Effect of Automated Cartographic Generalization on Linear Map Features

by

John A. Young

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE
in
Geography

APPROVED:

[illegible]

Laurence W. Carstensen, Jr., Chairman

James B Campbell — [illegible]

James B. Campbell — James L. Smith

September 1991
Blacksburg, Virginia

c. 2

LD
5655
V855
1991
Y485
C. 2

EFFECT OF AUTOMATED CARTOGRAPHIC GENERALIZATION ON LINEAR MAP FEATURES

by

John A. Young

Committee Chair: Laurence W. Carstensen
Geography

(ABSTRACT)

The process of automated cartographic generalization is critically reviewed and methods developed for implementation and analysis are discussed. The manner in which automated generalization relates to manual cartographic methods and feature representation is analyzed. It is suggested that the nature of representation of linear features on maps be considered in the analysis of effectiveness of automated generalization.

The development of a computer platform for evaluating linear generalization algorithms is described and three studies which make use of the platform are discussed. An analysis of the performance of five simplification algorithms is compared to performance of a random simplification algorithm. It was found that in most cases tested, the five simplification algorithms performed better than random. An analysis of the stability of fractal dimension estimated on simplified lines was conducted and it is suggested that the fractal dimension is a poor guide for linear simplification due the instability in measurement. An examination of the effect of generalization on linear features as represented by contoured topography and paired stream bank lines was performed. Through the use of measurements of slope on contour lines and width on stream lines, it was determined that automated generalization has an effect on linear feature representations. Guidelines for application of linear generalization algorithms are suggested and needs and direction for future research are discussed.

Acknowledgements

I wish to thank the General Dynamics Corporation, the Center for Innovative Technology, and the Va. Tech CDB project for financial support of this research. I also wish to thank my committee members for their assistance and patience. Finally, I wish to thank Tracy Mitchell for her support and encouragement and all of my friends at Virginia Tech for the many diversions which made my time in the graduate program an enjoyable, if somewhat lengthy experience.

Table of Contents

Figures

Tables

1. Introduction

Maps have long been important tools for geographical studies, education, planning, and environmental resource assessment. Traditionally maps were drafted by cartographers specifically trained to graphically represent the physical world in a manner that retains relative positions of geographic features. It is often said that an element of art was present in these manually drafted maps, and indeed an aspect of the artistic is needed to represent the world in any scaled down version of reality. It is impossible to transfer every detail of the environment to a scaled model (of which maps are but one example), and therefore some form of "generalization" is needed to select only the most important features or aspects of the particular environment being modeled.

With the advent of computerized methods in cartography, however, informal methods used to generalize map features are no longer adequate. Computerized storage and manipulation of geographic information require explicit formal rules. Every aspect of what was previously a mental process must be described and translated into computer languages. The ambiguities and subjectivity in manual generalization must be explored, objectively defined, and formalized in order for computer algorithms to attain the representational fidelity of manual methods.

Just how to best carry out this transition from manual to automated aspects of cartography is still in question. While several computer algorithms for generalization have been introduced during the past twenty-five years, much is still unknown regarding efficiency and reliability of generalization algorithms. The problems of automated generalization have thus become a primary topic for research in academic cartography and much progress has been made recently at evaluating computer algorithms for cartographic generalization.

Some of the questions which arise in this line of inquiry are; "what is the meaning of a cartographic 'feature', is it simply a collection of lines or does it have some other meaning independent of the components which represent it?", and "how does constraining a continuous line by a series of coordinates affect the ways in which it is perceived or used?". These questions become even more pressing when considered in the framework of multiple operating scales and present even more puzzling unknowns such as, "what happens to a digital line when it is simplified or generalized and displayed at a different scale than originally 'captured'?", "how does the manipulation of digital lines affect the feature which it represents on a map?", "what controls are necessary to assure that the digital line remains faithful to the piece of reality which it represents when not under direct human guidance?".

Such questions have only recently surfaced and *must* be satisfactorily answered if we are to place faith in results obtained from digital mapping and spatial analysis systems. The evaluation of simplification algorithms in terms of the representational nature of lines on maps must be considered in order for automated generalization to be truly effective. There is still much to be explored in terms of the optimum methods of automated generalization of different types of cartographic data, tradeoffs between retention of information and computational efficiency, and effects of automated generalization upon accuracy of measurement and analysis on digital cartographic representations.

In addition, we need to understand more fully the effects of generalization on linear features in the context of cartographic space. Lines on a map are paired or grouped to represent features such as the banks of a river or the contoured topography of a mountain. Effects of automated generalization upon lines representing such features are not known; what is their effect on measurement of (for example) the width of a river, or topographic slope? A more appropriate assessment of the accuracy and reliability of automated linear generalization techniques in such situations deserves further study.

This study considers these issues and presents the results of several experiments to evaluate automated generalization of linear map data. The goals of this research are to develop methods for analyzing automated generalization in areas not previously examined in the literature, to evaluate the performance of common line generalization algorithms against randomly generalized base lines, to examine the feasibility of using estimates of fractal dimension as a guide for simplification, and to examine the effects of automated generalization on linear map features.

An examination of the literature is presented which explores theoretical and operational aspects of generalization of map data. First, a review of the theoretical basis of map representations is conducted in regard to the manner in which automation of generalization might affect feature representation. Secondly, the development of algorithms for computer generalization of linear map data is explored in an attempt to build a framework for evaluation of different types of algorithms. Next, methods for evaluating the results of automated linear generalization are critically reviewed and additional areas of inquiry are suggested.

The development of a platform of computer programs for evaluating automated generalization is explained. The methodology and results of several experiments are then discussed. The first evaluates five algorithms by comparing algorithm performance against randomly generalized lines. The second experiment examines the use of fractal geometry measurements for evaluation of different algorithms and points up the limitations and appropriateness of this measure. A third experiment examines the effects of automated linear generalization upon map features as represented by paired or grouped lines. Measurements of stream width on paired stream lines and measurements of slope on contoured cartography are presented which demonstrate the degree to which automated generalization can be carried out before effects on feature representations become problematic.

Lastly, a summation of the current research effort is given which places this effort in the context of contemporary research in cartography and geographic information systems and which suggests some interesting areas for future research.

2. Literature Review

2.1 Definition of Generalization

Generalization is a fundamental activity of cartography, as no representation can be entirely true to the phenomenon it is modeling. No representation can capture all detail inherent in the 'real world', and therefore simplification and selective omission are necessary for comprehension of complex phenomena. Indeed, complete depiction of *all* detail is not desirable even if it were possible because maintenance of legibility and clarity require omission of unwanted detail. This is true whether the representation takes the form of a newspaper article, an oral description, or a map. It can be said that generalization is a fundamental *mental* activity which is necessary for clear description and communication of ideas.

2.2 Generalization in Cartography

For centuries, generalization has been conducted by individual cartographers in a subjective, intuitive manner. Cartographers used their own "artistic" sensibilities in deciding how to best represent features at smaller scales. Since this process was considered to be a part of the art of cartography it was thought unnecessary and perhaps useless to attempt to formalize the procedures of generalization. Generalization of cartographic detail in manual cartography is for the most part a process carried out in the mind of the cartographer. Few, if any, guidelines traditionally existed to guide the cartographer on the best method to represent natural features at various ranges in scale, nor were such guidelines needed. The artistic instincts on which the manual cartographer prides himself are the only guidelines which are normally adhered to, at least in the sense of a codified set of rules. Although the generalizations of natural phenomena thus produced are subjective interpretations, a skilled cartographer is capable of producing accurate (individual) representations of natural phenomena at a range of scales. Through the years, these manually rendered representations have been adequate for the purposes at hand and no doubt improved through time with the refinement of technical skill and material. Since the incorporation of computer assisted methods in cartography, however, many of these manual processes have required re-examination.

2.3 Automation in Cartography

As in many other fields, the rush to automate has consumed the field of cartography in the past several years. This flurry of activity has reinvigorated the discipline and has generated renewed interest in cartographic forms of expression and analysis. Indeed, it can easily be seen that entire new industries are being spawned by the ability to model spatial processes of the environment using digital cartography and associated automated databases. These new tools of analysis based on automated cartography have been termed Geographic Information Systems (GIS), Land Information Systems (LIS), and Automated Mapping and Facilities Management (AM/FM) and are rapidly being incorporated in natural resource agencies, local governments, international development groups and a host of other organizations. While renewed interest has certainly introduced cartographic forms of analysis into disciplines formerly ignorant of its promise, it has also brought operators, not educated or experienced in the traditions of cartography, into a position to construct illegible and inaccurate cartographic representations with a speed and ease never before imagined.

Automation in cartography has forced a re-assessment of aspects of cartography previously taken for granted. Chief among these is how natural and cultural features are generalized and represented on maps and the questions that arise when converting a manual cartographer's pen strokes into a series of digital coordinates, or cells in a computerized grid. This area of inquiry has immediate application as more and more decisions are being aided or even based upon results from digital mapping systems. A more complete understanding of prospects and limits of this technology is needed so that it can be appropriately applied to the pressing problems of the day such as environmental planning and management of global warming, toxic waste, rainforest depletion, etc. The management of such complex problems will increasingly require these systems. It is equally important for the discipline of cartography to explore the subtleties of the transition from manual to automated cartography so that appropriate guidance can be given to those who would benefit from such technology but who would be dissuaded from doing so due to lack of confidence in the mapped representations.

It therefore would be useful to take a close look at cartographic generalization, to examine the ways in which it has been automated, and to evaluate differences in application of automation. From this vantage it may then be possible to determine those implications, if any, that exist for manipulating digital cartographic representations at ranges of scale.

2.4 Elements of Cartographic Generalization

Generalization in cartography is a rather vague term which encompasses several important processes. Robinson, et.al. (1984) have termed these processes the 'elements' of cartographic generalization. They are (1) *simplification:* the identification and retention of important characteristics of data and the subsequent elimination of unwanted detail, (2) *classification*: the grouping together of data, (3) *symbolization:* the graphic representation of simplified or classed data, and (4) *induction:* the use of logical inference, going from the specific to the general (Robinson, et.al., 1984). Conceptually, this definition is applicable to both manual or automated generalization although, as will be seen later, the manner in which these processes are implemented differs substantially and automated generalization has additional objectives which must be met.

Cartographic generalization is employed for a number of purposes and can be implemented by a number of means. Chief among purposes is reduction of detail for scale reduction. Scale reduction without generalization leads to crowding on the map, unintentional coalescence of features, shortening of distances, and ultimately illegibility (Robinson, et.al, 1984). In order to counteract these negative effects of scale reduction, features must be simplified to retain only their important aspects and some set or class of features must be selected to be retained on the map while others are deleted (although selection is not considered by Robinson to be an element of *cartographic* generalization but rather a mental generalization operation (Robinson, et.al., 1984)). Other reasons for generalization in cartography are for simplification of detail *without* scale change for clarifying an aspect of the map to be communicated, and reduction of storage and processing requirements in a digital environment.

Generalization in cartography is also said to have *controls* on its implementation. These are proposed by Robinson, et.al. (1984) as including objectives of the map, the scale at which a map is to be produced, graphic limits in terms of ability to symbolize only a limited number of items, and quality of the data (the better the quality, the more detail available). All of these terms are difficult to quantify, however, and for the most part controls of generalization are determined at the time of map production in a subjective manner by individual cartographers. Although some attempt has been made to quantify the amount of information to be controlled during automated cartographic generalization (using concepts of information theory as discussed later), at present it is still primarily a subjective design consideration. However, the main reason for applying controls to generalization in both manual and automated realms is to maintain accuracy of *representation* of geographic features relative to objective, scale, and graphic symbolization as maps are intended to be relatively accurate devices which store and communicate ideas about spatial reality.

2.5 The Graphic Language of Maps

Maps, in the traditional sense can be considered to be forms of a graphic. As such, they have signs and symbols with which they communicate their messages to the map reader. The messages communicated are referents to reality, or some abstraction thereof. Thus maps may symbolize terrain in a particular place, political boundaries of a region, or a statistical distribution of some quantity. The ability of a map to represent many different ideas or quantities attests to the power of this form of graphic, but creates problems when attempting to discuss a particular aspect of map accuracy. Thus when attempting to analyze aspects of communication in maps or effects of methodological changes on map accuracy, it is necessary to define the type and purpose of the map under consideration.

The type of map under consideration in this study are commonly termed *reference* maps (Muller, 1987b). These are typically large scale, multipurpose maps whose main purpose is to store spatial data (such as the USGS topographic map series), as opposed to *thematic* maps whose main function is to present a specific theme or distribution at the expense of planimetric or positional accuracy (Muller, 1987b). This distinction is particularly important as any discussion of accuracy necessarily hinges upon the goals for which the map is designed. Most thematic maps are deliberately distorted so that a certain idea takes precedence over the rest of the graphic assuring that the idea or distribution is clearly illustrated. Generalization plays an important role in these maps, but is primarily a design consideration.

Reference maps, on the other hand, make specific claims of being an approximation of reality. These maps are typically used for positioning or for measurement of some quantity such as length, height, or area (Muller, 1987b). Therefore, accuracy issues become very important and the effects of methodological changes in map compilation take on a special relevance. These maps have associated with them certain standards of symbology and representation and present themselves as keepers of the 'truth' about the world, regardless of the degree to which they deserve this acclaim. The issues of generalization and automation are especially crucial to this class of map and therefore they will be the sole subject of this study.

In addition to the type of map under study, the element of generalization under consideration also needs to be defined. The process of simplification is one of the most arbitrary and enigmatic components of generalization, particularly in a digital environment. It is also one of the most important components of generalization on linear map data, the predominant symbol on most reference maps. Therefore the development and limitations of simplification processes under automation deserves further study.

2.6 Maps as Models of Reality

The main purpose of cartography has traditionally been to represent some portion of the surface of the earth on a plane (Eckert, 1908). Assumed in this representation is some degree of equivalence with what actually exists on that portion of the earth. This equivalence is necessary for the functionality of the map. As Miene (1974) describes, [reference] maps function in five basic manners: (1) as a means of orientation, (2) as a data and information carrier, (3) as a scientific means of expression, (4) as an analytical instrument of quantitative research, and (5) as a basis for [other] mapping and planning. It is clear from this description that maps are meant to serve as objective models of reality.

Implicit in this model is the assumption of scientific validity. That is, maps can be considered to be products of science as they *attempt* to provide an objective view of reality which can be independently verified. As we shall see, maps can achieve this goal with only limited success as their objectivity is compromised by the need for generalization.

2.7 Inherent Subjectivity in Maps

As Eckert described, in one of the earliest discussions of cartographic generalization, "In generalizing lies the difficulty in scientific map making, for it no longer allows the cartographer to rely merely on objective facts but requires him to interpret them subjectively" (1908, 3). In traditional or manual cartography, it is this subjective interpretation which requires an artistic element from the cartographer (aside from map layout or design). Thus, in changing scales, the cartographer derives a mental picture of how to best symbolize a feature, such as a meandering river, so that its general character is preserved but detail is reduced. This mental picture is then transformed by the cartographer and expressed by the graphic within the confines of cartographer's artistic license. As Eckert states in an often quoted passage, "With generalization art enters into the making of maps" (1908, 3).

Eckert makes a case for preserving the artistic in map making as he asserts that cartography, stripped of the artistic elements would, "sink to the level of a mere handicraft (1908, 3)". However, in later years cartographers would come to view the subjective artistic elements as the greatest limiting factor in the scientific validity of maps. Wright (1944), commenting on the subjective nature of maps, states, "maps are drawn by men and not turned out automatically by machines, and consequently are influenced by human shortcomings" (1944, 8). These shortcomings are the inability to remove an individual cartographer's subjective influence from the map, regardless of the degree to which the map is made to appear to be an objective 'picture' of reality. As will be seen, the removal of these subjective elements is difficult even in an automated environment. Wright's statement that, "the trim, precise, and clean-cut appearance that a well drawn map presents lends it an air of scientific authenticity that may or may not be deserved (1944, 8)," applies equally well to the clean appearance of digitally produced maps .

If traditionally drawn maps suffered to some degree from subjectivity, at least they were never entirely subjective. Most in fact consisted of a mix of partly objective reality and partly subjective interpretations (Wright, 1944). The degree to which subjective elements influenced the reliability of the final map depended in large part on the manner in which maps were compiled from 'raw' data. As stated by Wright:

> *Although a map's reliability is no higher than that of its sources, it may be considerably lower unless good judgment is exercised in its compilation. In this process the 'raw' information provided by the sources is transformed by the cartographer. Two operations may be carried out. If the 'raw' information is too intricate or abundant to be fully reproduced to the scale of the map as it stands, it may be simplified and generalized. If it is too scanty, it may be amplified and elaborated (1944, 13).*

2.8 Analysis of Map Subjectivity Using Cartographic Communication Theory

It is clear that the limiting element in map compilation comes in transformation of objects of the real world to their corresponding elements on a map. The manner in which this transformation is accomplished has been analyzed using a special case of information theory, known as 'cartographic communication theory', which has achieved a great amount of attention, especially among European cartographers. Cartographic communication theory attempts to trace the process by which ideas or conceptions of reality are transferred and symbolized in the graphic 'alphabet' of cartography. As will be seen, an attempt was made to use ideas of cartographic communication theory to create objective and thus easily automated generalization processes, but the success of this effort was limited due to the need for subjective human interpretation for proper symbolization of geographic features.

Cartographic communication theory typically consists of an analysis of the path or flow of information from its source (reality) to its destination (the map reader). An attempt is made to analyze cartographic communication in terms of energy inputs, flows, and losses (Robinson &

Petchenik, 1975). Basic ideas for this information flow were borrowed from telecommunication research and as such represented an analysis of a unidirectional flow of information (Shannon and Weaver, 1949). The basic system is outlined in figure 2.1.

The *source* in this system as applied to cartography is intended to represent reality, from which the cartographer would make a map and pass the information on to the map reader. As has happened, however, the use of the term "reality" opened the description of the system to philosophical questions on the realness of reality of which the cartographer is supposedly cataloging (Steward, 1974: Ratajski, 1978). As the aim of the cartographer was to record 'objective' reality, the degree to which this was possible was questionable (Miene, 1974). It came to be seen that a specialized communication system, such as in cartography, entails many exceptions to the idealized communication system, and information actually passes through several *filters*, such as the subjective conceptual interpretations of both the map maker and the map reader and improper methodological application (eg. improper generalization) (Robinson & Petchenik, 1975). These so called filters corresponded to *noise* in the communication system and limit a map's depiction of reality from what actually exists in the real world (Figure 2.2).

Thus, the ability of the cartographer to produce a truly objective map in the manual realm is limited by subjective influences from both faulty conceptions of reality and by subjective application of the methodologies or elements of generalization. This is the reason most generalizations in the manual realm are individualized products and are for the most part non-replicable.

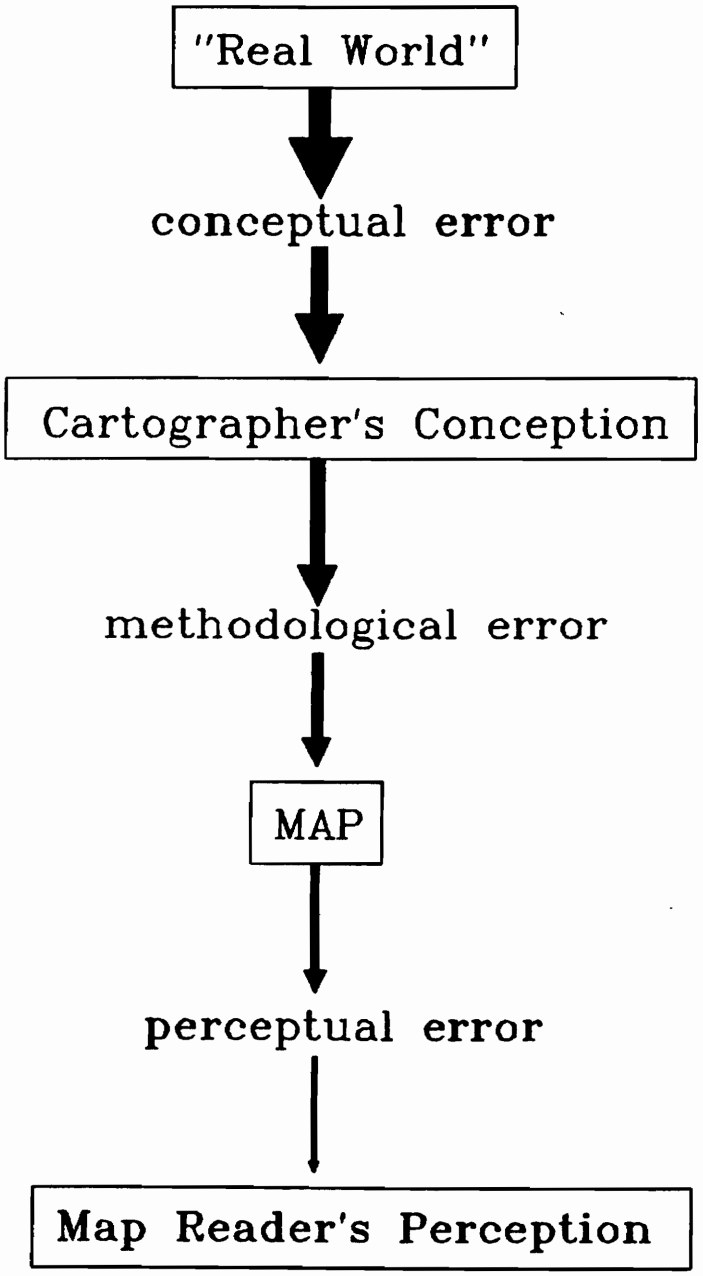


Figure 2.1. Information Flow in an Idealized Communication System (After Robinson and Petchenik, 1975).

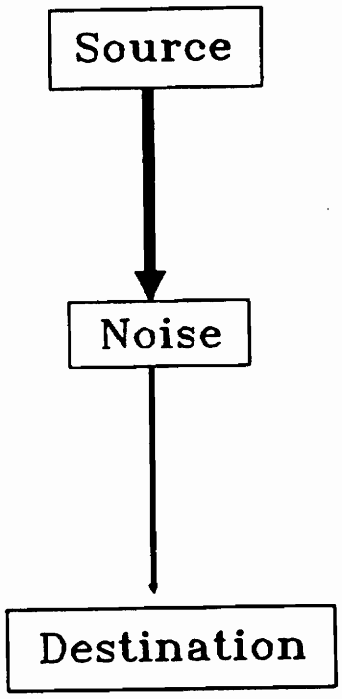


Figure 2.2. Information Flow in a Cartographic Communication System (After Robinson and Petchenik, 1975).

2.9 Error in Cartographic Communication

The component of 'noise' in the cartographic communication system can be considered to be equivalent to error, which to a greater or lesser degree is inherent in all communication systems. As Bertin observes, "there is in graphic symbology a one-to-one relation between what is meant and what is understood which leaves no room for ambiguity...the very presence of ambiguity is the witness of a mistake (1978, 119)" (** While error in map readers perception is certainly a major component in improper graphic communication, it is beyond the scope of this discussion and we will concern ourselves only with errors introduced in the steps leading to the graphic expression of reality in the form of a map **). Graphic noise can be considered to represent anything in the graphic display that obstructs the map reader (termed the *percipient*) from receiving the desired message (Robinson & Petchenik, 1975).

Examples of graphic noise as mentioned by Robinson and Petchinik are, "prominent patterns, eye-catching configurations, dense or over-powering lettering, or simultaneous contrasts of hue and value" (1975, 102). Another very important component of graphic noise, which certainly affects the interpretation of the intended message, is that of improper generalization. In this sense, the retention of too much detail or the over-elimination of detail could be considered to be a methodological error which interrupts the intended message, or at least lengthens the time of information reception (Robinson,et.al. 1984: Ratajski, 1978).

In the cartographic communication system, *any* obstruction of the signal from the source to the map reader could be considered to be interfering with the 'message' , the objective representation of reality. Thus, as Muller describes:

> *Error in a narrow mathematical sense, is the difference between a measured quantity and the true value for that quantity. In the cartographic context, any deviation between the map and the earth's surface could be considered as error (1987b, 2).*

From this standpoint it can be seen that a substantial opportunity for error exists in the representation of the real world. However, such a strict interpretation is rarely adopted. While error from improper conception may or may not ever be controlled or eliminated, error introduced as a component of generalization is more likely to yield to control, especially in a digital environment. Reduction of this source of error in mapping could have a large impact on the ability to accurately represent the environment.

Of course, error is not always considered to be an undesirable component in maps. As was mentioned previously in the discussion of thematic maps, error is sometimes used to the advantage of the cartographer in emphasizing certain aspects of a distribution. This type of error is termed 'controlled' error and is often used very effectively to communicate the general aspects of a feature even though the exact ground position of the feature is distorted (Muller, 1987b). For example, an intensely winding road on a small scale map might be symbolized by a wiggly line, even though the curves in the line do not correspond with their exact ground position (similar example given in Mark, 1989). The character of the winding road is preserved. As Muller states, "in order to tell the truth, one had to lie a little" (1987, 2).

However, in the context of automated generalization, it is not exactly clear how to separate controlled from uncontrolled error. Not enough is known as yet about effects of linear generalization to assess relative amounts of error in representations produced by automated methods. Indeed, many aspects of automated generalization remain ambiguous, including the amount or degree to which to apply generalization.

2.10 Application of Information Theory to Cartographic Generalization

The ideas of information theory have been applied in cartography in an attempt to quantify the amount or degree of generalization a given map would require for a given scale change as an objective means of conducting simplification and selection in a digital environment. In particular, attempts were made to quantify the information capacity of a map so that mathematical formulas could be derived to be used as rules for generalization (Topfer & Pilliwizer, 1966 : Srnka, 1970 : Sukhov, 1970 : Ratajski, 1978). Generalization, as approached by this method, sought to minimize 'data transmission', while at the same time passing a maximum of information to the map reader (Brassel, 1985) resulting in what Sukhov (1970) termed '*effective* information losses'.

Thus, the information capacity of a map was measured as the number of elements per square centimeter, or by some similar method whereby the map was broken down into graphic components (Sukhov, 1970). By separating the map into component pieces and measuring their density, comparisons could be made to other forms of information storage. Thus statements appeared such as that by Sukhov that, "the small-scale geographical map by its information loading appeared to exceed the book 22 times (1970, 41)". Moreover, Sukov (1970) proclaimed that information 'loading' greatly increased in the process of moving from one scale to another, presumably smaller scale.

From the breakdown of the map into 'elements', selection processes could be applied mathematically to reduce the number of elements in the map. Topfer & Pillewizer (1966) proposed a rule for selection, termed the *radical law*, which specified how much data should be eliminated when compiling from a given large scale map to a given smaller scale map. The basic equation is as follows:

$$\mathbf{Nc} = \mathbf{Ns}\sqrt{\mathbf{Sc}/\mathbf{Ss}}$$

Nc = the number of items on a compiled map with a scale fraction of **Sc** and,
Ns = the number of items on a source map with a scale fraction of **Ss**

Therefore, using a proportional or constant level of selection as described by Srnka (1970), elements could be evenly selected across a map sheet. However, from the start problems arose with this line of reasoning, the main problem being that a wholly objective selection process may tell the cartographer *how much* information to retain, but it doesn't tell him *which* information to retain. As Srnka, himself an advocate of information theory, notes:

> *A thorough analysis shows that proportional selection maintained on all maps of an entire series or set is neither theoretically nor practically justifiable. Although the quantitative characteristics of the territory are preserved, the qualitative ones are necessarily blurred (1970, 50).*

Although Srnka advocates development of a *variable* degree of selection, the critique of proportional selection points to the weaknesses inherent in applying a totally objective form of cartographic generalization. That is, symbols on a map cannot be considered only as individual elements and processed in a sequential fashion. As Robinson and Petchenik explain:

> *Because the communication of the 'information' on a map is in no way like that of a coded sequential message consisting of signals, the measurement of the 'information content' of a map and the amount transmitted by a cartographer to a map percipient cannot be effectively obtained with the techniques of Information Theory. (1975, 107).*

There is broad agreement that maps are actually read as information 'complexes' (Meine, 1974) and perceived in a "unified form of a gestalt" (Robinson & Petchenik, 1975) rather than as discrete elements perceived in isolation. Moreover, maps are primarily graphic images which describe *relations* between elements or groups of elements (Bertin, 1970: Unwin, 1981). Information is provided simultaneously to the map reader and therefore individual elements are identified in relation to other elements in a synthetic manner, identifying the features which they represent (Keates, 1972 : Wright, 1944). Nyerges (1991) has termed the distinction between individual map elements and the conceptual features which they represent "surface-structure" and "deep-structure" information, respectively. As Meine states, "the conveyance of individual pieces of information, therefore, is less important than the conveyance of information complexes, which at the same time, show causal relationships (1974, 73)".

This aspect of cartographic communication points up the differences between it and other types of communication systems. The failure of information theory to adequately objectify the generalization process is a result of the application of a linear analysis technique (first developed for analysis of natural language transmission) to what Ratajski terms "a *spatial* structure of reality (1978, 23)". Features on a map represent a spatial relationship that exists in 'reality', therefore to generalize features adequately, one has to consider the important components of the feature as a whole, rather than simply as an isolated series of points or lines. Therefore, selection of aspects of elements to simplify remains primarily an inherently subjective process (at the current state of knowledge) as an importance must be assigned by the cartographer to features, or aspects of features, to be retained.

2.11 Objectivity in Automated Generalization

Although strict application of information theory to cartographic communication and in particular to generalization is problematic, it is not without merit. From an examination of the cartographic communication process it is possible to determine areas where subjectivity enters the mapping process and to begin to examine tradeoffs between objective and subjective influences in manual generalization and the degree to which automation can and should make the process more objective.

The degree to which automated generalization should aim at making generalization more objective is apparently subject to some debate. As Brassel (1985) has observed, on one side are the traditional cartographers who believe that automated solutions to generalization are futile in the sense that, "generalization is an art which requires intuition and special skills that can only be acquired through long-term training and job experience", and that, "generally defined algorithms can never adequately consider the specifics of a local arrangement of features" (1985, 11). On the other side, according to Brassel, are typically researchers who view the solution of automation as an unsolved but solvable problem, but who "are unaware of the nature of cartographic communication", and are presently employing mechanical manipulations which do not adequately replicate manual generalization (1985, 11).

Some researchers, such as Srnka, have argued that the main aim of generalization should be to produce "theoretically correct and objective standards", minimizing subjective factors, and to "minimize the expenditure of creative intellectual work" (1970, 48). As can be expected, such statements are anathema to traditional cartographers, and at the least overstate the case as few would desire the replacement of creative intellectual work with repetitive, mechanical boredom. Some have seized on such ideas, however, and paint the automation of generalization as equivalent to aesthetic death (Steward, 1974).

In particular, some modern (actually *post*-modern) writers in historical cartography see the trend towards automation as the culmination of years of rhetoric aimed at establishing the premise that, "mappers engage in an unquestionably 'scientific' or 'objective' form of knowledge creation" while at the same time denying the influences of subjectivity introduced by the "norms and values of the order of social...tradition" (Harley, 1989, 2). To such writers, the loss of the individualized, subjective, artistic elements in cartography only intensifies the suspicion to which maps should be regarded as operating, "behind a mask of seemingly neutral science" (1989, 7). In addition, Harley is critical of the, 'belief in progress', that proposes, "by the application of science ever more precise representations of reality can be produced" (1989, 4). He admonishes the current 'culture of technics' as attempting to "incarcerate in the coordinates of the computer" the particularly human failings and subjectivity inherent in our outlook and representation of reality (Harley, 1989, 3).

While Harley's analysis is penetrating and his points are valid, the study of automated generalization has much to offer the field of cartography and does not necessarily imply a complete loss of subjectivity and creativity in mapping where those elements are appropriate. In addition, the search for more objective procedures through automated generalization should not be considered as merely an attempt to brush over the subjective influences in the mapping process with a veneer of digital accuracy. Rather, the aim of the study of automated generalization should be to identify the degree to which digital cartographic representations could be improved by more accurate portrayal at multiple scales, and from the ability to repeat generalizations in a predictable manner. Improvements in the objectivity with which map features are represented should therefore be of primary concern in future developments of automated generalization. Subjective influences in the mapping process which detract from the accuracy of geographic representation should be identified, removed, and replaced with objective rules and procedures, while those which enhance the visual or artistic aspects of maps (such as in design and layout) should be encouraged.

2.12 Defining Aims for Automating Generalization

Although there are limits to which maps can or should be made to be entirely objective (considering the many subjective influences in the mapping process), the goal of increasing objectivity and therefore accuracy in digital cartography is a worthy one. This is especially true of digital cartographic data which are used in a modeling environment in which assumptions are being made about the effects of actions taken in the real world. The ever increasing trend towards the use of map based information systems (eg. Geographic Information Systems) for natural resource analysis and community planning will require confidence in conclusions drawn from such systems as the results of faulty decisions based on these systems could have far reaching and potentially devastating impacts. The generalization of cartographic data in this context should be done in a manner in which accuracy of representation coincides to reality to the maximum extent possible. As Robinson, et. al. state, "In the simplification operation the elimination of information regarding a feature or an area must be done in a way that maintains, as far as possible, its intrinsic geographical nature (1984, 129)".

The need to pay closer attention to the nature of cartographic representations in automated generalization has been recognized as a significant problem by several authors (Meine, 1974: Kretschmer, 1978: Muehrcke, 1990). This will require identification of the phenomena represented and removal of sources of subjectivity in digital representations. That automated mapping at its present level is full of subjective influences is certainly not a new idea. Jenks (1981), in a now classic paper, discusses human influences and limitations of mapping accuracy despite automation of cartographic processes. Jenks details errors introduced through such processes as digitizing and implementation of automated simplification algorithms and makes the assertion that, "man-induced errors may exceed, indeed overshadow, errors attributable to equipment (1981, 1)". McMaster & Shea (1988) assert that there is more subjectivity inherent in selecting algorithmic parameters for two automated generalizations than in conducting two manual generalizations of the same feature.

Therefore, in terms of the automation of generalization processes, two general themes emerge. These are the need to define and test objective methods for simplification and the need to consider the accuracy of geographic representation in automated generalization. The goal for research in this area should be to determine, and if necessary improve, the accuracy with which digital mapping methods are able to inform us about the world in which we live. As Keates (1972) explains:

> *The present passion for converting map information into digital form, often at great expense, only makes sense if we are quite sure that this information is really what it may appear to be, and there is some concept of its real informational value. The growing ability to handle large amounts of information does not necessarily mean that we will be any better informed in the future, only that we will be more informed (1972, 180).*

The assessment of the informational value of map products produced by automated methods and subsequently generalized using computer algorithms needs to be better understood. Several authors have commented that uncertainty of the effects of generalization limits the confidence placed in digital map analysis (Sinton, 1978: Goodchild, 1980a). As Meine asserts, "Without clarified processes and procedures of generalization preserving or distorting, for instance, areas or structures ... no scientific background is given for the quality of the overall design of maps and charts (1978, 112)". Therefore, there is an obvious need to investigate the sources of generalization error and to quantify the magnitudes and effects of these errors in digital mapping systems.

The aim of this investigation does not necessarily have as its ultimate end the complete elimination of human influence in the mapping process, even if this were ultimately possible. Indeed there is growing agreement that the development of automated generalization methods should not be aimed at replacing the human element in cartography, but rather should be considered as enhancing confidence in cartographer's tools for design and representation (Weibel, 1990). However, knowledge of the processes and effects of automated generalization needs to be developed and explicitly defined for the proper application of this new technology.

2.13 Conceptual Frameworks for Automated Generalization

It is evident from the discussion above that digital generalization needs to be placed in a conceptual framework which can define the proper amount and type of generalization which should be applied to a given feature representation. Several authors have proposed frameworks for automation of generalization which attempt to define the proper context in which generalization should be applied. The need to explicitly define goals for automated generalization is due to the fact that the ill-defined nature of manual generalization precludes a simple transference of processes to automated methods. In addition, automated generalization has additional objectives which must be met over and above those of manual generalization.

That digital generalization incorporates processes and goals different and more extensive than manual generalization and the need to explicitly define these processes and goals necessitates a definition of automated generalization which can encompass all of its varied objectives. McMaster and Shea (1988) propose the following definition of automated generalization:

> *Generalization, in a digital environment, requires the application of both spatial and attribute transformations in order to maintain clarity, with appropriate content, at a given scale, for a chosen map purpose and intended audience (1988, 242).*

While this definition is broad enough to encompass many of the conceptual objectives of map generalization it places special emphasis on the *transformations* or processes by which automated generalization is to operate. The application of these transformations is placed into a framework of *intrinsic* objectives including *philosophical* objectives, *application* objectives, and *computational* objectives which govern why generalization is conducted (McMaster & Shea, 1988). The philosophical objectives of automated generalization should assure that automated generalizations adhere to cartographic principles (McMaster and Shea, 1988). McMaster and

Shea have listed the six philosophical objectives of automated generalization as: (1) to reduce complexity, (2) to maintain spatial accuracy, (3) to maintain attribute accuracy, (4) to maintain aesthetic quality, (5) to maintain a logical hierarchy, and (6) to consistently apply generalization rules (McMaster & Shea, 1988, 242).

The *application* objectives which McMaster and Shea (1988) discuss relate to specific requirements the map under consideration must meet to achieve the goals underlying its production (ie. specific concepts being communicated). The objectives to consider for automated generalization in this process are: (1) generalization to an appropriate scale, (2) consideration of map purpose and intended audience, and (3) maintenance of clarity (McMaster and Shea (1988)).

In addition, the transformations used in automated generalization have other objectives which differentiate it from being merely a transcription of manual processes. McMaster and Shea (1988) identify these aims as *computational* objectives. These computational objectives include (1) maximizing cost effectiveness of algorithms, (2) maximizing data reduction, and (3) minimizing computer memory and disk storage requirements (McMaster & Shea, 1988, 247). All of these objectives have the express aim of minimizing computer overhead while maintaining the maximum information content in the digital representation.

Other authors have attempted to define frameworks which place the operations or transformations involved in automated generalization into a context from which they should be applied. Brassel and Weibel (1988) define generalization as, "a special variant of spatial modeling" which is "heavily influenced by the fact that the major goal is not mere analysis of space but communication of spatial concepts" (1988, 230). They separate spatial modeling (through automated generalization) into two types of processes, *statistical* and *cartographic* generalization. Statistical generalization is defined as a filtering process whose main purpose is data reduction under statistical control to which, "concepts of reliability, tolerance, and error can be applied " (Brassel & Weibel, 1988, 232). The main goal of statistical generalization is the maintenance of minimum average displacement of the digital generalization (ie. a line) from its non-generalized counterpart, and is related to problems of error generation and propagation (Brassel & Weibel, 1988).

In contrast to statistical generalization is the concept of *cartographic* generalization. Cartographic generalization, according to Brassel & Weibel, is non-statistical and operates on "bias and elimination, emphasis and shape distortion (1988, 232)". Whereas statistical generalization has the aim of minimum average displacement, cartographic generalization has as its aim "representativeness in a holistic rather than an analytical sense" (Brassel & Weibel, 1988, 232). They make the assertion that cartographic generalization is used only for display and that map data generalized in this manner would not be a predictable source for analysis because manipulation processes introduce unpredictable errors such as feature displacement (Brassel & Weibel, 1988).

Central to the framework proposed by Brassel and Weibel is the concept of "processing based on understanding" (1988, 232). That is, for automated generalization to be truly effective, the separate processes of statistical and cartographic generalization must be structured in such a manner that the essential processes for generalization are formalized *(process recognition),* these processes are modeled *(process modeling),* and the features being modified are recognized (*structure recognition*) in order to guide the appropriate application of generalization operations (Brassel & Weibel, 1988). They stress the idea that the effectiveness of generalization algorithms can only be judged relative to the context in which they are used; in other words, it makes a difference what type of feature a line represents as to whether a generalization operation was successful (Brassel & Weibel, 1988). Brassel & Weibel advocate development of knowledge bases through "empirical tests of map perception and generalization effects" which can be used to guide the development of processes for structure recognition, process recognition, and process modeling (1988, 240).

Since the publication of the influential study by Brassel & Weibel, other researchers have expressed the importance of defining processes which deal effectively with the representational nature of map data. Mark (1989) advocates the recognition and incorporation of geometric structure of geographic features in automated generalization (and recasts Brassel & Weibel's idea of cartographic generalization as *graphical* generalization) in order that spatial relationships of geographic features be preserved. Mark (1989) emphasizes the importance of considering the *type* of feature being generalized so that the geomorphic or structural character of the feature being modeled is preserved. He terms this process "phenomenon-based generalization" and clearly makes the case that individual cartographic objects should not be generalized in isolation of the geographic feature which they represent (Mark, 1989, 69). Buttenfield (1987) also clearly makes the case that for effective guidance of simplification algorithms (ie. selection of input parameters or 'tolerance' values), the type of feature needs to be identified, especially in terms of its linear complexity.

Muller (1989) advocates the incorporation of geographic knowledge into automated generalization and is critical of the strictly algorithmic approach to application of automated generalization operations which make claims at objective generalization. Muller postulates that, "new rules that encompass the substantive content of maps must be stipulated [and] the complex functional relationships between cartographic objects, user requirements, and procedural tools for generalization need to be investigated (1989, 201)". Muller suggests that a holistic view of cartographic object recognition needs to be maintained in order to resolve problems such as "the resolution of spatial conflicts" (1989, 202). A rule-based approach, according to Muller, is needed which could go a step beyond the purely algorithmic approach and include not only the process used to conduct generalization, but also the substantive meaning of the geographic information being generalized.

In summary, the theoretical approaches to automating generalization seem to have come full circle as the attempt to make the operation entirely objective seems to have failed at present, and a new appreciation for the subjective influences in both map construction and map interpretation has emerged. The desire for objective representations of geographic phenomena must be balanced with the subjective necessities of feature recognition in map perception. Symbols on a map which are to be simplified must be considered in context of the features they represent. Algorithms which are used for generalization operations must therefore be evaluated in terms of the degree to which they maintain, or can be guided to maintain, relationships between cartographic features and geographic information.

2.14 Operationalizing Automated Generalization – Defining Algorithms for Simplification

In consideration of the concepts previously discussed, it is now possible to review the ways in which linear simplification has been automated. The goal of this discussion will be to evaluate critically the operation of several algorithms which have been implemented in the past to gain an appreciation for the manner in which digital map representations are manipulated in linear simplification processes. While in-depth reviews of generalization algorithms have previously been presented (ZYCOR, 1984: McMaster, 1987b: Clayton, 1985), a brief discussion of the major functional classes of algorithms is warranted so that we may assess the major differences in implementation of generalization algorithms. However, in order to understand fully the manner in which these algorithms operate, it is first necessary to briefly consider the nature of the digital line.

2.15 Nature of the Digital Line

Lines are represented in automated cartography in one of two ways. In the 'raster' approach, a line is represented as a set of contiguous grid cells each coded with the same value. In the 'vector' approach (of which this discussion is solely concerned), lines are represented as a series of points, each located in x,y coordinate space and connected by a (conceptually) dimensionless vector. Lines are typically 'captured' from analog (paper) maps by the use of a digitizing board or tablet upon which lines are traced and coordinates are recorded at various intervals (Boyle, 1970). These intervals can be arbitrarily decided upon by the digitizing operator, in which case the digitizer is said to be operating in 'point' mode, or a distance or time increment can be set so that coordinates are recorded at a given movement of the digitizing cursor, known as 'stream' mode digitizing.

In this manner the continuous cartographic line on the paper map is recorded or 'digitized'. As should be obvious from this description, a form of generalization necessarily takes place in the capture of analog map data as it is impossible to represent every detail of a continuous line with digital coordinates. However, points captured at a sufficiently fine density (eg. points per inch of line), especially through stream mode digitizing, can accurately represent most analog lines. It has been reported that lines captured in this manner may have an excess of coordinates over those necessary to visually recognize the line, and the desire to remove the superfluous points has been one of the primary motivations for developing linear simplification algorithms (Douglas and Peucker, 1973 : Opheim, 1982).

2.16 Linear Simplification Algorithms

Several algorithms for the simplification of linear data have been introduced in the past twenty-five years. All of these algorithms were developed for the purpose of 'weeding' coordinates from a digital line in order to (1) simplify the display of digital lines, (2) reduce drawing, plotting, and processing time of digital lines, and (3) reduce storage requirements in a digital environment (McMaster, 1987b). The goal of feature simplification is, as McMaster has stated, "the selection of the major geomorphological characteristics along a line...these are often called critical or salient points" (McMaster, 1983: also White, 1983). The degree to which simplification algorithms meet this goal varies with different algorithms due, in part, to the manner in which they operate.

Several authors have reviewed simplification algorithms and classified them according to the manner in which they process the digital line. ZYCOR, et al. (1984) place both raster and vector simplification algorithms into conceptual groupings which are based upon the type of operation used to perform simplification (Table 2.1). McMaster (1987b) divides (vector) simplification algorithms into 5 classes based upon the geometric extent, or "neighborliness" of computation (McMaster, 1987b) (Table 2.2). Clayton (1985), also classifies algorithms according to the extent of 'look' along the line for processing but uses only two groups, 'sequential point elimination' algorithms and 'global point elimination' algorithms (Table 2.3). The division of algorithms into functional classes is helpful in evaluation and comparison of the results between groups of algorithms and this discussion will consider algorithms from several of the functional classes. Due to the solely vector based approach of subsequent analysis, McMaster's (1987b) classification system of vector simplification algorithms will be followed for a functional comparison of algorithm groups.

The first functional group ZYCOR terms 'selection' and McMaster terms 'independent point' routines (ZYCOR, 1984: McMaster, 1987b). These algorithms operate by selection of points based on somewhat arbitrary parameters, such as sequential position along the line. The most commonly cited example of this class of algorithms is the "Nth point selection" routine.

Table 2.1. Zycor (1984) Simplification Algorithm Grouping.

Group	Example
Selection	Nth Point Selection Euclidian Distance Simplification
Angle Selection	Angular Tolerance Simplification
Simple Local Tolerance Band	Perpendicular Distance Simplification
Global Point Elimination	Douglas Algorithm

Table 2.2. McMaster's (1987) Simplification Algorithm Grouping.

Group	Example
Independent Point	Nth Point Selection Random Point Selection
Localized Processing	Euclidian Distance Simplification Angular Tolerance Simplification
Constrained Extended Local Processing	Opheim Algorithm
Unconstrained Extended Local Processing	Reumann-Witkam Algorithm
Global Routines	Douglas Algorithm

Table 2.3. Clayton's (1985) Simplification Algorithm Grouping.

Group	Example
Sequential Point Elimination	Nth Point Selection Euclidian Distance Simplification Angular Tolerance Simplification Perpendicular Distance Simplification
Global Point Elimination	Douglas Algorithm

The Nth point selection routine was introduced by Tobler in 1966 (Clayton, 1985). The Nth point algorithm operates by selection of points along a line according to a value entered by the operator which determines the sequential position of points to retain (Figure 2.3, see also source code listing in Appendix A). Thus, a value of 2 entered would instruct the algorithm to select every second point, a value of 5 would select every fifth point, and so on. The higher the value for N, the fewer points are selected, therefore the simpler the line becomes.

The Nth point algorithm has the advantages of being conceptually simple, easy to program, and very computationally efficient. However, it has many weaknesses, some of those identified in the literature being that the algorithm over-represents straight line segments, does not check for retention of critical points, and is overly influenced by the starting point (Clayton, 1985: ZYCOR, 1984: McMaster, 1987b).

Another algorithm which McMaster in a later paper (1989) classifies in the "Independent Point" algorithms is a random-selection of points. While this algorithm is not explicitly discussed elsewhere in the literature (although a conceptually similar idea is given in ZYCOR, 1984), the idea of a random selection of points may be useful in evaluating other algorithms. Using the random selection algorithm as a baseline, it may be possible to determine which algorithms perform at least better than random in selection of important points along a line.

The next functional class of algorithms according to McMaster (1987b) is "Localized Processing" routines. An example of an algorithm in this category was reported in 1963 by Hershey and in 1965 by Tobler (Clayton, 1985). This algorithm selects points to retain according to the point's distance to its closest neighbor. If the distance is greater than an operator specified tolerance distance, the point is retained (Figure 2.4). As originally specified by Tobler in 1965, the tolerance distance would be set equal to the width of the plotted line,

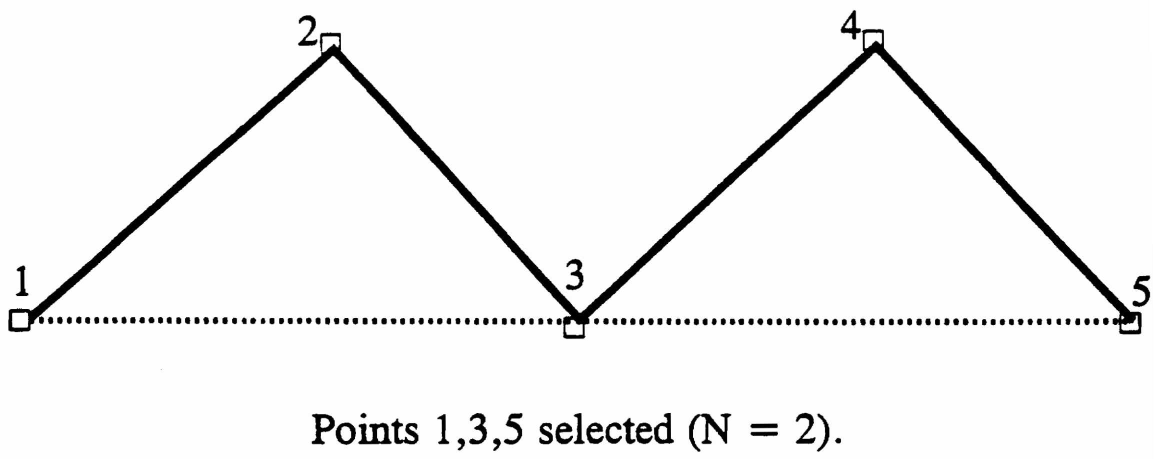
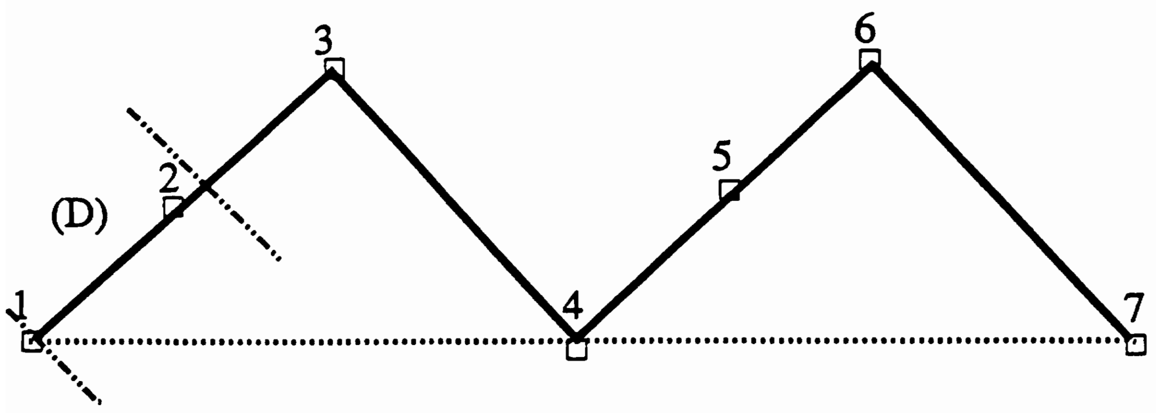


Figure 2.3 Operation of the Nth Point Selection Algorithm



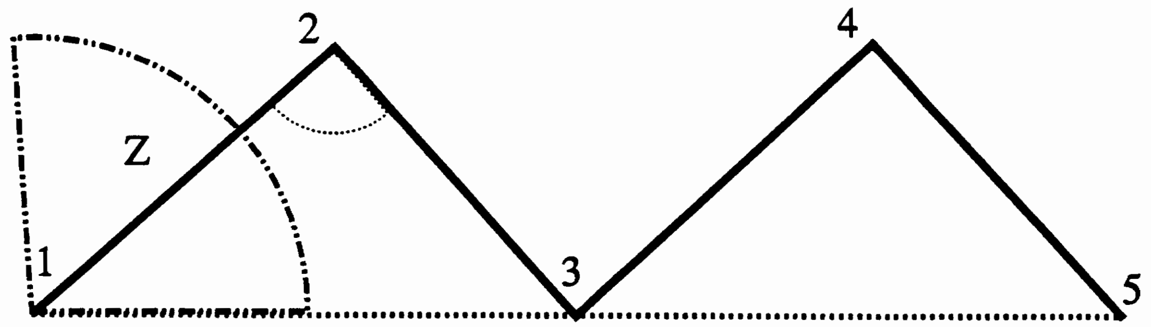
Points 1,4,7 selected with distance tolerance = D

Figure 2.4 Operation of the Euclidian Distance Simplification Algorithm.

although the tolerance value could conceptually be of any distance (McMaster, 1987b: Clayton, 1985). This algorithm will here be referred to as the "Euclidian Distance" algorithm as no other name has been assigned it in the literature. ZYCOR (1984) and Clayton (1985) classify this algorithm in the 'selection' category. The disadvantages associated with the Nth point routine also apply to the Euclidian distance algorithm.

The third class of algorithms discussed by McMaster (1987b) are "Extended Local Processing" routines. These algorithms 'look' beyond the immediate coordinate neighbors to consider sections of a line, sometimes referred to as 'tolerancing' (Clayton, 1985). An example of an algorithm in this functional class would be an angle tolerancing routine which compares angles between segments of a line with a user specified tolerance angle (termed "angle selection" by ZYCOR, 1984: also noted in Clayton, 1985: McMaster, 1987b). This algorithm operates by comparing the angle between two vectors connecting a triad of points; the angle formed between a vector connecting point one with point two and a vector connecting point two with point three is compared with the user specified tolerance angle. If the angle formed by the two vectors is greater than the user specified tolerance angle, point two is retained (Figure 2.5). The 'window' of three coordinates is then advanced to the next two segments in the line until the entire line is processed. A modification of this algorithm was later proposed Jenks (1985) by adding two distance checks between the triad of points comprising the comparison vectors; the angle between the segments is checked only if the distance between points one and two and between points two and three are less than a tolerance distance, otherwise the point in consideration is not retained.

Another "Extended Local Processing" routine which operates by tolerancing computes perpendicular distances between a triad of points and compares this distance with a user specified tolerance value (ZYCOR, 1984: Clayton, 1985, White, 1985: McMaster, 1987b).



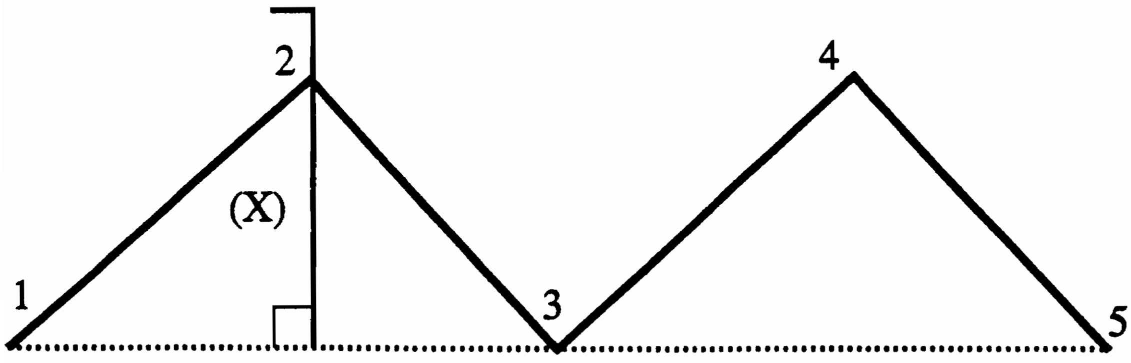
Points 1,5 selected with angle = Z.

Figure 2.5 Operation of the Angular Measure Simplification Algorithm.

This algorithm operates in a similar manner to the angle tolerancing routine discussed above by considering a 'window' of three consecutive points along a line. A reference vector is constructed between the first and third points in the triad of points, and the perpendicular distance between this vector and the second point is then computed (Figure 2.6). If the distance between the second point and the reference vector is greater than a user specified tolerance distance, the second point is retained. The window is then moved to the next triad of points until the entire line is sequentially processed. This algorithm is referred to as an example of "Simple Local Tolerance Band" algorithms by ZYCOR (1985), and a similar algorithm was reported by Lang (1969). Other algorithms included in this class are those proposed by Johannsen (1974), Deveau (1985), and Roberge (1985).

The next functional class of algorithms referenced by McMaster (1987b) are what he refers to as "Unconstrained Extended Local Processing" routines. Although this class of algorithm is not considered in this study, an example given by McMaster (1987b) is the Reumann-Witkam algorithm. This algorithm operates by considering a "search region" around the line by constructing two parallel lines about the line in question, tangent to the initial direction of the line; when the parallel lines intersect the target line, the point of intersection is retained as a 'critical' point and a new search region is formed (Clayton, 1985: McMaster, 1987b). This type of algorithm is referred to by Clayton (1985) as a "Corridor/Search Area" algorithm and is an example of algorithms which consider entire sections of a line rather than one or two segments. A similar algorithm was proposed by Opheim (1982) but uses a circular search region.

The final functional class of algorithms is termed by McMaster (1987b) as "Global" routines, by Clayton (1985) as "Global Point Elimination", and by ZYCOR (1984) as "Global Tolerance Band" algorithms. These algorithms consider the entire line in processing by



Points 1,5 selected with Perpendicular Distance (X).

Figure 2.6 Operation of the Perpendicular Distance Simplification Algorithm.

operating in a recursive or repeated fashion. The algorithm most often associated with this class of routines was proposed by Douglas and Peucker (now spelled Poiker) in 1973 (Douglas and Peucker, 1973), and is referred to as the "Douglas Algorithm" in the literature. This algorithm operates by constructing segment comparison vectors between points on the line and finds those points with the maximum perpendicular distance from successive comparison vectors. This is done initially between the first and last point on the line. The first point of the comparison vector is termed the "anchor point", and the second point is termed the "floating point" (Douglas and Peucker, 1973: ZYCOR 1984: Clayton, 1985: White, 1985: McMaster, 1987b). The point whose perpendicular distance is farthest from the comparison vector is compared to a user specified tolerance distance; if the distance between the point and the comparison vector is greater than the tolerance distance, the point is retained and becomes the new floating point (Douglas and Peucker, 1973). This process is repeated in a recursive manner until all portions of the line have been examined, and the stored floating points (stored in an array in the algorithm) become the new anchor points and thus the retained points for the generalized line (White, 1985) (Figure 2.7).

Other methods of generalizing lines have been implemented such as line smoothing through moving weighted averages of coordinates, Fourier transformations, and mathematical spline and curve fitting (ZYCOR, 1984). However, these methods in essence create entirely new representations of the feature being generalized as few or none of the original coordinates comprising the digital line are maintained, and thus fall outside the realm of traditional line simplification routines. While some of these methods have been combined with point elimination routines and are claimed to produce encouraging results (McMaster, 1989), their evaluation is beyond the current discussion (see ZYCOR, 1984 for a thorough discussion of these techniques).

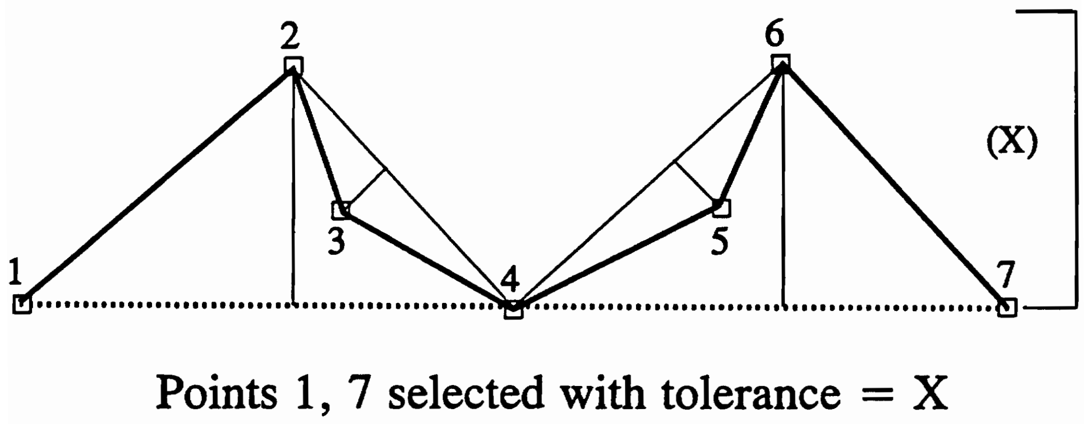


Figure 2.7 Operation of the Douglas Simplification Algorithm

2.17 Evaluation of Line Simplification Algorithms

Several authors have evaluated the relative performance of the algorithms discussed above. The evaluation of algorithms has been conducted along two main avenues, these are studies with a perceptual emphasis and parametric studies. These two focuses reflect the dual aims of digital generalization as producing representations which both 'look' right in a subjective sense to the map reader, and which maintain a measurable fidelity to the original (unsimplified) line.

2.18 Perceptual Evaluation of Linear Simplification Algorithms

Perceptual studies have evaluated linear generalization in terms of the degree to which simplification maintains recognizably important points of inflection along the line. These so called "characteristic" or "critical" points are points along the line consistently recognized by human operators as being of importance in determining the overall shape of the line and are typically points of maximum angular change (White, 1985). Atteneave (1954), in studying the psychological phenomenon of pattern recognition, determined that the most recognizable aspects of features occur at sharp angles or points of curvature. Dent (1972), and Freeman (1978) applied this concept to cartographic line drawings and found that perception of shape of cartographic features depended largely on recognition of points of maximum inflection.

Marino (1978, 1979) studied how two groups of subjects, one group being cartographers and the other non-cartographers, would determine which points along a line were critical points at different levels of generalization and found that both groups generally chose the same points for a set of individual lines even after the lines were generalized. These findings, along with earlier work, have established that critical points are points of maximum information retention and that these points should be preserved in generalization (Jenks, 1981: Marino, 1978, 1979:

White 1985: ZYCOR, 1984). However, reliance on the retention of critical points alone is not a suitable criteria for evaluation of linear generalization, as according to ZYCOR;

> *"It is unlikely that focusing on critical points for a single line fully defines the correct way to study this problem even though the vast majority of line generalization algorithms focus on one line at a time. It is the features represented by contours, for example, which are important rather than the contours themselves (Imhof, 1982, cited in ZYCOR, 1984, p. 97)".*

White (1985), built on Marino's work in evaluating critical points chosen by human operators, and evaluated the performance of linear simplification algorithms in terms of how well they maintained the same critical points as those chosen by the human subjects. White examined three algorithms (1983) and reported results for the three algorithms in a later paper (1985). These algorithms were the Nth point selection algorithm, the perpendicular distance algorithm, and the Douglas algorithm. White (1985) used three lines of varying complexity for her study, and as in Marino's study, had subjects place a given number of pins at characteristic points; the selection of these points was then compared against points selected by the three algorithms at a given simplification.

The results of this experiment revealed that the Douglas algorithm performed consistently better at choosing the same critical points as were chosen by human subjects at 4 different levels of simplification; the perpendicular distance algorithm ranked second, while the Nth point selection routine performed consistently worse at critical point detection due to its arbitrary point selection method (White, 1985). These results suggest the Douglas algorithm performs best of the tested algorithms at mimicking manual generalization. White (1985) also conducted a perceptual experiment in which respondents were asked to choose from three simplifications the one which best represented the original line. According to White, "the generalizations produced by the Douglas algorithm were overwhelmingly - by 86 percent of all sample subjects - deemed the best perceptual representations of the original lines (1985, 25)". Again, the perpendicular distance algorithm performed second best, and the Nth point routine worst in all of the experiments conducted (White, 1985).

Jenks evaluated the Douglas algorithm, the perpendicular distance algorithm, and his own angular/distance algorithm in a perceptual experiment designed to evaluate the degree to which lines could be simplified before human subjects recognized differences (Jenks, 1989). He found that respondents were able to separate only the excessively simplified lines in a manner consistent with the measured vector displacement, and lines with large numbers of coordinate pairs were so perceptually similar that they were impossible for the respondents to distinguish between (Jenks, 1989, 38). Jenks makes the assertion that, "two lines are identical if the viewer cannot perceive differences between them", and therefore up to seventy-five percent of coordinates in a given line could be considered superfluous when the line is simplified by the Douglas algorithm (Jenks, 1989, 40). However, in making such a statement Jenks is considering maps as used only for displays and not as tools for analysis upon which measurements are made. While maps are used primarily as human interpreted graphical displays, increasingly map based information systems are being used for measurement and analysis. In those instances, the degree of generalization may be a significant factor in influencing the measurements made upon such maps.

While perceptual evaluation of this type is valuable, it has significant limitations. Some of these limitations are the lack of suitable base lines for comparison, and the difficulty in distinguishing between two simplifications which appear to the human eye to be identical (ZYCOR, 1984: Jenks, 1989). For this reason, various parametric measures have been developed to quantitatively assess changes in line characteristics such as position, shape, and angularity with linear simplification. Some of these parametric measures have been combined with perceptual studies to quantify differences in simplifications perceived to be different by human operators.

2.19 Parametric Measures for Characterizing Digital Lines

Parametric measures have been developed for evaluation of linear simplification which compare attributes of single line complexity and which directly measure displacement between a linear feature and its simplification. These two categories of parametric measures have been termed by McMaster, "linear attribute measurements" and, "linear displacement measures" (1986, 108). Single line attributes, or quantitative measurements of linear complexity, can be transformed by ratioing or by standardization to serve as difference measures between a line and its simplification (McMaster, 1986). An evaluation of these measures will be valuable in determining attributes of lines which are important descriptors of linear complexity and those which can measure changes in complexity due to linear simplification. Many of these measures have been discussed by McMaster (1986: 1987a) and Jasinski (1990).

Measures for a single line can quantify aspects of a digital line such as, length, coordinate density, angularity, and sinuosity. The most basic of measurements on digital lines involve computing the distance between points in x,y coordinate space. Length measures compute the Euclidian distance between coordinates along a line, or the distance along segments of the line. The individual segment lengths can be summed to measure the overall line length. Another length measure, termed by Buttenfield (1989) and Jasinski (1990) the "anchor line" length, computes the simple Euclidian distance between the first and last points in the line.

Measures of coordinate density range from a simple count of the number of coordinates to computation of the average spacing between coordinates. A count of the number of points in the line can be ratioed between a line and its simplification to determine the degree of simplification. Coordinate density can be measured by dividing the number of points on a line by the line length resulting in the mean number of coordinates per unit of line (ie. coordinates per inch). A related measure calculates the average segment length between coordinates.

Measures of angularity compute between-segment angular deflections (McMaster, 1986). A useful measure of angularity is the average between segment angularity, or average angularity. This measure can be calculated by summing the between segment deflection angles, itself a useful measure, and dividing by the number of segments in the line. The total angularity in the line can be a useful measure when standardized by line length, accomplished by dividing the total angularity by line length.

Measures of sinuosity characterize a line according to the amount of crenulation, or 'wiggliness' of the line and are useful for assessing the complexity of linear features (Scheidegger, 1970). A ratio of the anchor line length to the overall line length determines a simple sinuosity measure, termed by Unwin (1981) the "sinuosity ratio". The higher the ratio between the anchor line length and the overall line length, the less sinuous is the line (Unwin, 1981).

Another sinuosity measure which has gained in popularity for application in automated cartography is the 'fractal dimension'. The fractal dimension is one aspect of a field of mathematics called 'fractal geometry' which has been applied to characterization of linear complexity in cartography by several authors (Buttenfield, 1985, 1986, 1987, 1989: Goodchild, 1980b: Goodchild and Mark, 1987: Longley and Batty, 1989a,1989b: Muller, 1986, 1987: Carstensen, 1989). The concept of fractal dimension in measuring linear attributes can be traced to Richardson (1961) who examined the phenomenon of increasing measures of length of a line with finer measurement resolutions; the closer the spacing of a pair of dividers walked along a line, the longer the measured length of line became, thus the length of the line tended towards infinity. Richardson discovered that if one plots the natural log of a series of measurements of length against the natural log of measurement interval (or step size of the divider) a predictable, linear relationship is evident (Goodchild and Mark, 1987). These plots are now termed 'Richardson plots' and have been used for close examination of linear structural geometry by Buttenfield (1986: 1987).

Mandlebrot (1982) developed Richardson's ideas into a theory of fractal geometry, whereby the tendency of line length to move toward infinity with finer resolution of measurement was redefined as a ratio of the increase in total line length (r) to the size of equal step lengths (N) required to traverse the line (Buttenfield, 1985, 14). This ratio (D) can be expressed as follows (After Buttenfield, 1985):

$$D = \ln N / \ln(1/r)$$

Due to the fact that the ratio expresses fractional values of dimensionality (traditionally thought of as ordinal quantities, 0 = points, 1 = lines, 2 = areas , 3 = volumes), this relationship was termed a 'fractal' , and describes the complexity of a figure, or the amount of space it fills, in addition to the class of objects to which it belongs (Buttenfield, 1985).

The measurement of fractal dimension has been operationalized in a number of methods (reviewed in Longley and Batty (1989a), however, a popular algorithm for estimation of fractal dimension for linear features is the one proposed by Shelberg, et.al. (1982). This algorithm simulates walking a divider of given step length along a line; the number of steps (and the remainder) needed to to traverse the line are stored and the step length is increased until less than one divider length is needed to traverse the line (Shelberg, et.al., 1982). By taking the absolute value of the slope of the regression line (from regressing the natural logarithm of step lengths against the natural logarithm of number of steps), the fractal dimension is derived (Shelberg, et.al., 1982).

Many recent publications in the cartographic literature have extolled the virtues of the 'fractal dimension' in cartographic analysis and generalization. Longley and Batty (1989) assert that, "measurement and generalization of cartographic lines through an understanding of fractals has a number of advantages over other forms of representation". Some authors have even suggested that fractals are the most important parameter of cartographic lines and have compared their importance to the measurement of arithmetic mean of a sample population (Goodchild and Mark, 1987). Several authors have postulated that fractal measurements are the ideal measure for quantifying the effects of cartographic generalization (Goodchild and Mark, 1987) and have proposed that the preservation of fractal dimension be used as the guideline for effective generalization in automated cartography (Muller, 1986:1987).

While fractal measures have offered a new method of quantifying line complexity, it is becoming increasingly apparent that their utility in evaluating results from cartographic generalization is limited. The initial excitement over the uses of fractals in cartography is beginning to wear thin as researchers are discovering that cartographic lines do not display the self-similarity of fractals across ranges of scale (Carstensen, 1989), and the preservation of fractal dimension in generalization is problematic due to the instability of the measurement on different digital versions of the same line (Jasinski, 1990).

As stated above, comparison measures can be derived by ratioing the characteristic parameters of digital lines between a line and its simplification. In addition, direct comparison measurements of the offset between a line and its simplification can be determined. These parameters quantify the degree to which the simplification is displaced or moved from the position of the original line (Jenks, 1989). Two methods are discussed by McMaster (1986). These are a measure of perpendicular discrepancies between a line and its simplification and a measure of the polygonal areas of offset created when a line and its simplification are overlaid (McMaster, 1986).

2.20 Parametric Evaluation of Simplification Algorithms

White (1985) employed a measure of areal offset in her analysis of perceptual differences in which polygons of displacement between the original line and its simplification were measured and summed for each line simplified. This measure was standardized by dividing by the length of the line and compared between algorithms. The results of this analysis generally supported the perceptual research in that simplifications using the Douglas algorithm exhibited less areal offset than those of the other algorithms, however, results from the perpendicular distance algorithm and the Nth point selection routine were mixed; the perpendicular distance algorithm generally performed better except on on a line of low complexity where the areal offset from the Nth point routine was lower (White, 1985). From these results it is obvious that parametric measures are capable of picking up subtle differences in characteristics of simplified lines which escape simple visual comparison.

McMaster (1986) evaluated 30 parametric measures (some of which are discussed above) and reduced these to 6 unique measures which he advocates be used, "as a set to produce a summary of change that has occurred as a result of the simplification process" (1986, 115). These six measures are:

(1) Percent change in number of coordinates.
(2) Percent change in the standard deviation of number of coordinates per inch of line.
(3) Percent change in angularity.
(4) Total vector displacement per inch.
(5) Total areal displacement per inch.
(6) Percent change in number of curvilinear segments.
(McMaster, 1986, 115).

Using 4 of the 6 unique measures, McMaster conducted an evaluation of line generalization algorithms on four lines: two rivers and two contours (McMaster, 1987a). Through an analysis of percent change in angularity, percent change in curvilinear segments, total vector displacement per inch, and total areal displacement per inch, nine algorithms were evaluated which represented all of the functional categories discussed by McMaster (1987b). These algorithms are the Nth point selection, the Euclidian distance algorithm, Jenk's angular/distance algorithm, the perpendicular distance algorithm, and algorithms proposed by Lang, Johannsen, Reumann-Witkam, Opheim, and Douglas (McMaster, 1987a, 334).

McMaster evaluated the algorithms by comparing the 4 displacement measures with percentage of coordinates eliminated and found that the Douglas algorithm produced less displacement overall than the other algorithms (McMaster, 1987a). However, several algorithms performed as well as or better than the Douglas algorithm in certain instances. The Jenks algorithm was somewhat better at maintaining angularity of the lines, although McMaster claims, "none of the algorithms are better, in terms of maintaining angularity and curvaliniarity, when simplifying lines of varying complexities" (McMaster, 1987a, 338). The Lang algorithm performed as well as or better than the Douglas algorithm in terms of minimizing areal displacement at high levels of coordinate reduction (McMaster, 1987a). McMaster comes to the conclusion that the Douglas algorithm may be algorithmic overkill in some instances, such as thematic cartography, as other less computationally demanding algorithms may perform as well, although the specific instances when this would be appropriate are left unclear (McMaster, 1987a).

ZYCOR, while not directly evaluating linear simplification algorithms nevertheless place the generalization algorithms they review into categories of "high potential" and "low potential" based on "potential for cartographic usefulness" and "ease of control" (1984, 111). Routines classified as low potential algorithms are difficult to control and cartographically suspect such as the Nth point selection routine (ZYCOR, 1984). Potential algorithms are those, "which have potential for use in line generalization but also have fundamental problems in their current state of development" such as angle detection and smoothing (ZYCOR, 1984, 110). High potential routines are algorithms most suited to cartographic generalization such as tolerance band algorithms (ie. Douglas algorithm) (ZYCOR, 1984, 110).

2.21 Further needs for Evaluation of Linear Generalization

While the Douglas algorithm has generally been reported as the best solution at present to automated linear simplification, there are unknowns associated with its use on lines of varying complexity (McMaster, 1987a). Tradeoffs between algorithm efficiency and complexity of linear features are at present unknown. More work along the lines of Buttenfield (1987) and Mark (1989) needs to be completed to relate feature type to generalization operator and guide the selection of tolerance values for appropriate generalization of linear features.

In addition, the methods used for evaluating generalization need to be further developed. A baseline for evaluation of algorithms needs to be established from which different functional classes of algorithms can be compared. Tests of algorithms against random generalization will allow assessment of those algorithms which function in a logical manner in selecting points of importance along a line. In addition the use and limitations of the fractal dimension in evaluating generalization need to be further explored.

More importantly perhaps, the effects of linear generalization upon the representational nature of lines as features needs to be addressed. While the Douglas algorithm has been shown to be effective in simplifying individual lines, no research thus far has attempted to evaluate the effectiveness of this algorithm (or any other) in the context of the map space. Several authors have pointed to this discrepancy such as Brassel and Weibel (1988), Monmonier (1987), and ZYCOR (1984). Monmonier (1987) cites the lack of study of the effects of linear generalization on neighboring lines as evidence of the failure of 'vector-only' approaches in automated generalization. Others have taken the position that computing technology is not sufficiently advanced to consider the effects of linear generalization of one feature upon another (McMaster and Shea, 1988).

The lack of study in this area is an enigma in that lines in the context of the map are often grouped to represent features such as the banks of a river, or contoured topography. The displacement of lines from their original positions, as often occurs in digital generalization, affects their positional accuracy and therefore also affects the representational fidelity of the feature of which the line is a part. It has been reported that positional accuracies are of critical importance to a number of applications (Sinton, 1978) and that digital representations of mapped data should conform to reality to the maximum extent possible considering the limitations of computational efficiency and scale (Muller, 1987: Shea and McMaster, 1989).

In addition, it is the *measurements* on digital map representations which makes their usefulness in Geographic Information Systems so apparent (Goodchild, 1980a). Without knowing the effects of linear generalization on neighboring lines it is difficult to place confidence in the results of analyses conducted on digitally generalized features. This research begins to bridge this gap by examining effects of linear generalization on representations of common map features and by examining results from typical analyses on simplifications of these features.

3. Research Design

3.1 Development of a Platform for Analysis of Linear Generalization

In order to test ideas explored in the previous chapter, several computer programs were developed. This resulted in a test platform from which linear generalization algorithms could be examined and from which parameters of digital lines could be measured. Several programs were written in the C language to evaluate line generalization algorithms from the functional classes developed by McMaster (1987b) and explained above. Routines were developed to calculate characteristic measures of linear complexity (linear attribute measures) and linear displacement (measured as area of offset) between a line and its simplification. In addition, computer programs were written to create random generalizations of lines for comparison between algorithms. Programs were also written which plot lines of random position and orientation upon digitized map features (ie. streams and mountains) so that measurements of stream width and slope could be calculated. Program source code listings are presented in Appendix A.

All of these programs operate on MOSS (Map Overlay Statistical System, U.S. Dept. of Interior, US Geological Survey) formatted ASCII files. This format is a simple text file in which lines are listed by a header record and records of x,y coordinate pairs. The header record identifies the start of a new feature and contains the feature type (O for points, 1 for lines, and 2 for areas), a 'geocode' or feature identifier, and number of points. The header record is followed by a listing of the x,y coordinate pairs, one record per line.

3.2 LINGEN - A Program to test Line Simplification Algorithms

A program was written (LINGEN) which incorporated five simplification algorithms with linear attribute and displacement measures, and which allows generalization and display of individual lines. The five simplification algorithms included in this program are the Nth point selection routine, the Euclidian distance algorithm, the Angular measure algorithm, the Perpendicular distance algorithm, and the Douglas algorithm (explained above and in ZYCOR, 1984: White, 1985). The program allows for simultaneous viewing of original and simplified lines along with associated linear attribute statistics. Appendix A lists the Turbo C source code used to compile the program. This program was used to generate linear simplifications and to evaluate the relative performance of the included algorithms.

The user of LINGEN is prompted to enter file names for three files, the input line file containing the MOSS formatted data, an output file for storing the simplified line coordinates, and an output file to store the calculated error measures between the original and simplified lines. The user is then prompted for the geocode of the feature to process, and the program searches through the input file for the correct geocode identifier. Lines can be practically any size (number of coordinates) as the program uses dynamic memory allocation to store the line. The user is next prompted to select a simplification algorithm, and upon selection of the algorithm is prompted to enter a tolerance value to control the simplification. LINGEN computes the simplification and displays the result on a color graphics monitor.

Linear attribute and displacement measures are calculated on the original and simplified lines and are stored in an error measures file. The measures calculated and stored are the algorithm used, the tolerance value, the geocode of the line processed, the number of coordinates in the original and simplified lines, the length of each line, the average angularity of each line, the sum of angles in each line, the percentage change in number of coordinates, the percentage change in length, the percentage change in average angularity, the area of offset between the original line and its simplification, and the time in whole seconds used by the algorithm. Error measures are written as one record per line processed, and the error measures file can be appended to contain the error measure for several simplifications of the same line.

An optional display from the program is a graphic overlay of the original and simplified lines. This overlay helps to visualize the areas where offsets between the original and simplified lines are greatest and provides a visual check of the amount of simplification induced in the generalized line from a particular algorithm.

3.3 LINESTAT - A Program to Measure Characteristics of Digital Lines

A program was written which computes measures of complexity for digital lines. These measures can be used to characterize changes in digital lines after processing by linear simplification algorithms. LINESTAT reads a MOSS formatted file and sequentially processes every line included in the file. Characteristic measures are computed and stored in an output file, one record for each line processed. The characteristic measures calculated and stored are the length of the line, the anchor line length, the sinuosity or 'wiggliness' ratio, the average segment length, the sum of angles in the line (total angularity), the total angularity standardized by line length, the average angularity, the maximum angle, the minimum angle, and the fractal dimension. These measures are discussed in Chapter 2, and descriptions can be found in Buttenfield (1987), Jasinski (1990), McMaster (1986), and Unwin (1981).

3.4 RANDLG - A Program to Produce Random Simplifications of Digital Lines

In order to evaluate line simplification algorithms in terms of a common baseline, a program was written which constructs random generalizations of a line given a target number of points. Linear attribute and linear displacement measures discussed above are computed and stored in an output file from RANDLG. For each line processed, 30 separate random simplifications are created in memory, complexity measures are calculated, and these measures are stored to an output file. The actual coordinates of each randomly generated simplification are not stored as the output of concern for this study is the attribute measures calculated on each line.

Random simplifications are generated in the following manner using RANDLG. Coordinates from the original line are read into an array. A separate array is created in which the sequence of coordinates is randomized by assigning a random value ranging from 1 to the number of points in the line to the coordinate's array position. The sequence array is searched using the value input by the user for the desired number of output points after simplification. Those coordinates whose random sequence number is less than the target value are selected. For example, if the input line contained 100 points, and the desired simplification was 50 points, an array would be created with randomly assigned values ranging from 1 to 100. This array would then be searched for values less than or equal to 50. The coordinates whose random sequence tag is 50 or less would be selected. This process skips the first and last points in the original line, or the 'nodes', as they are always selected in all algorithms to serve as anchor points for comparison. In this manner the original line is simplified in a random manner, and all coordinate pairs (excluding the nodes) in the line have an equal chance of being selected.

Complexity measures are then computed for each of the thirty randomly generated simplifications and are stored. Complexity measures output from this program are then available to use as a random base against which to compare the results from other simplification algorithms.

3.5 PLOTRIV - A Routine to Plot Pseudo Random Measurement Bisectors over Paired Lines

A program was written in C which creates plot files of MOSS format lines and computes and plots pseudo-random measurement bi-sectors over paired simplified lines. This program reads MOSS format line files, computes the bounding rectangle of coordinates, and plots the lines at either true scale (the scale at which they were recorded), scaled to fit one of two paper sizes (A or B size), or un-scaled and clipped to the appropriate paper size. Output files are written in HPGL format for use with a Hewlett-Packard 7475a series pen plotter (Hewlett-Packard Company, San Diego CA).

Random measurement lines are created by finding intersections of a random y coordinate with pairs of simplified lines. One of the pair of simplified lines is searched for the random y intercept value. If an intersection is found, the x and y coordinate values at that point are calculated and stored and a line of random length and position is constructed which intersects the second in the pair of lines processed. If no intersection is found, a new random intercept value is generated and the process is repeated. The resulting pseudo-random (or 'stratified random') measurement lines are stored so that they may be plotted in the same positions for multiple simplifications of the same set of lines.

3.6 PLOTFEAT - A Routine to Plot Random Measurement Lines Over Multiple Line Features

A program was also written which plots multiple line features (such as contoured topography) using HPGL plotter format files in much the same way as PLOTRIV. This program plots measurement lines of random length, position, and orientation over the bounding window of multiple line feature plots which can be used to measure changes in distance between multiple lines representing features. As in PLOTRIV, these measurement lines are stored in memory so that multiple simplifications of the same feature can be overlaid with the same measurement lines.

Random measurement lines are computed in the following manner. The bounding window of the feature is calculated and the x and y ranges are divided into 30 equal increments and loaded into an array. A random number generator is used to then select random subscripts of these x and y arrays. Pairs of randomly selected x and y coordinates are then used to plot measurement lines. In this manner lines of random length, position and orientation are plotted within the bounding window of the feature which can be used to assess positional changes in lines representing features.

3.7 Test of Simplification Algorithms Against Random Simplifications

An experiment was designed to test whether algorithms from the functional groupings proposed by McMaster (1987b) operate on lines in a geographically logical manner. This test was performed to compare results of algorithms on three distinct lines against a set of randomly produced base lines. The goal was to examine whether simplification algorithms from separate functional groupings performed significantly better than random in the manner in which they selected points to retain. That is, which algorithms take into consideration the representative nature of the lines as features and do not arbitrarily select points in isolation of other points which comprise the feature?

The hypothesis tested was that simplification algorithms which are constrained only to local processing will perform no better than random in terms of producing discrepancies between the original and simplified lines. Those algorithms which consider a greater portion of the line, or the entire line, can be expected to perform significantly better than random, and therefore are considering some aspect of the feature's characteristic shape and position in simplification.

3.8 Test on the Stability of the Fractal Dimension During Simplification

An experiment was conducted to examine the fractal dimension of cartographic lines when measured on digital generalizations of a range of simplifications. The goal was to assess how stable or self-similar lines are under generalization. It was hypothesized that lines would remain self-similar as estimated by the fractal dimension, and would tend to become simpler with generalization (ie. 'simplification').

In order to assess the stability of measurement of the fractal dimension, three digital lines were generalized using the Douglas simplification algorithm at tolerances which would produce approximate reductions in coordinates of roughly 15, 35, 50, 75, and 90 percent. Characteristic measures of length, angularity, and average segment length were computed on each stored generalized version of the line along with the fractal dimension. Tolerance values for the Douglas algorithm were modified for each line to produce approximately the same percentage reduction in coordinates in all three lines. The estimates of fractal dimension were compared within ranges of simplification for the three test lines and with changes in characteristic measures.

3.9 Experiments on the Effect of Automated Generalization on Feature Representations

As a means of assessing the effects of generalization of lines in the context of the map space, an experiment was conducted to assess differences in measurement of common quantities on map features over ranges of simplification. A set of digitized topographic and drainage features was processed using the Douglas line simplification algorithm. The results of the generalizations were quantified using several statistical measures such as change in line length, angularity, and percent change in number of coordinates. Measurements of topographic slope were taken on the generalized surface features, and relative widths between river banks were measured to determine the degree to which the generalized representations differ from the originals.

4. Results and Discussion

4.1 Results from Random Generalization Study

A test was performed to evaluate the use of random generalization as a base against which to compare simplification algorithms. In order to perform this test, the three distinct lines were digitized on a digitizing tablet from various map sheets in 'stream' mode at a table spacing of 0.020 inches. This table spacing allows for a consistently fine sampling of points while avoiding digitizer induced angularity influences in the line (Carstensen, 1990). The lines processed were of varying complexity, and size. Line 'A' was digitized from a stream bank from a Canadian series 1:250,000 map sheet of Fort Hope, Ontario. Line 'B' was a contour digitized from a 1:24,000 scale USGS quadrangle series map sheet of Arapahoe, Wyoming. Line 'C' was digitized from a stream off of the Tom's Brook, Virginia 1:24,000 USGS quadrangle map sheet. The three test lines are illustrated in Figure 4.1 (Note: lines are not displayed at the scale at which they were digitized).

Each of these lines was input to five simplification algorithms at various tolerance levels. The five simplification algorithms used were the Nth point selection, Euclidian distance, Angular tolerance, Perpendicular distance, and Douglas routines. Each of these algorithms has been discussed previously and are representative of the 'selection', 'local processing', 'extended local processing', and 'global' functional class of simplification algorithms discussed by McMaster (1987b). All five of these algorithms were implemented using the LINGEN program previously discussed.

Tolerance value selection for these algorithms was manipulated so that an equivalent percentage of points would be reduced by different algorithms on the same line in order to compare results at similar point reductions across algorithms. Point reduction equivalence was desired at 25, 50, 75, and 90 percent of original points so that a range of simplifications from

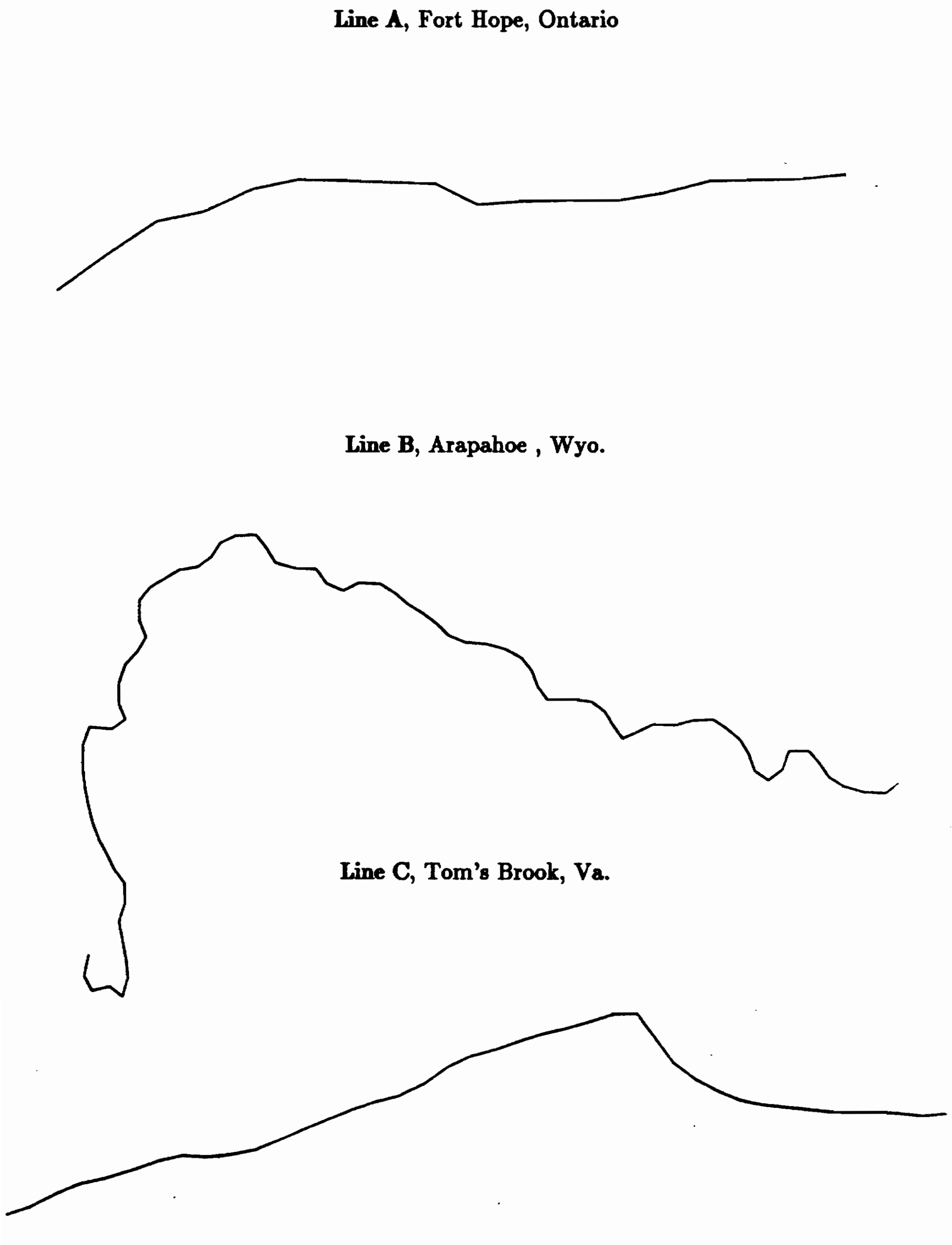


Figure 4.1 Test lines for Random generalization study.

slight to severe could be examined. To achieve equivalence across five algorithms with differing input parameters required that each line be processed repeatedly at varying tolerance intervals before a specific tolerance value could be found which would force a given algorithm to produce a simplification with, for example, a 25 percent point reduction (Table 4.1). This process had to be repeated for each line, as the same tolerance value used to simplify different lines will produce different results. Due to the time consuming nature of selecting equivalent tolerance values for differing lines, the set of test lines had to be severely limited.

Due to the difficulty in controlling the precise number of points output by algorithms, some slight deviations in percentage of points reduced were unavoidable. However, these deviations are never more than 1 percent from the targeted percent reduction. In addition, it was impossible to receive a 25 percent reduction from the Nth point selection routine. This algorithm is seeded to select every Nth point such that with an N value of 1, every point is selected (0 % point reduction), and with an N value of 2, every other point is selected (50% point reduction).

Linear attribute and displacement measures were recorded for each of the simplifications produced from the three lines and are given in Tables 4.2, 4.3, and 4.4. The output number of points for each line and each algorithm was then used as the target value for input to the RANDLG program. This program produced 30 random simplifications of the same line at the given target number of points. Linear complexity and displacement measures for each of the randomly produced generalizations were computed and stored.

A displacement attribute from the simplification algorithms was used for comparison to the randomly simplified lines. For each simplification algorithm, area of offset was used to compare against the area of offset for randomly simplified lines. One-tailed T tests were conducted to evaluate the difference in means. The results from each algorithm at a specific

Table 4.1. Tolerance Value Equivalence.

	Algorithm Tolerance Value (Units)				
Nth Point (N)	E. Dist (inch)	Ang.Tol (degrees)	Perp. Dist. (inch)	Douglas (inch)	approx. % reduced
LINE A					
---	.0215	18	.0015	.0018	25
2	.0290	42	.0050	.0057	50
4	.0700	84	.0108	.0140	75
10	.1950	140	.0180	.0380	90
LINE B					
---	.0209	14.4	.00147	.00157	25
2	.0275	43.05	.0042	.0049	50
4	.0660	106	.0101	.0155	75
10	.1550	172	.0190	.0444	90
LINE C					
---	.0217	5.545	.00053	.00057	25
2	.0298	12.4	.00138	.00178	50
4	.0820	27.1	.00345	.00545	75
10	.2030	54.5	.0077	.0210	90

Table 4.2 Results from simplification, Line A. Tabulation of characteristic measures of length, average angularity, sum of angles, and area of offset between original line and simplification for each of five algorithms.

Tolerance	Points	Length (inches)	Avg. Ang. (degrees)	Sum Ang (degrees)	Area Offset (sq. inches)
		Original line			
--------	188	4.2831	25.5819	1110.9259	0.0000
		Nth Point Simplification			
2.0000	95	4.0802	25.9993	592.6013	0.0098
4.0000	48	3.9264	28.3760	332.4450	0.0186
10.000	20	3.7625	34.0209	162.7560	0.0464
		Euclidian Distance Simplification			
0.0215	141	4.2038	27.9703	924.8575	0.0004
0.0290	95	4.0926	27.8480	632.8136	0.0051
0.0700	48	3.8701	26.0262	309.3452	0.0225
0.1950	19	3.6186	27.5660	129.5026	0.0610
		Angular Measure Algorithm			
18.000	141	4.2812	33.2546	1079.6949	0.0002
42.000	94	4.2594	43.0545	929.9543	0.0058
84.000	47	4.1300	59.3584	646.7706	0.0364
140.00	19	3.8170	67.3946	300.1595	0.0680
		Perpendicular Distance Algorithm			
0.0015	141	4.2818	33.2162	1078.3014	0.0007
0.0050	94	4.2592	43.2374	933.9480	0.0049
0.0108	47	4.1302	55.0620	599.9269	0.0270
0.0180	19	3.8226	55.4027	246.3873	0.0617
		Douglas Algorithm			
0.0018	141	4.2815	32.9627	1070.1437	0.0007
0.0057	94	4.2560	41.2932	892.6218	0.0037
0.0140	48	4.1428	50.4053	559.6824	0.0125
0.0380	19	3.9416	52.2230	225.2364	0.0229

Table 4.3. Results from simplification, Line B. Tabulation of characteristic measures of length, average angularity, sum of angles, and area of offset between original line and simplification for each of five algortihms.

Tolerance	Points	Length (inches)	Avg.Ang. (degrees)	Sum. Ang. (degrees)	Area Offset (sq. inches)
		Original line			
------	817	18.2403	24.5103	1095.1500	0.0000
		Nth Point Simplification			
2.000	409	17.4861	35.7396	831.8635	0.0371
4.000	205	16.0513	46.1716	583.9310	0.1124
10.000	83	13.5052	55.1900	331.0128	0.2633
		Euclidian Distance Simplification			
0.0209	612	17.8749	28.6373	977.2798	0.0115
0.0275	413	17.3991	35.5478	839.7083	0.0220
0.0660	205	15.7285	46.1072	595.0823	0.0789
0.1550	82	13.1538	52.6608	320.2776	0.2364
		Angular Tolerance Simplification			
14.400	613	18.2341	31.8793	1068.2322	0.0011
43.050	409	18.1346	43.5703	977.8591	0.0095
106.000	204	17.3586	62.7089	729.7347	0.0126
472.000	81	11.0336	74.5647	533.8805	0.5086
		Perpendicular Distance Simplification			
0.00147	613	18.2345	31.8896	1068.5547	0.0032
0.00420	409	18.1598	43.5474	975.9882	0.0219
0.01010	204	17.6462	64.3268	736.3612	0.0974
0.01900	82	15.7869	95.6721	484.8180	0.2370
		Douglas Simplification			
0.00157	611	18.2341	31.9308	1066.4562	0.0029
0.00490	403	18.1536	43.5231	961.3920	0.0166
0.01550	205	17.6645	61.0448	701.5253	0.0528
0.04440	83	16.3918	93.4103	461.5878	0.1527

Table 4.4. Results from simplification, Line C. Tabulation of characteristic measures of length, average angularity, sum of angles, and area of offset between original line and simplification for each of five algorithms.

Tolerance	Points	Length (inches)	Avg.Ang. (degrees)	Sum Ang. (degrees)	Mean Area Offset (sq. inches)
		Original Line			
------	412	9.4390	8.3851	364.2249	0.0000
		Nth Point Simplification			
2.000	207	9.3879	11.4972	251.0598	0.0078
4.000	104	9.2929	13.7245	150.6420	0.0261
10.000	43	9.0610	15.6364	70.7527	0.0915
		Euclidian Distance Simplification			
0.0217	309	9.4152	9.5919	312.7636	0.0033
0.0298	207	9.3879	11.4972	251.0598	0.0078
0.0820	104	9.2804	13.9508	153.3313	0.0264
0.2030	44	9.1367	17.4083	80.0233	0.0803
		Angular Tolerance Simplification			
5.545	310	9.4385	10.7564	351.0062	0.0009
12.400	207	9.4344	14.4352	313.6629	0.0033
27.100	103	9.4043	22.6314	243.0562	0.0203
54.500	42	9.2724	35.2240	151.9516	0.1167
		Perpendicular Distance Simplification			
0.00138	309	9.4385	10.8095	351.5941	0.0009
0.00053	206	9.4345	14.4343	312.1087	0.0028
0.00345	103	9.4062	21.7647	233.7006	0.0151
0.00770	43	9.3017	35.1800	155.0665	0.0974
		Douglas Simplification			
0.00178	309	9.4385	10.7746	350.4587	0.0009
0.00057	206	9.4342	14.3918	311.2018	0.0031
0.00545	104	9.4066	20.7499	225.0002	0.0103
0.02100	42	9.3064	28.5284	122.6189	0.0396

percentage point reduction were compared against the 30 randomly generated lines. The mean value of these displacement parameters for the random lines is compared against the actual values from the simplification algorithms in figures 4.2, 4.3, and 4.4. Tables 4.5, 4.6, and 4.7 summarize the findings for the comparison of means for the three lines.

The results from this experiment, while only preliminary due to the small number of lines tested, show that all of the generalization algorithms generally perform better than a merely random selection of points in area of offset produced between the original line and its simplification. The exception to this is the Angular measure algorithm which performs worse than random at high degrees of simplification for two of the lines processed (lines A and B).

This result is interesting in that even the Nth point selection algorithm, generally considered to be the worst generalization routine, at least performs better than random in minimizing discrepancies between the original and simplified lines. This is probably due to the fact that the algorithms tested operate consistently on all sections of a line rather than over or under emphasizing one particular section (as is possible with the random simplification), the discrepancies are generally more evenly distributed across the line than in a merely random simplification. The degree to which *local* discrepancies produced by simplification algorithms influence the representation, however, can not be directly inferred from this experiment although the manner which the length and average angularity change (Tables 4.2, 4.3, and 4.4) give some indication as to the angular areas of offset which are produced.

The performance of algorithms generally follows McMaster's algorithm hierarchy at the lower degrees of simplification as the Nth point selection routine performed worse than the other algorithms, while algorithms which consider larger sections of the line, such as the Douglas algorithm, performed better in minimizing offsets. However, at the higher degrees of simplification (75 and 90 percent) the hierarchy was not as evident as the Nth point selection

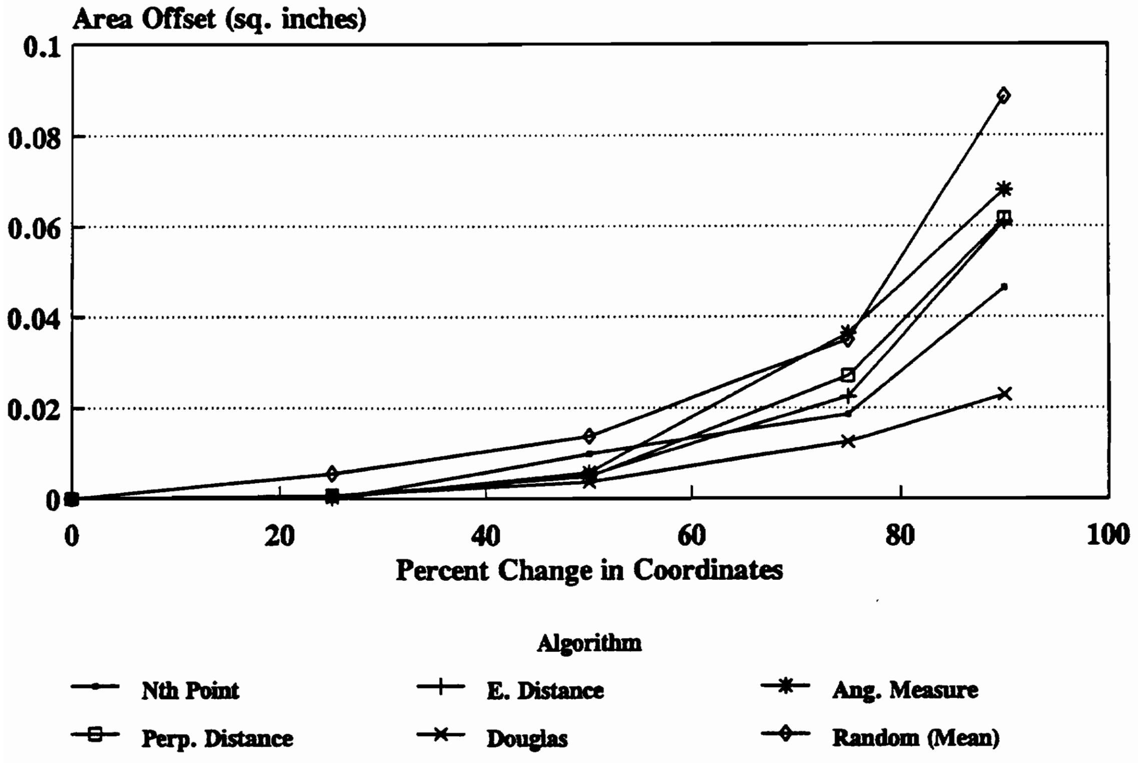


Figure 4.2 Comparison of results from simplification algorithms against random simplifications, **Line A.** Area of offset is plotted against percentage change in points over simplifications of 25, 50, 75, and 90 percent of original points.

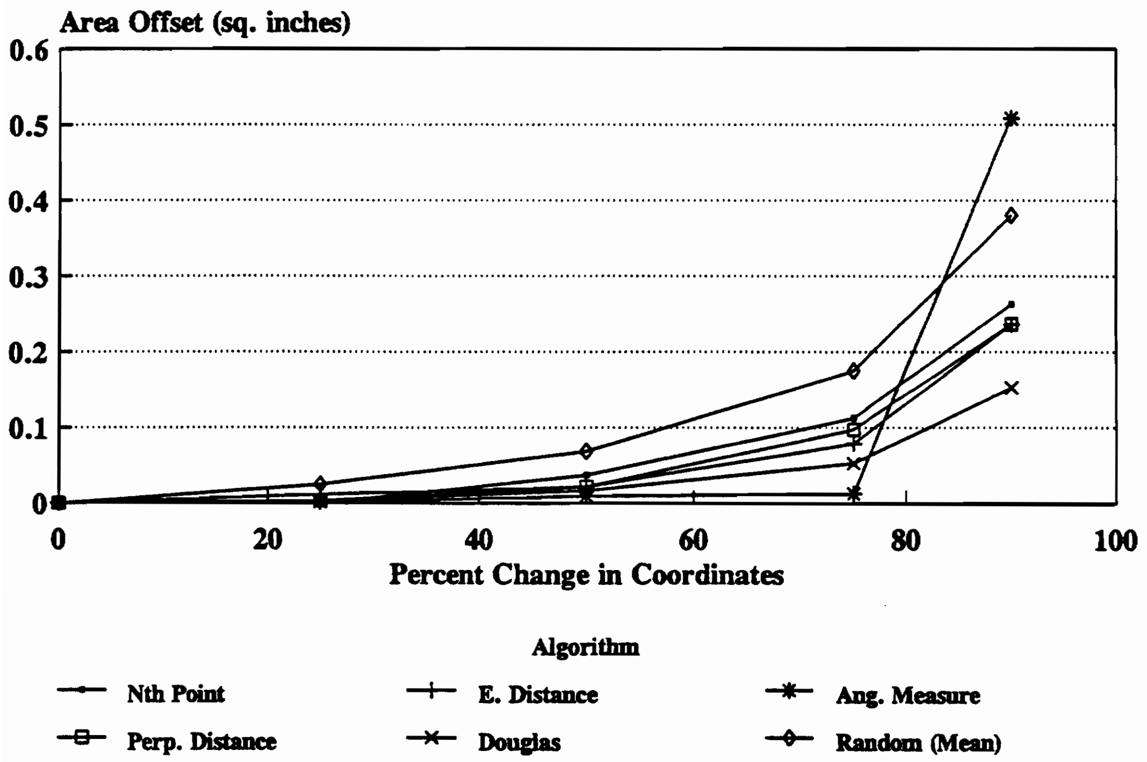


Figure 4.3 Comparison of results from simplification algorithms against random simplifications, Line B. Area of offset is plotted against percentage change in points over simplifications of 25, 50, 75, and 90 percent of original points.

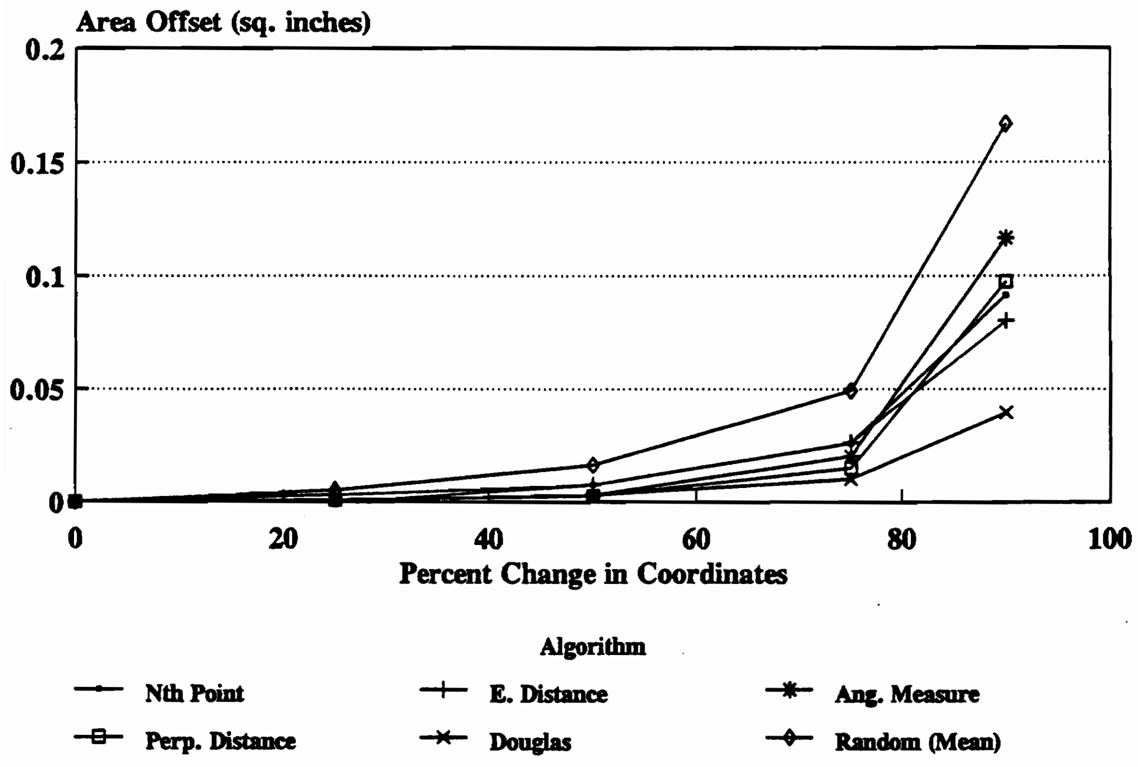


Figure 4.4 Comparison of results from simplification algorithms against random simplifications, Line C. Area of offset is plotted against percentage change in points over simplifications of 25, 50, 75, and 90 percent of original points.

Table 4.5 Results from T-tests of area offset produced by simplification algorithms compared to the mean area offset from 30 random simplifications for **Line A.** Test value is the area offset produced by simplification algorithms; random mean and standard deviation are from area offset calculated on 30 randomly generated simplifications. The Null hypothesis was that the algorithms do not perform better than random in terms of displacement from the original line.

Points	Test value	Rand mean	Std. Dev.	T-value	Null Hypothesis
	Nth Point Simplification				
95	.0098	.0145	.0022	11.764	reject
48	.0186	.0342	.0091	9.519	reject
20	.0464	.0826	.0262	7.678	reject
	Euclidian Distance Simplification				
141	.0004	.0055	.00090	31.525	reject
95	.0051	.0145	.0022	23.513	reject
48	.0225	.0342	.0091	7.152	reject
19	.0610	.0900	.0268	6.024	reject
	Angular Tolerance Simplification				
141	.0002	.0055	.0009	32.760	reject
94	.0058	.0132	.0016	24.547	reject
47	.0364	.0367	.0070	.2437	*fail*
19	.0680	.0900	.0268	4.572	reject
	Perpendicular Distance Simplification				
141	.0007	.0055	.0090	29.672	reject
94	.0049	.0132	.0016	27.530	reject
47	.0270	.0367	.0070	7.642	reject
19	.0617	.0900	.0268	5.879	reject
	Douglas Simplification				
141	.0007	.0055	.0090	29.672	reject
94	.0037	.0132	.0016	31.508	reject
48	.0125	.0342	.0091	13.222	reject
19	.0229	.0900	.0268	13.928	reject

Table 4.6 Results from T-tests of area offset produced by simplification algorithms compared to the mean area offset from 30 random simplifications for **Line B.** Test value is the area offset produced by simplification algorithms; random mean and standard deviation are from area offset calculated on 30 randomly generated simplifications. The Null hypothesis was that the algorithms do not perform better than random in terms of displacement from the original line.

Points	Test value	Rand mean	Std. Dev.	T-value	Null Hypothesis
	Nth Point Simplification				
409	.0371	.0678	.0047	36.013	reject
205	.1124	.1759	.0225	15.672	reject
83	.2633	.3899	.0498	14.140	reject
	Euclidian Distance Simplification				
612	.0115	.0244	.0020	34.781	reject
413	.0220	.0692	.0038	68.911	reject
205	.0789	.1759	.0225	23.939	reject
82	.2364	.3725	.0688	11.009	reject
	Angular Tolerance Simplification				
613	.0011	.0248	.0021	60.998	reject
409	.0095	.0678	.0047	68.297	reject
204	.0126	.1740	.0160	56.070	reject
81	.5086	.3785	.0628	- 11.518	*fail*
	Perpendicular Distance Simplification				
613	.0032	.0248	.0021	55.611	reject
409	.0219	.0678	.0047	53.793	reject
204	.0974	.1740	.0160	26.617	reject
82	.2370	.3725	.0688	10.960	reject
	Douglas Simplification Algorithm				
611	.0029	.0243	.0016	74.530	reject
403	.0166	.0706	.0051	58.491	reject
205	.0528	.1759	.0225	30.381	reject
83	.1527	.3899	.0498	26.491	reject

Table 4.7 Results from T-tests of area offset produced by simplification algorithms compared to the mean area offset from 30 random simplifications for **Line C.** Test value is the area offset produced by simplification algorithms; random mean and standard deviation are from area offset calculated on 30 randomly generated simplifications. The Null hypothesis was that the algorithms do not perform better than random in terms of displacement from the original line.

Points	Test value	Rand mean	Std. Dev.	T-value	Null Hypothesis
		Nth Point Simplification			
207	.0078	.0164	.0034	13.8631	reject
104	.0261	.0510	.0073	18.962	reject
43	.0915	.1605	.0054	7.1049	reject
		Euclidian Distance Simplification			
309	.0033	.0054	.0006	18.4869	reject
207	.0078	.0164	.0034	13.8631	reject
104	.0264	.0510	.0073	18.734	reject
44	.0803	.1665	.0049	9.7805	reject
		Angular Tolerance Algorithm			
310	.0009	.0051	.0007	31.7014	reject
207	.0033	.0164	.0034	21.0550	reject
103	.0203	.0468	.0089	16.505	reject
42	.1167	.1732	.0041	7.4984	reject
		Perpendicular Distance Algorithm			
309	.0009	.0054	.0006	39.1080	reject
206	.0028	.0167	.0026	29.748	reject
103	.0151	.0468	.0089	19.742	reject
43	.0974	.1605	.0054	6.498	reject
		Douglas Algorithm			
309	.0009	.0054	.0006	39.1080	reject
206	.0031	.0167	.0026	29.1088	reject
104	.0103	.0510	.0073	30.9680	reject
42	.0396	.1732	.0419	17.728	reject

algorithm and the Euclidian distance algorithm performed as well or better than the perpendicular distance and angular measure algorithms (lines A and C). This seems to be especially true of lines with lower initial angularity (line C) suggesting that for these lines, a simple weeding of coordinates as with the Euclidian distance algorithm may be sufficient to simplify the line while maintaining fidelity to the original line.

All of the algorithms tended to produce a much larger area of offset after simplifications of 75 percent and above with the lines tested. The angular measure algorithm also failed the random test only at these higher levels of simplification. These results tend to support the notion that after seventy-five percent of the points in a line are reduced, the line becomes much more angular, and the offset between a line and its simplification increases dramatically, decreasing the degree of representativeness of the simplification. Even the Douglas algorithm, shown here and in other studies (White, 1985: McMaster, 1987a) to generally be the best in minimizing discrepancies, performs proportionally worse at the higher levels of simplification. The increase in area of offset does not appear to be a linear, geometric increase, but rather an exponential increase after this degree of coordinate reduction.

4.2 Results From Fractal Geometry Test

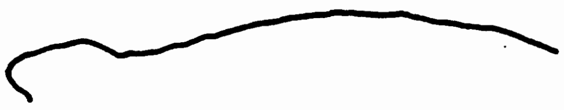
As described previously, an experiment was devised to gain a cursory understanding of the changes in estimates of fractal dimension over ranges of simplification. Three lines were digitized from paper maps and generalized to varying degrees using the Douglas simplification algorithm. The fractal dimension was then calculated on the digital simplifications using the software platform described in Chapter 3. Characteristic measures calculated on each line simplification from the LINGEN software were then compared to changes in the fractal dimension. The goal was to assess whether the fractal dimension changed (ie. decreased) in a predictable manner over simplification which would suggest a predictable guideline for simplification of an individual line.

The three lines used in this analysis were digitized from different USGS 7.5 minute quad sheets in stream mode at 20/1000 of an inch table spacing to provide a consistent sampling of the graphic lines. The three original lines vary in length, complexity, and feature type. Line 'D', McClain's Run, is a small stream digitized from the Tom's Brook, Va. 1:24,000 USGS quad sheet. Line 'E' is a section of the regularly meandering Shenandoah River in Virginia from the Tom's Brook, Va. quadrangle. Line 'F' is a section of the 5500 foot contour digitized from the Arapahoe, Wyoming 1:24,000 USGS quad sheet. Figure 4.5 illustrates the test lines used for this study.

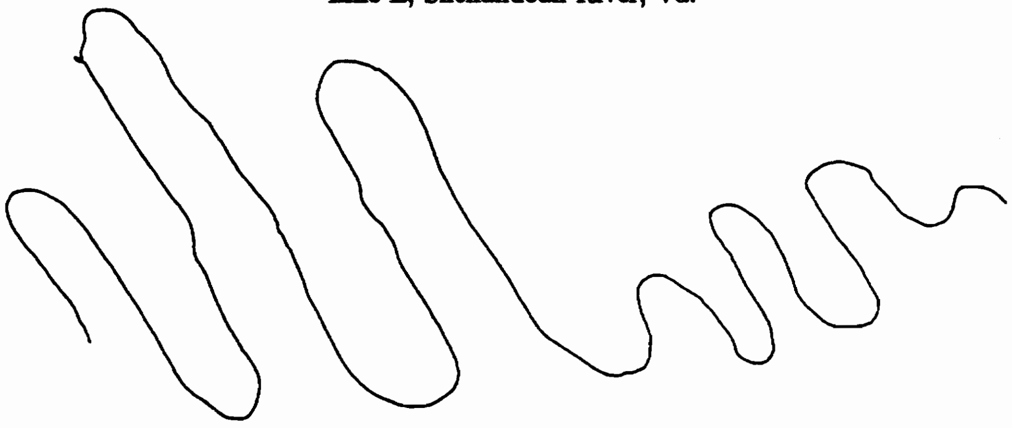
Characteristic measures of length, angularity, and average segment length were computed on each generalized version of the test lines along with the fractal dimension and are presented in Table 4.8. Tolerance values used for the Douglas algorithm were modified for each line to produce approximately the same percentage reduction in coordinates in all three lines.

The fractal dimension was computed for each original line and its successive generalizations by the "divider walk" algorithm developed by Shelberg, et.al. (1982) and used by Carstensen (1989), Goodchild and Mark (1987), Longley and Batty (1989), and Jasinski(1990). The algorithm operates by simulating a divider walk along a line with the spacing for the divider set initially to one-half of the average segment length of the original (digitized) line. Thereafter, the divider length is increased in geometrical progression until less than one length of the divider span can cover the entire line. The LOG of the number of steps necessary to cover the line is regressed against the LOG of the step length. The slope of the regression line gives the fractal dimension D by the equation [D = 1-beta]. All R^2 values from the regression were over .97. Table 4.8 lists the characteristic measures and estimates of fractal dimension

Line D, McLain's Run, Va.



Line E, Shenandoah River, Va.



Line F, Arapahoe Contour, Wyo.

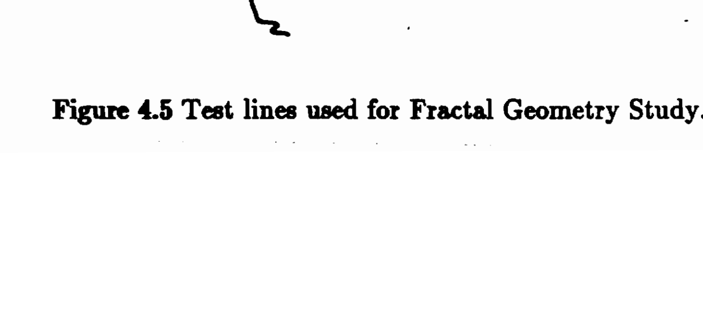


Figure 4.5 Test lines used for Fractal Geometry Study.

Table 4.8 Characteristic measures and fractal dimension calculated on simplifications of test lines. Measures presented are number of points in the line (Pts.), line length, average segment length (Seg. Len.), average between segment angularity (Avg. Ang.), sum of between segment angles divided by line length (Sum. Len.), minimum between segment angle (Min. Ang.), and estimate of fractal dimension (Fract. D.).

Pts.	Length (inches)	Seg. Len. (inches)	Avg. Ang. (degrees)	Sum. Len. (degrees)	Min. Ang (degrees)	Fract. D.
		Line D, McClain's Run				
173	3.98	.023	4.95	212.70	0.00	1.018
148	3.98	.027	5.82	210.83	0.53	1.015
114	3.98	.035	6.97	196.24	1.16	1.021
84	3.98	.048	8.65	178.24	1.25	1.020
58	3.97	.070	10.87	153.16	1.83	1.023
35	3.97	.117	13.91	115.54	4.24	1.027
15	3.95	.282	21.72	71.42	5.36	1.026
		Line E, Shenandoah River				
2002	49.11	.025	4.42	180.01	0.00	1.192
1296	49.11	.038	6.22	164.02	0.14	1.204
945	49.10	.052	7.51	144.14	0.14	1.220
618	49.08	.080	9.40	117.91	1.34	1.238
389	49.05	.126	11.89	117.11	2.03	1.261
169	48.91	.291	19.70	139.24	2.30	1.312
		Line F, Arapahoe Contour				
817	18.24	.022	24.51	1095.15	0.00	1.274
665	18.24	.027	29.70	1079.75	2.24	1.273
574	18.23	.032	33.69	1056.94	3.04	1.284
444	18.18	.041	40.62	987.51	4.40	1.318
311	18.03	.058	50.44	864.57	7.95	1.284
198	17.63	.089	62.56	695.67	12.56	1.327
94	16.63	.179	87.51	484.25	24.09	1.347

calculated on the three test lines over ranges of simplification.

Plots were produced of the progression of the fractal dimension with average angularity, average segment length, and minimum angle over successive reductions in the number of coordinates. Figures 4.6, 4.7, and 4.8 display the results for line 'D'; Figures 4.9, 4.10, and 4.11 display the results for line 'E' , and Figures 4.12, 4.13, and 4.14 display the results for line 'F'.

As evidenced by the plots of fractal dimension and by the lists of fractal dimensions in Table 4.8, the fractal dimension fluctuates noticeably for different digital versions of the same line. In all of the lines there was an overall increase in the fractal dimension as estimated from the original line to the 90 percent simplification. In the case of the Shenandoah River (line 'E') the increase in D was dramatic going from 1.192 to 1.312. These results mirror those of Jasinski (1990) who found a similar fluctuation and general increase in fractal dimension with increasing generalization with the divider walk algorithm. These results demonstrate a definite lack of self similarity among the different digital versions of the same line using the divider walk method.

The average angularity displayed a similar increase for all lines, showing maximum increases of from 20 to 80 degrees (Figures 4.6, 4.9 and 4.12). The progression in the increase in average angularity was much smoother than that of the fractal dimension for all three lines and all of the curves are of similar shape. Figures 4.7, 4.10, and 4.13 display plots for the fractal dimension and minimum angles in the lines. From these plots it is obvious that the minimum angles in the lines are increasing. The increase begins slowly at low percentage reductions but rises rapidly above 50 percent of the points retained with. Figures 4.8, 4.11, and 4.14 display the resulting plots of changes in average segment length. These plots demonstrate the smooth increases in average segment length with successive generalization.

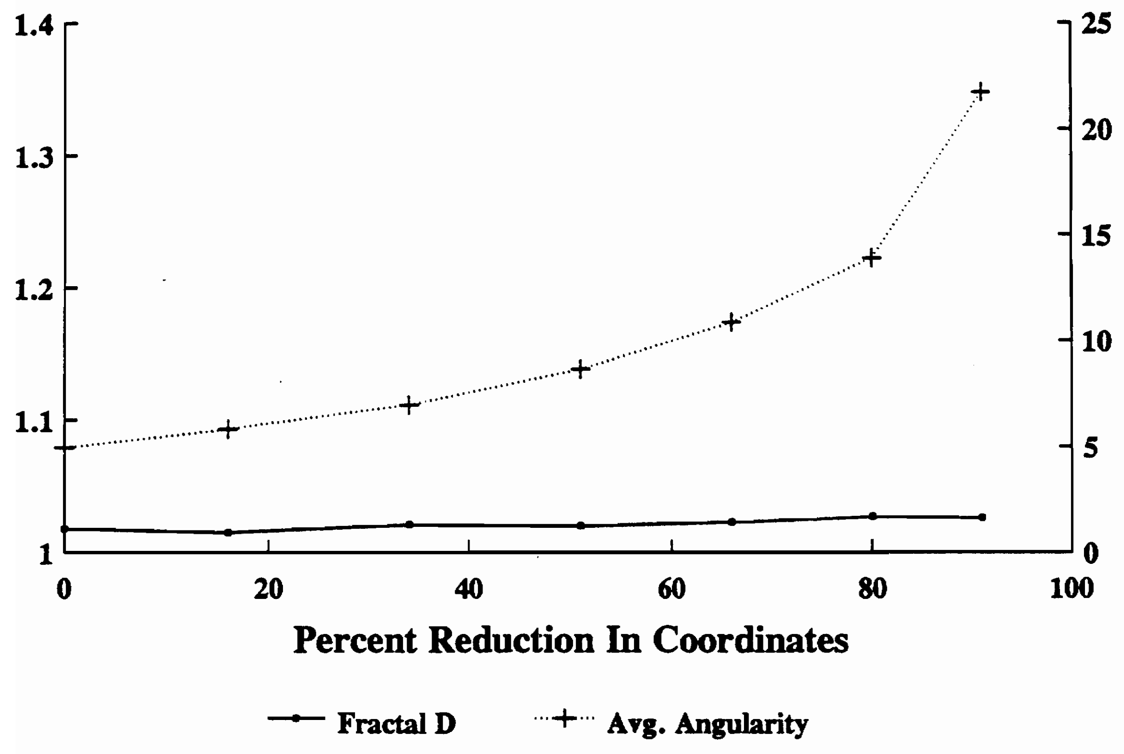


Figure 4.6 Comparison of measurements of Fractal Dimension against Average Angularity for Line D (McClain's Run).

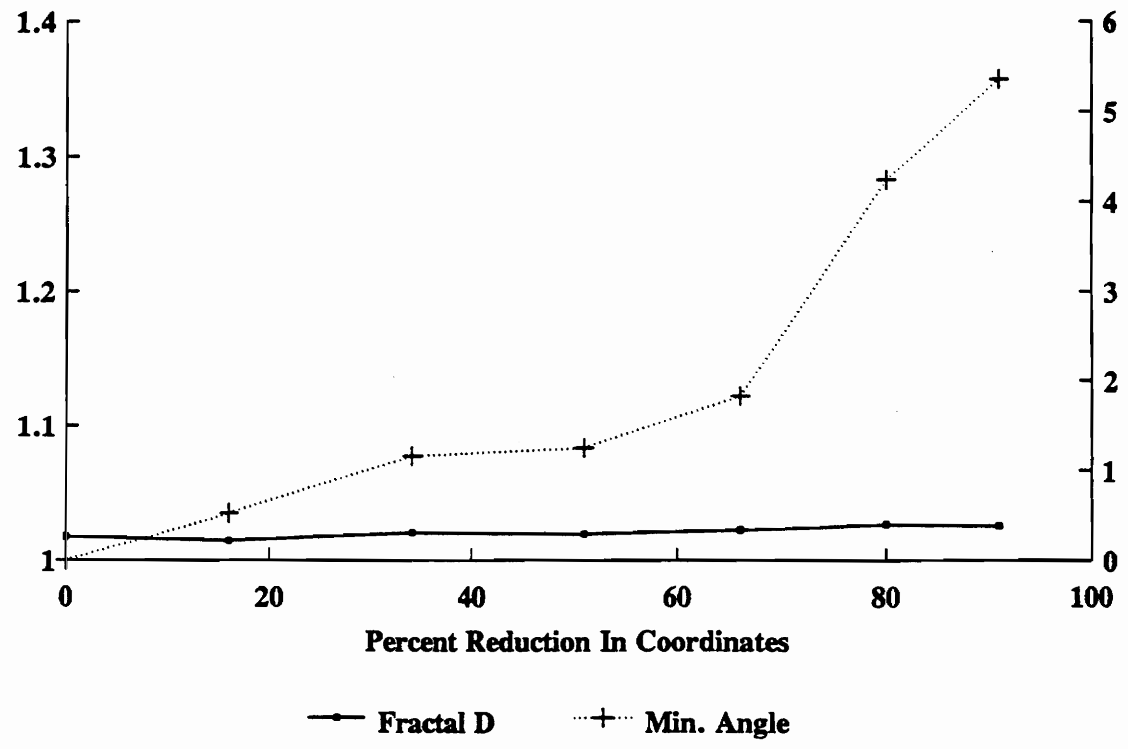


Figure 4.7 Comparison of measurements of Fractal Dimension against Minimum between segment Angle for Line D (McClain's Run).

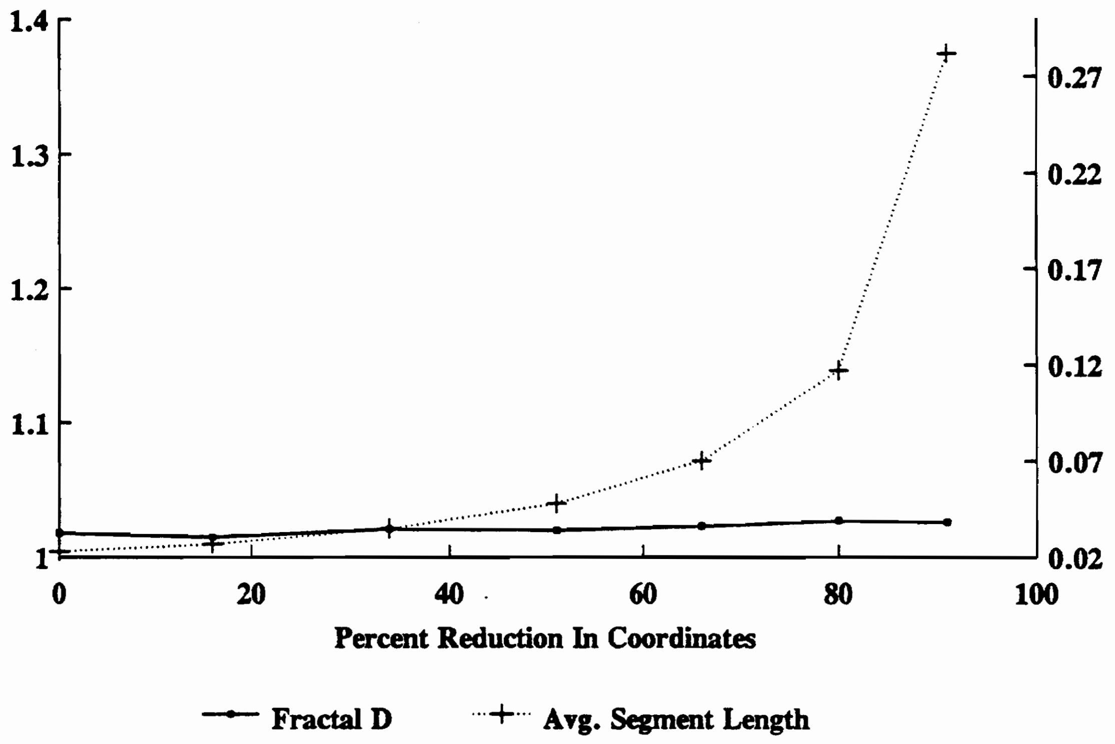


Figure 4.8 Comparison of measurements of Fractal Dimension against Average Segment Length for Line D (McClain's Run).

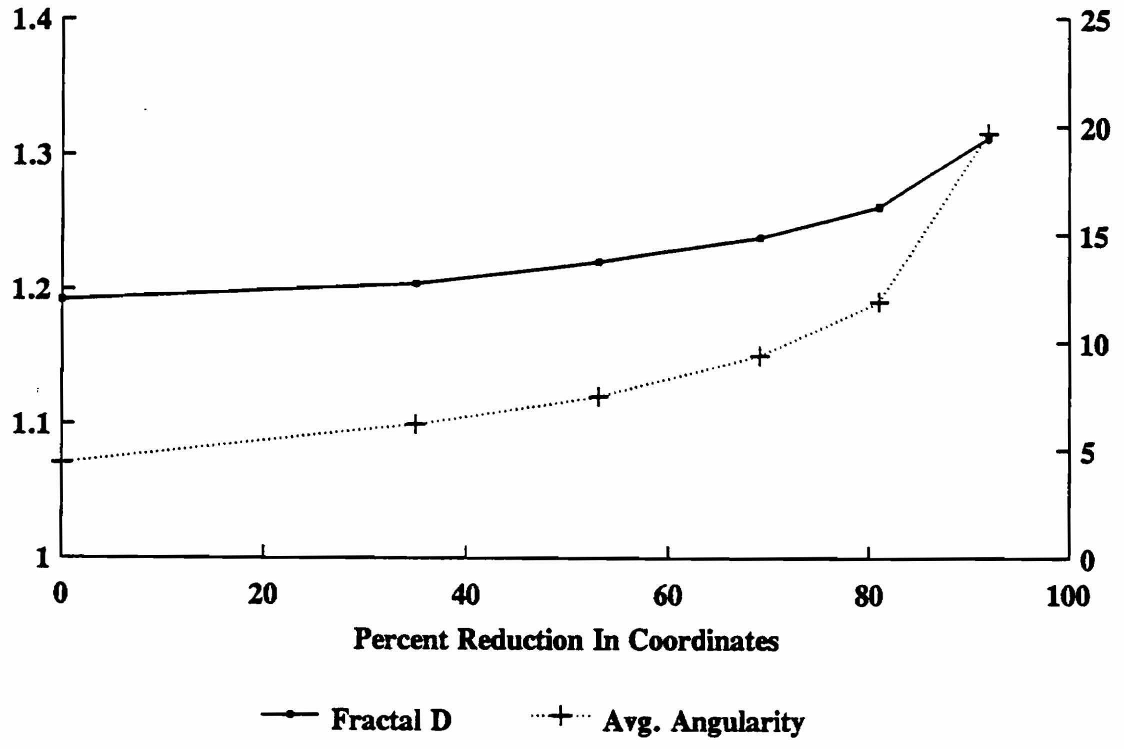


Figure 4.9 Comparison of measurements of Fractal Dimension against Average Angularity for Line E (Shenandoah River).

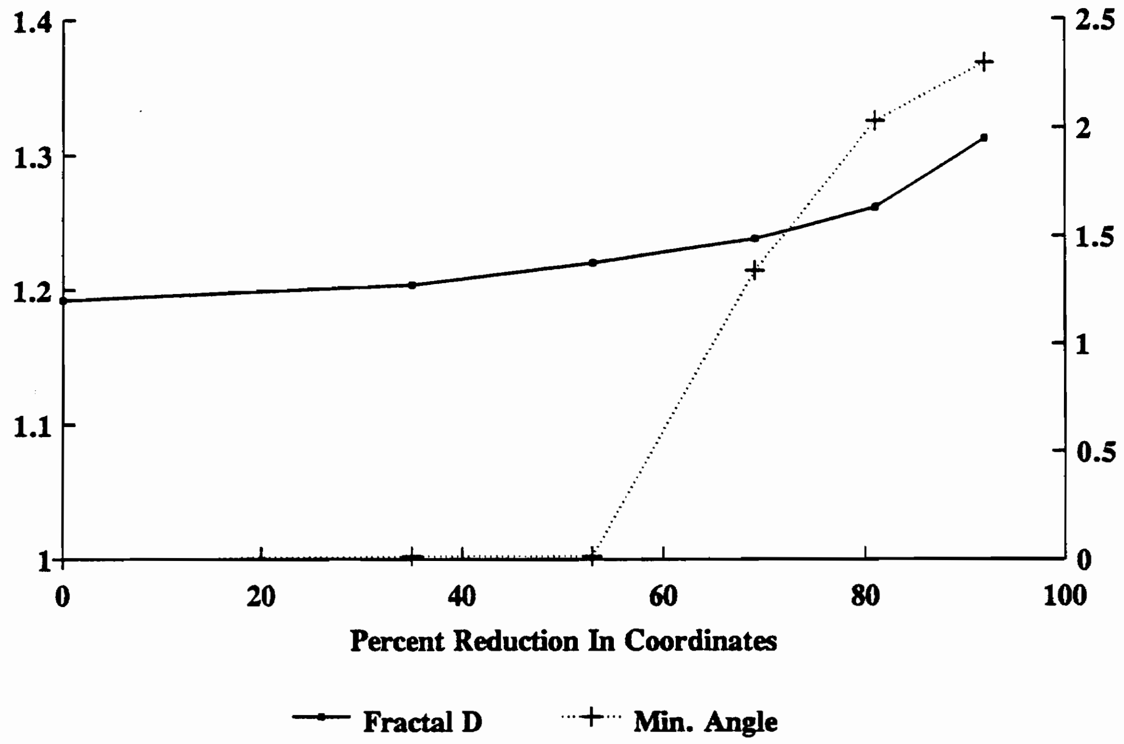


Figure 4.10 Comparison of measurements of Fractal Dimension against Minimum between segment Angle for **Line E** (Shenandoah River).

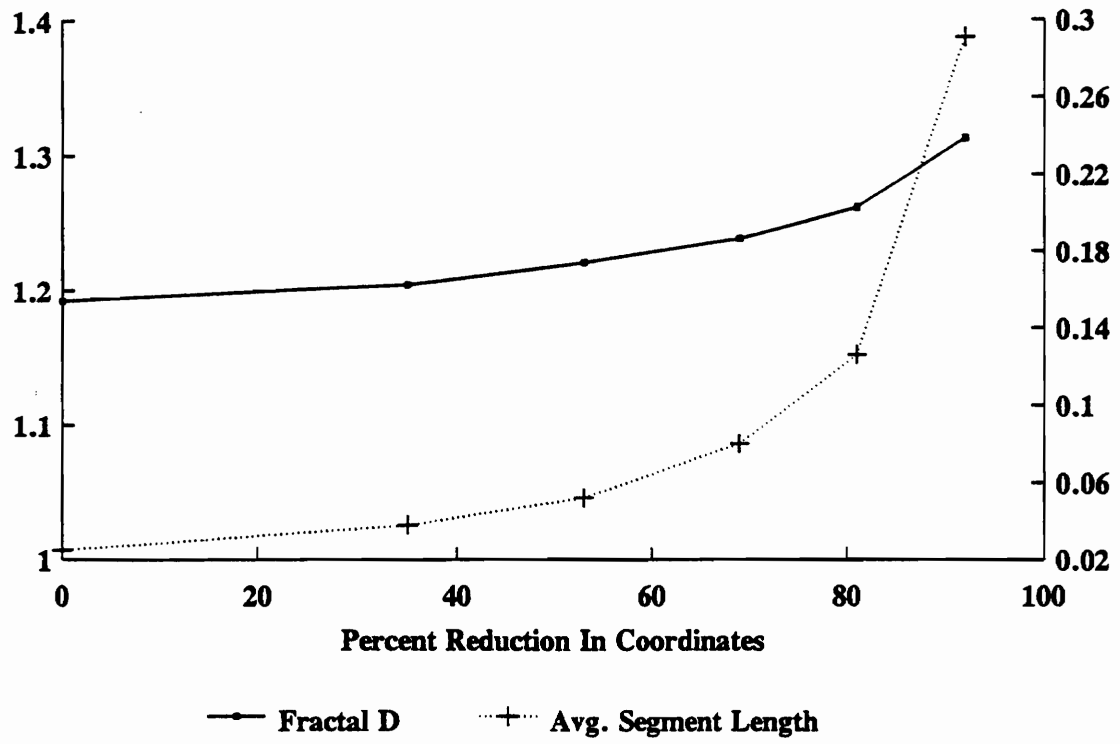


Figure 4.11 Comparison of measurements of Fractal Dimension against Average Segment Length for Line E (Shenandoah River).

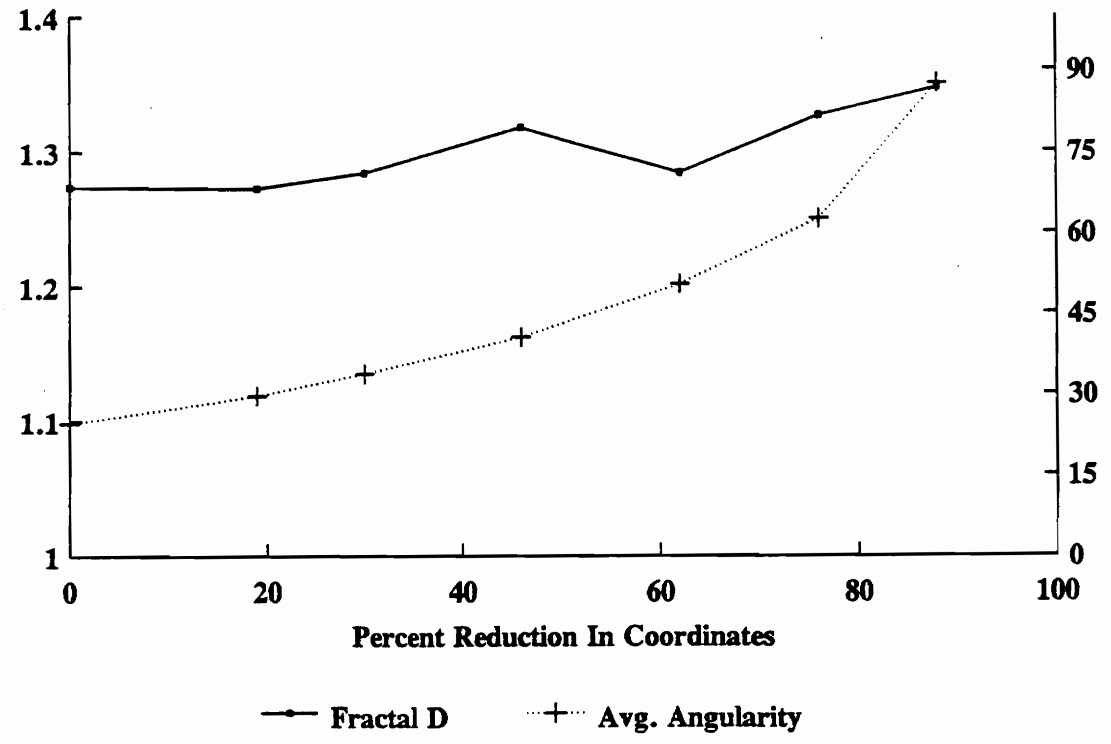


Figure 4.12 Comparison of measurements of Fractal Dimension with Average Angularity for Line F (Arapahoe Contour).

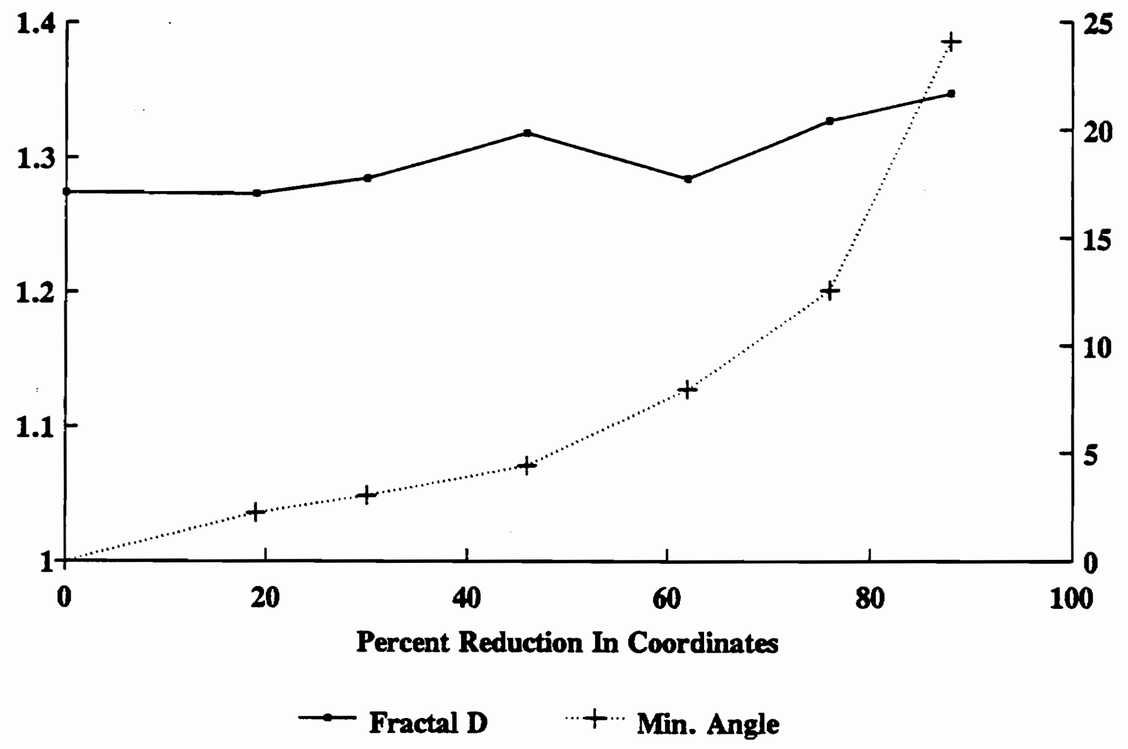


Figure 4.13 Comparison of measurements of Fractal Dimension Minimum between segment Angle for **Line F** (Arapahoe Contour).

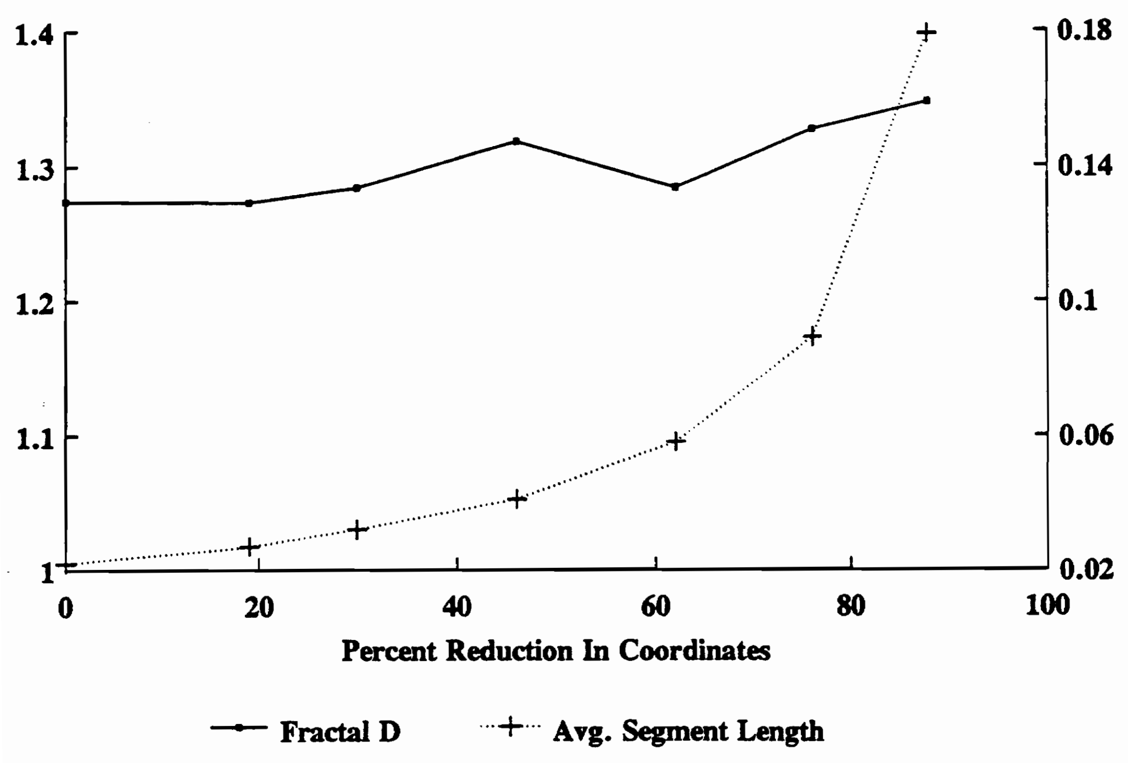


Figure 4.14 Comparison of measurements of Fractal Dimension with Average Segment Length for **Line F** (Arapahoe Contour).

These results suggest several reasons for the increases in fractal dimension estimates among the sample lines. With increasing generalization of a digital line, fewer segments make up the line and the sum of angles in the line decreases, at the same time the between segment angles and lengths increase. The result is a line that has much sharper angular changes. This relationship exists even for small amounts of point reductions. Figure 4.15 dramatically demonstrates the increases in angularity in the Arapahoe Contour (line 'F') at successive generalizations.

The increase in angularity leads to an increase in the fractal dimension because of the operation of the algorithm. The divider walk algorithm operates by finding an intersection with a segment of the line from a circle of radius step length from the starting point and then from subsequent intersections. It is asserted here that the increase in between segment angularity and the increase in sharp angular changes decreases the opportunities for intersections to be found from a divider of a given step length, therefore decreasing the number of steps necessary at larger step lengths. This would lead to a higher fractal dimension estimate. The remainder left at the end of the divider's walk to the last point is different for different versions of the line. These differences may be the reason for the fluctuation in the fractal dimension in a series of point reductions.

The fluctuations in fractal dimension are more severe for lines of higher initial complexity and crenulation such as the Arapahoe Contour, then for lines of smaller initial complexity such as McClain's Run. Subsequent generalization of highly crenulated lines such as Arapahoe Contour produces much higher relative angularities than for straighter lines such as McClain's Run. Smoothly angular lines such as Shenandoah River display a steady increase in the fractal dimension but not the wild fluctuations of lines such as Arapahoe Contour.

It is intuitively assumed that in generalization one is creating a more simple line from

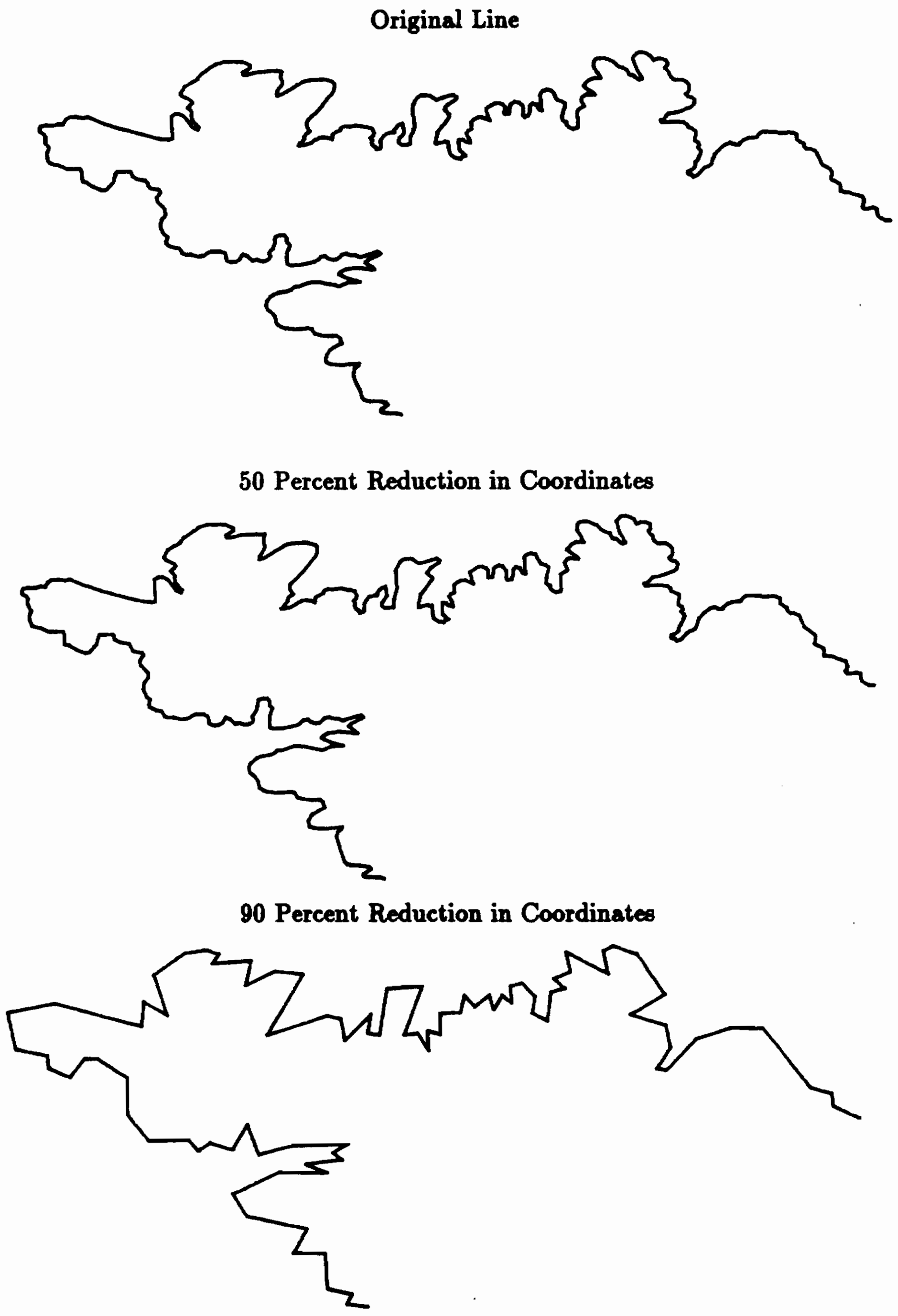

Original Line
50 Percent Reduction in Coordinates
90 Percent Reduction in Coordinates

Figure 4.15 **Effect of generalization upon the angularity of Line F** (Arapahoe Contour).

the original (hence the term 'simplification') so that less detail remains either for display at smaller scales or for reduction in storage. It therefore follows that the complexity of the linear feature will be reduced as one generalizes from a more complex original line to a simpler generalization. Due to the fact that the fractal dimension is a complexity measure, it would seem logical that the reductions in complexity with increasing generalization would be mirrored by the fractal dimension. This relationship is implied by Muller (1986) who shows a general decrease in fractal D at smaller scales from manually generalized lines.

Muller asserts that the preservation of fractal dimension should therefore be used as a guide for generalization. However, it is suggested from the results of this study that the fractal complexity of digital lines may actually increase with successive generalizations. Furthermore, the increases are not in a regular fashion but fluctuate, sometimes severely. The results of Jasinski (1990) support this assertion and even Muller's (1986) results show substantial fluctuations. It appears that the use of the fractal dimension as a guide for generalization is therefore highly questionable, and it is suggested that placing too much faith in measurements of fractal dimension over scale changes can be dangerous (the same conclusions reached by Carstensen (1989)). This could be further established with more research in this area.

Another result which is suggested from this study is the dramatic increase in angularity and angularity related characteristic measures after a reduction in coordinates of 75 percent or greater (Figures 4.8 to 4.14). Angularity measures increase almost linearly up to this point, after which they rise in a seemingly exponential manner. These results are mirrored in the random generalization study, and although the number of lines tested in this study is too small to suggest firm breaks, it appears that the 'cusp' of increase in discrepancy measures occurs in approximately the same place on all simplifications. This result may point to a logical breaking point after which it may be necessary to recompile a line for display at smaller scales.

4.3 Results From Feature Representation Test

In an effort to address the question of generalization effects on multiple line features (ie. feature representations), a set of five contoured features and five paired stream bank lines were generalized and examined for changes in representation. While there is some debate on the use of contoured representations of topography for analysis of generalization (see Mark (1989) and Brassel and Weibel (1988)), it was felt that sufficient opportunity existed for the use of these data (i.e., USGS Digital Line Graphs) to warrant attention.

All of the test features were digitized from USGS 1:24,000 map sheets. Features were captured as sets of lines which were digitized in stream mode at a distance increment of .020 in. Sets of lines representing features were processed individually using the Douglas algorithm programmed in the C language. The Douglas algorithm operates iteratively by retaining points which fall within a user-specified tolerance distance from successive segment comparators (described in detail above). Due to the fact that very little guidance exists in the literature on selection of input parameters for simplification algorithms, a rather arbitrary selection of five tolerance levels was chosen based upon experimentation with the algorithm. The tolerance levels used produced approximate data reductions of 50, 60, 75, 85, and 90 percent depending upon the complexity of the line. The simplified lines were then plotted at true scale using a .76 mm (.03 in.) width plotter pen.

In order to measure effects of generalization on positional changes in lines, the program PLOTFEAT was used to place lines of random length, position, and direction over plotted contoured features, and PLOTRIV was used to place random bisectors over the plotted pair of stream bank lines. Random measurement lines were produced at the same positions for all five simplifications of each feature. From the random measurement lines overlaid on contoured features, 'partial' slope was calculated. Muehrcke (1986) defines partial slope as slope calculated in any single direction as opposed to gradient which is calculated for the steepest direction only. Partial slope was used so a truly random placement of measurement lines could be generated to assess changes in feature representation equally throughout the entire feature. Figures 4.16 and 4.17 illustrate examples of the test features for the paired stream banks and the contoured topography and five simplifications overlaid with random measurement lines.

Measurements of stream width were conducted on the random measurement bisectors overlaid on paired stream lines. Measurements of partial slope and stream width were made manually using a reticule and engineer's scale to the nearest one-hundredth of an inch. On the paired stream lines, measurements of width were made between 'banks' of the feature at the intersections of measurement lines. On the grouped contour lines, measurements of width between selected lines were conducted along the intersection between the contours and the measurement lines. Using the reticule, measurements were made as close to the center of the plotted line as possible. However, uncertainty in precisely locating the center vector from the plotted lines may lead to an error tolerance at least as large as the width of the line (.03 inch). A more accurate assessment of the line center could have been achieved by programming an algorithm to compute the intersection of the random measurement line with the plotted feature. However, limited resources prevented development of this method at the present time.

For each of the ten features, five random measurements were conducted for each of the five simplifications resulting in 250 measured values. Partial slope was calculated for the topographic features using the elevation of each contour line as recorded from the USGS source map. A rule was established such that only that portion of the measurement line which went downhill was measured to avoid breaks in slope. The resulting slope measurements were tabulated as depicted in Figure 4.18. Means were then calculated down the columns of the tabulation to give a measure of total feature measurement changes by simplification.

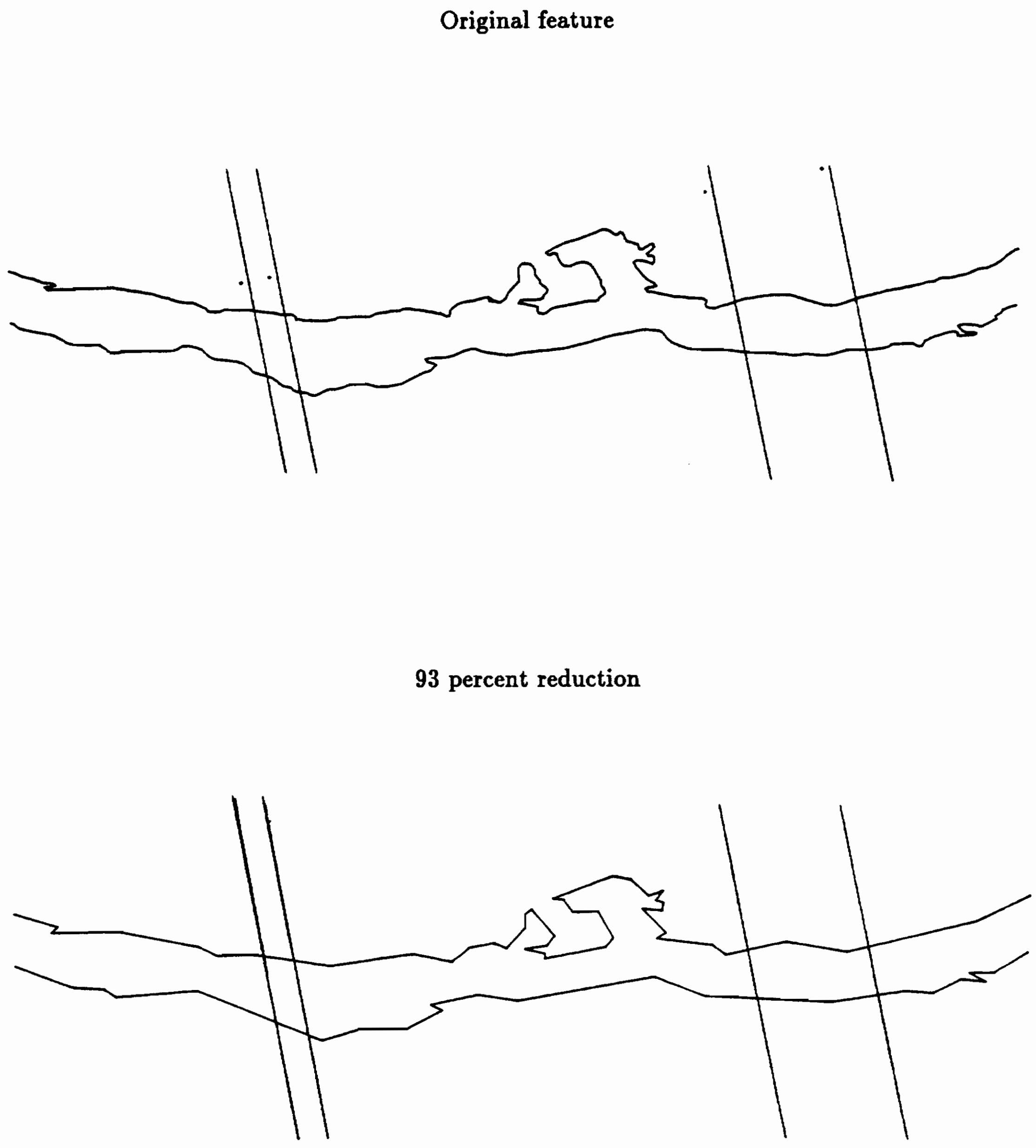
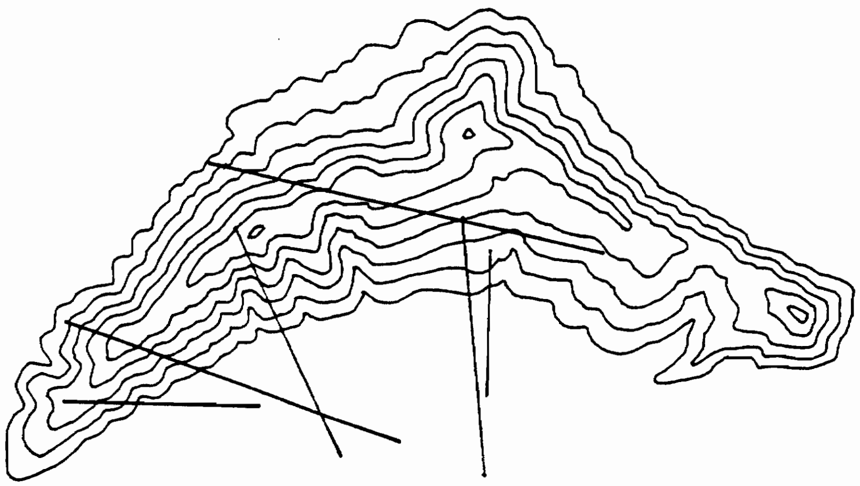


Figure 4.16 Example of paired stream banks used for feature simplification study. Arkansas River is shown with coordinate reductions of 0, and 93 percent overlaid with measurement lines.

Original feature



90 percent reduction

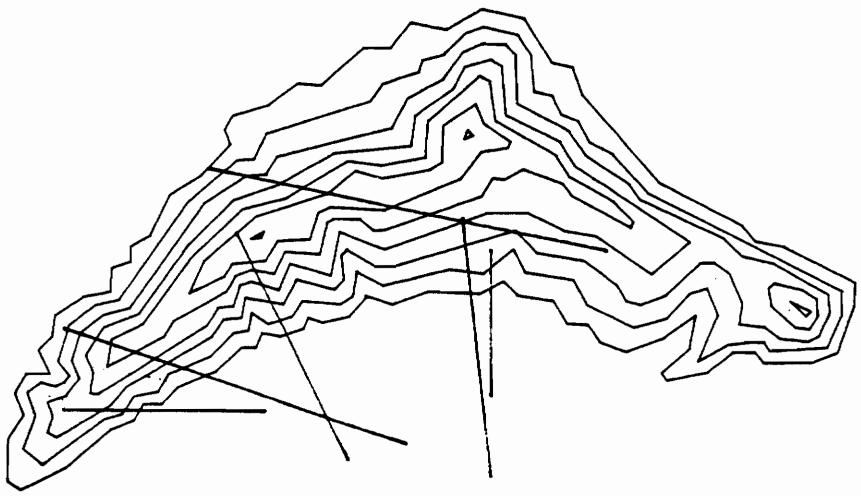


Figure 4.17 Example of contoured topography used for feature simplication study. Sallings Mtn. is shown with coordinate reductions of 0, 75, and 90 percent overlaid with measurement lines.

Sallings Mountain

		Simplification					
		0	1	2	3	4	5
	1	.270	.270	.270	.270	.263	.270
Meas.	2	.183	.183	.183	.181	.181	.179
Line	3	.431	.431	.431	.431	.420	.446
	4	.288	.288	.288	.283	.288	.300
	5	.435	.435	.426	.426	.404	.400

Figure 4.18 Example of data tabulation for feature representation study. Partial (percent) slope was calculated by measurement line for each simplification. Partial slopes were then summarized down the columns to give a deviation value for the entire feature at each simplification.

Measurements for stream widths were tabulated and averaged across measurement lines by simplification in a similar manner to the topographic features. Measurements for partial slope and stream width across features were then summarized and compared to changes in the number of coordinates necessary to represent the feature. Table 4.9 lists the results for the generalized topographic features and Table 4.10 lists the results for the generalized stream banks.

The relative complexity of the features can be ascertained by the degree of change in the percent of original coordinates. Those features which have a large jump in the percent of coordinates from the original representation to the first simplification have a large amount of superfluous points which do not contribute to the unique angular character of the lines as recognized by the Douglas algorithm. As can be seen from the tables, many of the coordinates in these features can be eliminated without having a substantial effect on the measurements of slope and stream width. However, high levels of simplification appear to have more erratic results upon less complex features using the Douglas algorithm.

From an examination of Tables 4.9 and 4.10, it is apparent that some deviation in measurements of slope and stream width occurred over simplification. All of the generalized features show the same pattern of measurement error as increasing with increasing generalization. Deviations appear minimal in the first three simplifications, and generally increase with progressing simplifications. However, several of the features showed greater deviation in the mid-ranges of simplification which decreased in the later simplifications. This occurred for Sallings Mtn., Star Mtn., and Wind River Plateau for the topographic features (Table 4.9), and for Connecticut River, and Shenandoah River stream features (Table 4.10). Figures 4.19, 4.20, and 4.21 illustrate this for the topographic features while Figures 4.22 and 4.23 display this phenomenon for the stream features.

Table 4.9. Measurements on contoured topographic features. Average partial slope is summarized across measurement lines by simplification.

Feature	Simplification	Percent of orig. coords	Average slope (%)	Deviation
Duncan Gap				
	0	100.0	23.88	0.0
	1	34.9	24.01	+0.13
	2	21.1	23.87	-0.01
	3	13.9	23.93	+0.05
	4	9.4	22.88	-1.0
	5	6.1	22.18	-1.7
Sallings Mtn.				
	0	100.0	32.14	0.0
	1	53.4	32.14	0.0
	2	36.6	31.96	-0.18
	3	24.2	31.82	-0.32
	4	16.0	31.12	-1.02
	5	9.5	31.9	-0.24
Miller Mtn.				
	0	100.0	33.24	0.0
	1	55.7	33.48	+0.24
	2	37.8	33.42	+0.18
	3	25.7	33.26	+0.02
	4	16.7	33.90	+0.66
	5	10.9	34.38	+1.14
Star Mtn.				
	0	100.0	18.63	0.0
	1	44.8	18.34	-0.29
	2	32.6	18.38	-0.33
	3	22.4	17.76	-0.87
	4	15.4	17.94	-0.69
	5	10.5	18.92	-0.29
Wind R. Plateau				
	0	100.0	5.29	0.0
	1	72.8	5.25	-0.04
	2	58.8	5.11	-0.18
	3	40.9	5.02	-0.27
	4	26.1	5.13	-0.12
	5	15.6	5.15	-0.14

Table 4.10. Measurements on paired stream lines. Average width is summarized across measurement lines by simplification.

Feature	Simplification	Percent of orig. coords.	Average width(m)	Deviation (m)	Average ang. (deg.)
Connecticut R.					
	0	100.0	285.29	0.0	3.1
	1	20.7	286.51	+1.22	6.4
	2	11.7	286.51	+1.22	8.1
	3	6.8	284.07	-1.22	11.1
	4	4.9	285.29	0.0	13.6
	5	3.2	287.73	+2.44	16.6
Arkansas R.					
	0	100.0	292.61	0.0	11.3
	1	42.1	292.61	0.0	22.4
	2	29.9	293.83	+1.22	28.7
	3	20.4	293.83	+1.22	37.0
	4	13.1	291.39	-1.22	45.8
	5	7.2	285.29	-7.32	57.9
Shenandoah R.					
	0	100.0	59.13	0.0	4.0
	1	28.2	58.52	-0.61	8.2
	2	17.6	57.91	-1.22	10.6
	3	10.8	57.3	-1.83	14.1
	4	7.2	58.52	-0.61	19.3
	5	4.7	62.18	+3.05	25.6
Popo Agie R.					
	0	100.0	48.15	0.0	7.1
	1	42.8	48.15	0.0	12.9
	2	28.7	48.15	0.0	16.7
	3	18.5	48.15	0.0	22.3
	4	12.6	48.77	+0.62	29.6
	5	8.8	49.99	+1.84	38.3
Powell R.					
	0	100.0	40.23	0.0	4.6
	1	31.6	39.62	-.61	9.5
	2	20.0	39.01	-1.22	12.4
	3	13.8	39.31	-0.92	15.7
	4	9.3	38.77	-1.46	20.6
	5	6.2	37.79	-2.44	27.1

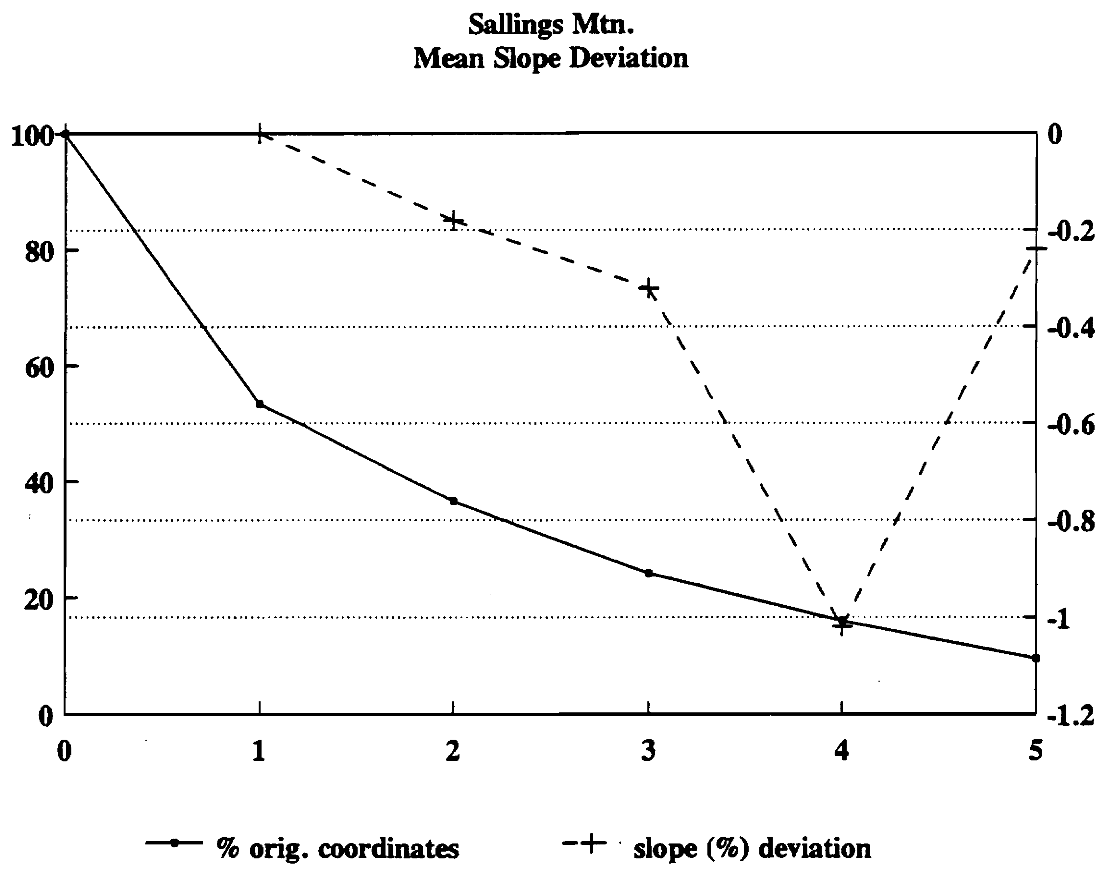


Figure 4.19. Mean slope deviation for Sallings Mtn. topographic feature showing the increase in deviation of slope measurements from the original representation at mid-ranges of simplification which decreases at higher ranges of simplification.

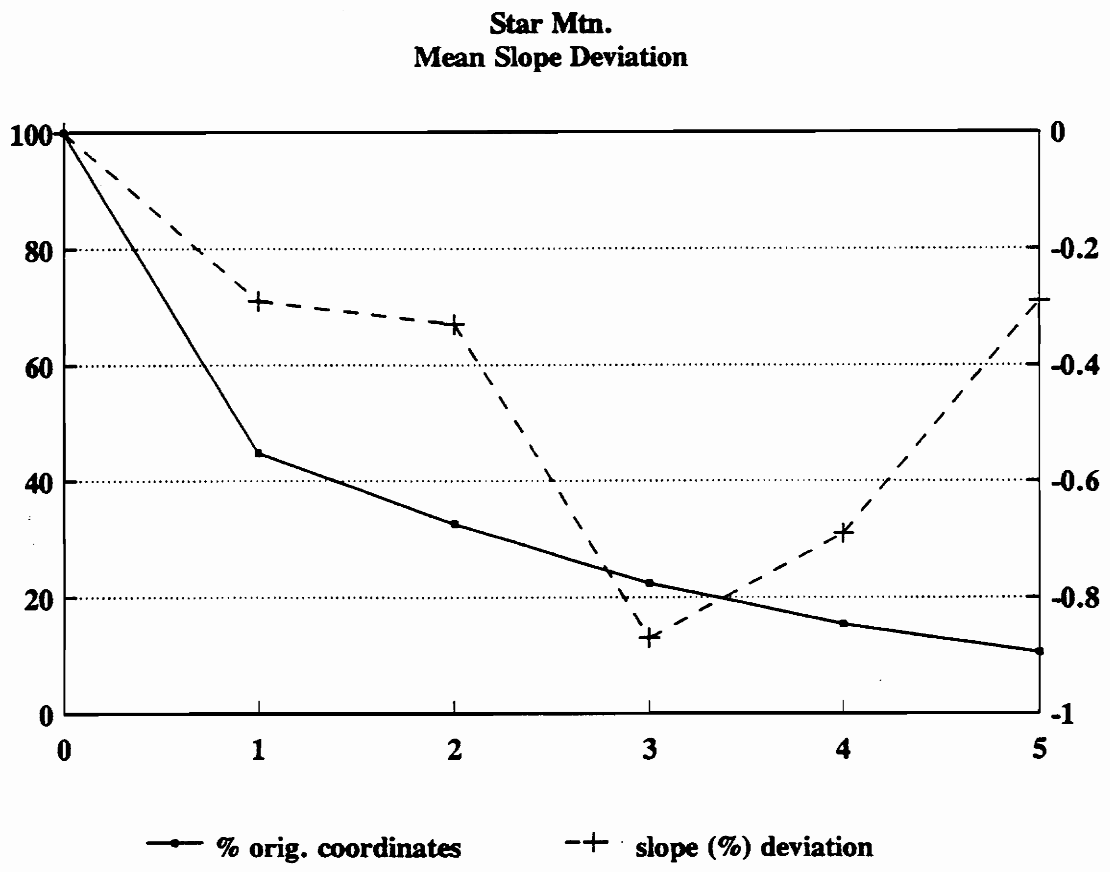


Figure 4.20. Mean slope deviation for Star Mtn. topographic feature showing the increase in deviation of slope measurements from the original representation at mid-ranges of simplification which decreases at higher ranges of simplification.

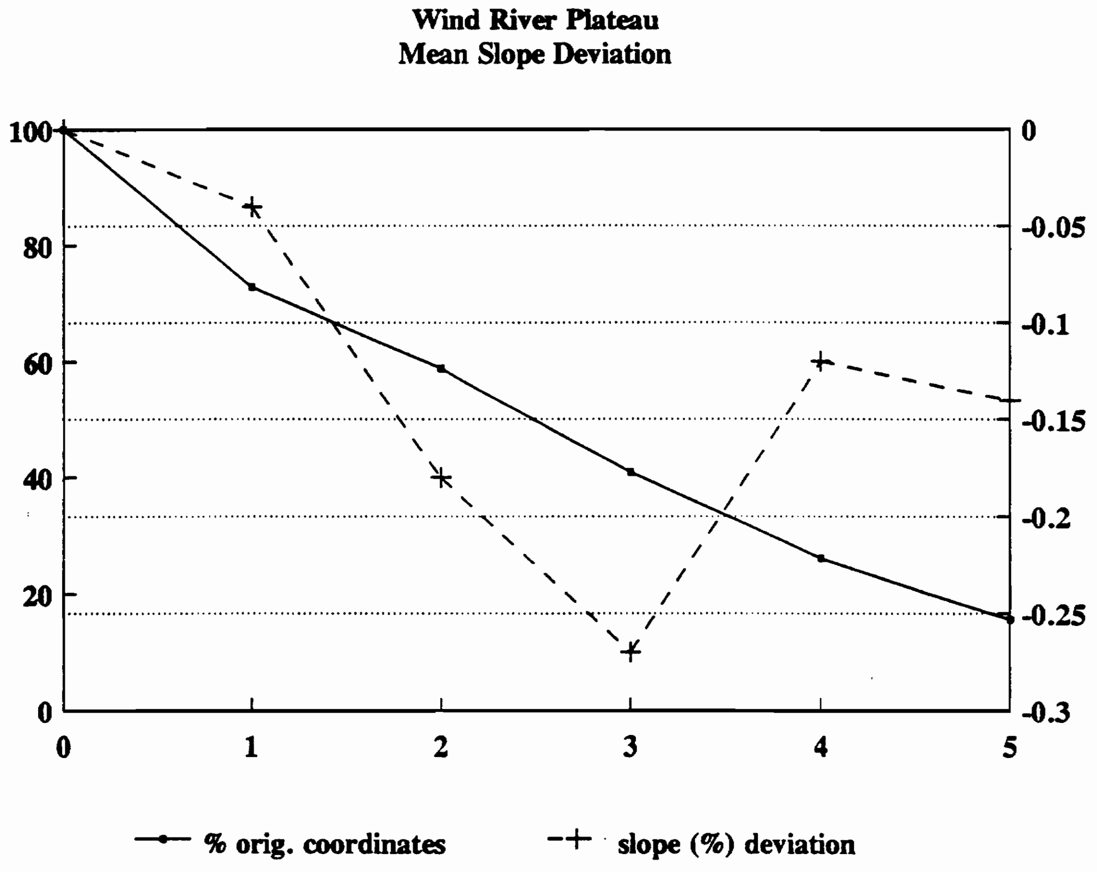


Figure 4.21. Mean slope deviation for Wind River Plateau topographic feature showing the increase in deviation of slope measurements from the original representation at mid-ranges of simplification which decreases at higher ranges of simplification.

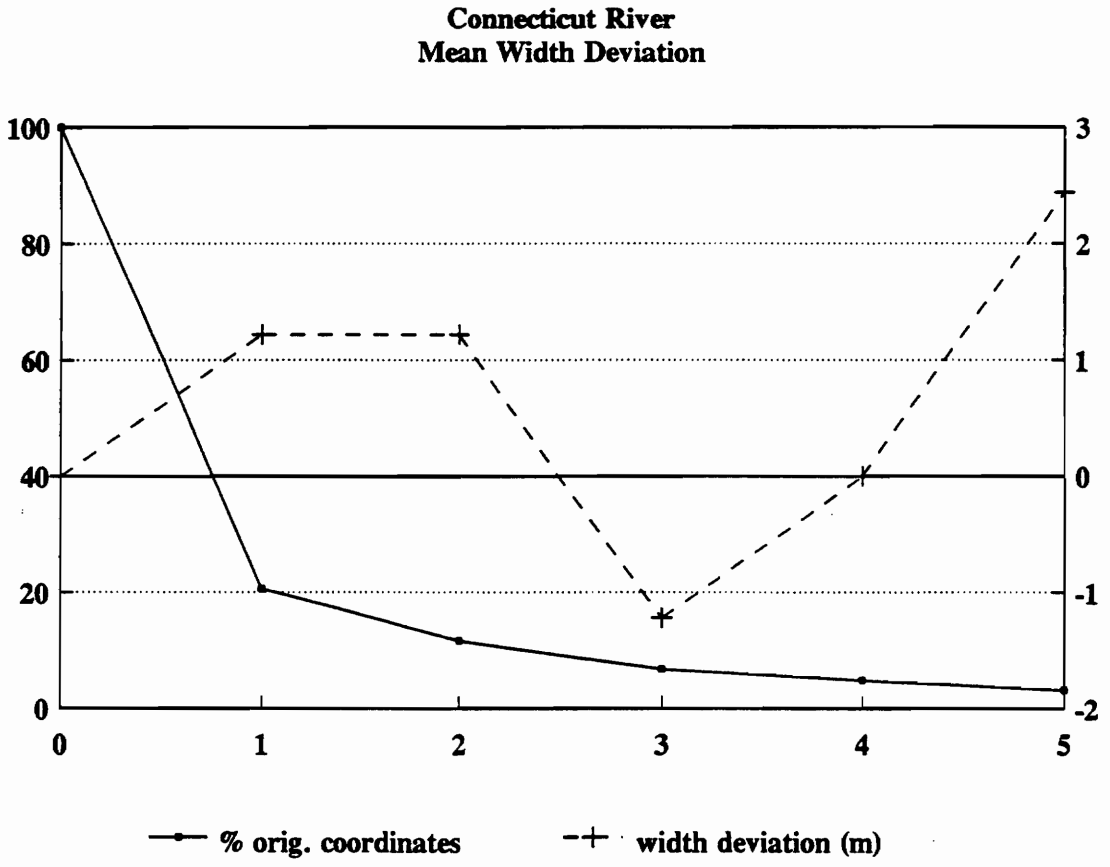


Figure 4.22. Mean stream width deviation for Connecticut River feature showing the increase in deviation of width measurements from the original representation at mid-ranges of simplification which fluctuates at higher ranges of simplification.

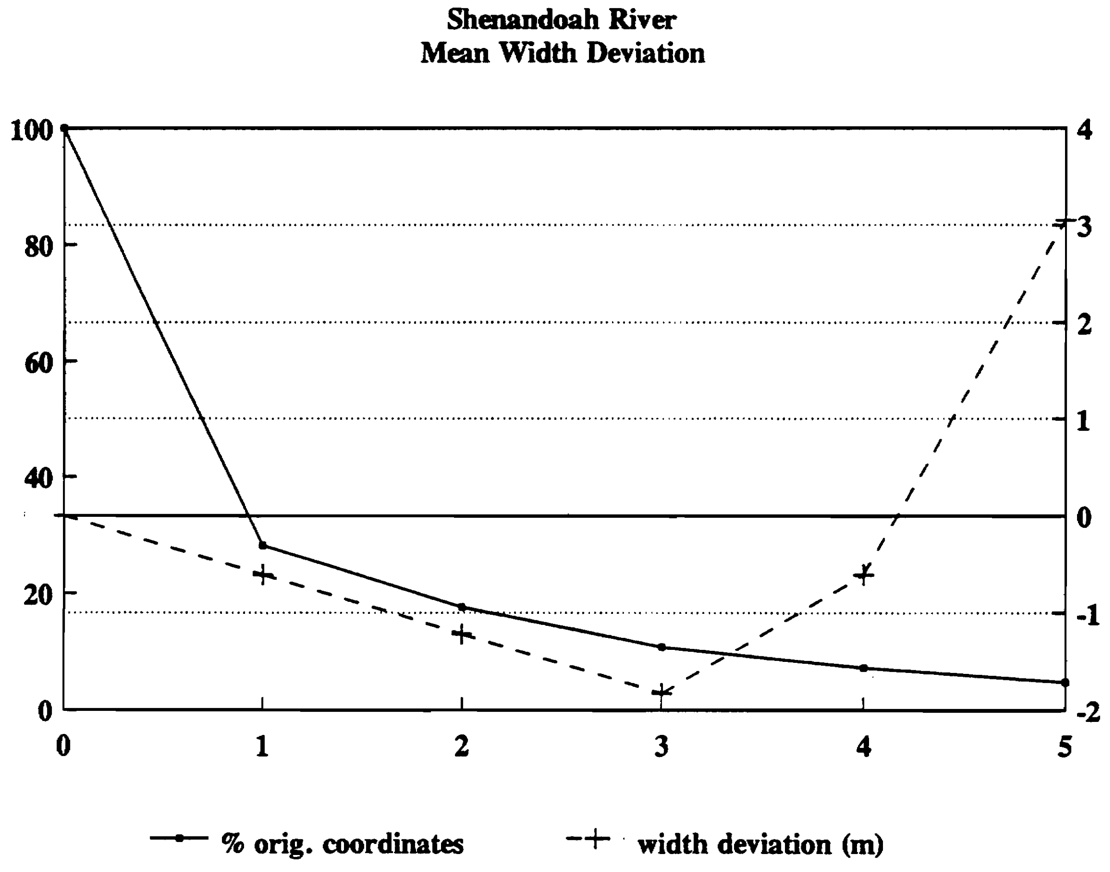


Figure 4.23. Mean stream width deviation for Shenandoah River feature showing the increase in deviation of width measurements from the original representation at mid-ranges of simplification which fluctuates at higher ranges of simplification.

These results seem to indicate that the digital line fluctuates through an axis of displacement as segments in the line are removed with increasing simplifications. This seems not to occur dramatically until after 75 to 90 percent of the coordinates are reduced, suggesting that some 'cusp' of line segments necessary to accurately represent the line has been surpassed. Many of the measurements in the individual data tabulations exhibited this behavior as simplifications of as high as 90 percent maintained low relative distance discrepancies between simplifications despite high angularity increases in the lines. Future research in this area may better establish this suggestion by comparing the deviations in measurement to offset measures and changes in position of each line in relation to the other lines in a feature.

While the results of this study are only preliminary, there appears to be evidence that digital generalization does have an effect upon the representational fidelity of map feature representations. The displacement of lines does appear to affect other lines in the context of the map space. Analyses conducted upon highly simplified digital representations of terrain features may result in errors of several meters on the measurement of distance, or several percent in the measurement of slope. However, substantial reductions in detail can be achieved without a great loss in representational fidelity.

More testing needs to be conducted in this area to firmly establish the trade-offs between data reduction and representational fidelity. Algorithms which have in the past been shown to be inferior to the Douglas algorithm in maintaining positional accuracy of individual lines should be evaluated in terms of their effects upon neighboring lines and feature representations before their relative worth can be assessed. In addition further research may reveal whether the 'cusps' of dramatic change suggested to occur after a large percent of the original coordinates of a line are removed represents a universal phenomenon or simply an artifact of this methodology.

5. Summary and Conclusions

Through the study of the automation of generalization in cartography, much can be learned about the subtleties of digital representation of the physical world. However, as is the case with many exploratory analyses, sometimes more questions arise than answers. The exploratory nature of this analysis is no exception as it has succeeded in raising many interesting questions while providing few firm answers. This study has attempted to uncover the holes in the current level of understanding of the effects of digital generalization and to begin to address these issues.

Much research has been completed in the past on automated generalization and it is clear that much more is needed before a resolution of the problem of generalization of linear map data is realized. However, this study has shown that an appreciation and attention to the representational nature of lines on maps is the next step in this resolution. Sophisticated algorithms for linear generalization can only be appropriately evaluated in terms of their effects on the representational nature of linear map data.

The methods developed thus far for linear generalization provide powerful tools for implementation and evaluation of linear generalization, although their proper application will require further research. This study has suggested that commonly implemented algorithms may be capable of producing simplifications which are significantly better than random generalizations. The Douglas algorithm, widely reported to be the most effective linear simplification routine is also reported here to be the most effective as measured against a random generalizer. Although a more efficient coordinate weeding algorithm such as the Euclidian distance routine may be nearly as effective on simple lines, the manner in which the Douglas algorithm minimizes offsets from the original line makes any additional investment in computing resources worthwhile.

Methods developed in the past for analysis of the structural characteristics of linear features can provide a powerful platform for analysis of automated generalization when collected in a standardized software package. From this platform, it is possible to evaluate the changes in the structural characteristics of a digital line at the level of individual segments. Application of structural complexity measures, such as the fractal dimension can then be evaluated in a manner which provides appropriate guidance for its ability to discriminate levels of simplification. In this study, it is suggested that the fractal dimension, as estimated from the divider walk algorithm, may not be a good measure of automated simplification due to its instability of estimation over ranges of simplification.

Application of structural complexity and linear displacement measures also provide a method to determine the degree to which digital simplifications can be carried out before a digital database must be re-compiled for representation at alternate scales. Although the small size of the line samples used in this study preclude making conclusions as to exactly when this occurs, it appears that after 75 percent of coordinates in a linear feature are reduced the character of a linear feature changes dramatically. Interestingly, this preliminary result coincides well with the perceptual research conducted by Jenks (1989).

The application of the tools for digital analysis of generalization described in this study to changes in the representation of features is also clearly beneficial. From the research conducted here it appears that measurements made on generalized map features appear to be effected by digital simplification. Variations in measurements of width may vary by several meters and those of slope by several percent at extreme simplifications. Further development of similar tools in this regard will greatly enhance the ability to precisely predict the appropriate generalization and representation of digital map features.

Automated generalization has been shown to be a field of inquiry which at our present level of knowledge leaves many questions unanswered. The transformation of a process formerly carried out in the mind of a cartographer into a set of computer instructions has proven to be a difficult task. The resolution of unknowns regarding automated generalization will require much more basic research into the effects of digital representation and manipulation of our conceptions of the physical world. That continued research into this area is warranted is clear as solution of the pressing environmental problems of today will require the sophistication of analysis which the technology of automated mapping and geographic information systems can provide. However, the limits and uncertainty of this technology, such as that faced in automating generalization, must be addressed so that its potential can be fully realized.

Bibliography

Attneave, F. (1954), "Some Informational Aspects of Visual Perception", Psychological Review, Vol. 61, 183-193.

Bertin, J. (1970), "Theory of Communication and Theory of the Graphic", International Yearbook of Cartography, Vol. 18, 118-126.

Boyle, A.R. (1970), "The Quantized Line", The Cartographic Journal, Vol. 7, No. 2, 91-94.

Brassel, K.E. (1985), "Strategies and Data Models for Computer-Aided Generalization", International Yearbook of Cartography, Vol. 25, 11-27.

___________, and R. Weibel (1988), "A Review and Conceptual Framework of Automated Generalization", International Journal of Geographic Information Systems, Vol. 2, No. 3, 229-244.

Buttenfield, B. (1985), "Treatment of the Cartographic Line", Cartographica, Vol. 22, No. 2, 1-26.

___________ (1986), "Digital Definitions of Scale Dependent Line Structure", Proceedings, AUTOCARTO London, Vol. 1, 497-506.

___________ (1987), "Automating the Identification of Cartographic Lines", The American Cartographer, Vol. 14, No.1, 7-20.

___________ (1989), "Scale-Dependence and Self Similarity in Cartographic Lines", Cartographica, Vol. 26, No. 1, 79-100.

Carstensen, L.W. (1989), "A Fractal Analysis of Cartographic Generalization", The American Cartographer, Vol. 16, No. 3, 181-189.

___________, L.W. (1990), "Angularity and Capture of the Cartographic Line During Digital Data Entry",Cartography and Geographic Information Systems, Volume 17, No. 3, 209-224.

Clayton, V. H. (1985), "Cartographic Generalization: A Review of Feature Simplification and Systematic Point Elimination Algorithms", NOAA Technical Report NOS 112, Rockville, Md., 26 pp.

Dent, B. D. (1972), "A Note on the Importance of Shape in Cartogram Communication", The Journal of Geography, Vol. 71, No. 7, 393-401.

Douglas, D. H., and T. K. Peucker (1973), "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature", The Canadian Cartographer, Vol 10, No. 2, 112-122.

Eckert, M. (1908), "On the Nature of Maps and Map Logic", Bulletin, Vol.40, 344-351, reprinted in Cartographica, Monograph No. 19, Vol. 14, 1977, 1-7.

Freeman, H. (1978), "Shape Description Via the use of Critical Points", Pattern Recognition, Vol. 10, No. 3, 159-166.

Goodchild, M.F. (1980a), "The Effects of Generalization in Geographical Data Encoding", in Map Data Processing, (H. Freeman and G. G. Peroni, eds.), Academic Press, New York, 191-204.

___________,(1980b),"Fractals and the Accuracy of Geographical Measures", Mathematical Geology, Vol. 12, No. 2, 85-98.

___________, and D. M. Mark (1987), "The Fractal Nature of Geographic Phenomena", Annals of the Association of American Geographers, Vol. 77, No. 2, 265-278.

Harley, J.B. (1989), "Deconstructing the Map", Cartographica, Vol.26, No.2, 1-20.

Jasinski, M. J. (1990), "Comparison of Complexity Measures for Cartographic Lines", NCGIA Technical Report 90 - 1, 73 pp.

Jenks, G.F. (1981), "Lines, Computers, and Human Frailties", Annals of the Association of American Geographers, Vol. 71, No. 1, 1-10.

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

___________(1989), "Geographic Logic in Line Generalization", Cartographica, Vol. 26, No. 1, 27-42.

Johannsen, T.M. (1974). "A Program for Editing and for Some Generalizing Operations", In Automation the New Trend in Cartography, E. Csati, ed., 131-138, Budapest, Hungary: The Geocartotechnical Research Department.

Keates, J. S. (1972), "Symbols and Meaning in Topographic Maps", International Yearbook of Cartography, Vol. 12, 168-181.

Kretschmer, I. (1978), "The Pressing Problems of Theoretical Cartography", International Yearbook of Cartography, Vol. 18, 33-39.

Lang, T. (1969), "Rules for Robot Draughtsman", Geographical Magazine, XLII (1), 50-51.

Longley, P. A., and M. Batty (1989a), "On the Fractal Measurement of Geographic Boundaries", Geographical Analysis, Vol. 21, No. 1, 47-67.

___________, and M. Batty (1989b), "Fractal Measurement and Line Generalization", Computers and Geosciences, Vol. 15, No. 2, 167-183.

Mandlebrot, B.B. (1982), The Fractal Geometry of Nature, San Francisco, W.H. Freeman Co.

Marino, J.S. (1978), "Characteristic Points and Their Significance in Cartographic Line Generalization", Master's Thesis, Lawrence, KS: University of Kansas.

___________(1979), "Identification of Characteristic Points Along Naturally Occurring Lines: An Empirical Study", The Canadian Cartographer, Vol. 16, pp. 70-80.

Mark, D.M. (1989), "Conceptual Basis for Geographic Line Generalization", Proceedings, 9th International Symposium on Computer Assisted Cartography (AUTOCARTO 9), Falls Church, Va., 68-77.

McMaster, R.B. (1983), "A Mathematical Evaluation of Simplification Algorithms.", In Proceedings of the Sixth International Conference on Automated Cartography (AUTOCARTO6), pp. 267-76.

McMaster, R.B. (1986), "A Statistical Analysis of Mathematical Measures for Linear Simplification", The American Cartographer, Vol. 13, No. 2, 103-116.

___________ (1987a), "The Geometric Properties of Numerical Generalization", Geographical Analysis, Vol. 19, No. 4, 330-346.

___________ (1987b), "Automated Line Generalization", Cartographica, Vol. 24, No. 2, Monograph 37, 74-111.

___________, and K.S. Shea (1988), "Cartographic Generalization in a Digital Environment: A Framework for Implementation in a Geographic Information System", Proceedings, GIS/LIS '88, Vol. 1, 240-249.

___________ (1989), "The Integration of Simplification and Smoothing Algorithms in Line Generalization", Cartographica, Vol. 26, No. 1, 1989, 101-121.

Meine, K.H. (1974), "Cartographic Communication Links and a Cartographic Alphabet", reprinted in Cartographica, Monograph No. 19, Vol.14, 1977, 72-91.

___________(1978), "Certain Aspects of Cartographic Communication in a System of Cartography as a Science", International Yearbook of Cartography, Vol. 18, 102-115.

Monmonier, M. (1986), "Toward a Practicable Model of Cartographic Generalization", Proceedings, AUTOCARTO London, Vol. 2, 257-266.

___________ (1987), "Displacement in Vector and Raster Mode Graphics", Cartographica, Vol. 24, No. 4, 25-36.

Muehrcke, P. C. (1986), Map Use: Reading, Analysis, Interpretation, 2nd Edition,, JP Publications, Madison, WI.

___________ (1990), "Cartography and Geographic Information Systems", Cartography and Geographic Information Systems, Vol. 17, No. 1, 7-15.

Muller, J.C. (1986), "Fractal Dimension and Inconsistencies in Cartographic Line Representations", The Cartographic Journal, Vol. 23, 123-130.

___________ (1987a), "Fractal and Automated Line Generalization", The Cartographic Journal, Vol. 24, 27-34.

___________ (1987b), "The Concept of Error in Cartography", Cartographica, Vol.24, No.3, 1-15.

___________(1989), "Theoretical Considerations for Automated Map Generalization", ITC Journal, Vol. 3/4, 200-204.

Nyerges, T.L.(1991), "Analytical Map Use", Cartography and Geographic Information Systems, Vol. 18, No. 1, pp. 11-22.

Opheim, H. (1982), "Fast Data Reduction of a Digitized Curve", Geo-Processing, Vol. 2, 33-40.

Ratajski, L. (1978), "The Main Characteristics of Cartographic Communication as a Part of Theoretical Cartography", International Yearbook of Cartography, Vol. 18, 21-32.

Richardson, L. F. (1961), "The Problem of Contiguity: An Appendix to the Statistics of Deadly Quarrels", General Systems Yearbook, Vol. 6, 139-187.

Roberge, J. (1985), "A Data Reduction Algorithm for Planar Curves", Computer Vision, Graphics, and Image Processing, Vol. 29, 168-195.

Robinson, A.H., and B.B. Petchenik (1975), "The Map as a Communication System", The Cartographic Journal, Vol. 12, 7-15.

___________, R.D. Sale, J.L. Morrison, and P.C. Muercke, Elements of Cartography, Wiley & Sons, New York, 1984, 544 pp.

Shannon, C.E. and W. Weaver, The Mathematical Theory of Communication, University of Illinois Press, Urbana, 1949.

Scheidegger, Adrian E., Theoretical Geomorphology, Springer-Verlag, Berlin, 1970.

Shea, S.K., and R.B. McMaster (1989), "Cartographic Generalization in a Digital Environment: When and How to Generalize", Proceedings, 9th International Symposium on Computer Assisted Cartography (AUTOCARTO 9), Falls Church, Va., 56-67.

Shelberg, M.C., H. Moellering, and N. Lam (1982), "Measuring the Fractal Dimension of Empirical Cartographic Curves", Proceedings, AUTOCARTO 5, 481-490.

Sinton, D.F., (1978), "The Inherent Structure of Information as a Constraint to Analysis: Mapped Thematic Data as a Case Study", In Harvard Papers on Geographic Information Systems: First International Symposium on Topological Data Structures for Geographic Information Systems, Vol. 7 (ed. G. Dutton), Cambridge, MA: Laboratory for Computer Graphics and Spatial Analysis.

Srnka, E. (1970), "The Analytical Solution of Regular Generalisation in Cartography", International Yearbook of Cartography, Vol. X, 47-60.

Steward, H.J., Cartographic Generalisation, Some Concepts and Explanation, Supplement No.1 to Canadian Cartographer, Vol. 11, 1974, University of Toronto Press, Toronto, Canada, 77 pp.

Sukhov, V. I. (1970), "Application of Information Theory in Generalisation of Map Contents", International Yearbook of Cartography, Vol. 10, 41-46.

Topfer, F. and W. Pillewizer (1966), "The Principles of Selection", The Cartographic Journal, Vol. 3, 10-16.

Unwin, D. (1981), Introductory Spatial Analysis, Methuen, New York.

Weibel, R. (1990), "Amplified Intelligence and Knowledge Base Systems for Map Generalization", Paper presented at the annual meeting of the Association of American Geographers, Toronto, Canada.

White, E. R. (1983), Perceptual Evaluation of Line Generalization Algorithms, Department of Geography, Master's Thesis, Virginia Polytechnic Institute and State University, 133 pages.

__________ (1985), "Assessment of Line-Generalization Algorithms Using Characteristic Points", The American Cartographer, Vol. 12, No. 1, 17-27.

Wright, J.K. (1944), "Map Makers are Human: Comments on the Subjective in Maps", Geographical Review, Vol. 32, 527-544, reprinted in Cartographica, Monograph No. 19, Vol. 14, 1977, 8-25.

ZYCOR, INC. (1984), Manual and Automated Feature Displacement, Report for the U. S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia, unclassified material, 2 Volumes, 204 pp.

Appendix - Program Source Code Listings

A.1. LINGEN.C

```
/*******************************************************************
 Turbo C Program:  LINGEN.C
                  Line Generalization program

                  Written by John A. Young
                  Graduate Student
                  Department of Geography
                  Virginia Polytechnic Institute
                  Blacksburg, Virginia  24061

  This program contains a set of line generalization algorithms.
  The user can select a generalization technique, input a file of
  lines in MOSS format, and test the effectiveness of
  the technique against the original line or other algorithms. Error
  measures are calculated on the differences between the original and
  simplified lines. Some simplification algorithms are after White (1983).

  This version is written for the HALO graphics program and the
  NUMBER9 graphics card (links to HALODVXX.OBJ, HALONINE.DEV
  and HALOTL.LIB drivers and libraries).
*******************************************************************/

/* Libraries */
#include <stdio.h>
#include <io.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>
#define pi 3.1415926
#define rad 57.29578

/* Function Definitions */
void npoint();
void distance();
void angmeas();
void perpmeas();
void dpuke();
double areaoff(float *xs2,float *ys2);
float length(float *x,float *y,int nopts);
float angle(float *x,float *y,int nopts);
void  err_meas(int algo,float tol,float tused);
void  plot1();
void  plot2();
void  plot3();
```

```
/* Global variables and Files */
int  mode,itype,gcode,npts,pts;
float xmin,ymin,xmax,ymax,txmin,tymin,txmax,tymax;
float *xt,*yt,*xs,*ys;
char device[]="halonine.dev";
FILE *filea;
FILE *fileb;
FILE *filec;

main()
{
int   choice,i,gc;
float x,y;
char  ifil[20];
char  errfil[20];

clrscr();
printf("                    Welcome to the\n");
printf("      Line Generalization and Analysis System.\n\n");
printf("\n\n This program will test the effectiveness of several");
printf("\n line generalization algorithms as compared to characteristics");
printf("\n of the original and simplified lines.");

/** open error measures file **/
printf("\n\nEnter name of file to store error measures:  ");
scanf("%s",errfil);
if ((filec=fopen(errfil,"w"))== NULL)
  {
   printf("error opening measures file!"); exit(0);
  }

file:printf("\n\nEnter name of input file:  ");
scanf("%s",ifil);
if ((filea=fopen(ifil,"r"))== NULL)
      {
  printf("error opening file\n"); exit(0);
}

line:printf("\nEnter the geocode of the line to process: ");
scanf("%d",&gc);

xmin=99999999.9;
ymin=99999999.9;
xmax=0.0;
ymax=0.0;
npts=0.0;
jmp:
fscanf(filea,"%1d%5d%5d\n",&itype,&gcode,&npts);
printf("\nProcessing geocode: %d    type: %d    points:%d\n",gcode,itype,npts);
```

```
if(gcode==gc)
{

/****** Memory allocation for lines ******************/
  xt=(float *) calloc(npts,sizeof(float));
  if(xt==NULL) {
    printf("Error in memory allocation for x value");
    exit(0);
  }
  yt=(float *) calloc(npts,sizeof(float));
  if(yt==NULL) {
    printf("Error in memory allocation for y value");
    exit(0);
  }
  xs=(float *) calloc(npts,sizeof(float));
  if(xs==NULL) {
    printf("Error in memory allocation for x value");
    exit(0);
  }
  ys=(float *) calloc(npts,sizeof(float));
  if(ys==NULL) {
    printf("Error in memory allocation for y value");
    exit(0);
  }

  /**************************************************/

  for(i=0;i<npts;i++)
{
  fscanf(filea,"%f %f\n",(xt+i),(yt+i));
  if(*(xt+i) < xmin) xmin=*(xt+i);
  if(*(yt+i) < ymin) ymin=*(yt+i);
  if(*(xt+i) > xmax) xmax=*(xt+i);
  if(*(yt+i) > ymax) ymax=*(yt+i);
}
}
else
{
for (i=0;i<npts;i++)
  {
  fscanf(filea,"%f %f\n",&x,&y);
  }
if(!feof(filea)){
 goto jmp;
}
else
{
printf("** Geocode not found  **");
rewind(filea);
goto line;
```

```
}
}

/**************** begin main menu *****************************************/

do {
do {
   clrscr();
   printf("\n\n\nLine Generalization Algorithms:\n");
   printf("\n     1: Nth Point Elimination\n");
   printf("     2: Euclidean Distance\n");
   printf("     3: Angular Measure\n");
   printf("     4: Perpendicular Measure\n");
   printf("     5: Douglas-Poiker\n");
   printf("     6: Get New Line\n");
   printf("     7: Get New File\n");
   printf("     8: Quit\n");
                printf("\n\n          Enter your choice:  ");
                scanf("%d",&choice);
} while(choice < 1 || choice > 8);

switch(choice) {

case 1:
  clrscr();
  printf("\n\nNth Point Elimination");
  npoint();
                        break;

case 2:
  clrscr();
  printf("\n\nEuclidean Distance");
  distance();
  break;

case 3:
  clrscr();
  printf("\n\nAngular Measure");
  angmeas();
  break;

case 4:
  clrscr();
  printf("\n\nPerpendicular Measure");
  perpmeas();
  break;

case 5:
  clrscr();
  printf("\n\nDouglas-Peucker Algorithm");
  dpuke();
  break;

case 6:
  clrscr();
  rewind(filea);
  free(xt);
  free(yt);
  free(xs);
  free(ys);
  goto line;

case 7:
  clrscr();
  fclose(filea);
  free(xt);
  free(yt);
  free(xs);
  free(ys);
  goto file;

}
printf("\n");
} while(choice!=8);
fclose(filea);
fclose(filec);
        free(xt);
free(yt);
free(xs);
free(ys);
}

/***************************************************************************/
/************ Nth point elimination algorithm ***************************/
/***************************************************************************/
   void npoint()
        {
int   i,j,count,save,algo;
time_t start,stop;
double tused,nth;

algo=1;
printf("\n\nThis algorithm will retain every Nth point in a line.");
printf("\n\nEnter a value for N: ");
scanf("%d",&save);
start=time(NULL);

pts=0;
count=0;
for(j=0; j<npts; j++)
```

```
{
   if(j==0){
*(xs+pts)=*(xt+j);
*(ys+pts)=*(yt+j);
pts=pts+1;
goto jump;
}
   if(j==npts-1){
*(xs+pts)=*(xt+j);
*(ys+pts)=*(yt+j);
pts=pts+1;
goto jump;
}
   count=count+1;
jump:
   if (count==save)
{
*(xs+pts)=*(xt+j);
*(ys+pts)=*(yt+j);
count=0;
pts=pts+1;
}
 }
nth=save;
stop=time(NULL);
tused = difftime(stop, start);
err_meas(algo,nth,tused);
        }
/**************************************************************************/
/************** Euclidean Distance Algorithm ***************************/
/**************************************************************************/
  void distance()
        {
int   i,count,algo;
float dist;
float len,lensum,x1,x2,y1,y2,acalc,bcalc;
        time_t start,stop;
double tused,tmp;

algo=2;
printf("\n\nThis algorithm will delete points in a line which are less");
printf("\nthan a preset distance tolerance from the previous point.");
printf("\n\nEnter a value for distance: ");
scanf("%f",&dist);
start=time(NULL);

/* Read first two points*/
pts=0;
x1=*(xt+pts);
y1=*(yt+pts);
```

```
for(count=1;count<npts;count++)
{
 x2=*(xt+count);
 y2=*(yt+count);
if(count==npts-1) goto jmp;
if(count==1) {
*(xs+pts)=x1;
*(ys+pts)=y1;
pts=1;
}

  /******* compute length **********/
acalc=y2-y1;
if(x2-x1==0) x2=x2+0.000001;
bcalc=x2-x1;
tmp=((acalc*acalc)+(bcalc*bcalc));
len=sqrt((double)tmp);

if (len>dist)
{
*(xs+pts)=x2;
*(ys+pts)=y2;
                pts=pts+1;
 x1=x2;
 y1=y2;
}
}
/************** save last point ***************/
jmp:    *(xs+pts)=x2;
*(ys+pts)=y2;
pts=pts+1;

        stop=time(NULL);
tused = difftime(stop, start);
err_meas(algo,dist,tused);
}
/*************************************************************************/
/************** Angular measure algorithm ********************************/
/*************************************************************************/
   void angmeas()
        {
int   i,point,algo;
float degree,maxa,angle,theta;
float x1,x2,x3,y1,y2,y3,a1,a2,a3,t1,t2,t3,t4;
        time_t start,stop;
double tused;

algo=3;
printf("\n\nThis algorithm eliminates points from a line by using");
printf("\nsequential triads of points. The second point in series");
printf("\nis eliminated if the angle between the first and third");
printf("\npoints is greater than the user defined maximum.");
printf("\n\nEnter maximum angle (degrees) between line segments: ");
scanf("%f",&degree);
start=time(NULL);

maxa=degree*(pi/180);
point=2;
pts=0;

/* Read and store first point */
x1=*(xt+pts);
y1=*(yt+pts);
*(xs+pts)=x1;
*(ys+pts)=y1;
pts=pts+1;

x2=*(xt+pts);
y2=*(yt+pts);

while(point<npts)
{
  x3=*(xt+point);
  y3=*(yt+point);
  point=point+1;

/*compute angle*/
if(x2-x1==0) x2=x2+0.000001;
        t1=y2-y1;
        t2=x2-x1;
a1=atan2(t1,t2);
if(x3-x2==0) x3=x3+0.000001;
        t3=y3-y2;
        t4=x3-x2;
a2=atan2(t3,t4);
if(x3-x2==0) x3=x3+0.000001;
a3=a1-a2;
angle=(pi*a3)-a3;
theta=fabs(angle);

   if (theta<maxa)
   {
x2=x3;
y2=y3;
   }
   else
   {
*(xs+pts)=x2;
*(ys+pts)=y2;
pts=pts+1;
x1=x2;
y1=y2;
x2=x3;
y2=y3;
   }
}
 *(xs+pts)=x2;
 *(ys+pts)=y2;
 pts=pts+1;

stop=time(NULL);
tused = difftime(stop, start);
err_meas(algo,degree,tused);
}
/**************************************************************************/
/************* Perpendicular distance algorithm *************************/
/**************************************************************************/
   void perpmeas()
{
int   i,point,algo;
float x1,x2,x3,x4,y1,y2,y3,y4,acalc,bcalc,ccalc,dcalc;
float ecalc,fcalc,dist,tol,temp,t1,t2;
        time_t start,stop;
double tused;

algo=4;
printf("\n\nThis algorithm eliminates points from a line by using");
printf("\nsuccessive triads of points (A,B,C). Point B is ");
printf("\neliminated if the perpendicular distance from B to the ");
printf("\nsegment AC is less than the user-defined maximum.");
printf("\n\nEnter a value for the minimum perpendicular distance:  ");
scanf("%f",&tol);
start=time(NULL);
        point=2;
pts=0;


/* Read first three points */
x1=*(xt+pts);
y1=*(yt+pts);
*(xs+pts)=x1;
*(ys+pts)=y1;
pts=1;

x2=*(xt+pts);
y2=*(yt+pts);

while(point<npts)
{
  x3=*(xt+point);
  y3=*(yt+point);
  point=point+1;


/* Compute Perpendicular Distance */
acalc=y1-y3;
bcalc=x3-x1;
ccalc=y3-y1;
dcalc=((y1*x3)-(y3*x1));
ecalc=((ccalc*y2)+(bcalc*x2));
fcalc=((acalc*ccalc)-(bcalc*bcalc));

if(fcalc==0) fcalc=0.000001;
x4=((ccalc*dcalc)-(bcalc*ecalc))/fcalc;
y4=((acalc*ecalc)-(bcalc*dcalc))/fcalc;
       t1=x2-x4;
       t2=y2-y4;
temp=((t1*t1)+(t2*t2));
dist=sqrt((double)temp);

  if(dist < tol)
  {
x2=x3;
y2=y3;
  }
  else
  {
  *(xs+pts)=x2;
  *(ys+pts)=y2;
  pts=pts+1;
  x1=x2;
  y1=y2;
  x2=x3;
  y2=y3;
  }
}
*(xs+pts)=x2;
*(ys+pts)=y2;
pts=pts+1;

       stop=time(NULL);
tused = difftime(stop, start);
err_meas(algo,tol,tused);
       }
/***************************************************************************/
/************ Douglas-Poiker algorithm ***********************************/
/***************************************************************************/
   void dpuke()
{
```

```
float xcoor,ycoor,xfirst,yfirst,xsec,ysec,xthird,ythird;
float xfour,yfour,xlast,ylast,xtemp,ytemp,xtemp2,ytemp2;
float acalc,bcalc,ccalc,dcalc,ecalc,fcalc,tol,t1,t2;
double dist2,dist1;
int   iend,jloop,lfst,llast,jsave,ll,lf;
int   istack[1000],i,n,algo;
time_t start,stop;
double tused;

algo=5;
printf("\n\nThis algorithm reduces the number of data points in a");
printf("\nline by measuring the maximum deviation from a straight");
printf("\nline connecting the first and last points, and subsequently,");
printf("\nthe Nth to Nth point. Algorithm originally written by ");
printf("\nDavid Douglas based on a concept by Thomas Peucker");
printf("\n\nEnter a value for the band tolerance:  ");
scanf("%f",&tol);
start=time(NULL);
pts=0;
n=0;

      /********        Write first coordinate on file    *********/
*(xs+pts)=*(xt+pts);
*(ys+pts)=*(yt+pts);
pts=pts+1;

/********       Assignment of variables          *********/
iend=npts-1;
lfst=0;
llast=npts-1;
dist2=0;

/******** Begin calculation loop           *********/
lf=lfst+1;
ll=llast-1;

 loopa:
for(jloop=lf;jloop<=ll;jloop++)
{

/********      check for last coordinate pair      *********/
if(lf>ll) goto skip;

/**  assignment of x-y coordinate pairs to variables *****/
/**for perpendicular calculation in function       *****/

xfirst=*(xt+lfst);
yfirst=*(yt+lfst);
xthird=*(xt+llast);
ythird=*(yt+llast);
xsec=*(xt+jloop);
ysec=*(yt+jloop);

/**  Determine if the distance is greater than the tolerance**/
/**by computing a perpendicular bisector between a line    **/
/**(connecting points 1 and 3) and point 2.        **/

acalc=yfirst-ythird;
bcalc=xthird-xfirst;
ccalc=ythird-yfirst;
dcalc=((yfirst*xthird)-(ythird*xfirst));
ecalc=((ccalc*ysec)+(bcalc*xsec));
fcalc=((acalc*ccalc)-(bcalc*bcalc));
if(fcalc==0) fcalc=0.00001;
xfour=((ccalc*dcalc)-(bcalc*ecalc))/fcalc;
yfour=((acalc*ecalc)-(bcalc*dcalc))/fcalc;
        t1=xsec-xfour;
        t2=ysec-yfour;
dist1=((t1*t1)+(t2*t2));
dist1=sqrt((double)dist1);

/**  check for distances (dist2 > dist1)                    **/

if(dist2>dist1) goto jump2;
        dist2=dist1;
jsave=jloop;
jump2:  continue;
}

if(dist2<tol) goto skip;
llast=jsave;
ll=llast-1;
dist2=0;
n=n+1;
istack[n]=jsave;

goto loopa;

 skip:
*(xs+pts)=*(xt+llast);
*(ys+pts)=*(yt+llast);
pts=pts+1;
lfst=llast;
lf=lfst+1;
n=n-1;
if(n==0) llast=iend;
ll=llast-1;
if(n==0) goto jump;
llast=istack[n];
ll=llast-1;
```

```
jump:   if(lfst==iend) goto end;
dist2=0;

goto loopa;

end:
stop=time(NULL);
tused = difftime(stop, start);
err_meas(algo,tol,tused);
}

/*************************************************************************/
/***** Procedure to plot original line in a window 1/2 size of screen ****/
/*************************************************************************/
void plot1()
{
int i,color,border,back,zero=0,one=1,two=2,three=63;
float xrange,yrange,xscale,yscale;
        float x1l,y1l,x1r,y1r,xmid,yup;

setdev(device);
mode=0;
initgraphics(&mode);
setcolor(&mode);

        x1l=0.01;
x1r=0.495;
y1l=0.01;
y1r=0.76;
border=7;
back=0;
color=200;

txmin=xmin;
txmax=xmax;
tymin=ymin;
tymax=ymax;
xrange=txmax-txmin;
yrange=tymax-tymin;
xscale=xrange/512.0;
yscale=yrange/484.0;
if(xscale > yscale) tymax=tymin+(yrange*(xscale/yscale));
if(yscale > xscale) txmax=txmin+(xrange*(yscale/xscale));
txmin=txmin-(0.05*(txmax-txmin));
tymin=tymin-(0.05*(tymax-tymin));
txmax=txmax+(0.05*(txmax-txmin));
tymax=tymax+(0.05*(tymax-tymin));
setcolor(&color);
setviewport(&x1l,&y1l,&x1r,&y1r,&border,&back);
setworld(&txmin,&tymin,&txmax,&tymax);
```

```
inittcur(&two,&two,&mode);
settext(&one,&one,&zero,&one);
settextclr(&three,&zero);


for(i=0;i<npts;i++)
   { if(i==0)
      { movabs((xt+i),(yt+i));
      }
      else
      { lnabs((xt+i),(yt+i));
      }
   }
 color=7;
 setcolor(&color);
 for(i=0;i<npts;i++)
   {
   ptabs((xt+i),(yt+i));
   }
xmid=((txmin+txmax)/2.0)-((txmax-txmin)/7);
yup=tymin+.01;
movtcurabs(&xmid,&yup);
btext("Original Line");
}

/************************************************************************/
/**************** Procedure to plot simplified line *********************/
/************************************************************************/
void plot2()
{
int i,color,border,back,zero=0,one=1,two=2,three=63;
float xmid,yup,x2l,y2l,x2r,y2r;

       x2l=0.505;
x2r=0.99;
y2l=0.01;
y2r=0.76;
border=7;
back=0;
color=200;

setcolor(&color);
setviewport(&x2l,&y2l,&x2r,&y2r,&border,&back);
setworld(&txmin,&tymin,&txmax,&tymax);
inittcur(&two,&two,&mode);
settext(&one,&one,&zero,&one);
settextclr(&three,&zero);

for(i=0;i<pts;i++)
   { if(i==0)
```

```
      { movabs((xs+i),(ys+i));
      }
      else
      { lnabs((xs+i),(ys+i));
      }
   }
 color=7;
 setcolor(&color);
 for(i=0;i<pts;i++)
   {
   ptabs((xs+i),(ys+i));
   }
 xmid=((txmin+txmax)/2.0)-((txmax-txmin)/7);
 yup=tymin+.01;
 movtcurabs(&xmid,&yup);
 btext("Simplification");
}

/*************************************************************************/
/*************** Procedure to overlay two line plots ********************/
/*************************************************************************/
void plot3()
{
int i,color,border,back,zero=0,one=1,two=2,three=63;
float x5l,y5l,x5r,y5r,xmid,yup;
setdev(device);
mode=0;
initgraphics(&mode);
setcolor(&mode);

x5l=0.01;
x5r=0.99;
y5l=0.01;
y5r=0.99;
border=7;
back=0;

setviewport(&x5l,&y5l,&x5r,&y5r,&border,&back);
setworld(&txmin,&tymin,&txmax,&tymax);
inittcur(&two,&two,&mode);
settext(&one,&one,&zero,&one);
settextclr(&three,&zero);

 color=63;
 setcolor(&color);
 for(i=0;i<pts;i++)
   { if(i==0)
      { movabs((xs+i),(ys+i));
      }
      else
```

```
    { lnabs((xs+i),(ys+i));
    }
  }

color=200;
setcolor(&color);
 for(i=0;i<npts;i++)
  { if(i==0)
    { movabs((xt+i),(yt+i));
    }
    else
    { lnabs((xt+i),(yt+i));
    }
  }
      xmid=((txmin+txmax)/2.0)-((txmax-txmin)/7);
 yup=tymin+.01;
 movtcurabs(&xmid,&yup);
 btext("Overlay");
}

/*****************************************************************************/
/******************** Error measures procedure ****************************/
/*****************************************************************************/
void err_meas(int algo,float tol,float tused)
{
int   i,j,digits=6,border,back,zero=0,three=63,four=5;
float len_chng,ang_chng,movtx,movty;
float length1,length2;
float angsum1,angsum2,avgang1,avgang2;
float x3l,y3l,x3r,y3r;
float cor_chng,coor1,coor2;
float area_offs;
char  ch,ofil[20],buffi[5],buffg[20];


clrscr();
length1=0.0;
length2=0.0;
len_chng=0.0;
angsum1=0.0;
angsum2=0.0;
avgang1=0.0;
avgang2=0.0;
ang_chng=0.0;
area_offs=0.0;

plot1();
plot2();

/******* scale and write error measures window *******/
```

```
x3l=0.01;
y3l=0.77;
x3r=0.99;
y3r=0.99;
border=7;
back=0;
setviewport(&x3l,&y3l,&x3r,&y3r,&border,&back);
setworld(&x3l,&y3l,&x3r,&y3r);
        settextclr(&four,&zero);
movtx=.05;
movty=.96;
movtcurabs(&movtx,&movty);
btext("algo:");
movtx=.40;
movtcurabs(&movtx,&movty);
btext("tol:");
movtx=.65;
movtcurabs(&movtx,&movty);
btext("time:");

settextclr(&three,&zero);
movtx=.15;
movtcurabs(&movtx,&movty);
if(algo==1) btext("Nth");
if(algo==2) btext("E.Dist");
if(algo==3) btext("Ang");
if(algo==4) btext("Perp");
if(algo==5) btext("D-P");
movtx=.48;
movtcurabs(&movtx,&movty);
gcvt(tol,digits,buffg);
btext(&buffg);
movtx=.75;
movtcurabs(&movtx,&movty);
gcvt(tused,digits,buffg);
btext(&buffg);

settextclr(&four,&zero);
movtx=.05;
movty=.93;
movtcurabs(&movtx,&movty);
btext("#pts:");
movty=.90;
movtcurabs(&movtx,&movty);
btext("length:");
movty=.87;
movtcurabs(&movtx,&movty);
btext("avg_ang:");
movty=.84;
movtcurabs(&movtx,&movty);
```

```
btext("sum_ang:");
movtx=.55;
movty=.93;
movtcurabs(&movtx,&movty);
btext("#pts:");
movty=.90;
movtcurabs(&movtx,&movty);
btext("length:");
movty=.87;
movtcurabs(&movtx,&movty);
btext("avg_ang:");
movty=.84;
movtcurabs(&movtx,&movty);
btext("sum_ang:");
movtx=.05;
movty=.81;
+movtcurabs(&movtx,&movty);
btext("%chng pts:");
movty=.78;
movtcurabs(&movtx,&movty);
btext("%chng length:");
movtx=.55;
movty=.81;
movtcurabs(&movtx,&movty);
btext("%chng ang:");
movty=.78;
movtcurabs(&movtx,&movty);
btext("area offset:");
settextclr(&three,&zero);


/********* measures for original line ****************/
coor2=pts;
coor1=npts;
length1=length(xt,yt,npts);
angsum1=angle(xt,yt,npts);
avgang1=angsum1/(npts-2);
angsum1=angsum1/length1;

itoa(npts,buffi,10);
movtx=.22;
movty=.93;
movtcurabs(&movtx,&movty);
btext(&buffi);
gcvt(length1,digits,buffg);
movty=.90;
movtcurabs(&movtx,&movty);
btext(&buffg);
gcvt(avgang1,digits,buffg);
movty=.87;
```

```
movtcurabs(&movtx,&movty);
btext(&buffg);
gcvt(angsum1,digits,buffg);
movty=.84;
movtcurabs(&movtx,&movty);
btext(&buffg);


/*********** measures for simplified line ******************/
length2=length(xs,ys,pts);
angsum2=angle(xs,ys,pts);
avgang2=angsum2/(coor2-2);
angsum2=angsum2/length2;

itoa(pts,buffi,10);
movtx=.72;
movty=.93;
movtcurabs(&movtx,&movty);
btext(&buffi);
gcvt(length2,digits,buffg);
movty=.90;
movtcurabs(&movtx,&movty);
btext(&buffg);
gcvt(avgang2,digits,buffg);
movty=.87;
movtcurabs(&movtx,&movty);
btext(&buffg);
gcvt(angsum2,digits,buffg);
movty=.84;
movtcurabs(&movtx,&movty);
btext(&buffg);

/*********** comparison error measures *********************/
cor_chng=(1-(coor2/coor1))*100;
len_chng=(1-(length2/length1))*100;
ang_chng=(1-(avgang1/avgang2))*100;
area_offs=areaoff(xs,ys);
gcvt(cor_chng,digits,buffg);
movtx=.28;
movty=.81;
movtcurabs(&movtx,&movty);
btext(&buffg);
gcvt(len_chng,digits,buffg);
movty=.78;
movtcurabs(&movtx,&movty);
btext(&buffg);
gcvt(ang_chng,digits,buffg);
movtx=.78;
movty=.81;
movtcurabs(&movtx,&movty);
```

```
btext(&buffg);
gcvt(area_offs,digits,buffg);
movty=.78;
movtcurabs(&movtx,&movty);
btext(&buffg);
printf("\n\nDisplay overlay of two lines (y or n)? ");
ch = getche();
if(ch=='y'){
  plot3();
}

/*********** save simplified line to file ******************/
printf("\n\nSave simplified line to file (y or n)? ");
ch = getche();
if(ch=='y')
  {
         printf("\nEnter the output file name:  ");
scanf("%s",ofil);
if ((fileb=fopen(ofil,"a"))== NULL)
   {
   printf("error opening file"); exit(0);
   }
 fprintf(fileb, "%1d%5d%5d\n",itype,gcode,pts);
 for(j=0;j<pts;j++){
 fprintf(fileb, "%f %f\n",*(xs+j),*(ys+j));
 }
  }

printf("\n\nSave error measures to file (y or n)? ");
ch = getche();
if(ch=='y')
 {
 printf("\n\nWriting to error measures file");
  fprintf(filec," %d %.3f %d  %d %.4f %.4f %.4f  %d %.4f %.4f %.4f  %.2f %.2f %.2f %.4f
%.2f\n",algo,tol,gcode,npts,length1,avgang1,angsum1,pts,length2,avgang2,angsum2,cor_chng,le
n_chng,ang_chng,area_offs,tused);
 }
fclose(fileb);
}
/****************************************************************************/
/********** Procedure to calculate length of line ************************/
/****************************************************************************/
 float length(float *x,float *y,int nopts)
{
int i,i2;
float tmp,len,lensum,x1,x2,y1,y2,acalc,bcalc;

lensum=0.0;
for(i=0;i<nopts;i++)
{
```

```
if(i==nopts-1) goto jmp;
x1=*(x+i);
y1=*(y+i);
i2=i+1;
x2=*(x+i2);
y2=*(y+i2);
acalc=y2-y1;
if(x2-x1==0) x2=x2+0.000001;
bcalc=x2-x1;
tmp=((acalc*acalc)+(bcalc*bcalc));
len=sqrt((double)tmp);
lensum=lensum+len;
}
jmp:
    return(lensum);
}
/************************************************************************/
/***** Procedure to calculate cumulative angles between line segments ****/
/************************************************************************/
 float angle(float *x,float *y,int nopts)
{
int i;
float a1,a2,x1,x2,y1,y2,x3,y3,t1,t2,t3,t4,xr2,xr3,yr2,yr3;
float angsum,angle1,angle2,theta,rotation;

angsum=0.0;
for(i=0;i<nopts;i++)
{
if(i==nopts-2) goto jmp;
x1=0.0;
y1=0.0;
x2=*(x+(i+1))-*(x+i);
y2=*(y+(i+1))-*(y+i);
x3=*(x+(i+2))-*(x+i);
y3=*(y+(i+2))-*(y+i);
if(x2-x1==0) x2=x2+0.000001;
t1=y2-y1;
t2=x2-x1;
a1=atan2(t1,t2);
rotation=-a1;
xr2=(x2*cos(rotation))-(y2*sin(rotation));
yr2=(x2*sin(rotation))+(y2*cos(rotation));
xr3=(x3*cos(rotation))-(y3*sin(rotation));
yr3=(x3*sin(rotation))+(y3*cos(rotation));
if(xr3-xr2==0) x3=x3+0.000001;
t3=yr3-yr2;
t4=xr3-xr2;
a2=atan2(t3,t4);
angle2=a2*57.29578;
theta=fabs(angle2);
```

```
angsum=angsum+theta;
}
jmp:
return(angsum);
}
/************************************************************************/
/***** Procedure to calculate area of offset between two lines **********/
/************************************************************************/
double  areaoff(float *xs2,float *ys2)
{
int    i,j,open;
float anchx,anchy,lastx,lasty;
float a,b,suma,sumb,dif,currx,curry,compx,compy;
double area_off,area;

i=0;
suma=0;
sumb=0;
open=0;
area_off=0;
  for(j=0;j<npts;j++){
currx=*(xt+j);
curry=*(yt+j);
compx=*(xs2+i);
compy=*(ys2+i);
    if((currx==compx && curry==compy) && open !=1){
    anchx=currx;
    anchy=curry;
    lastx=compx;
    lasty=compy;
    i=i+1;
    }
    else
    {
    open=1;
    a=(anchx * curry);
    b=(anchy * currx);
    suma=suma + a;
    sumb=sumb + b;
     if(currx==compx && curry==compy){
a=(currx * lasty);
b=(curry * lastx);
suma=suma + a;
sumb=sumb + b;
dif=suma-sumb;
area=fabs(dif/2);
area_off=area_off+area;
i=i+1;
open=0;
suma=0;
sumb=0;
lastx=compx;
lasty=compy;
     }
    anchx=currx;
    anchy=curry;
    }
  }
    return(area_off);
}
```

A.2. LINESTAT.C

```
/*************************************************************************
 Turbo C Program: LINESTAT.C
              Computes and stores characteristic measures of
              line complexity such as number of points, average
              angularity, wiggliness, length, and fractal
              dimension for a file of lines.

   Written by: John A. Young
   Graduate Student
   Department of Geography
   VPI & SU
   Blacksburg, VA  24061

     Line complexity measures after McMaster (1986:1987),
     Buttenfield (1989), Shelberg, Moellering, and Lam (1987),
     and Jasinski (1990).
*************************************************************************/
#include <stdio.h>
#include <io.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <alloc.h>
#define pi 3.1415926
#define rad 57.29578

/*************** define functions ******************************************/
float length(float *x,float *y,int nopts);
float anchlen(float *x,float *y,int nopts);
float angle(float *x,float *y,int nopts);
float fractd(float *px,float *py,float segavg);
float readxy(float *p2x,float *p2y,float step,float n2);
float chordlen(float *x1,float *y1,float x2,float y2,float *xp,float *yp,float step,float *n);
float linreg(float *xval,float *yval,int nval);
int   itype,gcode,npts;
float *xt,*yt,*ss,*n,maxang,minang;
FILE *filea;
FILE *fileb;

main() {

int   choice,i,j;
float length1,seglen1,angsum1,avgang1,sumlen1,anchl1,wiggly;
float con2,d1,x,y;
char  ofil[20];
char  ifil[20];

clrscr();
```

```
printf("LINESTAT: This program computes characteristic measures");
printf("\nof line complexity for a MOSS file of lines.");

/************* open input file and exit at error *****************/
printf("\n\nEnter name of input file: ");
scanf("%s",ifil);
if ((filea=fopen(ifil,"r"))== NULL)
      {
  printf("error opening file\n"); exit(0);
}

/************* open output file and exit at error ***************/
printf("\n\nEnter name file to store line measures: ");
scanf("%s",ofil);
if ((fileb=fopen(ofil,"w"))== NULL)
      {
  printf("error opening file\n"); exit(0);
}
/************* initialize measures to zero ********************/
length1=0.0;
seglen1=0.0;
angsum1=0.0;
avgang1=0.0;
sumlen1=0.0;
anchl1=0.0;
wiggly=0.0;
d1=0.0;

printf("\nProcessing...");

/************** get header of first line and begin loop ***********/
while(! feof(filea)){
fscanf(filea,"%1d%5d%5d\n",&itype,&gcode,&npts);

/************** Memory allocation for lines ***********************
   allocate enough memory only for size of line to process by
   setting memory block size to number of points
*****************************************************************/

  xt=(float *) calloc(npts,sizeof(float));
  if(xt==NULL) {
    printf("Error in memory allocation for x value");
    exit(0);
  }
  yt=(float *) calloc(npts,sizeof(float));
  if(yt==NULL) {
    printf("Error in memory allocation for y value");
    exit(0);
  }
```

```
 /****** set aside memory for other arrays ******/
  ss=(float *) calloc(npts,sizeof(float));
  if(ss==NULL) {
  printf("error in memory allocation");
  exit(0);
  }

  n=(float *) calloc(npts,sizeof(float));
  if(n==NULL) {
  printf("error in memory allocation");
  exit(0);
  }

/************* read line ******************************/
  for(i=0;i<npts;i++)
{
  fscanf(filea,"%f %f\n",(xt+i),(yt+i));
}

/********* call functions to
                         compute complexity measures for line *********/

con2=npts;                    /* number of points */
length1=length(xt,yt,npts);  /* length as sum of segments */
anchl1=anchlen(xt,yt,npts);  /* length between endpoints */
wiggly=anchl1/length1;       /* sinuosity ratio */
seglen1=length1/(con2-1);    /* average segment length */
angsum1=angle(xt,yt,npts);   /* sum of angules */
avgangl=angsum1/(npts-2);     /* average angularity */
sumlen1=angsum1/length1;      /* standardized angularity */
d1=fractd(xt,yt,seglen1);    /* fractal dimension */

/******** print measures to screen ************************/

clrscr();
printf("\n\nProcessing geocode: %d    points:%d\n",gcode,npts);
printf("\n\nlength:                        %.3f",length1);
printf("\nanchor line length:            %.3f",anchl1);
printf("\nwiggliness ratio:             %.3f",wiggly);
printf("\naverage segment length:          %.3f",seglen1);
printf("\nsum of angles:                %.3f",angsum1);
printf("\nsum of angles by length:        %.3f",sumlen1);
printf("\naverage angularity:           %.3f",avgang1);
printf("\nmax angle:                    %.3f",maxang);
printf("\nmin angle:                    %.3f",minang);
printf("\nfractal dimension:             %.3f",d1);
printf("\n\nHit any key for next line...");
getch(); /* print screen wait for response */
```

```
/* print measures to file */
fprintf(fileb,"  %d   %d   %.3f   %.3f   %.3f   %.3f   %.3f   %.3f   %.3f   %.3f   %.3f
%.3f\n",gcode,npts,length1,anchl1,wiggly,seglen1,angsum1,sumlen1,avgang1,maxang,minang,d1)

/* free memory before loading next line */
free(xt);
free(yt);
free(ss);
free(n);

/* continue loop until end of file */
}

/**** close files and free allocated memory ******/
fclose(filea);
fclose(fileb);
free(xt);
free(yt);
free(ss);
free(n);

}

/*************************************************************************/
/********** Procedure to calculate length of line ***********************/
/*************************************************************************/
 float length(float *x,float *y,int nopts)
{
int i,i2;
float tmp,len,lensum,x1,x2,y1,y2,acalc,bcalc;

/* initialize value set back to main program */
lensum=0.0;
      /* repeat measurement loop for number of segments in line */
for(i=0;i<nopts-1;i++)
{
            /* quit if at last point */
if(i==nopts-1) goto jmp;
            /* get x and y coords for endpoints of segment */
x1=*(x+i);
y1=*(y+i);
i2=i+1;
x2=*(x+i2);
y2=*(y+i2);
/* use distance formula to calculate length
               by formula of c = sqrt (a sqr + b sqr)  */
acalc=y2-y1;
/* keep from taking sqrt of zero */
if(x2-x1==0) x2=x2+0.000001;
```

```
bcalc=x2-x1;
tmp=((acalc*acalc)+(bcalc*bcalc));
len=sqrt((double)tmp);
/* cumulate length of segments to give length of line */
lensum=lensum+len;
}
jmp:
    return(lensum);
}
/***************************************************************************/
/********** Procedure to calculate anchor line length ********************/
/********* (straight line distance from first to last point) *************/
/***************************************************************************/
 float anchlen(float *x,float *y,int nopts)
{
int last;
float tmp,anchlen,x1,x2,y1,y2,acalc,bcalc;

last=nopts-1;
anchlen=0.0;
x1=*(x+0);
y1=*(y+0);
x2=*(x+last);
y2=*(y+last);
acalc=y2-y1;
if(x2-x1==0) x2=x2+0.000001;
bcalc=x2-x1;
tmp=((acalc*acalc)+(bcalc*bcalc));
anchlen=sqrt((double)tmp);
return(anchlen);
}

/***************************************************************************/
/* Procedure to calc max, min, and sum of angles between line segments ***/
/***************************************************************************/
  float angle(float *x,float *y,int nopts)
{
int i;
float a1,a2,x1,x2,y1,y2,x3,y3,t1,t2,t3,t4,xr2,xr3,yr2,yr3;
float angsum,angle1,angle2,theta,rotation;

angsum=0.0;
maxang=0.0;
minang=9999999.0;
for(i=0;i<nopts-2;i++)
{
if(i==nopts-2) goto jmp;
x1=0.0;
y1=0.0;
x2=*(x+(i+1))-*(x+i);
```

```
y2=*(y+(i+1))-*(y+i);
x3=*(x+(i+2))-*(x+i);
y3=*(y+(i+2))-*(y+i);
if(y2-y1==0) y2=y2+0.000001;
if(x2-x1==0) x2=x2+0.000001;
t1=y2-y1;
t2=x2-x1;
a1=atan2(t1,t2);
rotation=-a1;
xr2=(x2*cos(rotation))-(y2*sin(rotation));
yr2=(x2*sin(rotation))+(y2*cos(rotation));
xr3=(x3*cos(rotation))-(y3*sin(rotation));
yr3=(x3*sin(rotation))+(y3*cos(rotation));
if(yr3-yr2==0) yr3=yr3+0.000001;
if(xr3-xr2==0) xr3=xr3+0.000001;
t3=yr3-yr2;
t4=xr3-xr2;
a2=atan2(t3,t4);
angle2=a2*57.29578;
theta=fabs(angle2);
maxang=(float) max(maxang,theta);
minang=(float) min(minang,theta);
angsum=angsum+theta;
}
jmp:
return(angsum);
}

/***************************************************************************/
/**  Fractal measure procedure after Shelberg, Moellering & Lam (1987) ***/
/**  fractal walking algorithm (initial program by Dr M. O'Neill)      ***/
/**  step length initially set to 1/2 of average segment length       ***/
/**  increased geometrically thereafter                             ***/
/***************************************************************************/
float fractd(float *px,float *py,float segavg)
{
int nvals;
float step,n3,tmp1,tmp2,slope;

n3=2.0;
nvals=0;
step=segavg/2;
while (n3>1.0){
  n3=0.0;
  n3=readxy(px,py,step,n3);
  tmp1=log(step);
  tmp2=log(n3);
  *(ss+nvals)=tmp1;
  *(n+nvals)=tmp2;
  step=step*2.0;
```

```
 nvals=nvals+1;
 }

slope=linreg(ss,n,nvals);
slope=fabs(slope);
return(slope);
}

/******************* procedure readxy ************************************/
 float readxy(float *p2x,float *p2y,float step,float n2)
{
float x1,y1,x2,y2,xp,yp;
float resid;
double tmp;
int i;

x1=*(p2x++);
y1=*(p2y++);
xp=x1;
yp=y1;
i=0;
 while(i < (npts-1) ){
 x2=*(p2x++);
 y2=*(p2y++);
 chordlen(&x1,&y1,x2,y2,&xp,&yp,step,&n2);
 i=i+1;
 }
tmp=((xp-x2)*(xp-x2))+((yp-y2)*(yp-y2));
tmp=fabs(tmp);
resid=(sqrt((double)tmp))/step;
n2=n2+resid;
return(n2);
}
/********************** procedure chordlen *******************************/
float chordlen(float *x1,float *y1,float x2,float y2,float *xp,float *yp,float step,float *n)
  {
float dx1,dy1,dx2,dy2,dxa,dya,l2,m,itcp;
float a,b,c,r1,r2,xn,yn,tmp3;
double tmp1,tmp2,disc;

dx1=*x1-*xp;
dy1=*y1-*yp;
dx2=x2-*xp;
dy2=y2-*yp;
tmp1=(dy2*dy2)+(dx2*dx2);
l2=sqrt((double)tmp1);
if((fabs(l2-step) < 0.001) || (l2 > step))
{
 dxa=dx2-dx1;
 dya=dy2-dy1;
 if (dxa !=0.0)
 {
  m=dya/dxa;
  itcp=dy2 - m*dx2;
  a=1.0 + m*m;
  b=2.0 * itcp * m;
  c=(itcp*itcp)-(step*step);
  disc=fabs((b*b)-(4.0*a*c));
  tmp3=sqrt((double)disc);
  r1=(-b + tmp3)/(2.0 * a);
  r2=(-b - tmp3)/(2.0 * a);
  if (dx1 < dx2)
  {
   if(r1 > dx1){
xn=r1;
   }
   else{
xn=r2;
   }
  }
  else{
   if(r1 < dx1){
xn=r1;
   }
   else{
xn=r2;
   }
  }
  yn=xn*m + itcp;
 }
 else{
  xn=dx1;
  tmp2=fabs(step*step - dx1*dx1);
  yn=sqrt((double)tmp2);
  if(dy1 < dy2){
   if(yn < dy1) yn=-1.0*yn;
  }
  else{
   if(yn > dy1) yn=-1.0*yn;
  }
 }
 *x1= xn + *xp;
 *xp=*x1;
 *y1= yn + *yp;
 *yp=*y1;
 *n=*n+1.0;
 chordlen(x1,y1,x2,y2,xp,yp,step,n);
}
 *x1=x2;
 *y1=y2;
 return(1.0);
  }
/****************************************************************************/
/********** Procedure to fit a linear regression from x,y arrays ********/
/**********   (for a controlled [non-random] variable x only)     ********/
/****************************************************************************/
float linreg(float *xval,float *yval,int nval)
{
float sumx,sumy,sumxy,sumx2,sumy2;
float a,b,r,vals,temp;
int i;

/*r=0.0;*/
b=0.0;
/*temp=0.0;*/
sumx=0.0;
sumy=0.0;
sumxy=0.0;
sumx2=0.0;
sumy2=0.0;

for(i=0;i<nval;i++){
sumx=sumx+*(xval+i);
sumy=sumy+*(yval+i);
sumxy=sumxy+(*(xval+i)*(*(yval+i)));
sumx2=sumx2+(*(xval+i)*(*(xval+i)));
sumy2=sumy2+(*(yval+i)*(*(yval+i)));
}
vals=nval;
b=((vals*sumxy)-(sumy*sumx))/((vals*sumx2)-(sumx*sumx));
/*a=(sumx-b*sumy)/vals;
temp=(((vals*sumx2)-(sumx*sumx)) * ((vals*sumy2)-(sumy*sumy)));
r=((vals*sumxy)-(sumx*sumy))/(sqrt((double)temp));
printf("r: %f\n",r);*/
return(b);
}
```

A.3. RANDLG.C

```
/************************************************************************
 Turbo C Program:  RANDLG.C

A program to produce random simplifications of lines
to test against effectiveness of standard line
simplification algorithms

Written by John Young
Graduate Student
Department of Geography
Virginia Polytechnic Institute
Blacksburg, Virginia  24061

************************************************************************/

#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
double areaoff(float *xs2,float *ys2);
float length(float *x,float *y,int nopts);
float angle(float *x,float *y,int nopts);
void err_meas();
int  mode,itype,gcode,npts,pts,tval,count,*seq;
float *xt,*yt,*xs,*ys;
char  ifil[20],ofil[20];
FILE *filea;
FILE *fileb;

main()
{
int  i,j,nit,gc,randnum;
float x,y;

clrscr();
printf("RANDOM LINE GENERALIZATION");
printf("\nThis program will read a MOSS file of lines and generate ");
printf("\n(n) random simplifications of a line with a specified number ");
printf("\nof points. Error measures are calculated between the original");
printf("\nline and the simplifications and are stored.");

/* open input file */
file:printf("\n\nEnter name of input file:  ");
scanf("%s",ifil);
if ((filea=fopen(ifil,"r"))== NULL)
      {
  printf("error opening file\n"); exit(0);
}
```

```
/* open output file */
printf("\n\nEnter name of output file:  ");
scanf("%s",ofil);
if ((fileb=fopen(ofil,"w"))== NULL)
       {
  printf("error opening file\n"); exit(0);
}

/* get geocode of line to process */
line:printf("\nEnter the geocode of the line to process: ");
scanf("%d",&gc);
npts=0;

/* read header from MOSS file */
jmp:fscanf(filea,"%1d%5d%5d\n",&itype,&gcode,&npts);
gotoxy(1,13);
clreol();
printf("Processing geocode: %d    type: %d    points: %d",gcode,itype,npts);

if(gcode==gc)
{

 /* Memory allocation for original line  x,y's */
 xt=(float *) calloc(npts,sizeof(float));
  if(xt==NULL)  {
    printf("Error in memory allocation for x value");
    exit(0);
  }
 yt=(float *) calloc(npts,sizeof(float));
  if(yt==NULL)  {
    printf("Error in memory allocation for y value");
    exit(0);
  }

 /* read original line's points */
   for(i=0;i<npts;i++){
  fscanf(filea,"%f %f\n",(xt+i),(yt+i));
   }
}
else
{
for (i=0;i<npts;i++)
  {
  fscanf(filea,"%f %f\n",&x,&y);
  }
if(!feof(filea)){
 goto jmp;
}
else
```

```
{
printf("** Geocode not found  **");
rewind(filea);
goto line;
}
}

/* allocate memory for rand array */
seq = (int *) calloc(npts,sizeof(int));
  if(seq==NULL)  {
    printf("Error in memory allocation for seq value");
    exit(0);
  }

/* get target value for generalizations */
printf("\n\nEnter target number of points for generalization: ");
scanf("%d",&tval);

/* allocate memory for generalized line x,y's */
  xs = (float *) calloc(npts,sizeof(float));
  if(xs==NULL)  {
    printf("Error in memory allocation for xs value");
    exit(0);
  }
  ys = (float *) calloc(npts,sizeof(float));
  if(ys==NULL)  {
    printf("Error in memory allocation for ys value");
    exit(0);
  }

printf("\nEnter number of iterations (n) for lines of (target value) points: ");
scanf("%d",&nit);

for(j=0;j<=nit;j++)
{
  gotoxy(1,20);
  clreol();
  printf("processing iteration %d",j);
  pts=npts-2;
  count=0;

  /* initialize the random number generator */
  randomize();

  /* load scaled random numbers (1 -> npts) into seq array */
   while(count < pts)
   {
randnum = random(pts) + 1;

/* is number already present in seq array */
for(i=0; i<count; i++)
{
  if(randnum== *(seq+i)) break;
}

if(i>=count) /* not in sequence */
{
  *(seq+count) = randnum;
  count++;
}
  }

  /* search seq array for numbers between 1 -> tval */
  count=0;

  /* save first point */
  *(xs+count) = *(xt+count);
  *(ys+count) = *(yt+count);

  i=1;
  do{
   if(*(seq+i) < tval-1)
     {
     count++;
     *(xs+count) = *(xt+i);
     *(ys+count) = *(yt+i);
     }
   i++;
  }while(count<tval-1);

  /* save last point */
  *(xs+count) = *(xt+(npts-1));
  *(ys+count) = *(yt+(npts-1));

/*  printf("\n");
  for(i=0;i<=count;i++)
  {
  printf("xs: %.3f ys: %.3f count: %d\n",*(xs+i),*(ys+i),i);
  }*/
  count++;
  /* send simplified line to error measures routines */
  err_meas();

}

       free(xt);
free(yt);
free(xs);
free(ys);
free(seq);
```

```
fclose(filea);
fclose(fileb);
}


/**************************************************************************/
/******************** Error measures procedure ****************************/
/**************************************************************************/
void err_meas()
{
int   i,j;
float len_chng,ang_chng;
float length1,length2;
float angsum1,angsum2,avgang1,avgang2;
float cor_chng,coor1,coor2;
float area_offs;
char  ch,ofil[20];


length1=0.0;
length2=0.0;
len_chng=0.0;
angsum1=0.0;
angsum2=0.0;
avgang1=0.0;
avgang2=0.0;
ang_chng=0.0;
area_offs=0.0;

/********* measures for original line ****************/
coor2=count;
coor1=npts;
length1=length(xt,yt,npts);
angsum1=angle(xt,yt,npts);
avgang1=angsum1/(npts-2);
angsum1=angsum1/length1;


/*********** measures for simplified line *******************/
length2=length(xs,ys,count);
angsum2=angle(xs,ys,count);
avgang2=angsum2/(count-2);
angsum2=angsum2/length2;


/*********** comparison error measures *********************/
cor_chng=(1-(coor2/coor1))*100;
len_chng=(1-(length2/length1))*100;
ang_chng=(1-(avgang1/avgang2))*100;
```

```
area_offs=areaoff(xs,ys);

/*printf("\n\nWriting to error measures file");
printf("\n   %d   %d   %.4f   %.4f   %.4f     %d   %.4f   %.4f   %.4f     %.2f   %.2f   %.2f
%.4f\n",gcode,npts,length1,avgang1,angsum1,count,length2,avgang2,angsum2,cor_chng,ang_ch
ng,len_chng,area_offs);*/
fprintf(fileb,"   %d   %d   %.4f   %.4f   %.4f     %d   %.4f   %.4f   %.4f     %.2f   %.2f   %.2f
%.4f\n",gcode,npts,length1,avgang1,angsum1,count,length2,avgang2,angsum2,cor_chng,ang_ch
ng,len_chng,area_offs);

}
/****************************************************************************/
/********** Procedure to calculate length of line ***********************/
/****************************************************************************/
 float length(float *x,float *y,int nopts)
{
int i,i2;
float tmp,len,lensum,x1,x2,y1,y2,acalc,bcalc;

lensum=0.0;
for(i=0;i<nopts;i++)
{
if(i==nopts-1) goto jmp;
x1=*(x+i);
y1=*(y+i);
i2=i+1;
x2=*(x+i2);
y2=*(y+i2);
acalc=y2-y1;
if(x2-x1==0) x2=x2+0.000001;
bcalc=x2-x1;
tmp=((acalc*acalc)+(bcalc*bcalc));
len=sqrt((double)tmp);
lensum=lensum+len;
}
jmp:
    return(lensum);
}
/****************************************************************************/
/***** Procedure to calculate cumulative angles between line segments ****/
/****************************************************************************/
  float angle(float *x,float *y,int nopts)
{
int i;
float a1,a2,x1,x2,y1,y2,x3,y3,t1,t2,t3,t4,xr2,xr3,yr2,yr3;
float angsum,angle1,angle2,theta,rotation;

angsum=0.0;
for(i=0;i<nopts;i++)
{
```

```
if(i==nopts-2) goto jmp;
x1=0.0;
y1=0.0;
x2=*(x+(i+1))-*(x+i);
y2=*(y+(i+1))-*(y+i);
x3=*(x+(i+2))-*(x+i);
y3=*(y+(i+2))-*(y+i);
if(x2-x1==0) x2=x2+0.000001;
t1=y2-y1;
t2=x2-x1;
a1=atan2(t1,t2);
rotation=-a1;
xr2=(x2*cos(rotation))-(y2*sin(rotation));
yr2=(x2*sin(rotation))+(y2*cos(rotation));
xr3=(x3*cos(rotation))-(y3*sin(rotation));
yr3=(x3*sin(rotation))+(y3*cos(rotation));
if(xr3-xr2==0) x3=x3+0.000001;
t3=yr3-yr2;
t4=xr3-xr2;
a2=atan2(t3,t4);
angle2=a2*57.29578;
theta=fabs(angle2);
angsum=angsum+theta;
}
jmp:
return(angsum);
}
/***************************************************************************/
/***** Procedure to calculate area of offset between two lines **********/
/***************************************************************************/
double  areaoff(float *xs2,float *ys2)
{
int    i,j,open;
float anchx,anchy,lastx,lasty;
float a,b,suma,sumb,dif,currx,curry,compx,compy;
double area_off,area;

i=0;
suma=0;
sumb=0;
open=0;
area_off=0;
  for(j=0;j<npts;j++){
currx=*(xt+j);
curry=*(yt+j);
compx=*(xs2+i);
compy=*(ys2+i);
    if((currx==compx && curry==compy) && open !=1){
    anchx=currx;
    anchy=curry;
    lastx=compx;
    lasty=compy;
    i=i+1;
    }
    else
    {
    open=1;
    a=(anchx * curry);
    b=(anchy * currx);
    suma=suma + a;
    sumb=sumb + b;
     if(currx==compx && curry==compy){
a=(currx * lasty);
b=(curry * lastx);
suma=suma + a;
sumb=sumb + b;
dif=suma-sumb;
area=fabs(dif/2);
area_off=area_off+area;
i=i+1;
open=0;
suma=0;
sumb=0;
lastx=compx;
lasty=compy;
     }
    anchx=currx;
    anchy=curry;
    }
  }
    return(area_off);
}
```

A.4. PLOTRIV.C

```
/**************************************************************************
  Turbo C Program: plotriv.c

  Programmed by:  John Young (with technical assistance from
                  Loretta Bush and Dr. L.W. Carstensen)
                  Geography Graduate Student
                  Virginia Polytechnic Institute

  This program converts a MOSS file of a river feature into a file
  which can be plotted on a Hewlett-Packard plotter and plots
  a random measurement bi-sector to be used to measure stream width.
**************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <io.h>
#include <time.h>
FILE  *ifil;
FILE  *ofil;

main ()
  {
  int itype,gcode,gc,gc2,npts,n,point,i,j,xm,ym,choice,psiz,pts;
  char filea[20],fileb[20],yn;
  float xpt,ypt,*xt,*yt,xmin,xmax,ymin,ymax,xrange,yrange;
  float yincr,yval[30],suby,yr,yint,inslope,ydint;
  float yrnd,xrnd,x1,x2,y1,y2,slope,xcross[20],ycross[20];
  int usage = 0;

 cont:
  clrscr ();
  printf ("PLOTRIV: A program to create a plot file from a MOSS file.\n");
  printf ("Enter name of file to process -->  ");
  scanf ("%s",filea);
  if ((ifil = fopen (filea,"r")) == NULL) {
    printf ("Error in opening file %s.",&filea);
    printf ("\n\n\n\n");
    exit (0);
    }
  printf ("Enter name of file to save (*.PLT) -->  ");
  scanf ("%s",fileb);
  ofil = fopen (fileb,"w");

xmin=99999999.9;
ymin=99999999.9;
xmax=0.0;
ymax=0.0;
```

```
printf("determining window...\n\n");

do{
fscanf(ifil,"%1d%5d%5d\n",&itype,&gcode,&npts);

for(i=0;i<npts;i++)
{
  fscanf(ifil,"%f %f\n",&xpt,&ypt);
  if(xpt < xmin) xmin=xpt;
  if(ypt < ymin) ymin=ypt;
  if(xpt > xmax) xmax=xpt;
  if(ypt > ymax) ymax=ypt;
}
}while(!feof(ifil));
  rewind(ifil);
  xrange=(xmax-xmin)*100;
  yrange=(ymax-ymin)*100;
  printf("Range of plot dimensions:\n");
  printf("X range:  %.3f  Y range:  %.3f \n",xrange/100,yrange/100);

  if(xrange > 1015 || yrange > 780){
     do{
     printf("\n** Plot will not fit on 8.5 x 11 paper without scaling **");
     printf("\nChoose course of action to proceed: \n");
     printf("    1: Switch Paper Size to 11 x 17\n");
     printf("    2: Plot at original size on 8.5 x 11 paper and clip\n");
     printf("    3: Scale to fit on 8.5 x 11 paper\n");
     printf("         Enter choice:  ");
     scanf("%d",&choice);
     }while(choice < 1 || choice > 8);

     switch(choice){
case 1:
  psiz=0;
  xm=1630;
  ym=1015;
  printf("\n\n*** Be sure to load plotter with 11 x 17 paper! ***");
  break;
case 2:
  xm=1015;
  ym=780;
  break;
case 3:
  xm=xmax*100;
  ym=ymax*100;
  break;
}
  }
  else
```

```
 {
 xm=1015;
 ym=780;
 psiz=4;
 }

 fprintf (ofil,"IN;PS %d;IP 0,0,10365,7962;SC 0,%d,0,%d;\n",psiz,xm,ym);
 fprintf (ofil,"SP1;\n");
 do {
   fscanf(ifil,"%1d%5d%5d\n",&itype,&gcode,&npts);
   printf("\nProcessing geocode: %d    type: %d    points:%d\n",gcode,itype,npts);
   point=1;
   for (n=0; n<npts; n++)                        /* Creates plot file. */
     {
     fscanf(ifil,"%f %f\n",&xpt,&ypt);
     if (point == 1){
xpt=xpt*100;
ypt=ypt*100;
fprintf (ofil,"PA,PU  %3.3f  %3.3f;\n",xpt,ypt);
point = 9;
     }
     else
     {
xpt=xpt*100;
ypt=ypt*100;
fprintf (ofil,"PA,PD  %3.3f  %3.3f;\n",xpt,ypt);
     }
    }
    fprintf (ofil,"PU;\n");
 }while (!feof(ifil));

 rewind(ifil);

 printf("\n\nDo you wish to generate random line intersectors (Y/N)? ");
 scanf("%s",&yn);
 if (yn == 'Y'  ||  yn == 'y')
    {
    fscanf(ifil,"%1d%5d%5d\n",&itype,&gcode,&npts);

/****** Memory allocation for line ******************/
 xt=(float *) calloc(npts,sizeof(float));
 if(xt==NULL)  {
   printf("Error in memory allocation for x value");
   exit(0);
 }
 yt=(float *) calloc(npts,sizeof(float));
 if(yt==NULL)  {
   printf("Error in memory allocation for y value");
   exit(0);
 }
```

```
/**************************************************/
xmin=99999999.9;
ymin=99999999.9;
xmax=0.0;
ymax=0.0;

 for(i=0;i<npts;i++)
{
 fscanf(ifil,"%f %f\n",(xt+i),(yt+i));
 if(*(xt+i) < xmin) xmin=*(xt+i);
 if(*(yt+i) < ymin) ymin=*(yt+i);
 if(*(xt+i) > xmax) xmax=*(xt+i);
 if(*(yt+i) > ymax) ymax=*(yt+i);
}
     if(usage == 0){
     yincr=yrange/30;    /* divide range of y values into 30 increments */
     yr=ymin;
     for(i=0;i<30;i++){
 yval[i]=yr+yincr;     /* load array with increment values */
 yr=yr+yincr;
 }

     printf("\nEnter number of intersections to generate ( < 10): ");
     scanf("%d",&pts);
     pts=pts*2;
     randomize();            /* initialize the random number generator */
     fprintf (ofil,"SP2;\n");

j=0;
do{
suby=random(30);          /* get random subscripts for y array */
yrnd=yval[suby];

 for(i=0;i<npts;i+=2){    /* find segments with yrnd intercept */
    y1=*(yt+i)*100;
    y2=*(yt+(i+1))*100;
    x1=*(xt+i)*100;
    x2=*(xt+(i+1))*100;
    if(((y1 > yrnd) && (y2 < yrnd)) || ((y1 < yrnd) && (y2 > yrnd))){
if(x1 != x2)
 slope = (y1-y2)/(x1-x2);
else
 slope = 100000.0;
yint = y1-(slope * x1);
xrnd = (yrnd - yint)/slope;
inslope=-1/slope;
ydint = yrnd - (inslope * xrnd);
xcross[j]=(xrnd-50);
ycross[j]=(xcross[j]*inslope) + ydint;
xcross[j+1]=(xrnd+50);
ycross[j+1]=(xcross[j+1]*inslope) + ydint;
fprintf (ofil,"PA, PU %f %f;\n",xrnd+5,yrnd+5);
fprintf (ofil,"PA, PD %f %f;\n",xrnd+5,yrnd+5);
fprintf (ofil,"PA, PU %f %f;\n",xcross[j],ycross[j]);
fprintf (ofil,"PA, PD %f %f;\n",xcross[j+1],ycross[j+1]);
fprintf (ofil,"PU;\n");
j+=2;
     }
  }
}while(j<pts);
      usage = 1;
      }
     else
      {
fprintf (ofil,"SP2;\n");
for(i=0;i<pts;i+=2){
fprintf (ofil,"PA, PU %f %f;\n",xcross[i],ycross[i]);
fprintf (ofil,"PA, PD %f %f;\n",xcross[i+1],ycross[i+1]);
}
fprintf (ofil,"PU;\n");
      }
     }

  fprintf (ofil,"SP0;");
  printf ("\n\nThe plot file has been saved.");
  fclose (ifil);
  fclose (ofil);
  free(xt);
  free(yt);
  printf ("\n\nDo you wish to create another plot file (Y/N) ?  ");
  scanf ("%s",&yn);
  if (yn == 'Y'  || yn == 'y')
    goto cont;
  }
```

A.5. PLOTFEAT.C

```
/*************************************************************************
 Turbo C program: plotfeat.c

 Programmed by:  John Young and Loretta Bush
                 Geography Graduate Students
                 Virginia Polytechnic Institute

 This program converts a MOSS file into a file to be plotted
 on a Hewlett-Packard Plotter and overlays n lines of random length
 orientation, and position for measurement of distances between lines
*************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <io.h>
#include <time.h>
FILE *ifil;
FILE *ofil;

main ()
  {
  int itype,gcode,gc,gc2,npts,n,point,i,xm,ym,choice,psiz,pts;
  char filea[20],fileb[20],yn;
  float xpt,ypt,xt,yt,xmin,xmax,ymin,ymax,xrange,yrange;
  float xincr,yincr,xval[30],yval[30],subx,suby,xr,yr;
  float xrnd[30],yrnd[30];
  int usage = 0;

 cont:
  clrscr ();
  printf ("A program to create a plot file from a MOSS file.\n");
  printf ("** note: this will plot the entire MOSS file **\n\n\n");
  printf ("ENTER NAME OF FILE TO BE CONVERTED -->  ");
  scanf ("%s",filea);
  if ((ifil = fopen (filea,"r")) == NULL) {
    printf ("Error in opening file %s.",&filea);
    printf ("\n\n\n\n");
    exit (0);
    }
  printf ("ENTER NAME OF FILE TO SAVE (*.PLT) -->  ");
  scanf ("%s",fileb);
  ofil = fopen (fileb,"w");

xmin=99999999.9;
ymin=99999999.9;
xmax=0.0;
ymax=0.0;
```

```
printf("determining window...\n\n");

do{
fscanf(ifil,"%1d%5d%5d\n",&itype,&gcode,&npts);

for(i=0;i<npts;i++)
{
  fscanf(ifil,"%f %f\n",&xt,&yt);
  if(xt < xmin) xmin=xt;
  if(yt < ymin) ymin=yt;
  if(xt > xmax) xmax=xt;
  if(yt > ymax) ymax=yt;
}
 }while(!feof(ifil));

  rewind(ifil);
  xrange=(xmax-xmin)*100;
  yrange=(ymax-ymin)*100;
  printf("Range of plot dimensions:\n");
  printf("X range:  %.3f  Y range:  %.3f \n",xrange,yrange);

  if(xrange > 1015 || yrange > 780){
     do{
     printf("\n** Plot will not fit on 8.5 x 11 paper without scaling **");
     printf("\nChoose course of action to proceed: \n");
     printf("     1: Switch Paper Size to 11 x 17\n");
     printf("     2: Plot at original size on 8.5 x 11 paper and clip\n");
     printf("     3: Scale to fit on 8.5 x 11 paper\n");
     printf("          Enter choice:  ");
     scanf("%d",&choice);
     }while(choice < 1 || choice > 8);

     switch(choice){
case 1:
  psiz=0;
  xm=1630;
  ym=1015;
  printf("\n\n*** Be sure to load plotter with 11 x 17 paper! ***");
  break;
case 2:
  xm=1015;
  ym=780;
  break;
case 3:
  xm=xmax*100;
  ym=ymax*100;
  break;
}
  }
 else
 {
 xm=1015;
 ym=780;
 psiz=4;
 }

 fprintf (ofil,"IN;PS %d;IP 0,0,10365,7962;SC 0,%d,0,%d;\n",psiz,xm,ym);
 fprintf (ofil,"SP1;\n");
 do {
   clreol();
   fscanf(ifil,"%1d%5d%5d\n",&itype,&gcode,&npts);
   printf("\nProcessing geocode: %d    type: %d    points:%d\n",gcode,itype,npts);
   point=1;
   for (n=0; n<npts; n++)                       /* Creates plot file. */
     {
     fscanf (ifil,"%f %f\n",&xpt,&ypt);
     if (point == 1){
xpt=(xpt-xmin)*100;
ypt=(ypt-ymin)*100;
fprintf (ofil,"PA,PU  %3.3f  %3.3f;\n",xpt,ypt);
point = 9;
     }
     else
     {
xpt=(xpt-xmin)*100;
ypt=(ypt-ymin)*100;
fprintf (ofil,"PA,PD  %3.3f  %3.3f;\n",xpt,ypt);
     }
    }
    fprintf (ofil,"PU;\n");
 }while (!feof(ifil));

 printf("\n\nDo you wish to generate a random line overlay for feature (Y/N)? ");
 scanf("%s",&yn);
 if (yn == 'Y'  ||  yn == 'y')
    {
     if(usage == 0){
     xincr=xrange/30;        /* divide range into 20 increments */
     yincr=yrange/30;
     xr=0;
     yr=0;
     for(i=0;i<30;i++){
 xval[i]=xr+xincr;     /* load x and y arrays with successive */
 yval[i]=yr+yincr;     /* increment values */
 xr=xr+xincr;
 yr=yr+yincr;
 }

     printf("\nEnter number of lines to generate ( < 30): ");
     scanf("%d",&pts);
     pts=pts*2;
     randomize();  /* initialize the random number generator */
     fprintf (ofil,"SP2;\n");

for(i=0;i<=pts;i+=2){
subx=random(21);          /* get random subscripts for x and y */
suby=random(21);          /* arrays to choose random points */
xrnd[i]=xval[subx];
yrnd[i]=yval[suby];
fprintf (ofil,"PA, PU %f %f;\n",xrnd[i],yrnd[i]);
subx=random(21);
suby=random(21);
xrnd[i+1]=xval[subx];
yrnd[i+1]=yval[suby];
fprintf (ofil,"PA, PD %f %f;\n",xrnd[i+1],yrnd[i+1]);
}
     fprintf (ofil,"PU;\n");
     usage = 1;
     }
    else
     {
fprintf (ofil,"SP2;\n");
for(i=0;i<=pts;i+=2){
      fprintf (ofil,"PA, PU %f %f;\n",xrnd[i],yrnd[i]);
fprintf (ofil,"PA, PD %f %f;\n",xrnd[i+1],yrnd[i+1]);
}
fprintf (ofil,"PU;\n");
     }
    }

  fprintf (ofil,"SP0;");
  printf ("\n\nThe plot file has been saved.");
  fclose (ifil);
  fclose (ofil);
  printf ("\n\nDo you wish to create another plot file (Y/N) ?  ");
  scanf ("%s",&yn);
  if (yn == 'Y'  || yn == 'y')
    goto cont;
  }
```

Vita

John A. Young was born in Baytown, Texas on June 17, 1964. The seventh child in a family of nine children, he quickly learned the value of self-reliance. He received his Bachelor of Arts in Geography at Virginia Polytechnic Institute & State University in 1987, and continued in the program to pursue a Master's in Geography. He has held several positions in the mapping sciences for federal, state and local governments and has assisted in the development of geographic information systems for emergency planning, water quality assessment, integrated pest management, and wildlife research. His research interests include applications of geographic information systems and remote sensing systems to environmental planning, landscape ecology, and wildlife studies. He is currently employed as a Geographer with the US Forest Service in Olympia, Washington.

[illegible]