORY MAY BE. ZERO IS ASSUMED T
     34       C    BE THE MINIMUM.
     35       120   IF (CATGORY .LE. CATMAX. AND. CATGORY .GE. 0) GOTO 130
     36             WRITE(6,121) CATGORY
     37             CALL UPDATERR (2, UNIQUE, NODUMP)
     38       C
     39       C      THE FEATURE CODE MUST BE:
     40       C        1 - POINT TARGET
     41       C        2 - LINE TARGET
     42       C        3 - CLOSED BOUNDARY
     43       130   IF(FTCD .GE. 1 .AND. FTCD .LE. 3) GOTO 140
     44             WRITE(6,131) FTCD
     45             CALL UPDATERR (3, UNIQUE, NODUMP)
     46       140   RETURN
     47       C
     48       C*** E O F ON "INPUT"
     49       200   RETURN 1
     50       C
     51       C    INPUT IS -1,-1  INSTEAD OF A CATEGORY AND FC
     52       C
     53        300  WRITE(6,801)
```

```
20-79   20.771   T  C A T    READ THE NEXT CATEGORY AND FEATURE CODE                L

   54        801  FORMAT('<*><*><*>  UNEXPECTED (-1,-1) FOUND')
   55             RETURN 1
   56       C
   57         111  FORMAT(/'  CATEGORY =',I8,5X,'UNIQUE =',I8,
   58            &   5X,'FEATURE CODE =',I5)
   59        112   FORMAT(' ******UNKNOWN CATEGORY******', I8)
   60        121   FORMAT(' *******BAD CATEGORY**********', I8)
   61        131   FORMAT(' ******BAD FEATURE CODE*******',I8)
   62              END


 03-31-77  **SR4J**

(SEC)       .32          LINES/MINUTE 11302

NO   DIAGNOSTICS IN ABOVE COMPILATION
WERE USED FOR THIS COMPILATION
```

```
20-79   20.771

     1              SUBROUTINE GETXY (X,Y,ZSCALE,*,*)
     2              IMPLICIT INTEGER (A-Y)
     3              COMMON / IN / INPUT,TOTINPUT,ENDMARK
     4              DIMENSION IN(2,150)
     5              DATA PTR,BUFSIZ/ 1,150 /
     6              DATA LASTFC / 0 /
     7        C
     8              IF ( LASTFC .EQ. 0) LASTFC = INPUT
     9              IF ( LASTFC .NE. INPUT) GOTO 800
    10              IF ( PTR .EQ. 1) READ(INPUT,END=900) IN
    11              X = IN(1,PTR)
    12              Y = IN(2,PTR)
    13              PTR = PTR+1
    14              IF ( PTR .GT. BUFSIZ) PTR = 1
    15              IF ( Y .EQ. ENDMARK) GOTO 200
    16        C
    17              X = FLOAT(X)/ZSCALE
    18              Y = FLOAT(Y)/ZSCALE
    19              RETURN
    20        C
    21         200  PTR = 1
    22              LASTFC = 0
    23              RETURN 1
    24        C
    25         800  WRITE(6,801) INPUT,LASTFC
    26         801  FORMAT('<*><*><*>  ATTEMPTED TO READ FROM FC',I3,
    27             &    ' BEFORE BUFFER FROM FC',I3,' WAS EMPTY')
    28              STOP
    29        C
    30         900  RETURN 2
    31              END


 03-31-77  **SR4J**

SEC)       .23          LINES/MINUTE  7769

NO   DIAGNOSTICS IN ABOVE COMPILATION
WERE USED FOR THIS COMPILATION
```

```
20-79   20.772                                                          LE

    1               SUBROUTINE WRITER ( FILECD,CNT,X1,X2,X3,X4,X5)
    2               IMPLICIT INTEGER (A-Z)
    3               COMMON /IN / INPUT,TOTINPUT,ENDMARK
    4               DIMENSION X(5), OUTBUFF(310)
    5               DATA PTR,BUFSIZ / 1,310 /
    6               DATA LASTFC /0/
    7       C
    8               X(1) = X1
    9               IF ( CNT .GE. 2) X(2) = X2
   10               IF ( CNT .GE. 3) X(3) = X3
   11               IF ( CNT .GE. 4) X(4) = X4
   12               IF ( CNT .GE. 5) X(5) = X5
   13               IF ( LASTFC .EQ. 0) LASTFC = FILECD
   14               IF ( LASTFC .NE. FILECD) GOTO 800
   15               IF ( PTR+CNT .LE. BUFSIZ) GOTO 100
   16               WRITE(FILECD)OUTBUFF
   17               PTR = 1
   18        100    DO 200 I=1,CNT
   19               OUTBUFF(PTR) = X(I)
   20        200    PTR = PTR + 1
   21      C
   22               IF ( X(CNT) .NE. ENDMARK) RETURN
   23               WRITE(FILECD) OUTBUFF
   24               PTR = 1
   25               LASTFC = 0
   26               RETURN
   27      C
   28        800    WRITE(6,801) FILECD,LASTFC
   29        801    FORMAT(' <*><*><*>  ATTEMPTED TO WRITE TO FC',I3,
   30            &    ' BEFORE DUMPING BUFFER FOR FC ',I3)
   31               STOP
   32               END


 03-31-77  **SR4J**

SEC)       .24          LINES/MINUTE  7887

NO   DIAGNOSTICS IN ABOVE COMPILATION
WERE USED FOR THIS COMPILATION
```

```
,-79   15.635          A  R  E  A      F  I  X          FIX UP AREAS                L

   1       CAREA          A  R  E  A      F  I  X          FIX UP AREAS
   2       C
   3       C
   4       C      THIS PROGRAM IS THE THIRD IN THE SERIES USED TO CREATE DATA BASES.
   5       C   ITS PURPOSE IS TO ELIMINATE TANGENTS, ELIMINATE VERTICAL SEGMENTS,
   6       C   ASSIGN A CLASSIFICATION OF "TOP" OR "BOTTOM" TO EACH POINT.
   7       C      EACH OF THESE TASKS IS ACCOMPLISHED BY USING 1 OF 4 DECISION
   8       C   MATRICES. IN ORDER TO CLASSIFY A POINT, (I.E. "TOP", "BOTTOM", OR
   9       C   "TOSS") IT IS NECESSARY TO KNOW THE POINT IMMEDIATELY PRECEDING IT
  10       C   AND THE POINT IMMEDIATELY FOLLOWING IT. FOR THIS REASON, THE FIRST
  11       C   POINT MUST BE SAVED ALONG WITH THE POSITIONAL DIFFERENCE BETWEEN
  12       C   IT AND THE SECOND POINT. IN THIS MANNER, THE FIRST POINT CAN THEN
  13       C   BE PROPERLY PROCESSED AS THE VERY LAST POINT.
  14       C      THERE ARE FOUR DECISION MATRICES; "ASSGN1",...,"ASSGN4". THE MATRI
  15       C   THAT IS USED IS DEPENDENT UPON POSITIONAL DIFFERENCES WITH RESPECT TO
  16       C   THE Y-DIRECTION OF THE THREE POINTS. ONCE THE CORRECT MATRIX HAS BEEN
  17       C   CHOSEN, THE CLASSIFICATION CAN BE EXTRACTED. THIS IS DEPENDENT UPON
  18       C   THE POSITIONAL DIFFERENCES WITH RESPECT TO THE X-DIRECTION OF THE
  19       C   THREE POINTS. THE DECISION MATRICES ARE DESIGNED SUCH THAT POINTS
  20       C   CREATING TANGENTS AND POINTS THAT FALL IN THE MIDDLE OF A VERTICAL
  21       C   SEGMENT ARE ASSIGNED THE CLASSIFICATION "TOSS". IF A POINT IS
  22       C   ASSIGNED AS "TOSS", IT SIMPLY IS NOT OUTPUT. POINTS ASSIGNED AS
  23       C   "TOP" OR "BOTTOM" ARE OUTPUT IN THE FORMAT SPECIFIED BELOW.
  24       C      ALL OF THESE TASKS ARE EXPLICITLY PERFORMED WITHIN THE SUBROUTINE
  25       C   "ASSIGN". THE PARAMATERS TO "ASSIGN" ARE: THE COORDINATES OF THE
  26       C   POINT TO BE PROCESSED; THE POSITIONAL DIFFERENCES BETWEEN THAT
  27       C   POINT AND THE LAST POINT PROCESSED; AND THE POSITIONAL DIFFERENCE
  28       C   BETWEEN THAT POINT AND THE NEXT POINT TO BE PROCESSED. PRIOR TO
  29       C   RETURN, THE POINT JUST PROCESSED IS UPDATED TO THE NEXT POINT TO BE
  30       C   PROCESSED AND THE POSITIONAL DIFFERENECES BETWEEN THE POINT JUST
  31       C   PROCESSED AND THE POINT PROCESSED BEFORE IT ARE UPDATED TO THE
  32       C   POSITIONAL DIFFERENCES BETWEEN THE NEXT POINT TO BE PROCESSED AND
  33       C   THE POINT JUST PROCESSED.
  34       C
  35       C
  36       C
  37       C    INPUT
  38       C        THERE IS ONE INPUT TAPE TO THIS PROGRAM (REFERENCED BY "TAPEIN")
  39       C      THIS TAPE IS THE OUTPUT TAPE FROM THE SECOND PROGRAM AND CONTAINS
  40       C      THE CATEGORY AND BOUNDRIES FOR EACH "AREA" IN THE DATA BASE.
  41       C
  42       C                    THE TAPE IS ARRANGED IN THIS FORMAT
  43       C
  44       C                          **********
  45       C                       CATEOGRY 1,FC,BOUNDRY#
  46       C                            (X,Y) PAIR
  47       C                            (X,Y) PAIR
  48       C                                .
  49       C                                .
  50       C                                .
  51       C                            (X,Y) PAIR
  52       C                             (-1, -1)
  53       C                            CATEGORY 2
```

```
      C                                  .
      C                                  .
      C                                  .
      C                              'EOF' MARKER
      C                            ******************
      C
      C
      C   OUTPUT
      C        THERE IS ONE OUTPUT TAPE (REFERENCED AS "TAPEOUT"), AND SOME
      C     OUPUT TO THE LINE PRINTER (REFERENCED AS "PAPER").
      C
      C    OUTPUT TAPE  :
      C       FOR EACH POINT WHICH IS TO BE KEPT (ASSIGNED AS "TOP" OR
      C     "BOTTOM") THERE ARE TWO WORDS OF OUTPUT. THE FIRST WORD CONTAINS
      C     THE BOUNDARY NUMBER IN THE UPPER HALF OF THE WORD (BITS 0 - 17)
      C     AND THE X-COORDINATE IN THE LOWER HALF OF THE WORD (BITS 18 - 35)
      C     THE SECOND WORD CONTAINS FOUR PIECES OF INFORMATION: 1) ONE BIT
      C     FOR "TOP" (=1) OR "BOTTOM" (=0) BIT ZERO, THE LEFTMOST BIT IS USED
      C     2) 17 BITS (1 - 17) FOR THE Y-COORDINATE. 3)THE CATEGORY IN BITS
      C     18 - 36
      C
      C                                    WORD 1
      C
      C                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3
      CO*1*2*3*4*5*6*7*8*9*0*1*2*3*4*5*6*7*8*9*0*1*2*3*4*5*6*7*8*9*0*1*2*3*4*5
      C*             0 - 17                    *            18 - 35
      C*       BOUNDARY NUMBER                 *         X - COORDINATE
      C*                                       *
      C
      C
      C                                    WORD 2
      C
      C                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3
      CO*1*2*3*4*5*6*7*8*9*0*1*2*3*4*5*6*7*8*9*0*1*2*3*4*5*6*7*8*9*0*1*2*3*4*5
      CO*             1 - 17                    *                18 - 35
      CT*          Y - COORDINATE               *           CATEGORY NUMBER
      CB*                                       *                              *
      C
      C     OUTPUT TO LINE PRINTER:
      C        UPON SUCCESFUL PROCESSING OF EACH AREA, THERE IS SOME OUPUT
      C       TO PAPER. TE INFORMATION PRINTED IS: THE AREA COUNT, THE
      C       CATEGORY, NUMBER OF POINTS DISCARDED, NUMBER OF POINTS CALLED
      C       "TOP", AND THE NUMBER OF POINTS CALLED "BOTTOM".
      C         UPON SUCCESFUL COMPLETION OF ALL AREAS, AN APPROPRIATE
      C       MESSAGE IS PRINTED. IF AN ERROR OCCURS, A DESCRIPTIVE MESSAGE
      C       IS PRINTED AND PROGRAM EXECUTION IS HALTED.
      C
      C*************************************************************************
      C   VARIABLE DESCRIPTIONS:
      C*************************************************************************
      C
      C  "TYPECNT"  IS AN ARRAY USED TO COUNT THE NUMBER POINTS DISCARDED, THE
      C                NUMBER OF POINTS CALLED "TOP", AND THE NUMBER OF POINTS
```

```
      C                 CALLED "BOTTOM".
      C
      C  "FIRSTX"  IS THE X-COORDINATE OF FIRST POINT IN THIS AREA
      C
      C  "FIRSTY"  IS THE Y-COORDINATE OF FIRST POINT IN THIS AREA
      C
      C  "X"  IS THE X-COORDINATE OF THE CURRENT POINT TO BE PROCESSED
      C
      C  "Y"  IS THE Y-COORDINATE OF THE CURRENT POINT TO BE PROCESSED
      C
      C  "NEXTX"  IS THE X-COORDINATE OF THE NEXT POINT TO BE PROCESSED
      C
      C  "NEXTY"  IS THE Y-COORDINATE OF THE NEXT POINT TO BE PROCESSED
      C
      C  "FRSTXDIF"  IS THE DIFFERENCE BETWEEN X-COORDINATES OF 1ST 2 POINTS
      C
      C  "FRSTYDIF"  IS THE DIFFERENCE BETWEEN Y-COORDINATES OF 1ST 2 POINTS
      C
      C  "CATGRY"  IS THE THE CATEGORY NUMBER FOR THE CURRENT AREA
      C
      C  "AREAS"  IS THE COUNT OF THE NUMBER OF AREAS PROCESSED
      C
      C    ** CONSTANTS **
      C
      C
      C  "END" = -1   USED TO CHECK FOR END-OF-AREA MARKER (-1,-1)
      C
      C  "TAPEIN" = 1  IS THE DEVICE NUMBER FOR THE INPUT TAPE
      C  "TAPEOUT" = 2  IS THE DEVICE NUMBER FOR THE OUTPUT TAPE
      C
      C  "PAPER" = 6  IS THE DEVICE NUMBER FOR THE LINE PRINTER
      C
      C*************************************************************************
      C
            IMPLICIT INTEGER  (A-Z)
            LOGICAL DUMP, NODUMP, SWITCHXY
      C
            DIMENSION  TYPECNT(3), GAPERRS(20), BTERRS(20), RNGERRS(20)
            DIMENSION  PNTERRS(20)
      C
            COMMON /INPARAM/ TAPEIN, ENDMARK
            COMMON   /STATS/ CATGRY, TYPECNT, SWITCHXY
            COMMON    /GAPS/ GAPERRS, ERGAPNTR, RNGERRS, ERRNGPTR
            COMMON /DCIDE/ ASSGN1(3,3), ASSGN2(3,3), ASSGN3(3,3), ASSGN4(3,3)
            COMMON  /ERRORS/ ERRCOUNT
      C
            DATA  DUMP, NODUMP /.TRUE., .FALSE. /
            DATA    TAPEIN, TAPEOUT, PAPER / 1, 2, 6 /
            DATA ENDMARK / -1/
      C
      C
      C   THESE DATA STATEMENTS FILL UP THE DECISION MATRICES
      C
```

```
4-79   15.635          A  R  E  A      F  I  X          FIX UP AREAS          LAB

 160            DATA  ((ASSGN1(I,J),I=1,3),J=1,3) /0,-1,-1,0,-1,-1,-1,1,1/
 161       C
 162            DATA  ((ASSGN2(I,J),I=1,3),J=1,3) /0,0,-1,-1,-1,1,-1,-1,1/
 163       C
 164            DATA  ((ASSGN3(I,J),I=1,3),J=1,3) /0,-1,-1,-1,1,1,-1,1,1/
 165       C
 166            DATA  ((ASSGN4(I,J),I=1,3),J=1,3) /0,0,-1,0,0,-1,-1,-1,1/
 167       C
 168            DATA  ERRCOUNT / 0 /
 169            DATA  PNTERRS(1),RNGERRS(1),GAPERRS(1),BTERRS(1) /4*3/
 170            DATA  PNTERRS(2),RNGERRS(2),GAPERRS(2),BTERRS(2) /-3,-2,-1,0/
 171       C
 172       C
 173       C  *** EXECUTION STARTS HERE ***
 174       C
 175       C*************************************************************************
 176       C
 177       C
 178            WRITE (PAPER,1000)
 179            PRINT 1000
 180            OUTCNT = 0
 181            MINX = 100000
 182            MAXX = 0
 183       C
 184       C
 185       C   READ THE VARIABLE "SWITCHXY" (LOGICAL) FOR TOPS & BOTTOMS
 186       C
 187            CALL GETXY ( SWITCHXY, DUM, $91, $420 )
 188       91   CONTINUE
 189            IF (SWITCHXY) WRITE(PAPER,3)
 190            IF (SWITCHXY) PRINT 3
 191       C
 192       C
 193       C  READ THE CATEGORY FOR THIS AREA. IF THE 'EOF' MARKER IS ENCOUNTERED,
 194       C    THEN ALL AREAS HAVE BEEN PROCESSED. EXECUTION CONTINUES AT THE
 195       C    STATEMENT LABELLED '300' TO DO SOME OUTPUT BEFORE STOPPING.
 196       C
 197        100 CONTINUE
 198            CALL GETCAT ( AREAS, CATGRY, FC, $400, $300 )
 199       C
 200            WRITE (PAPER,910) AREAS, CATGRY
 201       C
 202       C
 203       C
 204       C  READ THE FIRST TWO POINTS FOR THIS AREA, CHECKING FOR 'EOF' MARKER
 205       C    AND READ ERRORS.
 206       C
 207            CALL GETXY ( FIRSTX, FIRSTY, $112, $420 )
 208            IF (FIRSTX .GT. MAXX) MAXX = FIRSTX
 209            IF (FIRSTX .LT. MINX) MINX = FIRSTX
 210       C
 211            CALL GETXY ( X, Y, $112, $420)
 212            IF (X .GT. MAXX) MAXX = X
```

```
      IF (X .LT. MINX) MINX = X
      CALL WRITER (TAPEOUT, 2, AREAS, AREAS)
C
      DUPCNTR = 0
C
C  SAVE THE POSITIONAL DIFFERNECES BETWEEN THESE TWO POINTS
C
      FRSTXDIF = X - FIRSTX
      FRSTYDIF = Y - FIRSTY
      XDIFF = FRSTXDIF
      YDIFF = FRSTYDIF
C
C  LOOP TO HANDLE ALL POINTS, BUT THE FIRST & LAST, IN THIS AREA
C    FIRST, CHECK AND IGNORE DUPLICATE POINTS
C    NEXT, CHECK FOR END POINT (-1,-1), IF FOUND GOTO LABEL 200
C    FINALLY, CALL SUBROUTINE "ASSIGN" TO DEAL WITH THE POINT
C
110   CONTINUE
      CALL GETXY ( NEXTX, NEXTY, $200, $420 )
C
        IF(NEXTX .NE. X .OR. NEXTY .NE. Y) GOTO 111
          DUPCNTR = DUPCNTR + 1
          GOTO 110
C
C
111     IF (NEXTX .GT. MAXX) MAXX = NEXTX
        IF (NEXTX .LT. MINX) MINX = NEXTX
        CALL  ASSIGN (X, Y, XDIFF, YDIFF, NEXTX-X, NEXTY-Y)
C
          GOTO  110
C
112   WRITE (PAPER,916)
      CALL UPDATERR(PNTERRS, AREAS,NODUMP)
      GOTO 100
C
C*************************************************************************
C
C
C  ONCE THE END-OF-AREA MARKER HAS BEEN READ (-1,-1), THE VERY LAST
C    POINT IS NOW ("X","Y"). THIS POINT MUST BE PROCESSED FIRST AND
C    THEN THE VERY FIRST POINT CAN BE PROCESSED.
C
200   CALL  ASSIGN (X, Y, XDIFF, YDIFF, FIRSTX-X, FIRSTY-Y)
C
C  NOW THE FIRST POINT CAN FINALLY BE PROCESSED. IT IS HERE THAT
C    THE POSITIONAL DIFFERENCES BETWEEN IT AND THE SECOND POINT ARE
C    NEEDED.
C
      CALL  ASSIGN (X, Y, XDIFF, YDIFF, FRSTXDIF, FRSTYDIF)
      CALL WRITER ( TAPEOUT, 2, ENDMARK, ENDMARK )
C
C
C  THIS AREA HAS NOW BEEN COMPLETELY PROCESSED,  OUTPUT (TO PAPER)
```

```
      C     THE NUMBER OF POINTS DISCARDED, THE NUMBER OF POINTS CALLED "TOP",
      C     & THE NUMBER OF POINTS CALLED "BOTTOM". LOOP BACK FOR THE NEXT AREA
      C
            IF (TYPECNT(2) .EQ. TYPECNT(3))  GOTO 202
              WRITE(PAPER,911) IABS( TYPECNT(2) - TYPECNT(3) )
              CALL UPDATERR ( BTERRS,  AREAS, NODUMP )
      202   IF (TYPECNT(2) .GT. 3  .AND.  TYPECNT(3) .GT. 3) GOTO 203
              WRITE (PAPER,917)
              CALL UPDATERR(BTERRS, AREAS,NODUMP)
      203   WRITE (PAPER,912) (TYPECNT(I),I=1,3)
            IF (DUPCNTR .GT. 0) WRITE (PAPER,913) DUPCNTR
            OUTCNT = OUTCNT + TYPECNT(2) + TYPECNT(3)
            DO 204  I = 1,3
              TYPECNT(I) = 0
      204     CONTINUE
            GOTO 100
      C
      C*************************************************************************
      C
      C
      C   IF THE 'EOF' MARKER WAS ENCOUNTERED WHEN ATTEMPTING TO READ A NEW
      C     CATEGORY, THEN ALL AREAS HAVE BEEN SUCCESSFULLY PROCESSED
      C
      300   WRITE (PAPER,915)
      301   WRITE ( PAPER,320)
       320  FORMAT(//,20X,'>< >< >< >< >< >< >< >< >< >< >< >< >< ><')
            WRITE (PAPER,321) MINX, MAXX, OUTCNT, SWITCHXY
       321  FORMAT(//,20X,'THESE VALUES ARE NEEDED AS INPUT FOR COUNT'
           &   ,///,20X,'MINX MAXX OUTCNT SWITCHXY =',2I8,I10,I3)
            WRITE(6,320)
      C
            CALL UPDATERR(BTERRS,  0, DUMP)
            CALL UPDATERR (GAPERRS,  0, DUMP)
            CALL UPDATERR (RNGERRS,  0, DUMP)
            CALL UPDATERR (PNTERRS,  0, DUMP)
            WRITE (PAPER,914) ERRCOUNT
            IF (SWITCHXY) WRITE(6,3)
            IF (SWITCHXY) PRINT 3
            WRITE (PAPER,919) MAXX-MINX+21
            PRINT 919, MAXX-MINX+21
            STOP
      C
      C*************************************************************************
      C
      C
      C   WHEN A READ ERROR OCCURS (OTHER THAN 'EOF') WHILE READING A CATEGORY,
      C     EXECUTION CONTINUES HERE TO OUTPUT A MESSAGE AND THEN STOP.
      C
      400   WRITE (PAPER,920) AREAS
            GOTO 301
      C
      C*************************************************************************
      C
```

```
4-79   15.635          A R E A     F I X          FIX UP AREAS                LA

 319       C
 320       C  WHEN A READ ERROR OCCURS (OTHER THAN 'EOF') WHILE READING AN (X,Y)
 321       C    PAIR, EXECUTION CONTINUES HERE TO PRINT A MESSAGE AND STOP.
 322       C
 323       410   WRITE (PAPER,921) AREAS
TATEMENT IS NEVER REFERENCED
 324             GOTO 301
 325       C
 326       C*************************************************************************
 327       C
 328       C
 329       C  WHEN THE 'EOF' MARKER IS READ WHILE ATTEMPTING TO READ AN (X,Y) PAIR,
 330       C    EXECUTION CONTINUES HERE TO OUTPUT A MESSAGE AND STOP.
 331       C
 332       420   WRITE (PAPER,930) AREAS
 333             GOTO 301
 334       C
 335       C*************************************************************************
 336       C
 337       C  FORMATS
 338       C
 339       3     FORMAT (1X,25('*'),'TOPS AND BOTTOMS WILL BE SWITCHED',25('*'))
 340       910   FORMAT (/3X,'AREA ',I5,', CATEGORY IS ',I5)
 341       911   FORMAT (/1X,'****** BOTTOMS AND TOPS ARE NOT EQUAL. DIFFERENCE',
 342            &        ' IS',I5,' ******'/,'  ')
 343       912   FORMAT (8X,I7,' POINTS DISCARDED,',I7,' BOTTOMS,',I7,' TOPS.')
 344       913   FORMAT (8X,I7,' DUPLICATE POINTS')
 345       914   FORMAT (///5X,I10,' ERRORS WERE DETECTED IN THIS RUN')
 346       915   FORMAT (///1X,'****** NORMAL TERMINATION ******')
 347       916   FORMAT (1X,'***** (-1,-1) WAS READ AS FIRST OR SECOND POINT.',
 348            &        ' THE AREA WAS IGNORED')
 349       917   FORMAT (1X,'***** FEWER THAN 4 BOTTOMS OR FEWER THAN 4 TOPS')
 350       919   FORMAT (2(1X,120('*')//)/' *****  MAKE SURE  THE ARRAY "XBINS" IN',
 351            &        ' THE NEXT PROGRAM "XCOUNT" IS DIMENSIONED TO AT LEAST',
 352            &        I10,'   *****'//2(1X,120('*')//))
 353       1000  FORMAT ('1')
 354       C
 355       920   FORMAT (3(1X,100('*')//)/5X,'READ ERROR WHILE READING CATEGORY ',
 356            &        'FOR AREA ',I6)
 357       C
 358       921   FORMAT (3(1X,100('*')//)/5X,'READ ERROR WHILE READING X,Y IN ',
 359            &        'AREA ',I6)
 360       C
 361       930   FORMAT (3(1X,100('*')//)/5X,'END OF INPUT TAPE WHILE READING X,Y ',
 362            &       'IN AREA ',I6)
 363       C
 364             END
EMORY EXPANDED. USE $LIMITS OR CORE= OPTION FOR NEXT RUN


 03-31-77  **SR4J**
```

```
4-79   15.635          A  R  E  A     F  I  X          FIX UP AREAS          LAE

SEC)      1.33              LINES/MINUTE 16299

    2 DIAGNOSTICS IN ABOVE COMPILATION
WERE USED FOR THIS COMPILATION
```

```
      CASSIGN        A S S I G N     SUB. FOR CLASIFICATION & OUTPUT
      C
      C       S U B R O U T I N E      A S S I G N
      C
      C
      C     THIS SUBROUTINE DECIDES WHAT TO DO WITH A POINT. IT ASSIGNS A
      C  CLASSIFICATION OF "TOSS", "TOP", OR "BOTTOM" TO THE POINT USING
      C  ONE OF THE TWO DECISION MATRICES ("ASSGN1" OR "ASSGN2"). POINTS
      C  ASSIGNED "TOP" OR "BOTTOM" ARE OUTPUT IN THE TWO WORD FORMAT WHILE
      C  POINTS ASSIGNED "TOSS" ARE NOT OUPUT.
      C     THIS SUBROUTINE ALSO COUNTS THE NUMBER OF POINTS IN EACH X-BIN
      C  AND COUNTS THE NUMBER OF POINTS WITH EACH CLASSIFICATION.
      C    FINALLY, BEFORE RETURNING, THIS SUBROUTINE SWAPS THE ARGUMENTS
      C  AROUND IN PREPARATION FOR THE PROCESSING OF THE NEXT POINT.
      C
      C
      C*************************************************************************
      C  VARIABLE DESCRIPTIONS
      C*************************************************************************
      C
      C  "TYPECNT"  IS THE ARRAY FOR THE # OF POINTS WITH EACH DIFFERENT
      C               TYPE ("TOSS", "BOTTOM", OR "TOP")
      C
      C  "PNTYPE"      CLASSIFICATION OF ("X","Y")
      C                -1 = "TOSS"
      C                 0 = "BOTTOM"
      C                 1 = "TOP"
      C
      C  "PACKED"      IS THE SECOND WORD OF OUTPUT. IT IS PACKED TO HOLD:
      C                "TOP"/"BOTTOM",Y-COORDINATE, CATEGORY, AND FEATURE
      C                TYPE (= "AREA").
      C
      C  *** PARAMATERS ***
      C
      C  "X"  IS THE X-COORDINATE OF THE POINT TO BE PROCESSED
      C
      C  "Y"  IS THE Y-COORDINATE OF THE POINT TO BE PROCESSED
      C
      C  "OLDXDIFF"  IS THE DIFFERENCE IN X-COORDINATES BETWEEN THE
      C                POINT TO BE PROCESSED AND THE LAST POINT PROCESSED
      C
      C  "OLDYDIFF"  IS THE DIFFERENCE IN Y-COORDINATES BETWEEN THE
      C                POINT TO BE PROCESSED AND THE LAST POINT PROCESSED
      C
      C  "XDIFF"  IS THE DIFFERENCE IN X-COORDINATES BETWEEN THE NEXT
      C             POINT TO BE PROCESSED AND THE CURRENT POINT TO BE
      C             PROCESSED
      C
      C  "YDIFF"  IS THE DIFFERENCE IN Y-COORDINATES BETWEEN THE NEXT
      C             POINT TO BE PROCESSED AND THE CURRENT POINT TO BE
      C             PROCESSED
      C
      C  *** CONSTANTS ***
```

```
      C
      C  "AREA" = 2  IS THE FEATURE TYPE FOR AREAS
      C
      C  "TAPEOUT" = 2  IS THE DEVICE NUMBER FOR THE TAPE DRIVE WITH THE
      C                   OUTPUT TAPE MOUNTED
      C
      C  "TOSS" = -1  IS THE INTEGER ASSOCIATED WITH "TOSS"
      C
      C
      C  "ASSGN1" AND "ASSGN2" ARE THE DECISION MATRICES
      C    THESE MATRICES ARE USED TO ASSIGN "TOP", "BOTTOM", OR "TOSS" TO
      C    THE POINT "X","Y" .
      C
      C*************************************************************************
      C
      C
            SUBROUTINE ASSIGN (X, Y, OLDXDIFF, OLDYDIFF, XDIFF, YDIFF)
      C
            IMPLICIT INTEGER  (A-Z)
            LOGICAL  DUMP, NODUMP, SWITCHXY
      C
            DIMENSION TYPECNT(3), ERRLIST(19), RNGERRS(19)
      C
            COMMON   /STATS/ CATGRY, TYPECNT, SWITCHXY
            COMMON   /GAPS/ ERRLIST, ERRPNTR, RNGERRS, ERRNGPTR
            COMMON /DCIDE/ ASSGN1(3,3), ASSGN2(3,3), ASSGN3(3,3), ASSGN4(3,3)
      C
      C
      C
            DATA  DUMP, NODUMP /.TRUE., .FALSE./
            DATA  TOSS, AREA, TAPEOUT, PAPER / -1, 2, 2, 6 /
      C
      C
      C *****  EXECUTION STARTS HERE  *****
      C
      C*************************************************************************
      C
      C
      C  DECIDE WHICH MATRIX TO USE BY THE DIFFERENCES IN Y-COORDINATES
      C    AND EXTRACT THE CLASSIFICATION FROM THE MATRIX BY THE DIFFERENCES
      C    IN X-COORDINATES
      C
      C
      C    FIRST MAKE SURE THERE IS NOT A GAP BETWEEN THE THREE POINTS
      C
            IF (IABS(OLDXDIFF) .GT. 1  .OR.  IABS(XDIFF) .GT. 1 .OR.
           &    IABS(OLDYDIFF) .GT. 1  .OR.  IABS(YDIFF) .GT. 1) GOTO 200
      C
            IF (YDIFF + OLDYDIFF) 120, 100, 110
      C
      100     IF (YDIFF) 103, 110, 104
        103     PNTYPE = ASSGN3(OLDXDIFF+2, XDIFF+2)
                GOTO 130
```

```
04-79  15.638          A S S I G N     SUB. FOR CLASIFICATION & OUTPUT          L

  107         104      PNTYPE = ASSGN4(OLDXDIFF+2, XDIFF+2)
  108                  GOTO 130
  109       C
  110       110      PNTYPE = ASSGN1(OLDXDIFF+2, XDIFF+2)
  111                GOTO 130
  112       C
  113       120      PNTYPE = ASSGN2(OLDXDIFF+2, XDIFF+2)
  114       C
  115       130    IF (PNTYPE .EQ. TOSS) GOTO 140
  116       C
  117       C
  118       C    IF THE LOGICAL VARIABLE "SWITCHXY" IS SET TO .TRUE., THEN THE
  119       C   X AND Y VALUES WERE SWITCHED IN THE PROGRAM "INITFIX" AND THUS
  120       C   TOPS AND BOTTOMS MUST BE SWITCHED HERE
  121       C
  122            IF (SWITCHXY)  PNTYPE = 1 - PNTYPE
  123       C
  124       C   THE POINT (X,Y) HAS BEEN CLASSIFIED AS A "TOP" OR A "BOTTOM"
  125       C     SO OUTPUT IT IN THE PRESCRIBED FORMAT
  126       C
  127              PACKED2 = ILS(PNTYPE,35) + ILS(Y,18) + CATGRY
  128              CALL WRITER ( TAPEOUT, 2, X, PACKED2 )
  129       C
  130       C
  131       C   THE POINT JUST PROCESSED ("X","Y") IS NO LONGER NEEDED. THE
  132       C     DIFFERENCE BETWEEN IT AND THE NEXT POINT ("XDIFF" AND "YDIFF") ARE
  133       C     STILL NEEDED. THE NEXT POINT CAN READILY BE CALCULATED SO IT IS
  134       C     DONE HERE IN ANTICIPATION OF THE NEXT CALL.
  135       C
  136       140    X = X + XDIFF
  137              Y = Y + YDIFF
  138              OLDXDIFF = XDIFF
  139              OLDYDIFF = YDIFF
  140       C
  141       C   UPDATE THE DIFFERENT TYPE COUNT
  142       C
  143              PNTYPE = PNTYPE + 2
  144              TYPECNT(PNTYPE) = TYPECNT(PNTYPE) + 1
  145              RETURN
  146       C
  147       C   A GAP WAS FOUND. OUTPUT A MESSAGE AND UPDATE THE PARAMATERS
  148       C
  149       200    WRITE (PAPER,900) X-OLDXDIFF, X, X+XDIFF, Y-OLDYDIFF, Y, Y+YDIFF,
  150            &                 OLDXDIFF, XDIFF, OLDYDIFF, YDIFF
  151              CALL UPDATERR (ERRLIST,  AREAS, NODUMP)
  152              X = X + XDIFF
  153              Y = Y + YDIFF
  154              OLDXDIFF = XDIFF
  155              OLDYDIFF = YDIFF
  156              RETURN
  157       C
  158       900    FORMAT (1X,'*************** GAP *************'/10X,
  159            &         2('  PREVIOUS',16X,'THIS',16X,'NEXT',10X)/10X,
```

```
4-79   15.638              A S S I G N     SUB. FOR CLASIFICATION & OUTPUT          LAB

 160            &          3(9X,'X',10X),3(9X,'Y',10X)/10X,6(I10,10X)/,
 161            &          2(20X,2(I10,10X)))
 162       901   FORMAT (1X,'***** X VALUE',I8,'  IS NOT BETWEEN',I8,' &',I8)
 163            END


 03-31-77  **SR4J**

SEC)       .61          LINES/MINUTE 15847

NO   DIAGNOSTICS IN ABOVE COMPILATION
WERE USED FOR THIS COMPILATION
```

```
04-79   15.639        U P D A T E   E R R O R S                    LA

    1       CUPDATE     U P D A T E   E R R O R S
    2       C
    3             SUBROUTINE  UPDATERR (ARRAY, AREA, DUMP)
    4             IMPLICIT INTEGER (A-Z)
    5             LOGICAL  DUMP
    6       C
    7             COMMON  /ERRORS/ ERRCOUNT
    8             DIMENSION  ARRAY (20)
    9       C
   10       C
   11       C
   12             PNTR = ARRAY(1)
   13             IF (DUMP) GOTO 100
   14             ERRCOUNT = ERRCOUNT + 1
   15             IF (ARRAY(PNTR-1) .EQ. AREA) RETURN
   16             ARRAY(PNTR) = AREA
   17             ARRAY(1) = ARRAY(1) + 1
   18             IF ( PNTR .NE. 20) RETURN
   19       100   IF (PNTR .EQ. 3) RETURN
   20               IF (ARRAY(2) .EQ. -3) PRINT 903
   21               IF (ARRAY(2) .EQ. -2) PRINT 904
   22               IF (ARRAY(2) .EQ. -1) PRINT 901
   23               IF (ARRAY(2) .EQ. 0) PRINT 902
   24               STOP = 20
   25               IF (DUMP) STOP = PNTR - 1
   26               PRINT 900, (ARRAY(I), I = 3,STOP)
   27               PNTR = 3
   28               RETURN
   29       900   FORMAT (18(1X,I6))
   30       901   FORMAT (1X,'***** GAPS OCCURRED IN THESE AREAS')
   31       902   FORMAT (1X,'***** PROBLEMS WITH BOTTOMS & TOPS IN THESE AREAS')
   32       903   FORMAT (1X,'***** X,Y PROBLEMS OCCURRED IN THESE AREAS')
   33       904   FORMAT (1X,'***** X VALUE OUT OF RANGE IN THESE AREAS')
   34             END


 03-31-77  **SR4J**

SEC)       .25          LINES/MINUTE  7973

NO   DIAGNOSTICS IN ABOVE COMPILATION
WERE USED FOR THIS COMPILATION
```

```
4-79   15.639                                                    LAB

   1            SUBROUTINE GETCAT (UNIQUE,CAT,FTCD,*,*)
   2            IMPLICIT INTEGER (A-Y)
   3            COMMON / INPARAM / INPUT,ENDMARK
   4            DIMENSION IN(310)
   5            DATA PTR,BUFSIZ/ 1,310 /
   6            DATA LASTFC / 0 /
   7      C
   8            IF ( LASTFC .NE. 0) GOTO 800
   9            LASTFC = INPUT
  10            READ(INPUT,END=900) IN
  11            UNIQUE = IN(1)
  12            CAT = IN(2)
  13            FTCD = IN(3)
  14            PTR = 4
  15            RETURN
  16      C
  17            ENTRY GETXY ( X, Y, *, * )
  18            IF ( LASTFC .EQ. 0) LASTFC = INPUT
  19            IF ( LASTFC .NE. INPUT) GOTO 800
  20      120   IF ( PTR .EQ. 1) READ(INPUT,END=900) IN
  21            IF ( PTR+2 .LE. BUFSIZ ) GOTO 125
  22            PTR = 1
  23            GOTO 120
  24      C
  25      125   X = IN(PTR)
  26            Y = IN(PTR+1)
  27            PTR = PTR+2
  28            IF ( Y .NE. ENDMARK) RETURN
  29      C
  30            PTR = 1
  31            LASTFC = 0
  32            RETURN 1
  33      C
  34       800  WRITE(6,801) INPUT,LASTFC
  35       801  FORMAT('<*><*><*>  ATTEMPTED TO READ FROM FC',I3,
  36           &   '  BEFORE BUFFER FROM FC',I3,' WAS EMPTY')
  37            STOP
  38      C
  39       900  RETURN 2
  40            END


 03-31-77  **SR4J**

SEC)      .26          LINES/MINUTE  8919

NO   DIAGNOSTICS IN ABOVE COMPILATION
WERE USED FOR THIS COMPILATION
```

```
14-79   15.640                                                          L

    1               SUBROUTINE WRITER ( FILECD,CNT,X1,X2,X3,X4,X5)
    2               IMPLICIT INTEGER (A-Z)
    3               COMMON /INPARAM / INPUT,ENDMARK
    4               DIMENSION X(5), OUTBUFF(310)
    5               DATA PTR,BUFSIZ / 1,310 /
    6               DATA LASTFC /0/
    7       C
    8               X(1) = X1
    9               IF ( CNT .GE. 2) X(2) = X2
   10               IF ( CNT .GE. 3) X(3) = X3
   11               IF ( CNT .GE. 4) X(4) = X4
   12               IF ( CNT .GE. 5) X(5) = X5
   13               IF ( LASTFC .EQ. 0) LASTFC = FILECD
   14               IF ( LASTFC .NE. FILECD) GOTO 800
   15               IF ( PTR+CNT .LE. BUFSIZ) GOTO 100
   16               WRITE(FILECD)OUTBUFF
   17               PTR = 1
   18        100    DO 200 I=1,CNT
   19               OUTBUFF(PTR) = X(I)
   20        200    PTR = PTR + 1
   21      C
   22               IF ( X(CNT) .NE. ENDMARK) RETURN
   23               WRITE(FILECD) OUTBUFF
   24               PTR = 1
   25               LASTFC = 0
   26               RETURN
   27      C
   28        800    WRITE(6,801) FILECD,LASTFC
   29        801    FORMAT(' <*><*><*>  ATTEMPTED TO WRITE TO FC',I3,
   30              &    ' BEFORE DUMPING BUFFER FOR FC ',I3)
   31               STOP
   32               END


 03-31-77  **SR4J**

SEC)       .25          LINES/MINUTE  7574

NO   DIAGNOSTICS IN ABOVE COMPILATION
WERE USED FOR THIS COMPILATION
```

```
      C COUNT
      C              COUNT NUMBER OF POINTS IN EACH XBIN
      C
      C   INPUT
      C     FILE CODE
      C       01       INPUT DATA IN THE FOLLOWING FORMAT
      C                ( BOUNDARY NUMBER )
      C     ( X-COORDINATE, [Y-COORDINATE,CATEGORY,FTCD] )
      C     ( X-COORDINATE, [Y-COORDINATE,CATEGORY,FTCD] )
      C                          .
      C                          .
      C                          .
      C                    ( -1, -1 )
      C                ( BOUNDARY NUMBER )
      C                          .
      C                          .
      C                          .
      C                        EOF
      C
      C       02       ONE 4 WORD RECORD AS FOLLOWS
      C       ( MINX, MAXX, TOTAL POINTS, "OUTBKWRD" (LOGICAL VARIABLE) )
      C
      C   OUTPUT
      C     FILE CODE
      C       03        THE INPUT ARRANGED IN THE FOLLOWING FORMAT
      C                    (EXCLUDING ANY AREAS DELETED)
      C
      C       ( [BOUNDARY #,X-COORD], [Y-COORD,CATEGORY #,FTCD])
      C       ( [BOUNDARY #,X-COORD], [Y-COORD,CATEFORY #,FTCD])
      C                                  .
      C                                  .
      C                                  .
      C                              ( -1, -1 )
      C       ( [BOUNDARY #,X-COORD], [Y-COORD,CATEGORY #,FTCD] )
      C                                  .
      C                                  .
      C                                  .
      C                                EOF
      C
      C       04        ONE 4 WORD RECORD IDENTICAL TO THE RECORD ON INPUT
      C                 02, PERHAPS MODIFIED THROUGH DELETES
      C              THE X-BIN COUNTS, TEN PER RECORD
      C                IF THE BOOLEAN VARIABLE "OUTBKWRD" IS SET .TRUE.,
      C                  THEN THE X-BIN COUNTS ARE WRITTEN OUT FROM
      C                  MAXIMUM X TO MINIMUM X.
      C                IF THE BOOLEAN VARIABLE "OUTBKWRD" IS SET .FALSE.,
      C                  THEN THE XBIN COUNTS ARE WRITTEN OUT FROM
      C                  MINIMUM X TO MAXIMUM X.
      C
      C              NOTE: THE BOOLEAN VARIABLE "OUTBKWRD" IS PASSED
      C                    IN THE RECORD OF INPUT 02 FROM THE
      C                    PRECEDING PROGRAM.
```

```
      C
      C
            IMPLICIT INTEGER  ( A-Z )
      C
            LOGICAL  OUTBKWRD
            DIMENSION  XBIN (3800)
            DIMENSION  DELE (1)
            COMMON BUFF(155,2),BUFPTR
      C
            DATA  MASK / 0777777 /  XRANGE, LASTDELE / 3800,1 /
            DATA  DELE / 0 /
      C
      C
            BUFPTR = 1
            READ (02)  MINX, MAXX, NUMPTS, OUTBKWRD
            IF (OUTBKWRD) WRITE (6,3)
            WRITE (6,1) 'INPUT ', MINX, MAXX, NUMPTS
            IF (MAXX-MINX+21 .GT. XRANGE) GOTO 200
      C
            DO 5  I=1,XRANGE
      5       XBIN(I) = 0
      C
            OFFSET = MINX - (XRANGE - MAXX + MINX)/2
            PNTSREAD = 0
            SKIPPED = 0
      C
      7     READ(01,END=100) BOUNDARY
              DO 8  I = 1,LASTDELE
                IF (BOUNDARY .NE. DELE(I)) GOTO 8
      9           READ (01,END=100) X,Y
                    IF (X .EQ. -1) GOTO 7
                    SKIPPED = SKIPPED + 1
                    GOTO 9
      C
      8         CONTINUE
      C
      10    READ (01,END=100)  X, Y
              IF (X .EQ. -1) GOTO 20
              XBIN (X-OFFSET) = XBIN (X-OFFSET) + 1
              PACKX = ILS(BOUNDARY,18) + X
              CALL DUMPB(PACKX,Y)
              PNTSREAD = PNTSREAD + 1
              GOTO 10
      C
       20   CALL DUMPB(-1,-1)
            GOTO 7
      C
      C
      100   TOTIN = PNTSREAD + SKIPPED
            START = 1
      101   IF (XBIN(START) .NE. 0) GOTO 102
              START = START + 1
```

```
105                 GOTO 101
106         C
107         102     MINX = START + OFFSET
108                 STOP = XRANGE
109         103     IF (XBIN(STOP) .NE. 0) GOTO 104
110                   STOP = STOP - 1
111                   GOTO 103
112         C
113         104     MAXX = STOP + OFFSET
114         C
115         C
116                 IF(BUFPTR .NE. 1) WRITE(03) BUFF
117         C
118         C
119                 WRITE(6,1) 'OUTPUT',MINX,MAXX,PNTSREAD
120                 IF (SKIPPED .NE. 0) WRITE(6,2) SKIPPED
121                 IF (TOTIN .LT. NUMPTS) WRITE (6,111) NUMPTS - TOTIN
122                 IF (TOTIN .GT. NUMPTS) WRITE (6,112) TOTIN - NUMPTS
123                 IF (X .NE. -1) WRITE (6,113) X, Y
124         C
125                 WRITE(04) MINX,MAXX,PNTSREAD,OUTBKWRD
126         C
127         C
128                 IF (.NOT. OUTBKWRD) GOTO 150
129         C
130                 DO 110   J = START, STOP, 10
131                   K = STOP + START - J + 1
132         110       WRITE(04) (XBIN(K-L), L = 1,10)
133                 STOP
134         150     DO 151  I=START,STOP,10
135         151       WRITE(04) (XBIN(J),J=I,I+9)
136                 STOP
137         1       FORMAT (///1X,A6/4X,'MINIMUM, MAXIMUM, TOTAL POINTS:',
138                &         I8,',',I8,',',I10)
139         3       FORMAT (1X,25('*'),' XBINS WERE WRITTEN IN DESCENDING ORDER')
140         2       FORMAT (1X,'*****',I8,' POINTS WERE DELETED')
141         111     FORMAT (' ***** EXPECTING',I10,' MORE POINTS AT EOF ')
142         112     FORMAT (' *****',I10,' UNEXPECTED POINTS READ AND COUNTED')
143         113     FORMAT (' ***** DID NOT END WITH (-1,-1) TRAILER POINT')
144         C
145         200     WRITE (6,201) MAXX-MINX+21
146                 STOP
147         201     FORMAT (' ********** THE ARRAY "XBIN" MUST BE DIMENSIONED AT',I10)
148                 END
EMORY EXPANDED. USE $LIMITS OR CORE= OPTION FOR NEXT RUN
```

```
      SUBROUTINE DUMPB( X,Y)
      IMPLICIT INTEGER (A-Z)
      COMMON BUFF(155,2),BUFPTR
C
      BUFF(BUFPTR,1) = X
      BUFF(BUFPTR,2) = Y
      BUFPTR = BUFPTR + 1
      IF(BUFPTR .LT. 156) RETURN
C
      WRITE(03) BUFF
      DO 100 I=1,155
      DO 100 J=1,2
  100 BUFF(I,J) = 0
      BUFPTR = 1
      RETURN
      END
```

```
78   02.353              Y S O R T                                    LABE

 1       C                  Y S O R T
 2       C
 3       C  THIS PROGRAM IS A MODIFICATION OF XSORT.
 4       C  THE PURPOSE IS TO SORT SOME INPUT DATA ACCORDING TO THE X-VALUE
 5       C  ASSOCIATED WITH THAT DATA.
 6       C
 7       C  IN THIS MODIFIED VERSION WE ARE ACTUALLY SORTING ACCORDING TO
 8       C  Y-VALUES FROM  LARGEST   TO   SMALLEST .
 9       C
10       C  RATHER THAN CHANGE ALL VARIABLE NAMES, ALL REFERENCES TO 'X'
11       C  WERE LEFT UNCHANGED.
12       C  THE CHANGES TO ALLOW SORTING FROM LARGEST TO SMALLEST INSTEAD OF
13       C  SMALLEST TO LARGEST ARE COMMENTED.
14       C     ALSO THE PRMFL MUST CONTAIN THE COUNTS FOR X VALUES FROM LARGEST
15       C  TO SMALLEST AS WELL.
16       C
17              IMPLICIT INTEGER (A-Z)
18              LOGICAL BIG2SMAL
19              DIMENSION XBIN(2000),TEST(2000),IN(2,36000)
20              COMMON BUFF(155,2),BUFPTR
21              DATA PRMFL,INTAP,OTAP /23,1,2/
22              DATA FC/2/
23       C
24       C                 <*<*<<*<*<*<*<*<*<*<        *>*>*>*>*>*>*>*>*>
25       C
26       C  LIMIT -- SHOULD BE SET TO THE DIMENSION OF 'IN(2,?)'
27       C
28              DATA NUMBIN,LIMIT,NUMPASS / 2000,36000,10/
29              DATA LOWMSK / 0777777/
30              DATA FLIP / 0400000000000/
31       C
32       C                 <*<*<*<*<*<*<<*<*<*<        *>*>*>*>*>*>*>*>*>
33       C
34       C
35       C  IN  -- ARRAY FOR DATA SORTED ACCORDING TO X - VALUES, WHICH IS THE
36       C         PROGRAM OUTPUT.  EACH DATA VALUE CONSISTS OF 2 WORDS.
37       C
38       C XBIN(J) - CONTAINS THE PROPER INDEX FOR ARRAY 'IN' IN WHICH TO PLACE T
39       C           NEXT DATA VALUE WHOSE X-VALUE IS 'J'.
40       C           SO IF ONE READS IN  ' X , DATA ' THE DATA IS STORED AT
41       C           IN ( XBIN (X) ), AND  XBIN(X) <- XBIN(X)+1 SO THAT THE
42       C           NEXT DATA READ IN WITH THE SAME X-VALUE WILL BE STORED
43       C           IN THE NEXT LOCATION IN 'IN'.
44       C
45       C TEST -- AUXILIARY ARRAY USED IN CONJUNCTION WITH XBIN SO THAT A SPECIA
46       C         FLAG CAN BE SET FOR THE FIRST DATA VALUE OF A BOUNDARY IN EACH
47       C         XBIN.  THIS ALLOWS US TO RECOGNIZE THE DATA IN AN XBIN THAT BE
48       C         TO THE SAME BOUNDARY.
49       C
50       C              THE PRMFL IS ASSUMED TO HAVE THE COUNT OF THE NUMBER OF
51       C  DATA VALUES FOR EACH XBIN, IN SEQUENCE. TO CONSERVE FILE SPACE, EACH
52       C  RECORD (IN BINARY UNFORMATTED) CONTAINS THE COUNT FOR 10 X-VALUES.
```

```
53       C
54               PASS = 0
55               EOF = 0
56       C
57       C BIG2SMAL -- LOGICAL VARIABLE WHICH INDICATES IF WE SHOULD FROM SMALLES
58       C                T0 LARGEST OR LARGEST TO SMALLEST.
59       C
60               READ(PRMFL)FIRSTX,MAXX,TOTPTS,BIG2SMAL
61               WRITE(6,66) FIRSTX,MAXX,TOTPTS,BIG2SMAL
62        66     FORMAT(' INPUT PARAMETERS ARE -  FIRSTX,MAXX,TOTPTS,',
63            &  'BIG2SMAL =',3I8,L2)
64               WRITE(OTAP)FIRSTX,MAXX
65               STRT = FIRSTX-1
66               IF(BIG2SMAL ) STRT = 0
67               STOP = STRT + 2
68         5     XBIN(1) = 1
69               INDEX = STOP-STRT
70         6     READ(PRMFL,END = 900) (XBIN(K),K=INDEX,INDEX+9)
71               DO 77 K=1,10
72               XBIN(INDEX) = XBIN(INDEX-1) + XBIN(INDEX)
73               IF(XBIN(INDEX) .GT. LIMIT) GOTO 100
74               IF(INDEX .GE. NUMBIN) GOTO 100
75               INDEX = INDEX+1
76               STOP = STOP+1
77               IF(STOP .GT. MAXX) GOTO 900
78         77    CONTINUE
79               GOTO 6
80       C
81       C WE CAN HANDLE ONLY XBINS 'STRT' THROUGH 'INDEX' WITHOUT OVERFLOWING
82       C THE OUTPUT ARRAY FOR SORTED X'S.
83       C SO WE PROCESS THESE X BINS AND THEN GO BACK FOR ANOTHER PASS THROUGH
84       C THE DATA
85       C
86          100  BACKSPACE(PRMFL)
87               STOP = STOP-K-1
88          101  PASS = PASS+1
89               CNT = 0
90       C
91               IF(BIG2SMAL ) WRITE(6,10) PASS, MAXX-STRT, MAXX+1-STOP
92               IF( .NOT. BIG2SMAL) WRITE(6,10) PASS, STRT+1, STOP
93         10    FORMAT(//,20X,'THIS IS PASS #',I3,' FOR XBINS FROM',I8,' TO',I8)
94               IF(PASS .GT. NUMPASS) STOP
95               REWIND (INTAP)
96               CALL READB(INTAP,EOF)
97               DO 110 I=1,LIMIT
98          110  IN(1,I) = 0
99       C
00        150   IF (BUFPTR .GE. 156) CALL READB( INTAP, EOF)
01               IF( EOF .EQ. 1) GOTO 200
02               X = BUFF(BUFPTR,1)
03               DATA = BUFF(BUFPTR,2)
04               BUFPTR = BUFPTR+1
```

```
78   02.353            Y S O R T                                    LABE

05                IF( X .EQ. 0 .AND. DATA .EQ. 0) GOTO 150
06         C
07                UNIQUE = IRS(X,18)
08                X = AND(X,LOWMSK)
09         C
10         C  ***************************************************************
11         C
12         C THIS CHANGE CAUSES US TO SORT FROM LARGEST TO SMALLEST
13         C
14                IF(BIG2SMAL )  X = MAXX+1 - X
15         C
16         C  ***************************************************************
17         C
18                IF(X .GT. 0) GOTO 180
19         C
20         C DATA ABOUT TO BEGIN A NEW BOUNDARY, SO WE REINITIALIZE TEST ARRAY
21         C THE FIRST POINT OF EACH BOUNDARY IN EACH XBIN GETS A SPECIAL FLAG
22         C
23                DO 155 M9 = 1,NUMBIN
24           155  TEST(M9) = 0
25                GOTO 150
26           180  ADJX = X-STRT
27                IF(ADJX .LE. 0 .OR. X .GT. STOP) GOTO 150
28                CNT = CNT+1
29                INDEX = XBIN(ADJX)
30                IF(INDEX .LE. LIMIT) GOTO 234
31                WRITE(6,235) X,DATA,INDEX,ADJX,XBIN(ADJX-1),XBIN(ADJX+1)
32           235  FORMAT(' INDEX OUT OF RANGE',6I8)
33                STOP
34           234  IF(IN(1,INDEX) .NE. 0) GOTO 800
35                IF(TEST(ADJX) .NE. 0) GOTO 188
36                TEST(ADJX) = 1
37         C
38         C  THIS DATA VALUE IS BEING FLAGGED BY SETTING THE FEATURE CODE (FC)
39         C  --THE LAST TWO BITS OF THE DATA VALUE -- TO ZERO
40         C  VARIABLE 'FC' IS THE VALUE FOR FEATURE CODE FOR AREAS
41         C
42                DATA = DATA - FC
43           188  IN(1,INDEX) = DATA
44                IN(2,INDEX) = UNIQUE
45                XBIN(ADJX) = INDEX + 1
46                GOTO 150
47         C
48         C  WRITE OUT FILLED ARRAY OF SORTED X'S TO TAPE
49         C
50            200  EOF = 0
51                WRITE(OTAP) XBIN(1)-1
52                IF(XBIN(1) .GT. 1)
53              &   WRITE(OTAP) ((IN(K,J),J=1,XBIN(1)-1),K=1,2)
54         C
55                LENG = STOP-STRT
56                DO 250 I=2,LENG
```

```
78   02.353              Y S O R T                                    LABI

57                 DIF = XBIN(I) - XBIN(I-1)
58                 WRITE(OTAP) DIF
59                 IF(DIF .GT. 0)
60                &  WRITE(OTAP) ((IN(K,J),J=XBIN(I-1),XBIN(I)-1),K=1,2)
61           250   CONTINUE
62           C
63                 WRITE(6,13) PASS,CNT,XBIN(LENG)-1
64           13    FORMAT(' IN PASS #',I3,I10,' PTS WERE WRITTEN TO TAPE.  WE USED',
65                &    I8,' LOCATIONS IN THE SORTING ARAY')
66                 STRT = STOP
67                 STOP = STOP+2
68                 IF(DONE .EQ. 0) GOTO 5
69                 WRITE(6,901)
70           901   FORMAT(///,20X,'##### WE ARE DONE')
71                 STOP
72           C
73           C  ERROR
74           C
75           800   WRITE(6,801) X,CNT,XBIN(ADJX-1),XBIN(ADJX),XBIN(ADJX+1)
76           801   FORMAT(' SORTING ERROR FOR X=',I6,5X,I6,' POINTS ALREADY '
77                &  'SORTED.  XBIN VALUES FOR (ADJX-1),ADJX,(ADJX+1)',3I8)
78                 STOP
79           C
80           900   DONE = 1
81                 STOP = STOP-1
82                 WRITE(6,921) INDEX-1,XBIN(INDEX-1),STRT,STOP
83           921   FORMAT(//,'NO MORE XBINS TO PROCESS.  LAST BIN USED',I6,
84                &  ' LAST LOCATION USED',I7,' START AND STOP',2I8)
85                 GOTO 101
86                 END
ORY EXPANDED. USE $LIMITS OR CORE= OPTION FOR NEXT RUN
```

```
78   02.356                                                        LABEL

 1                SUBROUTINE READB( INTAP, EOF)
 2                IMPLICIT INTEGER (A-Z)
 3                COMMON BUFF(155,2),BUFPTR
 4        C
 5                READ (INTAP,END=900) BUFF
 6                BUFPTR = 1
 7                RETURN
 8        C
 9         900    EOF = 1
10                RETURN
11                END
```

```
      C                    B U I L D
      C
      C  PROGRAM TO FIND 'BOTTOMS' AND 'TOPS' FOR FIELDS WITHIN EACH
      C  X-BIN, FOR CONSTRUCTION OF D.BASE
      C  OUTPUT IS FOR EACH X-BIN A SERIES OF RECORDS:
      C      (YSTART,YSTOP,CATEGORY)
      C
            IMPLICIT INTEGER (A-Z)
            DIMENSION OT(1500)
            COMMON IN(2,500),Y(500),TYPE(500)
      C
            DATA INTAP,OTAP,BOTTOM,TOP /1,2,0,1/
            DATA LIMIT /500/
      C
            READ(INTAP) FIRSTX,LASTX
            NUMX = LASTX - FIRSTX + 1
            WRITE(OTAP) NUMX
            DO 500 LOOP = 1,NUMX
            CNT = 1
            READ(INTAP) NUMY
            IF(NUMY .GT. 0) READ(INTAP) ((IN(K,J),J=1,NUMY),K=1,2)
            WRITE(6,10) LOOP+FIRSTX-1,NUMY
         10 FORMAT(/,30X,'< < < <  LINE ',I5,' HAS ',I4,' POINTS')
      C
            IF(NUMY .LE. LIMIT) GOTO 95
            WRITE(6,30) LIMIT
         30 FORMAT(///,' * * * * TOO MANY POINTS IN THIS COLUMN. PROGRAM '
           &   'DIMENSIONS MUST BE EXTENDED BEYOND ',I4)
            STOP
         95 IF(NUMY .EQ. 0) GOTO 300
            STRT = 1
      C
      C
        100 CALL SORT(NUMY,STRT,STOP,CAT,UNIQUE)
            OSTRT = STRT
            OSTOP = STOP
            IF (STRT .GT. NUMY)GOTO 300
            ERR = 0
         12 FORMAT('CAT=',I8,' HAS',I5,' POINTS, WITH RANGE=',2I8,
           &   '   UNIQUE =',I8)
      C
        120 IF(TYPE(STRT) .EQ. BOTTOM) GOTO 200
            ERR = ERR+1
            IF(ERR .LE. 1)
           & WRITE(6,12) CAT,OSTOP-OSTRT+1,Y(OSTRT),Y(OSTOP),UNIQUE
            IF(ERR .LE. 1) WRITE(6,19) (Y(I9),TYPE(I9),I9=OSTRT,OSTOP)
         19 FORMAT(20X,'THE YVALUES AND CORRESPONDING TYPES ARE',/,12(I8,I2))
      C
      C   CHECK FOR ADJACENT TOP AND BOTTOM OR IDENTICAL TOP AND BOTTOM
      C
            NXT = STRT
            IF (TYPE(NXT+1) .NE. BOTTOM) GOTO 121
              IF (Y(NXT) .EQ. Y(NXT+1)) GOTO 230
```

```
      DO 1210 DIFF=1,3
1210    IF (Y(NXT)+DIFF .EQ. Y(NXT+1)) GOTO 240
C
C
C     THESE POINTS HAVE THEIR TOP AND BOTTOM INTERCHANGED
C      SO ALLOW THEM TO BE OUTPUT
C
        IF (STRT+1 .GT. STOP) GOTO 121
        WRITE(6,4000)Y(STRT),Y(STRT+1)
4000    FORMAT(/,5X,'<*><*>WARNING, THESE POINTS HAD THEIR TOPS AND',
     &    ' BOTTOMS INTERCHANGED ',2I6)
        NXT = STRT + 1
        GOTO 250
C
121     WRITE(6,15) Y(STRT),Y(STRT+1),TYPE(STRT+1)
15      FORMAT(/,5X,'**** BEGIN FIELD W/O BOTTOM AT',I6,' IT WAS IGNORED'
     &  ' NEXT POINT AND TYPE IS',I8,I2)
        STRT = STRT+1
        GOTO 120
C
200     NXT = STRT+1
201     IF(NXT .LE. STOP) GOTO 220
        ERR = ERR+1
        IF(ERR .LE. 1)
     &  WRITE(6,12) CAT,OSTOP-OSTRT+1,Y(OSTRT),Y(OSTOP),UNIQUE
        IF(ERR .LE. 1) WRITE(6,19) (Y(I9),TYPE(I9),I9=OSTRT,OSTOP)
        WRITE(6,20) Y(STOP)
20      FORMAT(/,5X,'**** ENDED W/O TOP. THIS POINT IGNORED',2I8)
        NXT = STOP
        GOTO 290
C
220     IF(TYPE(NXT) .EQ. TOP) GOTO 250
        ERR = ERR+1
        IF(ERR .LE. 1)
     &    WRITE(6,12) CAT,OSTOP-OSTRT+1,Y(OSTRT),Y(OSTOP),UNIQUE
        IF(ERR .LE. 1) WRITE(6,19) (Y(I9),TYPE(I9),I9=OSTRT,OSTOP)
C
C    CHECK FOR ADJACENT TOP AND BOTTOM OR IDENTICAL TOP AND BOTTOM
C
        IF (TYPE(NXT+1) .NE. TOP) GOTO 221
          IF (Y(NXT) .EQ. Y(NXT+1)) GOTO 230
        DO 2200 DIFF=1,3
2200      IF (Y(NXT)+DIFF .EQ. Y(NXT+1)) GOTO 240
C
221     WRITE(6,22) Y(STRT),Y(NXT)
22      FORMAT(/,5X,'** 2 BOTTOMS IN A ROW',I8,' AND',I8,' 2ND IGNORED')
        NXT = NXT+1
        GOTO 201
C
C    TOP AND BOTTOM WERE ASSIGNED TO THE SAME Y VALJE FOR THIS X-BIN SO
C      IGNORE BOTH OF THEM
C
230     WRITE(6,2300) Y(NXT)
```

```
-12-79   14.449                    B U I L D

   107                GOTO 241
   108         C
   109         C
   110         C    ADJACENT POINTS WERE CALLED TOP AND BOTTOM, IGNORE THEM BOTH
   111         C
   112         240     WRITE(6,2400) (Y(I),TYPE(I), I = NXT, NXT+1),DIFF
   113         C
   114         C    IGNORE THE TWO POINTS THAT WERE TOP AND BOTTOM AND EQUAL OR ADJACENT
   115         C
   116         241     NXT = NXT + 1
   117                 IF (TYPE(NXT-1) .EQ. TOP) GOTO 290
   118                   NXT = NXT + 1
   119                   GOTO 201
   120         C
   121         2300    FORMAT (' *** SAME POINT CALLED TOP & BOTTOM AND IGNORED. Y=',I8)
   122         2400    FORMAT (' *** CLOSE POINTS CALLED TOP & BOTTOM AND IGNORED. Y',
   123              &          ' VALUES AND TYPES WERE',2(I8,I2,5X),'DIFFERENCE OF ',I2)
   124         C
   125         C
   126         C  EVERYTHING LOOKS OK. SO WRITE TO TAPE
   127         C
   128           250   OT(CNT) = Y(STRT)
   129                 OT(CNT+1)=Y(NXT)
   130                 OT(CNT+2)=CAT + ILS(UNIQUE,18)
   131                 CNT = CNT+3
   132           290   STRT = NXT+1
   133                 IF(STRT .LE. STOP) GOTO 120
   134                 IF(STRT .LE. NUMY) GOTO 100
   135           300   CNT = CNT-1
   136                 WRITE(OTAP) CNT
   137                 IF(CNT .GT. 0) WRITE(OTAP) (OT(J),J=1,CNT)
   138           500   CONTINUE
   139                 STOP
   140                 END


  03-31-77   **SR4J**

(SEC)        .77          LINES/MINUTE 10801

 NO    DIAGNOSTICS IN ABOVE COMPILATION
 WERE USED FOR THIS COMPILATION
```

```
      SUBROUTINE SORT(NUMY,STRT,STOP,CAT,UNIQUE)
      IMPLICIT INTEGER (A-Z)
      DIMENSION DELE(15),FLIP(15)
      COMMON IN(2,500),Y(500),TYPE(500)
      DATA MSKCAT /0777777/
      DATA DELE/2707,2215,4227,12*0/
C SPECIAL FIXES FOR FILE #1 ON 60716 (WATERTOWN #1)
C     DATA FLIP /126,499,532,589,805,833,1170,1190,1118,1192,1207,
C    &   1364,3*0/
C
      DATA FLIP / 15*0/
      DATA SIZE/3/
C
C
 1    STOP = STRT
      CAT = AND(IN(1,STRT),MSKCAT)
      UNIQUE = IN(2,STRT)
      DO 90 I=1,SIZE
        IF(UNIQUE .EQ. DELE(I)) GOTO 800
 90     CONTINUE
      BOTTOM = 0
      TOP = 1
      DO 99 II=1,SIZE
      IF(UNIQUE .NE. FLIP(II)) GOTO 99
      BOTTOM = 1
      TOP = 0
 99   CONTINUE
C
 5    STOP = STOP+1
      IF(STOP .GT. NUMY) GOTO 100
      NXT = IN(2,STOP)
      IF(UNIQUE .EQ. NXT) GOTO 5
C
 100  STOP = STOP-1
      DO 200 I=STRT,STOP-1
      MIN = ILS(IN(1,I),1)
      NXT = I + 1
      DO 190 J=NXT,STOP
      NUM = ILS(IN(1,J),1)
      IF(NUM-MIN) 150,140,190
 140  IF(IN(1,I) .GT. 0) GOTO 190
 150  T = IN(1,J)
      IN(1,J) = IN(1,I)
      IN(1,I) = T
      MIN = NUM
 190  CONTINUE
 200  CONTINUE
C
      DO 300 I=STRT,STOP
      NUM = IN(1,I)
      Y(I) = FLD(1,17,NUM)
      IF(NUM .LT. 0) TYPE(I) = TOP
      IF(NUM .GT. 0) TYPE(I) = BOTTOM
```

```
-12-79   14.450

    54          300  CONTINUE
    55        C
    56               IF(STRT .EQ. STOP) WRITE(6,700) STRT,UNIQUE,NXT
    57          700  FORMAT(/,'    * * * * SINGLE POINT FOR THIS FIELD',3I8)
    58               RETURN
    59          800  STRT = STRT+1
    60               IF (STRT .GT. NUMY) RETURN
    61               IF (IN(2,STRT) .NE. UNIQUE) GOTO 1
    62               GOTO 800
    63               END


  03-31-77   **SR4J**

(SEC)        .40          LINES/MINUTE  9366

 NO    DIAGNOSTICS IN ABOVE COMPILATION
 WERE USED FOR THIS COMPILATION
```

```
7-78   16.083                                                          L
```

```
   1              IMPLICIT INTEGER (A-Y)
   2              LOGICAL EMPTY
   3              DIMENSION LOG(10,100)
   4              COMMON START(1000),STOP(1000),CATNUM(1000),OT(5000)
   5              COMMON /STK/ STACK(4,10),TOP1
   6        C
   7              DATA INTAP,OTAP,YMIN,YMAX /1,2, 4500,8700 /
   8              DATA LIMIT /500/
   9              DATA GAP,OVRLAP /1,2/
  10              DATA LOWMSK/ 0777777/
  11        C
  12              TOP = 0
  13              TOP1 = 0
  14              READ(INTAP) NUMX
  15              DO 9876 I=1,3600
  16         9876 READ(INTAP)
  17        C
  18        C
  19        C    MAJOR LOOP.
  20        C
  21        C    EACH PASS CONSTRUCTS A COLUMN (XBIN) OF THE CATEGORY DATA BASE
  22        C
  23              DO 500 II = 1801,NUMX
  24              LOOP = II
  25              YRANGE = YMAX+1-YMIN
  26              DO 110 I=1,YRANGE
  27          110 OT(I) = 0
  28        C
  29              READ(INTAP) CNT
  30              CNT = CNT/3
  31        C
  32              IF (CNT .GT. 0)WRITE(6,12) LOOP,CNT
  33           12 FORMAT(20X,'XBIN =',I6,'  HAS',I5,' FIELDS')
  34              IF(CNT .EQ. 0) GOTO 450
  35              IF(CNT .LE. LIMIT) GOTO 113
  36              WRITE(6,114)
  37         114  FORMAT('*************** LIMIT MUST BE INCREASED *******')
  38              STOP
  39          113 READ(INTAP) (START(K),STOP(K),CATNUM(K),K=1,CNT)
  40           34 FORMAT(5(2I6,1X,O12))
  41              IF(CNT .GT. 1) CALL SORT(CNT)
  42              FIELD = 0
  43        C
  44        C  INNER LOOP TO MERGE DATA FOR A SINGLE SCAN LINE
  45        C
  46           50 FIELD = FIELD+1
  47              STRTFLD = START(FIELD)
  48              STPFLD = STOP(FIELD)
  49              CAT = AND(CATNUM(FIELD),LOWMSK)
  50              UNIQUE = IRS(CATNUM(FIELD),18)
  51        C
  52              IF(FIELD .GE. CNT) GOTO 400
```

```
                NXT = FIELD+1
                UNIQNXT = IRS(CATNUM(NXT),16)
                NXTCAT=AND(CATNUM(NXT),LOWMSK)
      C
       55       IF(STPFLD .LE. STOP(NXT)) GOTO 200
      C
      C  THIS FIELD ENCLOSES THE NEXT FIELD SO FILL TO START OF NEXT FIELD WIT
      C THE CATEGORY OF PRESENT FIELD.  ALSO SAVE THIS FIELD ON THE STACK
      C FOR FILL ON EXIT OF THE ENCLOSED FIELD(S)
      C
                CALL FILL(STRTFLD,START(NXT),CAT,UNIQUE,YMIN)
                IF(STPFLD-STOP(NXT) .GT. 7)
               &  CALL PUSH(STRTFLD,STPFLD,CAT,UNIQUE)
                GOTO 50
      C
      C   CURRENT FIELD DOES NOT ENCLOSE NEXT FIELD, SO
      C THE FIELDS MAY (1) OVERLAP, OR (2) HAVE A GAP BETWEEN THEM.
      C
      C <*<*<*<*<*<*  NOTE                A DIFFERENCE OF 7 (SEVEN) CELLS OR LESS
      C                TWO BOUNDARIES IS CONSIDERED A DIGITIZING ERROR AND THE
      C                FIELDS ARE MADE TO MATCH EXACTLY
      C                                                  *>*>*>*>*>*>*>*>*>*>*>
      C
        200   DIF = START(NXT) - STPFLD
                IF(DIF .GT. 0) GOTO 300
      C
      C  THESE TWO FIELDS OVERLAP.  IF IT IS SIGNIFICANT AN ERROR IS RECORDED
      C
                IF(DIF .LT. -7)
               & CALL PROB(OVRLAP,LOOP,STPFLD,UNIQUE,START(NXT),UNIQNXT,.FALSE.,
               &   CAT,NXTCAT)
                CALL FILL(STRTFLD,START(NXT),CAT,UNIQUE,YMIN)
                GOTO 50
      C
      C   THERE IS A GAP BETWEEN THE FIELDS
      C
        300   IF(DIF .GT. 7) GOTO 310
      C
      C INSIGNIFICANT GAP, SO MAKE THE FIELDS MEET
      C
                CALL FILL(STRTFLD,START(NXT),CAT,UNIQUE,YMIN)
                GOTO 50
      C
      C   SIGNIFICANT GAP.  PERHAPS THERE IS A LARGER FIELD ENCLOSING THESE
      C FIELDS WHICH WILL FILL IN THIS AREA.
      C CHECK THE STACK FOR THE ENCLOSING FIELD PARAMETERS.
      C
        310   CALL FILL(STRTFLD,STPFLD,CAT,UNIQUE,YMIN)
        311   CALL POP(STR1,STP,CAT1,UNIQ,EMPTY)
                LEVL = LEVL+1
                IF(EMPTY) GOTO 390
                IF(STP .LT. STPFLD) GOTO 380
```

```
105               STRTFLD = STPFLD + 1
106               STPFLD = STP
107               GOTO 55
108         C
109          380  IF(TOP .EQ. 0) GOTO 381
110               TEST = ILS(UNIQ,18) + UNIQUE
111               DO 383 M1=1,TOP
112               IF( TEST .NE. LOG(1,M1)) GOTO 383
113               LOG(10,M1) = -LOG(10,M1)-1
114               GOTO 311
115          383  CONTINUE
116         C
117         381   TOP = TOP + 1
118               LOG(1,TOP ) = TEST
119               LOG(2,TOP) = STRT
120               LOG(3,TOP) = STP
121               LOG(4,TOP) = CAT1
122               LOG(5,TOP) = STRTFLD
123               LOG(6,TOP) = STPFLD
124               LOG(7,TOP) = CAT
125               LOG(8,TOP) = START(NXT)
126               LOG(9,TOP) = AND(CATNUM(NXT),LOWMSK)
127               LOG(10,TOP) = -1
128               GOTO 311
129         C
130            390  CALL PROB(GAP,LOOP,STPFLD,UNIQUE,START(NXT),UNIQNXT,.FALSE.,
131                &    CAT,NXTCAT)
132                 GOTO 50
133         C
134         C   WE HAVE REACHED THE LAST FIELD IN THIS SCAN LINE.
135         C   CHECK THE STACK TO SEE IF THERE ARE ANY ENCLOSING FIELDS REMAINING
136         C
137            400  CALL FILL(STRTFLD,STPFLD,CAT,UNIQUE,YMIN)
138                 CALL POP(STRTFLD,STPFLD,CAT,UNIQUE,EMPTY)
139                 IF ( .NOT. EMPTY) GOTO 400
140            450  WRITE(OTAP) (OT(M9),M9=1,YRANGE)
141                 CALL PROB(DUM,LOOP,STRT,UNIQUE,STP,UNIQNXT,.TRUE.,
142                &    CAT,NXTCAT)
143         C
144         C
145               M1 = 1
146          707  IF( M1 .GT. TOP) GOTO 500
147               IF(LOG(10,M1) .GT. 0) GOTO 705
148               LOG(10,M1) = -LOG(10,M1)
149               M1 = M1 + 1
150               GOTO 707
151          705  WRITE(6,701) (LOG(M2,M1),M2=2,10)
152          701  FORMAT(/'  ENCLOSING FIELD STRT,STOP,CAT =',3I6,'   ENCLOSED ',
153                &  'FIELD STRT,STOP,CAT =',3I6/T50,'NEXT FIELD STRT,CAT =',2I6,
154                &  '   LENGTH OF ERROR =',I6)
155               DO 720 M2 = 1,10
156          720  LOG(M2,M1) = LOG(M2,TOP)
```

```
17-78   16.0b3

  157                 TOP = TOP-1
  158                 GOTO 707
  159       C
  160           500   CONTINUE
  161       C
  162                 WRITE(6,501)
  163           501   FORMAT(///,30X,'WE ARE DONE')
  164                 STOP
  165                 END
MEMORY EXPANDED. USE $LIMITS OR CORE= OPTION FOR NEXT RUN
```

```
      SUBROUTINE PROB(TYPE,XBIN,STRT,UNIQUE,STP,UNIQNXT,EOL,
     &    CAT,NXTCAT)
      IMPLICIT INTEGER (A-Z)
      LOGICAL EOL
      CHARACTER CHAR(2)
      DIMENSION LOG(2,6,100),TOP(2)
      DATA MAX,GAP,OVRLAP /100,1,2/
      DATA MASK /O'7777777'/
      CHAR(1) = '  GAP '
      CHAR(2) = 'OVRLAP'
C
      IF(EOL) GOTO 500
      WIDTH = IABS(STP-STRT)
      IF(TOP(TYPE) .GE. MAX) GOTO 800
      IF(TOP(TYPE) .EQ. 0) GOTO 110
      TEST = ILS(UNIQUE,18) + UNIQNXT
      DO 100 I=1,TOP(TYPE)
      IF( TEST .EQ. LOG(TYPE,1,I) ) GOTO 200
  100 CONTINUE
C
C  THIS IS A NEW PROBLEM SO OUTPUT ERROR MESSAGE AND RECORD IT ON LOG FI
C
  110 CONTINUE
C     WRITE(6,10) CHAR(TYPE),STRT,IABS(STP-STRT),UNIQUE,UNIQNXT
   10 FORMAT(/,1X,A6,'  YSTART AND WIDTH=',
     &  2I6,2X,' THE FIELDS ARE',2I8)
      TOP(TYPE) = TOP(TYPE)+1
      LOG(TYPE,1,TOP(TYPE)) = TEST
      LOG(TYPE,2,TOP(TYPE)) = UNIQNXT
      LOG(TYPE,3,TOP(TYPE)) = ILS(XBIN,18) + STRT
      LOG(TYPE,4,TOP(TYPE)) = WIDTH
      LOG(TYPE,5,TOP(TYPE)) = CAT
      LOG(TYPE,6,TOP(TYPE)) = NXTCAT
      RETURN
C
C  THIS PROBLEM EXISTED IN THE PREVIOUS LINE SO DO NOT REPEAT ERROR MES
C
  200 IF(WIDTH + LOG(TYPE,4,I) .GT. 0) GOTO 201
      LOG(TYPE,4,I) = -LOG(TYPE,4,I)
      RETURN
  201 LOG(TYPE,4,I) = WIDTH
      RETURN
C
C  END OF LINE.
C  OUTPUT MESSAGE FOR THOSE PROBLEMS FOUND IN PREVIOUS LINE BUT NOT
C  REPEATED IN THIS LINE.
C
  500 TYPE = GAP
  501 LOOP = 1
  510 IF(LOOP .GT. TOP(TYPE)) GOTO 800
      IF(LOG(TYPE,4,LOOP) .GT. 0) GOTO 550
      OLDX = IRS(LOG(TYPE,3,LOOP),18)
```

```
   53                LENGTH = XDIN-OLDX
   54                OLDY = AND(LOG(TYPE,3,LOOP),MASK)
   55                LOG(TYPE,1,LOOP) = IRS(LOG(TYPE,1,LOOP),18)
   56                IF(LENGTH .GT. 2)
   57               & WRITE(6,505) CHAR(TYPE), (LOG(TYPE,K,LOOP),K=1,2),
   58               &   (LOG(TYPE,K,LOOP),K=5,6),
   59               &   LENGTH,-LOG(TYPE,4,LOOP), OLDY
   60           505 FORMAT (/2X,A6,' BETWEEN FIELDS ',2I8,' CAT,NXTCAT =',2I6,
   61               &  '   LENGTH OF ERROR=',I6,' MAX WIDTH =',I5,' YSTART =',I6)
   62                DO 520 I=1,6
   63           520 LOG(TYPE,I,LOOP) = LOG(TYPE,I,TOP(TYPE))
   64                TOP(TYPE) = TOP(TYPE)-1
   65                IF(TOP(TYPE) .EQ. 0) GOTO 600
   66                GOTO 510
   67           C
   68           550 LOG(TYPE,4,LOOP) = -LOG(TYPE,4,LOOP)
   69                LOOP = LOOP+1
   70                GOTO 510
   71           C
   72           600 IF(TYPE .EQ. OVRLAP) RETURN
   73                TYPE = OVRLAP
   74                GOTO 501
   75           800 WRITE(6,801)
   76           801 FORMAT(///'******** STACK SIZE EXCEEDED IN SUBROUTINE')
   77                STOP
   78                END
```

```
      SUBROUTINE FILL(BEG,END,CAT,UNIQUE,BIASY)
      IMPLICIT INTEGER (A-Z)
      COMMON STRT(1000),STOP(1000),CATNUM(1000),OT(5000)
C
      VALUE = ILS(UNIQUE,18) + CAT
      IF(END .LT. BEG) GOTO 800
      FIRST = BEG +1 - BIASY
      LAST =  END +1 - BIASY
      IF( LAST .GT. 5000) WRITE(6,891) BEG,END,CAT,UNIQUE,BIASY
  891 FORMAT(' EXCEEDED YRANGE.  BEG,END,CAT,UNIQUE,BIASY=',6I8)
      DO 100 I=FIRST,LAST
  100 OT(I) = VALUE
      RETURN
C
C
  800 WRITE(6,801) BEG,END,CAT,UNIQUE
  801 FORMAT(//,'**** ERROR IN FILL - BEG,END,CAT,UNIQUE=',4I8)
      RETURN
      END
```

-17-78 16.086

```
      1                SUBROUTINE SORT(CNT)
      2                IMPLICIT INTEGER (A-Z)
      3                COMMON STRT(1000),STOP(1000),CATNUM(1000),OT(5000)
      4          C
      5                DO 100 I=1,CNT-1
      6                MIN = STRT(I)
      7                BEG = I+1
      8                DO 90 J=BEG,CNT
      9                IF(MIN .LE. STRT(J)) GOTO 90
     10                MIN = STRT(J)
     11                STRT(J)=STRT(I)
     12                STRT(I) = MIN
     13                T=STOP(J)
     14                STOP(J)=STOP(I)
     15                STOP(I)=T
     16                T = CATNUM(J)
     17                CATNUM(J)=CATNUM(I)
     18                CATNUM(I) = T
     19           90   CONTINUE
     20          100   CONTINUE
     21                RETURN
     22                END
```

```
    1              SUBROUTINE PUSH(STRT,STOP,CAT,UNIQUE)
    2              IMPLICIT INTEGER (A-Z)
    3              COMMON /STK/STACK(4,10),TOP
    4        C
    5              DATA LIMIT/ 10/
    6              TOP = TOP+1
    7              IF(TOP .GT. LIMIT) GOTO 800
    8              STACK(1,TOP)=STRT
    9              STACK(2,TOP)=STOP
   10              STACK(3,TOP)=CAT
   11              STACK(4,TOP)=UNIQUE
   12              RETURN
   13        C
   14          800  WRITE(6,801)
   15          801  FORMAT(//,'**** STACK OVERFLOW ******')
   16               STOP
   17               END
```

3-17-76 16.067

```
      1              SUBROUTINE POP(STRT,STOP,CAT,UNIQUE,EMPTY)
      2              IMPLICIT INTEGER (A-Z)
      3              LOGICAL EMPTY
      4              COMMON /STK/STACK(4,10),TOP
      5        C
      6              IF(TOP .GT. 0) GOTO 100
      7              EMPTY = .TRUE.
      8              RETURN
      9        C
     10          100  EMPTY = .FALSE.
     11               STRT = STACK(1,TOP)
     12               STOP = STACK(2,TOP)
     13               CAT = STACK(3,TOP)
     14               UNIQUE = STACK(4,TOP)
     15               TOP = TOP-1
     16               RETURN
     17               END
```

AD-A076 119 KANSAS UNIV/CENTER FOR RESEARCH INC LAWRENCE REMOTE --ETC F/G 17/9
RADAR IMAGE SIMULATION OF SEASONALLY DEPENDENT REFERENCE SCENES--ETC(U
APR 79 J C HOLTZMAN , J E BARE , V H KAUPP DAAK70-78-C-0062
UNCLASSIFIED RSL-TR-370-2 ETL-0188 NL

3 OF 3
AD
A076119



END
DATE
FILMED
12-79
DDC

APPENDIX C

SPECIALIZATION OF THE PSM PPI IMPLEMENTATION TO THE RADAR SYSTEM

C.0 SPECIALIZATION OF THE PSM PPI IMPLEMENTATION TO THE RADAR SYSTEM

The general PPI radar simulation model described in Section 1.6 was specialized for a specific "real-world" application. The application selected used a PPI radar with a Correlatron* in a terminal guidance configuration for a ballistic missile. This was selected as a quantitative test of the PSM because simulation results were to be tested versus actual radar data collected over the target site by the same radar as well as being simulated. The test involved specializing the PSM for the guidance radar parameters and scan format, building a data base of a specific target site, producing simulations, and testing these simulations by using the Correlatron to measure the two-dimensional cross-correlation between them and actual radar data collected over the same site. A FORTRAN listing of this specialization is presented and dicussed in Section C.4

C.1 Correlatron

The Correlatron is an electronic device which externally resembles a television camera tube, but its internal construction and function are quite different[6]. The function of the Correlatron is to accept two voltage inputs (V_S and V_R) and determine the cross-correlation ($R_{V_R V_S}$) between them. This is shown conceptually in Figure C1. In this figure, the Correlatron is shown as an electronic "black box" having two inputs and one outout. One of the inputs is shown to be an actual radar image and the other a simulated radar image produced by the PSM.

---

*Correlatron is the name of a two-dimensional cross-correlation measuring device manufactured by Goodyear Aerospace.

[6]Klass, P.J., "Guidance Device Set for Pershing Tests," Aviation Week and Space Technology, 12 May 1975.

The video output voltage ($V_R(x,y,t)$) of the radar receiver is applied to intensity modulate a LED (Light-Emitting Diode) corresponding to the signal strength of the target echoes recieved. Quoting from Klass[6], "This beam of light impinges on the photo-cathode to generate electrons, which in turn are caused to scan by the Correlatron deflection system so as to 'paint' the equivalent of the "real-world" radar display on the storage screen." Klass[6] further states that the electrons emitted by the photo-cathode " . . . are then attracted to a dielectrically coated fine wire mesh that is at a positive potential so that the image is stored on the mesh in the form of many different electrical charges. Typically, the mesh consists of 500 to 1,000 wires per inch, but up to 2,000 per inch have been used to achieve extremely high resolution."

Once the real image is placed on the storage screen, then the simulated radar image produced via the PSM is projected onto the photo-cathode and the resulting pattern of electron mission is deflected to correlate it with the real-world radar image on the storage mesh. In Figure C1, the simulated radar image input to the Correlatron is shown as a video voltage ($V_S(s,y)$). This is conceptually accurate but not precise. The simulated radar image is actually provided as a photograph to the Correlatron by optically projecting a transparency onto a photo-cathode. The electron current produced by the photo-cathode then produced the voltage, V_S.

In this way the Correlatron produces the cross-correlation between actual and simulated radar images. The output of the correlation, $R_{V_R V_S}(x,y)$, is illustrated conceptually in Figure C1. Guidance information is derived

---

[6]Klass, P.J., "Guidance Device Set for Pershing Tests," Aviation Week and Space Technology, 12 May 1975.

from the X- and Y-offset of the match point relative to the absolute coordinate system in which it is measured. The simulated radar image is said to be "good" if the cross-correlation peak, the match point, is greater than a threshold value.

C.2 Specialization Considerations

The first step in specialization of the PSM to model the terminal guidance system was to attempt to describe the operating parameters of the PPI radar, itself (i.e., specification of the simulation parameters). However, limited information about the operating characteristics of both the radar and Correlatron was available. Therefore, in the absence of system design data, the guidance simulation software was developed assuming an ideal system. For instance, the PPI radar (for simulation purposes) was given constant azimuthal gain between its 3 dB points with no sidelobes (an aspiration for any antenna designer!). The elevation pattern was chosen to be $(\csc^2\beta)(\cos\beta)$, where β is the depression angle. Past the rf portion, the receiver of the ideal system was made to map linearly the received power into video intensity. A realistic film transfer characteristic was employed (logarithmic) with a linear dynamic range of 20 dB. Outside this range, either in the "toe" or "shoulder" of the exposure curve, lack of sufficient exposure or saturation, respectively, would result.

It was secondly considered whether there should be additional modifications made to the guidance simulation model to account for the presence of the Correlatron. The Correlatron was assumed to have identical paths for both the simulated and actual video voltages. The process of converting a simulated radar scene stored on photographic film to a video signal was

assumed to be linear. Identical tests run at different times were assumed to result in the same degree of cross-correlation. All of these criteria were assumed for the Correlatron.

C.3 Geometric Considerations

Complicating the situation for specializing the PSM for simulating the guidance system was the fact that the direction of approach to the target was not specified. To optimize the chances of high correlation and to allow the simulated radar scenes to be useful for any radar position and angle of approach, it was necessary to make them as nearly onmi-directional as possible. This was shown to dictate a nadir-looking antenna because of the angular dependence of both radar shadow and the backscattered fields. The only information about the system available before either constructing the data bases or the simulated radar scenes was: (1) the simulated scene altitude, and (2) the corresponding diameter of each simulated image. Thus, each image was formed with the radar centered over the site and looking radially outward as though its trajectory was, at least momentarily, vertical to the Earth.

Figure C2 illustrates the image format of the guidance radar being employed in comparison with an ordinary PPI radar scan format. Data are recorded by the guidance radar for a full circular sweep of the scene instead of the usual sector associated with PPI radars. The ground imaged by the radar beam is within an angular ring bounded at the near range by 35^{o} (incidence angle) and at the far range by 65^{o} with the scene within 35^{o} blanked out, creating a "hole" in the image. The guidance simulation model does not produce imagery in exactly this format because of: (1) the likelihood of centering and angle-of-approach errors, and (2) the use of the Correlatron as the diagnostic device.

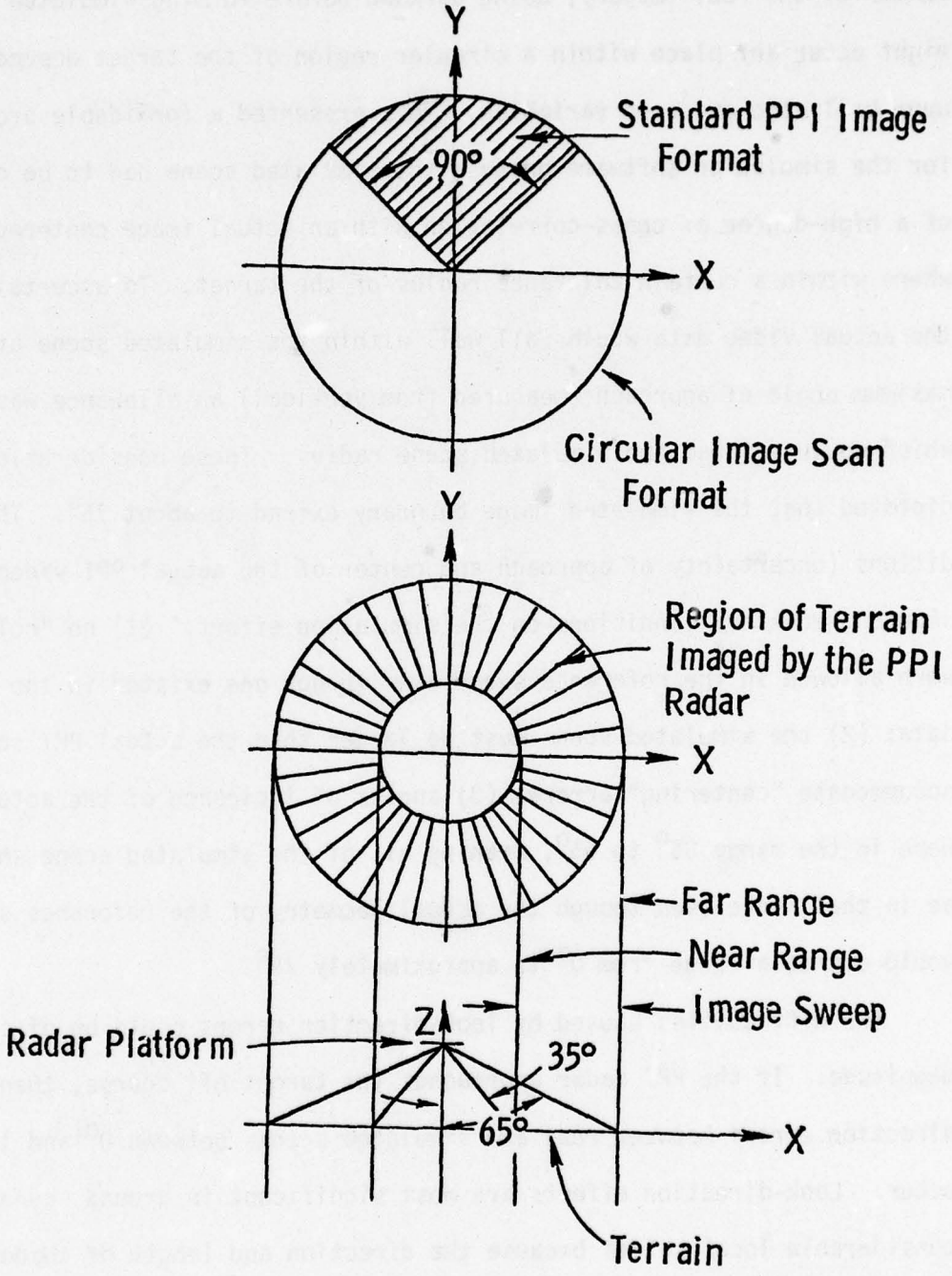


Figure C2. Special PPI Image Format.

The direction of approach of the missile and its PPI radar and the center of the real imagery, being unknown before forming simulated scenes, might occur any place within a circular region of the target depending upon ballistic guidance variables. This presented a formidable problem for the simulation software because the simulated scene had to be capable of a high-degree of cross-correlation with an actual image centered anywhere within a certain tolerance radius of the target. To ascertain that the actual video data would fall well within the simulated scene at the maximum angle of approach (measured from vertical) an allowance was made which would enlarge the simulated scene radius. These considerations dictated that the simulated image boundary extend to about 75^{o}. These conditions (uncertainty of approach and center of the actual PPI video signal) imposed necessary conditions on the simulation effort: (1) no "holes" were allowed in the reference scene even though one existed in the real data; (2) the simulated scene must be larger than the actual PPI scene to accommodate "centering" errors; (3) angles of incidence of the actual data were in the range 35^{o} to 65^{o}, meaning all of the simulated scene should also be in that range even though the actual geometry of the reference scene would decree a range from 0^{o} to approximately 75^{o}.

The difficulties caused by look-direction errors could be of severe magnitude. If the PPI radar approaches the target off course, then look-direction errors between real and simulated scenes between 0^{o} and 180^{o} occur. Look-direction effects are most significant in ground scenes having considerable local relief because the direction and length of shadows (and layover) in radar images are determined by the look direction. Figure C3 illustrates the problem for a look-direction error of 180^{o} between the real

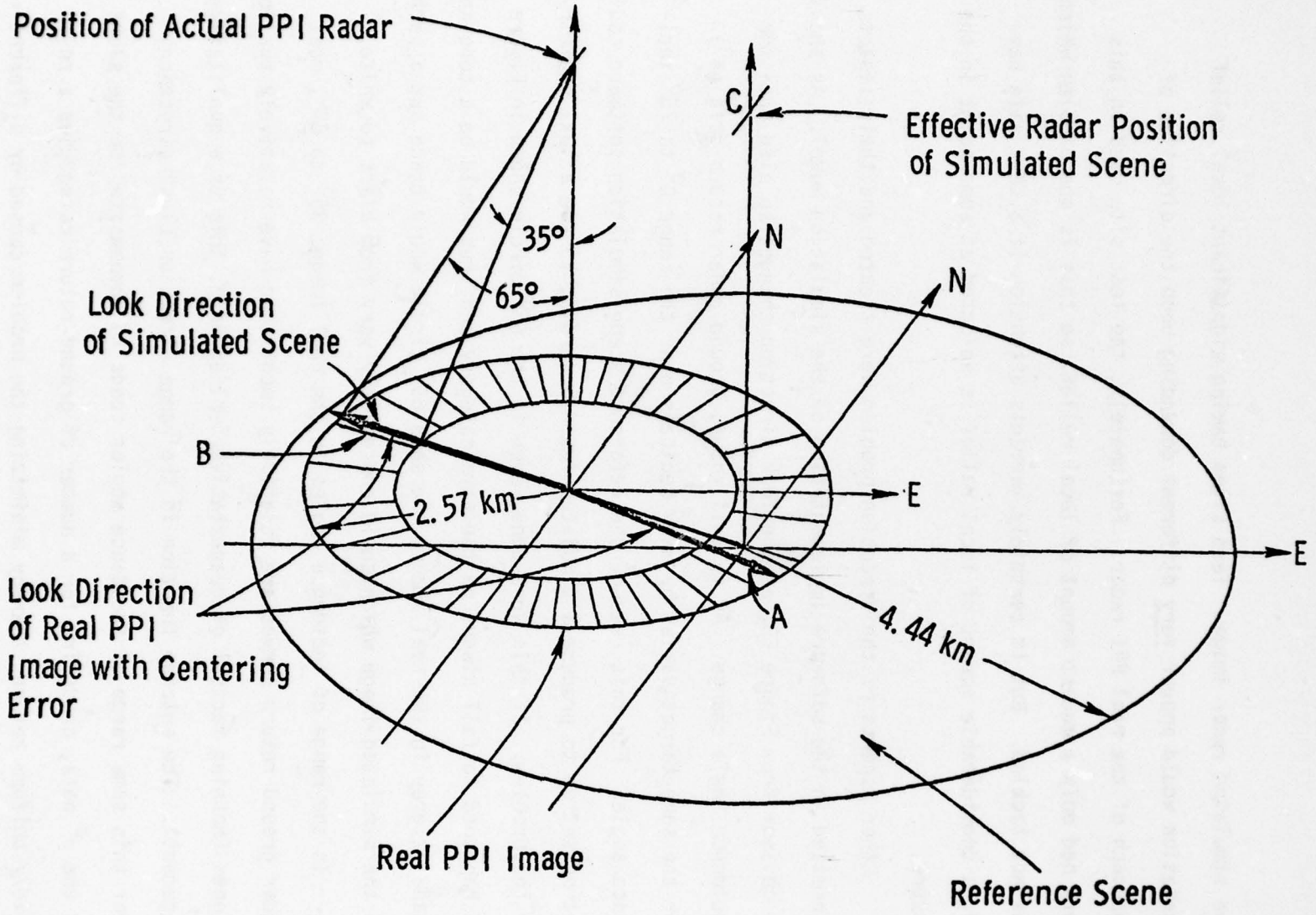


Figure C3. Comparison of Simulated Scene to PPI Radar Image Format.

and simulated radar image. Test sites having significant local relief variation would appear very different depending upon the direction of approach of the real PPI radar. Fortunately, the test site used in this work had only a modest amount of local relief, so this is one problem which was not tackled. But it certainly warrants attention if a test site having a considerable amount of local relief is selected at some point in the future.

Other aspects of the direction problem were treated and their impacts minimized in the software implementation of the simulation model. As should be obvious from Figure C3, as the look direction changes so also does the incidence angle change. As is well known, ground radar return data (σ^o) for the same target varies by many decibels over the range 0^o to 75^o incidence angle. For this reason, the reference scene simulation software could not be set up to produce a simulated image according to the actual geometry of the problem. If this were done, even if the 65^o circle shown in Figure C3 happened to fall always on the same category and thus would be a constant shade of grey in the real image, the same 65^o circle would trace out a path on the simulated image which could conceivably vary from black to white.

In the range of incidence angles in the real image, 35^o to 65^o, most radar ground return curves are relatively smooth and have relatively shallow slopes (nothing factual or quantitative implied here, this is a qualitative argument). The antenna function in the range direction (look-direction) over this same range of incidence angles tends to compensate for the slope of the σ^o data, producing for a number of ground return categories a relatively uniform return, thereby minimizing the problem caused by different look-directions. For these reasons it was decided that the minimum angle of incidence in the simulated scene would be 35^o and the maximum would be 75^o. The area in the scene lying between 35^o and 75^o angle of incidence

was simulated normally. The area lying within the 35^{o} circle was simulated as though the angle of incidence was a constant 35^{o}. This solution did not attempt to model the real situation exactly, but rather did attempt to minimize discrepancies between the simulated scene and the actual data produced in flight. This is not to say that local slope variations were not accounted for; they were indeed, incorporated. What is meant is that the incidence angle (θ) between the antenna "boresight" and the local vertical was always in the range 35^{o} to 75^{o}. Local slope variations then correctly altered the incidence angle to the local incidence angle (θ_{ℓ}). In fact, the limitations imposed on mimimum values of θ_{ℓ} come strictly from the local relief in the scene.

This solution to the angle of incidence problem created data handling problems for the computer program, and data base problems. For instance, 35^{o} angle of incidence specifies a resolution cell size for short-pulse and narrow-beamwidth radars. Yet, the geometry of the data base indicates that as data base cells get closer to the center (in polar coordinates), they get larger in the range direction and smaller in the azimuth direction. This problem was minimized by accurately modeling another feature of the real PPI; it recorded data in ground range mode. Ground range mode means that (for a flat Earth) equal size objects located in the near and far range will have equal sizes in the image format. This is normally accomplished by applying a nonlinear sweep to the electron beam of the viewing CRT (Cathode Ray Tube). But for simulation purposes, it simply meant building the simulation data base with equal size cells in the range direction. It should be noted at this time that in the presence of terrain having significant relief, ground range mode introduces large distortions, a fact to keep in mind for such future sites.

In summary, the general PSM radar image simulation model was specialized to the special requirements summarized in Table C1 which were imposed to simulate scenes for use on the Correlatron. A FORTRAN software listing of the implementation of the simulation model is presented in the following section.

C.4 FORTRAN Listing of PPI Computer Programs

The computer programs for the guidance PPI are provided in the following three sections:

C.4.1 Polar Conversion

A. Polar Create
B. Polar Array
C. Array Fix

C.4.2 Reference Scene

A. Power
B. Greytone

C.4.3 Rectangular Conversion

A. Rectangular Create
B. Rectangular Array

TABLE C1

GUIDANCE IMPLEMENTATION SPECIAL FEATURES

(1)	360^{o} PPI image scan format
(2)	Simulated area was larger than the real image to allow "centering" errors
(3)	No holes allowed, the reference scene was completely filled-in with radar image simulations
(4)	Minimum angle of incidence = 35^{o}
(5)	Maximum angle of incidence = 65^{o}
(6)	Local angle of incidence was properly treated
(7)	In the reference scene, the area between 0^{o} and 35^{o} was simulated at a constant 35^{o} angle of incidence
(8)	The area between 35^{o} and 75^{o} was simulated normally
(9)	Variations due to angle of incidence difference between real and simulated image were minimized
(10)	Simulated scenes were formed in the ground range mode
(11)	Layover and shadow were properly included
(12)	Local slope variations in the terrain were properly included

C.4.1 Polar Conversion Computer Program

This computer program was written in FORTRAN for implementation on a Honeywell 66/60. It consists of three subprograms:

	Page
(A) Polar Create	197
(B) Polar Array	203
(C) Array Fix	207

C.4.2 Reference Scene Computer Program

This computer program was written in FORTRAN for implementation on a Honeywell 66/60. It consists of two subprograms:

	Page
(A) Power	211
(B) Greytone	217

C.4.3 Rectangular Conversion Computer Program

This computer program was written in FORTRAN for implementation on a Honeywell 66/60. It consists of two subprograms:

	Page
(A) Rectangular Create	222
(B) Rectangular Array	224

```
13-78   09.801                POLAR CREATE                                  L

    1       C                    POLAR CREATE
    2       C
    3       C  THIS PROGRAM ACCEPTS DATA POINTS (RECTANGULAR FORMAT) FOR INPUT
    4       C  AND CREATES THE RESOLUTION CELL SIZE MATRIX IN POLAR COORDINATES
    5       C  TO BE USED AS DATA BASE FOR THE SIMULATION PROGRAM
    6       C
    7       C
    8              IMPLICIT INTEGER (A-Y)
    9              REAL ARCOS,FLOAT
   10              DIMENSION PRIOR(16,16),RECORD(3000),TABLE(1000),OT(4,300)
   11              DATA HALF,STRT/0,1/
   12              DATA CNT,NUMB/1,0/
   13              DATA OCT8,OCT2/OTOOOOOOOOO,01007
   14              DATA NUMCAT,HOLEFIX /16,50/
   15              DATA MSKCAT /0777/
   16       C
   17              WRITE(6,7)
   18            7 FORMAT(20X,'PRIORITY MATRIX FOR CATEGORIES')
   19              WRITE(02)NUMCAT,HOLEFIX,MSKCAT
   20       C
   21              DO 95 I=1,NUMCAT
   22              READ(05,5) (PRIOR(I,J),J=1,NUMCAT)
   23              WRITE(02)(PRIOR(I,J),J=1,NUMCAT)
   24              WRITE(6,5) (PRIOR(I,J),J=1,NUMCAT)
   25           95 CONTINUE
   26            5 FORMAT(1613)
   27       C
   28       C
   29       C  MIDX = DISTANCE (FEET) FROM LEFT EDGE OF DATA BASE TO TARGET CENTER
   30       C  MIDY = DISTANCE (FEET) FROM BOTTOM EDGE OF DATA BASE TO TARGET CENTER
   31       C  RADIUS = RADIUS (IN FEET) OF SIMULATION DESIRED <= MIN(MIDX,MIDY)
   32       C  CELSIZ = SIZE (IN FEET) REPRESENTED BY DATA POINTS - ASSUMED SQUARE
   33       C  NUMPT = NUMBER OF DATA POINTS PER RECORD ON INPUT TAPE
   34       C  NUMREC = NUMBER OF RECORDS ON INPUT TAPE
   35       C         EACH RECORD GOES FROM SOUTH TO NORTH.  RECORDS ON TAPE IN
   36       C         A WEST TO EAST ORDER
   37       C  WIDTH = FIXED SIZE FOR RANGE RESOLUTION
   38       C  ZBMWD = BEAMWIDTH (IN RADIANS)
   39       C
   40       C
   41       C  INPUT FOR PICKWICK 6 MILE RADIUS (7/9/77)
   42       C  31690 31690 29156 20 3109 3169 100 8000 .00875 .25
   43       C
   44       C  INPUT FOR PICKWICK 32000 FT ALTITUDE  (11/29/77)
   45       C  126,280 121,934  58312  82  2980  2932  328 16000 .00875  .25
   46       C
   47       C  ***********MODIFICATION*************(10/14/76)
   48       C
   49       C  ONE HAS THE OPTION TO MAKE RANGE RESOLUTION FIXED
   50       C  LET WIDTH = DESIRED RESOLUTION
   51       C  OR TO HAVE RANGE RESOLUTION VARY WITH RANGE
   52       C  LET WIDTH = 0 AND INPUT VALUE FOR PULSEWIDTH
```

```
      C  (FACTOR OF E-06 ASSUMED -- SO INPUT PARAMETER WILL LIKELY
      C  BE BETWEEN .1 AND 2.)
      C
      C  INPUT PARAMETERS FOR PICKWICK (10/14/76)
      C  31690  31690  26400  20  3169  3169 0  4000  .0175  .25
      C
      C
            READ(05,10) MIDX,MIDY,RADIUS,CELSIZ,NUMPT,NUMREC,WIDTH,ALT
            READ(05,11) ZBMWD,ZPULS
         10 FORMAT(8I9)
         11 FORMAT(2F10.6)
      C
            WRITE(6,21)
         21 FORMAT(//,30X,'INPUT PARAMETERS')
            WRITE(6,22) MIDX,MIDY,RADIUS,CELSIZ,NUMPT,NUMREC,
           &   WIDTH,ALT,ZBMWD,ZPULS
         22 FORMAT(' X,Y DIST TO TARGET CENTER',2I10,//,' SIMULATION'
           &  ' RADIUS',I10,//,' DATABASE RESOLUTION ',I8,//,' NUMBER OF'
           &  ' POINTS PER RECORD AND # OF RECORDS',2I8,//,' RANGE '
           &  ' RESOLUTION AND ALTITUDE ',2I10,//,' BEAMWIDTH AND '
           &  ' PULSE WIDTH ',2F12.7)
      C
      C
      C  QUARTER = FLAG TO DO ONLY ONE QUADRANT OF DATA BASE >0 YES
      C            <=0 DO FULL 360
      C
            READ(05,15) QUARTER
         15 FORMAT(I2)
      C
            ALT2 = ALT*ALT
      C
      C
      C  NUMR - MAXIMUM NUMBER OF CELLS IN RANGE DIRECTION IN
      C          RESOLUTION CELL MATRIX BEING CONSTRUCTED
      C
            IF(MIDX .GT. RADIUS .AND. MIDY .GT. RADIUS) GOTO 80
            WRITE(6,62) RADIUS,MIDX,MIDY
         62 FORMAT('  WARNING - DATA BASE TOO SMALL FOR DESIRED
           &   SCENE, LARGEST CIRCLE POSSIBLE WILL BE SIMULATED',//,
           &   ' INPUT PARAMETERS WERE RADIUS,MIDX,MIDY=',3I8)
      C
            IF(MIDX .LT. RADIUS) RADIUS = MIDX
            IF(MIDY .LT. RADIUS) RADIUS = MIDY
      C
         80 IF(WIDTH .EQ. 0) GOTO 85
            NUMR = RADIUS/WIDTH
            ZCW = FLOAT(CELSIZ)/FLOAT(WIDTH)
            GOTO 86
         85 ZCTAU = 983.57*ZPULS/2.
            MXSR = SQRT(RADIUS**2 + ALT**2)
            NUMR = FLOAT(MXSR - ALT)/ZCTAU
      C
```

```
      C  NUMANG - NUMBER OF ANGLE BINS TO BE CREATED IN RESOLUTION
      C         CELL MATRIX
      C
       86     N90 = 1.57/ZBMWD + 1
              N180 = N90*2 + 1
              NUMANG = N90*4
              IF(NUMANG .GT. 720) GOTO 815
      C
      C  X AND Y COORDINATES OF CENTER OF DATA BASE
      C
              CENTRX = MIDX/CELSIZ
              CENTRY = MIDY/CELSIZ
      C
      C  RAD - NUMBER OF DATA CELLS FROM CENTER TO EDGE OF
      C         SIMULATION AREA
      C
              RAD = RADIUS/CELSIZ
              WRITE(6,73) NUMR,NUMANG,CENTRX,CENTRY,RAD
       73     FORMAT(7//' INITIAL PARAMETERS ',5I8//)
              IF(RAD .GT. CENTRX .OR. RAD .GT. CENTRY) GOTO 810
      C
      C  TABLE = TABLE LOOK UP FOR ANGLE
      C          USE 1000 TIMES COSINE OF ANGLE AS INDEX
      C          RESULT IS PROPER ANGLE BIN FOR THE POINT
      C
              DO 100 I=1,1000
       100    TABLE(I)=ARCOS(FLOAT(I)/1000.)/ZBMWD + 1
      C
      C  WRITE PARAMETERS TO TAPE FOR DATA TO NEXT STEP
      C
              WRITE(02) NUMR,NUMANG,RADIUS,WIDTH,N90,ALT,ZCTAU
      C
      C
      C  BEGIN - FIRST LINE OF DATA BASE TO BE USED IN THIS
      C         SIMULATION
      C  IN CASE ONE WISHES TO SIMULATE ONLY A SEGMENT OF THE
      C  ENTIRE DATA BASE SOME LINES OF THE DATA BASE WILL BE
      C  UNUSED.  THIS LOOP POSITIONS THE USER AT THE FIRST LINE
      C  OF THE INPUT WHICH IS TO BE USED
      C
              BEGIN = CENTRX - RAD
              ENDREC = CENTRX + RAD +10
              IF(BEGIN .EQ. 0) GOTO 115
              DO 110 I=1,BEGIN
       110    READ(01)
       115    BEGIN = BEGIN+1
              WRITE(6,67) BEGIN
       67     FORMAT(' BEGIN =',I5//)
      C
      C
      C
              DO 200 I=BEGIN,ENDREC
```

```
      C
      C  READ IN NEW LINE OF INPUT
      C
            READ(01,END=800) (RECORD(N),N=1,NUMPT)
      C
      C  NX - DISTANCE (IN NUMBER OF CELLS) IN X DIRECTION FROM
      C  THE CENTER TO THE CURRENT LINE OF INPUT
      C
            NX = I-CENTRX
            IF( NX .EQ. 0) NX = 1
            IF(NX .GE. 0) HALF = 1
            IF(NX .GE. 0 .AND. OTF .EQ. 0) GOTO 600
  105       IF(QUARTER .GT. 0 .AND. NX .GE. 0) GOTO 500
            IF(NX .GT. CENTRX) GOTO 500
            NX2 = NX * NX
            ZX = ABS(NX)
      C
      C
      C  PLACE EACH POINT OF THE CURRENT INPUT LINE INTO THE
      C  APPROPRIATE CELL OF THE RESOLUTION CELL MATRIX BEING
      C  CREATED
      C  EACH PASS THROUGH THIS LOOP PROCESSES TWO POINTS OF THE
      C  LINE -- THE ONE J CELLS ABOVE THE CENTER LINE
      C  AND THE ONE J CELLS BELOW THE CENTER LINE
      C
            TCAT=0
            BCAT=0
            OLDR=0
            DO 180 JJ=1,RAD+1
            J = JJ-1
      C     IF( QUARTER .GT. 0 .AND. J .GT. IABS(NX)) GOTO 200
            NY2 = J*J
            ZR = SQRT(NX2 +NY2)
      C
      C  R - DISTANCE (IN NUMBER OF CELLS) FROM CENTER POINT TO
      C     THE CURRENT POINT
      C
            IF(WIDTH .EQ. 0) GOTO 113
            R = ZR*ZCW + 1.
            GOTO 313
  113       SR = SQRT(ALT2 + (ZR*CELSIZ)**2) - ALT
            IF(SR .LT. 0) GOTO 180
            R = SR/ZCTAU + 1.
  313       IF(R .GT. NUMR) GOTO 200
            COSANG = ZX/ZR * 1000.
      C
            IF(COSANG .LT. 0) WRITE(6,69) COSANG,I,NX,J,R
            IF(COSANG .GT. 1000) WRITE(6,69) COSANG,I,NX,J,R
   69       FORMAT(//' **ERROR - COS > 1',5I8/)
            IF(COSANG .GT. 10 .AND. COSANG .LT. 990) GOTO 117
            ANG= ARCOS(ZX/ZR)/ZBMWD + 1
            GOTO 118
```

```
      C
      C ANG - APPROPRIATE ANGLE BIN FOR THE CURRENT POINT
      C
  117   ANG = TABLE(COSANG)
      C
      C  HALF=1  IMPLIES RIGHT HALF OF THE SCENE IS BEING PROCESSED
      C  SO THE VARIABLE ANG IS MODIFIED APPROPRIATELY
      C
  118   CONTINUE
      C
  120   INDEX = CENTRY + J
        IF(OLDR .NE. R .OR. OLDANG .NE. ANG) GOTO 150
        CNT = CNT + 1
        CAT=AND(RECORD(INDEX),MSKCAT)
        IF(CAT .GT. NUMCAT) GOTO 820
        IF(TCAT .EQ. 0 .AND. CAT .GT. 0) TCAT=CAT
        IF(CAT .GT. 0) TCAT=PRIOR(TCAT,CAT)
        TELV = TELV + IRL(RECORD(INDEX),6)
      C
        INDEX = CENTRY - J
        CAT=AND(RECORD(INDEX),MSKCAT)
        IF(CAT .GT. NUMCAT) GOTO 820
        IF(BCAT .EQ. 0 .AND. CAT .GT. 0) BCAT=CAT
        IF(CAT .GT. 0) BCAT=PRIOR(BCAT,CAT)
        BELV = BELV + IRL(RECORD(INDEX),6)
        GOTO 180
      C
      C
  150   IF(OLDR .EQ. 0) GOTO 158
        TELV = TELV/CNT * OCT2 + TCAT
        BELV = BELV/CNT * OCT2 + BCAT
        NUMB=NUMB+1
        OT(1,NUMB)=OLDR
        OT(2,NUMB)=OLDANG
        OT(3,NUMB)=TELV
        OT(4,NUMB)=BELV
        OLDR = R
        OLDANG = ANG
        IF(NUMB .LT. 250) GOTO 157
        WRITE(02) NUMB
        WRITE(02) ((OT(L,M), L=1,4),M=1,NUMB)
        NUMB=0
  157   SUM = SUM+1
  158   TELV = IRL(RECORD(CENTRY +J),6)
        BELV = IRL(RECORD(CENTRY -J),6)
        TCAT = AND(RECORD(CENTRY+J),MSKCAT)
        BCAT = AND(RECORD(CENTRY-J),MSKCAT)
        TOTCNT = TOTCNT + CNT
        CNT = 1
      C
        OLDR=R
        OLDANG=ANG
```

```
261         180   CONTINUE
262         200   CONTINUE
263               GOTO 900
264       C
265       C
266       C  ERROR MESSAGES
267       C
268         800   WRITE(6,801) I
269         801   FORMAT(//'  UNEXPECTED END OF FILE AT RECORD ',I6)
270               GOTO 900
271         810   WRITE(6,811) RADIUS,MIDX,MIDY
272         811   FORMAT(//,'  ***ERROR   RADIUS EXCEEDS DATA BASE
273              1   SIZE - RAD=',I8,'  X=',I8,'  Y=',I8)
274               GOTO 900
275         815   WRITE(6,816) NUMR,RADIUS,WIDTH
276         816   FORMAT(//,' ***ERROR - SIZE EXCEEDS DIMENSIONS OF ARRAY
277              1  (MAX=160) RANGE CELLS=',I5,'RADIUS=',I8,' WID',I8)
278               GOTO 900
279         820   WRITE(6,821)CAT,I,J
280         821   FORMAT(//,5H*****,'CAT OUT OF RANGE. CAT,I,J ',3I6)
281               GOTO 900
282       C
283       C  WRITE OUT DATA BASE MATRIX FOR
284       C  VERIFICATION
285       C
286         600   OTF = 1
287               WRITE(02) NUMB
288               WRITE(02) ((OT(L,M), L=1,4),M=1,NUMB)
289               ENDFILE(02)
290               NUMB = 0
291               GOTO 105
292         500   CONTINUE
293         900   WRITE(02) NUMB
294               WRITE(02) ((OT(L,M), L=1,4),M=1,NUMB)
295               WRITE(6,901) SUM,TOTCNT
296         901   FORMAT(10X,'***DONE***',I8,'  RECORDS WRITTEN',
297              &    10X,I8,' POINTS PROCESSED')
298               STOP
299               END
```

```
      1      C                          POLAR ARRAY
      2      C
      3      C  PART TWO OF DATA BASE CREATION
      4      C
      5            IMPLICIT INTEGER (A-Y)
      6            DIMENSION A(357,181),PRIOR(16,16),DAT(4,300)
      7            DIMENSION OT(357),C1(357,31),C2(357,31)
      8            DATA MSKCAT /077/
      9            DATA OCT2,SHIFT,CTR/0100,0100000,010000000007/
     10      C
     11            OTF = 0
     12      C
     13            READ(01) NUMCAT,HOLEFIX,MSKCAT
     14            WRITE(02)NUMCAT,HOLEFIX,MSKCAT
     15            DO 412 I=1,NUMCAT
     16       412  READ(01)(PRIOR(I,J),J=1,NUMCAT)
     17            READ(01) NUMR,NUMANG,RADIUS,WIDTH,N90,ALT,ZCTAU
     18            WRITE(02) NUMR,NUMANG,RADIUS,WIDTH,ALT,ZCTAU
     19      C
     20      C
     21      C
     22        5   READ(01,END=500) NUMB
     23            READ(01) ((DAT(L,M),L=1,4),M=1,NUMB)
     24      C
     25            DO 200 I=1,NUMB
     26            R = DAT(1,I)
     27            ANG = DAT(2,I)
     28            CAT1=AND(DAT(3,I),MSKCAT)
     29            ELV1=IRL(DAT(3,I),6)
     30            CAT2=AND(DAT(4,I),MSKCAT)
     31            ELV2=IRL(DAT(4,I),6)
     32            A(R,ANG)=ELV1*SHIFT + ELV2 + CTR + A(R,ANG)
     33            WORD=ANG/6 + 1
     34            BIT = MOD(ANG,6)*6
     35            TAG = FLD(BIT,6,C1(R,WORD))
     36            IF(CAT1 .EQ. 0) GOTO 150
     37            IF(TAG .EQ. 0) FLD(BIT,6,C1(R,WORD))=CAT1
     38            IF(TAG .EQ. 0) GOTO 150
     39            IF(PRIOR(TAG,CAT1) .EQ. CAT1) FLD(BIT,6,C1(R,WORD))=CAT1
     40       150  TAG = FLD(BIT,6,C2(R,WORD))
     41            IF(CAT2 .EQ. 0) GOTO 200
     42            IF(TAG .EQ. 0) FLD(BIT,6,C2(R,WORD))=CAT2
     43            IF(TAG .EQ. 0) GOTO 200
     44            IF(PRIOR(TAG,CAT2) .EQ. CAT2) FLD(BIT,6,C2(R,WORD))=CAT2
     45       200    CONTINUE
     46            GOTO 5
     47      C
     48      C
     49        6   READ(01,END=500) NUMB
     50            READ(01) ((DAT(L,M),L=1,4),M=1,NUMB)
     51      C
     52            DO 292 I=1,NUMB
```

```
      R = DAT(1,I)
      ANG = DAT(2,I)
      CAT2=AND(DAT(3,I),MSKCAT)
      ELV2=IRL(DAT(3,I),6)
      CAT1=AND(DAT(4,I),MSKCAT)
      ELV1=IRL(DAT(4,I),6)
      A(R,ANG)=ELV1*SHIFT + ELV2 + CTR + A(R,ANG)
      WORD=ANG/6 + 1
      BIT = MOD(ANG,6)*6
      TAG = FLD(BIT,6,C1(R,WORD))
      IF(CAT1 .EQ. 0) GOTO 600
      IF(TAG .EQ. 0) FLD(BIT,6,C1(R,WORD))=CAT1
      IF(TAG .EQ. 0) GOTO 600
      IF(PRIOR(TAG,CAT1) .EQ. CAT1) FLD(BIT,6,C1(R,WORD))=CAT1
600   TAG = FLD(BIT,6,C2(R,WORD))
      IF(CAT2 .EQ. 0) GOTO 292
      IF(TAG .EQ. 0) FLD(BIT,6,C2(R,WORD))=CAT2
      IF(TAG .EQ. 0) GOTO 292
      IF(PRIOR(TAG,CAT2) .EQ. CAT2) FLD(BIT,6,C2(R,WORD))=CAT2
292     CONTINUE
      GOTO 6
C
C
500   CONTINUE
      DO 400 II=1,N90
      I= N90 + 1 - II
      WORD=I/6 + 1
      BIT = MOD(I,6)*6
C
      DO 300 J=1,NUMR
      MULT = FLD(0,6,A(J,I))
      IF(MULT .EQ. 0) GOTO 250
      ELV1 = FLD(6,15,A(J,I))/MULT
      ELV2 = FLD(21,15,A(J,I))/MULT
     GOTO 251
250     ELV1=0
      ELV2=0
C
251   OT(J)=ILS(ELV2,6) + FLD(BIT,6,C2(J,WORD))
      A(J,I)=ILS(ELV1,6) + FLD(BIT,6,C1(J,WORD))
300     CONTINUE
      WRITE(02) (OT(K),K=1,NUMR)
      IF(I .LT.4) WRITE(6,21) (OT(K),K=1,NUMR,3)
21    FORMAT(17(1X,06))
400   CONTINUE
C
      DO 420 I=1,N90
      WRITE(02) (A(J,I),J=1,NUMR)
C
C ***************************************
C
C SPECIAL CODE TO ROTATE POLAR DATA BASE TO MAGNETIC NORTH
```

```
105       C  RECTANGULAR DATA BASE IS ROTATED 30 DEGREES FROM NORTH
106       C
107       C ****************************************
108       C
109             IF(OTF .GT. 0 .AND. I .GT. 120) WRITE(03) (A(J,I),J=1,NUMR)
110       C
111             IF(I .LT. 4) WRITE(6,21) (A(J,1),J=1,NUMR,3)
112             DO 410 J=1,NUMR
113        410   A(J,I)=0
114        420   CONTINUE
115       C
116             DO 220 L=1,NUMR
117             DO 220 L2=1,31
118             C1(L,L2)=0
119        220  C2(L,L2)=0
120             OTF = 1 + OTF
121             CALL FCLOSE(01)
122             IF(OTF .LT. 2) GOTO 6
123             STOP
124             END
```

07-13-78 09.809

```
      1       C
      2       C  WRITE POLAR DATA BASE
      3       C
      4             IMPLICIT INTEGER (A-Y)
      5             DIMENSION IN(355)
      6       C
      7             REWIND (03)
      8             REWIND (02)
      9             READ(02) NUMCAT,HOLEFIX,MSKCAT
     10             READ(02) NUMR,NUMANG,RADIUS,WIDTH,ALT,ZCTAU
     11             WRITE(01) NUMCAT,HOLEFIX,MSKCAT
     12             WRITE(01) NUMR,NUMANG,RADIUS,WIDTH,ALT,ZCTAU
     13       C
     14             DO 100 I=1,60
     15             READ(03) IN
     16             WRITE(6,12) IN
     17          12 FORMAT(12(1X07))
     18             WRITE(01) IN
     19         100 CONTINUE
     20       C
     21             DO 110 I=61,720
     22             READ(02) IN
     23             WRITE(01) IN
     24         110 CONTINUE
     25       C
     26             WRITE(6,10)
     27          10 FORMAT(' WE ARE DONE ')
     28             STOP
     29             END
```

```
      C                    ARRAY FIX
      C
      C
      C       PURPOSE IS TO PATCH UP HOLES IN THE CENTER
      C       OF THE POLAR DATA BASE SO EVERY CELL HAS
      C       DATA.  HOLES WERE CAUSED IN CONVERSION FROM
      C       CARTESIAN TO POLAR COORDINATES.  HOLES WORKS
      C       ON THE FIRST 50 CELLS IN EACH RAY RADIATING
      C       FROM CENTER OF DATA BASE.
      C
      C
            IMPLICIT INTEGER (A-Y)
            REAL FLOAT
            DIMENSION BUF(500),IN(720,50)
            DATA OCT2,INTAP,OT /0100,01,02/
            READ(INTAP) NUMCAT,HOLEFIX,MSKCAT
            READ(INTAP)NUMR,NUMANG,RAD,WIDTH,ALT,ZCTAU
      C
      C  READ IN BAD DATA 1 RAY AT A TIME.
      C  FILL "IN" ARRAY WITH FIRST 50 POINTS OF DATA,
      C  ALL THE HOLES WILL BE FOUND IN THESE FIRST 50 PTS.
      C  "IN" HAS THE FIRST 50 PTS. FOR ALL 720 RAYS.
      C  "BUF"  IS ONLY A BUFFER TO READ IN FROM TAPE.
      C
      C
            DO 100 I=1,NUMANG
          6 READ(INTAP,END=1000)(BUF(J),J=1,NUMR)
            DO 90   J=1,HOLEFIX
       90   IN(I,J)=BUF(J)
      100   CONTINUE
      C
      C
      C  DUMP OUT POINTS AROUND DAM TO FIND WHY SO MANY
      C  HOLES GET FILLED WITH RESERVOIR CAT.
      C
      C
            DO 23 RAY=1,720
            PRINT,'    POINTS FOR RAY ',RAY
       23   PRINT 231,(IN(RAY,CK),CK=1,10)
      231   FORMAT(5X,10(O6,3X))
      C
      C
            DO 500 I=1,HOLEFIX
            MELV=0
            MLIN=0
            CNT=0
            DIF=0
            ZLOPE=0
      C
            DO 150 J=1,NUMANG
            NXT=J
            IF(IN(J,I) .NE. 0)GOTO 160
```

```
      150   CONTINUE
            WRITE(6,12)I
      12    FORMAT(' NO DATA IN LINE ',I4)
            GOTO 500
      160   LAST=0
            REFELV=IN(NXT,I)/OCT2
      C
            DO 400 J=1,NUMANG
            CNT=CNT+1
            IF(IN(J,I) .NE. 0)GOTO 350
            IF(LAST .EQ. 0)GOTO 180
            IF((J-LAST) .LT. (NXT-J))GOTO 200
            IF(NXT .GT. NUMANG)GOTO 200
      180   CAT=FLD(30,6,IN(NXT,I))
            IF(CAT .NE. 17)GOTO 190
            IF(LAST .NE. 0)CAT=FLD(30,6,IN(LAST,I))
      190   FLD(30,6,IN(J,I))=CAT
            ELV=REFELV+ZLOPE*CNT
            FLD(18,12,IN(J,I))=ELV
            IF(J .EQ. 1)GOTO 400
            DIF=IABS(ELV-FLD(18,12,IN(J-1,I)))
            MELV=MAX(MELV,DIF)
            GOTO 400
      C
      200   CAT=FLD(30,6,IN(LAST,I))
            IF(CAT .NE. 17)GOTO 210
            IF(NXT .LE. NUMANG)CAT=FLD(30,6,IN(NXT,I))
      210   FLD(30,6,IN(J,I))=CAT
            ELV=REFELV+ZLOPE*CNT
            FLD(18,12,IN(J,I))=ELV
            IF(J .EQ. 1)GOTO 400
            DIF=IABS(ELV-FLD(18,12,IN(J-1,I)))
            MELV=MAX(MELV,DIF)
            GOTO 400
      C
      350   LAST=NXT
            CNT=0
            DO 370 K=LAST+1,NUMANG
            NXT=K
            IF(IN(K,I) .NE. 0)GOTO 371
      370   CONTINUE
            NXT=NUMANG+1
            ZLOPE=0
            REFELV=IN(LAST,I)/OCT2
            GOTO 400
      371   REFELV=IN(LAST,I)/OCT2
            NUML=NXT-LAST
            ZLOPE=FLOAT(IN(NXT,I)/OCT2 - IN(LAST,I)/OCT2)/FLOAT(NUML)
            MLIN=MAX(MLIN,NUML)
      400   CONTINUE
      C
            WRITE(6,60)I,MLIN,MELV
```

```
      105       60    FORMAT(' RING',I3,'  MLIN AND MELV ',2I6)
      106       500   CONTINUE
      107       C
      108       C
      109       C     WRITE PATCHED DATABASE TO TAPE
      110       C
      111       C
      112             REWIND(INTAP)
      113             READ(INTAP)
      114             READ(INTAP)
      115             WRITE(OT)NUMR,NUMANG,RAD,WIDTH,ALT,ZCTAU,NUMCAT
      116       C
      117             DO 600 I=1,NUMANG
      118       666   READ(INTAP,END=1001)(BUF(J),J=1,NUMR)
      119             DO 550 J=1,HOLEFIX
      120       550   BUF(J)=IN(I,J)
      121             WRITE(OT)(BUF(J),J=1,NUMR)
      122             IF(MOD(I,3) .EQ. 1)WRITE(6,70)(BUF(J),J=1,100)
      123       70    FORMAT(/16(1X,06))
      124       600   CONTINUE
      125             GOTO 900
      126       1000  CALL FCLOSE(INTAP)
      127             GOTO 6
      128       1001  CALL FCLOSE(INTAP)
      129             GOTO 666
      130       900   STOP
      131             END
```

C.4.2 Reference Scene Computer Program

This computer program was written in FORTRAN for implementation on a Honeywell 66/60. It consists of two subprograms:

(A) Power

(B) Greytone

```
13-78   09.813                POWER                                    L

      1       C                    POWER
      2       C
      3       C  PROGRAM ACCEPTS DATA MATRIX IN POLAR COORDINATES FROM
      4       C  FILECODE 01 (CREATED BY ARRAY FIX) AND PRODUCES A
      5       C  SIMULATION
      6       C
      7       C
      8               IMPLICIT INTEGER (A-Y)
      9               REAL FLOAT,SIN,COS,ARCOS,RMS
     10               LOGICAL TAG
     11               COMMON /RANDOM/ ISEED
     12               COMMON ZTAB(1000),ZCF(16,4),ZS(16),LEN(400),TAG
     13               COMMON /IO/ BASE(400,3),CAT(400,3)
     14               COMMON /OT/ GT(400),ZGT(400,3),ZOT(400),ZSTRT(400,2)
     15               COMMON /PARAM/ NUMR,NUMANG,RADIUS,WIDTH,ZALT,ZCTAU,KLAPP,NUMCAT
     16       C
     17               DATA L1,L2,L3,GTREF/1,2,3,347
     18               DATA OCT2/0100/
     19       C
     20               ISEED = 1231236907
     21               KLAPP = 0
     22       C
     23       C
     24               READ(01)  NUMR,NUMANG,RADIUS,WIDTH,ALT,ZCTAU,NUMCAT
     25        10     FORMAT(4I8)
     26               READ(05,10) ALT
     27       C
     28               ZALT=FLOAT(ALT)
     29       C
     30               WRITE(6,11)
     31         11    FORMAT(//,30X,'THIRD ORDER COEFFICIENTS FOR SIGMA 0')
     32       C
     33               DO 100 I=1,NUMCAT
     34               READ(05,15) (ZCF(I,J),J=1,4)
     35               WRITE(6,15) (ZCF(I,J),J=1,4)
     36         100   CONTINUE
     37        15     FORMAT(4E14.7)
     38       C
     39               READ(05,2) ZS
     40         2     FORMAT(10(F6.2))
     41       C  LENGTH OF RESOLUTION CELL IN AZIMUTH INCREASES WITH RANGE
     42       C  THE ARRAY - LEN - CONTAINS THE RESOLTION CELL LENGTH
     43       C  (TIMES 2) AT EACH RANGE BIN.  USED TO CALCULATE LOCAL
     44       C  ACROSS TRACK SLOPE
     45       C
     46               R = -WIDTH/2
     47               DO 105 I=1,NUMR
     48               R = R + WIDTH
     49        105    LEN(I)= 3.1416*4. *R/FLOAT(NUMANG) + 1
     50       C
     51       C  WID2 = 2 TIMES WIDTH OF RESOLUTION CELL IN TRACK DIRECTION
     52       C       (A CONSTANT VALUE FOR THIS SIMULATION)
```

```
      C
            WID2 = WIDTH + WIDTH
      C
            ZCOS35 = COS(35/57.295)
            ZSIN35 = SIN(35./57.295)
            GD35 = ALT*ZSIN35/ZCOS35
            CEL35 = GD35/WIDTH
      C
      C
      C  TRANSFER PARAMETERS TO TAPE FOR OUTPUT ROUTINE
      C
            WRITE(02) NUMR,NUMANG,RADIUS,WIDTH,ALT,ZCTAU
            WRITE(6,707) NUMR,NUMANG,RADIUS,WIDTH,ALT
       707  FORMAT(' NUMR,NUMANG,RADIUS,WIDTH,ALT=',5I8)
      C
      C
            DO 110 I=1,1000
       110  ZTAB(I)= ARCOS(FLOAT(I)/1000.)
      C
            CALL NEXT(2,IEV)
            IF(IEV .GT. 0) GOTO 800
            DO 120 I=1,NUMR
            BASE(I,1)=BASE(I,2)
       120  CAT(I,1)=CAT(I,2)
      C
            DO 300 ANG = 1,NUMANG
            TAG=.FALSE.
            IF(MOD(ANG,30) .EQ. 0) TAG=.TRUE.
            IF (TAG) WRITE(6,23)ANG
       23   FORMAT(1H1,4HANG=,I3,/,4HALOC,7X,3HCAT,4X,6HSIGMAD,10X,5HPOWER,
           1    5X,10HFADE POWER,4H  GT)
            IF(ANG .LT. NUMANG) GOTO 160
            DO 205 I=1,NUMR
            BASE(I,L3)=BASE(I,L2)
       205  CAT(I,L3)=CAT(I,L2)
            GOTO 161
      C
       160  CALL NEXT(L3,IEV)
            IF(IEV .GT. 0) GOTO 800
       40   FORMAT(1X,30I4)
       161  CONTINUE
      C
            ZM = FLOAT(BASE(CEL35,L2)-ALT)/FLOAT(GD35)
      C
            DO 270 ROW=1,NUMR
            ROW1 = ROW -1
            IF(ROW1 .LE. 0) ROW1 = 1
            ROW2 = ROW + 1
            IF(ROW2 .GT. NUMR) ROW2 = NUMR
      C
            ZDELT = FLOAT(BASE(ROW2,L2)-BASE(ROW1,L2))/FLOAT(WID2)
            ZY = ABS(BASE(ROW,L3)-BASE(ROW,L1))
```

```
      ZHYP = SQRT(ZY*ZY + LEN(ROW)**2)
      ZRHO = ZY/FLOAT(LEN(ROW))
      ZCOSRHO = FLOAT(LEN(ROW))/ZHYP
C
      NALT = ALT - BASE(ROW,L2)
      IF(ROW .GT. CEL35) GOTO 230
      ZSINTH = ZSIN35
      ZCOSTH = ZCOS35
      GOTO 250
C
230   ZGDIS = ROW*WIDTH
      Y = ZM*ZGDIS + ALT
      IF( Y .GT. BASE(ROW,L2)) GOTO 270
      ZM = FLOAT(BASE(ROW,L2)-ALT)/ZGDIS
      ZSR = SQRT(ZGDIS**2 + NALT**2)
      ZSINTH = ZGDIS/ZSR
      ZCOSTH = FLOAT(NALT)/ZSR
C
250   CONTINUE
      IF(ZCF(CAT(ROW,L2),1) .LT.100.) GOTO 251
      ZOT(ROW)=10.
      GOTO 270
251   CALL RTPWR(ZRHO,ZCOSRHO,ZDELT,NALT,CAT(ROW,L2),ZCOSTH,
     1  ZSINTH,ZPWR)
      ZOT(ROW)= ZOT(ROW)+ZPWR
270   CONTINUE
C
      WRITE(02) (ZOT(J),J=1,NUMR)
C
      T = L1
      L1=L2
      L2=L3
      L3=T
C
C  OUTPUT LINE OF GREYTONE IMAGE (STILL IN POLAR FORMAT) TO
C  TEMP FILE
C
      DO 290 K=1,NUMR
      ZGT(K,L3)=0
290   ZOT(K)=0
300   CONTINUE
      STOP
800   WRITE(6,801) ANG
801   FORMAT(' RAN OUT OF DATA AT RECORD ',I5)
      STOP
      END
```

```
      SUBROUTINE NEXT(LINE,IEV)
      IMPLICIT INTEGER (A-Y)
      REAL RMS
C
      COMMON /RANDOM/ ISEED
      COMMON /PARAM/ NUMR,NUMANG,RADIUS,WIDTH,ZALT,ZCTAU,KLAPP,NUMCAT
      COMMON /IO/ BASE(400,3),CAT(400,3)
      DATA MASK,TREES / 077,2/
C
      READ(01,END=900) (BASE(I,LINE),I=1,NUMR)
C
      DO 100 I=1,NUMR
      CAT(I,LINE) = AND(BASE(I,LINE),MASK)
      BASE(I,LINE) = IRL(BASE(I,LINE),6)
C
C  SPECIAL ELEVEATION ADJUSTMENT FOR TREE CATAGORY
C
      IF(CAT(I,LINE) .NE. TREES) GOTO 100
      BASE(I,LINE) = BASE(I,LINE) + 70 + RMS(ISEED)*10
  100 CONTINUE
C
      RETURN
  900  IEV = 1
      RETURN
      END
```

```
13-78   09.816                                                          L

      1             SUBROUTINE RTPWR(RHO,COSRHO,DELT,NALT,ICAT,COSTH,SINTH,PWR)
      2       C
      3             LOGICAL TAG
      4             REAL RMS
      5             COMMON /RANDOM/ ISEED
      6             COMMON /PARAM/ NUMR,NUMANG,RADIUS,WIDTH,ZALT,ZCTAU,KLAPP,NUMCAT
      7             COMMON TABLE(1000),CF(16,4),S(16),LEN(400),TAG
      8             DATA FUDGE /-1.9193/
      9       C
     10             ITRACE = 1
     11       C
     12             DATA SIGREF/-.8/
     13             BASEALT = ZALT
     14             IF(ICAT .GT. 1) GOTO 505
     15             PWR = 0
     16             RETURN
     17       C
     18       C  CALCULATE LOCAL ANGLE OF INCIDENCE
     19       C
     20        505  ACOS = (COSTH + SINTH*DELT)/SQRT(1.+DELT**2+RHO**2)
     21             IF(ACOS .LT. 0.) GOTO 800
     22             NLOC = ACOS*1000.
     23             ALOC = TABLE(NLOC)*57.295
     24       C
     25             IF(NLOC .LT. 6 .OR. NLOC .GT. 995) ALOC = ARCOS(ACOS)*57.295
     26       C
     27       C  CALCULATE SIGMA ZERO FOR GIVEN CATAGORY AT THE LOCAL ANGLE
     28       C  OF INCIDENCE JUST CALCULATED
     29       C
     30             SIGO = ALOC*(ALOC*(ALOC*CF(ICAT,1)+CF(ICAT,2))+CF(ICAT,3))
     31            &       +CF(ICAT,4)
     32             IF(KLAPP .EQ. 1)SIGO = S(ICAT) + 10*ALOG10(ACOS) - FUDGE
     33             SIGO = SIGO/10. - SIGREF
     34       C
     35       C  THDELT = SINE OF ANGLE THETA-DELT
     36       C  WHICH IS NEEDED FOR THE POWER FORMULA
     37       C
     38             IF(DELT .LT. .05) GOTO 160
     39             THDELT = ABS((SINTH-COSTH*DELT)/SQRT(1.+DELT**2))
     40             GOTO 161
     41        160  THDELT = SINTH
     42        161  IF(THDELT .LT. .001) GOTO 810
     43       C
     44       C
     45             ALT = (BASEALT)/NALT
     46       C
     47       C  POWER EQUATION
     48       C
     49             PWR = (10**SIGO)*(ALT**3)/(2*COSTH*COSRHO*THDELT)
     50             GOTO 900
     51       C
     52       C
```

```
07-13-78   09.816

      53        800   IF(ITRACE .GT. 3) WRITE(6,801) COSTH,DELT,ACOS
      54        801   FORMAT(' DELTA IS > THETA ',3F10.4)
      55              PWR = 0
      56              GOTO 900
      57        810   IF(ITRACE .GT. 3) WRITE(6,811) SINTH,COSTH,DELT,RHO,THDELT
      58        811   FORMAT(' DELTA = THETA',5F12.4)
      59              PWR = 10.
      60        900   IF (PWR .LT. .001) PWR=.001
      61              GT=ALOG10(PWR)*32.+34.
      62              IF(TAG) WRITE(6,23)IFIX(ALOC),ICAT,SIGO,PWR1,PWR,IFIX(GT)
412 PWR1 IS NOT DEFINED
      63        23    FORMAT(I4,I10,F10.2,2E15.4,I4)
      64              RETURN
      65              END
```

```
      C                    GRAYTONE
      C
      C*****************************************************************
      :
      C
      C
      C       PARAMETERS TO BE ADJUSTED TO RUN THIS PROGRAM
      C            1.)  M,N    THE # OF CELLS TO AVERAGE
      C            2.)  RES    DETERMINES THE SIZE OF THE IMAGE
      C            3.)  NFILE WHICH DETERMINES WHICH RECORD
      C                       TO WRITE TO ON THE OUTPUT TAPE
      C            4.)  DICO FLAG,  WHICH DETERMINES OUTTAP FORMAT
      C                       (DICO=1 FOR DICO FORMAT,0 FOR IDECS).
      C
      C*****************************************************************
      C
      C       A CHANGE OF RESOLUTION CHANGES DIMENSIONS
      C       OF ARRAYS RECORD AND OT
      C       RECORD(RES) , OT(WORD)  WORD = RES/6 + 1
      C
      C       MOST OTHER DIMENSIONS DEPEND ON NUMR WHICH IN TURN
      C       DEPENDS ON THE RADIUS
      C       NUMR = RADIUS/WIDTH
      C       ZIN(NUMR),ZPWR(NUMR,12),ZUM(NUMR),IOT(NUMR)
      C       ZAVE(NUMR,12),BASE(NUMR,121),IN(NUMR)
      C
      C       A CHANGE IN DIMENSIONS SHOULD BE ACCOMPANIED
      C       BY A CHANGE IN THE LIMITS CARDS
      C
      C*****************************************************************
      C
      C       IF IDECS OUTPUT IS DESIRED,CHECK THE FOLLOWING
      C       DICO=0 IN DATA STATEMENT
      C       FFILE ON OUT TAPE INCLUDES ONLY A BUFSIZ SO THAT
      C       EACH LOGICAL RECORD IS A SCANLINE. (BUFSIZ=RES+20)
      C       PUT A DEN5 ON OUTPUT TAPE7 CARD
      C
      C       IF DICO OUTPUT IS DESIRED,CHECK THE FOLLOWING
      C       DICO=1 IN DATA STATEMENT
      C       FFILE IS PUT IN SPECIFYING NSTDLB,NOSRLS,ETC.
      C$       FFILE   02,NSTDLB,NOSRLS,FIXLNG/154,BUFSIZ/154
      C       WORD = RES/6 + 1
      C       REMOVE DEN5 FROM OUTPUT TAPE7 CARD
      C
      C*****************************************************************
      :
      C
      C
            IMPLICIT INTEGER (A-Y)
            REAL ALOG10,FLOAT
            COMMON ZIN(360),ZPWR(360,12)
            DIMENSION L(12),ZUM(360),IOT(360),ZAVE(360,12)
```

```
      C
            DATA L/1,2,3,4,5,6,7,8,9,10,11,12/
            DATA M,N,NFILE,DICO,RES /1,1,1,1,921/
            DATA GTREF /34/
      C
      C
            READ(02) NUMR,NUMANG,RADIUS,WIDTH,ALT,ZCTAU
            WRITE(6,607) NUMR,NUMANG,RADIUS,WIDTH,ALT,ZCTAU
  507   FORMAT(' NUMR,NUMANG,RADIUS,WIDTH',4I8,/,
     &     ' ALTITUDE AND PULSE LENGTH ARE ',I5,F10.5)
            WRITE(04)NUMR,NUMANG,RADIUS,WIDTH,ALT,ZCTAU
            WRITE(04)NFILE,DICO,RES
            WRITE(01) NUMR,NUMANG,RADIUS,WIDTH,ALT,ZCTAU
      C
            DO 100 I=2,12
            DO1 = I
            CALL GETLINE(DO1,N,NUMR,IEV)
            DO 90 J=1,NUMR
   90       ZAVE(J,I-1)=ZPWR(J,I)
  100       CONTINUE
      C
            STRT=7 - (M-1)/2
            END = STRT+M-1
      C
            DO 150 I=1,NUMR
            ZUM(I)=0.
            DO 140 J=STRT,END
  140       ZUM(I)= ZUM(I)+ZPWR(I,J)
  150       CONTINUE
      C
            OLD = STRT-1
            NEW = END
            KOMPLT = 0
      C
      C
      C
    5       PTS = M*(N-1)/2
      C
            DO 200 I=1,NUMR
            IF(I .LE. (N+1)/2) PTS = PTS+M
            IF( I .GE. NUMR-N/2) PTS = PTS-M
             IF( PTS .EQ. 0 )PTS=1
      C
            ZOT = ZUM(I)/PTS
            IF(ZOT .LT. .001) ZOT=.001
            IOT(I)= ALOG10(ZOT)*32. + GTREF
            IF(IOT(I) .GT. 63) IOT(I)=63
            IF(IOT(I) .LT. 0) IOT(I)=0
  200       CONTINUE
      C
            WRITE(01) (IOT(J),J=1,NUMR)
      C
```

```
      C
            KOMPLT = KOMPLT+1
            IF(KOMPLT .EQ. NUMANG) GOTO 950
      C
            T = L(1)
            DO 300 I=1,11
      300   L(I)=L(I+1)
            L(12)=T
      C
            IF(KOMPLT .GT. (NUMANG-11)) GOTO 400
            CALL GETLINE(L(12),N,NUMR,IEV)
            IF(IEV .GT. 1) GOTO 900
            GOTO 500
      C
      400   CNT=CNT+1
            DO 410 I=1,NUMR
      410   ZPWR(I,L(12))=ZAVE(I,CNT)
      C
      500   LOLD = L(OLD)
            LNEW=L(NEW)
            DO 510 I=1,NUMR
      510   ZUM(I)= ZUM(I)+ZPWR(I,LNEW)-ZPWR(I,LOLD)
            GOTO 5
      C
      900   WRITE(6,901) IEV,KOMPLT
      901   FORMAT(' ABNORMAL TERMINATION, IEV=',I5,' KOMPLT=',I5)
            STOP
      950   WRITE(6,951)
      951   FORMAT(' *** WE ARE DONE ***')
            STOP
            END
```

```
      1              SUBROUTINE GETLINE(LINE,NAVG,NUMR,IEV)
      2              IMPLICIT INTEGER (A-Y)
      3              COMMON ZIN(360),ZPWR(360,12)
      4      C
      5              READ(02,END=900) (ZIN(J),J=1,NUMR)
      6      C
      7              ZUM=0.
      8              IPTR = 1
      9              OPTR = 1
     10      C
     11       100   ZUM = ZIN(IPTR) + ZUM
     12             IPTR = IPTR+1
     13             IF(IPTR .LE. (NAVG+1)/2) GOTO 100
     14      C
     15             ZPWR(OPTR,LINE)=ZUM
     16             OPTR = OPTR+1
     17      C
     18       120   ZUM = ZUM + ZIN(IPTR)
     19             ZPWR(OPTR,LINE)= ZUM
     20             OPTR = OPTR+1
     21             IPTR = IPTR+1
     22             IF(IPTR .LE. NAVG) GOTO 120
     23      C
     24      C
     25             BOT = 1
     26       150   ZUM = ZUM +ZIN(IPTR)-ZIN(BOT)
     27             ZPWR(OPTR,LINE)=ZUM
     28             OPTR=OPTR+1
     29             IPTR=IPTR+1
     30             BOT = BOT+1
     31             IF(IPTR .LE. NUMR) GOTO 150
     32      C
     33       200   ZUM = ZUM - ZIN(BOT)
     34             ZPWR(OPTR,LINE)=ZUM
     35             OPTR = OPTR+1
     36             BOT = BOT+1
     37             IF(OPTR .LE. NUMR) GOTO 200
     38      C
     39             RETURN
     40      C
     41       900   WRITE(6,801)
     42       801   FORMAT(' UNEXPECTED END OF DATA')
     43             IEV=1
     44             RETURN
     45             END
```

C.4.3 Rectangular Conversion Computer Program

This computer program was written in FORTRAN for implementation on a Honeywell 66/60. It consists of two subprograms:

(A) Rectangular Create

(B) Rectangular Array

```
      C                    RECTANGULAR CREATE
      C
            IMPLICIT INTEGER (A-Y)
            REAL FLOAT,ARCOS
            DIMENSION BUF(600),TABLE(1000)
            DATA ZBMWD/.00875/
      C
            REWIND(04)
            READ(04) MAXDIS,MAXANG,RADIUS,WIDTH
            READ(04) NFILE,DICO,RES
            WRITE(03)RES,DICO,NFILE
      C
      C
            WRITE(6,9) MAXDIS,MAXANG,RADIUS,WIDTH
        9   FORMAT(' INPUT PARAMETERS - MAXDIS,MAXANG,RADIUS,WIDTH=',5I8)
            IF(WIDTH .EQ. 0) WIDTH = RADIUS/MAXDIS
      C
            DO 100 I=1,1000
      100   TABLE(I)= ARCOS(FLOAT(I)/1000.)/ZBMWD
      C
      C
      C
            HFRES = RES/2
            IF(2*HFRES .LT. RES) HFRES = HFRES +1
            ZSIZE = FLOAT(2*RADIUS)/FLOAT(RES) - .001
      C
      555   CONTINUE
      C
            DO 600 I=1,RES
            NX = I*ZSIZE - RADIUS
            IF(NX .EQ. 0) NX = ZSIZE/2
            ZX = ABS(NX)
            NX2 = NX * NX
            IF(NX.LT. 0 .OR. OTF .GT. 0) GOTO 200
            OTF = 1
            NEG = -1
            WRITE(03) NEG
      C
      200   DO 575 J=1,RES/2
            NY = J*ZSIZE -ZSIZE/2
            NY2 = NY * NY
            RDIS = SQRT(NX2+NY2)
            R = RDIS/WIDTH
            IF( R. GT. MAXDIS) GOTO 580
            COSANG = ZX/FLOAT(RDIS) * 1000.
            ANG = TABLE(COSANG)
            IF(COSANG .LT. 6 .OR. COSANG .GT. 994)
           &  ANG = ARCOS(ZX/FLOAT(RDIS))/ZBMWD
            IF(ANG .EQ. 0) ANG = 1
      C
      C
            IF(R .EQ. OLDR .AND. ANG .EQ. OLDANG) GOTO 575
```

```
      53              CNT = CNT+1
      54              IF(CNT .GT. 1) GOTO 585
      55              OLDR = R
      56              OLDANG = ANG
      57        585   IF(CNT .GT. 600) GOTO 800
      58              FLD(0,10,BUF(CNT))=J-1
      59              FLD(10,10,BUF(CNT))=OLDR
      60              FLD(20,10,BUF(CNT))=OLDANG
      61              OLDR = R
      62              OLDANG = ANG
      63        575   CONTINUE
      64        580   CNT = CNT + 1
      65              FLD(0,10,BUF(CNT))=J-1
      66              FLD(10,10,BUF(CNT))=OLDR
      67              FLD(20,10,BUF(CNT))=OLDANG
      68              OLDR = R
      69              OLDANG = ANG
      70              WRITE(03) CNT
      71              WRITE(03) (BUF(I9),I9=1,CNT)
      72              TOT = TOT + CNT
      73              MOST = MAX(MOST,CNT)
      74              CNT = 0
      75        590   CONTINUE
      76              GOTO 950
      77        800   WRITE(6,805) I
      78        805   FORMAT(' BUFFER OVERFLOW AT LINE',I5)
      79        C
      80        C
      81        950   WRITE(6,951) TOT,MOST
      82        951   FORMAT(' WE ARE DONE ',2I8)
      83              STOP
      84              END
```

```
      C                    RECTANGULAR ARRAY
      C
            IMPLICIT INTEGER (A-Y)
            DIMENSION BASE(360,121),BUF(600),RECORD(921),OI(154)
            DIMENSION IN(360)
      C
      C     IF DICO OUTPUT IS DESIRED SET DICO=1
      C     OTHERWISE IDECS OUTPUT FORMAT
      C
      C
      C
            REWIND(01)
            REWIND(03)
      C
            READ(01) NUMR,NUMANG,RADIUS,WIDTH
            READ(03)RES,DICO,NFILE
      C
            WRITE(6,141)RES,DICO,NFILE
        141 FORMAT('  RES,DICO,NFILE = ',3I8//)
      C
      C     POSITION OUTPUT TAPE TO PROPER FILE WITH POST.
      C     FILE IS CHOSEN BY SETTING 'NFILE' IN DATA STATEMENT.
      C
            IF(NFILE .NE. 1)CALL POST(02,0,NFILE,1,ERR)
            IF(ERR .NE. 0)WRITE(6,223)ERR
        223 FORMAT(' TROUBLE WITH POST ',I5)
            IF (ERR .NE. 0) STOP
      C
            HFRES = RES/2
      C
            N180 = NUMANG/2
            N90 = NUMANG/4
      C
            DO 110 I=1,NUMANG
            READ(01) (IN(J),J=1,NUMR)
            ANG = MOD((I+545),NUMANG)+1
            WORD= (ANG-1)/6 + 1
            BIT = MOD((ANG-1),6)*6
      C
            DO 105 J=1,NUMR
        105 FLD(BIT,6,BASE(J,WORD))= IN(J)
        110 CONTINUE
      C
          5 DO 500 I=1,RES
            NUM = 1
            READ(03) CNT
            IF(CNT .GT. 0) GOTO 7
            OTF = 1
            READ(03) CNT
          7 READ(03) (BUF(I9),I9=1,CNT)
            STRT = 1
            END = FLD(0,10,BUF(1))
```

```
      53              R = FLD(10,10,BUF(1))
      54              ANG = FLD(20,10,BUF(1))
      55              ANG = N90+1-ANG
      56      C
      57      C
      58       120    IF(OTF .GT. 0) GOTO 300
      59              WORD = (ANG-1)/6 + 1
      60              BIT = MOD((ANG-1),6)*6
      61              LEFT = FLD(BIT,6,BASE(R,WORD))
      62              TH = N180-ANG + 1
      63              WORD = (TH-1)/6 + 1
      64              BIT = MOD((TH-1),6)*6
      65              RGT = FLD(BIT,6,BASE(R,WORD))
      66              GOTO 121
      67      C
      68       300    TH = NUMANG - ANG + 1
      69              WORD = (TH-1)/6 + 1
      70              BIT = MOD((TH-1),6)*6
      71              LEFT = FLD(BIT,6,BASE(R,WORD))
      72              TH = N180 + ANG
      73              WORD = (TH-1)/6 + 1
      74              BIT = MOD((TH-1),6)*6
      75              RGT = FLD(BIT,6,BASE(R,WORD))
      76       121    IF(NUM .EQ. 1) RECORD(HFRES)=RGT
      77              DO 200 J=STRT,END
      78              J1=HFRES+J
      79              J2 = HFRES-J
      80              RECORD(J1)=RGT
      81       200    RECORD(J2)=LEFT
      82      C
      83              NUM = NUM+1
      84              IF(NUM .GT. CNT) GOTO 400
      85              STRT = END+1
      86              END = FLD(0,10,BUF(NUM))
      87              R = FLD(10,10,BUF(NUM))
      88              ANG = FLD(20,10,BUF(NUM))
      89              ANG = N90+1-ANG
      90              GOTO 120
      91       400    CONTINUE
      92      C
      93              IF(DICO .EQ. 1)GOTO 420
      94      C
      95      C       ********** THIS SECTION WRITES TO IDECS ********
      96      C
      97      C
      98      C
      99              WRITE(02) (RECORD(I9),I9=1,RES)
     100       22     FORMAT(1X,25I3)
     101       23     FORMAT(1X,123I1)
     102              GOTO 401
     103      C
     104      C
```

```
      C
      C
      C     *****  THIS SECTION WRITES IN DICO FORMAT  *****
      C
 420  DO 444 K=1,RES
      KK=K-1
      WORD = KK/6 + 1
      BIT = MOD(KK,6)*6
      DATA = 63 -  RECORD(K)
      FLD(BIT,6,OT(WORD))=DATA
 444  CONTINUE
      WRITE(02) OT
 401  IF(MOD(I,4) .NE. 0)GOTO 450
      CALL GREYMAP(RECORD,0,63,001,240,2,06)
      CALL GREYMAP(RECORD,0,63,241,480,2,10)
      CALL GREYMAP(RECORD,0,63,481,720,2,11)
      CALL GREYMAP(RECORD,0,63,721,921,2,12)
      :
 450  DO 490 M=1,RES
 490  RECORD(M)=0
 500  CONTINUE
      WRITE(6,501)
 501  FORMAT(//,' THAT IS ALL FOLKS')
      ENDFILE (02)
      WRITE(02)RES
 900  STOP
      END
```

```
5  07-13-78    09.828

          1                 SUBROUTINE GREYMAP(ARRAY,MIN,MAX,START,STOP,STEP,FC)
          2                 IMPLICIT INTEGER (A-Y)
          3                 CHARACTER LINE(3,125),DENSITY(3,13)
          4                 DIMENSION ARRAY(1)
          5        C
          6                 DATA (DENSITY(1,J),J=1,13) /1H ,1H.,1H',1H:,1H+,1H=,1H*,1HX,1
          7                &   1HM,1HT,1HM,1HM/
          8                 DATA (DENSITY(2,J),J=1,13) /6*1H ,1H(,1H.,1H=,1H=,1HN,1HW,1HW
          9                 DATA (DENSITY(3,J),J=1,13) / 10*1H ,1H(,1HS,1HS/
         10        C
         11        C
         12                 ZQUANT = FLOAT(MAX - MIN)/13.
         13        C
         14                 CNT = 0
         15                 DO 100 I=START,STOP,STEP
         16                 CNT = CNT+1
         17                 IF(CNT .GT. 125) GOTO 200
         18                 VALUE = FLOAT(ARRAY(I)-MIN)/ZQUANT + 1
         19                 IF(VALUE .LE. 0) VALUE = 1
         20                 IF(VALUE .GT. 13) VALUE = 13
         21                 DO 90 J=1,3
         22           90    LINE(J,CNT) = DENSITY(J,VALUE)
         23          100    CONTINUE
         24        C
         25          200    WRITE(FC,10) (LINE(1,J),J=1,CNT)
         26                 WRITE(FC,12) (LINE(2,J),J=1,CNT)
         27                 WRITE(FC,12) (LINE(3,J),J=1,CNT)
         28           10    FORMAT(1X,125A1)
         29           12    FORMAT(1H+,125A1)
         30                 RETURN
         31                 END
```