NOAA Technical Report NOS 127 CGS 12

Cartographic Generalization

Reston, VA
July 1988

U.S. DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration
National Ocean Service
Charting and Geodetic Services

NOAA TECHNICAL PUBLICATIONS

National Ocean Service/Charting and Geodetic Services

The Office of Charting and Geodetic Services (C&GS), National Ocean Service, NOAA, plans and directs programs to produce charts and related information for safe navigation of the Nation's waterways, territorial seas, and national airspace. It establishes and maintains the horizontal, vertical, and gravity geodetic networks which comprise the National Geodetic Reference System.

C&GS coordinates planning and execution of surveying, charting, and related geophysical data collections to meet national goals. In fulfilling these objectives, it conducts geodetic, gravimetric, hydrographic, coastal mapping, and related geophysical surveys; analyzes, compiles, reproduces, and distributes nautical and aeronautical charts and geodetic and other related geophysical data; establishes national mapping and charting standards; and conducts research and development to improve surveying and cartographic methods, instruments, equipment, data analysis, and national reference system datums.

NOAA geodetic and charting publications as well as relevant publications of the former U.S. Coast and Geodetic Survey are sold in paper form by the National Geodetic Information Branch. To obtain a price list or to place an order, contact:

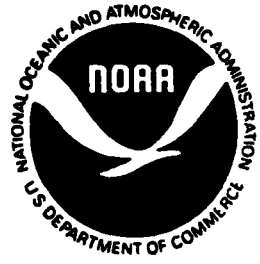
National Geodetic Information Branch (N/CG174)
Office of Charting and Geodetic Services
National Ocean Service, NOAA
Rockville, MD 20852

Make check or money order payable to: NOAA National Geodetic Survey. Do not send cash or stamps. Publications can also be charged to Visa, Mastercard, or prepaid Government Printing Office Deposit Account. Telephone orders are accepted (301 443-8631).

Publications can also be purchased over the counter at the National Geodetic Information Branch, 11400 Rockville Pike, room 26, Rockville, MD. (Do not send correspondence to this address.)

An excellent reference source for all Government publications is the National Depository Library Program, a network of about 1,400 designated libraries. Requests for borrowing Depository Library material may be made through your local library. A free listing of libraries in this system is available from the Marketing Office (mail stop MK), U.S. Government Printing Office, Washington, DC 20402 (202 275-3634).

NOAA Technical Report NOS 127 CGS 12



Cartographic Generalization

K. Stuart Shea

PAR Government Systems Corporation
Reston, VA
July 1988

U.S. DEPARTMENT OF COMMERCE
C. William Verity, Secretary

National Oceanic and Atmospheric Administration
William E. Evans, Under Secretary

National Ocean Service
Paul M. Wolff, Assistant Administrator

Charting and Geodetic Services
R. Adm. Wesley V. Hull, Director

PREFACE

This report on cartographic generalization addresses the need for the National Ocean Service (NOS) to select a technical and operational solution to the problem of cartographic generalization within the scope of the Automated Nautical Charting System II. Cartographic generalization is a most complex issue at NOS because of our unique approach to digital nautical cartography. A thorough understanding of the issues in cartographic generalization, such as, feature selection, point simplification, feature aggregation, feature displacement, are essential for developing an optimal implementation strategy.

The specific objective of this project is to develop a global conceptual model, while selecting and testing techniques that can potentially contribute to the operational solution.

This report was prepared by PAR Government Systems Corporation, June 1987, for the NOAA Charting Research and Development Laboratory, Charting and Geodetic Services, NOS, NOAA, Rockville, Maryland 20852, under a NOAA Science and Technology Grant, NOAA Contract No. 40AANC700230. This report is reprinted in its entirety.

CARTOGRAPHIC GENERALIZATION

by

PAR Government Systems Corporation
1840 Michael Faraday Drive, Suite 300
Reston, Virginia 22090
(703) 478-9690

Author:
Mr. K. Stuart Shea

prepared for the
CARTOGEN Program
in Response to
U.S. Department of Commerce
National Oceanic and Atmosperic Administration (NOAA)
National Ocean Service (NOS)
Charting and Geodetic Services (C&GS)
Order #40AANC700230

dated 30 June 1987

Automating Generalization: Fact or Fiction?

"Generalization is difficult to define, explain or specify in writing, or to restrict to bounds and limits." (A.M. Floyd)

"Only he who is master over the matter and can perform with his hands what his mind wishes, is able to generalise well." (E. von Sydow)

"The problems involved in practical generalization are so varied that it is virtually impossible to derive rules to cover all eventualities." (G.A. Montagano)

"...generalization depends on personal and subjective feelings," and therefore is "part of the 'art' that enters into the map making process." (M. Eckert)

"To a certain extent, generalization may be compared to the work of an artist." (A.J. Pannekoek)

"...the design factor in generalisation can clearly be based only on the cartographer's very personal, and therefore inherently biased, beliefs." (D.W. Rhind)

"...a largely undefined process and followed more or less the warm feeling of individual subjective intuition." (J. Neumann)

"...it can be seen that good generalization is, at least, a function of purpose plus objective evaluation...Since these are human factors, requiring intelligence and judgement, generalization is likely to remain outside the realm of electronic instrumentation." (D.E. Long)

"One of the difficulties...in an attempt to automate...is a consequence of the ambiguous, creative nature of the process which lacks definitive rules, guidelines, or systemization." (D.M. Brophy)

"...the automated generalization procedures should not necessarily be modeled on manual procedures." (G.E. Langren)

"Generalization algorithms exist at present, but more sophistication is needed." (D.R. Caldwell)

1.0 INTRODUCTION

This document is the Final Technical Report representing a summary of the research performed for the Cartographic Generalization (CARTOGEN) study conducted by PAR Government Systems Corporation (PGSC). This report has been prepared for the National Ocean Service's (NOS) Charting and Geodetic Services (C&GS). This introductory section will present a background to the problems addressed by the study, acquaint the reader to the CARTOGEN effort, and will briefly discuss the organization of the report.

1.1 Background

To provide a basis for discussing the development of a digital cartographic data generalization capability, we must first understand the trends toward the future which are prevalent throughout the Mapping, Charting, and Geodesy (MC&G) community. The transition towards large digital cartographic data bases to satisfy needs for multi-product exploitation, product flexibility, timely responsiveness to user demands, and lower production costs is a phenomenon familiar to all suppliers of cartographic information. Advances in processing techniques have great potential for benefits to be gained by all members of the MC&G community. Nowhere are these benefits more obvious than in the exploitation of digital cartographic data to support the production of nautical and bathymetric charts and related products.

The Nautical Charting Division (NCD) of the Office of Charting and Geodetic Services (C&GS) within the National Ocean Service (NOS) has the mission of providing nautical charts, marine related publications, and information required for safe and efficient transit of the Nation's coastal waters and inland waterways. NOS marine products also directly support development of offshore resources and defense of the Nation's coastal areas. NOS developed and implemented an automated chart production system in 1978 that partially supports various production requirements of nautical cartography. Although this computer assistance has enabled NOS to eliminate certain repetitive tasks, more critical activities associated with the nautical chart production process have not benefited with current automated technology. For example, document assessment, data evaluation, and response to demands for new products still are manual production efforts.

The production programs of the National Ocean Service are, however, currently in a state of transition to all-digital, softcopy production capabilities.[1] This transition includes

the establishment of uniform procedures relating to the collection, screening, evaluation, editing, symbolization, retrieval, and exchange of digital source and production data. The NOAA Charting R&D Laboratory intends to procure an integrated system to include: computer hardware, commercially available software, custom software, and design and developmental support for system integration to facilitate the implementation of the Automated Nautical Charting System II (ANCS II).[2] It is the goal of the ANCS II acquisition to provide a comprehensive computer system which can effectively and efficiently maintain a data management subsystem of approximately 50 million extensively attributed cartographic features as well as a chart production system of up to 3000 nautical chart panels which can be expressed in published graphic or digital form, and interactively display and provide full editing capability for both subsystems. There are six general categories of documents received by NOS for evaluation and application to the Marine Information Data Base (MIDB) and used as input to the ANCS II. These include: (1) letters; (2) blueprints; (3) Notice to Mariners; (4) Hydrographic Surveys; (5) Topographic Surveys; and (6) USGS Quads. About 16,000 documents are received each year, with an average of 7,200 documents per year applied to the MIDB, from which about 2.2 million features are selected. The amount of features selected from a document will vary greatly depending on the document. Given the number and size of documents to provide sufficient coverage for a production requirement, the size of this data base can be enormous.

In order for NOS to optimally exploit the digital cartographic data in the ANCS II production environment, the physical and memory size requirements of the digital information must be reduced and, concomitantly, exploited to the fullest extent. Current advances in digital storage technology (such as optical disk storage) allow large quantities of digital information to be collected and stored in limited physical environments. Even so, the large-area production requirements, and variety and types of information collected in digital form, cannot be stored in even the highest-technology storage media without resorting to some form of data compression, data elimination, or data reduction. Even if it could be, the NOS currently produces a variety of types, scales, and formats of products from the same data. As such, the cartographic information must be generalized to satisfy both the storage and scale constraints imposed by production requirements.

The overall problem of cartographic generalization as it relates to this report and the the ANCS II covers the entire range of the generalization process. This includes: (1) scale

change; and (2) feature generalization—including selection, simplification, conversion, refinement, smoothing, and compaction. The effective use of a digital cartographic data base supporting multi-product exploitation does present some technical challenges. By their very nature, cartographic data bases tend to be large for any coverage area of practical use. To be useful, the data must be of sufficiently high resolution; that is, the data must include all of the earth features (natural and man-made) of interest to a level of detail that will permit accurate navigation, landmark recognition, and production of varied product types. Memory size limitations within mass storage devices, along with the need to decrease overall processing costs, requires that the overall cartographic data volume is reduced to support the production environment.

1.2 An Overview of the Cartographic Generalization Study

The objective of this study was to provide an analysis of the cartographic reduction problem as it pertains to current and planned chart production systems at NOS. Approaches designed to yield high proportion data reduction for vector data were investigated. Implementation of these algorithms in a production scenario will allow for the generalization of the required cartographic data bases. A creative and well-engineered approach to this problem will provide a exciting exploitation of the MC&G data base. A project overview is presented below:

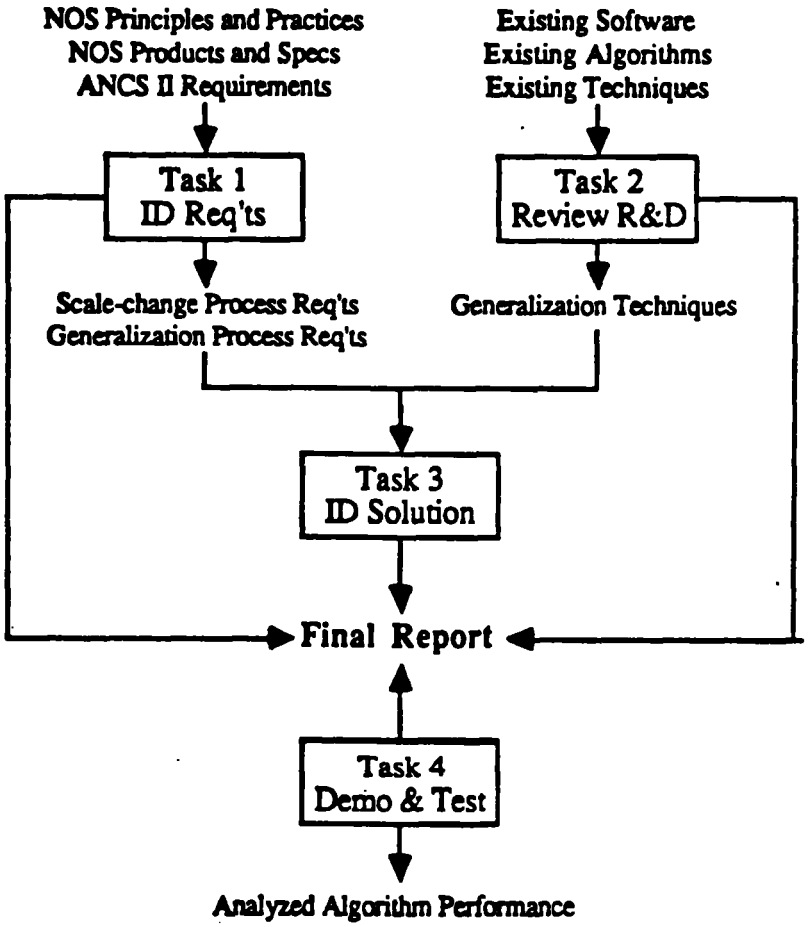
Task 1. **Requirements Identification**—comprised of a short-term analysis concentrating on reviewing current NOS's principles and practices, product specifications, and plans for the Automated Nautical Charting System II (ANCS II). The specific scale-change processes and other applications of generalization were reviewed.

Task 2. **Cartographic Generalization R&D Review**—concentrated on surveying the techniques, methods, standards, and requirements for cartographic generalization algorithms. This assessment served to: 1) categorize requirements for generalization; 2) determine the implicit and explicit relationships between the algorithms; and 3) evaluate the algorithms in terms of approach, operational issues, input/output, and strengths/weaknesses.

Task 3. **Solution Identification**—concentrated on identifying a global conceptual solution to the generalization problem.

Task 4. **Test/Demonstration of Contributing Techniques**—concentrated on developing an algorithm performance system which operates on limited data sets to examine specific generalization techniques.

An illustration of the CARTOGEN Project Model is presented below.



To accomplish these tasks the PGSC project team performed a comprehensive literature search examining the application of cartographic data generalization in disciplines such as cartography, image and picture processing, computer science, geology, electrical engineering, and computer vision. Secondly, discussions were held with the research component of NOAA's Charting R&D Laboratory. And, finally, PGSC drew upon its superb staff to conduct the program.

1.3 Organization of this Report

This report is organized into four (4) sections and two (2) appendices. The report proceeds from the general to the specific. In addition to this introduction, the material contained in each of the sections is highlighted as follows:

Section 2, **Overview of Cartographic Generalization**, outlines the particularly salient aspects of the generalization processes, including both scale-change and feature generalization process requirements and their impacts upon the current and planned cartographic production environment. A model for an automated generalization procedure is presented. This section also describes many of the cartographic generalization techniques which were discovered through our survey effort. This discussion proceeds in a top-down fashion by describing major coding techniques (selection, simplification, compaction, etc.) and then outlining appropriate examples.

Section 3, **NOS Generalization Requirements**, outlines some particularly salient aspects of the generalization processes at NOS, including both scale-change and generalization process requirements and their impacts upon the current and planned cartographic production environment.

Section 4, **Summary, Conclusions, and Recommendations**, synopsizes the report and describes what factors should be addressed when expanding this work towards a production capability.

Appendix A, **Bibliography**, provides a comprehensive set of references for cartographic generalization; these sources were used in the preparation of this report.

Appendix B, **Software Overview**, provides a comprehensive review of the software test environment that was developed under this effort.

1.4 Project References

References used throughout the preparation of this report are cited in the appropriate discussions of the individual generalization techniques (procedures) as well as accompanying individual algorithms. The sources listed in Appendix A were used as general references throughout the preparation of this report. This list is meant to illustrate to the reader the type and variation of sources used to compile the enclosed information; this information providing a solid foundation upon which the recommendations were made.

1.5 Terms and Abbreviations

Terms and abbreviations used throughout the preparation of this report are defined below.

CARTOGEN	Cartographic Generalization project
CG&S	Office of Charting and Geodetic Services of NOS
DBMS	Data Base Management System
MIDB	Marine Information Data Base
NOAA	National Oceanic and Atmospheric Administration
NOS	National Ocean Service of NOAA

NCD	Nautical Charting Division of the CG&S
MC&G	Mapping, Charting, and Geodetic Data
PGSC	PAR Government Systems Corporation

1.6 Endnotes

[1]Shea, K. Stuart (1987a).
[2]Department of Commerce (1986).

2.0 OVERVIEW OF CARTOGRAPHIC GENERALIZATION

The following discussion pertains to cartographic generalization. More specifically, the generalization of vector-based Mapping, Charting, and Geodesy (MC&G) data will be examined.

2.1 *Introduction to Cartographic Generalization*

All charts are reductions of some part of the environment. It would be impractical, if not impossible, to portray the entire Earth at a 1:1 scale. The reduction of the environment to a more comprehensible scale concomitantly yields a variety of undesirable consequences. These include: (1) a decrease in the distances separating features on the chart; (2) a loss of visual clarity due to overcrowding; and (3) a shift of visual importance from the specific to the general.[1] In order to depict the important aspects of the Earth's surface at a more reasonable scale, features must be reduced in size and some detail of features must be omitted. Also, entire features might have to be eliminated, enlarged, combined, and/or displaced to fit within the graphic constraints of a typical chart. To this end, the cartographer must apply a series of manipulations to the chart data in order to depict the important information at the reduced chart scale. These manipulations of the chart data are commonly referenced under the collective topic of ***Cartographic Generalization***. The generalization processes are important to both manual and digital cartography.

The establishment of rigid guidelines for generalization has heretofore been a cartographic enigma. This has been evident for a number of years in manual cartography, and is characteristically shown by the inability of cartographers to merely define a ubiquitous definition of generalization. Regardless of the apparent disparity in the definition of the term, cartographic generalization will be defined here as the selection and simplified representation of detail appropriate to the scale and/or purpose of the chart.[2]

2.1.1 The Generalization Process

Before a cartographer can begin the data modifications required by the generalization process, information must first be selected for portrayal; the information being consistent with the purpose of the chart. Generalization, therefore, can be seen to operate in two stages: (1) *selection* of the data to be portrayed; and (2) *generalization* of this

data with regard to the scale and format of the final product. Thus, selection is a necessary pre-processing step to generalization.

Selection of information is merely a dichotomous query; either the information is required or it is not. No modification of the information is required in the selection stage, and can thus be done without regard for chart format or scale. Selection, then, can be thought of as a sifting process; one which segregates out the information required for a particular product or to support a particular production requirement. For example, a digital MC&G data base might contain cartographic information to support the production of a variety of products, with data resolution possibly equaling the largest scale product in the data base, and data available for many geographic areas, not all of which may be required for a particular job. A sifting function can determine whether to include or exclude chart information for a particular product or group of products, with a specific geographic area in mind.

Subsequent to the selection process, the generalization of each set of data that constitutes the selected information can then be accomplished. These manipulations are commonly combined into four categories:[3]

Simplification: The determination of the important characteristics of the data, the retention and possible exaggeration of these important characteristics, and the elimination of unwanted detail.

Classification: The ordering or scaling and grouping of data.

Symbolization: The graphic coding of the scaled and/or grouped essential characteristics, comparative significances, and relative positions.

Induction: The application in cartography of the logical process of inference.

Selection, along with the above four processes together combine to form the "Generalization Process."

2.1.2 Automating the Generalization Process

Manual generalization of chart features often collectively includes the separate processes of selection and simplification all under the label of cartographic license. In a mere sweep of a pen, a cartographer will select a feature to be represented on his chart and draft his "generalized representation" of the feature. The cartographer's generalized

representation will inherently retain those characteristics that he deemed necessary to delineate the feature with or without optionally exaggerating those characteristics, while also deleting the characteristics of the feature not required for his intent and purposes.

The not-so-recent trend in cartography to a computer-assisted environment must address the same topics of chart generalization, yet each must be treated independently; this independence necessitated by the finite logic of a computer. The computer has made cartography faster, more consistent, and more accurate for many cartographic endeavors (such as projection transformations), yet computer-assisted chart generalization has lagged far behind.

Research in automating the generalization methods for cartographic data has yielded a plethora of papers, theories, and computer algorithms, emanating from such disciplines as Geography, Computer Science, Mathematics, and Engineering. Some of these algorithms, however, have been designed with little or no cartographic basis, with cartographers neglecting to apply logical cartographic principles. For example, many "line simplification algorithms are frequently developed with little understanding of the quality of their output."[4]

An obvious question is then: how is the concept of cartographic generalization instituted into computer-assisted cartographic practice?[5] This question must obviously be based within the framework of whether generalization in the digital domain will be fully automated, semi-automated, or highly interactive. If we view the generalization process as it truly is—subjective, interactive, undocumented, idiosyncratic, and, yet still, holistic in its perception and execution—then we have run head-first into an undefinable problem. In turn, this means an unsolvable one. As such, the notion of completely replacing the human cartographer in the generalization process is a goal doomed to failure. The limits of existing computing technology cannot perceive the chart as a whole as does the man and, therefore, cannot assess the impact of the generalization of one feature on another feature.

If, however, we merely aim to aid the cartographer in the generalization process, we are addressing a much more realistic and achievable goal. Our efforts, then, should be directed in that path; that is, on a path towards providing the cartographer with intelligent tools, rather than trying to emulate his intuitive chart-making knowledge. This does not mean, however, that the generalization process needs to be entirely interactive; instead, a

semi-automated approach seems both reasonable and within the grasp of current design sophistication and computing capabilities. Many generalization problem areas can be addressed today in a fully- or, at worst, semi-automated modes of operation. Line simplification routines, for example, are nearing an overall level of maturation and understanding that we can begin to apply the techniques with some assurance of success. Other areas, such as the refinement of disjoint line clusters to support scale reduction may, indeed, be many years away from having practical algorithms developed.

In the following pages, we will be discussing the generalization process in the context of *Simplification*, *Combination*, *Refinement*, *Conversion*, *Displacement*, *Smoothing*, and *Compaction* of MC&G data.[6,7] Two types of operations can be identified in the *Simplification* of data stored as vector coordinate strings.[8] They are: (1) Point Simplification and (2) Feature Simplification. Point simplification operates on the principle of coordinate removal, replacement, or reposition to provide a vector coordinate string which represents the location of the original line. Feature simplification, on the other hand, is similar to a sifting process where entire features are omitted since their inclusion is not essential to retain the overall message and characteristics of the chart. *Combination* techniques will be reviewed as they relate to combining like features into new, yet similar features; for instance, the combination of two small lakes into a larger lake would fall within this category. Feature *Refinement* procedures will be reviewed as they relate to selecting a representative subset of features to depict at the reduced scale of the product. As an example, this would include the selection of a subset of piers on a coastline to depict the overall navigational characteristics of the region being mapped. Feature type *Conversion* deals with the modification of the geometric attributes of a feature to represent it in a new form at the reduced scale. An example here would be the collapse of an areal feature to a linear feature representation. *Displacement*, or conflict resolution, techniques are used to counteract the problems that arise in feature conflict detection. The interest here lies in the ability to offset feature locations to allow for the application of symbology. *Smoothing* operators would be applied to features to create a more aesthetically pleasing product without violating the spatial accuracy. And, finally, data *Compaction* will be reviewed as it applies to post-processing the vector feature data to reduce the digital storage requirements.[9]

2.2 *An Automated Generalization Model*

In order to replace some portion of the human generalization process with computer-assisted algorithmic assessment, we must first understand the generalization process in an *automated* sense. Once we have modeled this process as rules or guidelines to follow, we can begin to understand where and how computers can provide processing assistance to the nautical cartographer. One significant problem that cartographers have encountered over the past two and one-half decades involves the development of objective rules for automated generalization. This results from a very simple fact: cartographers have never—and perhaps will never—developed objective rules for the generalization of data in manual mode. The problem of such subjectivity is well-documented in the literature and is discussed at length in a recent publication.[10] In order to develop such objective rules, most probably decades of research into the cognitive aspects of generalization would be necessary. It might be possible, however, to, in part, bypass such detailed cognitive understanding with the development of comprehensive models of generalization. Ultimately, this would allow cartographers to bypass such studies and perhaps develop *new* control structures based entirely on digital methods. Before such an endeavor is made, a clearer understanding of the rules of generalization is required.

McMaster and Shea have postulated that the primary goal of generalization may be stated as follows, **"To maintain clarity with appropriate content at a given scale for a chosen map purpose and intended audience."**[11] This, of course, requires some elaboration.

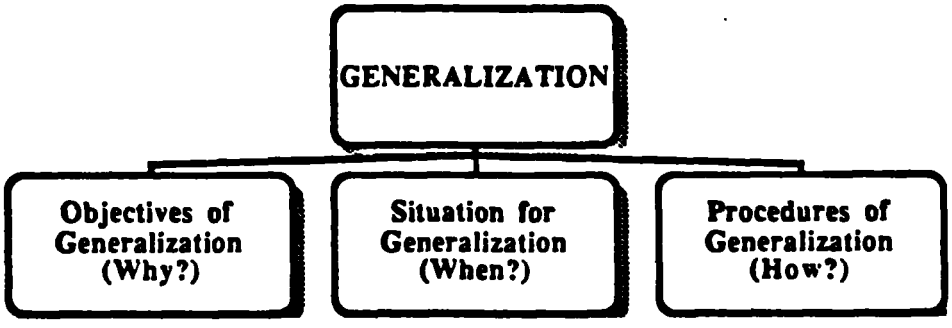
By **Clarity**, it is meant that the legibility or readability of the chart is maintained. It is not possible, under any circumstances, to reduce a chart scale and yet maintain the original level of detail. We can maintain such clarity by manipulating the mapped image using a variety of operators—omission, simplification, displacement, agglomeration, aggregation, collapse, conversion, and smoothing—that we ultimately wish to convert to computer algorithms.

The amount of detail retained after generalization is obviously a direct function of a change to a **Given Scale**. Unfortunately, at this time we still do not know the mathematical relationships between features retained and scale change. The extent to which details can be retained might be specified with formulas similar to the uniform density law

derived from Töpfer and Pillewizer to relate the number of features n_f on a chart at scale M_f to be retained from a source chart at scale M_a having n_a features.[12] Yet their formula $n_f = n_a\sqrt{M_a/M_f}$ does not directly address local feature density, which relates more directly to chart clutter than does the aggregate number of features. Although this introduces the problem of feature density, it assumes that feature types do not change as a result of the scale change operation.[13] Cartographers know this to be untrue. In addition to the need to decrease the absolute numbers and/or density of features at a reduced scale, many of the representations of features may alter due to the scale reduction. Area features will collapse to lines and points, lines collapse to points, multiple point features aggregate to areas, multiple area features agglomerate into new areas, and linear and point distributions are refined to depict representative patterns. Features need to be displaced and/or exaggerated to successfully communicate the intended message within the graphic constraints of the chart.

A chart has a **Chosen Purpose** and **Intended Audience**, which is fundamental to the design. Starting with an initial digital data base at a given scale, the cartographer may wish to reduce the scale of the product. However, one intended audience may have an application for the product which is entirely different than that of another audience. The generalization of most features for these two intended purposes would be accomplished with entirely different goals in mind.

In an automated environment, the generalization process must be guided by three thoughts: (1) **Why** we generalize; (2) **When** we generalize; and (3) **How** we generalize.

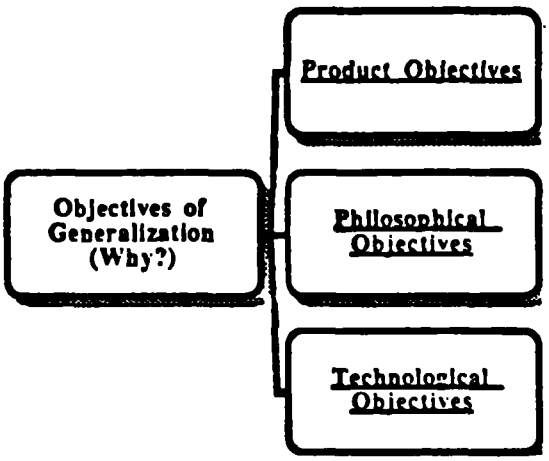


The illustration above provides such a model based within the context of generalization requirements, general cartographic principles and practices in manual production, and knowledge of existing research and development in the *automated cartography* discipline.

The following discussions will elaborate on each of the three areas: **Why**, **When**, and **How**.

2.2.1 Objectives of Generalization (Why to Generalize)

In order to fulfill the requirements of the primary goal of generalization stated above, cartographers must carefully consider a series of objectives subsumed within this major goal. These **Objectives of Generalization** can be thought of as **Why** we generalize.



The **Objectives of Generalization** can be viewed from three vantage points as illustrated in the figure on the left. These objectives can be based upon: (1) very specific requirements of the product, or group of products, being developed; (2) general cartographic principles (that is, the intuitive or philosophical objectives of cartography); and (3) governed by the requirements levied by existing computing technology.

2.2.1.1 Product Objectives

From a product perspective, a cartographer generalizes to meet the specific requirements of a product or group of products. To accomplish that goal, three objectives of each product, as illustrated to the right, should be sought towards this purpose.

Product Objectives
Clarity Scale Map Purpose and Intended Audience

A somewhat obvious, yet often overlooked, objective of generalization is to satisfy the specific requirements of the product(s) being produced. Cartographic feature data to be exploited by the processes required for the generation of products is organized in MC&G

data bases. The structure of these data bases are commonly designed to maintain both feature attributes as well as the topological relationships between the features. This design ensures that feature relationships may be efficiently ascertained and still available for multiple-product generation. MC&G data bases contain a wealth of information for many products and for many geographic areas. Since many products do not require the complete set of this stored information, methods to ignore unneeded data for a particular product are necessary. This process of culling the data is referred to as data segregation, or, simply, sifting.

A data base of MC&G information can contain cartographic data to support a variety of products wherein the resolution of the data may vary from geographic area to area. Let's look at a typical example of generalization as it applies to nautical chart products. NOS' Nautical Charting Division (NCD) is primarily a manual production environment. This environment is currently in a state of transition to an all-digital mode of operation and will include the establishment of uniform procedures relating to the collection, screening, evaluation, editing, symbolization, retrieval, and exchange of digital source and production data.[14] As part of this transition, the NCD has considered maintaining a single digital MC&G data base to support all nautical chart and marine related publications requirements. As a result, data for a specific geographic region—for example, Flushing, Long Island, New York—may be collected for, and support, many scales of a particular nautical product as illustrated by the chart below:

Chart #	Chart Name	Scale
12339	East River-Tallman Island to Queensboro Bridge	1:10,000
12366	L.I. Snd. and E. River -Hempstead Harbor to Tallman Island	1:20,000
12364	L.I. Snd.-N. Haven Hbr. Ent. & Pt. Jefferson to Throgs Neck	1:40,000
12363	L.I. Snd.-Western Part	1:80,000
12300	Approaches to NY, Nantucket Shoals to Five Fathom Bank	1:400,000
13006	West Quoddy Head to NY	1:675,000
13003	Cape Sable to Cape Hatteras	1:1,200,000

Although each product listed above is a nautical chart, scale dictates the specific information required to support individual requirements of the harbor and coastal classifications of each. On the other hand, this same data base might contain information to support the production of a completely different class of products such as:

Chart #	Chart Name	Scale
#76	Long Island West Costal Topographic/Bathymetric Chart	1:100,000
NK 18-12	N.Y. Outer Continental Shield or Bathymetric Chart	1:250,000

Here, not only is the nautical cartographer concerned with the selection of the information required for a given product scale, he also must be cognizant of the variety of products the data base will be used to prepare; that is, a knowledge of the product purpose.

During the generation of a product from a digital MC&G data base, part of the compilation process involves designating the products to be produced and the geographic area of the world covered by those products. In its most elementary form, computer-assisted feature selection depends upon feature codes that incorporate a ranking of features providing priorities so that a sufficient number of less important, or non-required, types of features can be suppressed to avoid cluttering the chart. A list of features must, then, be retrieved to support the generation of those products.

Let's look at an example. A unique NOS production requirement states that four products be prepared by the production staff. The four products to be derived for a given geographic area are a large-scale Harbor Chart, a smaller-scale Coast Chart, an even smaller-scale Sailing Chart, and a large-scale Topographic/Bathymetric Chart. When comparing these products individually, one fact is readily apparent: not all features for one product are needed in the other product. Furthermore, features stored in the MC&G data base might have feature codes which are entirely different from the individual product feature codes. These data base codes may be related to product codes by an association file. Using an association file, sifting may be accomplished by creating a catalog of required features that satisfy the area requirements and match the required product feature codes. The catalog consists of the feature ID number of the features that are candidates for at least one of the products based on area and feature codes. If the required feature catalog is to be used, then the full MC&G data base must be available to support feature retrievals. Obviously, this is expensive in terms of data storage requirements and data retrieval times. An alternative is to use this catalog to specify the features to populate a subset data base for a particular production requirement. Since this subset data base is smaller, storage and retrieval time requirements will correspondingly decrease.

The operation of creating a subset may be implemented by either reducing a copy of the full MC&G data base or by building the subset data base from a null data base. In the

first implementation, unwanted features are deleted from the copy of the MC&G central data base. In the second implementation, required features are added to the initially empty subset data base. In either implementation, basic operations to add, delete, modify, merge, and breakup features (point, line and area) and topology (node, edge and face) are necessary. These basic operations are available in most Data Base Management Systems (DBMSs) and are required to maintain the MC&G data base. Intrinsic to the basic operations is the validation of the MC&G data base. This software should guarantee that the structure of the MC&G data base is self-consistent after each operation.

A second product-specific objective arises when the data have been segregated out for a particular product but must now undergo a scale reduction. Here, many products differ in their "rules" of generalization. Take, for example, two products, both at a 1:50,000-scale, but differing in their purpose and intended audience—a topographic product and a bathymetric product. Each of these may contain common features located near the shoreline; an example here could be the depiction of gas wells. If the scale is reduced to 1:250,000, the resultant generalizations of the gas well features can be quite different. In one instance, they may be aggregated and re-represented as an area feature with a label of "numerous gas wells." Alternatively, these wells may be dropped entirely from the other product at the reduced scale. Even though both products require the same features, their handling of scale change, and its influence on generalization, are quite dissimilar.

Although both processes may be intuitively obvious, they are nonetheless important steps in the generalization of cartographic information. The necessity for this data segregation process is reduced substantially if multiple, product-specific digital cartographic data bases are maintained. For instance, maintaining separate data bases for General, Sailing, Coastal, Harbor, International, Small Craft, Canoe, Recreation, and Special Nautical Charts, along with others to support Coastal Topographic/Bathymetric, Outer Continental Shelf, and Smaller-scale regional Bathymetric Charts, will allow quicker and easier sifting processes based merely upon geographic areas, without the added requirement for determining product type and purpose. Unfortunately, this also requires the duplication of many features and their corresponding attributes between like products and scales. The storage overhead required for these multiple data bases may then outweigh the benefits. The ability to support scale change—a radical generalization—within a single

product must consider those goals, or specific requirements, of the particular product. And, also handle any impacts of this scale reduction on the processes selected for accomplishing the generalization.

2.2.1.2 Philosophical or Theoretical Objectives

From a philosophical or theoretical perspective, a cartographer generalizes to counteract the undesirable consequences of scale reduction. To accomplish that goal, six objectives should be sought towards this purpose.[15] These objectives, as illustrated to the right, are discussed below.

Philosophical Objectives
Reducing Complexity Retaining Spatial Accuracy Retaining Statistical Accuracy Maintaining Aesthetic Quality Maintaining a Logical Hierarchy Consistently Applying Generalization Rules

Reducing Complexity. For this purpose, complexity will be defined as the number and variety of phenomenon per unit area. Such complexity results, of course, as the scale is reduced and features become cramped together. This perhaps is the trickiest problem in all of generalization, for it requires that many of the operators discussed previously be applied to the problem either iteratively or simultaneously. This may be demonstrated with a simple example using the Thousand Island region of upper New York State.

As the name implies, between the U.S. and Canadian shoreline exists thousands of islands varying in size, importance, and many other geomorphic and political factors. Along the shoreline are numerous villages and cities also varying in size. Threaded through these islands is a critical shipping channel. Crossing the St. Lawrence River are numerous bridges. Imagine now taking a digital representation of this area collected/portrayed at a nominal scale of 1:25,000 and reducing the data to a 1:250,000-scale representation. Many of the islands now collapse together—they must be either agglomerated, omitted, or displaced. Other islands have now *collided* with the shipping channel and must be omitted or displaced. The river shoreline must be simplified at the reduced scale, yet the shipping channel must retain most of its geometric—planimetric—fidelity. Where bridges exist, however, the shoreline may not be moved through simplification. Additionally, at this reduced scale, most feature boundaries will have to be smoothed in order to eliminate the sharp angularity imposed by digitization. Some bridges may need to be deleted. The transportation networks associated with these bridge locations will need to be altered.

Geographic landmarks that serve as aids to navigation need to be maintained, but this must keep in mind the size, type, and/or location of the landmarks' associated features.

This, of course, describes only a few of the spatial decisions which either the nautical cartographer, or in a digital mode, the algorithm, would have to make in producing a "generalized" chart. The decisions as to the order in which the operators are applied is just as crucial as the selection of the operators and algorithms themselves. Significantly varying generalized versions of the original chart will be obtained through different ordering of the operators. Researchers are, unfortunately, many years away from determining either the correct—let alone the optimum—ordering of such operators or the parameters to use. The concept of complexity, then, and the methods that are necessary in order to reduce complexity and yet fulfill the other goals of generalization, is the single most ambiguous area of generalization. Certainly, though, one fact is clear: without substantial psychological and cognitive testing, decisions regarding these issues will be difficult.

Retaining Spatial Accuracy. The goal of retaining spatial accuracy is much more clear and measurable than the previous goal. Spatial accuracy can be directly related to displacement between the original and generalized features. Here, displacement refers to the planimetric difference and is measured with vector or areal displacement measures. These are well documented in the literature. Research goals for cartographers over the next few years should include the evaluation of algorithms based on their displacement quality.

Retaining Statistical Accuracy. The retention of spatial accuracy deals with what might be called geographical data—the points, lines, and areas that build the data base. One must also consider the accompanying statistical or attribute data associated with these spatial data. This goal is, for the most part, purely mathematical in nature and involves both statistical analysis and classification. It is also a more major concern with thematic mapping that with *general* or *topographic* mapping. The overall objective here is to minimize the alteration of statistical attribution of the features.

Maintaining Aesthetic Quality. The aesthetic quality of a chart—manual or digital—depends upon a multitude of factors, including the figure–ground relationships, overall balance, and layout. Design is a highly subjective and ultimately biased process that cartographers are just beginning to understand. Although specific rules for good design are impossible to formulate, general guidelines are now being proposed. It must be recognized, however, that imposing absolute precepts upon cartographic design is synonymous with asking Picaso for rules to be used in painting. As is commonly stated in cartography; the art must be retained. Those involved in digital methods who feel that ultimately the entire process can be automated are doomed to failure. There are many exciting possibilities, however, for greatly improving the design of digital products. Some

of these include: the proper implementation of smoothing algorithms and the antialiasing of raster images.

One excellent example of maintaining the aesthetic quality of the chart is related to the reduction of scale such that the size and extent of features is beyond the visual acuity of the eye. The reduction of objects in the chart space cannot be indefinite and must terminate at the limits of acuity of the human eye. Studies have shown that this relates to approximately 0.02mm at a distance of 30cm from the eye; any features smaller than 0.2mm cannot usually be distinguished. It is, however, not realistic to reduce the objects on the chart to the barely perceptible because visual importance is diminished, and the effects of lighting and printing methods on the communicative efficiency of the products can be impaired. Scale reduction, that is, generalization, must weigh the relationship between what is/not shown with the overall complexity of the resulting product.

Maintaining a Logical Hierarchy. This may be considered a subset of the above goal. A clear mapped image must contain an ordering of the mapped features. Large cities must be more prominent that smaller cities; interstate highways more prominent than country roads. This seems relatively straightforward for a single class of features—roads—but becomes more difficult when dealing with the entire mapped image. Areal, linear, and point features must be considered in a holistic sense. The major determinant of the graphic hierarchy amongst the features is the chart purpose.

Consistently Applying Generalization Rules. Many cartographers now working in the area of digital cartography truly—and somewhat naively—believe that automation of the process will enable the removal of subjectivity. Nothing could be farther from the truth. The problems here are clearly illustrated with Monmonier's work on raster-mode generalization.[16] There is probably more variation in the selection and application of a generalization algorithm in digital mode than in two manually drafted versions. In order to obtain consistent, unbiased, generalization, cartographers will have to determine three things: (1) exactly, which algorithm to use; (2) the order in which to apply these algorithms; and (3) the input parameters to obtain a given result at a given scale. Given that this information might be available (and must be obtained through additional research), a more unbiased and less subjective result is possible.

In summary, few of the above philosophical or theoretical objectives can be met with current computing technology. Of those goals that can be met, maintaining the spatial and statistical accuracies seem within grasp since these are essentially just computing the mathematical relationships between feature locations and/or attribution. The other objectives, however, can only be accomplished partially because of the holistic perceptual processing that is required to make adequate assessments of goal achievement. Since

perception is a highly individualistic response to a visual stimulus, the cognitive image of the chart will be idiosyncratic.[17] As a result, even though cartographers may be presented with the same generalization requirements, the individual generalizations will be both particular to, and characteristic of, each cartographer. Thus, trying to attain goals that are based within this perceptual realm, such as maintaining the aesthetic quality of the product, may yet be years away from being achieved.

2.2.1.3 Technological Objectives

From a technological perspective, generalization is extremely important in the digital domain. Here, a cartographer generalizes to balance the relationship between sampling interval of data, data complexity, storage requirements, and CPU-needs. To accomplish that goal, the two objectives illustrated to the right should be sought.

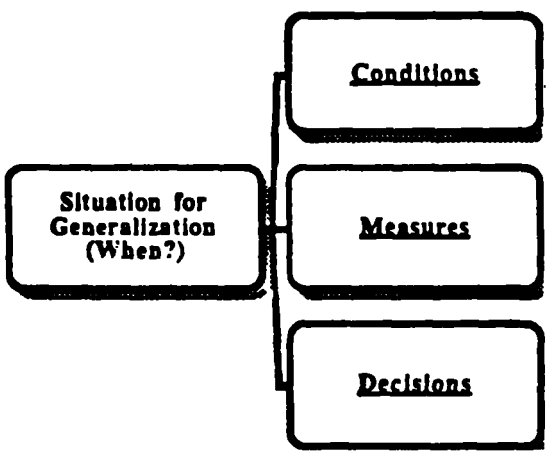
Technological Objectives
Cost Effectiveness of Algorithms Minimum Storage/Memory Requirements

Cost Effectiveness of Algorithms. In digital mode, a significant goal is to reduce the information in a cost efficient manner. This, of course, is relatively easy to ascertain. For instance, in the generalization of line data, we are reasonably sure that the Douglas corridor simplification algorithm is the best mathematically, but one of the worst in terms of computation requirements. Thus, for precise mapping requirements—shorelines, for example—the Douglas routine is perhaps most appropriate. For different requirements, though, other, more computationally efficient routines are probably sufficient. Consider, for example, the fact that another linear simplification algorithm, the Lang tolerancing algorithm, is nearly as accurate as the Douglas algorithm but at a substantially reduced processing cost. Thus, the overall goal here is to balance the cost of a computer algorithm against its "accuracy" of generalization.

Minimum Storage/Memory Requirements. A similar consideration of generalization in digital mode is to reduce the data storage requirements down as much as possible. This may be determined by three factors: (1) the final scale reduction of the chart; (2) the output resolution of the graphic device; and (3) the purpose of the chart. A detailed description of the relationship between these is provided in a forthcoming publication.[18] This can be achieved in two ways: (1) reducing the coordinate numbers required to represent the spatial entities; and (2) reducing the data structure to more compact, less storage-intensive, forms. Efforts here should be directed towards maintaining maximum information with a minimum of storage/memory size requirements.

In summary, both of the above objectives can be met with current computing technology. Much of the current research in cartographic generalization has been formulated with these two goals in mind. In fact, the cartographic literature is pervaded with many exciting research efforts that have specifically addressed these areas. Much research is still required, however, to coordinate these activities within the perceptual arena of cartography. A wiz-bang algorithm that performs some function of generalization "in a flash" and reduces the data set to a exiguous portion of the original data set, is of no use to the cartographer if the end product is perceptually unrecognizable from the original data or does not satisfy the purpose of the chart. Therefore, the algorithm selection and efficiency assessment must be based, in part, within the perceptual realm of cartographic communication.

2.2.2 Situation for Generalization (When to Generalize)



In the above discussions, we have considered the **Why** component of generalization for formulating objectives of the generalization process. Next, we will consider the situations in which generalization would be required. Ideally, these arise due to the success or failure of the chart product to meet the stated goals. Here, we will view the **When** of generalization from the three vantage points illustrated to the left.

The Conditions under which generalization procedures would be invoked would be based upon the Measures by which that determination was made, and the Decisions or control of the generalization techniques that will be employed to effect the change.

2.2.2.1 Conditions

Five conditions, that will occur under scale reduction, may be used to determine necessary generalization.[19] The conditions illustrated below each identify a problem area in generalization and are described below.

Conditions

- Congestion
- Coalescence
- Conflict
- Complication
- Inconsistency

Congestion. Congestion refers to the problem where too many features have been positioned in the same geographical space; that is, feature density is too high.

Coalescence. Coalescence is a condition where features will touch as a result of either of two factors: (1) the separating distance is smaller than the resolution of the output device; or (2) the features will touch as a result of the symbolization process.

Conflict. Conflict is a situation in which the feature is in conflict with the background.

Complication. Complication relates to an ambiguity in performance of techniques and order; that is, the results of the generalization are dependent on the iteration of techniques chosen to perform the scale reduction.

Inconsistency. Inconsistency refers to a set of generalization decisions applied non-uniformly across a given chart. Here, there would be a bias in the generalization between the topographic elements.

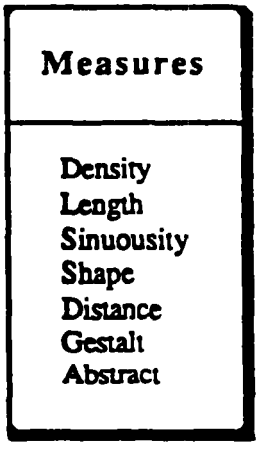
It is the above conditions which require that some type of generalization process occur to counteract, or eliminate, the undesirable consequences of scale change. Unfortunately, these conditions are highly subjective in nature and, at best, are difficult to quantify. Consider, for example, the problem of congestion. Simply stated, this refers to a condition where the density of features is greater than the available space on the graphic. One might question how this determination is made. Is it made in the absence or presence of the symbology? Is symbology's influence on perceived density—that is, the percent blackness covered by the symbology—the real factor that requires evaluation? What is the unit area that is used in the density calculation? Is this unit area dynamic or fixed? As one can see, even a supposedly simple term, density, is a relative enigma. The other remaining conditions—coalescence, conflict, complication, inconsistency—also can be highly subjective in their assessments. How, then, can one assess the state of the conditions if the quantification of those conditions is ill-defined?

It appears as though such conditions as expressed above may be detected by applying a series of mensuration techniques to the original and/or generalized chart to determine a conditional state. Unfortunately, these techniques may indeed be quite complicated and inconsistent between various products or even within a single product. To

eliminate these differences, therefore, the assessment of conditions must be based entirely from within a non-product viewpoint. That is, to view the chart as a graphic entity in its most elemental form—points, lines, and areas—and to judge the conditions based upon an analysis of those entities.[20] This can be achieved by providing simple tools that operate on these geometric configurations and can be combined in some logical fashion to achieve the intended analysis.[21] This is accomplished through the evaluation of conditional Measures.

2.2.2.2 Measures

Conditional measures can be assessed by examining some very basic geometric properties of the inter- and intra-feature relationships. Some of these assessments are evaluated in an singular feature sense, others between two independent features, while still others are computed in a multi-feature sense. These measures are summarized below.



Density calculations, as shown above, are evaluated by using multi-features. Length and Sinuosity calculations, on the other hand, operate on singular features and might be appropriate for determining conditions requiring generalization. An example, here, could be the calculation of stream network lengths, or overall complexity of the network (based on, say, average angular change per inch) to select an appropriate and representative depiction of a distribution at a reduced scale. Shape calculations are also useful in the determination of whether an area feature can be represented at its new scale. Conditional measures may also be compartmentalized into Distance calculations between the basic geometric forms: points, lines, and areas. Distances between each of these forms can be assessed by examining the appropriate shortest perpendicular distance (SPD) or shortest euclidean distance (SED) between each form. In the case of two geometric points, only three different distance calculations exist: (1) point-to-point; (2) point buffer-to-point buffer; and (3) point-to-point buffer.[22] These determinations can indicate if any generalization problems exist if, for instance under scale reduction, the line buffer and areal

buffer conflict. In addition to the geometric measures, other classes of measures can be computed. This includes **Gestalt** measures,[23] which indicate *perceptual* characteristics of the feature distributions, and **Abstract** measures, which reveal more *conceptual* evaluations of the spatial distributions. Below, a list of possible measures are tabulated. Although this list is by no means complete, it does provide a starting point from which to evaluate conditions within the chart which do require, or might require, generalization.

DENSITY MEASURES (Point, Line, Area)

number of point, line, or area features per unit area
average density of point, line, or area features
number and location of cluster nuclei of point, line, or area features

LENGTH MEASURES (Line, Area)

Total number of coordinates in line feature or area feature boundary
Total length of line feature or area feature boundary
Average number of coordinates per inch on line feature or area boundary
Standard deviation of coordinates per inch on line feature or area boundary

SINUOSITY MEASURES (Line, Area)

Total angular change of line feature or area boundary
Average angular change per inch on line feature or area boundary
Average angular change per angle on line feature or area boundary
Sum of positive or negative angles on line feature or area boundary
Total number of positive or negative angles on line feature or area
Total number of positive or negative runs on line feature or area boundary
Total number of runs on line feature or area boundary
Mean length of runs on line feature or area boundary

SHAPE MEASURES (Point, Line, Area)

Area of point, line, or area features (unsymbolized/symbolized)
Perimeter of point, line, or area features (unsymbolized/symbolized)
Centroid of point, line, or area features (unsymbolized/symbolized)
X and Y Variances of area features (unsymbolized/symbolized)
Covariance of X and Y of area features (unsymbolized/symbolized)
Standard Deviation of X and Y of area features (unsymbolized/symbolized)

DISTANCE MEASURES (Point, Line, Area)

Shortest Euclidean Distance (SED)

point—point	point—point buffer
point—line centroid	point—area centroid
line—line	line—line buffer
line—line centroid	line buffer—line buffer
line buffer—line centroid	line centroid—area buffer
line centroid—area centroid	line centroid—area edge
area buffer—area buffer	area buffer—area centroid
area buffer—area edge	area centroid—area centroid
area centroid—area edge	area edge—area edge

Shortest Perpendicular Distance (SPD)

point—line	point—line buffer
point—area buffer	point—area edge
point buffer—point buffer	point buffer—area buffer
point buffer—area centroid	point buffer—area edge
line—area centroid	line—area edge
line—area buffer	line buffer—area buffer
line buffer—area centroid	line buffer—area edge
line centroid—line centroid	

GESTALT MEASURES (Point, Line, Area)

Closure
Continuation
Proximity
Common Fate
Figure Ground

ABSTRACT MEASURES (Point, Line, Area)

Homogeneity
Symmetry
Repetition
Recurrence
Neighborliness
Complexity

Each of the above classes of measures can be determined in a digital domain. Their interaction, however, is not as clearly understood. Exactly which of these conditions must exist before a generalization action is taken depends on scale, purpose of the chart, and so on. In the end, it appears as though many of the prototype algorithms may first be developed and then tested and fit into the overall framework of generalization. The exact guidelines on how to apply the measures designed above can not be determined without precise knowledge of the algorithms.

2.2.2.3 Decisions

In order for the cartographer to obtain unbiased generalizations, three things need to be determined: (1) which algorithm to use; (2) the order in which to apply these algorithms; and (3) the input parameters to obtain a given result at a given scale. Thus, the decision process includes the factors on the right.

Decisions
Procedure Control Algorithm Selection

Obviously, an important constituent of the decision-making process is the availability and sophistication of the algorithms. Actually, algorithms is a rather overused,

and somewhat misused, term. Instead, the algorithms should more appropriately be called **Controlled Procedures** that requires access to **Algorithms**. Thus, the generalization process is accomplished through a variety of procedures—each attacking specific problems—which employ a variety of algorithms. In the case of line simplification, for example, the simplification procedure would access algorithms such as Lang, Douglas, Roberge, etc. Concomitantly, there may be permutations, combinations, and iterations of procedures, each employing permutations, combinations, and iterations of algorithms. The algorithms may, in turn, be controlled by multiple, maybe even interacting, parameters.

2.2.2.3.1 *Procedure Control*

The control of generalization procedures is probably the most difficult process in the entire concept of automating generalization. The control decisions must be based upon: (1) the importance of the individual features (this is, of course, related to the product purpose and intended audience); (2) the complexity of feature relationships both in an inter- and intra-feature sense; (3) the presence and resulting influence of chart clutter on the communicative efficiency of the product; (4) the need to vary generalization amount, type, or order on different features; and (5) the availability and robustness of generalization processes and computer algorithms.

The necessity for sequential data processing requires the establishment of a certain sequence of the generalization process in order to avoid repetitions of processes and frequent corrections. The sequence is determined by the effects which result in lack of space or, alternatively, excess of space and locational changes of features caused by the generalization processes. On the basis of mutual interdependencies resulting from such generalization efforts of the individual processes automatically carried out, a sequence of generalization processes for the ANCS II is proposed below:

1. Independent and Dependent Generalization Requirement Evaluation
    a. Selection of point, line, and area features
    b. Identification of regions not to be generalized
    c. Identification of regions to be generalized
    d. Identification of features not to be generalized
    e. Identification of features to be generalized independently
    f. Identification of features to be generalized by pairwise dependence
    g. Identification of features to be generalized by multi-dependence
    h. Evaluation of conditions for independent point, line, and area feature generalization

i. Evaluation of conditions for pairwise point/line, point/area, and line/area feature generalization
j. Evaluation of conditions for multiple point, line, and area feature generalization

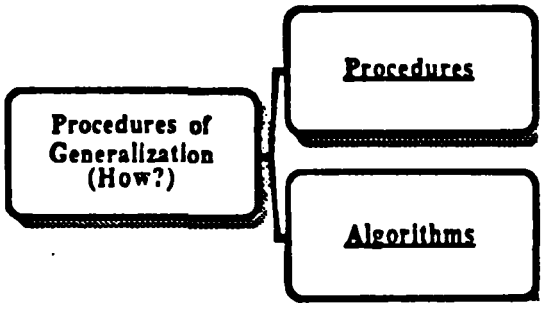
2. Point Aggregation
   a. Point Features to Area Features
3. Simplification (low-pass filter)
   a. Area Feature outlines
   b. Line Features
4. Feature Collapse
   a. Area Features to Line Features
   b. Area Features to Point Features
5. Distribution Refinement
   a. Area Features
6. Area Agglomeration
   a. Area Features to Area Features
7. Feature Collapse
   c. Line Features to Point Features
8. Distribution Refinement
   a. Point Features
   c. Line Features (disjoint)
   d. Line Features (connected)
9. Simplification
   a. Area Feature outlines
   b. Line Features
10. Smoothing
   a. Area Feature outlines
   b. Line Features
11. Compaction
   a. Area Feature outlines
   b. Line Features

2.2.2.3.2 *Algorithm Selection*

The selection of algorithms to support the generalization process must be based upon a variety of factors, not the least of which is proof of concept. The relative obscurity of generalization algorithms, coupled with a limited understanding of the generalization process, removes the selection process from merely conducting a cost-benefit analyses. There just are not algorithms to choose from. This means that many of the concepts need to be prototyped, tested, and evaluated during the design and development of the ANCS II. The evaluation process is usually the one that gets ignored or, at best, is only given a cursory review. Algorithms should be selected based upon cognitive studies, mathematical evaluation, and design/implementation trade-offs.[24] Once a candidate set of algorithms are available, they should be assessed in terms of their applicability to specific products.

Finally, each individual product may require different algorithms depending on feature type, scale, and/or purpose of the chart.

2.2.3 Procedures of Generalization (How to Generalize)



As a final third of the automated generalization model, we must consider the component of generalization that actually performs the processes of scale reduction. This How of generalization must be based within those areas of generalization techniques that have either arisen out of the emulation of the manual cartographer, or based solely on more mathematical efforts.

In the generalization process, we have determined that five basic categories of procedures exist to effect the required MC&G data changes to support the production requirements. These procedural categories are listed to the right.

Procedures
Line Simplification Feature Type Conversion or Refinement Feature Displacement Feature Smoothing Data Compaction

For many of these procedural areas, reviews of algorithms are included. As was stressed above, however, the algorithms (and procedures themselves) are affected by the factors listed on the right. The order of application, frequency of application, and limits of the algorithms must also be considered in the automated generalization process.

Algorithms
Permutations Combinations Iterations Parameters

2.2.3.1 Line Simplification

MC&G data bases created as a result of the feature selection process will contain only those features necessary to support the required products, to be presented at a required scale, with the minimum data storage requirements and data retrieval times. Even so, the digitization processes used to collect this information employs a variety of scales and/or resolutions of input media. This, in turn, means that superfluous data exists for the individual feature representations. In the digital domain, this means added execution times during processing, increased plotting times, and excessive data storage requirements. Some form of feature *simplification* can reduce the number of coordinate points required for feature representation. One of the more common uses of point simplification algorithms is their application to linear data sets for coordinate removal. These algorithms are commonly referred to as linear simplification, or merely, simplification routines. Simplification algorithms operate on the principle of point selection or point rejection.

Chart data that has been captured by electronic sampling devices must undergo a variety of transformations before it should be used as a digital representation of chart features. Data gathered by a sampling device such as a manual digitizer samples x,y coordinate pairs in discrete locations, established by the resolution of the input device. These discrete locations can be tied together by vectors to create a digitized line. Common digitizer resolutions result in recording a surplus of coordinate data for the representation of lines. In fact, although human discernability of coordinate differences is only on the order of about 0.02 inches, it is not uncommon to find that most digitizing systems capture coordinates at resolutions far beyond that (such as 0.001 inches). In addition, psychological and physiological errors are induced in the digitization process which create induced detail in the lines. Also, glitches are produced from electrical impulses in the sampling device and mechanical impulses in the operator's hand.

Ideally, a digitized representation of a linear chart feature should be accurate in its representation of the feature (shape, location, and character), yet also efficient in terms of retaining the least number of delimiting coordinate data points in storage. This profligate density of coordinates captured in the digitization stage should be reduced by selecting a subset of the original coordinate pairs, while retaining those points considered to be most representative of the line.[25] Glitches should be removed. And, finally, the line should be

smoothed to produce a line with a more aesthetically pleasing caricature.[26] Simplification algorithms will select the characteristic, or shape-describing, points to retain, or will reject the redundant point considered to be unnecessary to display the line's character.[27] Inevitably, though, simplification algorithms produce a reduction in the number of derived data points, which are unchanged in their x,y coordinate positions. Some practical considerations in the elimination of redundant or superfluous data gathered in the digitization stage includes reduced plotting time, increased line "crispness" due to higher plotting speeds, reduced storage, less problems in attaining plotter resolution due to scale change, and quicker vector to raster conversion. McMaster cites that five major types of linear simplification algorithms can be found in the literature.[28] They are: (1) Independent Point Routines; (2) Local Processing Routines; (3) Unconstrained Extended Local Processing Routines; (4) Constrained Local Processing Routines; and (5) Global Routines. Examples of each are discussed on the following pages.[29]

2.2.3.1.1 *Independent Point Routines*

Independent Point Routines are those in which no mathematical relationships between neighboring coordinate pairs are assessed.

N^{th} Point

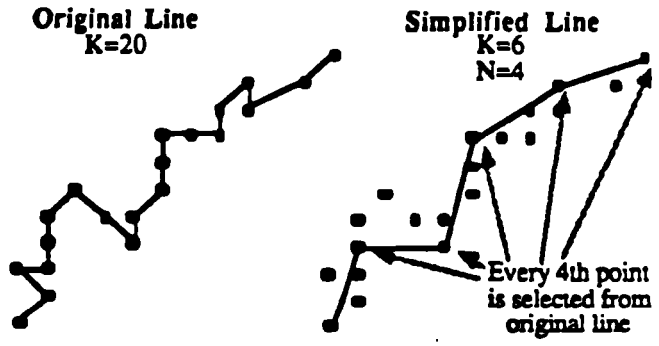
REFERENCE:

Tobler, Waldo R. (1964).

ALGORITHM DESCRIPTION:

After generating a random integer N, ranging from 1 to K (where K is the number of points in the data set), the algorithm reads the input data file sequentially and retains only every N^{th} coordinate pair. Larger values of N obviously yield greater simplifications.

GRAPHIC EXAMPLE:



In the figure above, a data set consisting of 20 coordinate pairs is being simplified based upon a selection of every 4^{th} coordinate pair. In addition, the first and last points have been retained. Note how the original line has been reduced to a simplification containing only 6 points, an extreme savings in storage, yet the character of the line has changed considerably.

ADVANTAGE:

Computationally one of the fastest line simplification routines and, therefore, one of the cheapest to run in terms of time and money. Simple to program and very straightforward in its operation.

DISADVANTAGE:

Straight lines are over represented, and critical points are not necessarily retained. It does not take distance between points into account. Therefore, the algorithm totally ignores the fact that some points are spaced closely while others may be far apart. As a result, the shape of the line derived from this simplification routine will depend entirely on what point in the feature is considered as the starting point—because it is here from which the counting of the N^{th} Point will be initiated. Modifications commonly applied to the algorithm includes retention of the first and last coordinate pairs (as in this example), regardless of the N^{th} position.

Random

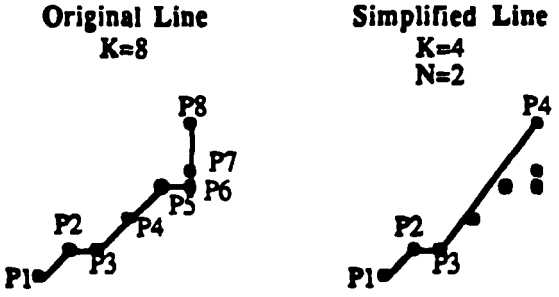
REFERENCE:

Robinson, Arthur H., et al. (1978).

ALGORITHM DESCRIPTION:

After generating an operator-selected number of random integers N, the algorithm reads the input data file sequentially and retains the first coordinate pair, and then only the coordinate pairs that fall on those random positions within the file. Finally, the last coordinate pair is saved, regardless of whether it fell on one of the random positions.

GRAPHIC EXAMPLE:



In the figure above, the original line contains 8 coordinate pairs. Assume that the cartographer has specified that only 2 random coordinates (N) and the end points of the original line are retained. A random generation of two numbers between 2 and 7 (since 1 and 8 are already retained) yields 2, 3. The simplification has been reduced to only 4 coordinates.

ADVANTAGE:

Computationally one of the fastest line simplification routines and, therefore, one of the cheapest to run in terms of time and money. Simple to program and very straightforward in its operation. The start and end points of a line will remain intact.

DISADVANTAGE:

This procedure has no cartographic basis and, therefore, important characteristics of the line may be lost in the simplification. It does not take distance between points into account. Therefore, the algorithm totally ignores the fact that some points are spaced closely while others may be far apart. As a result, the shape of the line derived from this simplification routine will depend entirely on what points are retained. This will change each time because of the random selection of coordinate pairs.

2.2.3.1.2 *Local Processing Routines*

Local Processing Routines are those in which the characteristics of immediate neighboring coordinate pairs are used.

Line Width

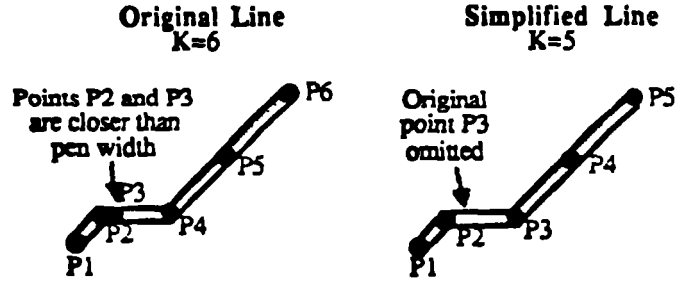
REFERENCE:

Tobler, Waldo R. (1965).

ALGORITHM DESCRIPTION:

The algorithm reads the input data file and retains the first coordinate pair as an anchor point. Then, reading sequentially through the coordinate file, the Euclidean distance is calculated between the anchor point and the next point. If the Euclidean distance between the two points is closer together than the width of the plotted line, the second pair is rejected. The algorithm then iteratively reads successive coordinate pairs until it finds one that falls outside of the distance determined by the line width. That point is now retained, it becomes the new anchor point, and the search for the next coordinate pair continues. Finally, the last coordinate pair is saved, regardless of whether it fell on outside of the selected tolerance.

GRAPHIC EXAMPLE:



In the figure above, a sample line contains 6 points. Points P2 and P3 are closer together than the width of the line and, as a result, only the first point encountered of these two points (P2) will be retained. The original line is reduced to 5 points.

ADVANTAGE:

Easy to program and fast computationally. Retains end points.

DISADVANTAGE:

Algorithm bears no cartographic logic and is subject to the same disadvantages as were identified in the N^{th} point algorithm.

Euclidean Distance

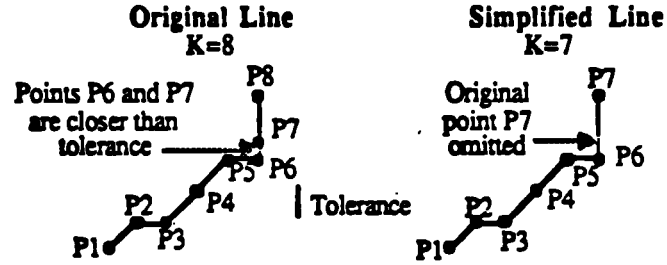
REFERENCE:

McMaster, Robert B. (1983a).

ALGORITHM DESCRIPTION:

The algorithm reads the input data file and retains the first coordinate pair as an anchor point. Then, reading sequentially through the coordinate file, the Euclidean distance is calculated between the anchor point and the next point. If the Euclidean distance between the two points is less than a pre-selected tolerance, the second pair is rejected. The algorithm then iteratively reads successive coordinate pairs until it finds one that falls outside of the preselected distance. That point is now retained, it becomes the new anchor point and the search for the next coordinate pair continues. Finally, the last coordinate pair is saved, regardless of whether it fell on outside of the selected tolerance.

GRAPHIC EXAMPLE:



In the figure above, a sample line contains 8 points. Points P6 and P7 are closer together than the Euclidean distance specified. As a result, only the first point encountered of these two point pairs (P6) will be retained. The original line is now reduced to 7 points.

ADVANTAGE:

Easy to program and fast computationally. Retains end points.

DISADVANTAGE:

Algorithm bears no cartographic logic.

United States Geological Survey - A

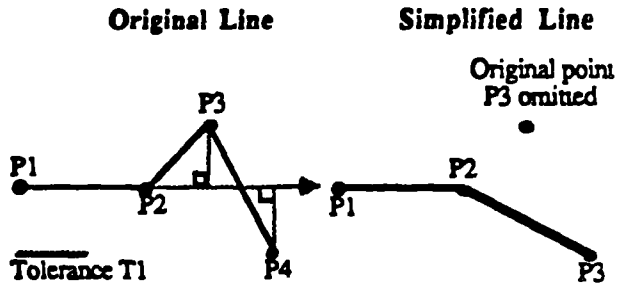
REFERENCE:

United States Geological Survey (personal communication).

ALGORITHM DESCRIPTION:

Corridor algorithms operate by specifying a distance either side of the data line, as a corridor, for point rejection or retention. A vector joining points P1 and P2 is extended as a projected straight line. The perpendicular distance from this extended line to P3 is calculated. Points are accepted if this distance is greater than a pre-tolerance T1. If the perpendicular distance is less than the tolerance, the point in question is rejected.

GRAPHIC EXAMPLE:



In the figure above, point P3 would be eliminated since it is within a threshold tolerance T1 from an imaginary vector drawn between P1 and P2. After P3 is rejected, the imaginary vector is again drawn through P1 and P2. The perpendicular distance of P4 from that line is greater than the tolerance, and as such, will be retained.

ADVANTAGE:

Fairly fast. Easy to program.

DISADVANTAGE:

Does not take distance between points into consideration.

United States Geological Survey - B

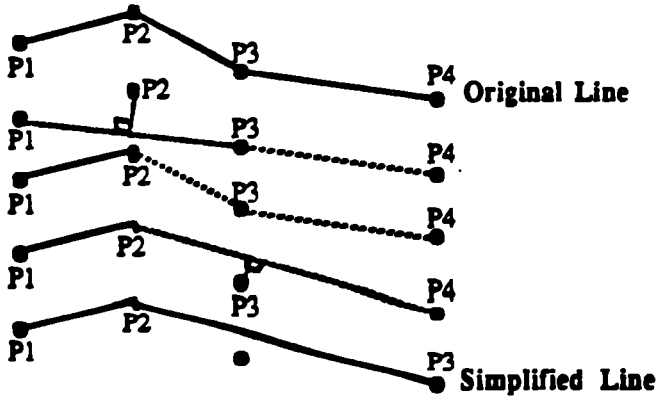
REFERENCE:

United States Geological Survey (personal communication).

ALGORITHM DESCRIPTION:

A second version of this algorithm operates on triads of points. A vector joining the first and third points in the triad is projected as a straight line. The perpendicular distance from this projected line to the middle point is calculated. This middle point is accepted only if this distance is greater than a pre-selected tolerance. If it is accepted, it becomes the new anchor point, the third point now becomes point 2, the next successive point (point 4) is read in as the new point 3, and the process repeats. If the distance is less than then specified corridor, the middle point is omitted, the third point now becomes the middle, and the next successive point (point 4) is read in as the new point 3, and the process repeats.

GRAPHIC EXAMPLE:



In the figure above, the distance of P2, from a computed vector drawn between P1 and P3, is greater than the selected tolerance. As such, P2 will be retained. Point P3 is now tested for its perpendicular distance from the computed vector drawn between points P2 and P4. This distance is less than the selected tolerance and, as such, P3 is rejected.

ADVANTAGE:

Fairly fast. Easy to program.

DISADVANTAGE:

Does not take distance between points into consideration.

Angle of Change

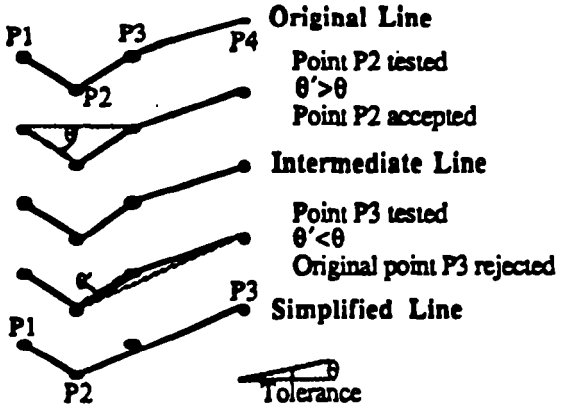
REFERENCE:

Tobler, Waldo R. (1964).[30]

ALGORITHM DESCRIPTION:

Algorithm compares the angular change between vectors connecting the first and second coordinate pairs, and the first and third coordinate pairs. A tolerance angle is selected by the cartographer and points are rejected if their angle is greater than the tolerance angle specified. Processing is repeated from this point to the next two points.

GRAPHIC EXAMPLE:



In the figure above, a sample line consists of 4 points.[31] Angle θ', the angular change between points P2, P1, and P3, is greater than the tolerance angle θ. As such, P2 is retained. The angular change θ' between the next successive three points (P3, P2, and P4), is less than the specified tolerance of θ and, as such, point P3 will be rejected. Simplified line is then reduced to three coordinates.

ADVANTAGE:

Good theoretical basis for point selection. Retains end points.

DISADVANTAGE:

Complexity of program dependent on the computer's resident function. Pure angle algorithms take no account of distance between coordinate pairs; this may have a detrimental effect on the curvature of the resultant line in that large gentle curves may be eliminated and replaced by straight line sections.

Distance and Angle

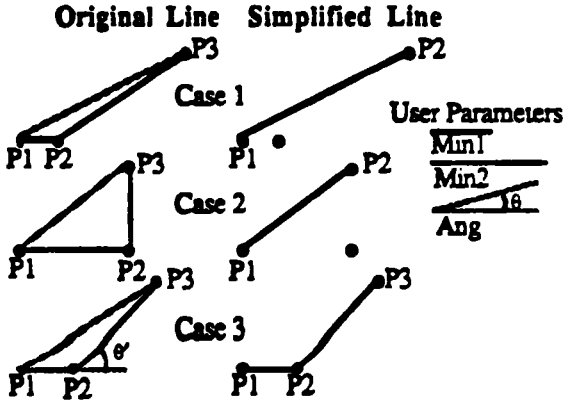
REFERENCE:

Jenks, George F. (1980) personal communication.

ALGORITHM DESCRIPTION:

Three parameters are specified: (1) a minimum allowable distance between points 1 and 2 (MIN1); (2) a minimum allowable distance between points 1 and 3 (MIN2); and (3) the maximum allowable tolerance angle between a line drawn through points 1 and 2, and 2 and 3 (ANG). If the distance from point 1 to point 2 is less than MIN1, or the distance from point 1 to point 3 is less than MIN2, point 2 will be rejected. If both distances are larger than the minimum allowable distances, the angular is calculated. If the angle is larger than the tolerance angle ANG, the point is accepted; if it is smaller, the point is rejected. Thus, points will be rejected if they are within the minimum distances or if their angle is less than the specified angle.

GRAPHIC EXAMPLE:



In the figure above there are three examples of this algorithm. In the top example, point P2 will be eliminated because it is closer to P1 than the tolerance distance MIN1. In the second case, P2 will be eliminated since the distance between P1 and P3 is less than MIN2. In the final example, the distance from P1 to P2 is greater than MIN1 and the distance from P1 to P3 is greater than distance from MIN2. The angular change from the two vectors connecting the three points (θ') is greater than ANG. Therefore, point P2 will be retained.

ADVANTAGE:

Combines the processing speed of a sequential algorithm and, using the sound basis of angular selection algorithms, this algorithm also incorporates a distance measurement.

DISADVANTAGE:

High computational time relative to other sequential algorithms, yet lower than corridor algorithms; a good alternative to both.

Field of View

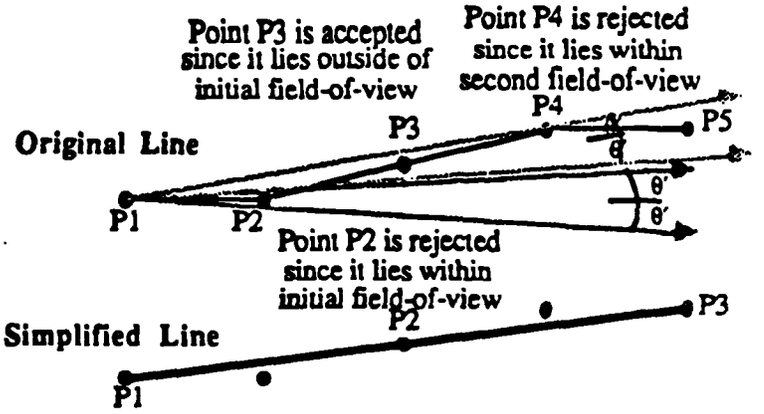
REFERENCE:

Jenks, George F. (1980) personal communication.

ALGORITHM DESCRIPTION:

Algorithm evaluates each point according to whether or not it lies within a "field of view" from the previous point. This field of view angle is pre-selected, and sets the orientation of two lines either side of the vector joining two coordinate pairs. If the point being sampled lies within this field of view, it is rejected and processing continues, computing angles and lines to the next point. If the point is outside of this field of view, it is accepted. When a point is accepted, it becomes the new anchor, or base, point and the procedure repeats again.

GRAPHIC EXAMPLE:



In the figure above, a angular threshold of θ' has been specified on each side of the Field of View direction. From P1 to P2, the Field of View does not include the next successive point P3, and as such, P3 will be retained. Constructing a field of view now from P1 to P3 includes P4 and it will therefore be eliminated. Because P5 is the end point it will be retained (as was P1). The simplification now consists of 3 coordinate pairs.

ADVANTAGE:

All angle algorithms present a good theoretical basis for point selection. Retains end points.

DISADVANTAGE:

Somewhat complicated to program; the complexity of the program dependent on the resident functions available on the particle host computer. Pure angle algorithms take no account of distance between coordinate pairs—this may have a detrimental effect on the curvature of the resultant line in that large gentle curves may be eliminated and replaced by straight line sections.

2.2.3.1.3 *Unconstrained Extended Local Processing Routines*

Unconstrained Extended Local Processing Routines are those in which the characteristics of neighboring coordinate pairs are used and in which the search region is expanded to sections of the line and not limited to the immediate neighbors.

Reumann-Witkam

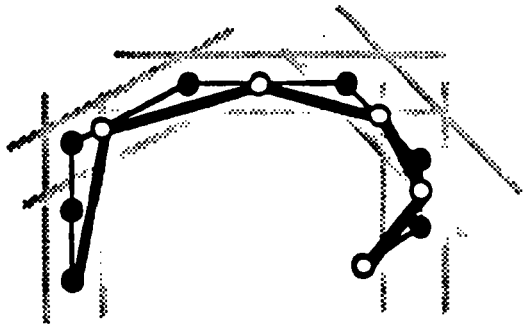
REFERENCE:

Reumann, K. and A.P.M. Witkam (1974).

ALGORITHM DESCRIPTION:

This algorithm searches the immediate neighboring coordinate pairs and evaluates sections of the line by using two parallel lines to define a search region. After calculating the initial slope of the search region, the line is processes sequentially until one of the edges of the search corridor intersects the line.

GRAPHIC EXAMPLE:



In the figure above, a search region is recalculated based on the last intersection point. A point is inserted where the curve crosses out of the band or the last input point contained within the band is selected to be retained. The algorithm continues until the last point and its tangent are used. Here, this figure shows all the calculated tolerance bands for the original line. The final simplified line is depicted as a dark band and the retained coordinates as circles.

ADVANTAGE:

Very fast.

DISADVANTAGE:

Does not operate well under severe simplifications. Requires calculation of the tangent to a digitized curve. Choice of the direction tangent is not well calculated where a straight line is drawn between the last two (2) points and used to derive the direction.

Roberge

REFERENCE:

Roberge, J. (1985).

ALGORITHM DESCRIPTION:

This algorithm is a modification of the Reumann-Witkam. His enhanced strip algorithm provides: (1) a more rigorous definition of the critical line; (2) a test for vertical critical lines; (3) a check for inflection points; and (4) an extension factor which enables extended critical lines to be constructed.

GRAPHIC EXAMPLE:

None provided.

ADVANTAGE:

Extension factor proves advantageous for reducing curves with slow rates of curvature.

DISADVANTAGE:

Does not operate well under severe simplifications. Requires calculation of the tangent to a digitized curve.

2.2.3.1.4 *Constrained Local Processing Routines*

Constrained Local Processing Routines are those in which the characteristics of neighboring coordinate pairs are used and in which the search region is expanded, yet restricted, to some sections of the line.

Lang Tolerancing

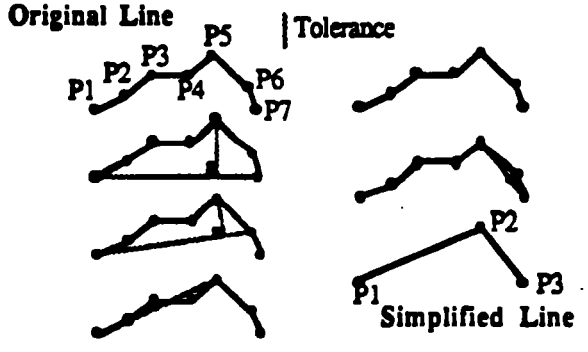
REFERENCE:

Lang, T. (1969).[32]

ALGORITHM DESCRIPTION:

A tolerance is specified as the nominated drawing accuracy for the plotter. Points are removed if they lie within the tolerance distance from a line drawn between an initial point and the end point being considered. If the specified tolerance is exceeded, the plotted line is drawn to the next end point assuming that these points satisfy the tolerance check. A modification to the algorithm differs in that only points that were distant from from the last plotted point by greater than the specified distance D were used for plotting.

GRAPHIC EXAMPLE:



In the figure above, a line connecting endpoints P1 to P7 is projected. If the perpendicular distance from this line to intervening points exceeds a specified tolerance, the line is repositioned from points P1 to P6 and the distances are again checked. Here, the distance from P5 to the vector drawn between points P1 and P7 is greater than the tolerance. The vector is now drawn between P1 and P6 and point P5 still lies outside of the specified tolerance. As the vector is moved to between P1 and P5, all the distances are within the tolerance and are deleted (points P2 through P4). The imaginary vector is now drawn between the new beginning point P5 and the end point P7, and a test of all intervening points is again computed. Since P6 is within the specified tolerance it is omitted and the simplified line now contains only three points of the original line.

ADVANTAGE:

The second algorithm is much faster than the first, but still relatively slow.

DISADVANTAGE:

Slow. Algorithm produces acceptable results on relatively smooth curves but does not detect the best representation points on sharp curves.

Johannsen Tolerancing

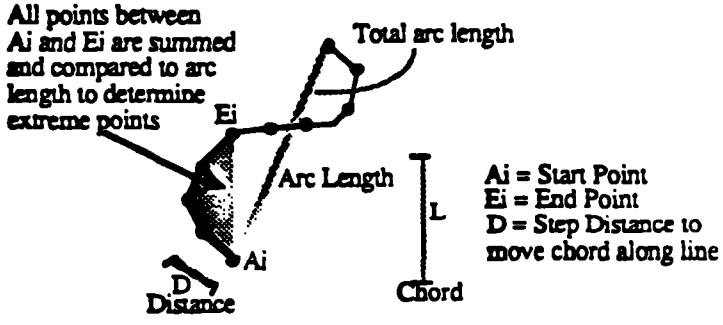
REFERENCE:

Johannsen, T. (1973).

ALGORITHM DESCRIPTION:

Of all the data points in a line, points are extracted which represent the maximum curvature after low frequency curves are suppressed. This algorithm processes by moving a chord of given length (l) along the line, by steps of a known distance (D). For each chord position (Di) all of the points between the start (Ai) and the end of the chord (Ei) are evaluated and summed. This is calculated as a function over the arc length to derive the extreme points. These are only extracted if they are maximum in relation to a set number of neighboring coordinate pairs.

GRAPHIC EXAMPLE:



In the figure above, an example of the Johannsen tolerancing algorithm is presented. A chord of length L is extended from the initial point Ai to some point, Ei, along the arc. All intermediate point between Ai and Ei are summed and compared to the total arc length to evaluate the total distance between the arc and the chord. If extreme points exist, the ration of distance to total arc length will be high and those points will be eliminated. Small distances compared to arc length imply relatively minor peturbations in the line and no points are removed.

ADVANTAGE:

None.

DISADVANTAGE:

CPU-intensive.

Opheim

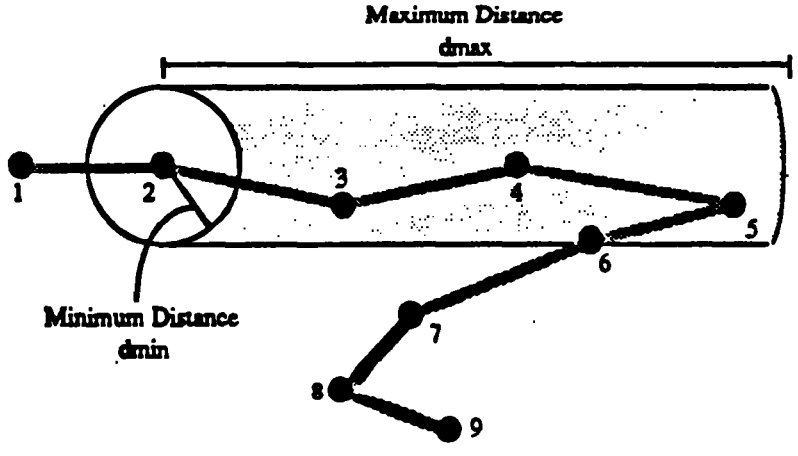
REFERENCE:

Opheim, H. (1982).

ALGORITHM DESCRIPTION:

The search region is restricted or constrained by a minimum and maximum distance check, much like the Distance/Angle algorithm. After the initial search region is set which is similar to the Reumann-Witkam, any point within the minimum distance are eliminated. However, as soon as the line "escapes" from the search region on any side, including distance maximum, a new search corridor is established and the last point within the region is saved.

GRAPHIC EXAMPLE:



In the figure above, points 3, 4, 5, and 6 are eliminated since they fall within the search region tolerance band. Point 7 becomes part of the simplified line since it is the last point to fall inside the search region.

ADVANTAGE:

None. Not well analyzed yet.

DISADVANTAGE:

If line makes any sudden bends within the maximum distance search region the critical point of the bend will be eliminated.

2.2.3.1.5 *Global Processing Routines*

Global Processing Routines are those in which the entire line is examined in a holistic sense and not processed sequentially as in all the other classifications.

Douglas Corridor

REFERENCE:

Douglas, David H., and Thomas K. Peucker (1973).[33]

ALGORITHM DESCRIPTION:

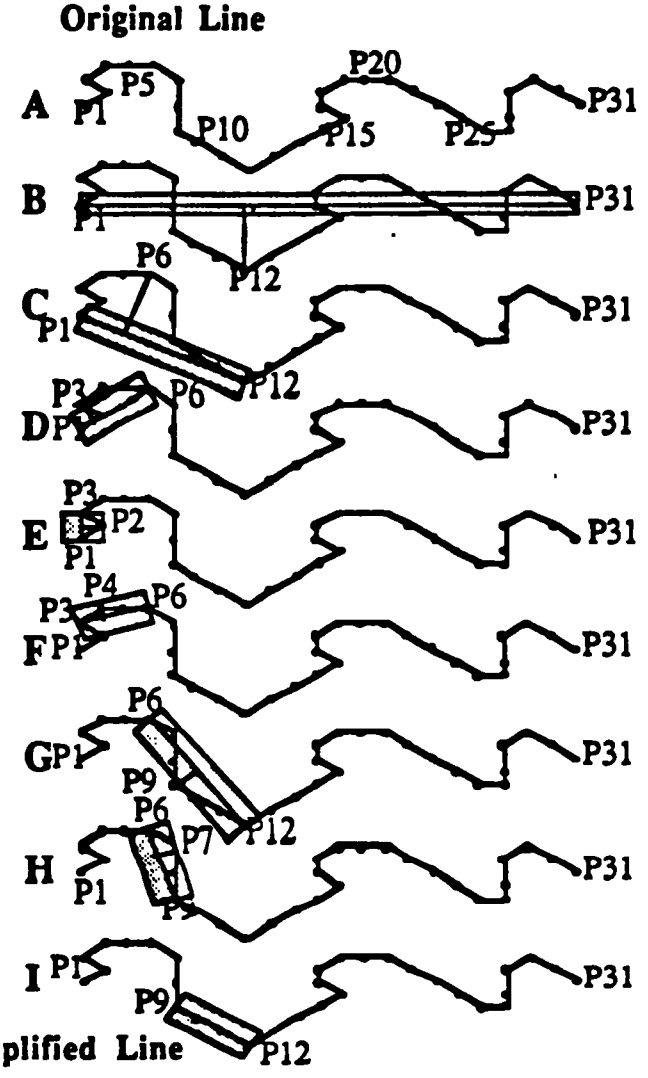
This algorithm operates globally on a data set, processing an entire line at a time..The algorithm begins by defining the first point in the line as an anchor, and the last point as a floating point position. These two points are now connected by a straight line segment. Intervening points along the line are now examined to determine the one with the greatest perpendicular distance between it and the straight line defined by the anchor and floater points. If this maximum perpendicular distance is less than a maximum tolerable distance, the straight segment is considered suitable to represent the entire line. In cases where the distance condition is not met, the point lying furthest away becomes the new floating point, and the process continues until all points in each segment lie within a given tolerance.

In the figure below, a diagrammatic example of the Douglas Algorithm operation is provided. This description is provided since the Douglas algorithm is probably the most *cartographically-sound* linear simplification algorithm. As such, it is an excellent choice for implementation.

The following discussion details the operation of the Douglas algorithm on a sample line containing 31 coordinates (reference the figure below, A).

- Tolerance Band is selected by a cartographer, shown as a shaded area on the figure below, B. This tolerance band is the computed distance in length either side of the line constructed between the current anchor point and the floater point. In this case, the anchor is P1, the floater is P31.
- Push P1 on anchor stack.
- Push P31 on floater stack.
- Calculate perpendicular distance to all intermediary points from the imaginary vector drawn between anchor P1 and floater P31 (reference the figure below, B).
- Sort for maximum perpendicular distance. In this case, P12.
- Push P12 on floater stack.
- Calculate perpendicular distance to all intermediary points from the imaginary vector drawn between anchor P1 and floater P12 (reference the figure below, C).
- Sort for maximum perpendicular distance. In this case, P6.
- Compare maximum perpendicular distance to tolerance. Distance is greater than tolerance.
- Push P6 on floater stack.
- Calculate perpendicular distance to all intermediary points from the imaginary vector drawn between anchor P1 and floater P6 (reference the figure below, D).
- Sort for maximum perpendicular distance. In this case, P3.
- Compare maximum perpendicular distance to tolerance. Distance is greater than tolerance.
- Push P3 on floater stack.
- Calculate perpendicular distance to all intermediary points from the imaginary vector drawn between anchor P1 and floater P3 (reference the figure below, E).
- Sort for maximum perpendicular distance. In this case, P2.

GRAPHIC EXAMPLE:



- Compare maximum perpendicular distance to tolerance. Distance is greater than tolerance.
- Compare maximum perpendicular distance to tolerance. Distance is less than tolerance.
- Pop last point, P3, off floater stack.
- Push P3 onto Anchor stack.
- Calculate perpendicular distance to all intermediary points from the imaginary vector drawn between new anchor P3 and floater P6 (reference the figure above, F).
- Sort for maximum perpendicular distance. In this case, P4.
- Compare maximum perpendicular distance to tolerance. Distance is less than tolerance.
- Pop last point, P6, off floater stack.
- Push P6 onto Anchor stack.
- Calculate perpendicular distance to all intermediary points from the imaginary vector drawn between new anchor P6 and floater P12 (reference the figure above, G).
- Sort for maximum perpendicular distance. In this case, P9.
- Compare maximum perpendicular distance to tolerance. Distance is greater than tolerance.
- Push P9 on floater stack.
- Calculate perpendicular distance to all intermediary points from the imaginary vector drawn between anchor P6 and floater P9 (reference the figure above, H).
- Sort for maximum perpendicular distance. In this case, P7.
- Compare maximum perpendicular distance to tolerance. Distance is less than tolerance.
- Pop last point, P9, off floater stack.
- Push P9 onto Anchor stack.

- Calculate perpendicular distance to all intermediary points from the imaginary vector drawn between new anchor P9 and floater P12 (reference the figure above, I).
- Sort for maximum perpendicular distance. In this case, P10 or P11 (equal).
- Compare maximum perpendicular distance to tolerance. Distance is less than tolerance.
- Pop last point, P12, off floater stack.
- Push P12 onto Anchor stack.
- Process is now complete for all coordinates lying between points P1 and P12. To continue, the above sequence of events would be followed for all points lying between P12 and P31.

ADVANTAGE:

Perhaps the most highly respected linear simplification algorithm developed; it is based upon sound cartographic principles. The Douglas algorithm has proven to be both mathematically and perceptually significant.[34] In fact, it has been shown that the algorithm most closely replicates the human generalization process in terms of retaining critical points on the line.[35] These critical points can be related to the physical characteristics of a line, or those related to man-made or perceived positions of importance.

DISADVANTAGE:

Requires a large amount of processing time since it continually works through a line several times until all portions of it have been examined. However, this increased processing time can be considered a trade-off since the Douglas algorithm achieves such ideal simplified representations of the line.

2.2.3.2 Feature Type Conversion/Refinement

As a chart is a reduced representation of the Earth's surface, and as all other phenomena are shown in relation to this, the scale of the resultant product largely determines the amount of information which can be shown. As a result, the generalization of cartographic features to support scale reduction must obviously change the way features look in order to fit them within the constraints of the graphic.

The information that is contained within the graphic has two components—location and meaning—and generalization affects both. As the amount of space available for portraying the cartographic information decreases with decreasing scale, less locational information can be given about features, both individually and collectively. As part of this requirement for feature type conversion, or feature refinement process, the graphic depiction of the features changes to suit the scale-specific needs. The transformation processes here include: (1) Aggregation/Agglomeration; (2) Combination; (3) Collapse; and (4) Distribution/Network Refinement. Each of these are discussed below.

Point Aggregation. There are many instances when the number or density of like point features within a region prohibits each from being portrayed and symbolized within the graphic. Still, their importance, both from a landmark and military significance, require that they be portrayed. To accomplish that goal, the point features must be aggregated into a higher order class feature—areas. One of the best examples of this requirement is the aggregation of a feature like gas wells into an areal outline that is labelled as "numerous gas wells."

Area Agglomeration. This type of generalization is extremely important when portraying features such as hydrography. Through combination of individual features into a larger element, it is often possible to retain the general characteristics of an area despite the scale reduction. For example, a region containing numerous small lakes—each too small to be depicted separately—could with a judicious combination of the areas retain, very closely, the original chart characteristic. One of the limiting factors of this process is that there is no fixed rule for the degree of detail to be shown at various scales; the end-user must dictate what is of most value.

Line Feature Combination. If the scale change is substantial, it may be impossible to preserve the character of individual liner features. As such, these linear features must be combined. As an example, both divided highways and railroad yards are normally represented as two adjacent lines, with a separating distance between them. Upon scale reduction, these two lines require that they be combined into one positioned approximately halfway between the original two.

Area and Line Feature Collapse. As scale is reduced, many features shown as areas must eventually be symbolized as points or lines. The decomposition of line and area features to point features, or area features to line feature, can also be thought of as a generalization process. Settlements, airports, rivers, lakes, islands, and buildings, often portrayed as area features on large scale charts, can become point or line features at smaller scales. Areal tolerances guide this transformation.

Point Distribution Refinement. In many cases, areas that are encountered containing similar point features that are either too numerous or too small to show to scale, no attempt should be made to show all the points. Instead, a representative pattern of the symbols should be added to cover the area, augmented by an appropriate explanatory note. Here, the point features should be thinned out; however, the general pattern of the features must be maintained with the features shown in their correct locations. This typification process retains the general characteristics of the points at a reduced complexity.

Line Network Refinement. In many cases, areas that are encountered containing similar line features that are either too numerous, too small, or too close together to show to scale, no attempt should be made to show all the lines. Instead, a representative pattern of the symbols should be added to cover the area, augmented by an appropriate explanatory note. Here, the line features should be thinned out; however, the general pattern of the features must be maintained with the features shown in their correct locations. This typification process retains the general characteristics of the lines at a reduced complexity.

Area Polygon Refinement. In many cases, areas that are encountered containing similar area features that are either too numerous, too small, or too close together to show to scale, no attempt should be made to show all the areas. Instead, a representative pattern of the symbols should be added to cover the area, augmented by an appropriate explanatory note. Here, the area features should be thinned out; however, the general pattern of the features must be maintained with the features shown in their correct locations. This typification process retains the general characteristics of the areas at a reduced complexity.

2.2.3.2.1 Algorithms for Conversion/Refinement

On the following pages, a sample of the types of algorithms that could be used for various aspects of the feature type conversion/refinement procedures are discussed. This section does not provide designs for new algorithms; instead, it merely reports on existing algorithms within the cartographic literature. It will be immediately obvious that existing research in these areas of generalization lag far behind that of line simplification discussed previously. It should be noted that the follwoing algorithms can be applied to different features than they were originally intended for. For instance, a derivative of the Drainage Network Refinement procedure could be used to support a typification process to select a representative pattern of piers, piers in ruins, or some other disjoint network feature.

Building Combination

REFERENCE:

Lichtner, W. (1978).

ALGORITHM DESCRIPTION:

The generalization of buildings can be seen to operate in stages: (1) selecting and emphasizing the small buildings; (2) simplifying the building outlines; and (3) combining the buildings. Lichtner suggests that buildings be combined by ascertaining if the distance between the buildings falls below a minimum distance and, if so, then attaching the smaller building to the larger and combining the individual outlines to create a new larger building symbol. The basic principle in the combination of buildings is to move the smaller to the larger buildings. To achieve this goal, all buildings within some "generalization area" are sorted according to increasing size. Beginning with the smallest building, the immediately adjacent buildings are found, and the smallest gap to an adjacent building is found and compared to the gap limit imposed by the algorithm. If it falls within that limit, the smaller building is moved to the larger one. If several buildings are situated too close to the original one, they are moved against the adjacent building with the smallest gap distance.

Settlement Selection by Population/Location

REFERENCE:

Peucker, T. (1973).

ALGORITHM DESCRIPTION:

Many algorithms exist to automatically select settlements from a data base to be shown on charts. Poiker's (then Peucker) algorithms operates by first drawing an imaginary circle around each town whose radius is inversely proportional to the population of that town. Thus, a small town has a large radius and a large town has a small radius. No other town may intrude into the area of this circle. Settlement density is controlled by an exponent a. The selection process begins with the largest settlement, adding to it the other settlements one by one, largest to smallest, which do not fall within the radii of any of the previously selected settlements. The formula used to compute the radii is:

$$\text{radius (I)} = \left(\frac{\text{reference city population}}{\text{population (I)}} \right)^n \cdot \text{reference radius.}$$

Settlement Selection by Nearest Neighbor

REFERENCE:

Peucker, T. (1973).

ALGORITHM DESCRIPTION:

The Nearest Neighbor Index (R) is normally used to estimate clustering or dispersion processes in a distribution to select an appropriate areal distribution of settlements. R is computed for the five largest settlements, and recomputed as settlements are added to the distribution in order of decreasing size. A decrease in R denotes increased cluttering; hence, a settlement is selected only if its introduction to the distribution increases or causes no change to R.

Uniform Density Law

REFERENCE:

Töpfer, F. and W. Pillewizer (1966).

ALGORITHM DESCRIPTION:

The extent to which details can be retained might be specified with formulas similar to the uniform density law derived from Töpfer and Pillewizer to relate the number of features n_f on a map at scale M_f to be retained from a source map at scale M_a having n_a features. Yet their formula $n_f = n_a\sqrt{M_a/M_f}$ does not directly address local feature density, which relates more directly to map clutter than does the aggregate number of features.

Drainage Network Refinement

REFERENCE:

Catlow, D.R. and D. Du (1984).

ALGORITHM DESCRIPTION:

Drainage networks, because of their interaction with many of the other geomorphological characteristics of the region being mapped, must retain their basic geographic characteristics at the reduced scale. Catlow and Du introduced a *Data Rationalisation*, *Stream Ordering*, and *River Generalization* to refine drainage networks. First, each river segment is joined into a topological data set to ensure continuity of the linework. Next, each river segment is divided at a point where neighboring items overlay or touch it. This now sets up the drainage network such that the number of river data items corresponds to the number of river segments, and each river confluence is defined by a data node. A stream ordering method—such as that proposed by Strahler (1952)—is then used to place a stream order code and a catchment area code on each data item. A data point is then inserted at the mouth of each drainage network, and the connecting river segments to these seed points are then identified. Stream orders are now calculated based upon the number of linking items.

Although selecting all stream order 1 streams is a simple method of distribution refinement, it does not produce an acceptable product because drainage networks are considerably reduced in length, while single river systems without headstream tributaries are automatically omitted. Thus, it is necessary to consider the more important of the stream order 1 rivers, whether they form simply a single-river system, or whether they area part of a larger drainage network. The generalization process is, then, best performed by selecting not only on the basis of stream order, but also on length of order 1 segments, islands on the basis of area, and lakes on a combination of area and their relation to rivers in the drainage network.

Polygon Refinement through Epsilon Filtering

REFERENCE:

Chrisman, Nicholas R. (1983).

ALGORITHM DESCRIPTION:

This algorithm, developed as part of the ODYSSEY system for geographic information processing at the Laboratory for Computer Graphics (Cambridge, MA), is similar to the work of Julian Perkal (1965). The program starts with a topologically structured file of polygonal boundary lines and uses a geometric

search strategy of divide-and-conquer to limit search requirements. Clusters form while examining line intersections in which a cluster groups all points that can be linked together by a chain of epsilon tolerances. Point selection to depict the area boundary is accomplished by selecting the point within epsilon of most other points. Once the points are selected, some lines are moved to become congruent. This allows double-line feature (rivers, inlets) to be converted to single-line features, and also supports the attachment of small islands near shores to become part of the mainland.

2.2.3.3 Feature Displacement

Feature displacement, or conflict resolution, techniques are used to counteract the problems that arise in feature conflict detection. The interest here lies in the ability to offset feature locations to allow for the application of symbology. The graphic limits of a chart make it necessary to move features from what would otherwise be their true locations. If every feature could realistically be represented at its true scale and location, this displacement would not be necessary. Unfortunately, however, feature boundaries are often an infinitesimal width; when that boundary is represented as a cartographic line, it has a finite width and thereby occupies a finite area on the chart surface. These conflicts need to be compensated for by shifting the features from their true locations, modifying the features, or deleting them entirely from the graphic.

In the following discussion, conflict detection and cartographic cost resolution are the processes required to automatically detect and resolve conflicts between symbolized topological entities in graphic products. Product specific rules, standard rules, special feature–to–feature rules, and general cartographic rules are utilized to define and determine what constitutes a conflict, as well as how to resolve a conflict. Conflict detection and resolution rules are used to: (1) determine the candidate conflicting feature types; (2) define the pairwise conflicts between features based on coincidence, overlap of symbols, or proximity of symbols; (3) provide further definition of complex conflicts involving structural relationships between objects in pairwise conflicts; and (4) for each conflict defined, provide the resolution strategies possible, and the cost of each resolution strategy. The cartographic cost of a resolution strategy is defined here as the degree of reduction in chart accuracy, information content, and quality as a result of affecting a specific resolution strategy.

2.2.3.3.1 *Conflict Detection*

Conflict conditions requiring detection and resolution include cases of: (1) Proximity; (2) Overlap; (3) Special Cases; (4) Coincidence; and (5) Exceptions.

> **Proximity.** Two topological entities are proximatrial if their separation at any point is < x mm, where x is product specific and variable with scale. Entities can be described as coalescing, too close to plot, etc.

Overlap. Two topological entities overlap if their associated symbols intersect and their centerlines do not. This case is also referred to as an overprint. Overlap cases may be acceptable in specific examples or unacceptable and requiring cartographic cost resolution.

Special. Includes special conflict conditions or geometric patterns such as parallel lines and sandwich effects.

Coincident. Two topological entities coincide if they share the same topology.

Exception. Under certain circumstances the listed conflict rule does not apply and the exception is invoked.

2.2.3.3.2 *Conflict Resolution*

The baseline for resolution of cartographic conflicts between symbolized topological entities can be comprised of a rule set and a hierarchical listing of symbolized features according to their value to the product and the end user. The rule set is defined as:

General Rules. These rules provide general guidance in the formulation of the chart product.

Product Specific Rules. Product specific rules are tailored for a particular product(s). Items of interest for the chart producer are offered here detailing any special treatment requirements for symbology, exceptions to standard rules, and the identification and guidance for treatment of required or critical information and features.

Standard Rules. Standard rules are those rules generated when no specific product oriented rules are available to resolve a conflict.

Special Feature–to–Feature Rules. These rules are invoked to determine if a conflict exists for a pair of features when no specific rule addressing the two features is available.

In order to select the best resolution strategy for implementing a rule a means is needed to assign a relative cost to a particular binary conflict resolution action. Candidate resolution strategies include displacement, deletion, symbol alteration, interruption, replacement or special symbolization. Each method of resolution can be associated with a set of cartographic costs for the features involved. In deletion, a feature symbol that would have appeared on the chart in the absence of a conflict is removed. This creates a reduction in the chart's information content. The degree of information content reduction is related to the deleted feature's importance. A feature displacement hierarchy which represents a view of features' relative importance may serve as an initial ranking. In feature deletion, certain

features, such as key landmarks, may never be deleted. For these features a deletion cost that is prohibitively high would be assigned. Conversely, certain feature's inclusion in the chart may be optional (that is, they may be removed if they are located in a congested area). These features could receive a deletion cost of zero to indicate no cost associated with deleting such a feature.

When the centerline of a feature (and therefore its symbology) is moved or displaced from ground truth, the accuracy of the chart is reduced. The degree of reduction in accuracy is a function of the overall amount of movement as well as the specific feature involved. Certain features, such as spot elevations, may never be moved. These features would therefore be assigned a very high cost of movement. Less important features when moved may have a lesser impact on the quality of the chart and would therefore be assigned a lower cost of movement. Cost of movement would consist of the weight of a features importance multiplied by the overall distance of displacement. For line features, the displacement distance would be the summation of the individual movement distances for each node.

The other resolution methods, including symbol change or alteration, interruption, and scaling have a less well defined impact on the quality of a chart. Used properly, these methods, since they retain the feature symbol in its proper location, may not have any negative impact and may be assigned a zero cost of resolution.

For each of the resolution strategies that affect cost a quantitative weight is generated to be applied to each feature included in the product. This weight, based on the feature hierarchy, the rules, and other information, will be a measure of a specified feature's relative impact on chart accuracy and quality when subjected to displacement, deletion, symbol change, etc. The following illustration contains three categories of factors related to the chart product, which individually could have a weight assigned to assist in determining the impact of the cost of conflict and its resolution to the overall accuracy of the product.

Cost of Resolution Strategy	Cost of Resolution Factors	Conflict Cost Requirements
Cost for Alteration	Feature Type	Horizontal Accuracy
Cost for Deletion	Distance	Vertical Accuracy
Cost for Interruption	Total Number of Displacements	
Cost for Movement	Method of Resolution	
Cost for Scaling	Total Product Features	
Cost for Rotation	Type Conflict	
Cost for Displacement	Condition of Conflict	
Cost for Replacement	Known Factors	
Cost for Exceptions	Unknown Factors	
Cost for No Action Taken		
Cost for Change		

With the weighting of the factors presented above, the cost of resolution of a conflict can be mathematically determined. The conflict cost requirements (product accuracy requirements), will set the ultimate goal of acceptance of the product. Analysis of known factors as well as unknown factors will be the next step in the process of determining the exact method which can be used for determining cost.

A determination of acceptability accuracy tolerances for the chart product entering the Conflict Detection/Resolution phase of finalization must be determined. This determination must reconcile the goal state of generating a product with a 100% accuracy rating, with the reality that the other phases in the cycle of product generation take away from the ideal situation of 100% accuracy and require conflict resolution to bring the product up to acceptable accuracy tolerances?

The detection and resolution of cartographic conflicts arising from the need to portray real-world features at a greatly reduced scale while maintaining chart readability is a task traditionally dependent on cartographers' skill and judgement. In this analysis it is demonstrated that it is feasible, for a specific chart product, to define a comprehensive set of requirements for conflict detection and resolution. These requirements, organized in matrix form, can be used to quickly determine if two chart features can create a conflict due to the overlap, proximity or coincidence of their symbology. Given that a conflict does exist, the concept of cartographic cost represents a means to select the best (least costly) conflict resolution strategy. Although more work is needed to complete the requirements for all problems these results demonstrate that this approach is a practical one.

2.2.3.4 Feature Smoothing

Once a set of cartographic features have had the shape-, or boundary-describing points reduced to a minimum, the lines can be adjusted to produce more natural-looking, smoother lines. Smoothing Algorithms are a major category of algorithms which operate on a line by physically moving point coordinate locations. Essentially, these algorithms produce a derived data set, which has had a cosmetic effect applied to it. Here, coordinates are shifted from their digitized locations. This is accomplished by diminishing variations in direction and reducing angles. In general, smoothing operators do not remove coordinates from the data file, they merely readjust their locations. The context of this section will be to view how smoothing algorithms can be applied to MC&G data once all the required features for a given product have been selected and simplified. Four major types of linear smoothing algorithms can be found in the literature. They are: (1) Averaging; (2) Epsilon Filtering; (3) Arc Substitution; and (4) Waveform Processing. Examples of each are presented on the following pages.

2.2.3.4.1 *Averaging*

Averaging Routines are those in which a local number of coordinate locations are summed and averaged to provide a new location for some n^{th} coordinate in the set.

Simple Averaging

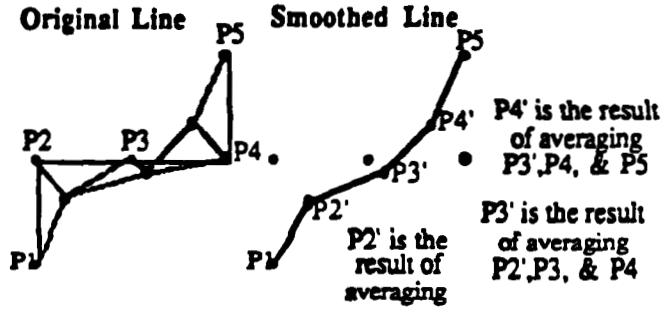
REFERENCE:

Koeman, C. and F.L.T. Van der Weiden (1970).[36]

ALGORITHM DESCRIPTION:

Averaging is a means of smoothing sequential x,y coordinate data by taking an average value for a set of recurring values along a line. These are generally referred to as moving averages, as the average is computed while processing along a string of x,y coordinates. A simple moving average will derive points for plotting by taking an unweighted mean of the positions of every N stored points, where N is an integer specified by a cartographer.

GRAPHIC EXAMPLE:



In the figure above, a sample line contains 5 coordinate positions. Point P2 is replaced by the average position of the triad of points P1, P2, and P3. A new triad of points is examined, and P3 is now replaced with the average position of points P2', P3, and P4. The process continues until the smoothed line contains five modified coordinate positions (points P1, P2', P3', P4', and P5).

ADVANTAGE:

Straightforward programming. Algorithm can be modified to retain the starting and ending points of a line.

DISADVANTAGE:

Algorithm is influenced by starting point. Tends to distort peaks and troughs.

Averaging (continued)

Weighted Moving Averaging

REFERENCE:

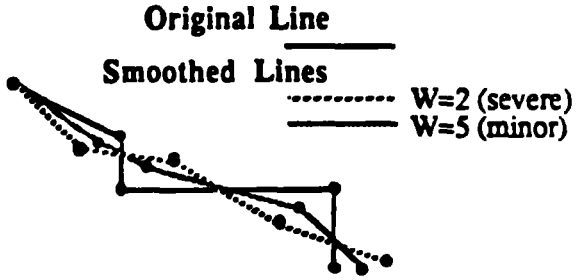
Tobler, Waldo R. (1966).[37]

ALGORITHM DESCRIPTION:

The moving average is augmented by weighting. The method assigns weighting values to each point in the calculation in order to increase or decrease its influence on the final point position. Generally, the central point in the set is weighted most heavily since it is the point being moved and moving it too far could seriously affect the character of the line. The weighting factor and the degree of smoothing bear an inverse relationship, with higher weighting factors resulting in lower smoothings. The weighted average coordinate positions (using a weighting factor W) are computed as follows:

$$X = \frac{X_1 + WX_2 + X_3}{W + 2} \quad \text{and} \quad Y = \frac{Y_1 + WY_2 + Y_3}{W + 2}$$

GRAPHIC EXAMPLE:



In the figure above, an original line has been shown as a solid line connecting 5 coordinate pairs. In addition, two smoothed lines are represented. The combination dashed/dotted line represents a minor or moderate smoothing of the original line, based upon a weighting factor of 5. The dashed line has had a weighting factor of 2 applied to it and, therefore, results in a more severe smoothing.

ADVANTAGE:

Data can be repeatedly smoothed by subsequent applications of this algorithm.

DISADVANTAGE:

Influenced by its starting point. Distort peaks and troughs, but less than simple averaging. Places more emphasis on the middle points being averaged. Smoothing level is dependent on weighting factor.

Forward-Look Interpolation

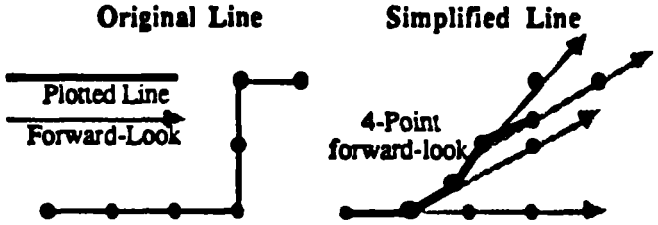
REFERENCE:

Boyle, A.R. (1970).

ALGORITHM DESCRIPTION:

Algorithm was designed to be applied to the data at the plotting stage. Plotting begins from the first point in a line segment (start point), with the plotter being driven towards the N^{th} point along the line (end point), where N is an integer specified by the cartographer. Plotting is halted when 1/N distance is reached along this line. The direction is recomputed to the next end point and plotting continues. As a result, a series of small vectors are created which vary from one another by only a small angle.

GRAPHIC EXAMPLE:



In the figure above, a sample line contains 7 coordinates. Assuming that we are employing a four-point forward look interpolation, plotting will commence, aimed four points down the line, and continue until it reaches $1/4^{th}$ the distance. At this point a coordinate position is accepted and plotting is redirected towards the next point down the line. Note how the simplified line is displaced substantially from the original line.

ADVANTAGE:

None.

DISADVANTAGE:

The computer is required to compute the distance from the start point to the end point every time the procedure is repeated—its application at the plotting stage will slow down plotting time considerably. In addition, the resultant line will be displaced somewhat from the original line. The caricature of the simplified line is highly dependent on the amount of forward look (such as 4 point versus 10 point).

2.2.3.4.2 *Epsilon Filtering*

Epsilon Filtering Routines are those in which an ε-generalized zone is created around a linear feature by rolling a ball along the linear edge to eliminate regions of divergence.

Epsilon Generalization (Perkal's Rolling Ball)

REFERENCE:

Perkal, J. (1965b).

ALGORITHM DESCRIPTION:

Perkal examined difficulties of length measurement and proposed a simple concept for linear generalization with the use of a circle of diameter epsilon ε. Here, the degree of generalization (smoothing) is defined by a real number ε which represents the length of a line segment. This line segment is considered to be the diameter of a circle rolling along a line. If the line as considered to be a hard surface, and the circle a wheel rolling on that surface, the circle would ride over the narrow ruts in the surface. Those points, or indentation, which are not covered or touched by the edge of the circle are eliminated. Points that are touched by the circle are retained.

GRAPHIC EXAMPLE:

In the figure above, note that within a region D, some points P included within the region have the property that there exists a circle of diameter ε which lies entirely within D and which contains the point. There are, however, points Q in D, such that no circle of diameter ε can contain the point. The use of a larger ball (that is, a larger ε) would result in greater divergence and thus greater generalization.

ADVANTAGE:

Reasonable approach for line smoothing necessitated by scale reduction.

DISADVANTAGE:

Can be applied to both sides of a line and to produce two lines known as the ε-generalized-boundary. The area between these two lines can be represented by a "heavy line," which would be aesthetically displeasing. Secondly, some features would be more simplified than others, and no provision is made for the graphic exaggeration of important features.

Epsilon Generalization (Brophy's Rolling Ball)

REFERENCE:

Brophy, David M. (1973).

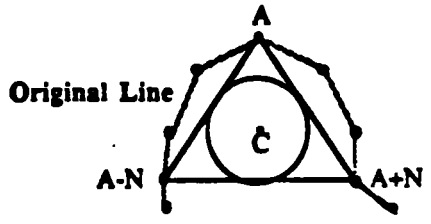
ALGORITHM DESCRIPTION:

Formulated after Perkal, is a complicated, interactive algorithm designed to affect scale, line width, generalization levels, and exaggeration or elimination of features. Line smoothing algorithm is based on systematically moving each point toward the center of curvature along the radius of curvature. This is guided by the curve being represented by approximating polygon of known vertices location. The program consists of operator-controlled components:

1. Determination of a subset of equally spaced points along the line on the basis of scale reduction.
2. Re-definition of the curve by connecting the coordinate pairs as a series of tangent points of finite width and equal to the line weight of the simplified line.
3. Selection of optional control points to retain critical points.
4. Elimination of unwanted features.
5. Systematic smoothing or exaggeration of non-straight sections of line.
6. Generation of plotting commands.

The smoothing operator (5), is affected by curvature. Each individual point is processed sequentially. Every N^{th} data point from the point under consideration defines a polygonal curve which approximates the actual curve. Around the point under consideration, A, a triangle is formed and an inscribing circle is placed within it. Simplification is achieved by moving the point toward the center of the intangent circle of the triangle. This amount of movement is proportional to N, and specified by the operator as the level of generalization.

GRAPHIC EXAMPLE:



In the figure above, an example of Brophy's smoothing algorithm shows point A being moved towards the center C of the intangent circle.

ADVANTAGE:

Theory is based on sound mathematical and geographical reasoning. The interactive mode of operation is highly desirable.

DISADVANTAGE:

Very complex and high computation time.

2.2.3.4.3 *Arc Substitution*

Arc Substitution Routines are those in which a mathematical representation of the original line replaces the original line.

Pseudo-Hyperbola

REFERENCE:

Vanicek P. and D.F. Woolnough (1975).

ALGORITHM DESCRIPTION:

The algorithm works on the principle of expressing generalization by a theorem applicable to any arc. Essentially, the digitized line is replaced by a series of segments or arcs of known radius. More specifically, the parameters of the curve are transformed into a set of pseudo-hyperbolae. Originally developed as a mathematical packing method, each digitized line is transformed into linear segments, such that no point lies outside of a given tube of tolerance epsilon (ε), surrounding the original curve. The coefficients of the pseudo-hyperbolae are determined using a set of x and y coordinates of line data, input and output scales, digitizer increment and the final required plotting accuracy ε. The equation for the coefficients of pseudo-hyperbolae may be expressed as $y = \pm (c_1 + c_2)/(x + c_3)$. By taking the average direction of the first three points in a stream of coordinates, successive points are selected until they fail to lie within a tolerance of width $\pm 8\varepsilon$. Using the beginning and ending points of this segment for proper direction, a check is made to determine if the internal points lie within a $\pm 1\varepsilon$ corridor. A new pseudo-hyperbola from this last point is defined in the direction of the previous segment, and sampling for the next segment proceeds.

GRAPHIC EXAMPLE:

None provided.

ADVANTAGE:

Will pack coordinates to a minimum number required for a given resolution.

DISADVANTAGE:

The longer the segment lengths, the fewer the segments, and consequently the greater the reduction in the amount of data stored. Thus, the packing procedure produces two coordinate points and an interlying segment for storage. It will pack coordinates for curves which are to be reproduced at the same scale too. There may be more than enough points on the curve to reproduce it with a given resolution and, if so, this program will reduce the number of points to the minimum required for any given resolution. Will not work for closed loops.

Polynomial Curves

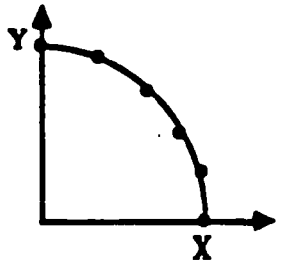
REFERENCE:

Breward, R.W. (1972).

ALGORITHM DESCRIPTION:

A polynomial is an algebraic expression having more than one term that has received a lot of attention in shape description. Shapes generally cannot be equated with single value functions, and are commonly represented in parametric form; this is where a two-dimensional shape is represented by a set of parameters t for each coordinate of a point location (x,y). Each coordinate of a point is represented by a function of one or more parameters. The line is represented by a series of polynomial equations, where the coefficients of the fitted polynomial and end coordinates of the contour section are stored.

GRAPHIC EXAMPLE:



In the figure above, an example of a parametric polynomial curve is shown. Here, each point on the original line is represented by the coefficients of a polynomial. In this case, the curve shown can be represented by a polynomial equation defined by:

$$x = \frac{1 - t^2}{1 + t^2} \quad \text{and} \quad y = \frac{2t}{1 + t^2}$$

ADVANTAGE:

Can save 85% in storage. To obtain the original coordinates, the polynomials are simply evaluated at successive positions along the line.

DISADVANTAGE:

Computation time is significant. Furthermore, the choice of criteria for terminating the process and the means of splitting the contour into sections is a problem. To terminate, Breward suggested that the procedure be stopped when the k^{th} order polynomial provides a worse fit than the k-1 order equation. Splitting the contour is done by a segmentation which has optimized the savings in storage versus the possible loss of accuracy.

Arc Substitution (continued)

Bezier Curve

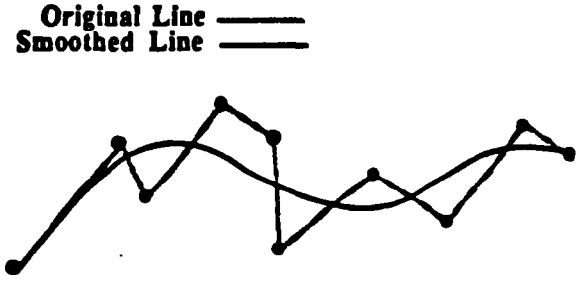
REFERENCE:

Bezier, P. (1971) in Clark, J.H. (1974).[38]

ALGORITHM DESCRIPTION:

A Bezier Curve is a method of curve description that is associated with the vertices of a polygon defining the curve shape. The curve can be defined by an open polygon, only the first and last vertices of which actually lie on the curve, other vertices describe the order and shape of the curve. Changing the vertices of this polygon will alter the curve shape in that area of the curve. Thus, the user can vary the curve shape and order by controlling the input parameters until the desired shape is reached. The mathematical basis of the Bezier Curve is a polynomial blending function which interpolates between the first and last vertices, and operates globally on a curve.

GRAPHIC EXAMPLE:



In the figure above, a Bezier Curve has been generated for the 10 coordinate points on the line.

ADVANTAGE:

None.

DISADVANTAGE:

Two characteristics of Bezier Curves limit their flexibility. First, there is no local control of the curve; if one point (polygon vertex) is altered, the curve changes shape throughout its length. Second, the number of polygon vertices specified fixes the order of the resulting polynomial which describes the curve. So, the only way to reduce the order of the curve is to reduce the number of vertices and, obviously, the only way to increase the order of the curve is to increase the number of vertices.

B-Spline

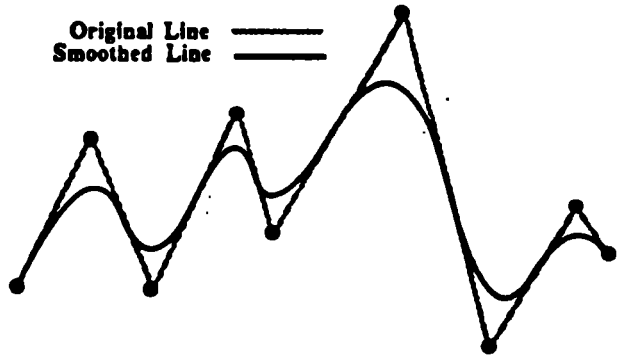
REFERENCE:

Riesenfeld, R.F. (1972) in Clark, J.H. (1974).[39]

ALGORITHM DESCRIPTION:

B-Spline theory is a spline function associated with the vertices of a polygon defining the curve shape. The curve can be defined by an open polygon, only the first and last vertices of which actually lie on the curve. This theory operates in a non-global, or local, realm. Each vertex of the polygon defining the curve affects the shape of the curve only over a range or parameters surrounding it. The B-Spline also allows the order of the resulting curve to be changed without changing the number of defining polygon vertices. Similar in theory to the Bezier Curves, B-Splines are mathematically based on a polynomial blending function which interpolates between the vertices of the defining polygon. However, here the blending function is formulated differently. A B-Spline curve is a weighted average of the vertex coordinates with the basis functions as weights (each vertex is associated with a unique basis function).

GRAPHIC EXAMPLE:



In the figure above, a B-Spline Curve has been generated for the 9 coordinates.

ADVANTAGE:

B-Spline curves are more desirable for geographic data than Bezier Curves because they operate locally and will smooth a line more gently.

DISADVANTAGE:

No cartographic basis. There is little local control of the curve; if one point (polygon vertex) is altered, the curve changes shape throughout its length.

2.2.3.4.4 *Waveform Processing*

Waveform Processing Routines are those in which the line is treated as a repetitive waveform that can be decomposed into a series of harmonic constituents with known amplitude and frequency; smoothing operates on these constituents.

Fourier Analysis

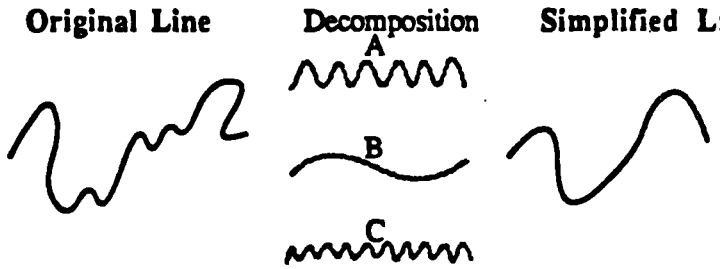
REFERENCE:

Anstey, N.A. (1965).

ALGORITHM DESCRIPTION:

The basic concept is to fitting sine waves to a curve. The principle behind Fourier analysis is that a line can be decomposed into its harmonic constituents. Thus, any repetitive waveform can be viewed as the addition of sine or cosine waves whose frequencies are integral multiples of that basic repetition. The basic repetition is called the fundamental, and the frequencies which are "x" times the fundamental, are called the harmonics. The algorithm, then, can analyze a line and break it down into a series of waves of known amplitude and frequency. For smoothing, once the harmonic constituents are calculated, the smallest can be eliminated and the others recombined to create a new line; the smallest being considered as insignificant, or noise, in the data set.

GRAPHIC EXAMPLE:



In the figure above, an example of Fourier Analysis is shown. Here, the original line (curve) has been decomposed into 3 sine waves. The smallest frequency sine wave, C, has been omitted as noise. Sine waves A and B are recombined to produce a smoothed version of the original curve.

ADVANTAGE:

None.

DISADVANTAGE:

Operates globally on the whole line at a time and therefore takes much time (and also more money) for computation. The major problem with the procedure is that it cannot cope with a line that doubles back on itself. It can only process sine waves, where for every location on the x-axis there is only one y-value. Removal of noise may, in fact, be destroying the characteristic, or shape-describing, inflections in the line.

Hysteresis Smoothing

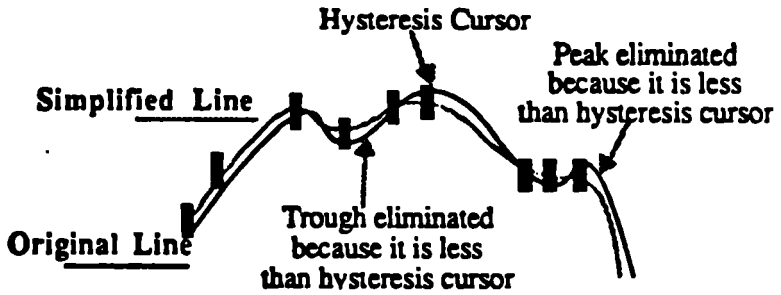
REFERENCE:

Ehrich, R.W. (1978).

ALGORITHM DESCRIPTION:

Algorithm operates by passing a hysteresis cursor, or tolerance band, of known width along a line to decrease the amplitude of the peaks and troughs in the wave. The tolerance band should have a width at least equal to the longest peak or valley to be removed. As the cursor is moved along the line the cursor looks-ahead, and peaks and valleys whose amplitudes are smaller than the cursor size are eliminated.

GRAPHIC EXAMPLE:



In the figure above, an example of Hysteresis Smoothing is shown. As the Hysteresis cursor, or tolerance band, is moved along the line, peaks and valleys whose amplitudes are smaller than the cursor size are eliminated.

ADVANTAGE:

Simple and relatively fast method of removing minor fluctuations or noise from a random line. Has an advantage over linear or global filtering in that it can remove small waveform fluctuations without reducing resolution.

DISADVANTAGE:

Can not function on lines which double back or on lines that are extremely sinuous, thus its utility for most cartographic lines is limited.

2.2.3.5 Data Compaction

Once all the cartographic features have been smoothed to produce more aesthetically-pleasing representations, the digital data is now ready for its final stage of reduction. Compaction algorithms operate on the storage structure of the information and address the physical data formats of the logical data. Issues at this level include the degree to which logical structures are computed or encoded.

In general, compaction operations do not remove coordinates from, or adjust coordinates within, the data file. The context of this section will be to view how compaction algorithms can be applied to MC&G data once all the required features for a given product have been selected, simplified, and smoothed.

Vector digitization results in the collection of large volumes of data. The development of coding schemes for vector data has hinged primarily on the need for data compression, with specific concern to the type of data captured and stored, as well as the techniques utilized to process and manipulate the data. The most common data structure for cartographic applications is the linear list.[40] The most prevalent linear-list substructure in cartographic use today is chain coding. The chain code is a slope-intrinsic representation of a shape that has been used extensively for representing curves or sequences of points. Although many other types of compaction algorithms can be found in the literature, only the Chain Coding type is discussed because of its prevalence. Chain coding can have many variations, and examples of each are presented on the following pages.

2.2.3.5.1 *Chain Coding*

Chain Coding algorithms produce a compressed data set, which has had a compaction applied to it to reduce the amount of storage required to represent the feature.

Basic and Differential Chain Coding

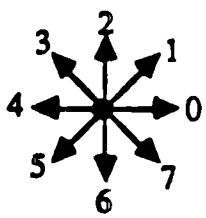
REFERENCE:

Freeman, Herbert (1961).[41]

ALGORITHM DESCRIPTION:

The simplest way to describe a curve is to record x,y coordinate pairs for each point on the curve. This method of storage is inefficient, however, and can be improved by recognizing that any single point in a rectilinear array has only 8 possible nearest neighbors.

The chain coding scheme records x,y coordinates relative to a previous location in terms of direction. An entire curve can be described by an initial x,y position followed by a sequence of directions to adjacent points. If the n^{th} point of the curve is at position (i,j), then the chain element corresponding to the change in position from n^{th} point to the $(n+1)^{st}$ point is shown in the figure to the right.



Several variations of the basic chain code have been offered to improve efficiency. One of these is a differential chain code where points are represented by a difference between two successive absolute points. The number of directions is the same as the basic chain code but are given the values; 0, ±1, ±2, ±3, ±4. For smooth curves, the values 0, ±1 occur more frequently. This makes it possible to utilize a variable-length encoding scheme with the differential chain code. Pavlidis has found that such an encoding usually requires no more than two bits per point on the average.

GRAPHIC EXAMPLE:

None provided.

ADVANTAGE:

Simple and relatively fast method of compacting data.

DISADVANTAGE:

Plotting times are increased since data requires decompaction.

Octant and Quadrant Chain Coding

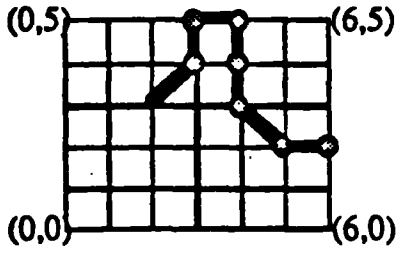
REFERENCE:

Baudelair, P. and M. Stone (1980).

ALGORITHM DESCRIPTION:

Two additional variations on the differential chain code have been described. The first one is based on the concept of quadrants and uses two bits to represent the differential increment. This scheme divides the eight possible curve directions into four quadrants represented by 0, 1, 2, or 3. Within each quadrant there are three possible directions or increments which are assigned the values 1 to 3. The encoding of a curve would start with the quadrant number (0 to 3) followed by the increment codes (1 to 3) and terminated by a 0. The second scheme divides the set of eight possible directions into eight quadrants. Within each quadrant there are only two possible directions which can be represented by one bit. Two bit streams are used: one indicates the octant followed by the number of one-bit increments; the second holds the actual one-bit increments.[42]

GRAPHIC EXAMPLE:



In the figure above, an example of Basic Chain Coding is shown. In normal cartesian coordinates, this linear feature of 8 points would be represented as (2,3), (3,4), (3,5), (4,5), (4,4), (3,4), (5,2), (6,2). In basic chain coding, the same line would be (2,3)1206670. Using the variable-length differential chain code, the same line would be represented as +1, +2, 0, -2, -2, -1, 0 (which would be encoded as 010111001111011110110).

ADVANTAGE:

This basic chain code scheme only requires 3 bits to store the direction, thus providing substantial savings in storage and is computationally efficient. The octal method offers the advantage of understanding the behavior of a curve by examining the octant codes alone. The higher order chain codes appear to provide potential advantages to cartographic data because of improved efficiency in storage, smoothness, and reduced processing times.

DISADVANTAGE:

Since the chain code is a slope intrinsic representation, it is not rotation invarient. In fact, rotating a curve can even change the length of the chain code. Higher-order chain codes are more complex to encode.

2.3 Endnotes

[1]White, Ellen R. (1983).

[2]Robinson, Arthur H., et al. (1978).

[3]ibid (1978), p.150.

[4]McMaster, Robert B. (1983a).

[5]Morrison, Joel L. (1978).

[6]The selection process discussed previously will only be reviewed as it applies to the selection of features required for the presentation of a particular chart product or group of products.

[7]The discussions on Line Simplification, Smoothing, and Compaction routines are more robust than the other areas of generalization for a number of reasons: (1) these areas have received greater attention in the literature; and (2) the algorithms and procedures are much more well defined—this report demonstrates our knowledge of the requisite subject materials, literature, and algorithms that are important to the development of the ANCS II.

[8]This form of linear point simplification is commonly referred to as line generalization. This includes all aspects of line manipulation such as simplification, smoothing, and feature displacement.

[9]In computer-assisted cartography, two basic forms of computer-readable storage currently dominate: (1) data stored as strings of coordinates (as a result of lineal digitization or the conversion of raster data to vector form); and (2) data stored as picture elements (remotely sensed or scanned). While the process of generalization can operate on both types of stored data, their implementation is quite different. Cartographic feature data contained in the ANCS II MC&G data base, however, is envisioned to be in the form of coordinate data representing vector strings; as a result, our discussion of cartographic generalization algorithms in this report will focus there. Image coding techniques for pixel-based information are outside the scope of this effort.

[10]McMaster, Robert B. (1983a).

[11]McMaster, Robert B. and K. Stuart Shea (no date) are discussing these concepts in a forthcoming publication.

[12]F. Töpfer and W. Pillewizer (1966).

[13]It should be stressed to the reader that the generalization process is more complex than merely simplifying lines as is often thought to be the full extent of map generalization.

[14]Shea, K. Stuart (1987a).

[15]McMaster, Robert B. and K. Stuart Shea (no date) are discussing these concepts in a forthcoming publication.

[16]Monmonier, Mark Stephen (1983).

[17]Shea, K. Stuart (1987b) discusses these concepts in a forthcoming article.

[18]McMaster, Robert B. (1987, in press) discusses these concepts in a forthcoming article.

[19]McMaster, Robert B. and K. Stuart Shea (no date) are discussing these concepts in a forthcoming article.

[20]For the time being, then, we are limiting the analysis of the map to geometric evaluations. Other problems—such as whether a *Stranded Wreck* and a *Sunken Wreck* is more complex than two (2) *Sunken Wrecks*—is beyond the scope of this analysis. These product-specific conditions must be addressed separately.

[21]How these measures are combined is beyond the scope of this report.

[22]Here, point buffer delineates the region around a point that accounts for the symbology. The same holds true for line and area features.

[23]Wertheimer, M. (1958).

[24]As part of this effort, PGSC developed a testbed environment for the analysis of linear simplification and smoothing algorithms. A review of this software is provided in Appendix B of this report.

[25]Jenks, George F. (1981).

[26]Jenks, George F. (1979).

[27]Jenks has suggested that simplification routines may, in fact, reduce a data set by as much as 70% without changing the perceptual characteristics of the line.

[28]McMaster, Robert B. (1987) personal communication.

[29]Although the following discussion will deal primarily with the generalization (that is, simplification) of linear map features, it should be noted that features and not merely lines are generalized. The reader should be cognizant of that concept. However, a solid understanding of simplification algorithms, a main constituent of the generalization process, is mandated.

[30]McMaster, Robert B. (1983b).

[31]McMaster, Robert B. (1983), "A Quantitative Analysis of Mathematical Measures in Linear Simplification," unpublished Ph.D. dissertation, Department of Geography-Meteorology, The University of Kansas.

[32]see also Lang, T. (1971).

[33]see also Peucker, Thomas K. (1975), Ramer, Urs (1972), and Reumann, K., and A.P.M. Witkam (1974).

[34]McMaster, Robert B. (1983), "A Quantitative Analysis of Mathematical Measures in Linear Simplification," unpublished Ph.D. dissertation, Department of Geography-Meteorology, The University of Kansas.

[35]Marino, Jill S. (1978), "Characteristic Points and their Significance in Cartographic Line Generalization, unpublished Masters Thesis, Department of Geography-Meteorology, The University of Kansas.

[36]see also Gottschalk, Hans-Jorg (1974) and Lichtner, Werner (1978).

[37]see also Connelly, Daniel S. (1971).

[38]see also Gordon, W.J., and R.F. Riesenfeld (1974).

[39]see also Rogers, D.F. and J.A. Adams (1976). In addition to B-Splines, Cubic Splines, Relaxed Splines, P-Splines, Q-Splines, and E-Splines can also be applied although their applicability to smoothing cartographic feature data has not been assessed to date.

[40]see also Horowitz, E. and S. Shani (1978).

[41]see also Pavlidis, T. (1977) and Baudelair, P. and M. Stone (1980).

[42]Freeman has also produced higher-order encoding schemes based upon sixteen, twenty-four, and higher chain codes.

THIS PAGE INTENTIONALLY LEFT BLANK

3.0 NOS GENERALIZATION REQUIREMENTS

Automating nautical chart production raises critical issues related to generalization including scale change, data selection, single versus multiple data bases, accuracy preservation, the role of spatial data structures, and many others. In comparison to typical mapping applications, the nautical charting situation represents a particularly problematic area for automated generalization and related issues. Nautical charts cover a wide range of scales, even over the same area of interest. Paragraph 2.2.1.1 of this report provides a classic example for a section of Long Island, New York, in which seven (7) differently-scaled charts ranging from 1:10,000 to 1:1,200,000 include the same geographic coverage. As stated in the Nautical Chart Manual—NOS's documentation governing all future nautical chart production—chart "accuracy of position, legibility, and uniform consistency in selection and placement of charted features, names, notes and other details are the chief requirements in nautical chart compilation."[1] Unfortunately, these criteria may be conflicting and necessitate trade-offs between them and other criteria such as production time and cost.

3.1 Accuracy Constraints on Generalization in Nautical Charting

Preserving accuracy is particularly critical for nautical charts because of the need to ensure the safety of navigation by the accurate portrayal of navigationally critical elements such as physical hazards, aids to navigation, and hydrography. The nautical charts produced by the NOS are premier examples of highly accurate and dependable products. NOS's unique approach to cartography, high accuracy standards, and product liability, makes them an anomaly in the mapping community. Since the nautical chart has such a unique requirement for detailed and accurate portrayal of the coastline and water forms, it must be considered the preeminent argument for accuracy-driven product generation. As such, the generalization process, when applied to nautical charting, obviates a need for increased awareness as to the influence of generalization on accuracy degradation. All forms of generalization, including the most radical form—scale change—must limit their influence on the accurate portrayal of features, both positionally and in attribution.

Limiting generalization's influence on accuracy preservation is not a trivial problem. Charts are reductions of reality; generalization, therefore, is inherently part of the nautical charting process. The manual production of charts present many situations where

generalization occurs. For example, NOS's guidelines require that for line data, the compiled or engraved/scribed line be within 1/2 the symbol lineweight (not to exceed 0.15mm) of true position of the line. Similar requirements exist for point data and soundings. Digital chart production, however, raises the generalization process to a more critical level. The representation of cartographic information as digital entities creates an illusion of independence from scale. Mathematical transformations can create many differently-scaled products from a single digital data base, but these processes are not independent from the influence of generalization. The meaning of charted features, the graphic representation of features, and the characterization of features is not scale free. The unique accuracy requirements for NOS products constrain the ability to (1) achieve maximum legibility across scales and (2) make use of a single or small number of source digital charting data bases to support the wide range of required scales. In a digital production environment, such as that envisioned in the ANCS II, how then will generalization affect the charting process?

3.2 Proposed ANCS II Generalization Processes

The ANCS II Draft Specification contains a high-level concept of operations for the digital compilation of a chart. Generalization requirements can be extracted from this discussion. For example, a generalized coastline is used as a background display for Source Data Index File retrievals. More directly related to chart production is the process that creates and uses the Chart Edit Packet or Work File. For the area of interest identified, the Chart Edit Package contains the data base feature records along with appropriate header information. This data file is then processed to create vector nautical charting symbology. The resulting reformatted Chart Edit Packet is used to generate a Chart-Specific Edit Packet for the first chart to be produced. Prior to this step, the symbol coordinates are transformed into the *x,y*-coordinate system of the largest-scale chart within the work area.

For line features and others indicated as being modifiable, a point elimination routine will be applied to delete excessive points contained in the source data base. In addition, features represented as closed polygons that coalesce at the chart scale under consideration will be automatically converted to a suitable point symbol representation. The Chart-Specific Edit Packet for the largest-scale chart may be used as a model from which selection is made for developing Chart-Specific Edit Packets for smaller scale charts. Thus, an iterative procedure is envisioned in which processing is carried out for successively

smaller scale charts based on the results of the previous processing and resulting Edit Packet.

This conceptual processing flow for chart production clearly utilizes generalization and also makes use of some implicit assumptions regarding scale change and generalization. One assumption that may be made is that there exists a single source data base for the features to be portrayed on all charts. This source data base includes a single, centerline representation of features at the full level of detail, resolution, and accuracy obtained during source data collection. This data is then extracted for a defined coverage area and symbolized in vector format. At this point, no generalization has occurred and the work file is product- or chart-independent. (Note: this assumes identical symbolization rules for all charts.)

The conceptual chart production flow then begins to create Chart-Specific Work Files beginning with the largest scale chart(s) to be produced within the coverage area. The use of a hierarchical, incremental generalization procedure has many attractive aspects and is appealing as a logically simple approach to the problem of producing charts at a variety of scales in the same area of interest. However, it also raises some questions. For example:

- For a given coverage area, will the entire area be symbolized/processed at the largest chart scale within the area? In other words, if a few isolated pockets of a very large scale coverage area is required, would the entire work file area be processed at that scale? If yes, much extra processing will be required. If no, the conceptually appealing simple iterative processing flow is impossible. That is, at any given step in the process the *source* work files for a particular chart scale may exist at different scales.

- Is the process of generalization strictly monotonic and incremental? In other words, is it possible that some features may have been correctly eliminated or generalized during processing for a large-scale chart and does this create problems during processing for a smaller-scale chart? It may be the case that in converting the work file for a 1:40,000-scale nautical chart to the 1:80,000-scale chart, some features have been eliminated or converted from polygons to point symbols. However, in producing a 1:100,000-scale topographic/bathymetric chart of the same area, the eliminated feature is required or the feature converted from area to point needs to be shown as an area. The ANCS II processing flow assumes total scale dependence for all generalization processes; further analysis is needed to determine if this assumption is correct or is a good one. There may be some important product dependencies which are somewhat independent of scale.

- In certain generalization algorithms and applications, it may be beneficial to retain the most detailed version of the feature to create accurate generalized versions of the feature. The incremental approach may make this possible.

- Does this approach to scale change assume incorrectly that the same generalization algorithms/processes that are appropriate in, for example, converting a 1:10,000-scale work file to a 1:20,000-scale work file, are also the most suitable for converting a 1:675,000-scale work file to a 1:1,200,000-scale work file. The optimal techniques to apply at each of the scales may differ.

- Does this approach assume that the source data base, for a specified coverage area, exists at a single level of detail/resolution/scale? If not, different generalization requirements will exist over the coverage area creating a more complex processing situation than that described in the ANCS II Specification.

- Pre-generalization, one-time symbolization. It may not be appropriate for the scale-change/generalization processing to operate directly on symbolized data. Will the original centerline data be available? In many case generalization will lead to changes in symbolizations; why not symbolize following the creation of Chart-Specific Work Files?

3.3 Discussion

As one can surmise from the discussion so far, the NOS is faced with a significant problem in terms of automating this generalization process in the ANCS II. Variations in the precision and detail required to satisfy the needs of different users give rise to a requirement for a variety of chart scales. Nautical charts vary in scale with the importance of the geographic area, the purpose for which the chart is designed, and the necessity for showing clearly all dangers within that area. The NOS has the specific task of publishing and maintaining over 900 nautical charts for the safety of navigation in the coastal waters of the United States and its possessions. The nautical charts produced by the NOS are respected worldwide as an excellent display of accuracy and dependability. Any generalization process must obviously limit the degradation of that accuracy.

3.3.1 Shorelines—A Generalization Example

To limit the discussion of generalization requirements to fit within the scope of this effort, one feature type was selected for examination: **Shorelines**. Of the many features that appear on NOS products, the shoreline is the most prominent line on the chart. This is evident for a number of reasons, not the least of which is that it forms one of the most obvious dangers to waterborne navigation. Safe navigation of our coastal areas and harbors is, in part, based upon the accuracy of shoreline portrayal. The long coastline of the United

States totals over 100,000 miles of tidal shoreline that presents many and varying problems in coastal geography. Add to this a vast array of extensive intracoastal waterways, bays, and harbors, it is obvious that shoreline portrayal represents an important constituent of modern nautical mapping.

By definition, a shoreline is the intersection of the land with the water surface. The shoreline on charts represents the line of contact between the land and a body of water at a selected water elevation. The exact location of the shoreline depicted depends, in part, on the vertical datum upon which the chart is based. This dividing line between land and water is referred to as the "Shoreline Plane of Reference" (SPOR). In areas affected by tidal fluctuations, this line of contact is usually the Mean High Water line. In confined coastal waters, where there is diminished tidal influence, a mean water level line may be used. The shoreline on charts of interior waters (rivers and lakes) is usually based on a specific river or lake datum.

The shoreline's charted position, because of its importance to navigation, must have high positional accuracy. In most cases this is true, but accuracy degradation is allowed in others. For example:

> "The accurately determined shoreline reveals the physical geography of the shore. It reflects effects of prevailing currents, wave fronts, and storms. The shoreline delineates the seaward limits of both marsh and swamp areas, for to the mariner this limit appears as the visible shoreline...The seaward extent of marsh is accurately surveyed, but the inshore boundary may be generalized, as the ragged indentations into the fast land are of little importance on the nautical chart...The vegetation of swamp land makes it appear as fast land to the mariner; knowledge only of its general location is sufficient for charting."

How then can generalization of the shoreline be controlled?

3.3.2 Suggested Approaches to Shoreline Generalization

Line Simplification routines, such as the Douglas (1973) corridor or the Lang (1969) tolerancing algorithms, are particularly useful for removing unnecessary coordinates from digital files. The resulting simplified representations of the shorelines will have minimal vector and areal displacement from the original lines. These routines require modifications, though, to account for the specific representations of shorelines the NOS employs. For example, shorelines are broken for soundings in narrow rivers and tributaries

where the sounding units would otherwise be obscured by the shoreline. Although this is a manual production step, the ANCS II system may use this type of digital encoding scheme for shoreline representation.

In support of scale reduction for multi-product exploitation from a single MC&G data base, line simplification routines offer some, but not all, the requisite techniques. For small scale changes, these routines provide an appropriate method for removal of superfluous coordinates to support the scale reduction; for larger scale changes, however, little work has been done to determine the effects of scale change on simplification. Other routines, then, such as the epsilon filtering routines discussed by Perkal (1965), Brophy (1972), and Chrisman (1983), show promise for large scale reductions but refinements are needed to optimize their performance.

The NOS must undertake a serious study determining the specific requirements of generalization for all features, for all charts, for all circumstances, in their envisioned production scenario. The ANCS II is a promising chart production and maintenance system but to fully exploit its capabilities, more work is needed in the areas of data base structure design and data base management. Once a decision is made as to the question of *separate data bases-single product* versus *single data base-multiple products*, a better appreciation for the role of generalization within the system can be gained. The ability to generate multiple products from a single data base is obviously a desired approach but current scale change routines are not sophisticated enough to outweigh the overhead costs of carrying multiple scale coverages. This report, however, pointed out some of the techniques, and provided a framework within which a complete generalization/scale-change system could be designed.

3.4 Endnotes

[1]U.S. Department of Commerce (1986), p.2-119. The Nautical Chart Manual is intended to provide a comprehensive documentation of cartographic standards, procedures, and policies for use within the National Ocean Service in the production of nautical charts.

THIS PAGE INTENTIONALLY LEFT BLANK

4.0 SUMMARY, CONCLUSIONS, & RECOMMENDATIONS

There are a wide variety of generalization procedures and algorithms available for vector data processing. These are derived from many different disciplines and, as such, do not necessarily comply with the specific requirements of geographical/cartographic data. It was shown in this report that "generalization" is a complex concern that must be addressed by NOS to reduce the size and complexity of an MC&G data base and to support scale reductions for multi-product exploitation of a single data base. An integrated approach to transforming the input data—existing digital data bases, raster-scanned maps, charts, and images, as well as new lineal digitized data—into "reduced" data sets can be realized given an understanding of the techniques available.[1] This report has accomplished that.

This section of the report will first provide a brief summary of the topics considered to be of importance when evaluating the effectiveness of the various algorithms for the entire range of the data reduction process.

4.1 Summary—General Observations

A number of general considerations should be made when judging the effectiveness of all generalization algorithms. For example:

- The ability of chart generalization to mimic manual methods is not essential; manual methods merely provide a baseline to evaluate the automated methods.[2]

- The generalization practice cannot, and should not, be carried out without adequate accompanying information to aid in the determination of the proper generalization procedures to select. This obviously raises some serious concerns in an automated- or semi-automated production environment that is involved in only the product finishing stages of production. A thorough knowledge of the original distribution is required in order to accurately perform the generalization; it is inadvisable to base one generalization upon another.

- Omission of features based simply upon size is likely to be erroneous. The physical character of an area may be expressed by the domination of many like features. If smaller features were arbitrarily eliminated the resultant genralization would indicate a completely different character. Thus, the true geographical and geomorphological characteristics of areas need to be maintained, even if it requires combination, exaggeration, and displacement.

- Automated generalization routines should mimic manual methods only in terms of selecting the same characteristic, or shape-critical, spatial relationships that a cartographer, or *most* cartographers, would choose.

- Topological relationships should be retained even after the generalization in the event that the data is used for analytical processes.

- Different types of features may require differing generalization routines, or the same routine with different levels of generalization tolerances applied.

- Each technique has an associated impact on data set integrity—techniques either (1) retain all data; or (2) result in the loss or modification of data. Implementation of a particular generalization algorithm must acknowledge this underlying concern by forecasting the expected usage of the data set. If the data is to be used for navigation and guidance or some other accuracy-critical function, then the technique must not cause any significant data loss that will impair the functionality of the corresponding system. This, of course, must be within the context of the scale-change required to support a particular product. On the other hand, data used for merely visual reference can undergo limited loss or modification of the data as long as these changes are not at the expense of the visual integrity of the data.

- Following the assumption that some amount of data loss is inevitable (whether intentional or not), the visible affects of data reduction, either through heuristic observations or based upon statistical and quantitative support, must not be detrimental to the intended use of the product.

- Each technique has some statistical accuracy associated with the generalization process—evaluation of this supporting statistical data is warranted if data accuracy is important.

- The balance of generalization between different features needs to be carefully controlled. Thus, features which are related in some fashion—such as the form of the land surface and drainage—must be considered together.

- The temptation to under-generalize where there is available space, and to over-generalize where features are crowded together, must be avoided; that is, generalization should be consistently applied across the chart.

- Processing times, as previously noted, are limited in the production environment. Some techniques that have an *almost perfect* generalization at a significantly reduced processing cost over the *perfect* technique might be a more logical choice.

- There are some instances where the selection of one technique may be more appropriate over another technique even within the same data type. For example, sounding data compression for display purposes obviously can undergo more significant reductions than can sounding information used for navigation purposes.

- The order of processing for each of the techniques is an area that requires much more knowledge than is currently available today. There is little indication that there is an "average" generalization process, or even requirement, so developing a standard procedure within a serial computer can only solve a limited set of the

problems. The drive here is to solve *as many* of those problems with a limited number of techniques.

- Generalization can be performed in software, hardware, firmware, etc. The advantages and disadvantage of each of these should be evaluated against initial and replacement costs, implementation timelines, processor speeds, and any other factors that influence a make/buy decision. For example, are there any examples within the current market place? How cost effective are they? What are their reliability factors?

- One must consider how data base management strategies affect the selection of a particular technique. For example, do sector mapping strategies, update strip sizing trade-offs, and indexing scheme complexities affect the generalization process.

- How does the selection of particular data structures affect the generalization? Will the selection of one over another aid or hinder the generalization process? For example, will the selection of topological data structures' usefulness outweigh the overhead costs of carrying the topological pointers?

- What type of data base indexing scheme is most appropriate for storage and retrievals; that is, is an R-tree, KDB-tree, Quadtree, B-tree, or some other indexing scheme most appropriate for all, or some, of the data?

4.2 Summary—Specific Observations

Simplification. When evaluating the effectiveness of simplification algorithms, the following factors should be considered:[3]

- Simplification algorithms should ideally reduce a data set to a minimum of points, by rejection of redundant points, or through the selection of significant points.

- Simplification algorithms should operate within the imperceptible realm, whereby map readers can perceive no difference in the line before and after simplification.

- Feature locations should not deviate significantly from their correct locations.

- Features are generalized, not lines. This implies that inter- and intra-relationships between various feature sets must be considered in the generalization process.

- Small irregularities should be removed from the lines, however, the character of the line should be maintained.

Combination. Many cartographic features will need to be combined to support product- or scale-change objectives. The effectiveness of combination algorithms must address the following:

- Combination of like features must obviously combine only those features that can be combined according to the specific product requirements.
- The combination of features must take into account the nature of the physical separation between features. If the attribution is simply a classification of a major element—such as small areas of forest combining into one because the forest and intervening spaces characterize different aspects of the land surface—then they can be grouped together. On the other hand, if small features are separated by different physical elements—small island (land) with intervening water—they should not generally be combined.
- Features that would otherwise be deleted from the product because of scale implications must only be combined to suit the needs of the chosen chart purpose.
- The new spatial depiction of the combined feature must be a logical extension of the individual entities grouped; that is, the general form of the features (such as shape) should be maintained.
- The measured area of the combined area should remain roughly the same as the area of the individual components.
- The implication of the combination must be assessed before the features are grouped. For instance, will the agglomeration of numerous small lakes into a single body of water violate political ownership and resulting depiction on the graphic.

Refinement. Distributions of cartographic features will need to be refined to support scale-change objectives. The effectiveness of refinement algorithms must address the following:

- In general, the original character, form, size, and spaces of the features should be maintained despite decreasing number.
- The generalization is not only the deletion of features but also the graphic representation of the true distribution by fewer and coarser means.
- As less information is shown locationally, it becomes increasingly important for the symbology to reflect the important characteristics of the feature. This requires an understanding of the real geographic features involved. For example, during scale reduction it is not possible to shown all meanders of a river in the true locations, but the fact that the river is characterized by meanders should not be lost.

- Spacings between features, shapes of the original features, and orientations should be maintained as closely as possible in the reduction; that is, the general impression of the black-white ratio and distributional character should be consistent between the original and reduced scales.

- Features of landmark significance must be omitted from the refinement procedure so as not to lose that significant characteristic.

Conversion. The geometric depiction of many cartographic features will need to be converted to support product- or scale-change objectives. The effectiveness of conversion algorithms must address the following:

- In the collapse of features, centerlines should be mainatained.

- Conversion of like features must obviously convert only those features that can be converted according to the specific product requirements.

- Conversion of like features must examine the specific attribution of the features to ensure that only exact features be converted.

- The conversion must not detract from the individual importance of any significant feature.

- Relationships with other features must be assessed and evaluated against the conversion result; that is, will the aggregation of two houses into a larger house violate the topological relationships of a road that runs between them?

- Features that would otherwise be deleted from the product because of scale implications must only be converted to suit the needs of the chosen chart purpose.

- The new spatial depiction of the converted feature must be a logical extension of the original feature or features; that is, the general form of the features (such as shape) should be maintained.

Displacement. Due to the symbolization step of product finishing, many cartographic features will need to be displaced in order to fit within the graphic constraints of a chart. The effectiveness of displacement algorithms must address the following:

- Feature associations must be considered. This is especially important in feature displacement, knowing what the spatial relationships of the features are.

- The displacement must allow not only for the printing resolution of features on the graphic, but also the visual acuity of the map readers. Features will tend to blend into others.

- The importance of features must guide the displacement process; that is, less important features must be displaced away from more important ones.

- The impact of displacement on other conflicts must be determined. Displacemement propagation needs to be evaluated.
- The creation of labelling conflicts due to feature type conversion must be addressed in the symbolization stage, not in the type placement stage.
- Features that are associated with one another (such as a road and railroad going over the same bridge) must be displaced the same relative to other features.
- Conflicts with the chart background need to be eliminated.

Smoothing. Vector data set manipulations should not cease with the simplification processes. Instead, these data can be further manipulated through effective exploitation of linear smoothing algorithms. The effectiveness of linear smoothing algorithms must address the following:

- Smoothing algorithms operate before and/or after simplification, and produce smoother, more natural-looking linear features which have been modified in their spatial locations.
- The algorithms should operate within the barely-perceptible realm, whereby map readers can perceive no major difference in the line before and after smoothing.
- The requirement for smoothing operations when dealing with raster-graphic display devices may not be warranted depending on the resolution and type of the display monitor. Raster displays will render smoothing operations ineffective due to aliasing effects. Vector-based graphic displays, however, can be improved with smoothing operations.

Compaction. The following considerations should be addressed when evaluating the effectiveness of automated linear compaction algorithms.

- If a compaction technique is chosen, speed is of the essence in the decompaction process in the production process. Compaction, on the other hand, is not as time-critical. Yet, there are implicit/explicit relationships between the compression and decompression processes. These need to be considered in the selection of a particular technique.
- The highest compaction ratios that can be expected from a particular approach are not necessarily the ideal selection. This compression ratio must be balanced with the overall encoder/decoder complexity.
- Compaction algorithms should ideally reduce the overall memory, transmission, and storage requirements of the data set without resulting in any obvious associated increase in processing times due to the compaction algorithm selection.

- Besides compressing the data sets, the compaction routines should make the processing of the data more efficient by removing the unnecessary, and sometimes arbitrary, level of detail.

- Compaction encoder complexity must consider the associated decompaction decoder complexity. A n-fold decrease in storage requirements is not an appropriate selection if there is an n-fold increase in decoding processing time or decoder complexity.

- Linear digital data sets can be compacted using a technique such as Chain Coding. It is not appropriate at this time to forecast a percentage change in the storage requirements, however, because this "increase" or "decrease" will be dependent, in part, on the overall complexity of the source material. Large numbers of point features obviously will not be affected by compaction routines. On the other hand, a greater number of linear and/or areal features will result in an associated decrease in storage using compaction routines.

4.3 Conclusions and Recommendations

Preliminary recommendations for set of integrated techniques that could together meet the requirements require that vector data sets undergo some form of cartographic generalization to reduce the overall data base size. This should be an integrated process of feature selection, simplification, combination, conversion, refinement, displacement, smoothing, and compaction to support the generation of scale-, application-, or function-specific data bases. Selection should be based upon the intended use of the product/data base. Simplification should be be cartographically-sound using a linear simplification algorithm such as the Lang Tolerancing or Douglas Corridor selection. Feature Conversion, Combination, Refinement, and Displacement must be performed with respect to the individual product requirements, while still maintaining the characteristics of the original information. It is here that the level of current automated generalization maturity—or, more appropriately, immaturity—must be most evident. Smoothing of the linear digital data files should be consistent with the type and resolution of the graphics display. Here, a simple weighting function should suffice. Finally, some form of compaction, such as Chain Coding, should be applied to all linear data files.

4.4 Endnotes

[1]"Reduced" data in this sense indicates the end-result, transformed data bases that have undergone the entire range of reduction manipulations—compression, generalization, simplification, compaction, and coding—to reduce the storage, memory, and transmission requirements.

[2]Caldwell, Douglas R., et al. (1984).

[3]Linear simplification of vector-based data is still an emerging research topic. Even so, the cartographic community maintains that its development cycle has passed its infancy stages and is now at an overall level of sophistication whereby these techniques can be effectively applied to linear digital data sets to support the data reduction processes. At present, simplification algorithms, such as the Lang Tolerancing and Douglas Corridor, are considered to be the most cartographically-sound for point removal. The Lang algorithm is an excellent choice as an initial "cleaning" or low-pass filter, while the Douglas algorithm is more appropriate for those features requiring more stringent generalization. Both choices should be considered in the development of the ANCS II production system.

Appendix A—Bibliography

Allam, M. M. (1978), "DTM Application in Topographic Mapping," *Proceedings*, Digital Terrain Model Symposium, ASP-ACSM, St. Louis.

Arnheim, Rudolf (1976). "The Perception of Maps." *The American Cartographer*, 3(1): 5-10.

Attenave, F. (1954), "Some Informational Aspects of Visual Perception," *Psychological Review*, 61:183-193.

Barrow, H.C. and R. J. Popplestone (1971), "Relational Description in Picture Processing," in B. Meltzer and D. Michie, Eds., *Machine Intelligence*, (New York:Elsevier), 6.

Basset, K. (1972), "Numerical Methods for Map Analysis," *Progress in Human Geography*, 4:219-254.

Baudelair, P. and M. Stone (1980), "Techniques for Interactive Raster Graphics," *Computer Graphics*, 14:314-320.

Beckett, P. (1976), "Cartographic Generalization," *The Cartographic Journal*, 14(1):49-50.

Bie, S.W. (1980), "Map Generalization - A Statement on General Problems," in H. Opheim, Ed., *Contributions to Map Generalization Proceedings*, (Oslo, Norway:Norwegian Computing Center), pp. 5-10.

Blakemore, M. (1983), "Generalization and Error in Spatial Data Bases," *Proceedings*, Sixth International Symposium on Automated Cartography, Automated Cartography: International Perspectives on Achievements and Challenges, AUTO-CARTO VI, held in Ottawa, Canada, 16-21 October 1983. Barry S. Wellar, Ed., (Ontario:The Steering Committee Sixth International Symposium on Automated Cartography), pp.313-322.

Bookstein, F.L. (1979), "Fitting Conic Sections to Scatter Data," *Computer Graphics and Image Processing*, 9:56-71.

Borodin, A.V. (1974), "Quantitative Criteria for Generalisation of Contents of Geographical Maps on an Electronic Computer," in Erno Csati, Ed., *Automation the New Trend in Cartography*, (Budapest, Hungary:The Geocartotechnical Research Department), pp. 154-164.

Boyer, L. (1981), "Generalization in Semi-Detailed Geomorphological Mapping," *ITC Journal*, 1:98-123.

Boyle, A. Raymond (1970), "The Quantised Line," *The Cartographic Journal*, 7(2):91-94.

Breward, R.W., (1972), "A Mathematical Approach to the Storage of Contours," *The Cartographic Journal*, 9(2):82-86.

Briggs, I.C. (1974), "Machine Contouring Using Minimum Curvature," *Geophysics*, 39:39-48.

Brophy, D.M. (1972), "Automated Linear Generalization in Thematic Cartography," unpublished Master's Thesis, Department of Geography, University of Wisconsin, 99p.

_____. (1973), "An Automated Methodology For Linear Generalization in Thematic Cartography," *Proceedings* of the ACSM, Washington, D.C., pp. 300-314.

Caldwell, Douglas R., Steven Zoraster, and Marc Hugus (1984), "Automating Generalization and Displacement Lessons from Manual Methods," Technical Papers of the 44th Annual Meeting of the American Congress on Surveying and Mapping, 11-16 March, Washington, D.C., pp.254-263.

Catlow, D. and D. Du (1984), "The Structuring and Cartographic Generalization of Digital River Data," *Proceedings* of the ACSM, Washington, D.C., pp. 511-520.

Chrisman, N.R. (1983), "Epsilon Filtering: A Technique For Automated Scale Changing," *Proceedings* of the ACSM, Washington, D.C., pp. 322-331.

Christ, Fred (1973), "Digitizing, Digitizer Editing and Graphic Output of Topographic Map Data," *Informations Relative to Cartography and Geodesy*, Translations, No. 30:5-10.

_____. (1974a), "Proposal of a Standard Test for the Examination of Interactive Cartographic Systems," *Proceedings*, Seventh International Conference on Cartography, Madrid, Spain, pp. 3-8.

_____. (1974b), "Rationalization of Map Information on a Topographic Map in Relation to Automatic Production Procedures," *Proceedings* of the Seventh International Conference on Cartography, Madrid, Spain, pp. 15-21.

_____. (1975), "Automatically Symbolized Output of Map Data Compiled and Selected From a Data Base," *Informations Relative to Cartography and Geodesy*, Translations, 32:5-16.

_____. (1976), "Fully Automated and Semi-Automated Interactive Generalization, Symbolization and Light Drawing of a Small Scale Topographic Map," *Nachricten aus dem Karten-und Vermessungswesen*, Uhersetzunge, Heft nr. 33:19-36.

_____. (1978), "A Program for the Fully Automated Displacement of Point and Line Features in Cartographic Generalizations," *Informations Relative to Cartography and Geodesy*, Translations, 35:5-30.

Clark, J.H. (1974), "3-D Design of Free-Form B-Spline Surfaces," Department of Computer Science, University of Utah.

Connelly, Daniel S. (1971), "An Experiment in Contour Map Smoothing on the Experimental Cartographic Unit (ECU) Automated Contouring System," *The Cartographic Journal*, 8(1):59-66.

Cosgriff, R.L. (1960), "Identification of Shape," Ohio State University Research Foundation, Columbus, Ohio, Report 620-11, ASTIA AD 254 792.

Dasgupta, Sivaprasad P. (1964), "Some Measures of Generalization on Thematic Maps," *The Geographical Review of India*, 26:73-78.

Davis, D.M., J. Downing, and S. Zoraster (1982), "Algorithms for Digital Terrain Data Modeling," Final Report for Contract DAAK70-80-C-0248, (Fort Belvoir, Virginia:U.S. Army Engineering Topographic Laboratories), 209p.

Davis, J.C. (1973), Statistics and Data Analysis in Geology, (New York:John Wiley and Sons), 550p.

Davis, L.S. (1977), "Understanding Shape: Angles and Sides," *IEEE Transactions on Computers*, C-26(3).

Dent, Borden D. (date unknown), "A Note on the Importance of Shape in Cartographic Communication," *The Journal of Cartography*, 71(7):393-401.

Department of Commerce (1986), "Specification for an Automated Nautical Charting System II," Cartographic Technology Programs, NOAA Charting Research and Development Laboratory, National Ocean Service.

Department of Defense (1981), Glossary of Mapping, Charting, and Geodesy Terms, 4th Edition.

Deveau, T.J. (1985), "Reducing the Number of Points in a Plane Curve Representation," *Proceedings*, Seventh International Symposium on Automated Cartography, Digital Representation of Spatial Knowledge, AUTO-CARTO VII, held in Washington, D.C., 11-14 March 1985. (Falls Church, VA:American Society of Photogrammetry and the American Congress on Surveying and Mapping), pp. 152-160.

Dettori, G. (1982), "An Outline Algorithm for Polygonal Approximation of Digitized Plane Curves," *Proceedings*, Sixth International Conference on Pattern Recognition, Munich, Germany.

Dettori, G. and B. Falcidiena (1982), "An Algorithm for Selecting Main Points on a Line," *Computers and Geosciences*, 8(1):3-10.

Dougenik, J.A. (1980), "WHIRLPOOL: A Program for Polygon Overlay, *Proceedings*, of the International Symposium on Cartography and Computing: Applications in Health and Environment, AUTO-CARTO IV, American Congress on Surveying and Mapping and American Society of Photogrammetry, pp. 304-311.

Douglas, David H. and Thomas K. Peucker (1973), "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Character," *The Canadian Cartographer*, 10(2):112-123.

Duda, R. and P. Hart (1973), Pattern Classification and Scene Analysis, (New York:John Wiley and Sons).

Dutton, G.H. (1981), "Fractal Enhancement of Cartographic Line Detail," *The American Cartographer*, 8(1):23-40.

Eastman, J. Ronald (1981), "The Perception of Scale Change in Small-Scale Map Series," *The American Cartographer*, 8(1):5-22.

Eckert, Max (1908), "On the Nature of Maps and Map Logic," Trans. by W. Joerg. *Bulletin of the American Geographical Society*, 40(6):344-351.

Ehrich, R.W. (1978), "A Symmetric Hysteresis Smoothing Algorithm that Preserves Principal Features," *Computer Graphics and Image Processing*, 8:121-126.

Fahey, Lawrence (1954), Generalization as Applied to Cartography. M.A. Thesis. (Columbus, Ohio: Ohio State University), 180 p.

Feder J. and Herbert Freeman (1966), "Digital Curve Matching Using a Contour Correlation Algorithm," *IEEE International Convention Record*, 3:69.

Floyd, A.M. (date unknown), "Generalization Problems in Derived Mapping," *Symposium on Cartography*, pp 4-7.

Freeman, Herbert (1961), "On the Encoding of Arbitrary Geometric Configurations," *IEEE Transactions on Computers*, EC-10:260-268.

_____. (1967), "On the Classification of Line Drawing Data," In Wathen-Dunn, Ed., *Models for the Perception of Speech and Visual Form*. (Cambridge MA), pp. 408-412.

_____. (1969), "A Review of Relevant Problems in the Processing of Line Drawing Data," In Graselli, A., Ed., *Automatic Interpretation and Classification of Images*, (New York), pp. 155-174.

_____. (1974), "Computer Processing of Line Drawing Images," *Computing Surveys* 6(1):57-97.

_____. (1976), "Analysis of Line Drawings," In J. Simon and A. Rosenfeld, Eds., *Digital Image Processing and Analysis*.

_____. (1978), "Shape Description Via the Use of Critical Points," *Pattern Recognition*, 10:159-166.

Freeman, Herbert and Jeremy M. Glass (1969), "On the Quantization of Line Drawing Data," *IEEE Transactions on Systems, Sciences and Cybernetics*, SCC-5(1):70-79.

Freeman, Herbert and Goffredo G. Pieroni, Eds. (1978). Map Data Processing: Proceedings of a NATO Advanced Study Institute on Map Data Processing held in Maratea, Italy, June 18-29, 1978 (New York:Academic Press, Inc.).

Gardiner, V. (1977), "On Generalizations Concerning Generalization," *The Cartographic Journal*, 14(2):135-136.

_____. (1982), "Stream Networks and Digital Cartography," *Cartographica*, 19(2):38-44.

Goodchild, Michael F. (date unknown), "Fractals and the Accuracy of Geographic Measures," *International Association for Mathematical Geology Journal*, 12(2):85-98.

_____. (1978), "The Effects of Generalization in Geographical Data Encoding," in Freeman, Herbert and Goffredo G. Pieroni, eds. (1978). Map Data Processing: Proceedings of a NATO Advanced Study Institute on Map Data Processing held in Maratea, Italy, June 18-29, 1978 (New York:Academic Press, Inc.) pp.191-204.

Gordon, W.J. and Riesenfeld, R.F. (1974), "Berstein-Bezier Methods for the Computer Design of Free Form Curves and Surfaces," *Journal of Association for Computer Machinery*, 2(2):293-310.

Gottschalk, Hans-Jorg (1971), "Smoothing, Reduction, and Generalization of Data of Digitized Line Elements," International Cartographic Association, Commission 3, *Automation of Cartography*, pp. 1-17.

_____. (1973), "The Derivation of a Measure for the Diminished Content of Information of Cartographic Line Smoothed by Means of a Gliding Arithmetic Mean," *Informations Relative to Cartography and Geodesy*, Translations, Number 30:11-16.

_____. (1974a), "The Derivation of a Measure for the Diminished Content of Information of Cartographic Line Smoothed by Means of a Gliding Arithmetic Mean," in Erno Csati, Ed., *Automation the New Trend in Cartography*, (Budapest, Hungary:The Geocartotechnical Research Department), pp. 61-65.

_____. (1974b), "Automatic Generalization of Settlements, Traffic Lines, Contours, Drainage, and Vegetation Boundaries For a Small Topographic Map," *Proceedings*, Seventh International Conference on Cartography, Madrid, Spain, pp. 9-14.

_____. (1978), "Data Generalization and Attribute Code Schemes," *Proceedings*, Third International Conference on Computer-Assisted Cartography, AUTO-CARTO III, San Francisco, California, pp. 219-226.

Hajek, Milan (1972), "Automation and the System of Cartographic Generalization: Problems," *International Cartographic Association 6th Technical Conference*, Montreal and Ottawa, August 16-25.

Hajek, Milan, Mitasova, Irena, and Sipos, Jan (1974), "The Problem of Analytical Cartographic Generalization," in Erno Csati, Ed., *Automation the New Trend in Cartography*, (Budapest, Hungary:The Geocartotechnical Research Department), pp. 177-186.

Hanan, Maurice and J.M. Kurtzberg (1972), "Placement Techniques," *Design Automation of Digital Systems*, 1:213-262.

Harbaugh, J.W., and A. Merriam (1968), Computer Applications in Stratigraphic Analysis, (New York:John Wiley and Sons).

Harris, Katie M. (1981), "Algorithms for Line Simplification and Smoothing and Their Applicability to Geographic Data," Research Paper, Department of Geography-Meteorology, (Lawrence, KS: University of Kansas).

Harrison, C.J. and R.F. Mercu (1979), "Two-Dimensional Recursive Filtering Using the Rotated Filter Design Technique," *Mathematical Geology* 11(6):669-689.

Hershey, A.V. (1963), "The Plotting of Maps on CRT Printer," *Naval Weapons Laboratory Report 1844*, (Dahlgren, Virginia:U.S. Navy).

Holloway, J.L. (1958), "Smoothing and Filtering of Time Series and Space Fields," *Advances in Geophysics*, 4.

Hoffman, Frank (1974), "Mathematical Modelling as a Basis of Cartometrical Analysis and Automated Generalization," in Erno Csati, Ed., *Automation the New Trend in Cartography*, (Budapest, Hungary:The Geocartotechnical Research Department), pp. 53-60.

Horowitz, E. and S. Shani (1978). Fundamentals of Data Structures, (Rockville, MD:Computer Science Press).

Imhof, Eduard (1957), "Generalisierung der Höhenkurven," *Kartographische Studien*, Erganzungband Nr. 264 zu *Petermanns Geographische Mitteilungen*, 89-99.

_____. (1963), "Tasks and Methods of Theoretical Cartography," *International Yearbook of Cartography*, 3:12-25.

_____. (1982), Cartographic Relief Representation. (Berlin:Walter de Gruyter and Co.), 389p.

International Cartographic Association (1973), Multilingual Dictionary of Technical Terms in Cartography, Franz Steiner, Ed., (Wiesbaden, Germany:Verlag GMBH).

Jancaitis, J. and J. Junkins (1973), "Mathematical Techniques for Automated Cartography," *Final Technical Report for Contract DAAK02-72-C-0256*, (Fort Belvoir, Virginia:U.S. Army Engineer Topographic Laboratories).

Jenks, George F. (date unknown), "Generalization in Statistical Mapping," *Annals* of Association of American Geographers, 53(1).

_____. (1975), "Under Generalization and Figure Ground Relationships in Statistical Mapping,"*Proceedings*, Second International Symposium on Computer-Assisted Cartography, AUTO–CARTO II, 21-25 September 1975, (Washington, D.C.:U.S. Department of Commerce, Bureau of the Census and the American Congress on Surveying and Mapping), pp. 149-154.

_____. (1978), "Perceptual Thresholds in Linear Simplification by Computer," Paper Presented to 74th Annual Meeting of the Association of American Geographers.

_____. (1980), "Thoughts On Line Generalization," *Proceedings*, Fourth International Symposium on Cartography and Computing: Applications in Health and Environment, AUTO–CARTO IV, American Congress on Surveying and Mapping and American Society of Photogrammetry, 1:209-221.

_____. (1981), "Lines, Computers and Human Frailties," *Annals* of the Association of American Geographers, 71,(1):1-10.

_____. (1985), "Linear Simplification:How Far Can We Go?," Paper Presented to the 10th Annual Meeting of the Canadian Cartographic Association, Fredericton, New Brunswick, June 1985.

Johannsen, T. (1973), "A Program for Editing and for Some Generalizing Operations (For Derivation of a Small Scale Map From Digitized Data in 1:50,000)", *Informations Relative to Cartography and Geodesy*, Translations, 30:17-22.

_____. (1976), "Automated Procedures With Interactive Editing, Rim-Adaption, Junction-Purging, and Lettering for a Small Scale Topographic Map," *Informations Relative to Cartography and Geodesy*, Translations, 33,:43-55.

Kadman, N. (1972), "Automated Selection of Settlements in Map Generalization," *Cartographic Journal*, 9(2).

Kashyap, R.L. and Oommen, B.J. (1983), "A Scale Preserving Smoothing Technique," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-S.

Keates, J.S. (1973), Cartographic Design and Production. (New York:John Wiley and Sons).

Knorr, H. (1963), "Generalization, Revision, and Automation: Three Fundamental Problems in Cartography. *Nachrichten aus dem Karten- und Vermessungswesen*, Reihe 5, No. 4.

_____. (1969), "Comparison of a Generalization in 1:200,000 Scale, Derived From Topographic Maps, With a Generalization Derived From Aerial Photographs," *Informations Relative to Cartography and Geodesy*, German Contributions in Foreign Languages, 25:5-16.

Koeman, C. and F.L.T. Van der Weiden (1970), "The Application of Computation and Automatic Drawing Instruments to Structural Generalization," *The Cartographic Journal*, 7(1):47-49.

Kristoffersen, E. (1980), "Conflicting Symbol Handling by Microcomputer," in H. Opheim, Ed., *Contributions to Map Generalization Proceedings*, (Oslo, Norway:Norwegian Computing Center), pp. 105-109.

Laidlaw, G.A. (date unknown), "Generalization of Relief Portrayal," *Symposium on Cartography*, pp 18-21.

Lang, T. (1969), "Rules For the Robot Draughtsmen," *The Geographical Magazine*, 42(1):50-51.

_____. (1971), Computer Programs for Mapping, (London:Architectural Press).

Langridge, D. (1972), "On the Computation of Shape," *Frontiers of Pattern Recognition*, (New York:Academic Press, Inc.), S. Watanabe, Ed., pp. 347-365.

Leberl, F.W. (1986), "ASTRA - A System for Automated Scale Transition," *Photogrammetric Engineering and Remote Sensing*, 52(2):251-258.

Liao, Y. (1981), "A Two-Stage Method of Fitting Conic Arcs and Straight Line Segments to Digitized Contours," *Proceedings* of IEEE Pattern Recognition and Image Processing Conference, Dallas, pages 224-229.

Lichtner, Werner (1978), "Locational Characteristics and the Sequence of Computer Assisted Processes of Cartographic Generalization," *Nachrichten aus dem Karten- und Vermessungwegen Reike II*, Ubersetzugen Heft nr.35, pp.65-75.

_____. (1979), "Computer-Assisted Processing of Cartographic Generalization in Topographic Maps," *Geo-Processing*, 1:183-199.

Long, D.E.. (date unknown), "Generalization of Culture in Derived Mapping," *Symposium on Cartography*, pp 12-17.

Loon, J.C. (1978), Cartographic Generalization of Digital Terrain Models, Dissertation Paper, (Ann Arbor, Michigan:University Microfilms International), 199p.

Lundquist, Gösta (1959a), "Generalization - A Preliminary Survey of an Important Subject," *Nachrichten aus dem Karten- und Vermessungswesen: Reihe II*: Deutsche Beitrage in fremden Sprachen, heft nr. 3:46-51.

_____. (1959b), "Generalization—A Preliminary Survey of an Important Subject," *The Canadian Surveyor*, 14(10):466-470.

MacKinnon, Ross D. and G.M. Barber (1970), "A New Approach to Network Generalization and Map Representation: the Linear Case of Allocation Problem," *Geographical Analysis*, 2:156-168.

Maling, D.H. (1963), "Some Quantitative Ideas About Cartographic Generalization," Bulletin No. 4, International Cartographic Association, Sonderdruck from *Nachrichten aus dem Karten und Vermessungswesen*, 5(5):6-22.

_____. (1968), "How Long is a Piece of String?", *Cartographic Journal*, 5,:147-156.

Mandelbrot, B.B. (1977), Fractals: Form, Chance, and Dimension. (San Francisco:W.H. Freeman).

Marino, Jill S. (1978), "Characteristic Points and their Significance in Cartographic Line Generalization, unpublished Masters Thesis, Department of Geography-Meteorology, The University of Kansas, 92p.

McCullagh, M.J. (1981), "Creation of Smooth Contours of Irregularly Distributed Data Using Local Surface Patches," *Geographical Analysis*, 13:51-62.

McEwen, R.B. and H.W. Caulkins (1982), "Digital Cartography in the USGS National Mapping Division," *Cartographica*, 19:11-26.

McMaster, Robert B. (1982), "An Examination of Line Simplification Algorithms," Paper presented at the Annual Meeting, Association of American Geographers, San Antonio, TX.

_____. (1983a), "A Mathematical Evaluation of Simplification Algorithms," *Proceedings*, Sixth International Symposium on Automated Cartography, Automated Cartography: International Perspectives on Achievements and Challenges, AUTO-CARTO VI, held in Ottawa, Canada, 16-21 October 1983. Barry S. Wellar, Ed., (Ontario:The Steering Committee Sixth International Symposium on Automated Cartography), II, pp. 267-276.

_____. (1983b), "A Quantitative Analysis of Mathematical Measures in Linear Simplification," unpublished Ph.D. dissertation, Department of Geography-Meteorology, The University of Kansas, 1983.

_____. (1986), "A Statistical Analysis of Mathematical Measures for Linear Simplification," *The American Cartographer*, 13(2):103-116.

Miller, O.M. and Robert J. Voskuil (1964), "Thematic Map Generalization," *Geographical Review*, 54(1):13-19.

Moellering, H. and J.N. Rayner (1982), "The Dual Axis Fourier Shape Analysis of Closed Cartographic Forms," *The Cartographic Journal*, 19(1).

Monmonier, Mark S. (1982). Computer-Assisted Cartography: Principles and Prospects. (Englewood Cliffs,NJ:Prentice-Hall, Inc.), 214p.

Monmonier, Mark Stephen (1983), "Raster-Mode Area Generalization for Land Use and Land Cover Maps," *Cartographica*, vol. 20, no.4, pp. 65-91.

Montagano, G.A. (date unknown), "Generalization of Hydrographic Features," *Symposium on Cartography*, pp 8-11.

Montanari, U. (1970), "A Note on Minimal Length Polygon Approximation to a Digitized Contour," *Communications of the ACM*, 13:41-47.

Morrison, Joel L. (1974), "A Theoretical Framework for Cartographic Generalization with Emphasis on the Process of Symbolization," *International Yearbook of Cartography*, 13:59-67.

_____. (1975), "Map Generalization: Theory, Practice, and Economics," *Proceedings*, Second International Symposium on Computer-Assisted Cartography, AUTO-CARTO II, 21-25 September 1975, (Washington, D.C.:U.S. Department of Commerce, Bureau of the Census and the American Congress on Surveying and Mapping), pp.99-112.

Muller, Jean-Claude (1983), "Visual Versus Computerized Seriation: The Implications for Automated Map Generalization," *Proceedings*, Sixth International Symposium on Automated Cartography, Automated Cartography: International Perspectives on Achievements and Challenges, AUTO-CARTO VI, held in Ottawa, Canada, 16-21 October 1983. Barry S. Wellar, Ed., (Ontario:The Steering Committee Sixth International Symposium on Automated Cartography), pp. 277-287.

Neumann, Joachim (1973), "Begriffsgeschichte und Definition des Begriffs Kartographische Generalisierung," *International Yearbook of Cartography*, 13:59-67.

Nickerson, Bradford G. and Herbert R. Freeman (1986), "Development of a Rule-based System for Automatic Map Generalization," Proceedings, Second International Symposium on Spatial Data Handling, Seattle, Washington, July 5-10, 1986, (Williamsville, N.Y.:International Geographical Union Commission on Geographical Data Sensing and Processing), pp. 537-556.

Noton, David and Lawrence Stark (1971), "Eye Movements and Visual Perception," *Scientific American*, 224(6):34-43.

Oommon, B.J. and R.L. Kashyap (1983), "Scale Preserving Smoothing of Islands and Lakes," *Proceedings*, Sixth International Symposium on Automated Cartography, 1:243-251.

Opheim, Harold (1963), "Principles of Generalization in Cartography with Special Reference to Contours," *Nachrichten aus dem Karten- und Vermessungswesen*, 5.

_____. (1980), "A New Method for Data Reduction of a Digitized Curve," *Norwegian Computing Center Publication*, Number 676, 44p.

_____. (1981), "Smoothing a Digitized Curve by Data Reduction Methods," *EUROGRAPHICS '81*, J.L. Encarnaco, Ed., (Amsterdam: North-Holland Publishing Company), pp. 127-135.

_____. (1982), "Fast Reduction of a Digitized Curve," *Geo-Processing*, 2:33-40.

Pannekoek, A.J. (1962), "Generalization of Coastlines and Contours," *International Yearbook of Cartography*, 2:55-75.

Pavlidis, T. (1974), "The Use of Algorithms of Piecewise Approximation for Picture Processing Applications," *Informal Proceedings of a Conference on Mathematical Software*, ACM/SIAM, Purdue University, pp. 130-159.

_____. (1977), Structural Pattern Recognition, (New York:Springer-Verlag), 302p.

_____. (1978), "A Review of Algorithms for Shape Analysis," *Computer Graphics and Image Processing*, 7: 243-258.

_____. (1982), "Curve Fitting as a Pattern Recognition Problem," *Proceedings*, Sixth International Conference on Pattern Recognition, Munich, pp. 853-858.

_____. (1983), "Curve Fitting with Conic Splines," *ACM Transactions on Graphics*, 2:1-31.

Pavlidis, T. and S.L. Horowitz (1973), "Piecewise Approximation of Plane Curves," *Proceedings*, First International Joint Conference on Pattern Recognition, pp. 396-405.

Perkal, Julian (1965a), "An Attempt at Objective Generalization," Translated by W. Jackowski from Julian Perkal, "Proba obiektywnej generalizacj," *Geodezja i Kartografia*, Tom VI, Zeszyt 2 (1958), pp 130-142. In Nystuen, John (ed.), *Michigan Inter-University Community of Mathematical Geographers*, Discussion Paper 10, (Ann Arbor, Michigan:University of Michigan), 1966.

_____. (1965b), "On the Length of Empirical Curves," Translated by W. Jackowski from Julian Perkal, "O Dlugosci Krzywych Empiriycznych," *Zastosowania Matematyki*, III, Zeszyt 2 (1958), pp 257-286. In Nystuen, John (ed.), *Michigan Inter-University Community of Mathematical Geographers*, Discussion Paper 10, (Ann Arbor, Michigan:University of Michigan), 1966.

Peucker, Thomas K. (1973), "Four Methods for Quantitative Settlement Selection," *(unpublished)*.

_____. (1975), "A Theory of the Cartographic Line," *Proceedings*, Second International Symposium on Computer-Assisted Cartography, AUTO-CARTO II, September 21-25, 1975 (Washington, D.C.:U.S. Dept. of Commerce, Bureau of Census and ACSM, Cartography Division), pp. 508-518.

Philbrick, Allen K. (1953), "Towards a Unity of Cartographical Forms and Geographical Content," *The Professional Geographer*, 5(5):11-15.

Poiker, Thomas K., R. Squirrell, and Shun-En Xie (1982), "The Use of Computer Science and Artificial Intelligence in Cartographic Design," *Proceedings*, Fifth International Symposium on Computer-Assisted Cartography and International Society for Photogrammetry and Remote Sensing Commission IV: Cartographic and Data Bank Application of Photogrammetry and Remote Sensing, Environmental Assessment and Resource Management, AUTO-CARTO V, held in Crystal City, Virginia, 22-28 August 1982. Jack Foreman, Ed., (Falls Church, VA:American Society of Photogrammetry and American Congress on Surveying and Mapping).

Ramer, Urs (1972), "An Iterative Procedure for the Polygonal Approximation of Plane Curves," *Computer Graphics and Image Processing*, 1:244-256.

Ratajski, Lech (1967), "Phenomenes des points de generalisation," in Konrad Frenzel, Ed., *International Yearbook of Cartography*, 7:143-151.

Raudseps, J.C. (1975), "Some Aspects of the Tangent-Angle Versus Arc Length Representation of Contours," Report 1801-6, ASTIA AD 462 877, (Columbus, Ohio:Ohio State University Research Foundation).

Reumann, K., and A.P.M. Witkam (1974), "Optimizing Curve Segmentation in Computer Graphics," *International Computing Symposium*, (Amsterdam: North-Holland Publishing Company), pp.467-472.

Rhind, David W. (1973). "Generalization and Realism within Automated Cartographic Systems," *The Canadian Cartographer*, 10(1):51-62.

Robinson, Arthur H., Randall Sale, and Joel L. Morrison. (1978). Elements of Cartography, Fourth Edition, (NY:John Wiley and Sons, Inc.).

Rogers, D.F. and J.A. Adams (1976), "Mathematical Elements for Computer Graphics, (NY:McGraw-Hill Book Co.), 239p.

Rosenberg, B. (1972), "The Analysis of Convex Blobs," *Computer Graphics and Image Processing*, 1:183-192.

Rosenfeld, A. and E. Johnston (1973), "Angle Detection on Digital Curves," *IEEE Transactions on Computers*, C-22:875-878.

Rosenfeld, A. and J. Weszka (1974), "An Improved Method of Angle Detection on Digital Curves," *IEEE Transaction on Computers*, C-24,:940-941.

Rutkowski, W.S. (1981), "Shape Segmentation Using Arc/Chorc Properties," *Computer Graphics and Image Processing*, 17:114-129.

Salichtchev, Konstantin A. (1976), "History and Contemporary Development of Cartographic Generalization," *International Yearbook of Cartography*, 16:158-172.

Saloritskiy, B.V. (1974), "Some Possibilities for Automatic Generalization of Outlines," *Geodesy, Mapping, and Photogrammetry*, 16:187-189.

Sampson, P.D. (1982), "Fitting Conic Sections to Very Scattered Data: An Iterative Refinement to the Bookstein Algorithm," *Computer Graphics and Image Processing*, 18:97-108.

Sankar, P.V. and C.U. Sharma (1978), "Parallel Procedure for the Detection of Dominant Points on a Digital Curve," *Computer Graphics and Image Processing*, 7:403-412.

Sarvarayodu, G.P.F., and I.K. Sethi (1983), "Walsh Descriptors for Polygonal Curves," *Pattern Recognition*, 16(3):327-336.

Schittenhelm, R. (1976), "The Problem of Displacement in Cartographic Generalization Attempting a Computer Assisted Solution," *Informations Relative to Cartography and Geodesy*, Translations, 33:65-74.

Selden, David D. and Michael A. Domaratz (1983), "Digital Map Generalization and Production Techniques," *Proceedings*, Fifth International Symposium on Computer-Assisted Cartography and International Society for Photogrammetry and Remote Sensing Commission IV: Cartographic and Data Bank Application of Photogrammetry and Remote Sensing, Environmental Assessment and Resource Management, AUTO-CARTO V, held in Crystal City, Virginia, 22-28 August 1982. Jack Foreman, Ed., (Falls Church, VA:American Society of Photogrammetry and American Congress on Surveying and Mapping), pp. 241-248.

Shea, K. Stuart (1987a), "Cartographic Data Entry Through Automatic Feature Tracking," *Proceedings*, Eighth International Symposium on Computer-Assisted Cartography, AUTO-CARTO VIII, Nicholas R. Chrisman, ed., Baltimore, MD, March 30 - April 2, 1987, pp. 660-669.

Shea, K. Stuart (1987b, currently under review, *The American Cartographer*), "Dot Size Differentiation and Region Perception."

Sklansky, J. and V. Gonzalez (1980), "Fast Polygon Approximation of Digitized Curves," *Pattern Recognition*, 12:327-331.

Srnka, Erhart (1970), "The Analytical Solution of Regular Generalization in Cartography," *International Yearbook of Cartography*, 10:48-60.

_____. (1974), "Mathematico-Logical Models in Cartographic Generalization," in Erno Csati, Ed., *Automation the New Trend in Cartography*, (Budapest, Hungary:The Geocartotechnical Research Department), pp. 45-52.

Stegnena, Lajos (1974), "Tools for the Automation of Map Generalization: The Filter Theory and the Coding Theory," in Erno Csati, Ed., *Automation the New Trend in Cartography*, (Budapest, Hungary:The Geocartotechnical Research Department), pp. 66-95.

Steward, H.J. (1974), "Cartographic Generalization, Some Concepts and Explanation," *The Canadian Cartographer*, Supplement, 11:77p.

Strahler, A.N., (1952), "Hypsometric (area-altitude) Analysis of Erosional Topography," *Bulletin of Geographical Society of America*, 69:279-300.

Sukhov, V.I. (1970), "Application of Information Theory in Generalization of Map Contents," *International Yearbook of Cartography*, 10:48-62.

Tai, H.T., C.C. Li, and S.H. Chiang (1982), "Application of Fourier Shape Descriptors to Classification of Fine Particles," *Proceedings*, Sixth International Conference on Pattern Recognition, Munich, Germany, pp. 748-751.

Taketa, R.A. (1979), Structure and Meaning in Map Generalization, Dissertation Paper, (Ann Arbor, Michigan:University Microfilms International), 207 Pages.

Tamigochi, R., M. Yokota, E. Kawaguchi, T. Tamati (1982), "Picture Understanding and Retrieving System of Weather Charts," *Proceedings*, 6th International Conference on Pattern Recognition, Munich, Germany, 1982, pp. 803-805.

Taylor, D.R. Fraser, Ed. (1980), The Computer in Contemporary Cartography. Volume I. Progress in Contemporary Cartography, (Chicester: J. Wiley and Sons), 252p.

Tobler, Waldo R. (1964), "An Experiment in the Computer Generalization of Maps," Technical Report No. 1, *Office of Naval Research Task No. 389-137*, Contract No. 1224(48), Office of Naval Research, Geography Branch, 35p.

_____. (1965), "Automation in the Preparation of Thematic Maps," *The Cartographic Journal*, 2(1):32-38.

_____. (1966), "Numerical Map Generalization," In John Nystuen, Ed., *Michigan Inter-University Community of Mathematical Geographers*, Discussion Paper 8, (Ann Arbor, Michigan:University of Michigan), 25 p.

Töpfer, F. (1962). "Das Wurzelgesetz und seine Anwendung bei der Reliefgeneralisierung," *Vermussungstechnik*, 10(2):37-42.

_____. (1963). "Untersuchungen zum Anwendungsberich des Wurzelgesetzes bei Kartographischen Generalisierung," *Vermessungstechnik*, 11(5):179-186.

Töpfer, F. and W. Pillewizer (1966). "The Principles of Selection, A Means of Cartographic Generalisation," *Cartographic Journal*, 3(1):10-16.

Traylor, Charles T. (1979). "The Evaluation of a Methodology to Measure Manual Digitization Error in Cartographic Data Base," unpublished Ph.D. Dissertation, Department of Geography-Meteorology, The University of Kansas.

United Nations (1964), "Generalization and Omission of Detail: an Economic Approach to Chart Maintenance Problems," *Proceedings*, Fourth UN Regional Cartographic Conference for Asia and the Far East, II: 242-249.

Van Horn, E.K. (1985), "Generalizing Cartographic Data Bases," *Proceedings*, Seventh International Symposium on Automated Cartography, Digital Representation of

Spatial Knowledge, AUTO-CARTO VII, held in Washington, D.C., 11-14 March 1985. (Falls Church, VA:American Society of Photogrammetry and the American Congress on Surveying and Mapping), pp. 532-541.

Vanicek, P. and D.F. Woolnough (1973), "A Program Package for Packing and Generalizing Digital Cartographic Data," Technical Report Number 23, Department of Surveying Engineering, University of New Brunswick.

_____. (1975), "Reduction of Linear Cartographic Data Based on Generalization of Pseudo-Hyperbolae," *The Cartographic Journal*, 12(2):112-119.

Watson, W.C. (1970), "A Study of the Generalization a Small-Scale Map Series," *International Yearbook of Cartography*, 10:24-32.

Weber, W. (1980), "An Information Science Approach to Map Generalization," in H. Opheim, Ed., *Contributions to Map Generalization Proceedings*, (Oslo, Norway:Norwegian Computing Center), pp. 31-52.

Wertheimer, M. (1958), "Principles of Perceptual Organization," in Readings in Perception, D. Beardsley and M. Wertheimer, Eds. Princeton, N.J.:Van Nostrand.

White, Ellen R. (1983), "Perceptual Evaluation of Line Generalization Algorithms," unpublished Masters Thesis, Department of Geography, Virginia Polytechnic Institute and State University, 133p.

Wright, John K. (1942), "Map Makers Are Human," *Geographical Review*, 32:527-544.

Wofendale, P.C.F. (1967), "Machine Accuracies in Automatic Cartography," *The Cartographic Journal*, 4(1):24-28.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B—Software Overview

B.1 Introduction

As part of the CARTOGEN program, PGSC designed and developed a **Line Simplification Shell** to support the test and performance of various software algorithms for line simplification, smoothing, and measurement. The shell allows for cycles of simplification, smoothing, and measurement to be performed (and their results evaluated) repeatedly until the **Quit** option is selected. It was developed and programmed in C and consists of 2783 lines of executable code.[1] The Shell runs on Sun workstations under SunView's windowing environment.

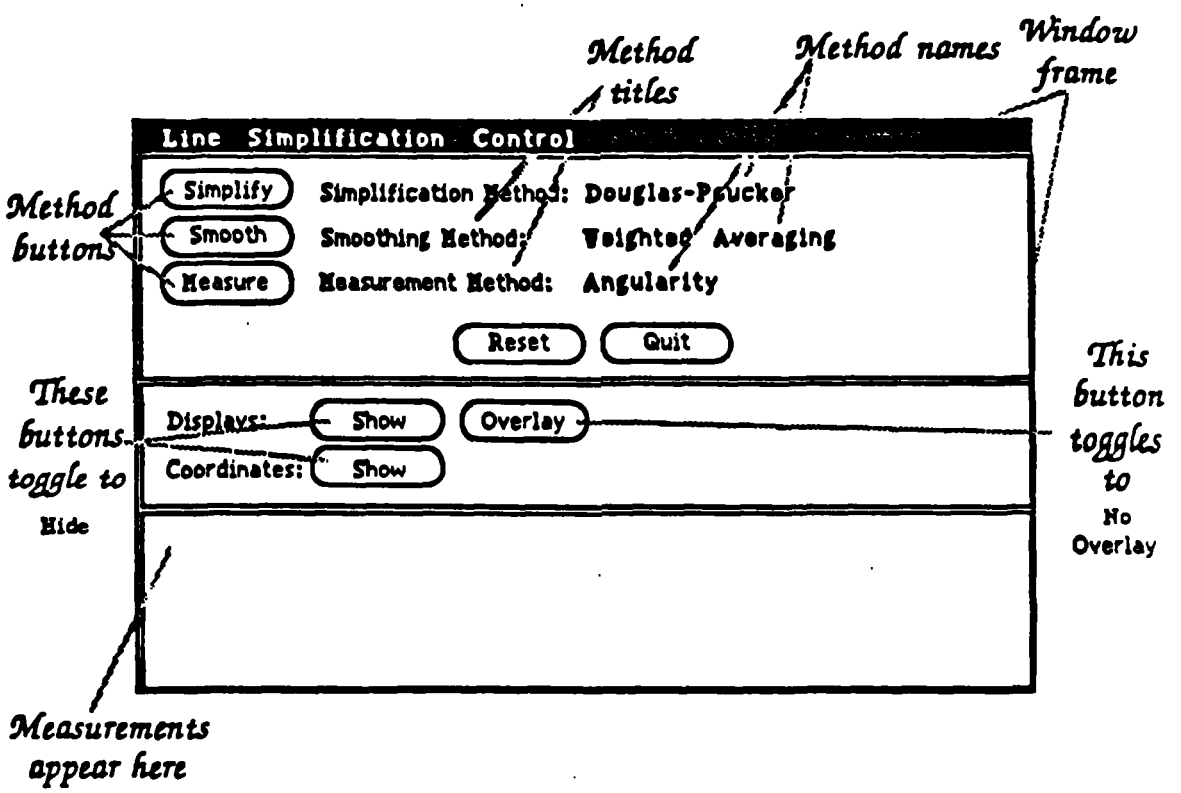
The shell is invoked by entering the `linesimp` command at the Unix™ C shell prompt. After initialization, the shell's icon is displayed in the upper left corner of the screen. "Opening" this icon by selecting it with the left (*selection*) mouse button brings up the shell's **Control Panel**, which allows for the interactive selection of the shell's various algorithms and options. The Control Panel is explained in detail in Appendix B.2 of this report. The shell also provides a **Display Window** to graphically present the original (input) line and the modified (output) line. The input line can be drawn from scratch using the mouse. This window is further explained in Appendix B.3 of this report. Finally, a **Coordinates Window** is provided for actual line coordinate data, which can be examined and saved to a file and, in the case of the input line, entered manually or loaded from an existing file. This window is fully explained in Appendix B.4 of this report.

B.2 Control Panel

The basic event flow for the Line Simplification shell is as follows:

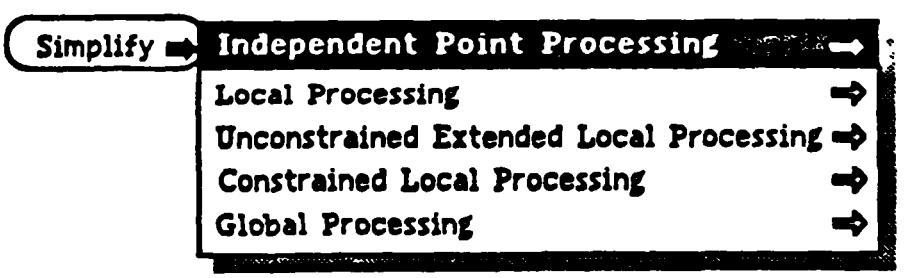
- Select or verify the coordinate data for algorithm execution. If desired, enter or load new data into the input coordinates side;
- On the Control Panel, select an algorithm to be performed for *simplification*, *smoothing*, or *measurement*;
- Execute the algorithm and examine its results.

The following figure explains the Control Panel.



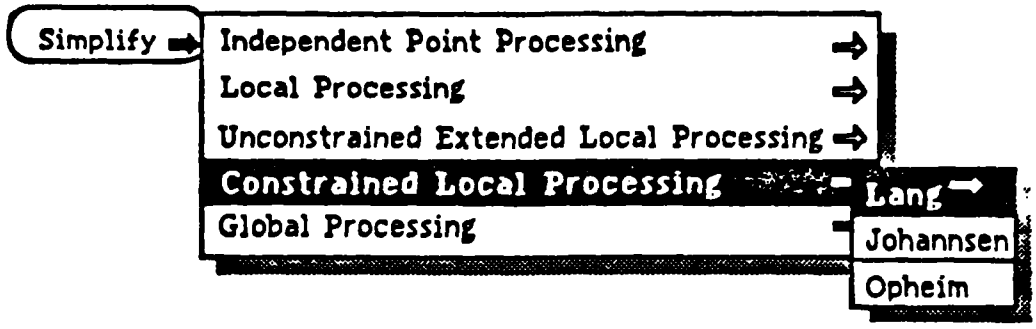
B.2.1 Algorithm Selection Using Pop-up Menus

The currently selected methods for simplification, smoothing, and measurement are displayed in bold face on the control panel. To change an individual setting, bring the mouse pointer to point to any of the following: the method *button* (shown in oval on the left), *title* (in the middle), or current *name* (in boldface on the right). Then press and hold the right (*menu*) mouse button, which pops up the top-most menu for that method set (see figure below).



Any item on the menu which has a *right arrow* at its right indicates the existence of a subordinate menu for that item. These "pull-right" menus are then displayed by—while

still holding down the mouse menu button—moving the mouse pointer over to the right arrow until the pull-right menu pops up, as in the figure below.



These subordinate menus may have pull-rights of their own, allowing an extensive hierarchy to be displayed in a neat and concise manner without permanently tying up screen space (this is why these menus are often referred to as "walking" menus for the step-like method used to display them).

Actual algorithm names are at the bottom end of the menu structure, and are separated by lines within their menus. Once the desired algorithm is selected (highlighted) within this bottom-level menu, releasing the mouse menu button now sets the actual algorithm name for that method on the Control Panel. This remains in effect until explicitly changed later. This menu selection process is similar for all menus in the SunView environment.

There is a menu structure for each of *simplification*, *smoothing*, and *measurement*, independent from each other; that is, setting the simplification method has no effect on the smoothing method, etc.

B.2.2 Executing the Desired Algorithm

Once set using the walking menus, the desired algorithm is executed by pointing to the method button (in the oval) and selecting it with the mouse. Again, executing one has no effect on the other methods; that is, selecting the **Simplify** button strictly performs the simplification with no smoothing and no measurements. In the case of **Simplify** or **Smooth**, the results are displayed, after automatically clearing any previous output, on the Display and Coordinates Windows' output sides. Some algorithms require user input for parameter values, such as a *tolerance*. If so, a "pop-up" window is displayed to allow entry

of the required value. Then select the **OK** button to continue. In the case of **Measure**, the results are shown at the bottom of the Control Panel.

B.2.3 **Reset** and **Quit** Buttons

Selecting the **Reset** button resets the three method names to their respective default. Selecting the **Quit** button exits the shell.

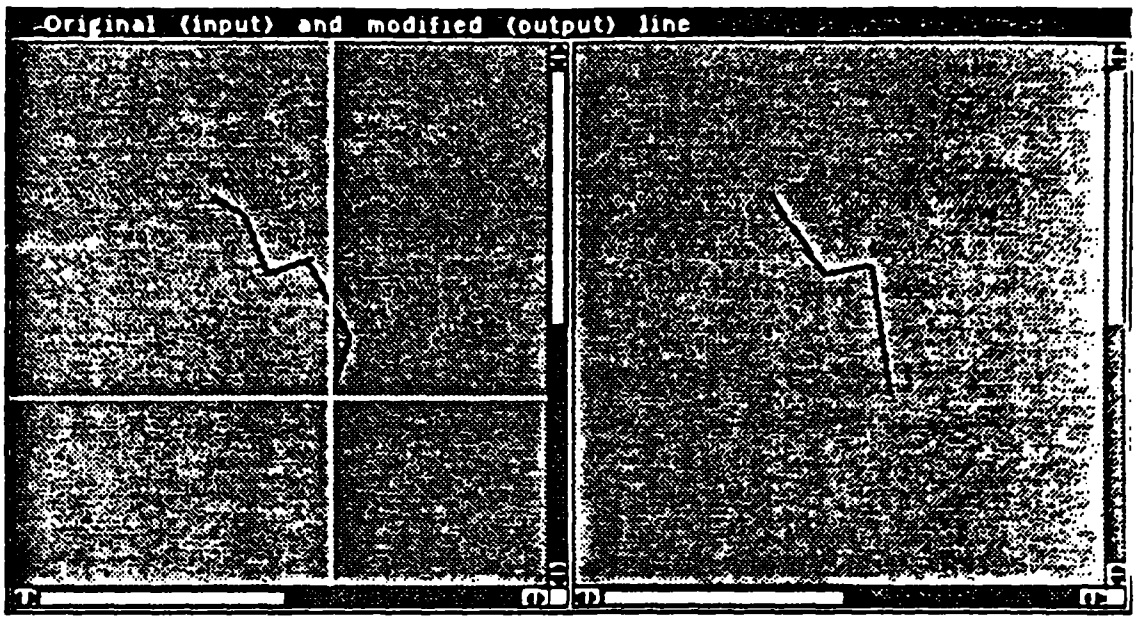
B.2.4 **Show/Hide** and **Overlay/No Overlay** Buttons

If either the Display or the Coordinates Window is not visible on the screen, selecting the appropriate **Show** button on the Control Panel displays that window. Once either window is shown, its **Show** button then toggles to **Hide**, so you can control what is being displayed on the screen at any time. Of course, these windows behave in otherwise standard SunView fashion by providing the standard "window" menu shown when pressing the mouse menu button anywhere along the window's frame.

The **Overlay** button is provided to allow you to examine the modified line overlaid on top of the original line. Once the lines are overlaid, this button toggles to **No Overlay**, allowing you to reset the input display.

B.3 The Displays Window

As mentioned above, this window allows graphical representations for the original and modified lines. This window is made up of two "canvases" on which the lines are drawn, each of which is fully scrollable. Currently, they default to 1000 x 1000 pixels in size, partially shown in a window 500 x 500 pixels wide. These parameters are, however, easily modified. An example of the Display Window is shown below.



The cursor here is made up of two fine "cross-hairs," shown here in white, which allows precise alignment while creating input lines.

B.3.1 Creating an Input Line

In addition to loading an existing line from a standard Unix file, you can create or modify the current input side of the display by using the mouse.[2] Simply clicking the left, or selection, mouse button anywhere in the input canvas creates a new coordinate. Creating new coordinates then draws vectors between each pair of coordinates, building the line segment by segment. You can also "drag" the mouse (while holding the mouse selection button down) creating a sequence of more finely spaced coordinates. This method allows a smoother line shape, but creates coordinates more rapidly. Notice that each line point is drawn enlarged and of a different color than the line segments. These colors can be controlled as explained below.

B.3.2 Using the Display Menu

There is a menu shared between the left and right sides of the Display Window. This menu is shown by pressing and holding the mouse menu button within the borders of

either display. The available options, which generally apply to the side where the menu is popped up, are:

- **Clear**
- **Zoom**
- **Color**

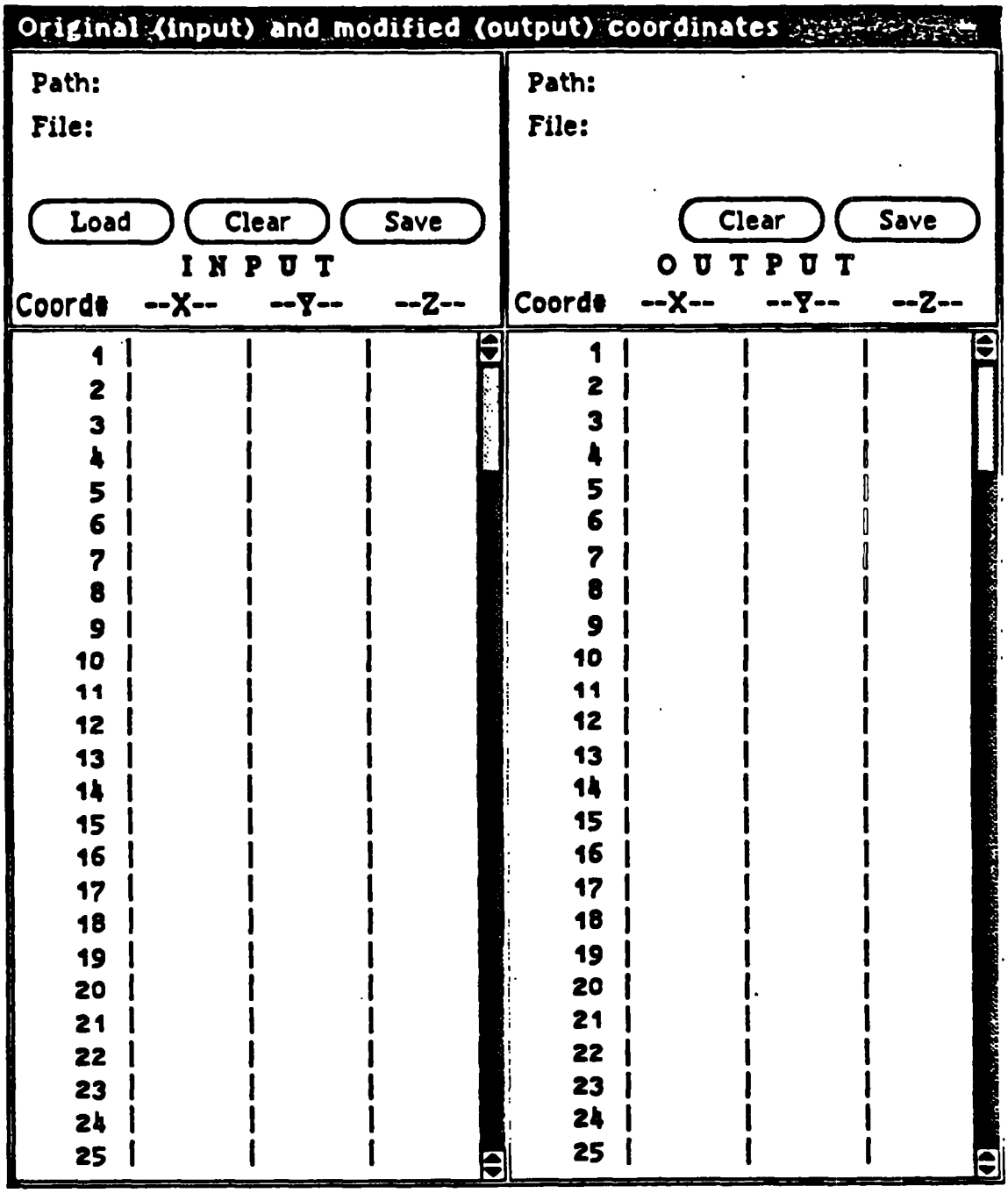
The **Clear** option simply clears the appropriate canvas, along with its corresponding coordinate matrix in the Coordinates Window. If selected on the input side, the output side is additionally cleared.

The **Zoom** option allows the capability to zoom in on a small area of the canvas, enlarging each pixel in that area. This option does not presently exist in the software, but, if it did, could provide for several levels of zooming—such as 2X, 4X, 10X, etc.

The **Color** option has a pull-right menu which controls the color of **Points**, **Lines**, **Areas**, **Foreground**, and **Background**, each of which has a pull-right for selecting from a set of actual display colors. Note that choosing a color for the foreground (cross-hairs and scrollbars) and background applies to both canvases simultaneously.

B.4 The Coordinates Window

The Coordinates Window displays the X, Y, and Z coordinate values for each point on the input and output lines. The Z coordinate is provided to accomodate data which contains the third dimension. Be aware however that some algorithms are not affected by the Z values! Each side of the window contains a file and button panel at the top, and a spreadsheet-like matrix for the actual coordinates below. The Coordinates Windows consist of two parts: (1) the File Panels; and (2) the Coordinate Matrices. An example of the Coordinates Window is illustrated below.



B.4.1 The File Panels

You can **Save** and, for the input side, **Load** the coordinate data using standard Unix files. The **Load** button is shown only when the input side is empty, while the **Save** buttons are shown only when the appropriate side contains coordinate data. There is also a

Clear button, provided for each side when data exists, duplicating the function of the **Clear** menu choice described in Appendix B.3 above.

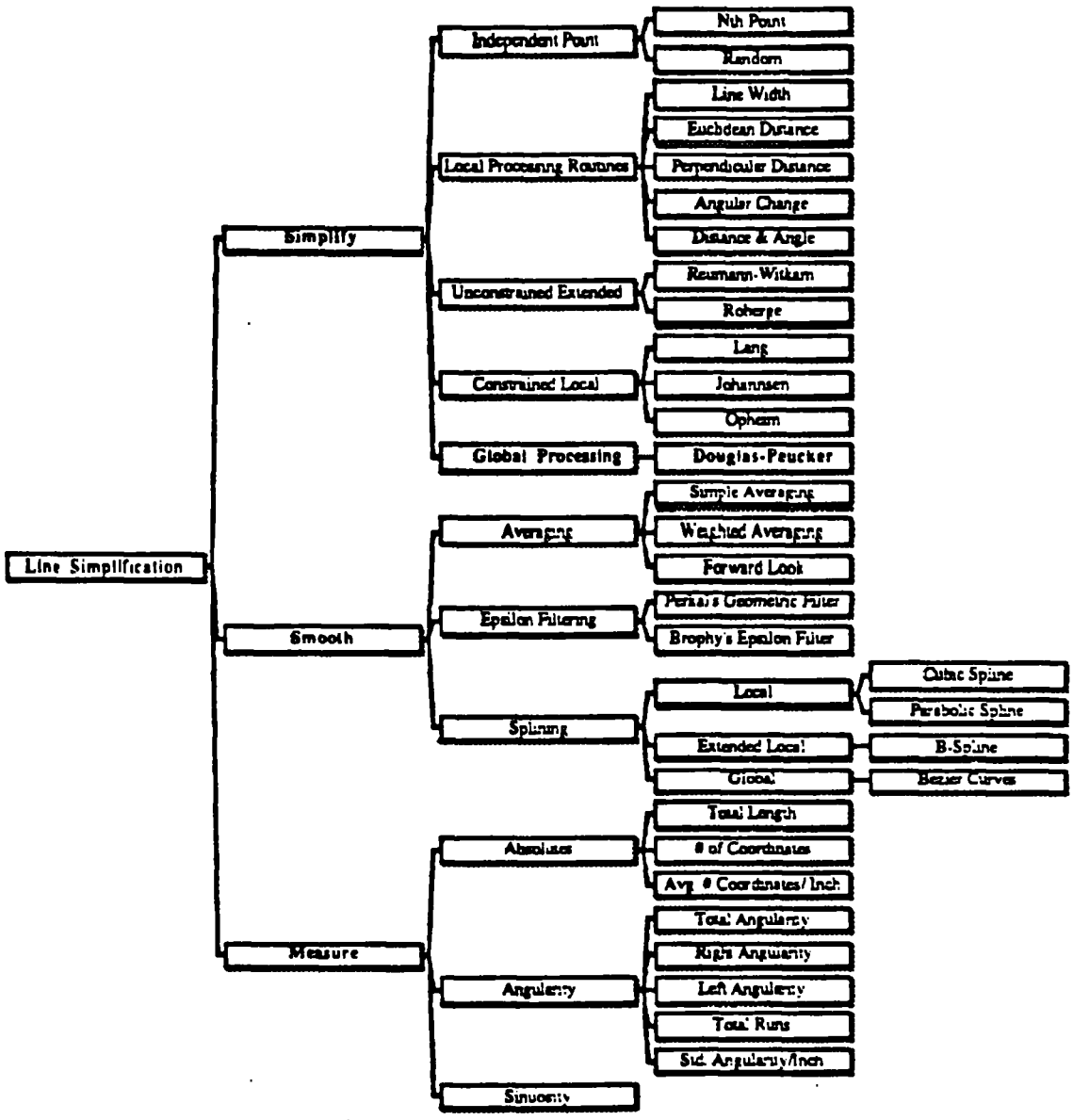
In order to load or save data, the appropriate file name (and full or partial Unix path name, if applicable) must be entered in the spaces provided. This must be an existing file for the input **Load** option, while it may be a new or existing file for the **Save** option. A short message is displayed below the file name indicating the status of the **Load** or **Save** action when the appropriate button is selected with the mouse.

B.4.2 The Coordinate Matrices

Each coordinate matrix simply displays the X, Y, and Z values for the lines, if any, shown in the Display Window. In addition, the input matrix can be used as a standard spreadsheet to enter coordinate data (this is why the cursor takes the shape of a cross in this panel). The current cell contains a blinking caret, indicating where data entered from the keyboard will be applied. A value is entered by typing it and hitting Tab or Return, advancing the caret to the next cell in the matrix. You can also use Shift-Tab or Shift-Return to "backspace" the caret to the previous cell in the matrix. Finally, you can use the mouse cursor by pointing at and selecting any cell in the matrix, thus making it the current cell. Note also that each coordinate matrix can be scrolled individually in the vertical direction. As each set (X-Y-Z) of coordinates is entered, the line segment corresponding to the line between the new point and the last is drawn on the input canvas.

B.5 Current System Implementation

The software shell currently includes menu support to Simplify, Smooth, and Measure lines. The following figure illustrates the overall menu structure with those items that are currently implemented, shaded.[3]



B.6 Conclusion

The Line Simplification Shell provides NOS with a cohesive testbed environment in which to design, implement, and evaluate linear simplification and smoothing algorithms for application to nautical charting data. This tool provides a platform from which an intelligent assessment can be made of the performance of generalization algorithms and their applicability to NOS products.

B.7 Endnotes

[1]The source code is included at the end of section B.7 of this report.

[2]An existing line may have been previously digitized or entered manually.

[3]It should be noted that the breakout of smoothing algoritms does not parallel that which is presented in section 2.2.3.4 (Feature Smoothing) of this report. During the evolutionary process of developing this software shell, an initial breakout was used to prepare the menu structure. After some further work in the generalization study effort, this decomposition was updated and is reflected so in the text. The software, however, remains in the original form.

```
/*=============================================================================*
 * Line simplification (generalization) process shell
 * by Yvon Perreault, PAR Government Systems Corp.
 * April-May 1987.
 *=============================================================================*/

#include <suntool/sunview.h>
#include <suntool/panel.h>
#include <suntool/canvas.h>
#include <suntool/scrollbar.h>
#include <stdio.h>
#include <math.h>

/*
 * Simplification menu constants
 */
#define SIMP_NTH_PT     11
#define SIMP_RANDOM_PT  12
#define SIMP_LINE_WIDTH 21
#define SIMP_EUCLIDEAN  22
#define SIMP_PERPENDIC  23
#define SIMP_ANGULAR    24
#define SIMP_DIST_ANGLE 25
#define SIMP_REUMAN     31
#define SIMP_ROBERGE    32
#define SIMP_LANG       41
#define SIMP_JOHANNSEN  42
#define SIMP_OPHEIM     43
#define SIMP_DOUGLAS    51

/*
 * Smoothing menu constants
 */
#define SMOO_SIMPLE_AVE 11
#define SMOO_WEIGHT_AVE 12
#define SMOO_FWD_LOOK   13
#define SMOO_PERKALS    21
#define SMOO_BROPHYS    22
#define SMOO_CUBIC_SP   31
#define SMOO_PARAB_SP   32
#define SMOO_B_SPLINE   33
#define SMOO_BEZIER_CUR 34

/*
 * Measurement menu constants
 */
#define MEAS_ABS        2
#define MEAS_ANG        3
#define MEAS_SIN        4

/*
 * Display menu constants
 */
#define DISP_CLEAR      1
#define DISP_ZOOM       2
#define DISP_COLOR      3

/*
 * Control panel constants
 */
#define CONTROL_WIDTH_1 36
#define CONTROL_WIDTH_2 30
#define SIMP_ROW        0
```

```
 64  #define SMOO_ROW          1
 65  #define MEAS_ROW          2
 66  #define BUTTON_ROW        3
 67  #define DISP_ROW          0
 68  #define COOR_ROW          1
 69  #define SIMP_DEFAULT      "Douglas-Peucker"
 70  #define SMOO_DEFAULT      "None"
 71  #define MEAS_DEFAULT      "Absolutes"
 72  #define SIMP_DEFAULT_VALUE SIMP_DOUGLAS
 73  #define SMOO_DEFAULT_VALUE SMOO_NONE
 74  #define MEAS_DEFAULT_VALUE MEAS_ABS
 75  #define MAX_MEASURES      5
 76  #define DPI               87.0      /* rounded # pixels in 1 inch ("Dots Per Inch") */
 77  #define TOL_DEFAULT       "10"
 78
 79  /*
 80   *  Graphics canvases constants
 81   */
 82  #define CANVAS_MAX_X      1000
 83  #define CANVAS_MAX_Y      1000
 84  #define INIT_WIDTH        500
 85  #define INIT_HEIGHT       500
 86
 87  /*
 88   *  Color map constants
 89   */
 90  #define R                 0
 91  #define G                 1
 92  #define B                 2
 93  #define COLOR_MAP_SIZE    8
 94  #define BACKGROUND        0
 95  #define FOREGROUND        1
 96  #define IN_POINT_COLOR    2
 97  #define IN_LINE_COLOR     3
 98  #define IN_AREA_COLOR     4
 99  #define OUT_POINT_COLOR   5
100  #define OUT_LINE_COLOR    6
101  #define OUT_AREA_COLOR    7
102  /*
103   *  Color menu constants
104   */
105  #define WHITE             1
106  #define GREEN             2
107  #define RED               3
108  #define BLUE              4
109  #define YELLOW            5
110  #define CYAN              6
111  #define MAGENTA           7
112  #define BLACK             8
113  #define GRAY              9
114  #define LIGHT_RED         10
115  #define LIGHT_GREEN       11
116  #define LIGHT_BLUE        12
117
118  /*
119   *  Coordinate panel constants
120   */
121  #define MAX_COORDS        250
122  #define COORDS_COLS       32
123  #define COORDS_ROWS       25
124  #define LABEL_LEN         4
125  #define VALUE_LEN         5
126  #define NAME_LEN          25
```

```
127  #define X                0
128  #define Y                1
129  #define Z                2
130
131  /*
132   *  Scrollbar constants
133   */
134  #define VERTICAL_LOC     SCROLL_EAST
135  #define HORIZONTAL_LOC   SCROLL_SOUTH
136  #define BUBBLE_MARGIN    1
137
138      static
139      Frame        control_frame,
140                   coordinate_frame,
141                   display_frame,
142                   tolerance_popup;
143
144      static
145      Panel        control_panel,
146                   bottom_panel,
147                   measurements_panel,
148                   file_in_panel,
149                   file_out_panel,
150                   coord_in_panel,
151                   coord_out_panel,
152                   tolerance_panel;
153
154      static
155      Panel_item   simplify_button,
156                   simplification_method,
157                   current_simplification,
158                   smoothe_button,
159                   smoothing_method,
160                   current_smoothing,
161                   measure_button,
162                   measurement_method,
163                   current_measurement,
164                   reset_button,
165                   quit_button,
166
167                   display_title,
168                   disp_show_hide_button,
169                   disp_overlay_button,
170                   coordinate_title,
171                   coor_show_hide_button,
172
173                   measurement_line[MAX_MEASURES],
174
175                   path_in_item,
176                   file_in_item,
177                   file_in_message,
178                   coor_in_load_button,
179                   coor_in_clear_button,
180                   coor_in_save_button,
181                   input_header,
182                   coord_in_header,
183                   coord_in_label[MAX_COORDS],
184                   coord_in_cell [MAX_COORDS][3],
185                   coord_in_ender[MAX_COORDS],
186
187                   path_out_item,
188                   file_out_item,
189                   file_out_message,
```

```
190                 coor_out_clear_button,
191                 coor_out_save_button,
192                 output_header,
193                 coord_out_header,
194                 coord_out_label[MAX_COORDS],
195                 coord_out_start[MAX_COORDS][3],
196                 coord_out_cell [MAX_COORDS][3],
197                 coord_out_ender[MAX_COORDS],
198
199                 tolerance_text_item,
200                 tolerance_ok_button;
201
202     static
203     Canvas       input_canvas,
204                 output_canvas;
205
206     static
207     Pixwin      * input_pw,
208                 *output_pw;
209
210     static struct
211     pixrect     *simplify_button_image,
212                 *smoothe_button_image,
213                 *measure_button_image,
214                 *reset_button_image,
215                 *quit_button_image,
216                 *show_button_image,
217                 *hide_button_image,
218                 *overlay_button_image,
219                 *no_overlay_button_image,
220                 *load_button_image,
221                 *clear_button_image,
222                 *save_button_image;
223
224     static struct
225     rect        canvas_rect = {0, 0, CANVAS_MAX_X, CANVAS_MAX_Y};
226
227     static
228     Cursor      coord_cursor,
229                 draw_cursor;
230
231     static
232     Menu        simplification_menu,
233                 simp_indep_pt_menu,
234                 simp_local_menu,
235                 simp_uncons_local_menu,
236                 simp_cons_local_menu,
237                 simp_global_menu,
238                 smoothing_menu,
239                 smoo_averaging_menu,
240                 smoo_epsilon_menu,
241                 smoo_splining_menu,
242                 smoo_splining_local_menu,
243                 smoo_splining_extended_menu,
244                 smoo_splining_global_menu,
245                 measurement_menu,
246                 angular_measure_menu,
247                 sinuous_measure_menu,
248                 display_menu,
249                 color_types_menu,
250                 back_color_menu,
251                 color_menu;
252
```

```
253     static
254     Icon            linesimp_icon;
255
256     int             i, j, k,
257                     row = 0,
258                     col = 0,
259                     max_reached = FALSE,
260                     overlaid = FALSE,
261                     input,
262                     choice,
263                     simplification_value,
264                     smoothing_value,
265                     measurement_value,
266                     icoord[MAX_COORDS][3],
267                     ocoord[MAX_COORDS][3];
268
269     static
270     char            * in_format = "%5u %5u %5u",
271                     *out_format = "%5u %5u %5u %c",
272                     *read_mode  = "r",
273                     *write_mode = "w",
274                     *coord_column_header = "Coord#  --X--    --Y--    --Z--";
275
276     static
277     struct          measures {
278        float        total_length,
279                     total_angularity,
280                     right_angularity,
281                     left_angularity,
282                     std_angularity_inch,
283                     num_coordinates,
284                     total_runs;
285                             };
286
287     static struct
288     singlecolor
289                     control_bg_color = {255, 255, 255}, /* white */
290                     control_fg_color = {000, 000, 255}, /* blue */
291                     popup_bg_color   = {255, 255, 255}, /* white */
292                     popup_fg_color   = {255, 000, 000}; /* red */
293
294     static struct
295     colormapseg
296                     cms;
297
298     static struct
299     cms_map
300                     map;
301
302     static
303     unsigned char
304                     RGB[3][COLOR_MAP_SIZE];
305
306     static
307     short           hairs_image[256] = {
308 #include "../cursors/hairs"
309                                         };
310     mpr_static (hairs_pixrect, 16, 16, 1, hairs_image);
311
312     static
313     short           cross_image[256] = {
314 #include "../cursors/cross"
315                                         };
```

```
316     mpr_static (cross_pixrect, 16, 16, 1, cross_image);
317
318     static
319     short       icon_image[256] = {
320 #include "../icons/linesimp"
321                                     };
322     mpr_static (icon_pixrect, ICON_DEFAULT_WIDTH, ICON_DEFAULT_HEIGHT, 1, icon_image);
323
324     /* Internal procedures & functions */
325     double              calc_distance                   ();
326     void                clear_coordinates               ();
327     void                define_menus                    ();
328     void                define_windows                  ();
329     void                do_color_choice                 ();
330     void                do_display_choice               ();
331     void                do_done                         ();
332     void                do_douglas_peucker              ();
333     void                do_measure_absolutes            ();
334     void                do_measure_right_left_ang       ();
335     void                do_measure_standardized_ang     ();
336     void                do_measure_total_ang            ();
337     void                do_measure_total_runs           ();
338     void                do_measure_total_sin            ();
339     void                do_measurement_choice           ();
340     void                do_process                      ();
341     void                do_quit                         ();
342     void                do_reset                        ();
343     void                do_simplification_choice        ();
344     void                do_smoothing_choice             ();
345     void                draw_canvas                     ();
346     void                draw_point                      ();
347     Panel_setting       enter_coord_char                ();
348     void                enter_in_coordinate             ();
349     void                enter_new_point                 ();
350     void                enter_out_coordinates           ();
351     void                file_i_o                        ();
352     void                handle_canvas_event             ();
353     void                locate_item                     ();
354     void                make_color_map                  ();
355     void                ok_button                       ();
356     void                overlay_displays                ();
357     void                set_point_coordinates           ();
358     void                set_color                       ();
359     void                show_button_menu                ();
360     void                show_hide_coordinates           ();
361     void                show_hide_displays              ();
362
```

```
363 /*-----------------------------------------------------------------------*/
364
365 main (argc, argv)
366     int             argc;
367     char            **argv[];
368
369 /*
370  *  Define user interface items (menus, frames, panels, etc.), then begin processing.
371  */
372
373 {
374     define_menus ();
375     define_windows (argc, argv);
376
377     window_set (control_frame, FRAME_CLOSED, TRUE, 0); /* iconic at beginning */
378
379     window_main_loop (control_frame); /* initiate SunView processing */
380
381     exit (0);
382
383 } /* main */
384
```

```
385  /*-----------------------------------------------------------------------*/
386
387  double
388  calc_distance (p1, p2)
389     int          p1[3], p2[3];
390
391  /*
392   *  Calculate distance between p1 and p2
393   */
394
395  {
396     int          a, b;
397     double       d;
398
399     a = p1[X] - p2[X];
400     b = p1[Y] - p2[Y];
401     d = sqrt ( (double) (a * a) + (b * b) );
402     return (d);
403
404  } /* calc_distance */
405
```

```
406 /*=============================================================================*/
407
408 void
409 clear_coordinates (item, event)
410    Panel_item        item;
411    Event             *event;
412
413 /*
414  *  Clear the current input coordinates
415  */
416
417 {
418    if (item == coor_in_clear_button) {       /* clear input side */
419       pw_lock (input_pw, &canvas_rect);
420       pw_writebackground (input_pw, 0, 0, CANVAS_MAX_X, CANVAS_MAX_Y, PIX_SRC);
421       pw_unlock (input_pw);
422       for (i = 0; i < MAX_COORDS && icoord[i][X] >= 0; i++)
423          /* clear the input coordinates */
424          for (j = X; j <= Z; j++) {
425             panel_set_value (coord_in_cell[i][j], "");
426             icoord[i][j] = -1;
427          }
428       max_reached = FALSE;                   /* reset */
429       row = 0;                               /* reset */
430       col = X;                               /* reset */
431       window_set (coord_in_panel, PANEL_CARET_ITEM, coord_in_cell[row][col], 0);
432       /* hide input "clear" and "save" options */
433       panel_set (coor_in_clear_button, PANEL_SHOW_ITEM, FALSE, 0);
434       panel_set (coor_in_save_button , PANEL_SHOW_ITEM, FALSE, 0);
435       panel_set (file_in_message, PANEL_LABEL_STRING, "", 0);
436    } /* clear input side */
437
438    /* clear output side in either case */
439    pw_lock (output_pw, &canvas_rect);
440    pw_writebackground (output_pw, 0, 0, CANVAS_MAX_X, CANVAS_MAX_Y, PIX_SRC);
441    pw_unlock (output_pw);
442    for (i = 0; i < MAX_COORDS && ocoord[i][X] >= 0; i++) {
443       /* clear the output coordinates */
444       for (j = X; j <= Z; j++) {
445          panel_set (coord_out_cell[i][j], PANEL_LABEL_STRING, "", 0);
446          ocoord[i][j] = -1;
447       }
448       if (i < MAX_MEASURES) /* reset measurements */
449          panel_set (measurement_line[i], PANEL_LABEL_STRING, "", 0);
450    }
451    /* hide output "clear" and "save" options */
452    panel_set (coor_out_clear_button, PANEL_SHOW_ITEM, FALSE, 0);
453    panel_set (coor_out_save_button , PANEL_SHOW_ITEM, FALSE, 0);
454    /* reset output file message */
455    panel_set (file_out_message, PANEL_LABEL_STRING, "", 0);
456    /* reset overlay status */
457    panel_set (disp_overlay_button, PANEL_LABEL_IMAGE, overlay_button_image, 0);
458    overlaid = FALSE;
459
460    window_set (measurements_panel, WIN_ROWS, 0, 0);
461    window_fit_height (measurements_panel);
462    window_fit (control_frame);
463
464    if (item == coor_in_clear_button) /* finally, show input "load" option */
465       panel_set (coor_in_load_button, PANEL_SHOW_ITEM, TRUE, 0);
466
467 } /* clear_coordinates */
468
```

```
469 /*-----------------------------------------------------------------------*/
470
471 void
472 define_menus ()
473
474 /*
475  *  Define required "walking" menus
476  */
477
478 {
479    simp_indep_pt_menu =
480       menu_create (        MENU_STRING_ITEM,          "Nth Point",
481                                                       SIMP_NTH_PT,
482                            MENU_STRING_ITEM,          "Random",
483                                                       SIMP_RANDOM_PT,
484                            MENU_NOTIFY_PROC,          do_simplification_choice,
485                            MENU_BOXED,                TRUE,
486                            0);
487    simp_local_menu =
488       menu_create (        MENU_STRING_ITEM,          "Line Width",
489                                                       SIMP_LINE_WIDTH,
490                            MENU_STRING_ITEM,          "Euclidean Distance",
491                                                       SIMP_EUCLIDEAN,
492                            MENU_STRING_ITEM,          "Perpendicular Distance",
493                                                       SIMP_PERPENDIC,
494                            MENU_STRING_ITEM,          "Angular Change",
495                                                       SIMP_ANGULAR,
496                            MENU_STRING_ITEM,          "Distance & Angle",
497                                                       SIMP_DIST_ANGLE,
498                            MENU_NOTIFY_PROC,          do_simplification_choice,
499                            MENU_BOXED,                TRUE,
500                            0);
501    simp_uncons_local_menu =
502       menu_create (        MENU_STRING_ITEM,          "Reuman-Witkam",
503                                                       SIMP_REUMAN,
504                            MENU_STRING_ITEM,          "Roberge",
505                                                       SIMP_ROBERGE,
506                            MENU_NOTIFY_PROC,          do_simplification_choice,
507                            MENU_BOXED,                TRUE,
508                            0);
509    simp_cons_local_menu =
510       menu_create (        MENU_STRING_ITEM,          "Lang",
511                                                       SIMP_LANG,
512                            MENU_STRING_ITEM,          "Johannsen",
513                                                       SIMP_JOHANNSEN,
514                            MENU_STRING_ITEM,          "Opheim",
515                                                       SIMP_OPHEIM,
516                            MENU_NOTIFY_PROC,          do_simplification_choice,
517                            MENU_BOXED,                TRUE,
518                            0);
519    simp_global_menu =
520       menu_create (        MENU_STRING_ITEM,          "Douglas-Peucker",
521                                                       SIMP_DOUGLAS,
522                            MENU_NOTIFY_PROC,          do_simplification_choice,
523                            MENU_BOXED,                TRUE,
524                            0);
525    simplification_menu =
526       menu_create (        MENU_PULLRIGHT_ITEM,       "Independent Point Processing",
527                                                       simp_indep_pt_menu,
528                            MENU_PULLRIGHT_ITEM,       "Local Processing",
529                                                       simp_local_menu,
530                            MENU_PULLRIGHT_ITEM,
531                                                  "Unconstrained Extended Local Processing",
```

```
532                                                 simp_uncons_local_menu,
533                             MENU_PULLRIGHT_ITEM,    "Constrained Local Processing",
534                                                 simp_cons_local_menu,
535                             MENU_PULLRIGHT_ITEM,    "Global Processing",
536                                                 simp_global_menu,
537                             0);
538
539     smoo_averaging_menu =
540         menu_create (       MENU_STRING_ITEM,       "Simple Averaging",
541                                                     SMOO_SIMPLE_AVE,
542                             MENU_STRING_ITEM,       "Weighted Averaging",
543                                                     SMOO_WEIGHT_AVE,
544                             MENU_STRING_ITEM,       "Forward Look Interpolation",
545                                                     SMOO_FWD_LOOK,
546                             MENU_NOTIFY_PROC,       do_smoothing_choice,
547                             MENU_BOXED,             TRUE,
548                             0);
549     smoo_epsilon_menu =
550         menu_create (       MENU_STRING_ITEM,       "Perkal's Geometric Filter",
551                                                     SMOO_PERKALS,
552                             MENU_STRING_ITEM,       "Brophy's Epsilon Filter",
553                                                     SMOO_BROPHYS,
554                             MENU_NOTIFY_PROC,       do_smoothing_choice,
555                             MENU_BOXED,             TRUE,
556                             0);
557     smoo_splining_local_menu =
558         menu_create (       MENU_STRING_ITEM,       "Cubic Spline",
559                                                     SMOO_CUBIC_SP,
560                             MENU_STRING_ITEM,       "Parabolic Spline",
561                                                     SMOO_PARAB_SP,
562                             MENU_NOTIFY_PROC,       do_smoothing_choice,
563                             MENU_BOXED,             TRUE,
564                             0);
565     smoo_splining_extended_menu =
566         menu_create (       MENU_STRING_ITEM,       "B-Spline",
567                                                     SMOO_B_SPLINE,
568                             MENU_NOTIFY_PROC,       do_smoothing_choice,
569                             MENU_BOXED,             TRUE,
570                             0);
571     smoo_splining_global_menu =
572         menu_create (       MENU_STRING_ITEM,       "Bezier Curves",
573                                                     SMOO_BEZIER_CUR,
574                             MENU_NOTIFY_PROC,       do_smoothing_choice,
575                             MENU_BOXED,             TRUE,
576                             0);
577     smoo_splining_menu =
578         menu_create (       MENU_PULLRIGHT_ITEM,    "Local",
579                                                     smoo_splining_local_menu,
580                             MENU_PULLRIGHT_ITEM,    "Extended Local",
581                                                     smoo_splining_extended_menu,
582                             MENU_PULLRIGHT_ITEM,    "Global",
583                                                     smoo_splining_global_menu,
584                             0);
585     smoothing_menu =
586         menu_create (       MENU_PULLRIGHT_ITEM,    "Averaging",
587                                                     smoo_averaging_menu,
588                             MENU_PULLRIGHT_ITEM,    "Epsilon Filtering",
589                                                     smoo_epsilon_menu,
590                             MENU_PULLRIGHT_ITEM,    "Splining",
591                                                     smoo_splining_menu,
592                             0);
593     measurement_menu =
594         menu_create (       MENU_ITEM,
```

```
595                             MENU_STRING,            "Absolutes",
596                             MENU_VALUE,             MEAS_ABS,
597                             MENU_ACTION_PROC,       do_measurement_choice,
598                             0,
599                         MENU_ITEM,
600                             MENU_STRING,            "Angularity",
601                             MENU_VALUE,             MEAS_ANG,
602                             MENU_ACTION_PROC,       do_measurement_choice,
603                             0,
604                         MENU_ITEM,
605                             MENU_STRING,            "Sinuosity",
606                             MENU_VALUE,             MEAS_SIN,
607                             MENU_ACTION_PROC,       do_measurement_choice,
608                             0,
609                         0);
610     color_menu =
611         menu_create (       MENU_STRING_ITEM,       "White",
612                                                     WHITE,
613                             MENU_STRING_ITEM,       "Red",
614                                                     RED,
615                             MENU_STRING_ITEM,       "Green",
616                                                     GREEN,
617                             MENU_STRING_ITEM,       "Blue",
618                                                     BLUE,
619                             MENU_STRING_ITEM,       "Yellow",
620                                                     YELLOW,
621                             MENU_STRING_ITEM,       "Cyan",
622                                                     CYAN,
623                             MENU_STRING_ITEM,       "Magenta",
624                                                     MAGENTA,
625                             MENU_STRING_ITEM,       "Black",
626                                                     BLACK,
627                             MENU_BOXED,             TRUE,
628                             MENU_NOTIFY_PROC,       do_color_choice,
629                             0);
630     back_color_menu =
631         menu_create (       MENU_STRING_ITEM,       "White",
632                                                     WHITE,
633                             MENU_STRING_ITEM,       "Light Red",
634                                                     LIGHT_RED,
635                             MENU_STRING_ITEM,       "Light Green",
636                                                     LIGHT_GREEN,
637                             MENU_STRING_ITEM,       "Light Blue",
638                                                     LIGHT_BLUE,
639                             MENU_STRING_ITEM,       "Gray",
640                                                     GRAY,
641                             MENU_STRING_ITEM,       "Black",
642                                                     BLACK,
643                             MENU_BOXED,             TRUE,
644                             MENU_NOTIFY_PROC,       do_color_choice,
645                             0);
646     color_types_menu =
647         menu_create (       MENU_PULLRIGHT_ITEM,    "Points",
648                                                     color_menu,
649                             MENU_PULLRIGHT_ITEM,    "Lines",
650                                                     color_menu,
651                             MENU_PULLRIGHT_ITEM,    "Areas",
652                                                     color_menu,
653                             MENU_PULLRIGHT_ITEM,    "Foreground",
654                                                     color_menu,
655                             MENU_PULLRIGHT_ITEM,    "Background",
656                                                     back_color_menu,
657                             0);
```

```
658     display_menu =
659         menu_create (       MENU_ITEM,
660                                 MENU_STRING,            "Clear",
661                                 MENU_VALUE,             DISP_CLEAR,
662                                 MENU_ACTION_PROC,       do_display_choice,
663                                 0,
664                             MENU_ITEM,
665                                 MENU_STRING,            "Zoom in",
666                                 MENU_VALUE,             DISP_ZOOM,
667                                 MENU_ACTION_PROC,       do_display_choice,
668                                 0,
669                             MENU_PULLRIGHT_ITEM,        "Color",
670                                                         color_types_menu,
671                             0);
672
673 } /* define_menus */
674
```

```
675  /*=============================================================================*/
676
677  void
678  define_windows (argc, argv)
679     int          argc;
680     char         **argv[];
681
682  /*
683   *  Define needed SunView frames, panels, canvases, etc.
684   */
685
686  {
687
688     linesimp_icon = /* control_frame (shell) icon */
689        icon_create (        ICON_IMAGE,                &icon_pixrect,
690                             0);
691
692     /*
693      *  Define main control panel, with process control panel and data control panel
694      */
695     control_frame =
696        window_create (      NULL, FRAME,
697                             FRAME_ARGS,                argc, argv,
698                             FRAME_ICON,                linesimp_icon,
699                             FRAME_BACKGROUND_COLOR,    &control_bg_color,
700                             FRAME_FOREGROUND_COLOR,    &control_fg_color,
701                             FRAME_INHERIT_COLORS,      FALSE,
702                             FRAME_LABEL,               "Line Simplification Control",
703                             WIN_X,                     0,
704                             WIN_Y,                     0,
705                             WIN_SHOW,                  TRUE,
706                             0);
707     control_panel =
708        window_create (      control_frame, PANEL,
709                             WIN_COLUMNS,               CONTROL_WIDTH_1+CONTROL_WIDTH_2,
710                             0);
711     simplify_button_image =
712        panel_button_image ( control_panel, "Simplify", 10, NULL);
713     smoothe_button_image =
714        panel_button_image ( control_panel, "Smoothe" , 10, NULL);
715     measure_button_image =
716        panel_button_image ( control_panel, "Measure" , 10, NULL);
717     reset_button_image =
718        panel_button_image ( control_panel, "Reset"   , 10, NULL);
719     quit_button_image =
720        panel_button_image ( control_panel, "Quit"    , 10, NULL);
721     simplify_button =
722        panel_create_item (  control_panel, PANEL_BUTTON,
723                             PANEL_LABEL_IMAGE,         simplify_button_image,
724                             PANEL_ITEM_X,              ATTR_COL(0),
725                             PANEL_ITEM_Y,              ATTR_ROW(SIMP_ROW)-2,
726                             PANEL_NOTIFY_PROC,         do_process,
727                             PANEL_EVENT_PROC,          show_button_menu,
728                             0);
729     simplification_method =
730        panel_create_item (  control_panel, PANEL_MESSAGE,
731                             PANEL_LABEL_STRING,        "Simplification Method:",
732                             PANEL_ITEM_X,              ATTR_COL(13),
733                             PANEL_ITEM_Y,              ATTR_ROW(SIMP_ROW),
734                             PANEL_EVENT_PROC,          show_button_menu,
735                             0);
736     current_simplification =
737        panel_create_item (  control_panel, PANEL_MESSAGE,
```

```
                            PANEL_LABEL_STRING,         SIMP_DEFAULT,
                            PANEL_LABEL_BOLD,           TRUE,
                            PANEL_ITEM_X,               ATTR_COL(CONTROL_WIDTH_1),
                            PANEL_ITEM_Y,               ATTR_ROW(SIMP_ROW),
                            PANEL_EVENT_PROC,           show_button_menu,
                            0);
    simplification_value = SIMP_DEFAULT_VALUE;

    smoothe_button =
       panel_create_item (  control_panel, PANEL_BUTTON,
                            PANEL_LABEL_IMAGE,          smoothe_button_image,
                            PANEL_ITEM_X,               ATTR_COL(0),
                            PANEL_ITEM_Y,               ATTR_ROW(SMOO_ROW)-2,
                            PANEL_NOTIFY_PROC,          do_process,
                            PANEL_EVENT_PROC,           show_button_menu,
                            0);
    smoothing_method =
       panel_create_item (  control_panel, PANEL_MESSAGE,
                            PANEL_LABEL_STRING,         "Smoothing Method:",
                            PANEL_ITEM_X,               ATTR_COL(13),
                            PANEL_ITEM_Y,               ATTR_ROW(SMOO_ROW),
                            PANEL_EVENT_PROC,           show_button_menu,
                            0);
    current_smoothing =
       panel_create_item (  control_panel, PANEL_MESSAGE,
                            PANEL_LABEL_STRING,         SMOO_DEFAULT,
                            PANEL_LABEL_BOLD,           TRUE,
                            PANEL_ITEM_X,               ATTR_COL(CONTROL_WIDTH_1),
                            PANEL_ITEM_Y,               ATTR_ROW(SMOO_ROW),
                            PANEL_EVENT_PROC,           show_button_menu,
                            0);
    smoothing_value = SMOO_DEFAULT_VALUE;

    measure_button =
       panel_create_item (  control_panel, PANEL_BUTTON,
                            PANEL_LABEL_IMAGE,          measure_button_image,
                            PANEL_ITEM_X,               ATTR_COL(0),
                            PANEL_ITEM_Y,               ATTR_ROW(MEAS_ROW)-2,
                            PANEL_NOTIFY_PROC,          do_process,
                            PANEL_EVENT_PROC,           show_button_menu,
                            0);
    measurement_method =
       panel_create_item (  control_panel, PANEL_MESSAGE,
                            PANEL_LABEL_STRING,         "Measurement Method:",
                            PANEL_ITEM_X,               ATTR_COL(13),
                            PANEL_ITEM_Y,               ATTR_ROW(MEAS_ROW),
                            PANEL_EVENT_PROC,           show_button_menu,
                            0);
    current_measurement =
       panel_create_item (  control_panel, PANEL_MESSAGE,
                            PANEL_LABEL_STRING,         MEAS_DEFAULT,
                            PANEL_LABEL_BOLD,           TRUE,
                            PANEL_ITEM_X,               ATTR_COL(CONTROL_WIDTH_1),
                            PANEL_ITEM_Y,               ATTR_ROW(MEAS_ROW),
                            PANEL_EVENT_PROC,           show_button_menu,
                            0);
    measurement_value = MEAS_DEFAULT_VALUE;

    reset_button =
       panel_create_item (  control_panel, PANEL_BUTTON,
                            PANEL_LABEL_IMAGE,          reset_button_image,
                            PANEL_ITEM_X,               ATTR_COL(
                               (CONTROL_WIDTH_1+CONTROL_WIDTH_2)/2-11),
```

```
                        PANEL_ITEM_Y,           ATTR_ROW(BUTTON_ROW)-2,
                        PANEL_NOTIFY_PROC,      do_reset,
                        0);
    quit_button =
       panel_create_item (  control_panel, PANEL_BUTTON,
                        PANEL_LABEL_IMAGE,      quit_button_image,
                        PANEL_ITEM_X,           ATTR_COL(
                           (CONTROL_WIDTH_1+CONTROL_WIDTH_2)/2+1),
                        PANEL_ITEM_Y,           ATTR_ROW(BUTTON_ROW)-2,
                        PANEL_NOTIFY_PROC,      do_quit,
                        0);
    window_fit_height (control_panel);

    bottom_panel =
       window_create (      control_frame, PANEL,
                        WIN_COLUMNS,            CONTROL_WIDTH_1+CONTROL_WIDTH_2,
                        WIN_BELOW,              control_panel,
                        WIN_X,                  0,
                        0);
    display_title =
       panel_create_item (  bottom_panel, PANEL_MESSAGE,
                        PANEL_LABEL_STRING,     "Displays:",
                        PANEL_ITEM_X,           ATTR_COL(0),
                        PANEL_ITEM_Y,           ATTR_ROW(DISP_ROW),
                        0);
    show_button_image =
       panel_button_image ( bottom_panel, "Show"      , 10, NULL);
    hide_button_image =
       panel_button_image ( bottom_panel, "Hide"      , 10, NULL);
    overlay_button_image =
       panel_button_image ( bottom_panel, "Overlay"   , 10, NULL),
    no_overlay_button_image =
       panel_button_image ( bottom_panel, "No Overlay", 10, NULL),
    disp_show_hide_button =
       panel_create_item (  bottom_panel, PANEL_BUTTON,
                        PANEL_LABEL_IMAGE,      show_button_image,
                        PANEL_ITEM_X,           ATTR_COL(13),
                        PANEL_ITEM_Y,           ATTR_ROW(DISP_ROW)-2,
                        PANEL_NOTIFY_PROC,      show_hide_displays,
                        0);
    disp_overlay_button =
       panel_create_item (  bottom_panel, PANEL_BUTTON,
                        PANEL_LABEL_IMAGE,      overlay_button_image,
                        PANEL_ITEM_X,           ATTR_COL(25),
                        PANEL_ITEM_Y,           ATTR_ROW(DISP_ROW)-2,
                        PANEL_NOTIFY_PROC,      overlay_displays,
                        0);
    coordinate_title =
       panel_create_item (  bottom_panel, PANEL_MESSAGE,
                        PANEL_LABEL_STRING,     "Coordinates:",
                        PANEL_ITEM_X,           ATTR_COL(0),
                        PANEL_ITEM_Y,           ATTR_ROW(COOR_ROW),
                        0);
    coor_show_hide_button =
       panel_create_item (  bottom_panel, PANEL_BUTTON,
                        PANEL_LABEL_IMAGE,      show_button_image,
                        PANEL_ITEM_X,           ATTR_COL(13),
                        PANEL_ITEM_Y,           ATTR_ROW(COOR_ROW)-2,
                        PANEL_NOTIFY_PROC,      show_hide_coordinates,
                        0);
    window_fit_height (bottom_panel);

    measurements_panel =
```

```
864         window_create (       control_frame, PANEL,
865                               PANEL_LABEL_BOLD,           FALSE,
866                               WIN_ROWS,                   MAX_MEASURES,
867                               WIN_COLUMNS,                CONTROL_WIDTH_1+CONTROL_WIDTH_2,
868                               WIN_BELOW,                  bottom_panel,
869                               WIN_X,                      0,
870                               0);
871     for (i = 0; i < MAX_MEASURES; i++)
872        measurement_line[i] =
873           panel_create_item(measurements_panel, PANEL_MESSAGE,
874                               PANEL_VALUE_DISPLAY_LENGTH, CONTROL_WIDTH_1+CONTROL_WIDTH_2,
875                               PANEL_ITEM_X,               ATTR_COL(0),
876                               PANEL_ITEM_Y,               ATTR_ROW(i),
877                               0);
878     window_set (measurements_panel, WIN_ROWS, 0, 0);
879     window_fit_height (measurements_panel);
880     window_fit (control_frame);
881
882     tolerance_popup =
883        window_create (       control_frame, FRAME,
884                               FRAME_BACKGROUND_COLOR,     &popup_bg_color,
885                               FRAME_FOREGROUND_COLOR,     &popup_fg_color,
886                               WIN_ROWS,                   3,
887                               WIN_SHOW,                   FALSE,
888                               WIN_X,                      50,
889                               WIN_Y,                      265,
890                               0);
891     tolerance_panel =
892        window_create (       tolerance_popup, PANEL,
893                               0);
894     tolerance_text_item =
895        panel_create_item (   tolerance_panel, PANEL_TEXT,
896                               PANEL_LABEL_STRING,         "Enter tolerance (in pixels):",
897                               PANEL_VALUE,                TOL_DEFAULT,
898                               PANEL_VALUE_DISPLAY_LENGTH, VALUE_LEN,
899                               PANEL_VALUE_STORED_LENGTH,  VALUE_LEN,
900                               0);
901     tolerance_ok_button =
902        panel_create_item (   tolerance_panel, PANEL_BUTTON,
903                               PANEL_LABEL_IMAGE,          panel_button_image (
904                                  tolerance_panel, "OK", 10, NULL),
905                               PANEL_NOTIFY_PROC,          ok_button,
906                               PANEL_ITEM_X,               ATTR_COL(11),
907                               PANEL_ITEM_Y,               ATTR_ROW(2),
908                               0);
909     window_fit (tolerance_panel);
910     window_fit (tolerance_popup);
911
912     /*
913      *  Define coordinate panel for input and output coordinates
914      */
915     coordinate_frame =
916        window_create (       control_frame, FRAME,
917                               FRAME_LABEL,
918                                  "Original (input) and modified (output) coordinates",
919                               FRAME_INHERIT_COLORS,       FALSE,
920                               FRAME_BACKGROUND_COLOR,     &control_bg_color,
921                               FRAME_FOREGROUND_COLOR,     &control_fg_color,
922                               FRAME_DONE_PROC,            do_done,
923                               WIN_SHOW,                   FALSE,
924                               WIN_X,                      545,
925                               WIN_Y,                      0,
926                               0);
```

```
927     file_in_panel =
928        window_create (      coordinate_frame, PANEL,
929                             PANEL_LABEL_BOLD,          TRUE,
930                             WIN_COLUMNS,               COORDS_COLS,
931                             WIN_ROWS,                  6,
932                             0);
933     load_button_image =
934        panel_button_image ( file_in_panel, "Load" , 8, NULL);
935     clear_button_image =
936        panel_button_image ( file_in_panel, "Clear", 8, NULL);
937     save_button_image =
938        panel_button_image ( file_in_panel, "Save" , 8, NULL);
939     path_in_item =
940        panel_create_item (  file_in_panel, PANEL_TEXT,
941                             PANEL_LABEL_STRING,        "Path:",
942                             PANEL_VALUE_DISPLAY_LENGTH, NAME_LEN,
943                             PANEL_VALUE,               "",
944                             PANEL_ITEM_X,              ATTR_COL(0),
945                             PANEL_ITEM_Y,              ATTR_ROW(0),
946                             0);
947     file_in_item =
948        panel_create_item (  file_in_panel, PANEL_TEXT,
949                             PANEL_LABEL_STRING,        "File:",
950                             PANEL_VALUE_DISPLAY_LENGTH, NAME_LEN,
951                             PANEL_VALUE,               "",
952                             PANEL_ITEM_X,              ATTR_COL(0),
953                             PANEL_ITEM_Y,              ATTR_ROW(1),
954                             0);
955     file_in_message =
956        panel_create_item (  file_in_panel, PANEL_MESSAGE,
957                             PANEL_ITEM_X,              ATTR_COL(0),
958                             PANEL_ITEM_Y,              ATTR_ROW(2),
959                             PANEL_VALUE_DISPLAY_LENGTH, NAME_LEN+6,
960                             0);
961     coor_in_load_button =
962        panel_create_item (  file_in_panel, PANEL_BUTTON,
963                             PANEL_LABEL_IMAGE,         load_button_image,
964                             PANEL_ITEM_X,              ATTR_COL(0),
965                             PANEL_ITEM_Y,              ATTR_ROW(3),
966                             PANEL_NOTIFY_PROC,         file_i_o,
967                             0);
968     coor_in_clear_button =
969        panel_create_item (  file_in_panel, PANEL_BUTTON,
970                             PANEL_LABEL_IMAGE,         clear_button_image,
971                             PANEL_SHOW_ITEM,           FALSE,
972                             PANEL_ITEM_X,              ATTR_COL(11),
973                             PANEL_ITEM_Y,              ATTR_ROW(3),
974                             PANEL_NOTIFY_PROC,         clear_coordinates,
975                             0);
976     coor_in_save_button =
977        panel_create_item (  file_in_panel, PANEL_BUTTON,
978                             PANEL_LABEL_IMAGE,         save_button_image,
979                             PANEL_SHOW_ITEM,           FALSE,
980                             PANEL_ITEM_X,              ATTR_COL(22),
981                             PANEL_ITEM_Y,              ATTR_ROW(3),
982                             PANEL_NOTIFY_PROC,         file_i_o,
983                             0);
984     input_header =
985        panel_create_item (  file_in_panel, PANEL_MESSAGE,
986                             PANEL_LABEL_STRING,        "I N P U T",
987                             PANEL_ITEM_X,              ATTR_COL(COORDS_COLS/2-5),
988                             PANEL_ITEM_Y,              ATTR_ROW(4),
989                             0);
```

```
990     coord_in_header =
991        panel_create_item (  file_in_panel, PANEL_MESSAGE,
992                             PANEL_LABEL_STRING,         coord_column_header,
993                             PANEL_ITEM_X,               ATTR_COL(0),
994                             PANEL_ITEM_Y,               ATTR_ROW(5),
995                             0);
996     window_fit_height (file_in_panel);
997     file_out_panel =
998        window_create (      coordinate_frame, PANEL,
999                             PANEL_LABEL_BOLD,           TRUE,
1000                            WIN_RIGHT_OF,               file_in_panel,
1001                            WIN_Y,                      0,
1002                            WIN_COLUMNS,                COORDS_COLS,
1003                            WIN_ROWS,                   6,
1004                            0);
1005    path_out_item =
1006       panel_create_item (  file_out_panel, PANEL_TEXT,
1007                            PANEL_LABEL_STRING,         "Path:",
1008                            PANEL_VALUE_DISPLAY_LENGTH, NAME_LEN,
1009                            PANEL_VALUE,                "",
1010                            PANEL_ITEM_X,               ATTR_COL(0),
1011                            PANEL_ITEM_Y,               ATTR_ROW(0),
1012                            0);
1013    file_out_item =
1014       panel_create_item (  file_out_panel, PANEL_TEXT,
1015                            PANEL_LABEL_STRING,         "File:",
1016                            PANEL_VALUE_DISPLAY_LENGTH, NAME_LEN,
1017                            PANEL_VALUE,                "",
1018                            PANEL_ITEM_X,               ATTR_COL(0),
1019                            PANEL_ITEM_Y,               ATTR_ROW(1),
1020                            0);
1021    file_out_message =
1022       panel_create_item (  file_out_panel, PANEL_MESSAGE,
1023                            PANEL_ITEM_X,               ATTR_COL(0),
1024                            PANEL_ITEM_Y,               ATTR_ROW(2),
1025                            PANEL_VALUE_DISPLAY_LENGTH, NAME_LEN+6,
1026                            0);
1027    coor_out_clear_button =
1028       panel_create_item (  file_out_panel,   PANEL_BUTTON,
1029                            PANEL_LABEL_IMAGE,          clear_button_image,
1030                            PANEL_SHOW_ITEM,            FALSE,
1031                            PANEL_ITEM_X,               ATTR_COL(11),
1032                            PANEL_ITEM_Y,               ATTR_ROW(3),
1033                            PANEL_NOTIFY_PROC,          clear_coordinates,
1034                            0);
1035    coor_out_save_button =
1036       panel_create_item (  file_out_panel,   PANEL_BUTTON,
1037                            PANEL_LABEL_IMAGE,          save_button_image,
1038                            PANEL_SHOW_ITEM,            FALSE,
1039                            PANEL_ITEM_X,               ATTR_COL(22),
1040                            PANEL_ITEM_Y,               ATTR_ROW(3),
1041                            PANEL_NOTIFY_PROC,          file_i_o,
1042                            0);
1043    output_header =
1044       panel_create_item (  file_out_panel, PANEL_MESSAGE,
1045                            PANEL_LABEL_STRING,         "O U T P U T",
1046                            PANEL_ITEM_X,               ATTR_COL(COORDS_COLS/2-6),
1047                            PANEL_ITEM_Y,               ATTR_ROW(4),
1048                            0);
1049    coord_out_header =
1050       panel_create_item (  file_out_panel, PANEL_MESSAGE,
1051                            PANEL_LABEL_STRING,         coord_column_header,
1052                            PANEL_ITEM_X,               ATTR_COL(0),
```

```
1053                            PANEL_ITEM_Y,               ATTR_ROW(5),
1054                            0);
1055        window_fit_height (file_out_panel);
1056
1057        coord_cursor =
1058           cursor_create (      CURSOR_IMAGE,               &cross_pixrect,
1059                                CURSOR_XHOT,                8,
1060                                CURSOR_YHOT,                8,
1061                                CURSOR_OP,                  (PIX_SRC | PIX_DST),
1062                                0);
1063
1064        coord_in_panel =
1065           window_create (      coordinate_frame, PANEL,
1066                                PANEL_LABEL_BOLD,           TRUE,
1067                                WIN_BELOW,                  file_in_panel,
1068                                WIN_X,                      0,
1069                                WIN_WIDTH,
1070                                   (int) window_get (file_in_panel, WIN_WIDTH, 0),
1071                                WIN_ROWS,                   COORDS_ROWS,
1072                                WIN_LEFT_MARGIN,            4,
1073                                WIN_RIGHT_MARGIN,           4,
1074                                WIN_TOP_MARGIN,             4,
1075                                WIN_BOTTOM_MARGIN,          0,
1076                                WIN_CURSOR,                 coord_cursor,
1077                                WIN_VERTICAL_SCROLLBAR,     scrollbar_create (
1078                                   SCROLL_PLACEMENT,        VERTICAL_LOC,
1079                                   SCROLL_BUBBLE_MARGIN,    BUBBLE_MARGIN,
1080                                   0),
1081                                0);
1082        coord_out_panel =
1083           window_create (      coordinate_frame, PANEL,
1084                                PANEL_LABEL_BOLD,           TRUE,
1085                                WIN_BELOW,                  file_out_panel,
1086                                WIN_RIGHT_OF,               coord_in_panel,
1087                                WIN_WIDTH,
1088                                   (int) window_get (file_out_panel, WIN_WIDTH, 0),
1089                                WIN_ROWS,                   COORDS_ROWS,
1090                                WIN_LEFT_MARGIN,            4,
1091                                WIN_RIGHT_MARGIN,           4,
1092                                WIN_TOP_MARGIN,             4,
1093                                WIN_BOTTOM_MARGIN,          0,
1094                                WIN_VERTICAL_SCROLLBAR,     scrollbar_create (
1095                                   SCROLL_PLACEMENT,        VERTICAL_LOC,
1096                                   SCROLL_BUBBLE_MARGIN,    BUBBLE_MARGIN,
1097                                   0),
1098                                0);
1099        for (i = 0; i < MAX_COORDS; i++) {
1100           char         label[LABEL_LEN];
1101           sprintf (label, "%4d", i+1);
1102           coord_in_label[i] = panel_create_item (
1103                                coord_in_panel, PANEL_MESSAGE,
1104                                PANEL_LABEL_STRING,         label,
1105                                PANEL_ITEM_X,               ATTR_COL(0),
1106                                PANEL_ITEM_Y,               ATTR_ROW(i),
1107                                0);
1108           coord_out_label[i] = panel_create_item (
1109                                coord_out_panel, PANEL_MESSAGE,
1110                                PANEL_LABEL_STRING,         label,
1111                                PANEL_ITEM_X,               ATTR_COL(0),
1112                                PANEL_ITEM_Y,               ATTR_ROW(i),
1113                                0);
1114           for (j = X; j <= Z; j++) {
1115              icoord[i][j] =
```

```
1116            ocoord[i][j] = -1;
1117            coord_in_cell[i][j] = panel_create_item (
1118                                    coord_in_panel, PANEL_TEXT,
1119                                    PANEL_LABEL_STRING,         "|",
1120                                    PANEL_VALUE_STORED_LENGTH, VALUE_LEN,
1121                                    PANEL_VALUE_DISPLAY_LENGTH,VALUE_LEN,
1122                                    PANEL_ITEM_X,
1123                                       ATTR_COL(LABEL_LEN+1+(j*(VALUE_LEN+3))),
1124                                    PANEL_ITEM_Y,               ATTR_ROW(i),
1125                                    PANEL_NOTIFY_LEVEL,         PANEL_ALL,
1126                                    PANEL_NOTIFY_PROC,          enter_coord_char,
1127                                    PANEL_EVENT_PROC,           locate_item,
1128                                    0);
1129            coord_out_start[i][j] = panel_create_item (
1130                                    coord_out_panel, PANEL_MESSAGE,
1131                                    PANEL_LABEL_STRING,         "|",
1132                                    PANEL_ITEM_X,
1133                                       ATTR_COL(LABEL_LEN+1+(j*(VALUE_LEN+3))),
1134                                    PANEL_ITEM_Y,               ATTR_ROW(i),
1135                                    0);
1136            coord_out_cell[i][j] = panel_create_item (
1137                                    coord_out_panel, PANEL_MESSAGE,
1138                                    PANEL_LABEL_BOLD,           FALSE,
1139                                    PANEL_ITEM_X,
1140                                       ATTR_COL(LABEL_LEN+3+(j*(VALUE_LEN+3))),
1141                                    PANEL_ITEM_Y,               ATTR_ROW(i),
1142                                    0);
1143        }
1144        coord_in_ender[i] = panel_create_item (
1145                                    coord_in_panel, PANEL_MESSAGE,
1146                                    PANEL_LABEL_STRING,         "|",
1147                                    PANEL_ITEM_X,               ATTR_COL(29),
1148                                    PANEL_ITEM_Y,               ATTR_ROW(i),
1149                                    0);
1150        coord_out_ender[i] = panel_create_item (
1151                                    coord_out_panel, PANEL_MESSAGE,
1152                                    PANEL_LABEL_STRING,         "|",
1153                                    PANEL_ITEM_X,               ATTR_COL(29),
1154                                    PANEL_ITEM_Y,               ATTR_ROW(i),
1155                                    0);
1156    } /* for i */
1157    window_fit (coordinate_frame);
1158
1159  /*
1160   *  Input & output canvases for graphic representations of original and modified lines
1161   */
1162    display_frame =
1163       window_create (          control_frame, FRAME,
1164                                FRAME_LABEL,
1165                                   "Original (input) and modified (output) line:",
1166                                FRAME_DONE_PROC,            do_done,
1167                                WIN_X,                      0,
1168                                WIN_Y,                      378,
1169                                WIN_SHOW,                   FALSE,
1170                                0);
1171    draw_cursor =
1172       cursor_create (          CURSOR_IMAGE,               &hairs_pixrect,
1173                                CURSOR_XHOT,                8,
1174                                CURSOR_YHOT,                8,
1175                                CURSOR_SHOW_CROSSHAIRS,     TRUE,
1176                                CURSOR_CROSSHAIR_GAP,       10,
1177                                0);
1178    input_canvas =
```

```
1179        window_create (      display_frame, CANVAS,
1180                             CANVAS_WIDTH,                CANVAS_MAX_X,
1181                             CANVAS_HEIGHT,               CANVAS_MAX_Y,
1182                             CANVAS_AUTO_SHRINK,          FALSE,
1183                             WIN_WIDTH,                   INIT_WIDTH,
1184                             WIN_HEIGHT,                  INIT_HEIGHT,
1185                             WIN_CURSOR,                  draw_cursor,
1186                             WIN_VERTICAL_SCROLLBAR,      scrollbar_create (
1187                                SCROLL_PLACEMENT,         VERTICAL_LOC,
1188                                SCROLL_BUBBLE_MARGIN,     BUBBLE_MARGIN,
1189                                0),
1190                             WIN_HORIZONTAL_SCROLLBAR,    scrollbar_create (
1191                                SCROLL_DIRECTION,         SCROLL_HORIZONTAL,
1192                                SCROLL_PLACEMENT,         HORIZONTAL_LOC,
1193                                SCROLL_BUBBLE_MARGIN,     BUBBLE_MARGIN,
1194                                0),
1195                             /* need all mouse buttons for scrolling! */
1196                             WIN_CONSUME_PICK_EVENTS,     WIN_MOUSE_BUTTONS,
1197                                                          LOC_DRAG,
1198                                                          0,
1199                             WIN_EVENT_PROC,              handle_canvas_event,
1200                             0);
1201     output_canvas =
1202        window_create (      display_frame, CANVAS,
1203                             CANVAS_WIDTH,                CANVAS_MAX_X,
1204                             CANVAS_HEIGHT,               CANVAS_MAX_Y,
1205                             CANVAS_AUTO_SHRINK,          FALSE,
1206                             WIN_RIGHT_OF,                input_canvas,
1207                             WIN_WIDTH,                   INIT_WIDTH,
1208                             WIN_HEIGHT,                  INIT_HEIGHT,
1209                             WIN_CURSOR,                  draw_cursor,
1210                             WIN_VERTICAL_SCROLLBAR,      scrollbar_create (
1211                                SCROLL_PLACEMENT,         VERTICAL_LOC,
1212                                SCROLL_BUBBLE_MARGIN,     BUBBLE_MARGIN,
1213                                0),
1214                             WIN_HORIZONTAL_SCROLLBAR,    scrollbar_create (
1215                                SCROLL_DIRECTION,         SCROLL_HORIZONTAL,
1216                                SCROLL_PLACEMENT,         HORIZONTAL_LOC,
1217                                SCROLL_BUBBLE_MARGIN,     BUBBLE_MARGIN,
1218                                0),
1219                             /* need all mouse buttons for scrolling! */
1220                             WIN_CONSUME_PICK_EVENTS,     WIN_MOUSE_BUTTONS, 0,
1221                             WIN_IGNORE_PICK_EVENT,       LOC_DRAG,
1222                             WIN_EVENT_PROC,              handle_canvas_event,
1223                             0);
1224     window_fit (display_frame);
1225
1226      input_pw = canvas_pixwin ( input_canvas);
1227     output_pw = canvas_pixwin (output_canvas);
1228
1229     make_color_map ();
1230
1231  } /* define_windows */
1232
```

```
1233  /*=============================================================================*/
1234
1235  void
1236  do_color_choice (menu, menu_item, np)
1237     Menu               menu;
1238     Menu_item          menu_item;
1239     caddr_t            (*np) ();
1240
1241  /*
1242   *  Process the desired color choice
1243   */
1244
1245  {
1246     int                color;
1247     color = (int) menu_get (menu_item, MENU_VALUE);
1248
1249     if (menu_get (menu, MENU_PARENT) ==
1250           menu_find (color_types_menu, MENU_STRING, "Points", 0) )
1251        set_color ( (input) ? IN_POINT_COLOR : OUT_POINT_COLOR, color);
1252     else
1253     if (menu_get (menu, MENU_PARENT) ==
1254           menu_find (color_types_menu, MENU_STRING, "Lines", 0) )
1255        set_color ( (input) ? IN_LINE_COLOR : OUT_LINE_COLOR, color);
1256     else
1257     if (menu_get (menu, MENU_PARENT) ==
1258           menu_find (color_types_menu, MENU_STRING, "Areas", 0) )
1259        set_color ( (input) ? IN_AREA_COLOR : OUT_AREA_COLOR, color);
1260     else
1261     if (menu_get (menu, MENU_PARENT) ==
1262           menu_find (color_types_menu, MENU_STRING, "Foreground", 0) )
1263        set_color (FOREGROUND, color);
1264     else
1265     if (menu_get (menu, MENU_PARENT) ==
1266           menu_find (color_types_menu, MENU_STRING, "Background", 0) )
1267        set_color (BACKGROUND, color);
1268
1269     /* set up new colormap for the canvas */
1270     if (input) {
1271        pw_setcmsname  ( input_pw, cms.cms_name);
1272        pw_putcolormap ( input_pw, 0, COLOR_MAP_SIZE, RGB[R], RGB[G], RGB[B]);
1273     }
1274     else {
1275        pw_setcmsname  (output_pw, cms.cms_name);
1276        pw_putcolormap (output_pw, 0, COLOR_MAP_SIZE, RGB[R], RGB[G], RGB[B]);
1277     }
1278
1279  } /* do_color_choice */
1280
```

```
1281  /*-----------------------------------------------------------------------*/
1282
1283  void
1284  do_display_choice (menu, menu_item, np)
1285     Menu                    menu;
1286     Menu_item               menu_item;
1287     caddr_t                 (*np) ();
1288
1289  /*
1290   *  Process the desired display menu choice
1291   */
1292
1293  {
1294     Event          *null_event;
1295
1296     menu_get (menu_item, MENU_VALUE); /* force evaluation of pullrights */
1297     choice = (int) menu_get (menu_item, MENU_VALUE);
1298     switch (choice) {
1299        case DISP_CLEAR:
1300           clear_coordinates ( (input) ? coor_in_clear_button
1301                                       : coor_out_clear_button, null_event);
1302           break;
1303        case DISP_ZOOM:
1304           /* future capability */
1305           break;
1306        case DISP_COLOR:
1307           /* nothing needed */
1308           break;
1309        default:
1310           break;
1311     } /* switch */
1312
1313  } /* do_display_choice */
1314
```

```
1315  /*-----------------------------------------------------------------------*/
1316
1317  void
1318  do_done (window, event, arg)
1319     Window          window;
1320     Event           *event;
1321     caddr_t         arg;
1322
1323  {
1324     Panel_item      button;
1325
1326     window_set (window, WIN_SHOW, FALSE, 0); /* close window */
1327
1328     button = (window == coordinate_frame) ? coor_show_hide_button
1329                                           : disp_show_hide_button;
1330
1331     panel_set (button, PANEL_LABEL_IMAGE, show_button_image, 0);
1332
1333  } /* do_done */
1334
```

```
1335  /*-----------------------------------------------------------------------*/
1336
1337  void
1338  do_douglas_peucker (event)
1339     Event              *event;
1340
1341  /*
1342   *  Perform Douglas-Peucker simplification algorithm on coordinates
1343   */
1344
1345  {
1346     int                i, cnt, tol;
1347
1348     tol = atoi ( window_loop (tolerance_popup) ); /* force user to press "OK" */
1349
1350     if (ocoord[0][X] >= 0) /* clear old output data */
1351        clear_coordinates (coor_out_clear_button, event);
1352
1353     cnt = (max_reached) ? row+1 : row;
1354     cnt = Douglas_Peucker (tol, cnt); /* simplify it! */
1355
1356     for (i = 0; i < cnt; i++) {
1357        ocoord[i][Z] = icoord[i][Z]; /* Z axis not handled by this algorithm */
1358        enter_out_coordinates (i); /* draw new line segment */
1359     }
1360
1361  } /* do_douglas_peucker */
1362
```

```
1363  /*-------------------------------------------------------------------------*/
1364
1365  void
1366  do_measure_absolutes ()
1367
1368  /*
1369   *  Measure number of coordinates, total length, total runs of a line
1370   */
1371
1372  {
1373     static
1374     int             i, icnt, ocnt;
1375     static
1376     double          ilen, olen;
1377     static
1378     char            measurement[80],
1379     .               number[8];
1380
1381     window_set (measurements_panel, WIN_ROWS, 3, 0);
1382
1383     /*-------------------------------------------------------------------*
1384      *  Number of coordinates
1385      *-------------------------------------------------------------------*/
1386     measurement[0] = '\0'; /* reset */
1387     strcat (measurement, "Number of coordinates:       ");
1388     if (icoord[0][X] >= 0) {
1389        for (icnt = 0; icnt < MAX_COORDS && icoord[icnt][X] >= 0; icnt++);
1390        strcat (measurement, " IN ==> ");
1391        sprintf (number, "%8u", icnt);
1392        strcat (measurement, number);
1393        if (ocoord[0][X] >= 0) {
1394           for (ocnt = 0; ocnt < MAX_COORDS && ocoord[ocnt][X] >= 0; ocnt++);
1395           strcat (measurement, "; OUT ==> ");
1396           sprintf (number, "%8u", ocnt);
1397           strcat (measurement, number);
1398        }
1399     }
1400     panel_set (measurement_line[0], PANEL_LABEL_STRING, measurement, 0);
1401
1402     /*-------------------------------------------------------------------*
1403      *  Total length in inches
1404      *-------------------------------------------------------------------*/
1405     measurement[0] = '\0'; /* reset */
1406     strcat (measurement, "Total length (inches):       ");
1407     if (icoord[0][X] >= 0) {
1408        ilen = 0.0;
1409        for (i = 1; i < MAX_COORDS && icoord[i][X] >= 0; i++)
1410           ilen += calc_distance (icoord[i-1], icoord[i]);
1411        ilen = (ilen / DPI) + 0.005; /* rounded */
1412        strcat (measurement, " IN ==> ");
1413        sprintf (number, "%8.2f", (float) ilen);
1414        strcat (measurement, number);
1415        if (ocoord[0][X] >= 0) {
1416           olen = 0.0;
1417           for (i = 1; i < MAX_COORDS && ocoord[i][X] >= 0; i++)
1418              olen += calc_distance (ocoord[i-1], ocoord[i]);
1419           olen = (olen / DPI) + 0.005; /* rounded */
1420           strcat (measurement, "; OUT ==> ");
1421           sprintf (number, "%8.2f", (float) olen);
1422           strcat (measurement, number);
1423        }
1424     }
1425     panel_set (measurement_line[1], PANEL_LABEL_STRING, measurement, 0);
```

```
1426
1427     /*-------------------------------------------------------------------------*
1428      *  Average number of coordinates per inch
1429      *-------------------------------------------------------------------------*/
1430     measurement[0] = '\0'; /* reset */
1431     strcat (measurement, "Average # coordinates/inch:  ");
1432     if (icoord[0][X] >= 0) {
1433        strcat (measurement, " IN ==> ");
1434        sprintf (number, "%8.2f", (float) (icnt / ilen) );
1435        strcat (measurement, number);
1436        if (ocoord[0][X] >= 0) {
1437           strcat (measurement, "; OUT ==> ");
1438           sprintf (number, "%8.2f", (float) (ocnt / olen) );
1439           strcat (measurement, number);
1440        }
1441     }
1442     panel_set (measurement_line[2], PANEL_LABEL_STRING, measurement, 0);
1443
1444     panel_set (measurement_line[3], PANEL_LABEL_STRING, "", 0);
1445     panel_set (measurement_line[4], PANEL_LABEL_STRING, "", 0);
1446
1447     window_fit_height (measurements_panel);
1448     window_fit (control_frame);
1449
1450  } /* do_measure_absolutes */
1451
```

```
1452  /*=============================================================================*/
1453
1454  void
1455  do_measure_angularity ()
1456
1457  /*
1458   *  Measure angularity of a line
1459   */
1460
1461  {
1462     static
1463     char           measurement[80],
1464                    number[8];
1465     static struct
1466     measures       imeas, omeas;
1467
1468     window_set (measurements_panel, WIN_ROWS, 5, 0);
1469
1470     if (icoord[0][X] >= 0) {
1471        measure (icoord, &imeas);
1472        if (ocoord[0][X] >= 0)
1473           measure (ocoord, &omeas);
1474
1475        /* total angularity */
1476        measurement[0] = '\0'; /* reset */
1477        strcat (measurement, "Total Angularity:            ");
1478        strcat (measurement, " IN ==> ");
1479        sprintf (number, "%8.2f", imeas.total_angularity);
1480        strcat (measurement, number);
1481        if (ocoord[0][X] >= 0) {
1482           strcat (measurement, "; OUT ==> ");
1483           sprintf (number, "%8.2f", omeas.total_angularity);
1484           strcat (measurement, number);
1485        }
1486        panel_set (measurement_line[0], PANEL_LABEL_STRING, measurement, 0);
1487
1488        /* right angularity */
1489        measurement[0] = '\0'; /* reset */
1490        strcat (measurement, "Right Angularity:            ");
1491        strcat (measurement, " IN ==> ");
1492        sprintf (number, "%8.2f", imeas.right_angularity);
1493        strcat (measurement, number);
1494        if (ocoord[0][X] >= 0) {
1495           strcat (measurement, "; OUT ==> ");
1496           sprintf (number, "%8.2f", omeas.right_angularity);
1497           strcat (measurement, number);
1498        }
1499        panel_set (measurement_line[1], PANEL_LABEL_STRING, measurement, 0);
1500
1501        /* left angularity */
1502        measurement[0] = '\0'; /* reset */
1503        strcat (measurement, "Left Angularity:             ");
1504        strcat (measurement, " IN ==> ");
1505        sprintf (number, "%8.2f", imeas.left_angularity);
1506        strcat (measurement, number);
1507        if (ocoord[0][X] >= 0) {
1508           strcat (measurement, "; OUT ==> ");
1509           sprintf (number, "%8.2f", omeas.left_angularity);
1510           strcat (measurement, number);
1511        }
1512        panel_set (measurement_line[2], PANEL_LABEL_STRING, measurement, 0);
1513
1514        /* standardized angularity per inch */
```

```
1515        measurement[0] = '\0'; /* reset */
1516        strcat (measurement, "Standardized Angularity/Inch:");
1517        strcat (measurement, " IN ==> ");
1518        sprintf (number, "%8.2f", imeas.std_angularity_inch);
1519        strcat (measurement, number);
1520        if (ocoord[0][X] >= 0) {
1521           strcat (measurement, "; OUT ==> ");
1522           sprintf (number, "%8.2f", omeas.std_angularity_inch);
1523           strcat (measurement, number);
1524        }
1525        panel_set (measurement_line[3], PANEL_LABEL_STRING, measurement, 0);
1526
1527        /* total runs */
1528        measurement[0] = '\0'; /* reset */
1529        strcat (measurement, "Total Runs:                    ");
1530        strcat (measurement, " IN ==> ");
1531        sprintf (number, "%8u", (int) imeas.total_runs);
1532        strcat (measurement, number);
1533        if (ocoord[0][X] >= 0) {
1534           strcat (measurement, "; OUT ==> ");
1535           sprintf (number, "%8u", (int) omeas.total_runs);
1536           strcat (measurement, number);
1537        }
1538        panel_set (measurement_line[4], PANEL_LABEL_STRING, measurement, 0);
1539     }
1540
1541     window_fit_height (measurements_panel);
1542     window_fit (control_frame);
1543
1544  } /* do_measure_angularity */
1545
```

```
1546  /*=============================================================================*/
1547
1548  void
1549  do_measure_sinuosity ()
1550
1551  /*
1552   *  Measure sinuosity of a line
1553   */
1554
1555  {
1556
1557  } /* do_measure_sinuosity */
1558
```

```
1559  /*-----------------------------------------------------------------------*/
1560
1561  void
1562  do_measurement_choice (menu, menu_item, np)
1563     Menu                  menu;
1564     Menu_item             menu_item;
1565     caddr_t               (*np) ();
1566
1567  /*
1568   *  Set up the desired measurement method
1569   */
1570
1571  {
1572     menu_get (menu_item, MENU_VALUE); /* Force evaluation of pullrights */
1573     measurement_value = (int) menu_get (menu_item, MENU_VALUE);
1574
1575     /* display the measurement method on the control panel */
1576     panel_set (current_measurement, PANEL_LABEL_STRING,
1577        menu_get (menu_item, MENU_STRING), 0);
1578
1579  } /* do_measurement_choice */
1580
```

```
1581  /*-----------------------------------------------------------------------*/
1582
1583  void
1584  do_process (item, event)
1585     Panel_item           item;
1586     Event                *event;
1587
1588  /*
1589   *  Process the coordinates according to the current control method settings
1590   */
1591
1592  {
1593     if (item == simplify_button) {
1594        if (overlaid)
1595           overlay_displays (); /* reset overlay */
1596        switch (simplification_value) {
1597           case SIMP_NTH_PT:
1598              break;
1599           case SIMP_RANDOM_PT:
1600              break;
1601           case SIMP_LINE_WIDTH:
1602              break;
1603           case SIMP_EUCLIDEAN:
1604              break;
1605           case SIMP_PERPENDIC:
1606              break;
1607           case SIMP_ANGULAR:
1608              break;
1609           case SIMP_DIST_ANGLE:
1610              break;
1611           case SIMP_REUMAN:
1612              break;
1613           case SIMP_ROBERGE:
1614              break;
1615           case SIMP_LANG:
1616              break;
1617           case SIMP_JOHANNSEN:
1618              break;
1619           case SIMP_OPHEIM:
1620              break;
1621           case SIMP_DOUGLAS:
1622              do_douglas_peucker (event);
1623              break;
1624           default:
1625              break;
1626        } /* switch */
1627     } /* simplify */
1628     else
1629     if (item == smoothe_button) {
1630        if (overlaid)
1631           overlay_displays (); /* reset overlay */
1632        switch (smoothing_value) {
1633           case SMOO_SIMPLE_AVE:
1634              break;
1635           case SMOO_WEIGHT_AVE:
1636              break;
1637           case SMOO_FWD_LOOK:
1638              break;
1639           case SMOO_PERKALS:
1640              break;
1641           case SMOO_BROPHYS:
1642              break;
1643           case SMOO_CUBIC_SP:
```

```
1644                break;
1645             case SMOO_PARAB_SP:
1646                break;
1647             case SMOO_B_SPLINE:
1648                break;
1649             case SMOO_BEZIER_CUR:
1650                break;
1651             default:
1652                break;
1653          } /* switch */
1654       } /* smoothe */
1655       else
1656       if (item == measure_button)
1657          switch (measurement_value) {
1658             case MEAS_ABS:
1659                do_measure_absolutes ();
1660                break;
1661             case MEAS_ANG:
1662                do_measure_angularity ();
1663                break;
1664             case MEAS_SIN:
1665                do_measure_sinuosity ();
1666                break;
1667             default:
1668                break;
1669          } /* switch */
1670
1671    } /* do_process */
1672
```

```
1673  /*-----------------------------------------------------------------------*/
1674
1675  void
1676  do_quit ()
1677
1678  /*
1679   *  Quit the shell
1680   */
1681
1682  {
1683     /* quit with user confirmation */
1684     window_destroy (control_frame);
1685
1686  } /* do_quit */
1687
```

```
1688  /*-----------------------------------------------------------------------*/
1689
1690  void
1691  do_reset ()
1692
1693  /*
1694   *  Reset the control method settings to the default values, reset measurements
1695   */
1696
1697  {
1698     panel_set (current_simplification, PANEL_LABEL_STRING, SIMP_DEFAULT, 0);
1699     panel_set (current_smoothing     , PANEL_LABEL_STRING, SMOO_DEFAULT, 0);
1700     panel_set (current_measurement   , PANEL_LABEL_STRING, MEAS_DEFAULT, 0);
1701     simplification_value = SIMP_DEFAULT_VALUE;
1702     smoothing_value      = SMOO_DEFAULT_VALUE;
1703     measurement_value    = MEAS_DEFAULT_VALUE;
1704
1705     for (i = 0; i < MAX_MEASURES; i++)
1706        panel_set (measurement_line[i], PANEL_LABEL_STRING, "", 0);
1707
1708  } /* do_reset */
1709
```

```
1710  /*-----------------------------------------------------------------------*/
1711
1712  void
1713  do_simplification_choice (menu, menu_item, np)
1714     Menu                 menu;
1715     Menu_item            menu_item;
1716     caddr_t              (*np) ();
1717
1718  /*
1719   *  Set up the desired simplification method
1720   */
1721
1722  {
1723     menu_get (menu_item, MENU_VALUE); /* Force evaluation of pullrights */
1724     simplification_value = (int) menu_get (menu_item, MENU_VALUE);
1725
1726     /* display the simplification method on the control panel */
1727     panel_set (current_simplification, PANEL_LABEL_STRING,
1728        menu_get (menu_item, MENU_STRING), 0);
1729
1730  } /* do_simplification_choice */
1731
```

```
1732  /*-----------------------------------------------------------------------*/
1733
1734  void
1735  do_smoothing_choice (menu, menu_item, np)
1736     Menu                 menu;
1737     Menu_item            menu_item;
1738     caddr_t              (*np) ();
1739
1740  /*
1741   *  Set up the desired smoothing method
1742   */
1743
1744  {
1745     menu_get (menu_item, MENU_VALUE); /* Force evaluation of pullrights */
1746     smoothing_value = (int) menu_get (menu_item, MENU_VALUE);
1747
1748     /* display the smoothing method on the control panel */
1749     panel_set (current_smoothing, PANEL_LABEL_STRING,
1750        menu_get (menu_item, MENU_STRING), 0);
1751
1752  } /* do_smoothing_choice */
1753
```

```
1754  /*-----------------------------------------------------------------------*/
1755
1756  void
1757  draw_canvas (pw, input)
1758     Pixwin          *pw;
1759     int             input;
1760
1761  {
1762     if (input)
1763        for (i = 0; i < MAX_COORDS && icoord[i][X] >= 0 &&
1764                                      icoord[i][Y] >= 0; i++) {
1765           draw_point (pw, icoord[i][X], icoord[i][Y], IN_POINT_COLOR);
1766           if (i > 0)
1767              pw_vector (pw, icoord[i-1][X], icoord[i-1][Y],
1768                             icoord[i  ][X], icoord[i  ][Y],
1769                             PIX_SRC, IN_LINE_COLOR);
1770        }
1771     else
1772        for (i = 0; i < MAX_COORDS && ocoord[i][X] >= 0 &&
1773                                      ocoord[i][Y] >= 0; i++) {
1774           draw_point (pw, ocoord[i][X], ocoord[i][Y], OUT_POINT_COLOR);
1775           if (i > 0)
1776              pw_vector (pw, ocoord[i-1][X], ocoord[i-1][Y],
1777                             ocoord[i  ][X], ocoord[i  ][Y],
1778                             PIX_SRC, OUT_LINE_COLOR);
1779        }
1780
1781  } /* draw_canvas */
1782
```

```
1783  /*-----------------------------------------------------------------------*/
1784
1785  void
1786  draw_point (pw, x, y, color)
1787     Pixwin            *pw;
1788     int               x, y, color;
1789
1790  /*
1791   *  Draw a 3x3 pixel square of color around x,y on pw to hilite a geographic point
1792   */
1793
1794  {
1795     pw_vector (pw, x-1, y-1, x  , y-1, PIX_SRC, color);
1796     pw_vector (pw, x+1, y-1, x+1, y  , PIX_SRC, color);
1797     pw_vector (pw, x+1, y+1, x  , y+1, PIX_SRC, color);
1798     pw_vector (pw, x-1, y+1, x-1, y  , PIX_SRC, color);
1799
1800  } /* draw_point */
1801
```

```
1802  /*-----------------------------------------------------------------------*/
1803
1804  Panel_setting
1805  enter_coord_char (item, event)
1806     Panel_item        item;
1807     Event             *event;
1808
1809  /*
1810   *  Read & process the latest coordinate character entered by the user
1811   */
1812
1813  {
1814     char              value[VALUE_LEN];
1815
1816     switch ( event_id (event) ) {
1817        case '0':
1818        case '1':
1819        case '2':
1820        case '3':
1821        case '4':
1822        case '5':
1823        case '6':
1824        case '7':
1825        case '8':
1826        case '9':
1827           return (PANEL_INSERT); /* Accept it */
1828           break;
1829        case '\n': /* "new line" */
1830        case '\r': /* "return" */
1831        case '\t': /* "tab" */
1832           enter_in_coordinate (row, col, (int) panel_get_value (item) );
1833           /* allow input "save" option */
1834           panel_set (coor_in_save_button, PANEL_SHOW_ITEM, TRUE, 0);
1835           if ( event_shift_is_down (event) )
1836              if (col > X) {
1837                 col--;
1838                 return (PANEL_PREVIOUS); /* previous cell */
1839              }
1840              else
1841                 if (row == 0)
1842                    return (PANEL_NONE); /* don't backtrack! */
1843                 else {
1844                    col = Z;
1845                    row--;
1846                    max_reached = FALSE; /* reset if set */
1847                    return (PANEL_PREVIOUS); /* previous cell */
1848                 }
1849           else /* shift key NOT used */
1850              if (col < Z) {
1851                 col++;
1852                 return (PANEL_NEXT); /* next cell */
1853              }
1854              else
1855                 if (row == MAX_COORDS-1) {
1856                    max_reached = TRUE;
1857                    return (PANEL_NONE); /* don't advance! */
1858                 }
1859                 else {
1860                    row++;
1861                    col = X;
1862                    return (PANEL_NEXT); /* next cell */
1863                 }
1864           break;
```

```
1865        default:
1866           return (PANEL_NONE); /* Ignore it! */
1867           break;
1868     } /* switch */
1869
1870  } /* enter_coord_char */
1871
```

```
1872  /*-----------------------------------------------------------------------*/
1873
1874  void
1875  enter_in_coordinate (i, j, value)
1876     int            i, j, value;
1877
1878  /*
1879   *  Enter a coordinate value in an input coordinate cell
1880   */
1881
1882  {
1883     int            save[3];
1884     char           string[VALUE_LEN];
1885
1886     if (j == Y && i > 0)
1887        if (icoord[i][X] == icoord[i-1][X] &&
1888                value == icoord[i-1][Y]) { /* duplicate coordinate - eliminate it! */
1889           panel_set_value (coord_in_cell[i][X], "");
1890           panel_set_value (coord_in_cell[i][Z], "");
1891           panel_set (coord_in_panel, PANEL_CARET_ITEM, coord_in_cell[i][X], 0);
1892           icoord[i][X] = -1;
1893           row--;
1894           col = X;
1895           return;
1896        }
1897
1898     if (!max_reached) { /* enter it */
1899        for (k = X; k <= Z; k++)
1900           save[k] = icoord[i][k];
1901        icoord[i][j] = value;
1902        /* provide right-justified feedback to user */
1903        sprintf (string, "%5u", value);
1904        panel_set_value (coord_in_cell[i][j], string);
1905
1906        pw_lock (input_pw, &canvas_rect);
1907
1908        if (save[X] >= 0 && save[Y] >= 0) /* erase old point */
1909           draw_point (input_pw, save[X], save[Y], BACKGROUND);
1910
1911        if (icoord[i][X] >= 0 && icoord[i][Y] >= 0) /* draw new point */
1912           draw_point (input_pw, icoord[i][X], icoord[i][Y], IN_POINT_COLOR);
1913
1914        if (i > 0)
1915           if (icoord[i-1][X] >= 0 && icoord[i-1][Y] >= 0) { /* there is a prev point */
1916              /* check to erase old line between old and preceding coordinates */
1917              if (save[X] >= 0 && save[Y] >= 0)
1918                 pw_vector  (input_pw, icoord[i-1][X], icoord[i-1][Y],
1919                                       save       [X], save       [Y],
1920                                       PIX_SRC, BACKGROUND);
1921              /* draw line between new and preceding coordinates */
1922              if (icoord[i][X] >= 0 && icoord[i][Y] >= 0)
1923                 pw_vector  (input_pw, icoord[i-1][X], icoord[i-1][Y],
1924                                       icoord[i  ][X], icoord[i  ][Y],
1925                                       PIX_SRC, IN_LINE_COLOR);
1926           }
1927
1928        if (i < MAX_COORDS-1)
1929           if (icoord[i+1][X] >= 0 && icoord[i+1][Y] >= 0) { /* there is a next point */
1930              if (save[X] >= 0 || save[Y] >= 0)
1931                 pw_vector  (input_pw, icoord[i+1][X], icoord[i+1][Y],
1932                                       save       [X], save       [Y],
1933                                       PIX_SRC, BACKGROUND);
1934              /* check to draw new line between new and following coordinates */
```

```
1935              if (icoord[i][X] >= 0 && icoord[i][Y] >= 0)
1936                 pw_vector  (input_pw, icoord[i  ][X], icoord[i  ][Y],
1937                                       icoord[i+1][X], icoord[i+1][Y],
1938                                       PIX_SRC, IN_LINE_COLOR);
1939           }
1940
1941        pw_unlock (input_pw);
1942
1943     } /* max not reached */
1944
1945     if ( (int) panel_get (coor_in_load_button, PANEL_SHOW_ITEM) ) {
1946        /* eliminate input "load" option, allow "clear" and "save" options */
1947        panel_set (coor_in_load_button , PANEL_SHOW_ITEM, FALSE, 0);
1948        panel_set (coor_in_clear_button, PANEL_SHOW_ITEM, TRUE , 0);
1949     }
1950
1951  } /* enter_in_coordinate */
1952
```

```
1953  /*-----------------------------------------------------------------------*/
1954
1955  void
1956  enter_new_point (event)
1957     Event          *event;
1958
1959  /*
1960   *  Set the icoord[row] cells to the X & Y coordinates of the point picked
1961   */
1962
1963  {
1964     char           value[VALUE_LEN];
1965
1966     if ( (event_id (event) == LOC_DRAG) || (event_is_down (event) ) ) {
1967
1968        enter_in_coordinate (row, X, event_x (event) );
1969        enter_in_coordinate (row, Y, event_y (event) );
1970        enter_in_coordinate (row, Z, 0);
1971
1972        if (row == MAX_COORDS-1)
1973           max_reached = TRUE;
1974        else { /* Advance the caret to the next panel line */
1975           col = X;
1976           panel_set (coord_in_panel, PANEL_CARET_ITEM, coord_in_cell[++row][col], 0);
1977        }
1978
1979        /* allow input "save" option */
1980        panel_set (coor_in_save_button, PANEL_SHOW_ITEM, TRUE, 0);
1981     }
1982
1983  } /* enter_new_point */
1984
```

```
1985  /*-----------------------------------------------------------------------*/
1986
1987  void
1988  enter_out_coordinates (i)
1989     int                i;
1990
1991  /*
1992   *  Enter coordinate values in output coordinate cells
1993   */
1994
1995  {
1996     char            string[VALUE_LEN];
1997
1998     pw_lock (output_pw, &canvas_rect);
1999
2000     if (ocoord[i][X] >= 0 && ocoord[i][Y] >= 0) /* draw the point */
2001        draw_point (output_pw, ocoord[i][X], ocoord[i][Y], OUT_POINT_COLOR);
2002
2003     if (i > 0)
2004        /* check if a line can be drawn between the previous two coordinates */
2005        if ( (ocoord[i-1][X] >= 0 && ocoord[i-1][Y] >= 0) &&
2006             (ocoord[i  ][X] >= 0 && ocoord[i  ][Y] >= 0) )
2007           pw_vector (output_pw,  ocoord[i-1][X], ocoord[i-1][Y],
2008                                  ocoord[i  ][X], ocoord[i  ][Y],
2009                                  PIX_SRC, OUT_LINE_COLOR);
2010
2011     pw_unlock (output_pw);
2012
2013     /* provide right-justified feedback to user in coordinate panel */
2014     for (j = X; j <= Z; j++) {
2015        sprintf (string, "%5u", ocoord[i][j]);
2016        panel_set (coord_out_cell[i][j], PANEL_LABEL_STRING, string, 0);
2017     }
2018
2019     if ( !(int) panel_get (coor_out_clear_button, PANEL_SHOW_ITEM) ) {
2020        /* allow output "clear" and "save" options */
2021        panel_set (coor_out_clear_button, PANEL_SHOW_ITEM, TRUE, 0);
2022        panel_set (coor_out_save_button , PANEL_SHOW_ITEM, TRUE, 0);
2023     }
2024
2025  } /* enter_out_coordinates */
2026
```

```
2027  /*-------------------------------------------------------------------------*/
2028
2029  void
2030  file_i_o (item, event)
2031     Panel_item        item;
2032     Event             *event;
2033
2034  /*
2035   *  Load/save the coordinates from/to specified file
2036   */
2037
2038  {
2039     static
2040     char              *enter_msg  = "Enter a file name.",
2041                       *open_msg   = "File OPEN error.",
2042                       *loaded_msg = "File loaded.",
2043                       *saved_msg  = "File saved.";
2044     FILE              *file_ptr;
2045     char              *path,
2046                       *file,
2047                       name[80];
2048
2049     name[0] = '\0'; /* reset */
2050     if (item == coor_in_load_button) {
2051        path = (char *) panel_get_value (path_in_item);
2052        file = (char *) panel_get_value (file_in_item);
2053        if (file == "") {
2054           panel_set (file_in_message, PANEL_LABEL_STRING, enter_msg, 0);
2055           window_bell (file_in_panel);
2056        }
2057        else {
2058           strcat (name, path);
2059           strcat (name, file);
2060           if ( (file_ptr = fopen (name, read_mode) ) == NULL) {
2061              /* open error of some sort */
2062              panel_set (file_in_message, PANEL_LABEL_STRING, open_msg, 0);
2063              window_bell (file_in_panel);
2064           }
2065           else {
2066              int   io_result = 0;
2067              for (row = 0; row < MAX_COORDS && io_result != EOF; row++) {
2068                 io_result = fscanf (file_ptr, in_format,
2069                                     &icoord[row][X], &icoord[row][Y], &icoord[row][Z]);
2070                 if (io_result != EOF) { /* enter & draw coordinates */
2071                    enter_in_coordinate (row, X, icoord[row][X]);
2072                    enter_in_coordinate (row, Y, icoord[row][Y]);
2073                    enter_in_coordinate (row, Z, icoord[row][Z]);
2074                 }
2075              }
2076              fclose (file_ptr);
2077              if (io_result != EOF)
2078                 max_reached = TRUE;
2079              row--; /* adjust after loop */
2080              panel_set (coord_in_panel, PANEL_CARET_ITEM, coord_in_cell[row][X], 0);
2081              panel_set (file_in_message, PANEL_LABEL_STRING, loaded_msg, 0);
2082           }
2083        }
2084     } /* coor_in_load_button */
2085     else
2086     if (item == coor_in_save_button) {
2087        path = (char *) panel_get_value (path_in_item);
2088        file = (char *) panel_get_value (file_in_item);
2089        if (file == "") {
```

```
2090            panel_set (file_in_message, PANEL_LABEL_STRING, enter_msg, 0);
2091            window_bell (file_in_panel);
2092         }
2093         else {
2094            strcat (name, path);
2095            strcat (name, file);
2096            if ( (file_ptr = fopen (name, write_mode) ) == NULL) {
2097               /* open error of some sort */
2098               panel_set (file_in_message, PANEL_LABEL_STRING, open_msg, 0);
2099               window_bell (file_in_panel);
2100            }
2101            else {
2102               for (i = 0; icoord[i][X] >= 0; i++)
2103                  fprintf (file_ptr, out_format,
2104                           icoord[i][X], icoord[i][Y], icoord[i][Z], '\n');
2105               fclose (file_ptr);
2106               /* eliminate the input "save" option */
2107               panel_set (coor_in_save_button, PANEL_SHOW_ITEM, FALSE, 0);
2108               panel_set (file_in_message, PANEL_LABEL_STRING, saved_msg, 0);
2109            }
2110         }
2111      } /* coor_in_save_button */
2112      else {
2113         /* coor_out_save_button */
2114         path = (char *) panel_get_value (path_out_item);
2115         file = (char *) panel_get_value (file_out_item);
2116         if (file == "") {
2117            panel_set (file_out_message, PANEL_LABEL_STRING, enter_msg, 0);
2118            window_bell (file_out_panel);
2119         }
2120         else {
2121            strcat (name, path);
2122            strcat (name, file);
2123            if ( (file_ptr = fopen (name, write_mode) ) == NULL) {
2124               /* open error of some sort */
2125               panel_set (file_out_message, PANEL_LABEL_STRING, open_msg, 0);
2126               window_bell (file_out_panel);
2127            }
2128            else {
2129               for (i = 0; ocoord[i][X] >= 0; i++)
2130                  fprintf (file_ptr, out_format,
2131                           ocoord[i][X], ocoord[i][Y], ocoord[i][Z], '\n');
2132               fclose (file_ptr);
2133               /* eliminate the output "save" option */
2134               panel_set (coor_out_save_button, PANEL_SHOW_ITEM, FALSE, 0);
2135               panel_set (file_out_message, PANEL_LABEL_STRING, saved_msg, 0);
2136            }
2137         }
2138      }
2139
2140   } /* file_i_o */
2141
```

```
2142  /*-----------------------------------------------------------------------*/
2143
2144  void
2145  handle_canvas_event (canvas, event, arg)
2146     Canvas                canvas;
2147     Event                 *event;
2148     caddr_t               arg;
2149
2150  /*
2151   *  Handle canvas pick event
2152   */
2153
2154  {
2155     switch ( event_id (event) ) {
2156        case MS_RIGHT:
2157           input = (canvas == input_canvas) ? TRUE : FALSE;
2158           menu_show (display_menu, canvas, event, 0);
2159           break;
2160        case MS_LEFT:
2161        case LOC_DRAG:
2162           if (canvas == input_canvas)
2163              enter_new_point (event);
2164           break;
2165        default:
2166           break;
2167     } /* switch */
2168
2169  } /* handle_canvas_event */
2170
```

```
2171  /*-----------------------------------------------------------------------*/
2172
2173  void
2174  locate_item (item, event)
2175     Panel_item          item;
2176     Event               *event;
2177
2178  /*
2179   *  Locate newly picked input coordinate cell, set row and col to its position
2180   */
2181
2182  {
2183     /* perform standard processing first */
2184     panel_default_handle_event (item, event);
2185
2186     if (event_id (event) == MS_LEFT)
2187        for (row = 0; row < MAX_COORDS; row++) {
2188           for (col = X; col <= Z; col++)
2189              if (item == coord_in_cell[row][col])
2190                 break;
2191           if (item == coord_in_cell[row][col])
2192              break;
2193        } /* for row */
2194
2195  } /* locate_item */
2196
```

```
2197  /*-------------------------------------------------------------------------*/
2198
2199  void
2200  make_color_map ()
2201
2202  /*
2203   *  Define color map for canvases
2204   */
2205
2206  {
2207     cms.cms_size = COLOR_MAP_SIZE;
2208     strcpy (cms.cms_name, "linesimp colors");
2209
2210     map.cm_red   = RGB[R];
2211     map.cm_green = RGB[G];
2212     map.cm_blue  = RGB[B];
2213
2214     pw_setdefaultcms (&cms, &map); /* set default color map to ours */
2215
2216     set_color (BACKGROUND     , GRAY   );
2217     set_color (FOREGROUND     , CYAN   );
2218     set_color ( IN_POINT_COLOR, GREEN  );
2219     set_color (  IN_LINE_COLOR, RED    );
2220     set_color (  IN_AREA_COLOR, MAGENTA);
2221     set_color (OUT_POINT_COLOR, YELLOW );
2222     set_color ( OUT_LINE_COLOR, BLUE   );
2223     set_color ( OUT_AREA_COLOR, WHITE  );
2224
2225     pw_setcmsname  ( input_pw, cms.cms_name);
2226     pw_putcolormap ( input_pw, 0, COLOR_MAP_SIZE, RGB[R], RGB[G], RGB[B]);
2227     pw_setcmsname  (output_pw, cms.cms_name);
2228     pw_putcolormap (output_pw, 0, COLOR_MAP_SIZE, RGB[R], RGB[G], RGB[B]);
2229
2230  } /* make_color_map */
2231
```

```
2232  /*-----------------------------------------------------------------------*/
2233
2234  void
2235  ok_button (item, event)
2236     Panel_item           item;
2237     Event                *event;
2238
2239  /*
2240   *  OK button has been pressed in tolerance popup
2241   */
2242
2243  {
2244     window_return ( panel_get_value (tolerance_text_item) );
2245  }
2246
```

```
2247  /*-----------------------------------------------------------------------*/
2248
2249  void
2250  overlay_displays ()
2251
2252  /*
2253   *  Overlay the output canvas on top of the input canvas, or reset the input canvas
2254   */
2255
2256  {
2257     if (!overlaid) { /* overlay them */
2258        if (ocoord[0][X] < 0) /* nothing to overlay! */
2259           return;
2260        pw_lock (input_pw, &canvas_rect);
2261        draw_canvas (input_pw, FALSE);
2262        pw_unlock (input_pw);
2263        panel_set (disp_overlay_button, PANEL_LABEL_IMAGE, no_overlay_button_image, 0);
2264        overlaid = TRUE;
2265     }
2266     else { /* erase */
2267        pw_lock (input_pw, &canvas_rect);
2268        pw_writebackground (input_pw, 0, 0, CANVAS_MAX_X, CANVAS_MAX_Y, PIX_SRC);
2269        draw_canvas (input_pw, TRUE);
2270        pw_unlock (input_pw);
2271        panel_set (disp_overlay_button, PANEL_LABEL_IMAGE, overlay_button_image, 0);
2272        overlaid = FALSE;
2273     }
2274
2275  } /* overlay_displays */
2276
```

```
2277  /*-----------------------------------------------------------------------*/
2278
2279  void
2280  set_color (index, color)
2281     int         index, color;
2282
2283  {
2284     switch (color) {
2285        case WHITE:
2286           RGB[R][index] = 255;
2287           RGB[G][index] = 255;
2288           RGB[B][index] = 255;
2289           break;
2290        case RED:
2291           RGB[R][index] = 255;
2292           RGB[G][index] =   0;
2293           RGB[B][index] =   0;
2294           break;
2295        case GREEN:
2296           RGB[R][index] =   0;
2297           RGB[G][index] = 255;
2298           RGB[B][index] =   0;
2299           break;
2300        case BLUE:
2301           RGB[R][index] =   0;
2302           RGB[G][index] =   0;
2303           RGB[B][index] = 255;
2304           break;
2305        case YELLOW:
2306           RGB[R][index] = 255;
2307           RGB[G][index] = 255;
2308           RGB[B][index] =   0;
2309           break;
2310        case CYAN:
2311           RGB[R][index] =   0;
2312           RGB[G][index] = 255;
2313           RGB[B][index] = 255;
2314           break;
2315        case MAGENTA:
2316           RGB[R][index] = 255;
2317           RGB[G][index] =   0;
2318           RGB[B][index] = 255;
2319           break;
2320        case LIGHT_RED:
2321           RGB[R][index] = 255;
2322           RGB[G][index] = 225;
2323           RGB[B][index] = 225;
2324           break;
2325        case LIGHT_GREEN:
2326           RGB[R][index] = 225;
2327           RGB[G][index] = 255;
2328           RGB[B][index] = 225;
2329           break;
2330        case LIGHT_BLUE:
2331           RGB[R][index] = 225;
2332           RGB[G][index] = 225;
2333           RGB[B][index] = 255;
2334           break;
2335        case GRAY:
2336           RGB[R][index] = 128;
2337           RGB[G][index] = 128;
2338           RGB[B][index] = 128;
2339           break;
```

```
2340        case BLACK:
2341           RGB[R][index] =   0;
2342           RGB[G][index] =   0;
2343           RGB[B][index] =   0;
2344           break;
2345        default:
2346           break;
2347     } /* switch */
2348
2349  } /* set_color */
2350
```

```
2351  /*-----------------------------------------------------------------------*/
2352
2353  void
2354  show_button_menu (item, event)
2355     Panel_item           item;
2356     Event                *event;
2357
2358  /*
2359   *  Display appropriate control panel menu upon RIGHT mouse button down
2360   */
2361
2362  {
2363     if (event_id (event) == MS_RIGHT)
2364        if (item == simplify_button ||
2365            item == simplification_method ||
2366            item == current_simplification)
2367           menu_show (simplification_menu, control_panel, event, 0);
2368        else
2369        if (item == smoothe_button ||
2370            item == smoothing_method ||
2371            item == current_smoothing)
2372           menu_show (smoothing_menu, control_panel, event, 0);
2373        else
2374        if (item == measure_button ||
2375            item == measurement_method ||
2376            item == current_measurement)
2377           menu_show (measurement_menu, control_panel, event, 0);
2378        else
2379           panel_default_handle_event (item, event);
2380     else
2381        panel_default_handle_event (item, event);
2382
2383  } /* show_button_menu */
2384
```

```
2385  /*-----------------------------------------------------------------------*/
2386
2387  void
2388  show_hide_coordinates (item, event)
2389     Panel_item        item;
2390     Event             *event;
2391
2392  /*
2393   *  Show or hide the coordinates as currently appropriate
2394   */
2395
2396  {
2397     if (window_get (coordinate_frame, WIN_SHOW) == FALSE) /* show! */ {
2398        window_set (coordinate_frame, WIN_SHOW, TRUE, 0);
2399        panel_set (coor_show_hide_button, PANEL_LABEL_IMAGE, hide_button_image, 0);
2400     }
2401     else /* hide! */ {
2402        window_set (coordinate_frame, WIN_SHOW, FALSE, 0);
2403        panel_set (coor_show_hide_button, PANEL_LABEL_IMAGE, show_button_image, 0);
2404     }
2405
2406  } /* show_hide_coordinates */
2407
```

```
2408  /*-----------------------------------------------------------------------*/
2409
2410  void
2411  show_hide_displays ()
2412
2413  /*
2414   *  Show or hide the display canvases as currently appropriate
2415   */
2416
2417  {
2418     if (window_get (display_frame, WIN_SHOW) == FALSE) /* show! */ {
2419        window_set (display_frame, WIN_SHOW, TRUE, 0);
2420        panel_set (disp_show_hide_button, PANEL_LABEL_IMAGE, hide_button_image, 0);
2421     }
2422     else /* hide! */ {
2423        window_set (display_frame, WIN_SHOW, FALSE, 0);
2424        panel_set (disp_show_hide_button, PANEL_LABEL_IMAGE, show_button_image, 0);
2425     }
2426
2427  } /* show_hide_displays */
```

```
/*-----------------------------------------------------------------------*
 *  Line simplification (generalization) process shell
 *  by Yvon Perreault, PAR Government Systems Corp.
 *  April-May 1987.
 *-----------------------------------------------------------------------*/

#include <suntool/sunview.h>
#include <suntool/panel.h>
#include <suntool/canvas.h>
#include <suntool/scrollbar.h>
#include <stdio.h>
#include <math.h>

/*
 *  Simplification menu constants
 */
#define SIMP_NTH_PT      11
#define SIMP_RANDOM_PT   12
#define SIMP_LINE_WIDTH  21
#define SIMP_EUCLIDEAN   22
#define SIMP_PERPENDIC   23
#define SIMP_ANGULAR     24
#define SIMP_DIST_ANGLE  25
#define SIMP_REUMAN      31
#define SIMP_ROBERGE     32
#define SIMP_LANG        41
#define SIMP_JOHANNSEN   42
#define SIMP_OPHEIM      43
#define SIMP_DOUGLAS     51

/*
 *  Smoothing menu constants
 */
#define SMOO_SIMPLE_AVE  11
#define SMOO_WEIGHT_AVE  12
#define SMOO_FWD_LOOK    13
#define SMOO_PERKALS     21
#define SMOO_BROPHYS     22
#define SMOO_CUBIC_SP    31
#define SMOO_PARAB_SP    32
#define SMOO_B_SPLINE    33
#define SMOO_BEZIER_CUR  34

/*
 *  Measurement menu constants
 */
#define MEAS_ABS         2
#define MEAS_ANG         3
#define MEAS_SIN         4

/*
 *  Display menu constants
 */
#define DISP_CLEAR       1
#define DISP_ZOOM        2
#define DISP_COLOR       3

/*
 *  Control panel constants
 */
#define CONTROL_WIDTH_1 36
#define CONTROL_WIDTH_2 30
#define SIMP_ROW         0
```

```
64  #define SMOO_ROW          1
65  #define MEAS_ROW          2
66  #define BUTTON_ROW        3
67  #define DISP_ROW          0
68  #define COOR_ROW          1
69  #define SIMP_DEFAULT      "Douglas-Peucker"
70  #define SMOO_DEFAULT      "None"
71  #define MEAS_DEFAULT      "Absolutes"
72  #define SIMP_DEFAULT_VALUE SIMP_DOUGLAS
73  #define SMOO_DEFAULT_VALUE SMOO_NONE
74  #define MEAS_DEFAULT_VALUE MEAS_ABS
75  #define MAX_MEASURES      5
76  #define DPI               87.0      /* rounded # pixels in 1 inch ("Dots Per Inch") */
77  #define TOL_DEFAULT       "10"
78
79  /*
80   *  Graphics canvases constants
81   */
82  #define CANVAS_MAX_X      1000
83  #define CANVAS_MAX_Y      1000
84  #define INIT_WIDTH        500
85  #define INIT_HEIGHT       500
86
87  /*
88   *  Color map constants
89   */
90  #define R                 0
91  #define G                 1
92  #define B                 2
93  #define COLOR_MAP_SIZE    8
94  #define BACKGROUND        0
95  #define FOREGROUND        1
96  #define IN_POINT_COLOR    2
97  #define IN_LINE_COLOR     3
98  #define IN_AREA_COLOR     4
99  #define OUT_POINT_COLOR   5
100 #define OUT_LINE_COLOR    6
101 #define OUT_AREA_COLOR    7
102 /*
103  *  Color menu constants
104  */
105 #define WHITE             1
106 #define GREEN             2
107 #define RED               3
108 #define BLUE              4
109 #define YELLOW            5
110 #define CYAN              6
111 #define MAGENTA           7
112 #define BLACK             8
113 #define GRAY              9
114 #define LIGHT_RED         10
115 #define LIGHT_GREEN       11
116 #define LIGHT_BLUE        12
117
118 /*
119  *  Coordinate panel constants
120  */
121 #define MAX_COORDS        250
122 #define COORDS_COLS       32
123 #define COORDS_ROWS       25
124 #define LABEL_LEN         4
125 #define VALUE_LEN         5
126 #define NAME_LEN          25
```

```
127  #define X                0
128  #define Y                1
129  #define Z                2
130
131  /*
132   *  Scrollbar constants
133   */
134  #define VERTICAL_LOC     SCROLL_EAST
135  #define HORIZONTAL_LOC   SCROLL_SOUTH
136  #define BUBBLE_MARGIN    1
137
138      static
139      Frame        control_frame,
140                   coordinate_frame,
141                   display_frame,
142                   tolerance_popup;
143
144      static
145      Panel        control_panel,
146                   bottom_panel,
147                   measurements_panel,
148                   file_in_panel,
149                   file_out_panel,
150                   coord_in_panel,
151                   coord_out_panel,
152                   tolerance_panel;
153
154      static
155      Panel_item   simplify_button,
156                   simplification_method,
157                   current_simplification,
158                   smooths_button,
159                   smoothing_method,
160                   current_smoothing,
161                   measure_button,
162                   measurement_method,
163                   current_measurement,
164                   reset_button,
165                   quit_button,
166
167                   display_title,
168                   disp_show_hide_button,
169                   disp_overlay_button,
170                   coordinate_title,
171                   coor_show_hide_button,
172
173                   measurement_line[MAX_MEASURES],
174
175                   path_in_item,
176                   file_in_item,
177                   file_in_message,
178                   coor_in_load_button,
179                   coor_in_clear_button,
180                   coor_in_save_button,
181                   input_header,
182                   coord_in_header,
183                   coord_in_label[MAX_COORDS],
184                   coord_in_cell [MAX_COORDS][3],
185                   coord_in_ender[MAX_COORDS],
186
187                   path_out_item,
188                   file_out_item,
189                   file_out_message,
```

```
190                 coor_out_clear_button,
191                 coor_out_save_button,
192                 output_header,
193                 coord_out_header,
194                 coord_out_label[MAX_COORDS],
195                 coord_out_start[MAX_COORDS][3],
196                 coord_out_cell [MAX_COORDS][3],
197                 coord_out_ender[MAX_COORDS],
198
199                 tolerance_text_item,
200                 tolerance_ok_button;
201
202     static
203     Canvas       input_canvas,
204                 output_canvas;
205
206     static
207     Pixwin      * input_pw,
208                 *output_pw;
209
210     static struct
211     pixrect     *simplify_button_image,
212                 *smoothe_button_image,
213                 *measure_button_image,
214                 *reset_button_image,
215                 *quit_button_image,
216                 *show_button_image,
217                 *hide_button_image,
218                 *overlay_button_image,
219                 *no_overlay_button_image,
220                 *load_button_image,
221                 *clear_button_image,
222                 *save_button_image;
223
224     static struct
225     rect        canvas_rect = {0, 0, CANVAS_MAX_X, CANVAS_MAX_Y};
226
227     static
228     Cursor      coord_cursor,
229                 draw_cursor;
230
231     static
232     Menu        simplification_menu,
233                 simp_indep_pt_menu,
234                 simp_local_menu,
235                 simp_uncons_local_menu,
236                 simp_cons_local_menu,
237                 simp_global_menu,
238                 smoothing_menu,
239                 smoo_averaging_menu,
240                 smoo_epsilon_menu,
241                 smoo_splining_menu,
242                 smoo_splining_local_menu,
243                 smoo_splining_extended_menu,
244                 smoo_splining_global_menu,
245                 measurement_menu,
246                 angular_measure_menu,
247                 sinuous_measure_menu,
248                 display_menu,
249                 color_types_menu,
250                 back_color_menu,
251                 color_menu;
252
```

```
253     static
254     Icon            linesimp_icon;
255
256     int             i, j, k,
257                     row = 0,
258                     col = 0,
259                     max_reached = FALSE,
260                     overlaid = FALSE,
261                     input,
262                     choice,
263                     simplification_value,
264                     smoothing_value,
265                     measurement_value,
266                     icoord[MAX_COORDS][3],
267                     ocoord[MAX_COORDS][3];
268
269     static
270     char            * in_format = "%5u %5u %5u",
271                     *out_format = "%5u %5u %5u %c",
272                     *read_mode  = "r",
273                     *write_mode = "w",
274                     *coord_column_header = "Coord# --X--   --Y--   --Z--";
275
276     static
277     struct          measures {
278        float        total_length,
279                     total_angularity,
280                     right_angularity,
281                     left_angularity,
282                     std_angularity_inch,
283                     num_coordinates,
284                     total_runs;
285                             };
286
287     static struct
288     singlecolor
289                     control_bg_color = {255, 255, 255}, /* white */
290                     control_fg_color = {000, 000, 255}, /* blue */
291                     popup_bg_color   = {255, 255, 255}, /* white */
292                     popup_fg_color   = {255, 000, 000}; /* red */
293
294     static struct
295     colormapseg
296                     cms;
297
298     static struct
299     cms_map
300                     map;
301
302     static
303     unsigned char
304                     RGB[3][COLOR_MAP_SIZE];
305
306     static
307     short           hairs_image[256] = {
308 #include "../cursors/hairs"
309                                         };
310     mpr_static (hairs_pixrect, 16, 16, 1, hairs_image);
311
312     static
313     short           cross_image[256] = {
314 #include "../cursors/cross"
315                                         };
```

```
316     mpr_static (cross_pixrect, 16, 16, 1, cross_image);
317
318     static
319     short       icon_image[256] = {
320 #include "../icons/linesimp"
321                                 };
322     mpr_static (icon_pixrect, ICON_DEFAULT_WIDTH, ICON_DEFAULT_HEIGHT, 1, icon_image);
323
324     /* Internal procedures & functions */
325     double              calc_distance                   ();
326     void                clear_coordinates               ();
327     void                define_menus                    ();
328     void                define_windows                  ();
329     void                do_color_choice                 ();
330     void                do_display_choice               ();
331     void                do_done                         ();
332     void                do_douglas_peucker              ();
333     void                do_measure_absolutes            ();
334     void                do_measure_right_left_ang       ();
335     void                do_measure_standardized_ang     ();
336     void                do_measure_total_ang            ();
337     void                do_measure_total_runs           ();
338     void                do_measure_total_sin            ();
339     void                do_measurement_choice           ();
340     void                do_process                      ();
341     void                do_quit                         ();
342     void                do_reset                        ();
343     void                do_simplification_choice        ();
344     void                do_smoothing_choice             ();
345     void                draw_canvas                     ();
346     void                draw_point                      ();
347     Panel_setting       enter_coord_char                ();
348     void                enter_in_coordinate             ();
349     void                enter_new_point                 ();
350     void                enter_out_coordinates           ();
351     void                file_i_o                        ();
352     void                handle_canvas_event             ();
353     void                locate_item                     ();
354     void                make_color_map                  ();
355     void                ok_button                       ();
356     void                overlay_displays                ();
357     void                set_point_coordinates           ();
358     void                set_color                       ();
359     void                show_button_menu                ();
360     void                show_hide_coordinates           ();
361     void                show_hide_displays              ();
362
```

```
/*=============================================================*/
/* Measure routine by Robert B. McMaster, UCLA, April 1987.    */
/* Adapted from FORTRAN by Yvon Perreault, PAR Gov't Sys. Corp., May 1987.   */
/*=============================================================*/

#define X       0
#define Y       1

#include <math.h>

struct
measures {
   float        total_length,
                total_angularity,
                right_angularity,
                left_angularity,
                std_angularity_inch,
                num_coordinates,
                total_runs;
         };

static
float           pi = 3.1415927;

float           get_angle          ();
float           get_length         ();

/*=============================================================*/

void
measure (coord, meas)
   int          coord[][3];     /* coordinate array */
   struct
   measures     *meas;          /* measurements to be computed */

/*
 *  Compute the following measurements on the line in array coord:
 *      - Total length
 *      - Total angularity
 *      - Right & left angularity
 *      - Standardized angularity per inch
 *      - Number of coordinates
 *      - Total runs
 */

{
   float            p1[2], p2[2], p3[2], angle,
                    pos_cnt, neg_cnt,
                    pos_ang, neg_ang, tot_ang,
                    tot_len, tot_run,
                    run_len, run_len_sum, run_len_sq_sum, run_mean, run_std,
                    cin, cin_avg, cin_sum, cin_sq_sum, cin_mean, cin_std,
                    zone, z2, z3, z4, zscore,
                    dist2, temp1, temp2;
   int              i, last,
                    p_run, n_run,
                    p_cnt, n_cnt,
                    p_sin, n_sin;

   /* initialize counters */
   pos_ang = neg_ang = tot_ang = tot_len  = 0.0;
   run_len = run_len_sum = run_len_sq_sum = 0.0;
   cin                                    = 1.0;
```

```
64      cin_sum = cin_sq_sum                    = 0.0;
65      last = p_run = n_run                    = 0;
66      p_cnt = n_cnt = p_sin = n_sin           = 0;
67
68      /* process each triad - assume unused array entries are set to (-1,-1) */
69      for (i = 0; coord[i+2][X] >= 0; i++) {
70
71         /* assign points */
72         p1[X] = coord[i  ][X];
73         p1[Y] = coord[i  ][Y];
74         p2[X] = coord[i+1][X];
75         p2[Y] = coord[i+1][Y];
76         p3[X] = coord[i+2][X];
77         p3[Y] = coord[i+2][Y];
78
79         /* calculate and accumulate both lengths in the triad */
80         tot_len += get_length (p1, p2);
81         dist2    = get_length (p2, p3);
82         tot_len += dist2;
83
84         /* accumulate for coordinates per inch */
85         if (tot_len >= cin) {
86            temp1       = (float) (i+1 - last);
87            last        = i+1;
88            cin_sum    += temp1;
89            cin_sq_sum += temp1 * temp1;
90            cin++;
91         }
92
93         /* compute angle */
94         angle = get_angle (p1, p2, p3);
95
96         /* determine if angle is positive or negative and increment counters */
97         if (angle < 0.0001 && angle > -0.0001) /* approximately zero */
98            angle = 0.0;
99         else
100           if (angle > 0.0) { /* positive */
101              pos_ang += angle;
102              p_cnt++;
103              p_sin++;
104              if (n_sin > 0) { /* wrap-up previous negative run */
105                 n_run++;
106                 n_sin           = 0;
107                 run_len_sum    += run_len;
108                 run_len_sq_sum += run_len * run_len;
109                 run_len         = 0.0;
110              }
111           }
112           else { /* negative */
113              neg_ang += angle;
114              n_cnt++;
115              n_sin++;
116              if (p_sin > 0) { /* wrap-up previous positive run */
117                 p_run++;
118                 p_sin           = 0;
119                 run_len_sum    += run_len;
120                 run_len_sq_sum += run_len * run_len;
121                 run_len         = 0.0;
122              }
123           }
124        run_len += dist2;
125
126     } /* for i */
```

```
127     i += 2; /* adjust after loop */
128
```

```
 98  /*-----------------------------------------------------------------------*/
 99
100  int
101  perp_dist (p1, p2, p3)
102     int             p1[], p2[], p3[];
103
104  /*
105   *  Calculate perpendicular distance from p2 to line segment between p1 and p3
106   */
107
108  {
109     int             p4[Y+1],
110                     a, b, c, d, e, f;
111
112     a = p1[Y] - p3[Y];
113     b = p3[X] - p1[X];
114     c = p3[Y] - p1[Y];
115     d = (p1[Y] * p3[X]) - (p3[Y] * p1[X]);
116     e = (c * p2[Y]) + (b * p2[X]);
117     f = (a * c) - (b * b);
118
119     p4[X] = ( (c * d) - (b * e) ) / f;
120     p4[Y] = ( (a * e) - (b * d) ) / f;
121
122     a = p2[X] - p4[X];
123     b = p2[Y] - p4[Y];
124     return ( (int) (0.5 + sqrt ( (double) ((a * a) + (b * b)) ) ) );
125
126  } /* perp_dist */
```

```
129     /* check for last run */
130     if (p_sin > 0)
131        p_run++;
132     if (n_sin > 0)
133        n_run++;
134     run_len_sum    += run_len;
135     run_len_sq_sum += run_len * run_len;
136
137     /* computation of measurements */
138     tot_ang = pos_ang - neg_ang;
139     cin_avg = tot_ang / ((float) i - 2.0);
140     tot_run = (float) (p_run + n_run);
141     pos_cnt = (float) p_cnt;
142     neg_cnt = (float) n_cnt;
143
144     /* z-score calculation - are these needed? */
145     temp1  = pos_cnt * neg_cnt * 2.0;
146     temp2  = pos_cnt + neg_cnt;
147     zone   = temp1 / temp2 - 1.0;
148     z2     = temp1 * (temp1 - pos_cnt - neg_cnt);
149     z3     = ((temp2 * temp2) * (temp2 - 1.0));
150     z4     = (float) sqrt ( (double) (z2 / z3) );
151     zscore = ( (tot_run - zone) / z4);
152
153     /* run statistic data */
154     run_mean = run_len_sum / tot_run;
155     temp1 = ((tot_run * run_len_sq_sum) - (run_len_sum * run_len_sum)) /
156                 (tot_run * (tot_run - 1.0));
157     run_std = (float) sqrt (temp1);
158
159     /* coordinates per inch */
160     cin--;
161     cin_mean = (float) i / tot_len;
162     temp2 = ((cin * cin_sq_sum) - (cin_sum * cin_sum)) / (cin * (cin - 1.0));
163     cin_std = (float) sqrt (temp2);
164
165     /* return measurements */
166     meas->total_length        = tot_len;
167     meas->total_angularity    = tot_ang;
168     meas-> left_angularity    = pos_ang;
169     meas->right_angularity    = neg_ang;
170     meas->std_angularity_inch = cin_std;
171     meas->num_coordinates     = i;
172     meas->total_runs          = tot_run;
173
174  } /* measure */
175
```

```
176 /*-------------------------------------------------------------------------*/
177
178 float
179 get_angle (p1, p2, p3)
180    float       p1[], p2[], p3[];
181
182 /*
183  *  Compute the angle of change between two connected vectors
184  */
185
186 {
187    float       angle,
188                p4[2], p5[2];
189
190    p4[X] = p1[X] - p2[X];
191    p4[Y] = p1[Y] - p2[Y];
192    p5[X] = p3[X] - p2[X];
193    p5[Y] = p3[Y] - p2[Y];
194
195    /* check for negative infinity */
196    if (p4[X] == 0.0)
197        p4[X] = 0.000000001;
198    if (p4[Y] == 0.0)
199        p4[Y] = 0.000000001;
200    if (p5[X] == 0.0)
201        p5[X] = 0.000000001;
202    if (p5[Y] == 0.0)
203        p5[Y] = 0.000000001;
204
205    /* compute angle */
206    angle = atan2 (p4[Y], p4[X]) - atan2 (p5[Y], p5[X]);
207    angle = (angle < 0.0) ? -pi - angle : pi - angle;
208
209    return (angle);
210
211 } /* get_angle */
212
```

```
213  /*-----------------------------------------------------------------------*/
214
215  float
216  get_length (p1, p2)
217     float       p1[], p2[];
218
219  /*
220   *  Compute the segment length from p1 to p2
221   */
222
223  {
224     float       len[2];
225
226     len[X] = p2[X] - p1[X];
227     len[Y] = p2[Y] - p1[Y];
228     return ((float) sqrt ((double) ((len[X] * len[X]) + (len[Y] * len[Y]))));
229
230  }/* get_length */
```

```
#define SSIZE        200                        /* stack size */
#define X            0
#define Y            1
#define Z            2

#include <math.h>

   extern
   int               icoord[][3],
                     ocoord[][3];

   int               Douglas_Peucker   (),
                     perp_dist         ();

/*-----------------------------------------------------------------------*/

int
Douglas_Peucker (tol, cnt)
   int               tol,                       /* tolerance (pixels) */
                     cnt;                       /* input point count */

/*
 *  Douglas-Peucker line simplification algorithm
 */

{
   int               anchor[3],                 /* current anchor  point */
                     floatr[3],                 /* current floater point */
                     a_stack[SSIZE][3],         /* anchor  stack */
                     f_stack[SSIZE][3],         /* floater stack */
                     ipt[3],                    /* point being tested */
                     a,                         /* anchor stack pointer */
                     f,                         /* float  stack pointer */
                     i,                         /* general index */
                     ai,                        /* anchor index */
                     fi,                        /* floater index */
                     mi,                        /* point index with max perp distance */
                     max_dist,                  /* maximum perp distance calculated */
                     dist;                      /* perp distance calculated (pixels) */

   if (icoord[0][X] == icoord[cnt-1][X] &&      /* Check that endpoints are different */
       icoord[0][Y] == icoord[cnt-1][Y])
      return (-1);

   i = ai = a = f = 0;
   a_stack[a  ][X] = icoord[ai   ][X];
   a_stack[a++][Y] = icoord[ai   ][Y];
   f_stack[f  ][X] = icoord[cnt-1][X];
   f_stack[f++][Y] = icoord[cnt-1][Y];

   while (f) {                                  /* floater stack is not empty */
      anchor[X] = a_stack[a-1][X];
      anchor[Y] = a_stack[a-1][Y];
      floatr[X] = f_stack[f-1][X];
      floatr[Y] = f_stack[f-1][Y];
      /* adjust index for current floater */
      for (fi = ai+1; icoord[fi][X] != floatr[X] ||
                      icoord[fi][Y] != floatr[Y]; fi++);
      if (floatr[X] == icoord[ai+1][X] &&
          floatr[Y] == icoord[ai+1][Y]) {       /* anchor and floater are adjacent */
         a_stack[a  ][X] = f_stack[--f][X];     /* pop floater onto anchor stack */
         a_stack[a++][Y] = f_stack[ f][Y];
         ai++;                                  /* bump anchor index */
```

```
64          }
65          else { /* not adjacent */
66             max_dist = 0;
67             for (i = ai+1; icoord[i][X] != floatr[X] ||
68                            icoord[i][Y] != floatr[Y]; i++) {
69                ipt[X] = icoord[i][X];
70                ipt[Y] = icoord[i][Y];
71                dist = perp_dist (anchor, ipt, floatr);
72                if (dist > max_dist) {              /* point has maximum perp distance! */
73                   max_dist = dist;
74                   mi = i;
75                }
76             } /* for i */
77             if (max_dist <= tol) {
78                a_stack[a  ][X] = f_stack[--f][X];  /* pop floater onto anchor stack */
79                a_stack[a++][Y] = f_stack[  f][Y];
80                ai = fi;                            /* adjust anchor index to new floater */
81             }
82             else {
83                f_stack[f  ][X] = icoord[mi][X];    /* new floater point */
84                f_stack[f++][Y] = icoord[mi][Y];
85                fi = mi;                            /* adjust floater index */
86             }
87          } /* not adjacent */
88       } /* while */
89
90       for (i = 0; i < a; i++) {
91          ocoord[i][X] = a_stack[i][X];             /* set up output points */
92          ocoord[i][Y] = a_stack[i][Y];
93       }
94       return (a);                                  /* return the count of output points */
95
96    } /* Douglas_Peucker */
97
```

75 Years Stimulating America's Progress
1913-1988