

GENERALIZING CARTOGRAPHIC DATA BASES
Eric K. Van Horn
Geographic Data Technology
P.O. Box 126
Academy Road
Thetford, VT 05074

BIOGRAPHICAL SKETCH

The author received his B.A. from Goddard College in 1973. His initial exposure to computers was as a health planner for Hoffman-LaRoche, Inc., in 1975, but his first serious programming efforts were doing solar system simulations on a microcomputer in 1977. In 1979 he went to work for Creative Computing Magazine as the production manager for the software department, and later became the software development manager. He moved to Vermont and started working for Geographic Data Technology in 1980 as system manager. In 1984 he was promoted to technical director.

ABSTRACT

In applications where the scale and resolution of end use is known, it makes sense to rescale digital maps to this resolution. Maps can be broken up into windows that represent the most likely viewing areas, and each window can be rescaled separately. The algorithm presented by David Douglas and Thomas Peucker in 1973 can then be applied to remove undisplayable detail. Rescaling as a pre-processing step causes more nodes to be removed with less distortion than the Douglas-Peucker algorithm applied alone. Rescaling also aids the deletion of islands and lakes that do not add to the graphic reproduction at the target resolution. In a sample file, the state boundary of Virginia was reduced 73% more than the stand-alone Douglas-Peucker algorithm, yet the amount of detail is greater.

INTRODUCTION

Generalizing, to quote David Douglas and Thomas Peucker, is the "reduction of the number of points used to represent a digitized line". There are three advantages to doing this. First, generalizing reduces the size of a file, requiring less external storage. In microcomputer applications, where external storage is often limited to floppy disks or 10 mb Winchester disks, this is important. Second, the display of maps is a compute-bound process. Ideally, the display of a map should be instantaneous, but on general purpose computers which lack special hardware, this is not the case. On a Zenith Z-100, the display of 4000 segments takes about 6 seconds. Our county boundary file, which has 224758 segments, would take over 5.5 minutes to display at this speed. Certain operations, like continuous zooming and panning, are not possible at these speeds.

Finally, in applications where the map display is done on a low or medium resolution display (up to 400 X 600 pixels), details such as shorelines may compress into blobs when rescaled to fit the resolution of the display. Just as cartographers use less detail when working on large scale maps, computer maps will look better when the amount of detail in the file matches the scale and resolution at which it is displayed.

GENERALIZING METHODS

Generalizing By Point Removal.

The process of generalizing has its origins in applications where maps were stream digitized. In stream digitizing, a map's features are traced and coordinate pairs are sampled at a set rate. A long straight line will potentially be represented by many more points than is necessary for accurate display or computer processing. Generalizing algorithms were developed to reduce the number of points. The most obvious methodology is to eliminate every nth point, effectively reducing the sampling rate. A variation on this is removing a point if it is closer than a given distance from the previous point. Point removal disregards entirely the nature of the terrain. In addition, straight lines are still not necessarily represented by only two points, and square corners may be sliced off at an angle. Examples of reduction by point removal are illustrated in Figure 1.

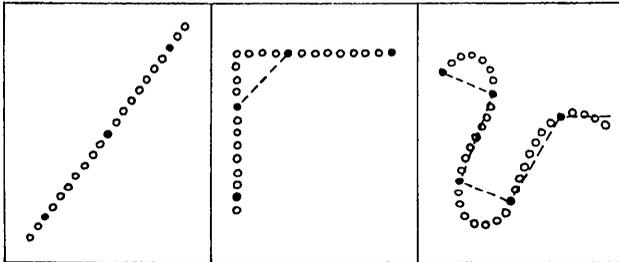


Figure 1. Line reduction by selecting every 8th point.

Feature Analysis.

Attempts have also been made to automate the ways in which a cartographer makes judgments about which features to keep and which ones to ignore. Artificial Intelligence methods can, in theory, be used to teach a program the pattern recognition skills necessary to do this. In reality, this attempts to duplicate a very complex procedure. Other methods try to evaluate the importance of features / mathematically. This is also, in fact, a non-trivial solution with many special cases. Neither approach represents a simple generalizing solution.

Line Straightening.

In 1973 David Douglas and Thomas Peucker published a paper called "Algorithms For The Reduction Of The Number Of Points Required To Represent A Digitized Line Or Its Caricature". In their algorithm, a chain of line segments is selected for reduction. A straight line is drawn from the start point to the end point. If the perpendicular distances of all the points in the chain fall within a given tolerance of the straight line, the two end points are selected and all others are discarded. Otherwise, the next point closer to the start point is selected as the end point for the straight line test. The end point continues to move closer to the start point until the test is successful. In the worst case, the start point and end point are adjacent. Figure 2 illustrates how the Douglas-Peucker algorithm works.

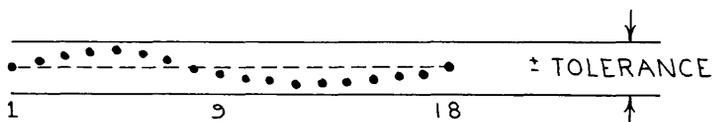


Figure 2. The Douglas-Peucker method. Points 2 through 17 all fall within a given perpendicular distance from the straight line formed by connecting points 1 and 18. In this case only points 1 and 18 would be selected.

This algorithm is simple and useful when used to remove extraneous points along a generally straight line. However, when the tolerance becomes large, features can become distorted and, as stated in their original paper, become a caricature. Figure 3 shows generalizing using the Douglas-Peucker algorithm on the state boundary of Virginia. The original file contains 6519 nodes. The coordinates for this file are in degrees and ten thousandths of a degree of latitude and longitude. At a tolerance of .0001 (one least significant unit), only 4 nodes are removed. Up to a tolerance of .0005, only 45 nodes are removed. When the tolerance is increased to .0010, although the node count is reduced by 2856 nodes to 3663 nodes, the northern border becomes distorted and is no longer a good representation of the boundary. Yet the eastern coastline along the Chesapeake Bay, which is the area most in need of generalization, has been virtually untouched.

RESCALING BEFORE GENERALIZING

Lattices.

All digital devices work in fixed increments. The precision of any digital system depends on the size of this increment. In two dimensional applications like digital mapping, this means there is a lattice or grid which determines the absolute precision of the display. A plotter with a resolution of .0005" may, at a given scale, only differentiate between features that are at least 20 feet

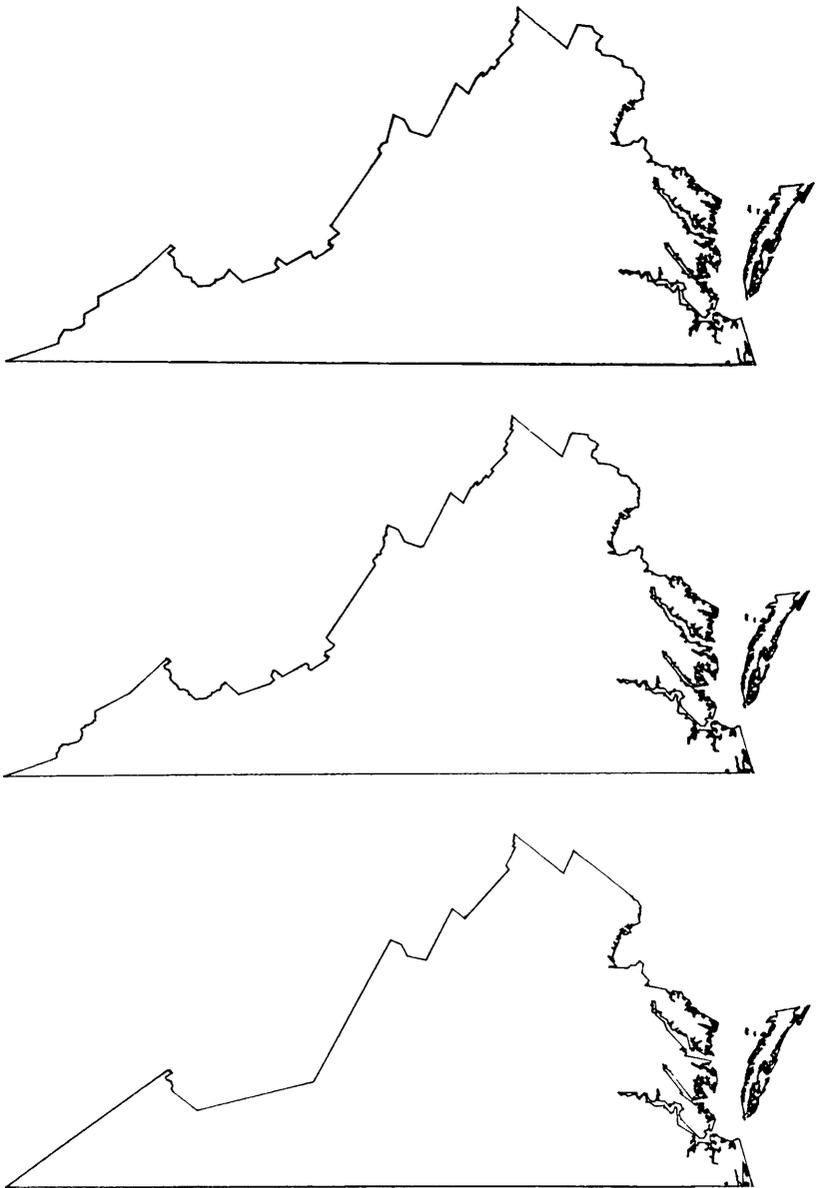


Figure 3. The state boundary of Virginia generalized using the Douglas-Peucker algorithm. From top to bottom, the files were generalized using tolerances of .0001, .0005, and .0010. The largest tolerance reduces the number of nodes from 6519 to 3663, but distorts the northern boundary.

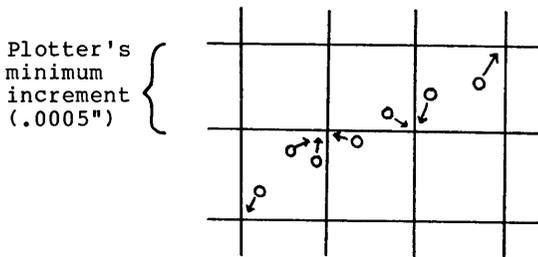


Figure 4. The lattice effect. Points scattered about in space are pulled to the nearest grid location as defined by the digital precision of the device.

apart. Coordinates that fall in between are 'pulled' to the nearest lattice location. Other display devices such as video displays and dot matrix printers have less precision than a plotter. The same map that can be plotted in increments of 20 feet on a plotter will only display to the nearest 734 feet on a 60 pixel/inch graphics screen. Figure 4 shows how measurements in the real world may be mapped onto this lattice.

In the archival storage of maps, the scale or precision of the end use is not known, so the greatest amount of detail possible is necessary. However, in individual applications where the scale and precision of the end use is known, it makes sense to rescale the data to the end use lattice and remove nodes that do not add to the graphic representation at that precision. The Douglas-Peucker algorithm works well when generalizing the rescaled coordinates, because the rescaling, i.e., 'pulling' to a grosser lattice, makes the generalizing a straight line problem at which the algorithm excels.

Figure 5 shows Virginia rescaled to fit a 600 X 400 resolution lattice (the precision of a medium resolution graphics display), and generalized using the Douglas-Peucker algorithm at a tolerance of 1 (1 pixel, or 1 least significant unit). The northern border no longer has the distortion that was present in the non-rescaled .0010 tolerance reduction, even though 5545 nodes have now been removed. The file size has been reduced to 974 nodes, 73% smaller than the .0010 tolerance reduction. Yet, as can be seen, the boundary is a more detailed representation of the boundary.

Window Definition.

In Figure 5, the coordinate extremes for the entire file were determined, and these extremes were used to calculate the factor used in rescaling. In cases where the entire file is likely to be viewed as a single entity, this method suffices. However, if you were to zoom in on a portion of the coastline, it would appear caricatured and unrealistic.

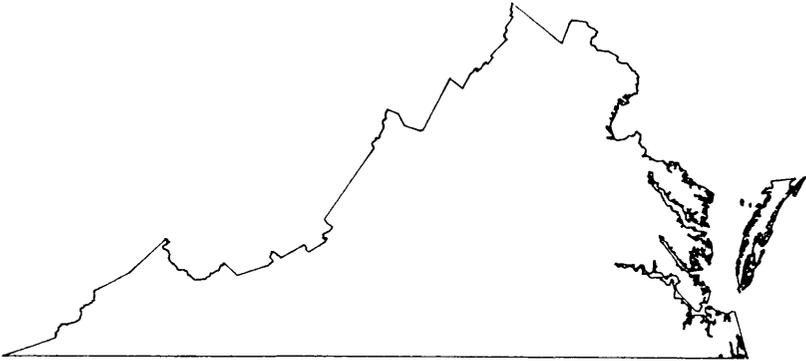


Figure 5. Virginia rescaled to fit a 600 X 400 lattice and generalized at a tolerance of 1 using the Douglas-Peucker method.

In this case it is desirable to specify that portion of the coast to be rescaled as a separate window. Because our primary application is the generalization of boundary files, the software at Geographic Data Technology has been set up to do window definition on the basis of groups of zones (counties, zip codes, etc.).

Figure 6 shows zip code boundaries for Denver. The urban core has been defined as one window, with the remainder of the Denver SMSA defined as another. The zip codes in the urban core were rescaled to the lattice using the coordinate extremes of the urban core. All other zip codes were rescaled using the coordinate extremes of the entire SMSA. In the boundaries that separate the urban and non-urban core, the greater resolution is used.

Window definition can also be done on the basis of geographic heirarchy (where one exists). MCD's, for example, can be rescaled using the coordinate extremes of the containing county. This is much simpler than manually defining zone aggregates.

Rescaling and Islands.

Rescaling and line straightening does not work well in areas where there are many islands. In these cases, small islands must be deleted. Monroe County Florida, which contains the Keys, is shown in Figure 7. In addition to the boundary generalization, islands with an area of less than 25 lattice units (pixels) have been deleted. The size of this file is 37% smaller, yet the character of the shoreline and the sense of "islandness" have been retained. This deletion also applies to lakes.

CAVEATS

It is possible that using rescaling and straightening will

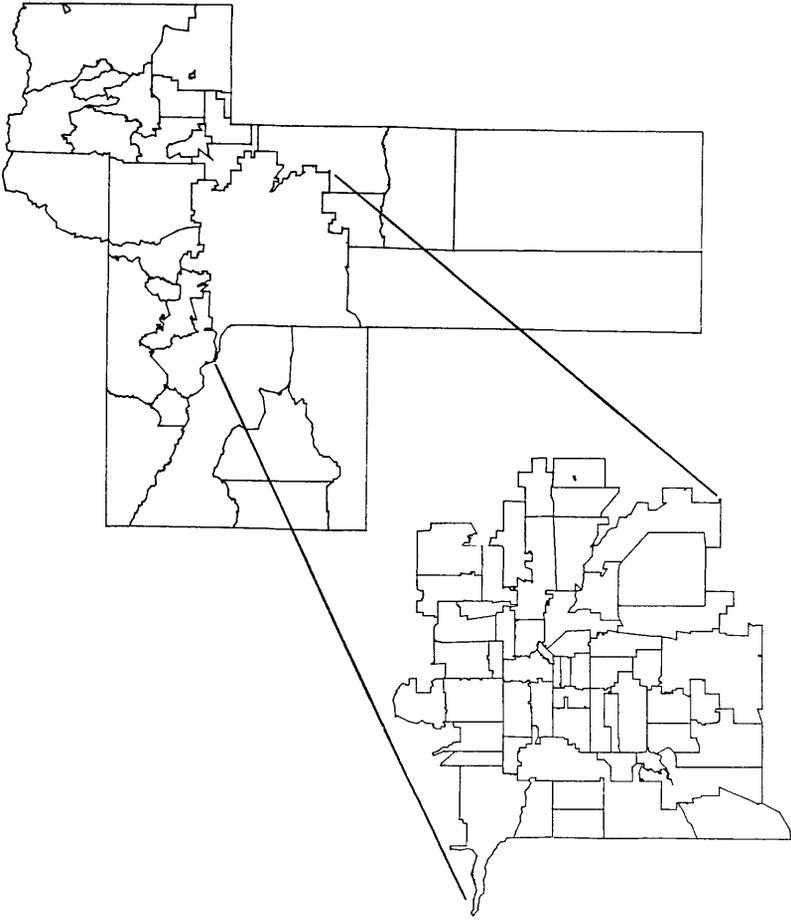


Figure 6. Denver zip code boundaries. The urban core has been generalized as one viewing window, reducing the node count from 1239 to 941. The remainder of the SMSA has been processed as another window, reducing the node count from 1981 to 930. This reflects the way in which the area is most likely to be viewed.

cause lines to cross. There are two reasons this may happen. One is that if the rescaled coordinates are then scaled back into the same units as the original ones, the values may not be the same. The rescaling destroys the precision of the coordinate value. So although the generalizing should be done on the rescaled value, the original value should be retained and used for the final output. This maintains the integrity of the absolute value of the node.

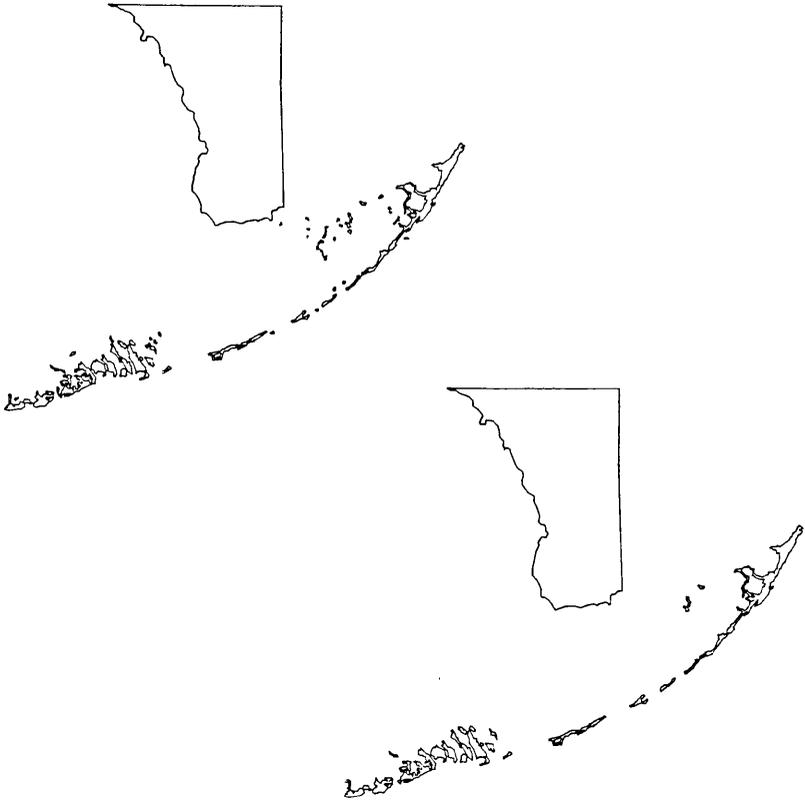


Figure 7. Monroe County Florida shown before and after generalizing. In addition to generalization of the boundaries, small islands have been deleted on the basis of minimum area.

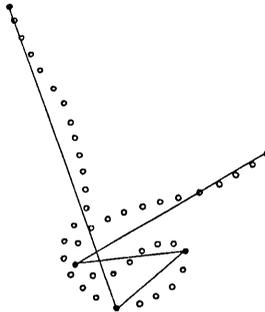


Figure 8. Crossing segments. Files generalized using rescaling and line straightening may cause lines to cross.

Even if the original values are retained, however, some lines may be made to cross. The grid pulling may cause one line to be pulled slightly one way and another line to be pulled slightly the other such that in the in-between areas of the lattice the lines may end up crossing. This phenomenon is shown in Figure 8. For this reason, an edit, such as David Douglas's cross algorithm, should be done that checks to see if any lines cross. If they do, a manual procedure can be used to move the necessary nodes.

Where geographic details are extremely jagged, such as the docks in San Francisco or along the Maine coast, line straightening with rescaling will only work if these areas are rescaled using a window set up specifically for them. Otherwise, these areas will still show up as a blob. One way to do this is as part of an interactive process. An operator can zoom in on an area, point to the window, and tell the system to generalize that window at the current viewing scale.

JOB STREAM SUMMARY

Here is a sample job stream to do rescaling and line straightening:

- 1) Define the normal viewing windows
- 2) Define the lattice
- 3) Set the tolerance for the Douglas-Peucker algorithm
- 4) Rescale the coordinates inside each window
- 5) Apply the Douglas-Peucker point reduction algorithm
- 6) Write out the remaining coordinate pairs
- 7) Load each polygon
- 8) Determine the area
- 9) Delete the polygon if the area is too small
- 10) Check for crossing segments
- 11) Manually correct segment crossings

CONCLUSION

The Douglas-Peucker line reduction method can be enhanced by rescaling coordinates to fit the lattice of the target environment before it is applied. This causes greater reduction in file size with little or no loss in detail.

REFERENCES

Douglas, David and Peucker, Thomas, 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, pp. 112-122

Douglas, David, It Makes Me So CROSS.