

Fall 2014

Gradual Generalization of Nautical Chart Contours with a B-Spline Snake Method

Dandan Miao

University of New Hampshire - Main Campus

Follow this and additional works at: <https://scholars.unh.edu/thesis>

Recommended Citation

Miao, Dandan, "Gradual Generalization of Nautical Chart Contours with a B-Spline Snake Method" (2014). *Master's Theses and Capstones*. 7.
<https://scholars.unh.edu/thesis/7>

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact nicole.hentz@unh.edu.

**GRADUAL GENERALIZATION OF NAUTICAL CHART CONTOURS
WITH A B-SPLINE SNAKE METHOD**

BY

DANDAN MIAO

BS in Geographic Information Systems, Wuhan University, 2009

THESIS

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science
in
Ocean Engineering

September, 2014

ALL RIGHTS RESERVED

© 2014

Dandan Miao

This thesis has been examined and approved.

Dr. Brian Calder,
Associate Research Professor of Ocean Engineering

Dr. Kurt Schwehr
Affiliate Associate Professor of Ocean Engineering

Dr. Steve Wineberg
Lecturer, Mathematics and Statistics

Date

ACKNOWLEDGMENTS

This study was sponsored by NOAA grant NA10NOS400007, and supported by the Center for Coastal and Ocean Mapping. Professor Larry Mayer introduced me to the world of Ocean Mapping, and taught me new information about geological oceanography; Professor Brian Calder initiated this study and has always been able to selflessly help me with any questions; Professor Steven Wineberg gave me many insights of how to transfer math concepts to graphic behavior; Professor Kurt Schwehr helped me with many intelligent thoughts and suggestion about computer programming implementation. I am grateful for all their selfless help and patience, and I would like to thank all of them for their guidance, encouragement and proof-reading of this thesis: without them and CCOM's support, this work would not have happened.

Finally, I would like to thank my parents and friends, for encouragement and trust. Your love and faith gave me the strength to keep holding on and finally make it work! Love you all!

With greatest thankfulness,

Dandan Miao

TABLE OF CONTENTS

ACKNOWLEDGMENTS	vi
TABLE OF CONTENTS.....	vii
LIST OF FIGURES	xi
ACRONYMS.....	xv
ABSTRACT.....	xvi
CHAPTER	PAGE
I. INTRODUCTION.....	17
1.1 Problem Statement	17
1.2 Research Background.....	26
1.2.1 Contours and Nautical Charts	26
1.2.2 Shoal-biased Rule of Nautical Chart Contours	28
1.3 Prior Work.....	29
1.4 Contribution of This Thesis.....	37
II. Background Theory.....	38
2.1 B-spline Curve.....	38
2.1.1 B-spline Curve Definition.....	38
2.1.2 Cubic B-spline Curve.....	39
2.2 Snake Method.....	40
2.2.1 Snake Method Definition.....	40
2.3 B-spline Snake Method	42
2.3.1 B-spline Snake Energy Terms for Polylines	42
2.3.2 B-spline Snake Energy Terms for Polygons	47
2.3.3 Smoothing Operator.....	48

2.3.3.1 Smoothing Operator Implementation.....	50
2.3.4 Simplification Operator	50
2.3.4.1 Simplificatioin Operator Implementation	51
2.3.5 Shoal-biased Operator.....	53
2.3.5.1 Shoal-biased Operator Method	54
2.3.5.2 Shoal-biased Operator Implementation	54
III. OPERATORS, AUXILIARY FUNCTIONS AND WORKFLOW DESIGN	56
3.1 Operators	56
3.1.1 Aggregate Operator.....	56
3.1.1.1 Aggregate Operator Method	58
3.1.1.2 Aggregate Operator Implementation	60
3.1.1.2.1 Aggregate Operator Case One Implementation	60
3.1.1.2.2 Aggregate Operator Case Two Implementation	63
3.1.1.2.3 Aggregate Operator Case Three Implementation	67
3.1.1.2.4 Aggregate Operator Case Four Implementation	69
3.1.1.2.5 Two Polylines	70
3.1.2 Exaggerate Operator	71
3.1.2.1 Exaggerate Operator Method.....	74
3.1.2.2 Exaggerate Operator Implementation	75
3.1.1.2.1 Polygon Exaggerate Operator Implementation.....	75
3.1.1.2.2 Polyline Exaggerate Operator Implementation.....	76
3.2 Auxiliary Functions.....	77
3.2.1 B-spline Snake calculation.....	78
3.2.2 Preprocess Polygon Contour Function.....	79
3.2.3 Maintain Minimum Distance Between Neighbor Contours Function	80
3.2.4 Polygon Group Intersection Prevention Function.....	80

3.2.5 Self Intersection Removal Function.....	81
3.3 Workflow	82
3.3.1 Workflow for Single Polyline Contour Generalization	82
3.3.2 Workflow for One Polyline and Multiple Polygons On One Side.....	83
3.3.3 Workflow for A Group of Polygon Contours Exaggeration and Aggregation	85
3.3.4 Workflow for Concentric Polygon Contours Exaggeration and Aggregation	86
3.3.5 Workflow for Polyline and Polygon Contour Exaggeration and Aggregation	87
3.4 Summary	89
IV. TEST EXAMPLES.....	90
4.1 Test Scenario One: One Line Example	90
4.2 Test Scenario Two: Polyline Contour and Polygon Contour Aggregation	97
4.3 Test Scenario Three: A Group of Polygon Contours Exaggeration and Aggregation.....	103
4.4 Test Scenario Four: Concentric Polygon Contours Exaggeration and Aggregation	110
4.5 Test scenario Five: Polyline and Polygon Contours Exaggeration and Aggregation.....	115
V. DISCUSSION AND FUTURE WORK.....	124
5.1 Discussion	124
5.2 Future Work	125
LIST OF REFERENCES	127
APPENDICES	129
A.0 Distance Unit Definition.....	130
A.1 Parameter in Algorithm 3-2.....	130
A.2 Parameter in Algorithm 3-4.....	132
A.3 Parameter in Algorithm 3-13.....	133
A.4 Point Adding Parameter in Algorithm 3-14	134
A.5 Neighbor Points Parameter in Algorithm 3-14 and 3-9.....	135

A.6 Delete Points Parameter in Algorithm 3-14 and 2-1	135
A.7 Parameter in Algorithm 3-8.....	135
A.8 Parameter in Algorithm 3-10.....	136
A.9 Parameter in Algorithm 3-11	136
A.10 Parameter in Algorithm 3-13, 3-14, 3-16, 3-17	137

LIST OF FIGURES

Figure 1-1: Generalization of paper nautical chart	17
Figure 1-2: Generalization result of a B-spline Snake Method.....	18
Figure 1-3: One 60 foot contour on a 1:20,000 scale raster chart.....	20
Figure 1-4: One 30 foot contour on a 1:80,000 scale raster chart.....	21
Figure 1-5: One 60 foot polyline contour and several 60 foot polygon contourson	22
Figure 1-6: One 60 foot contour from the 1:80,000 scale raster chart 13286.....	23
Figure 1-7: Polygon contours on 1:20,000 scale raster chart.....	24
Figure 1-8: Polygon contours on 1:80,000 scale raster chart.....	25
Figure 1-9: Example illustrating the shoal-biased rule	29
Figure 1-10: Operators of generalization	30
Figure 1-11: Example of simplification process	31
Figure 1-12: The Douglas-Peucker algorithm	32
Figure 1-13: Example of smoothing process	32
Figure 1-14: Example of aggregation process	33
Figure 1-15: Example of exaggeration process.....	34
Figure 1-16: Example of how each Shea and McMaster operators works.....	35
Figure 2-1: Control points ($Q_0, Q_1 \cdots Q_7$) of a B-spline curve (grey solid line)	39
Figure 2-2: First three degrees of basis functions for B-spline curves	40
Figure 2-3: Curvature definition at parameter φ_i	43
Figure 2-4a: Internal and external forces in the generalization process.....	46
Figure 2-4b: Internal and external forces in the generalization process	46
Figure 2-4c: Internal and external forces in the generalization process.....	47
Figure 2-5: Sample spatial transformation of smoothing operator	48

Figure 2-6: Sample contour line from 1:20,000 scale raster chart.....	49
Figure 2-7: Sample contour line from 1:40,000 scale raster chart.....	49
Figure 2-8: Sample spatial transformation of simplification operator	51
Figure 2-9: Sample of contours in 1:40,000 scale and 1:80,000 scale raster chart.....	51
Figure 2-10: Example of shoal-biased principle	53
Figure 3-1: Sample spatial transformation of aggregate operator.....	56
Figure 3-2. Sample of aggregation of a polyline and polygons	57
Figure 3-3: Sample of aggregation of two polygons.....	58
Figure 3-4: Sample of aggregation of a group of simple polygon contours	58
Figure 3-5: Samples of aggregate operator Cases One - Four	59
Figure 3-6: Two steps of aggregate operator in Case One.....	60
Figure 3-7: Finding the starting and ending index for the selected segment of the polyline and polygon.....	62
Figure 3-8: Check if line S_1S_2 and line E_1E_2 intersect with the polygon	62
Figure 3-9: Aggregate operator Case One result	63
Figure 3-10: Aggregation Case Two.....	64
Figure 3-11: Angles formed by the supporting line and neighbor segment should be obtuse	64
Figure 3-12: Aggregate operator Case Two result.....	67
Figure 3-13: Aggregate operator Case Three step one	68
Figure 3-14: Aggregate operator Case Three step two	68
Figure 3-15: Aggregate operator Case Three result.....	69
Figure 3-16: Two steps of aggregate operator in Case Four.....	69
Figure 3-17: Sample spatial transformation of exaggerate operator.....	71
Figure 3-18a: Sample of exaggeration of simple polygon contour0.....	71
Figure 3-18b: Sample of exaggeration of simple polygon contour.....	72

Figure 3-19: Sample exaggeration of simple polygon contour	72
Figure 3-20: Sample of exaggeration of one complex long polyline.....	73
Figure 3-21: Illustration of balloon force at one vertex of a polygon contour.....	74
Figure 3-22: Illustration of a polygon contour after exaggeration.....	76
Figure 3-23: Illustration of when polyline exaggeration is needed.....	76
Figure 4-1: Original input with background of raster chart 13283	91
Figure 4-2: Original input on raster chart 13278	92
Figure 4-3: Original input contour line with star symbol at each vertex	93
Figure 4-4: Intermediate stage of the generalization process.....	94
Figure 4-5: Generalization result of one simple polyline contour generalization.....	95
Figure 4-6: Original input line versus gradual generalization results	96
Figure 4-7: Original input contours with background of raster chart	97
Figure 4-8: The input contours on the background of raster chart.....	98
Figure 4-9: Original input of one polyline and nine polygon contours.....	99
Figure 4-10: An early stage of generalization at about 10% of the whole iteration	100
Figure 4-11: Intermediate stage of generalization at about 30% of the whole iteration	101
Figure 4-12: The final state of generalization	102
Figure 4-13: Input polygon contours on a background of raster chart 13283.....	103
Figure 4-14: Input polygon contours on a background of raster chart 13286.....	104
Figure 4-15: Input of a set of simple polygon contours.....	105
Figure 4-16: Intermediate stage of the generalization process at 18 th iteration	106
Figure 4-17: Intermediate stage of the generalization process at 68 th iteration	107
Figure 4-18: Intermediate stage of the generalization process at 140 th iteration	108
Figure 4-19: Generalization result of test scenario 3 at 244 th iteration	109
Figure 4-20: Input polygon contours on a background of raster chart 13283.....	110
Figure 4-21: Selected area on raster chart 13286.....	111

Figure 4-22: Input concentric contours in depth coded color at iteration 0	112
Figure 4-23: Intermediate stage of the generalization process at 44 th iteration	113
Figure 4-24: Generalization result of test scenario four at 251 st iteration.....	114
Figure 4-25: Input data with a background of raster chart 13283.....	115
Figure 4-26: Input data on the background of 1:80,000 scale raster chart.....	116
Figure 4-27: Initial input data at iteration 0	117
Figure 4-28: Intermediate stage of the generalization process at 11 th iteration	118
Figure 4-29: Intermediate stage of the generalization process at 17 th iteration	119
Figure 4-30: Intermediate stage of the generalization process at 150 th iteration	120
Figure 4-31: Intermediate stage of the generalization process at 900 th iteration	121
Figure 4-32: Generalization result at 1600 th iteration	122
Figure 4-33: Generalization result comparison.....	123
FIGURES IN APPENDICES	
Figure A.1: Location of vertices in algorithm 3-2	132
Figure A.4: Location of vertices in algorithm 3-14	134

ACRONYMS

ECDIS----- Electronic Chart Display and Information System

ECS ----- Electronic Chart System

ENC ----- Electronic Navigational Charts

NOAA-----National Oceanic and Atmospheric Administration

ABSTRACT

GRADUAL GENERALIZATION OF NAUTICAL CHART CONTOURS WITH A B-SPLINE SNAKE METHOD

By

Dandan Miao

University of New Hampshire, September, 2014

B-spline snake methods have been used in cartographic generalization in the past decade, particularly in the generalization of nautical charts where these methods yield good results with respect to the shoal-bias rules for the generalization of chart contours. However, previous studies only show generalization results at particular generalization (or scale) levels, and show only two states of the algorithm: before and after generalization, but nothing in between. This thesis presents an improved method of using B-spline snakes and other auxiliary functions and workflows for generalization in the context of nautical charts which can generalize multiple nautical chart features from large scale to small scale without creating any invalid intermediate features that require special processing to resolve. This process allows users to generate charts at any intermediate scale without cartographic irregularities, and is capable of extension to include more specialized generalization operators.

CHAPTER I

INTRODUCTION

1.1 Problem Statement

Generalization is a branch of cartography which studies the process of how the contents of a map change when the scale of the map changes (Figure 1-1). The generalization process is traditionally done by cartographers manually, even though computers are widely used in the map production process. Generalization, due to its complex nature, remains a procedure that requires large amounts of manual processing.

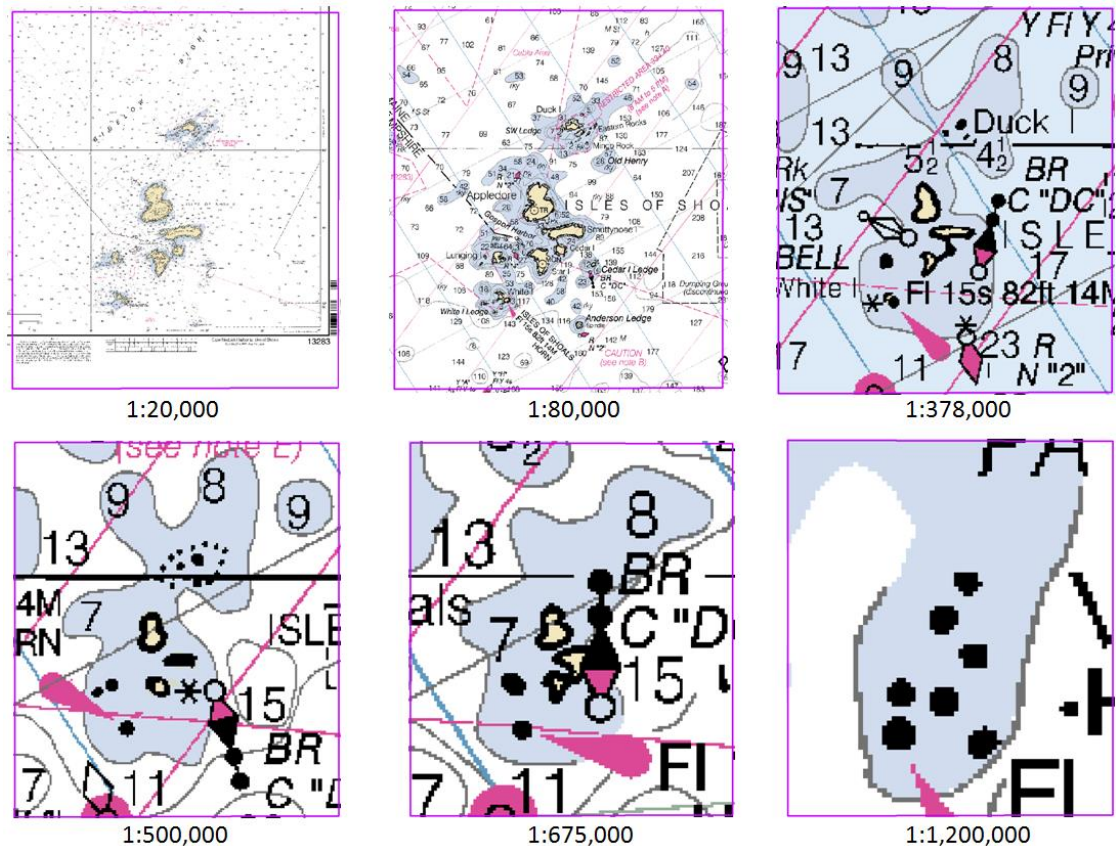


Figure 1-1: Generalization of paper nautical chart

Selected area of paper nautical charts of Duck Island (coastal New Hampshire) at various scales. All of the images represent the same geographic area.

Although generalization is a complex process, many studies have been done in this field. For land maps, studies have been conducted to establish principles of generalization (Shea and McMaster, 1989; Wang and Muller, 1998; Ware, 2003). However, the studies listed here focus on land maps. A nautical chart, on the other hand, is another type of map; it is a graphic representation of a maritime area and adjacent coastal regions. The contents of a nautical chart are different from a land map, and the purposes are different too, which leads to distinct generalization rules for nautical charts. Studies on nautical chart generalization are not as frequent as for land maps. Guilbert and collaborators (Guilbert and Lin, 2007; Guilbert and Saux, 2008) used a B-spline Snake method to generalize the contours of a nautical chart. However, their method only showed the starting and finishing status of contours after the generalization process (Figure 1-2).

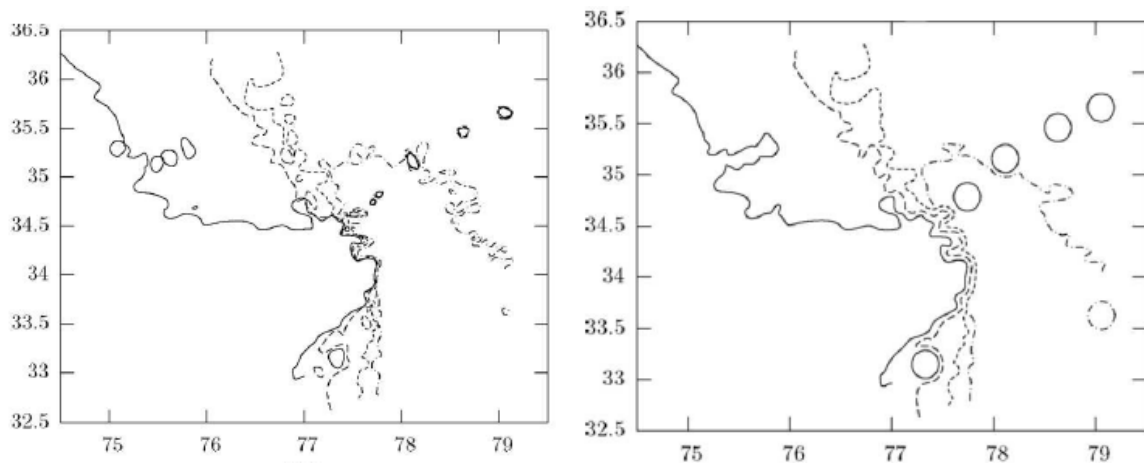


Figure 1-2: Generalization result of a B-spline Snake Method

The figure on the left side is the selected nautical chart before generalization; the figure on the right side is the generalized result (Guilbert and Saux, 2008).

Generalization should be a gradual process between scales: when the scale gradually changes the contents should change gradually too. This study focuses on finding generalization tools, operators, and workflows to make generalization a gradual process, and to carry out generalization without causing cartographic difficulties in the process.

Current generalization processes are mostly done by cartographers manually. With their previous knowledge and experience, cartographers draw new contours on a smaller scale chart based on contours and sounding data from larger scale charts. By examining how cartographers do generalization, two rules can be summarized as principles of chart contour generalization:

- 1) From a large scale chart to a small scale chart, contours are simplified and smoothed, and when their shape is changed, they are only moved to the deeper side of the original curve (Figure 1-3 and Figure 1-4).

From Figure 1-3 and Figure 1-4, compared to the 30 foot contour on the 1:20,000 scale chart, the 30 foot contour on the 1:80,000 scale chart is simplified and smoothed, and when it is smoothed, its shape is changed such that the 30 foot contour on the 1:80,000 scale chart is moved to the deeper side of the 30 foot contour on the 1:20,000 scale chart. The reason why smoothing is only done by shifting the contour to the deeper side (primarily navigational safety) is discussed in section 1.2.2.

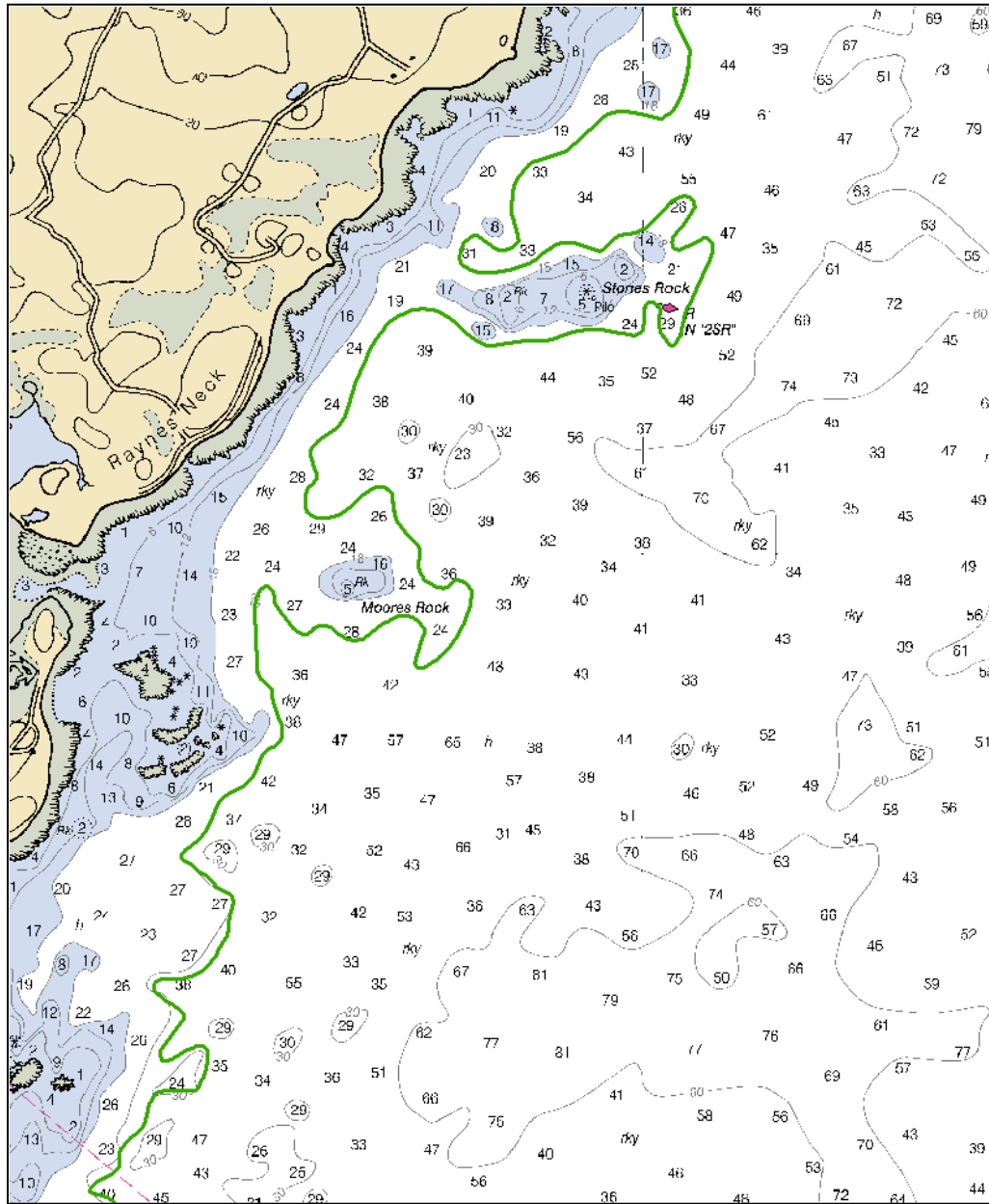


Figure 1-3: One 30 foot contour on a 1:20,000 scale raster chart

The green contour in this figure is a 30 foot contour from raster chart 13283 (scale 1:20,000) of Portsmouth Harbor, NH. Together with Figure 1-4, these two figures show how the contour representing the same depth has different shapes in charts of different scales.

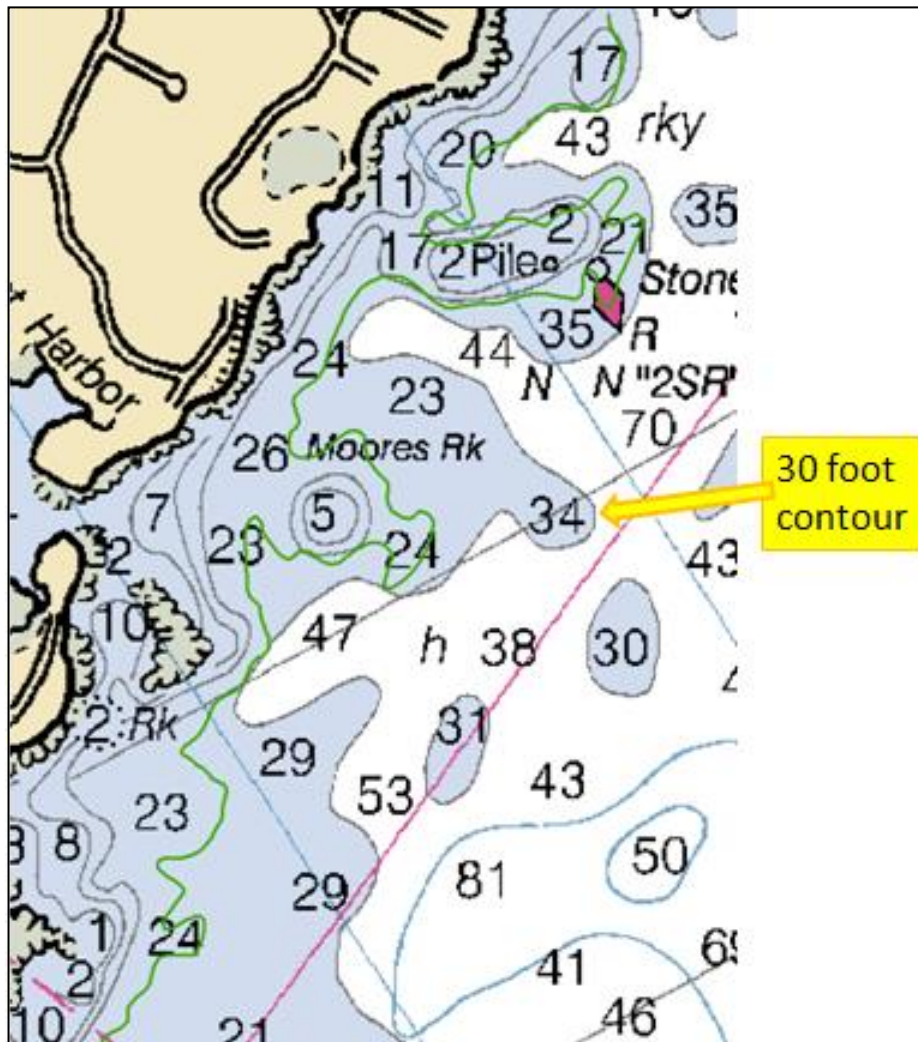


Figure 1-4: One 30 foot contour on a 1:80,000 scale raster chart

The grey line indicated by the green arrow is the 30 foot contour from raster chart 13278 (scale 1:80,000) of Portsmouth Harbor, NH. The green line is the 30 foot contour on the 1:20,000 scale chart. The grey line does not overlap with the green line, it moves to the deeper side of the original green 30 foot contour, and the shape of the grey contour is less complex than the green contour.

- 2) From a large scale chart to a small scale chart, polyline (open contours) and polygon contours (enclosed contour) will be aggregated; polygon contours will aggregate with each other and eventually be aggregated with the polyline contour. Figure 1-5 and Figure 1-6 demonstrate how cartographers aggregated polygon contours with a polyline contour during generalization: in the 1:80,000 scale chart, the polygon contours are deleted and

aggregated into the 60 foot contour line. Figure 1-7 and Figure 1-8 show that cartographers aggregate small polygon contours from the 1:20,000 scale chart with large polygon contours from the 1:80,000 scale chart.

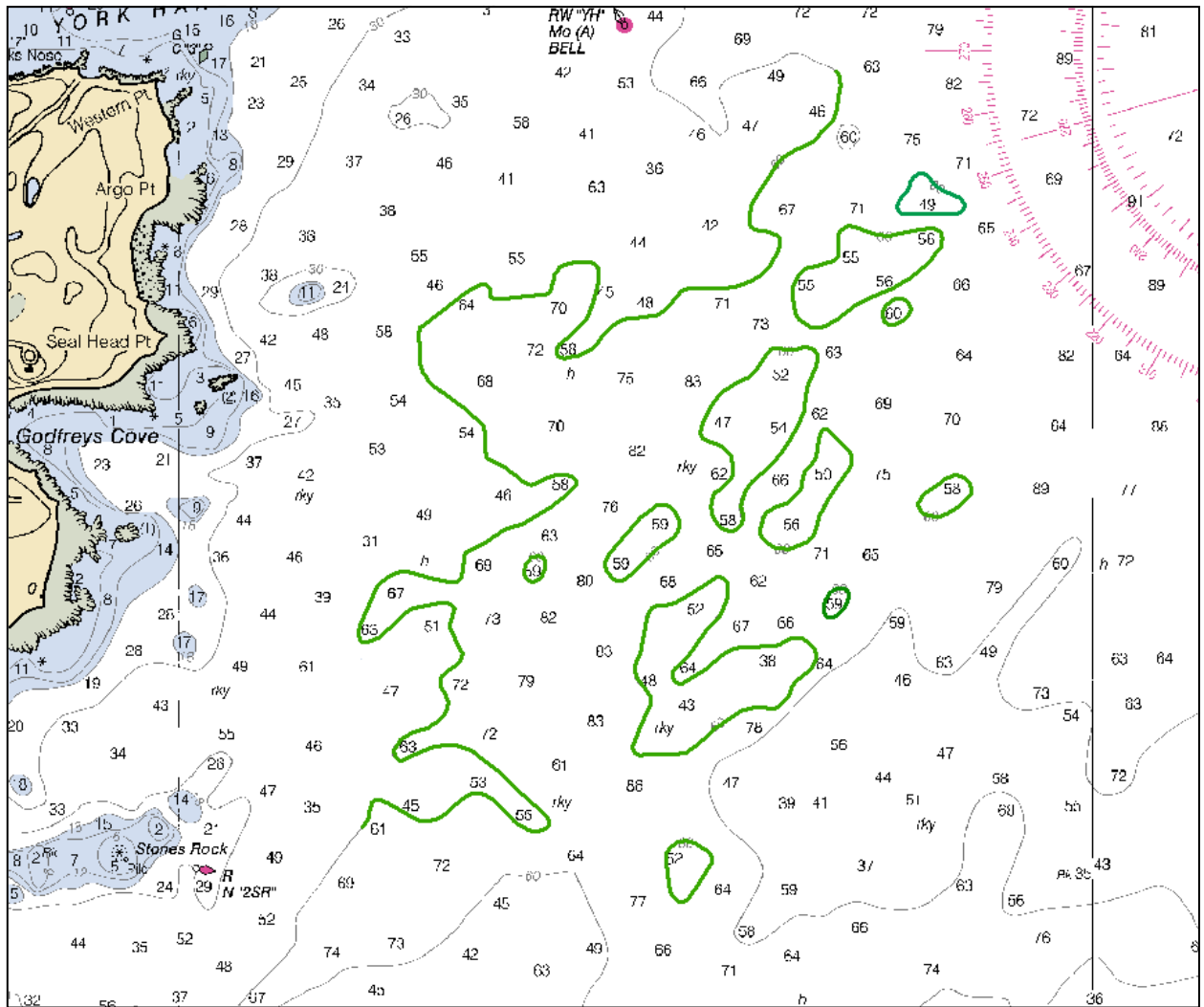


Figure 1-5: One 60 foot polyline contour and several 60 foot polygon contours

The green contours are the selected 60 foot contours from the 1:20,000 scale raster chart 13283 of Portsmouth Harbor, NH. This figure shows how one 60 foot polyline and several 60 foot polygon contours on the 1:20,000 scale chart aggregate with each other. Together with Figure 1-6, these two figures show how polygon contours are aggregated in cartographers' manual process of generalization.

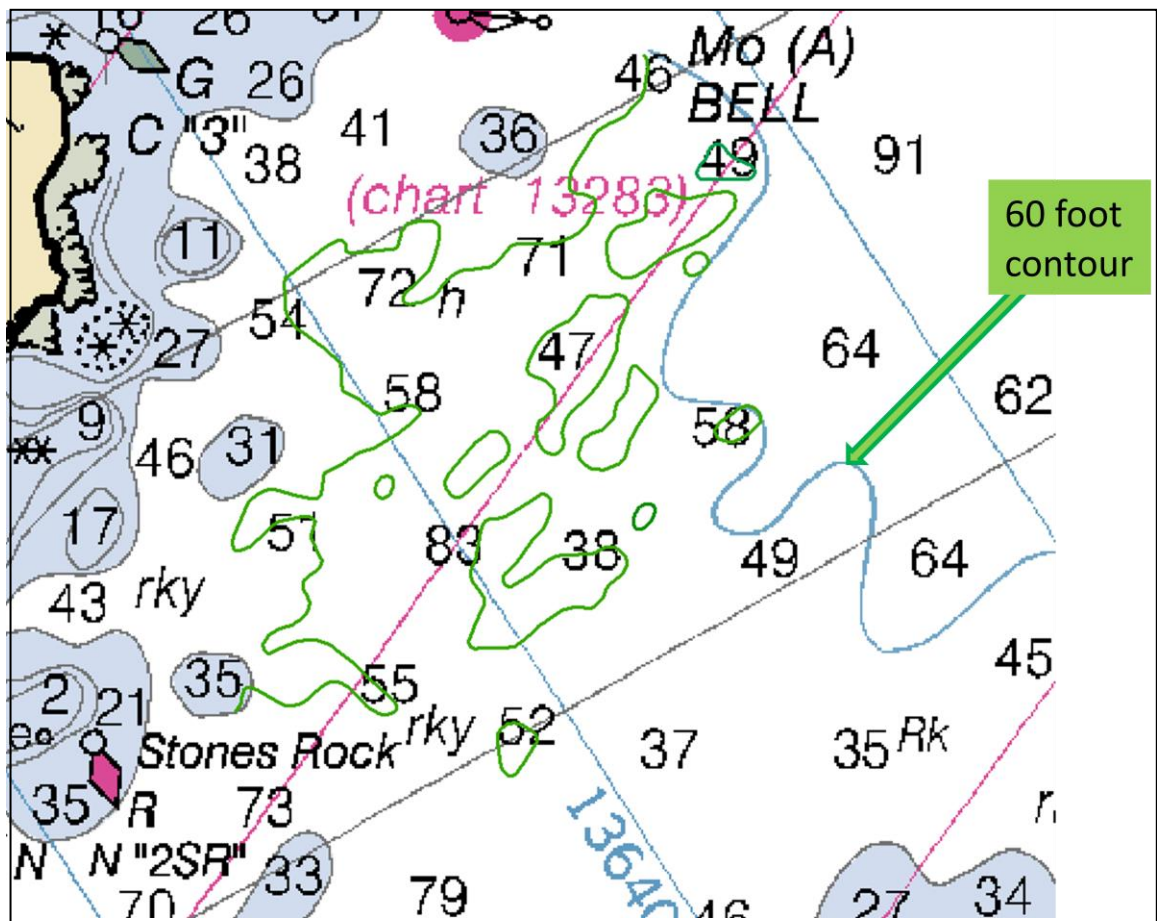


Figure 1-6: One 60 foot contour from the 1:80,000 scale raster chart 13286

The green contours are the 60 foot polyline and polygon contours from the 1:20,000 scale chart; the blue contour (indicated by the green arrow) is the 60 foot contour from the 1:80,000 scale raster chart 13826 of Portsmouth Harbor, NH. In this figure, all these 60 foot green contours, which are originally from the 1:20,000 scale chart, are all deleted. There is only one 60 foot contour on the 1:80,000 scale chart. All polygon contours are aggregated with the polyline contour, and the generalized result is only one polyline contour (the blue curve). Note: the pink "Chart 13283" label means the detail information at this area can be found on chart 13283.

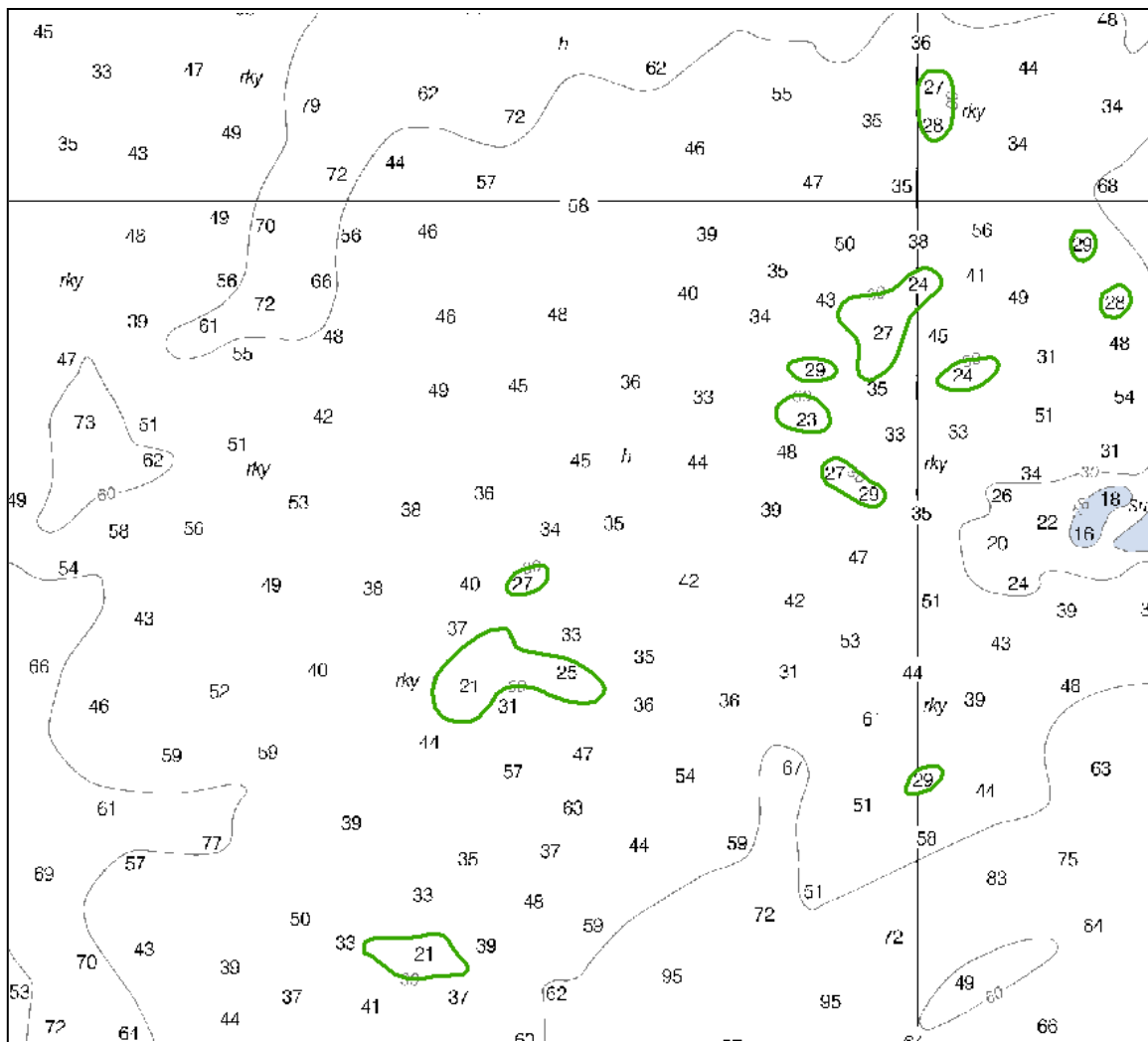


Figure 1-7: Polygon contours on 1:20,000 scale raster chart

The green contours are the selected 30 foot polygon contours from raster chart 13283 of Portsmouth Harbor, NH. This figure and Figure 1-8 show how polygon contours aggregate with each other in cartographers' generalization.

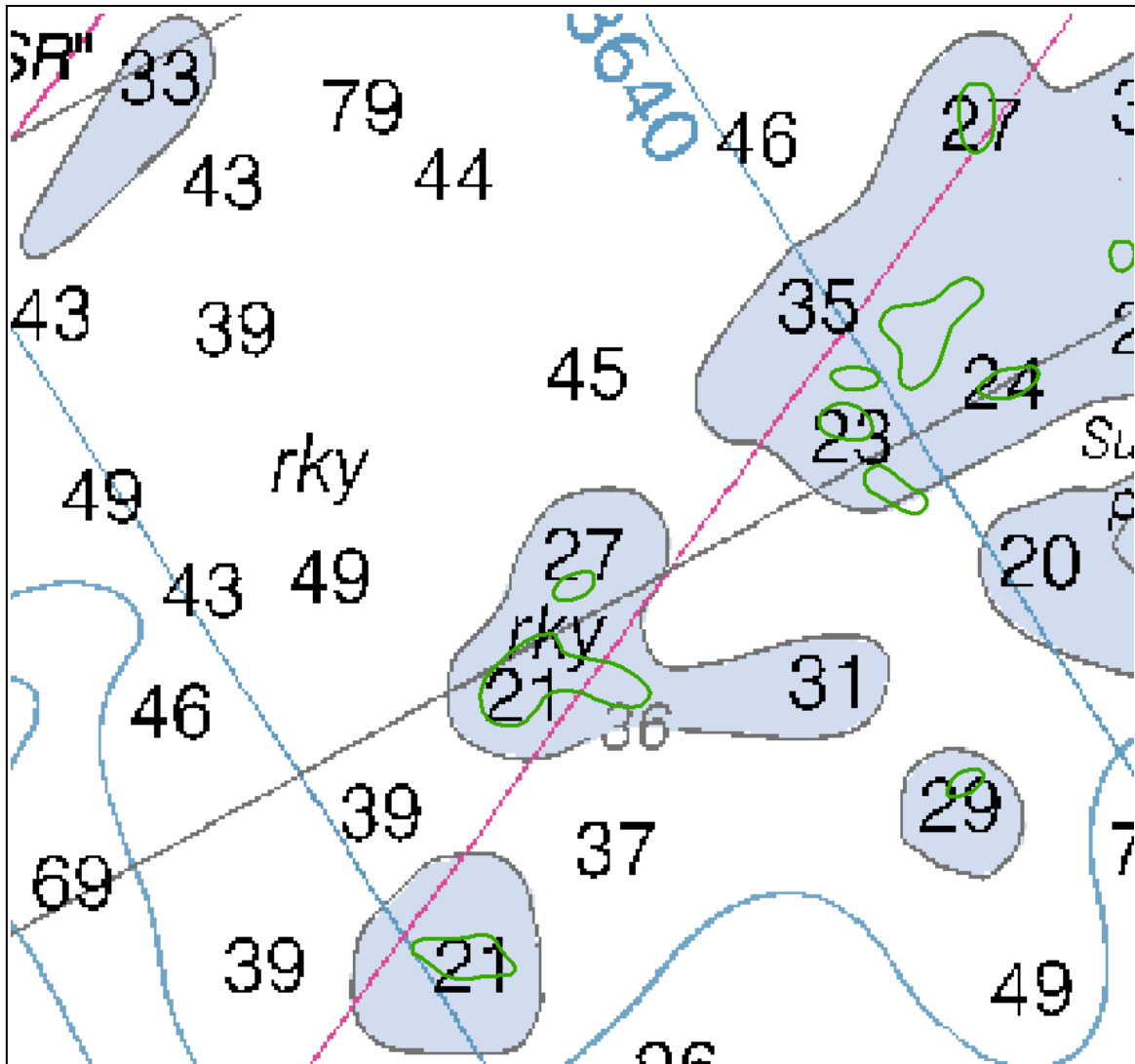


Figure 1-8: Polygon contours on 1:80,000 scale raster chart

The grey polygon contours with blue tint inside are the corresponding contours to Figure 1-7 on raster chart 13826 of Portsmouth Harbor, NH. The green contours are the contours with the same depth on the 1:20,000 scale chart. Compared to the green contours, the grey contours are larger and each grey polygon covers several green polygon contours. This figure and Figure 1-7 show that in the generalization process, the cartographers exaggerate each polygon, and aggregate neighbor polygon when they get too close.

However, in the manual process of generalization of raster charts, cartographers only provide contours at certain scales, for example at the 1:20,000 and 1:80,000 scales in Figure 1-5 to Figure 1-8. In reality, users might want more scales in between, as it is a large scale change from 1:20,000 to 1:80,000. The question is how to create a process that does generalization similar to the way cartographers do it manually with the ability to show contours at intermediate stages. This

study will focus on developing algorithms that simplify and smooth the contours, and exaggerate and aggregate contours when needed. These algorithms will be combined into workflows that generate a gradual generalization process, such that the intermediate stages of generalization will be available, and large scale nautical chart features can be generalized into small scales without creating any invalid intermediate cartographic errors.

1.2 Research Background

1.2.1 Contours and Nautical Charts

Contours are one of the primary bathymetric features on nautical charts. They depict the geomorphologic shape of the seafloor, indicate the shallow areas, and provide safety of navigation information for mariners. Nautical charts make a distinction between isobaths (i.e., a line that connects all points with the same depth) and contours (i.e., a line that contains all points shallower [shoaler] than a given depth). This thesis is concerned with contours, as they are a more general description of a depth boundary, and required for maintenance of navigational safety when constructing a chart.

A nautical chart is a different form of map; it is a graphic representation of a maritime area and adjacent coastal regions. Unlike a map, which is oriented to terrestrial use, the nautical chart provides information relevant to marine navigation (NOAA, 1997). The focus of the nautical chart is on water areas, providing data on water depths, aids to navigation (ATONs), hazards, etc. (NOAA, 1997).

Charts are generally constructed from multiple sources of bathymetric data (e.g., soundings from various sources, contours, indications of obstructions) and non-bathymetric data (e.g., floating aids to navigation, shore-line constructions, tides currents). Traditionally, charts were constructed at a particular scale of representation in order to depict the information at a level of detail suitable for the intended use (e.g., very large scale, perhaps 1:5,000, for docking charts,

through to very small scale, perhaps 1:1,000,000 or less, for planning an ocean crossing). Most often, the source surveys for the charts were conducted at a scale twice that of the largest scale charts for the area being surveyed and smaller scale charts were constructed from the larger scale charts by a process of generalization. As the scale of the chart changes, the contents shown on the chart are necessarily different, as the space available to represent any given physical area is smaller. The detail available at the largest scale cannot be shown clearly at smaller scales. Clarity of representation is essential in a chart in order to provide a useful working document, and to promote navigational safety for surface vessels. Generalization is the process of choosing which contents should be shown and how they will be represented on the chart to achieve these goals.

More recent practice has been to construct fully electronic charts (i.e., Electronic Navigational Charts [ENCs]) for use in computer-based bridge navigation systems. These systems allow the user to zoom in and out essentially continuously and therefore require that the display system (either an Electronic Chart System [ECS] or Electronic Chart Display and Information System [ECDIS]) provide generalized data to the user on demand. Currently, navigation systems select the best chart available for the region from a set of charts (typically the chart with the closest scale match to that required), and display it, generalizing only within the limits of the scale minimum and maximum information coded into the chart's source data. These systems are essentially autonomous of the cartographer. Once the source data is supplied, automatic methods for generalization are even more important than they are in the traditional paper-based chart construction pipeline: here they need to be usable for safe navigation, and preferably aesthetically pleasing, without human intervention.

Nautical charts differ from land maps in that they do not intend to faithfully represent the true nature of the seafloor in the area of interest, or, necessarily, all of the other components in the region. Rather, the goal, is to provide a representation of the area that is as faithful to the known true configuration of the seabed as possible (in as much as the – usually limited – source data provides information on the true configuration of the seabed), modified such that the information

is inherently safe for surface navigation. For example, the nautical cartographer might move an indicated sounding in order to improve the clarity of the display, or intentionally modify the representation in order to suggest to the mariner that an area of the chart is unsuitable for transit. In all cases, the nautical chart must obey shoal-bias rules, meaning that the chart always shows the shallowest depth at a given position, or a modification of the known configuration of the seafloor such that the depth indicated on the chart is shallower than the cartographer knows where the water to be. This difference requires the process of nautical chart generalization to be very different from land map generalization.

1.2.2 Shoal-biased Rule of Nautical Chart Contours

A contour in a nautical chart is different from a contour in a topographic map. A nautical chart contour has another property due to the navigation purpose of a nautical chart.

For ships, one of the largest dangers when cruising in the water is running aground. Mariners always want to ensure the water they are in is deeper than the vessel's draft. For that reason, the depths on the chart always represent the shallowest water depth at that location. That is why the chart datum is chosen to be the mean lower low water level, and hydrographic survey data is traditionally processed by selecting the shallowest value. These practices all follow the shoal-biased rule. For contours, the shoal-biased rule is also applied, which means if a contour represents a depth of 30 feet, it will only be drawn around the positions where the real depths are the same as or deeper than 30 feet. This characteristic of chart contours leads to another rule in chart contour deformation: if a contour needs to be moved to another position due to generalization, it can only be moved to a position deeper than that contour's depth.

As shown in Figure 1-9, the five meter contour cannot be moved toward the inside of its original polygon, as the real depth will be shallower than five meters. It can only be moved toward the outside of its original polygon, as the real depth at those positions will be deeper.

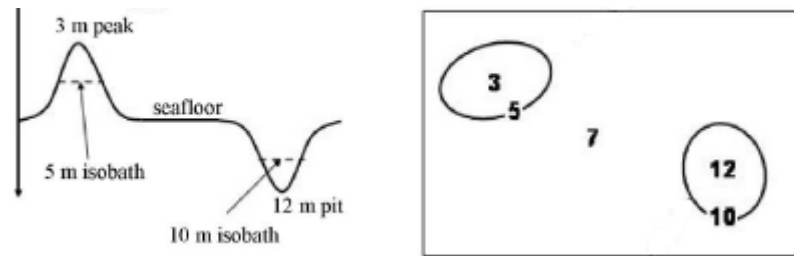


Figure 1-9: Example illustrating the shoal-biased rule

The left sub-figure shows the seafloor geomorphology, the right sub-figure shows nautical chart contours at the corresponding position (Guilbert and Saux, 2008). In the right sub-figure, the five meter contour cannot move toward the shallow side (shrinking towards label three), as if it shrinks its shape, people will think the area immediately outside the contour is deeper than five meters, but in reality it is not, which will be a hazard to vessels: ships might run into the shallower area, and cause damage to the vessel. However, it is safe for the contour to move to the deeper side. If the five meter contour expands towards the sounding label seven, it will be safe, as the real water depth in the expanded area is deeper than the contour value, and mariners will not have the risk of running aground.

1.3 Prior Work

Generalization has been mainly studied on land maps in prior work. Although land maps are different from nautical charts, a subset of research results on land map generalization can be applied to nautical charts.

Generalization in GIS contains two main aspects: database generalization and view generalization (Peng, 2000). Database generalization is also called model generalization, and is generalization through changes in the conceptual model, which consists of “manipulating the geometric and thematic descriptions of spatial objects and their relationships with respect to certain changes of the uncertainty application model” (Harrie, 2001). View generalization is also called graphic generalization or cartographic generalization, and is “mapping/transforming the digital description of spatial objects and their relationships into a graphic description, which is confined to graphic legibility and cartographic principles” (Harrie, 2001).

Shea and McMaster (1989) proposed a complete concept of operators for generalization. They divided the generalization process into several operators. The generalization process has long been a subjective process: by dividing generalization functions into operators, the generalization process can be described more objectively. The operators are summarized in Figure 1-10.

The Shea and McMaster operators are not all applicable to both types of generalization. Some operators can only be applied to graphic generalization, while some can only be applied to model generalization. This thesis research is focused on graphic generalization, so only certain operators will be studied.

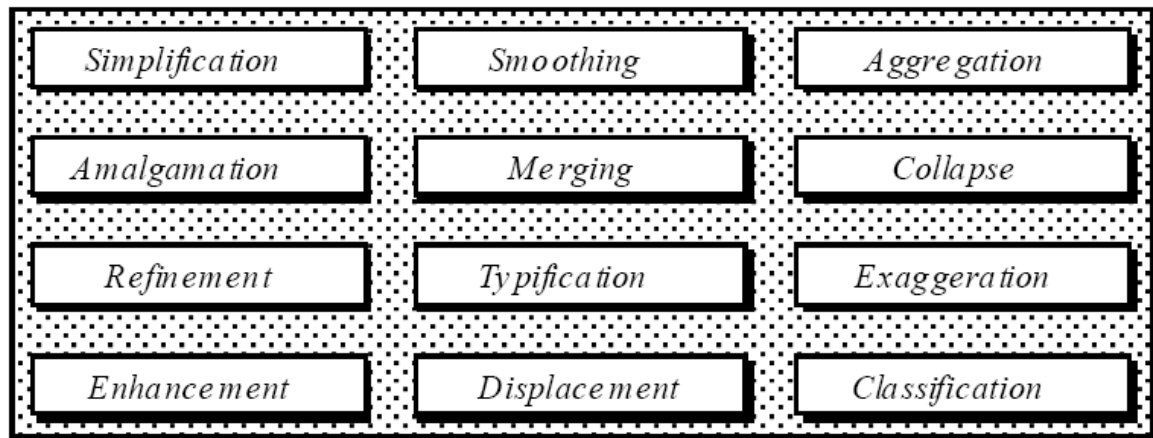


Figure 1-10: Operators of generalization

This figure shows 12 types of operators (Shea and McMaster, 1989).

Shea and McMaster decompose generalization into 12 types of operators. In this work, however, only four operators (simplification, smoothing, aggregation, and exaggeration) will be considered.

The simplification operator produces a reduction in the number of derived data points by selecting a subset of the original coordinate pairs, retaining those points considered to be the most representative of the line (Shea and McMaster, 1989). It is useful when the input data are complex. It increases the calculation speed and reduces the space required for storage. Figure 1-11 is an

example of the simplification process. The original line has seven vertices. The simplification operator selects four of them, such that those four points represent the original shape best.



Figure 1-11: Example of simplification process

The simplification operator is also the operator that been studied most, and several widely used algorithms have been developed to implement the simplification operator.

The Douglas-Peucker algorithm (Douglas and Peucker, 1973) is by far the most used simplification algorithm. The algorithm begins by defining the first point on the line as an anchor and the last point as a floating point (Figure 1-12). These two points then define a line segment and the orthogonal distance to the other points on the line is computed. If the distance is longer than the threshold distance, the point lying furthest away becomes the new floating point (Harrie, 2001). This cycle is repeated and the floating point moves towards the anchor point. When all the distances (from the line segment between the anchor and the floating point and intervening points) are less than the threshold distance, the anchor is moved to the floating point and the last point is reassigned as the new floating point. The algorithm ends when the last point becomes the anchor (Harrie, 2001).

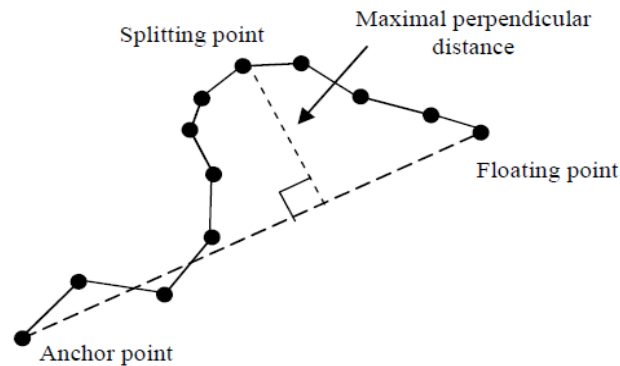


Figure 1-12: The Douglas-Peucker algorithm

If the maximal perpendicular distance is longer than the threshold value, the splitting point will become the new floating point and the procedure is repeated. If the maximal perpendicular distance is shorter than the threshold value, the line segment between the anchor and the floating point is set to represent that part of the line (Harrie, 2001).

The smoothing operator acts on a line by relocating or shifting coordinate pairs in an attempt to plane away small perturbations and capture only the most significant trends of the line (Shea and McMaster, 1989). The smoothing operator reduces the angularity of lines. Figure 1-13 shows how the smoothing process works: all vertices are preserved, but some of them are relocated.

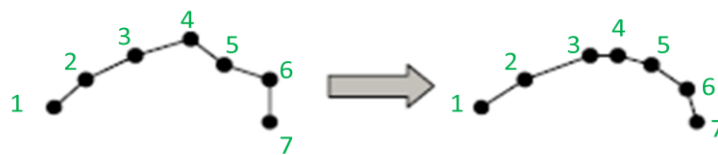


Figure 1-13: Example of smoothing process

Smoothing is another operator that has been studied in detail. Since it is not deleting any points but shifting the position of the vertices, it is mostly implemented by using a mathematical model such as a smoothing kernel, or spline functions. Gaussian smoothing is one of the common smoothing method, where the line is convolved with a Gaussian kernel. B-splines are used

frequently too due to their continuity and smoothness properties. In this study, a B-spline smoothing method is used. The detailed properties of B-splines are explained in Chapter II.

The aggregation operator is used to combine several features into one feature to symbolize the feature when the space on the map is limited and the features are important and need to be shown (Figure 1-14). There are few studies specially focusing on the aggregation operator. For different generalization objects, the aggregation operator may be implemented differently: if point features are to be aggregated, algorithms might be related to point elimination. In this study, the aggregating objects are polyline and polygon features, so a computer graphic approach is developed to implement the aggregation operator, details of which are in Chapter III.

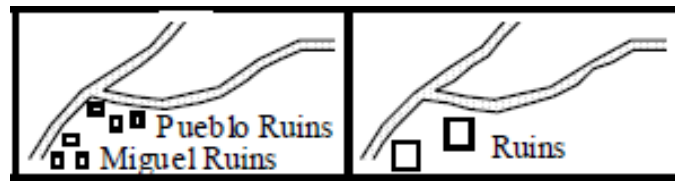


Figure 1-14: Example of aggregation process

In the left sub-figure, there are six polygon features representing ruins; in the right sub-figure, there are just two figures representing the ruins, each one of these two polygons represents three small ruins in the left sub-figure. This aggregation process provides more space on the map to clearly draw the features, but still shows the two distinct group of ruins from the original (Shea and McMaster, 1989).

The exaggeration operator is used in the generalization process such that the shapes and sizes of features can meet the specific requirements of a map (Shea and McMaster, 1989). Figure 1-15 shows an example of exaggeration of an inlet. Inlets need to be opened and streams need to be widened if the map must depict important navigational information for shipping (Shea and McMaster, 1989). As with the aggregation operator, exaggeration has not been studied much in previous research. In this work, a method to implement an exaggeration operator is developed; details of the exaggeration operator are in Chapter III.

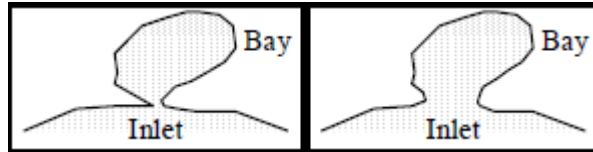


Figure 1-15: Example of exaggeration process

In the left sub-figure the entrance of the inlet is relatively narrow. In the right sub-figure, the entrance is enlarged, such that it is not closed if the whole contour shrinks. This process maintains a legible size of the inlet feature such that it will be visible when the whole shape shrinks due to the scale change (Shea and McMaster, 1989).

Other operators are widely used in the generalization process. In this study, only the four operators discussed above are used, the remaining operators will not be illustrated. Figure 1-16 illustrates how the remaining Shea and McMaster operators work.

An operator is just a concept that represents the transformation of geographic features, but to accomplish generalization automatically, algorithms are needed to implement those transformations. Many studies of generalization algorithms and workflows have been done. One algorithm from this research is to use a snake method to do line simplification, smoothing and displacement (Steiniger and Meier, 2004; Burghardt, 2005). The reason this method is superior to traditional line simplification method such as the Douglas-Peucker (1973) or Li-Openshaw (1993) methods is that it can combine several operators (such as simplification, smoothing, and displacement) together (Steiniger and Meier, 2004), and also preserve the compound shape of linear features better (Burghardt, 2005).


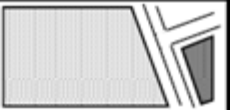




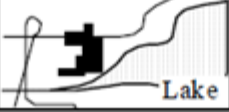





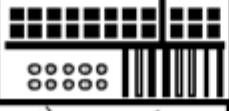








Amalgamation			
Merge			
Collapse			
Refinement			
Typification			
Enhancement			
Displacement			
Classification	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20	1-5, 6-10, 11-15, 16-20	Not Applicable

Figure 1-16: Examples of how each Shea and McMaster operator works

Eight other operators are illustrated in this figure. Amalgamation is similar to aggregation but is only applied on polygon features. Merge is when two polylines do not have enough space in between when the scale decreases, then one of the polylines is deleted, and only one polyline feature is retained. Collapse is to change polyline or polygon features' shapes when the shapes are too complicated, and the details will not be maintained in the new smaller scale map. Refinement is to select random features from a group of features, and use these symbols to represent the original group of features. Typification is similar to Refinement, but the process of selecting subset features follows certain rules instead of random selection. Enhancement is similar to exaggeration, but it deals with more than one feature. Displacement is similar to exaggeration, but it deals with the condition when two features are already in conflict. Classification is the process of grouping features into categories with respect to certain rules.

Besides the large amount of research on land maps, there is also some research specifically on nautical charts. NOAA has conducted several studies about nautical chart cartographic generalization (Shea, 1988), and nautical chart production (NOAA, 1996). Shea's cartographic generalization study provided a system that was made of several generalization operators, but those operators can only be applied one by one, and the user cannot generate a globally controlled

generalization result. Besides that, the study did not take the special characteristics of chart features into account; those generalization operators may produce incorrect results. The 1996 study is preliminary research focused on proposing a new concept of how the future chart production procedure should be. Not much was mentioned about generalization.

Besides the above NOAA conducted research, Guilbert and Lin (Guilbert and Lin, 2007) introduced a B-spline snake method to nautical chart contour generalization. This method demonstrates several generalization operators, and takes the shoal-bias rule into consideration. However, this process only creates results at a given level of generalization, and there is no intermediate result between the original chart scale and the generalized scale. In reality, when a chart with a generalization function is being displayed on an ECS or ECDIS screen, it is more appropriate to have the generalization happen smoothly as the user zooms in and out between scales. Current generalization studies all provide generalization results at some given generalization level, but no research has shown gradual generalization on a nautical chart; this thesis addresses that question.

In summary, the current generalization process has limitations. It is a very subjective process done by cartographers manually, which is time consuming, and cannot be included in ENCs (Electronic Nautical Chart, which have to rely on pre-generalized contours). It limits the generalization to fixed scale bands, which means it cannot readily deal with a continuously variable scale. The methods that have been attempted for this allow mistakes to happen, and then resolve them, which is sub-optimal and leads to special rules that make the process complex. The problem here is to find a scheme that will allow for automated generalization that maintains nautical cartography rules, while allowing for generalization to any scale from a high resolution source of survey data and avoiding the creation of invalid intermediate solutions that would require special processing to resolve.

1.4 Contribution of This Thesis

This thesis presents an improved method of using B-spline snakes and other operators, auxiliary functions and workflows for generalization in the context of nautical charts, where the generalization process is done gradually, and large scale nautical chart features with more details are generalized into smaller scales without creating any invalid intermediate features that require special processing to resolve. During the generalization process, multiple contours are aware of each other, and follow appropriate cartographic rules. This workflow also allows a user to generate chart features at any scale, and it is capable of adding more operators, functions and forces into its current structure.

CHAPTER II

BACKGROUND THEORY

In this chapter, a B-spline snake method will be discussed which implements the simplify and smoothing operators while following the shoal-biased rule. Section 2.1 introduces the background and method of construction for a B-spline curve; section 2.2 introduces background on Snake methods; and section 2.3 discusses how a B-spline Snake method works, and how it acts as a simplify and smoothing operator while obeying the shoal-bias rule.

2.1 B-spline Curve

2.1.1 B-spline Curve Definition

A spline is a piecewise polynomial function. A B-spline is a spline function consisting of a sum of B-spline basis functions (see Equation (2) below).

A two-dimensional B-spline curve is a parametric function $f(u) \in \mathbb{R}^2$ defined on an interval: $u \in I = [a, b] \subset \mathbb{R}$:

$$f(u) = \sum_{i=0}^m Q_i N_i^k(u) \quad (1)$$

The points $Q_i \in \mathbb{R}^2$ are the control points of the curve (Figure 2-1); they define the control polygon of f .

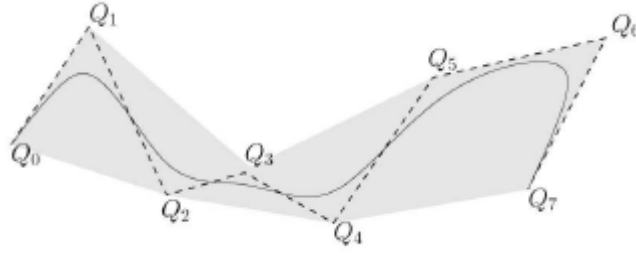


Figure 2-1: Control points ($Q_0, Q_1 \dots Q_7$) of a B-spline curve (grey solid line)

Source: Guilbert and Lin, 2007.

N_i^k are the B-spline basis functions. They are piecewise polynomial functions of degree $k-1$ defined on I . To define them, we need a series of real values, which are called the knot vector ($u_0 = a \leq u_1 \leq \dots \leq u_i \leq \dots \leq u_{m+k} = b$). The basis functions are defined recursively (Guilbert and Lin, 2007):

$$N_i^1(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$N_i^j(u) = \frac{u - u_i}{u_{i+j-1} - u_i} N_i^{j-1}(u) + \frac{u_{i+j} - u}{u_{i+j} - u_{i+1}} N_{i+1}^{j-1}(u) \text{ for } 2 \leq j \leq k$$

k is the order number, and $k-1$ is the degree of the polynomial pieces. Degree three is the most widely used, and the B-spline with degree three is also called a cubic B-spline.

2.1.2 Cubic B-spline Curve

A curve with a continuous first derivative is called C^1 continuous, and a curve with a continuous second derivative is called C^2 continuous. Figure 2-2 illustrates basis functions for degrees 1, 2 and 3..

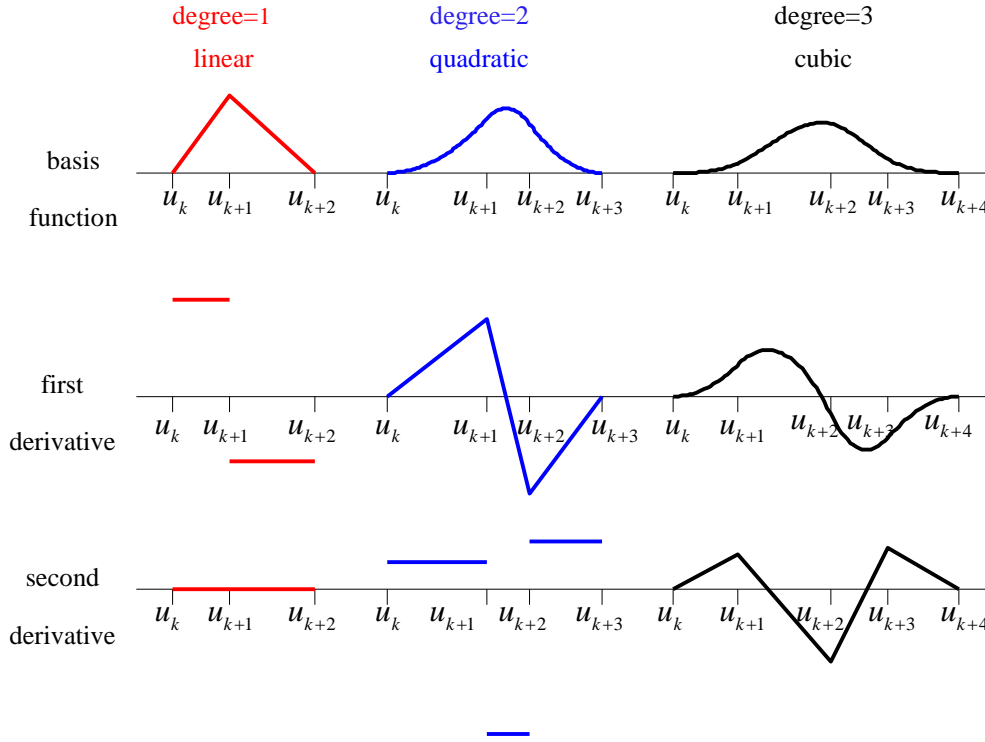


Figure 2-2: First three degrees of basis functions for B-spline curves

The u_k are the knot vector values that determine the shape and differentiability of the basis functions. When the knots u_k are all distinct, a B-spline basis function of degree k is k times continuously differentiable. When the degree is one, the curve is continuous at points u_k , u_{k+1} and u_{k+2} , but not differentiable. When the degree is two, the curve is differentiable at the points u_k , u_{k+1} , u_{k+2} and u_{k+3} , but does not have a continuous second derivative. When the degree is three, the curve has continuous first and second derivatives at points u_k , u_{k+1} , u_{k+2} , u_{k+3} and u_{k+4} . Degree three (cubic) B-splines have C^2 continuity at each knot, and requires a relatively small amount of calculation.

2.2 Snake Method

2.2.1 Snake Method Definition

Snakes, also called active contours, were first used in image processing by Kass *et al.* (1987). In image processing, a snake is a curve defined within an image domain that can move under the influence of internal forces that describe the curve itself and external forces computed

from the image data (Xu and Prince, 1998). The snake is defined as a two-dimensional parametric curve $X(u)=[x(u), y(u)]$, $u \in [0,1]$, on which the forces are defined through an energy-like term:

$$E_{total} = E_{int}(X(u)) + E_{ext}(X(u)) \quad (3)$$

$E_{int}(X(u))$ is the internal energy of the curve, describing the smoothness, and $E_{ext}(X(u))$ is the external energy, which expresses external constraints on the system. In the system defined here, these external constraints are used to represent the shoal-bias rule such that when the external energy is minimized, the shoal-bias rule has been satisfied. The snake in use here is an optimization algorithm that attempts to find the $X(u)$ that minimizes E_{total} . In general, the algorithm seeks a shape of the curve to balance the effects of the internal and external energies such that the resultant curve is as smooth as possible while still satisfying the external constraints, which may be either hard constraints – i.e., that must be satisfied – or soft constraints that express a degree of preference.

In the most common snake method, the internal energy is represented as:

$$E_{int} = \int_0^1 \frac{(\alpha |X'(u)|^2 + \beta |X''(u)|^2)}{2} du \quad (4)$$

$X'(u)$ and $X''(u)$ are the first and second derivative of $X(u)$ with respect to u , α and β are weighting parameters that control the balance between the snake's tension and rigidity respectively (Xu and Prince, 1998), and are adjusted to emphasize the required features for the given problem. The exact expression of internal energy and external energy can be different according to the particular purpose of the snake curve. Here, both terms have different definitions for contour generalization purposes. The details of the definition are in the next section.

2.3 B-spline Snake Method

In this study, input data points representing the original contour are approximated by a cubic B-spline curve as in (1), where $N_i^k(u)$ are the piecewise approximating polynomials and the Q_i are the control points, i.e. the weights of the polygon. So, before the generalization starts, the input contour is seen as a B-spline curve. Points on that B-spline curve are designated $X_0(u_j)$. Because these points are an approximation of the original contour, a polygonal line (polyline) with the $X_0(u_j)$ as its vertices can be viewed as an approximation of the original contour defined by the input data points. Then a “curvature” of this polyline can be defined at its vertices as in section 2.3.1 below.

Because this approximation is only used on input contours with complex shape and large numbers of vertices (more than 1000 vertices), the control points are normally so close to the original contour that the human eyes cannot distinguish between them and the points $X_0(u_j)$. As a consequence, generalizations in this paper use the control points themselves as proxies for the $X_0(u_j)$, so the polyline formed with the control points as vertices is the line to be simplified and smoothed. For future work, using the correct approximation points $X_0(u_j)$, and the correct spline curvature at those points, would give greater accuracy and flexibility, especially in cases where the number of points is not so large.

At the end of the iterative generalization process, the result is a final polyline, and a final B-spline is fitted to its vertices.

2.3.1 B-spline Snake Energy Terms for Polyline

For use in this work, the geomorphologic constraints depend mainly on the rigidity of the snake, and therefore the value of α is set to zero. Guilbert *et al.* (2006), show that the α value

has little influences on the final result, so it has been set to zero to simplify the calculation. In this work a cubic B-spline is fitted to the data points representing the original contour. Points on that B-spline, designated $X_0(u_j)$, are used for the subsequent contour generalization. A polygonal line is drawn with the $X_0(u_j)$ as vertices, and then a curvature of this polygonal line is defined at its vertices (Figure 2-3) by finding the internal angle φ_j between three consecutive points on the curve, ,

$$P_{j-1} \equiv X_0(u_{j-1}), P_j \equiv X_0(u_j), \text{ and } P_{j+1} \equiv X_0(u_{j+1})$$

and then approximating the curvature (Guilbert *et al.*, 2006) as

$$k(u_j) = \frac{\sin(\varphi_j)}{\frac{1}{2} \left\| \hat{P}_{j+1} - \hat{P}_{j-1} \right\|} \quad (5)$$

A more accurate approximation is outlined in Chapter V.

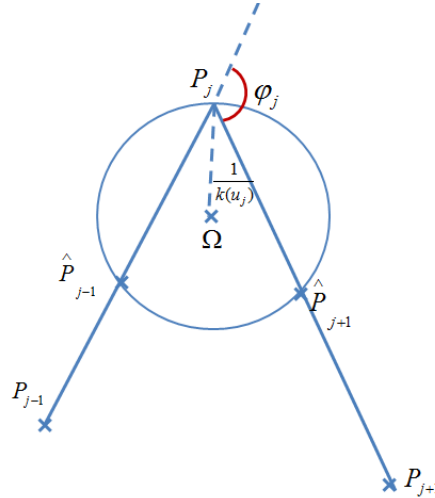


Figure 2-3: Curvature definition at parameter φ_j

This figure illustrates how the curvature of the polyline is approximated at one of its vertices. The curvature at point P_j is deduced by estimating the radius of the osculating circle from vertices P_{j-1}, P_j, P_{j+1} . As the curvature does not depend on the length of the segments $P_{j-1}P_j$, we introduce two points $\hat{P}_{j-1}, \hat{P}_{j+1}$ such that each point belongs respectively to lines $P_{j-1}P_j$ and $P_j P_{j+1}$ and that $\|\hat{P}_{j-1}P_j\| = \|\hat{P}_{j+1}P_j\| = 1$ (Guilbert *et al.*, 2006).

With approximated curvature $k(u_j)$ the internal energy (Guilbert *et al.*, 2006) is:

$$E_{int} = \frac{\beta}{2} \sum_{j=0}^J |k(u_j)|^2 \quad (6)$$

In E(4), the internal energy is represented as the sum of first derivative and second derivative, but as α is set to zero, the internal energy here is only the second term, which represents the curvature. In (6), the curvature is calculated by the approximation with k instead of the second derivative in (4).

In image processing applications, snakes are often used to match contours in the image (Kass *et al.*, 1987). The external energy term, therefore, often uses distance between the current location and some image-derived contour information. In the case of contour generalization, however, there is no definite target as the ENC contours move continuously offshore as the scale of the chart decreases. The primary constraint, therefore, is that the generalized snake should be on the seaward side of the original curve, and the external energy can be set to a one-sided function (Guilbert *et al.*, 2006),

$$E_{ext}(X(u_j)) = \begin{cases} c_0 \frac{\|X_0(u_j) - X(u_j)\|^2}{\varepsilon_{vis}^2} & \text{if } X(u_j) \text{ on the shoal side} \\ 0 & \text{otherwise } (c_0=0) \end{cases} \quad (7)$$

where $X_0(u_j)$ is on the original curve, $X(u_j)$ is on the contour generalization, and c_0 is a coefficient used in the calculation. When $X(u_j)$ is on the shoal side, c_0 is 1, but if $X(u_j)$ is not on the shoal side, c_0 is set to 0 such that there is only a penalty when the constraint is broken. The penalty term here increases according to the severity with which the generalized curve crosses to the wrong side of the original (how far it is on the wrong side), but uses a normalization term to represent the ‘minimum visualizable distance’ set according to the target scale of generalization. ε_{vis}^2 reflects the fact that lines on the chart display are non-ideal, and have a defined thickness. The

snake used here is made up of finite points. The energy of the whole snake is the summation of the energy at each point:

$$E_{total} = \sum_{j=0}^J \left(\frac{1}{2} \beta k(u_j)^2 + \frac{1}{\mathcal{E}_{vis}^2} \|X_0(u_j) - X(u_j)\|^2 \right) \quad (8)$$

Here $k(u_j)$ is the curvature at point $X(u_j)$.

When E_{total} is minimized, the curve will be moved to the desired position. The $X(u_j)$ that minimizes E_{total} are the coordinates of the vertices of the desired polyline. One way to calculate the minimum of a function is to calculate the gradient; when the gradient of E_{total} is zero, E_{total} might reach its maximum or minimum value. However, as the curvature of a curve can be infinitely large, there is no upper bound for the total energy, so there is no maximum E_{total} . That means, when the gradient is zero, E_{total} reaches a minimum (although it may be only a local minimum).

The gradient of the function is:

$$\nabla E_{total} = \nabla E_{int} + \nabla E_{ext} \quad (9)$$

that is:

$$\nabla E_{total}(u_j) = \nabla \left(\frac{1}{2} \beta k(u_j)^2 \right) + \nabla \left(\frac{\|X_0(u_j) - X(u_j)\|^2}{\mathcal{E}_{vis}^2} \right) \quad (10)$$

Here, ∇E_{int} and ∇E_{ext} are the gradient of the internal and external energy, but they can also be considered as forces on the curve. A solution of (9) can be seen either as realizing the equilibrium of the forces (Figure 2-4) in the equation or reaching the minimum of the energy (Cohen, 1991). The curve that minimizes (9) is formed by numerically approximating the gradient terms. The details of the solution are in section 2.3.3.1.

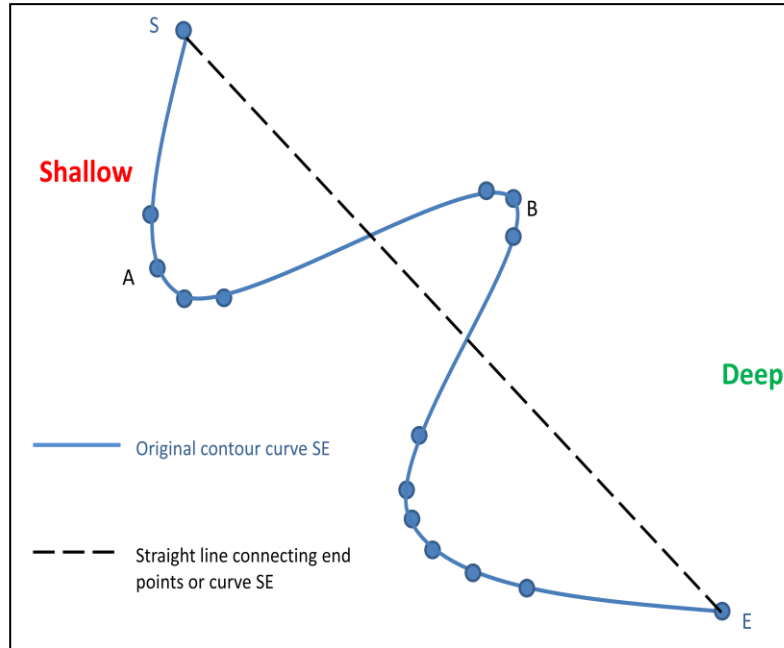


Figure 2-4a: Internal and external forces in the generalization process

In this figure, the black line is the straight line connecting the contour SE's end points. It also represents the position where the curvature is zero.

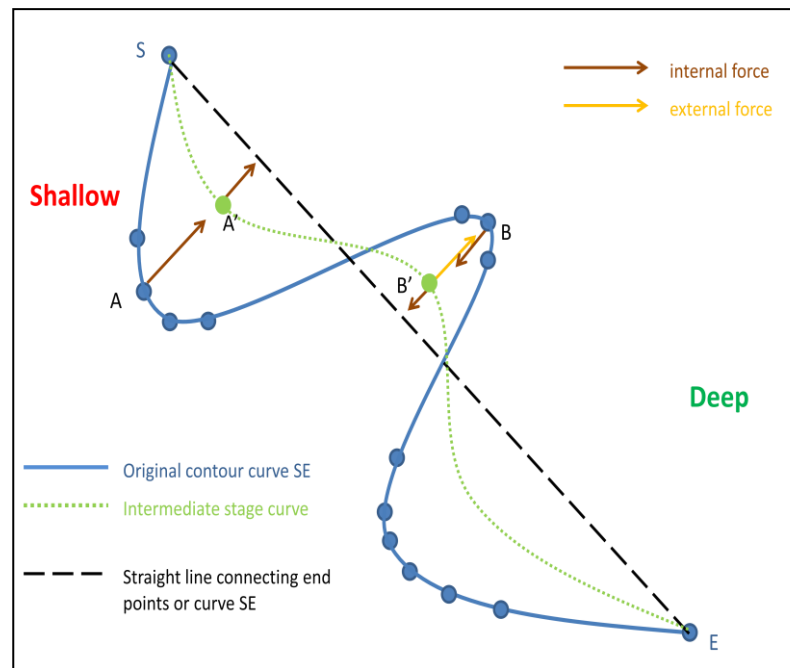


Figure 2-4b: Internal and external forces in the generalization process

At the beginning of generalization, at point A of the contour, as its curvature is larger than zero, an internal force (brown arrow) is applied on it. Point A is on the original line, so there is no external force applied on it. In the intermediate stage of generalization, point A moves to A', as A' is on the same side of the zero curvature line, the internal force keeps its direction. For point

B, at the beginning, as its curvature is very large, so the internal force is applied, and there is no external force, so it is moving towards the black dashed line. But during the generalization, as other parts of the contour have all moved (the green dashed line), the curvature at point B' is smaller, and at this time, the external energy is relatively larger, so point B' is then moved towards the deeper side of the original contour.

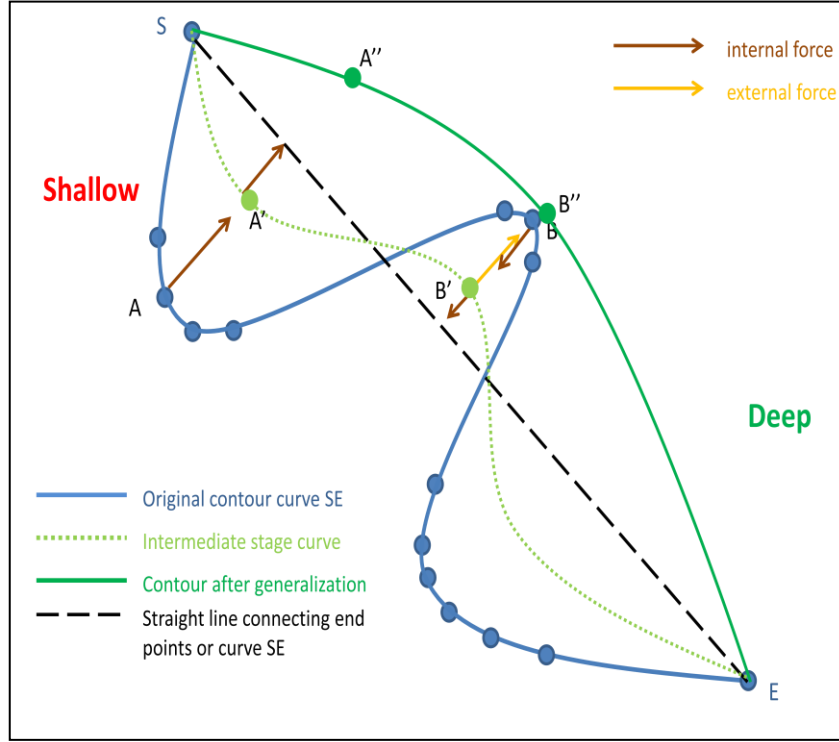


Figure 2-4c: Internal and external forces in the generalization process

Eventually, the curve stops at the green solid line where the total forces are all balanced, and the total energy is minimized. The green dashed line in Figure 2-4b and Figure 2-4c is a hypothetical line, which this generalization will never get to due to the effect of the external forces.

2.3.2 B-spline Snake Energy Terms for Polygons

The internal energy term for polygons is the same as the polyline term (6). For polygon features, however, there will be an exaggeration operator in the generalization process. For the exaggeration operator, another force term is introduced in the energy equation. This new force (Cohen and Cohen, 1993) is represented as:

$$F_{balloon} = b\vec{n}(u_j)$$

(11)

The $\vec{n}(u_j)$ is the unit vector normal to the curve at point $X(u)$, and b is the amplitude of this force (Cohen and Cohen, 1993). This term generates a pressure force to push the polygon curve outward, as if air is introduced inside; the curve will be inflated like a balloon, so this force is called a balloon force. This new term gives the polygon curve a force to expand when it is exaggerated during the generalization.

By adding the new balloon force to the external energy, the total force becomes:

$$\nabla E_{\text{int}} + \nabla E_{\text{ext}} + F_{\text{balloon}} \quad (12)$$

which is represented as:

$$\nabla \left(\frac{1}{2} \beta k(u_j)^2 \right) + \nabla \left(\frac{\|X_0(u_j) - X(u_j)\|^2}{\varepsilon_{\text{vis}}^2} \right) + b \vec{n}(u_j) \quad (13)$$

2.3.3 Smoothing Operator

The smoothing operator has been studied extensively. The most commonly used smoothing method is to apply filters on polylines. Figure 2-5 illustrates how the smoothing operator works on a polyline feature. Figure 2-6 and Figure 2-7 shows an example of smoothing in a cartographer's manual process of generalization.

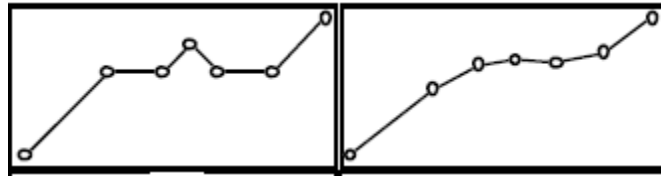


Figure 2-5: Sample spatial transformation of smoothing operator

The left sub-figure is the sample polyline; the right sub-figure is the result after smoothing operator is applied. The number of the total vertices of the curve remains the same, but the positions of these vertices are changed, such that the curve is smoothed (Shea and McMaster, 1989).

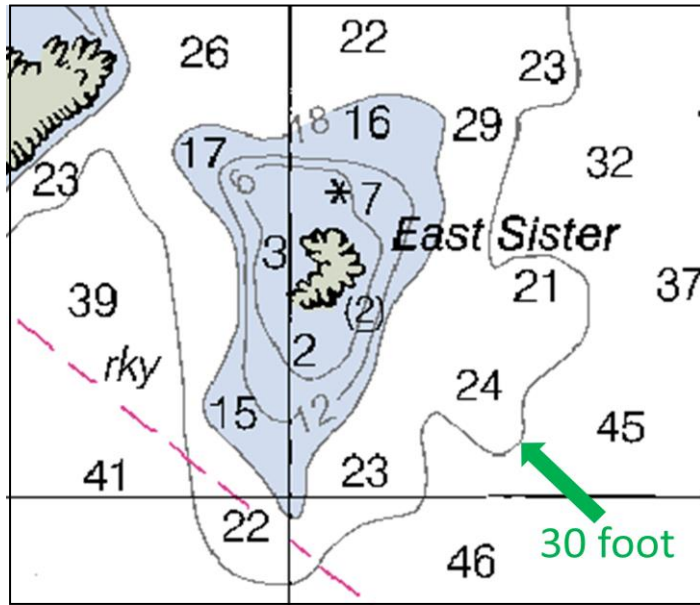


Figure 2-6: Sample contour line from 1:20,000 scale raster chart

The grey contour line pointed to by the green arrow is the 30 foot contour line from raster chart 13283 of Portsmouth Harbor, NH. This grey contour has a relatively complex shape.



Figure 2-7: Sample contour line from 1:40,000 scale raster chart

The contour line pointed to by the green arrow is the 30 foot contour line from raster chart 13274 of Portsmouth Harbor, NH. Compared to the 30 foot contour in Figure 2-6, the 30 foot contour in this figure has fewer details, and is smoother.

2.3.3.1 Smoothing Operator Implementation

The smoothing operator is implemented by calculating the gradient of internal energy, which is:

$$\nabla E_{int} = \beta k(u_j) \nabla k((x_i, y_i)) \quad (14)$$

where (x_i, y_i) is a vertex of the polyline

$\nabla k((x_i, y_i))$ is not calculated at the end points, only at the internal points :

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \equiv \text{polygonal line vertices, } (1 \leq i \leq m-1)$$

The gradient
$$\nabla k = \left[\frac{\partial k}{\partial x_1}, \frac{\partial k}{\partial y_1}, \dots, \frac{\partial k}{\partial x_{m-1}}, \frac{\partial k}{\partial y_{m-1}} \right],$$

For the curvature at vertex i , $k_i \equiv k(x_i, y_i)$, the derivatives are approximated by the central difference method :

$$\begin{aligned} \frac{\partial k}{\partial x_i} &\approx \frac{1}{2} \left(\frac{k_i - k_{i-1}}{x_i - x_{i-1}} + \frac{k_{i+1} - k_i}{x_{i+1} - x_i} \right) \\ \frac{\partial k}{\partial y_i} &\approx \frac{1}{2} \left(\frac{k_i - k_{i-1}}{y_i - y_{i-1}} + \frac{k_{i+1} - k_i}{y_{i+1} - y_i} \right) \end{aligned} \quad (15)$$

This gradient approximation is used for the internal energy in each iterative step (the gradient of external energy will be discussed in section 2.3.5.2)

2.3.4 Simplification Operator

There is research on simplification operators in previous studies. For example, the Douglas-Peucker algorithm (illustrated in section 1.3; 1973) is the most widely used algorithm to simplify linear features. Other researches prefer the bend detect algorithm developed by Wang (1998). However, both these method do not consider the shoal-bias rule of nautical chart features,

and they cannot be used in this study. Figure 2-8 illustrates how the simplification operator works on a polyline.

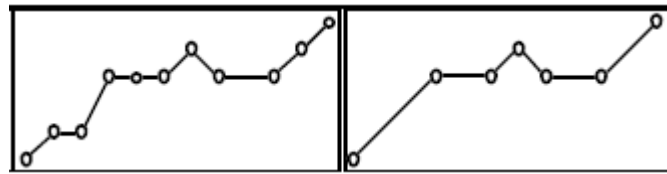


Figure 2-8: Sample spatial transformation of simplification operator

The figure on the left side is the sample polyline, the figure on the right side is the result after the simplification operator is applied. The number of points is reduced after the simplification operator is applied, and the new line contains fewer details than the previous line. (Shea and McMaster, 1989).

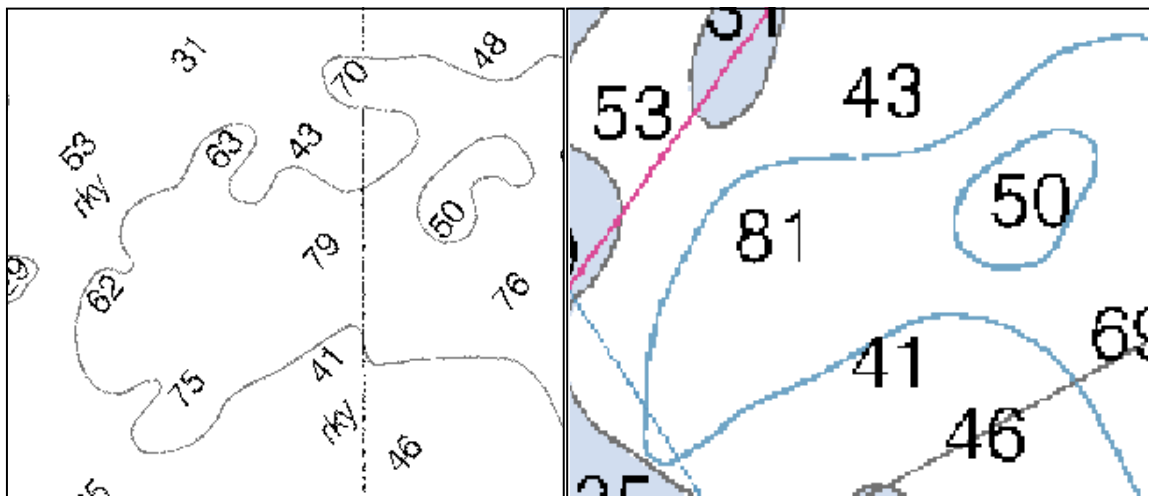


Figure 2-9: Sample of contours in 1:40,000 scale and 1:80,000 scale raster chart

The grey contour on the left sub-figure is a 60 foot polyline contour on the 1:40,000 scale raster chart, 13274, of Portsmouth Harbor, NH. The blue polyline contour on the right sub-figure is the same 60 foot contour on the 1:80,000 scale raster chart, 13278, of Portsmouth Harbor, NH. The blue polyline contains fewer details and is more simplified than the grey contour on the left.

2.3.4.1 Simplification Operator Implementation

The simplification operator can be implemented by two methods. One is to reduce the number of control points of the B-spline curve before the generalization process at the data preprocess step, the other method is to reduce the number of points on the polyline.

B-spline approximation is the first step of the generalization workflow. In Chapter II,

section 1, a B-spline is defined by control points and basis functions:

$$f(u) = \sum_{i=0}^m Q_i N_i^k(u) \quad (18)$$

Q_i are the control points and u is a series of real values are called parameters or the knot vector ($u_0 = a \leq u_1 \leq \dots \leq u_i \leq \dots \leq u_{m+k} = b$). When approximating a polyline with a B-spline, the number of the B-spline parameters is the same as the number of data points on the polyline, but the number of control points can be smaller than the number of parameters. As the basis function can be calculated recursively, the only unknowns of each B-spline are the locations of the control points. To approximate the original contour curve with a B-spline curve, instead of storing all the data of original polyline, only the control points need to be stored (Saux and Daniel, 1998).

The second method is to reduce the number of data points in a polyline. This method has been used during the generalization process in this thesis. When neighbor points are closer than a threshold during deformation, points will be deleted. The pseudo code is as follows:

Algorithm 2-1:

Input: one polyline (a set of control points of an approximating B-spline)

1. Calculate the distance between adjacent control points
2. Select all the indices for which the neighbor distance (Cartesian distance between two adjacent vertices) is smaller than a threshold
3. Iterate through the selected indices of step 2
 - 3.1 If there are three or more continuous indices in the selected indices (like index 4, 5, 6 and 7)
 - 3.1.1 Divide this continuous segment into small segments such that each contains three continuous indexes
 - 3.1.2 For each segment

3.1.2.1 Keep the first point; delete the second and third point

3.1.3 End for

3.2 End If

4. End Iterate

Note: The rational for using three points in step 3.1 is illustrated in Appendix A.6.

2.3.5 Shoal-biased Operator

The shoal-biased rule is a special generalization rule for nautical charts features. As introduced in the previous section, contours of a nautical chart can only be moved to a deeper position (Figure 2-10). This shoal-biased constraint operator has always been used together with other operators like the smoothing operator.

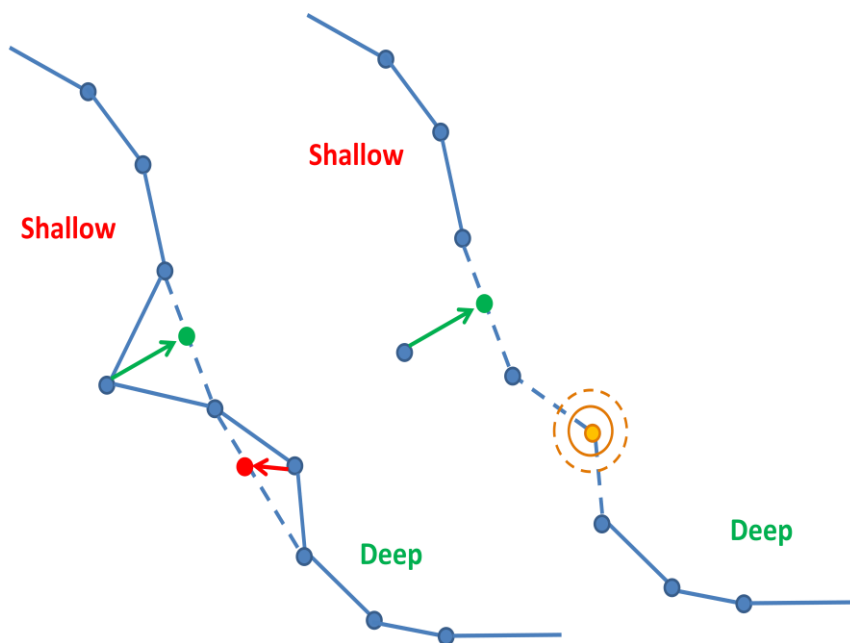


Figure 2-10: Example of shoal-biased principle

The blue lines are the example contour lines, the blue dots on them are the vertices. The red and green arrows point out the direction of the movement of the vertices during the generalization process. The red arrow in the left side means that vertex cannot move to the shallower side. It should stay in its current position as the yellow point and yellow circle in the right side shows. The green arrows in both lines means the vertex can move to the deeper side. The vertex in orange with orange dash line circle around it means that this vertex must stay stationary if there are forces attempt to move it to the shallower side..

2.3.5.1 Shoal-biased Operator Method

The shoal-biased operator has been implemented as an external energy in the total energy equation. By calculating the gradient of external energy, the shoal-biased operator is implemented. As in section 2.3.1, this external energy is defined as:

$$E_{ext}(u_j) = \begin{cases} \frac{\|X_0(u_j) - X(u_j)\|^2}{\epsilon_{vis}^2} & \text{if } X(u_j) \text{ on the shoal side} \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

Where $X_0(u_j)$ is the original curve. Adding this energy term in the total energy ensures that an affected vertex will always be maintained on the original curve or move to the deeper side. The details of the calculation are in the next section.

2.3.5.2 Shoal-biased Operator Implementation

Using the notation $X_0(u_j) \equiv (x_{0j}, y_{0j})$ and $X(u_j) \equiv (x_j, y_j)$, (19) can be rewritten as:

$$E_{ext}(u_j) = \frac{(x_j - x_{0j})^2 + (y_j - y_{0j})^2}{\epsilon_{vis}^2} \quad (20)$$

So the gradient of external energy $\nabla E_{ext}(u_j) = \left(\frac{\partial E_{ext}(u_j)}{\partial x}, \frac{\partial E_{ext}(u_j)}{\partial y} \right)$ is

$$\frac{\partial E_{ext}(u_j)}{\partial x} = \frac{2}{\epsilon_{vis}^2} (x_j - x_{0j}) \quad (21)$$

$$\frac{\partial E_{ext}(u_j)}{\partial y} = \frac{2}{\epsilon_{vis}^2} (y_j - y_{0j}) \quad (22)$$

By adding these terms, the new point will obey the shoal-biased rule.

Now defining a vector of all vertex points on the polyline at step n:

$$\vec{X}^n \equiv \left[(x_1^n, y_1^n) \cdots (x_m^n, y_m^n) \right], \quad (0 \leq n \equiv \text{step number})$$

with $(x_j^0, y_j^0) \equiv (x_{0j}, y_{0j}) \equiv X_0(u_j)$, the initial contour,

the iterative step is $\vec{X}_{\text{new guess}}^{n+1} = \vec{X}_{\text{old guess}}^n - \nabla E_{\text{int}} - \nabla E_{\text{ext}}$

In this thesis, the iteration stops at a point when the input contours are generalized to a very simplified line. For different input contours, the iteration number varies, but it is generally a number larger than 1000. In section 5.2 of future study, there is more discussion of the iteration number.

CHAPTER III

OPERATORS, AUXILIARY FUNCTIONS AND WORKFLOW DESIGN

In this thesis, generalization is being done by using operators; each operator has its specific generalization purpose. By combining different operators together in particular workflows, a generalization process can be achieved with respect to holistic and aesthetic purposes. The first part of this chapter introduces all of the generalization operators' purposes and how they are implemented; the second part of this chapter is about the workflow for how these operators are combined in the overall generalization process. The operators are defined phenomenologically, and so the best way to describe them is through pseudo code. The unit of the distance values used in the following calculations is 1/100,000 degree (see Appendix A.0).

3.1 Operators

3.1.1 Aggregate Operator

An aggregate operator is used to combine two or more features into one. As shown in Figure 3-1, three small Ruins features in the left figure are aggregated into two large features in the right figure.



Figure 3-1: Sample spatial transformation of aggregate operator

In the left sub-figure, there are six polygon features representing ruins; in the right sub-figure, there are just two figures representing the ruins, each one of these two polygons represents three small ruins in the left figure. This aggregation process provides more space on the map to clearly draw the features, but still shows the two distinct group of ruins from the original (Shea and McMaster, 1989).

The features being aggregated should be in the same category. In this research, the research objects are just contours, but they are at various depths. Only contours with the same depth can be aggregated.

In cartographers' manual process of generalization, aggregations are done in many circumstances. As Figure 3-2 to Figure 3-4 demonstrate, aggregation is done between polylines and polygon contours, two polygon contours, and a group of polygons.

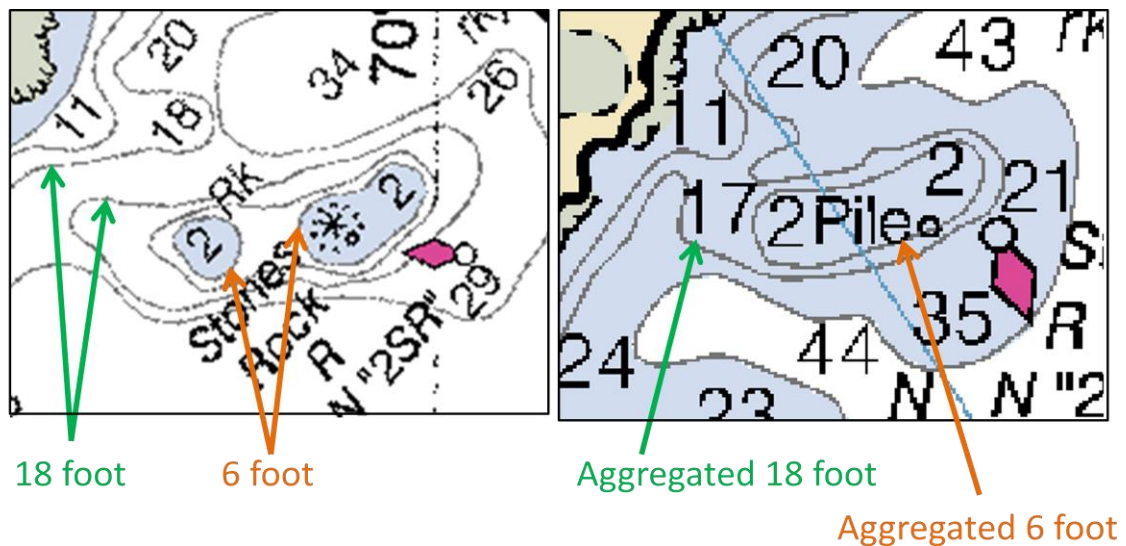


Figure 3-2: Sample of aggregation of a polyline and polygons

The left sub-figure is a selected area of the 1:40,000 scale raster chart, 13274; the right sub-figure is the same area of the 1:80,000 scale raster chart, 13286. In the left sub-figure, there is an 18 foot polyline contour and also an 18 foot polygon contour (indicated by green arrows), which is the outside-most contour of the concentric polygon group. In the right sub-figure, these two contours are aggregated into one long complex polyline contour (green arrow), which still has the depth value of 18 feet. In the left sub-figure, there are two six foot polygon contours (indicated by yellow arrows), in the right sub-figure, they are aggregated into one large six foot contour (indicated by the orange arrow).

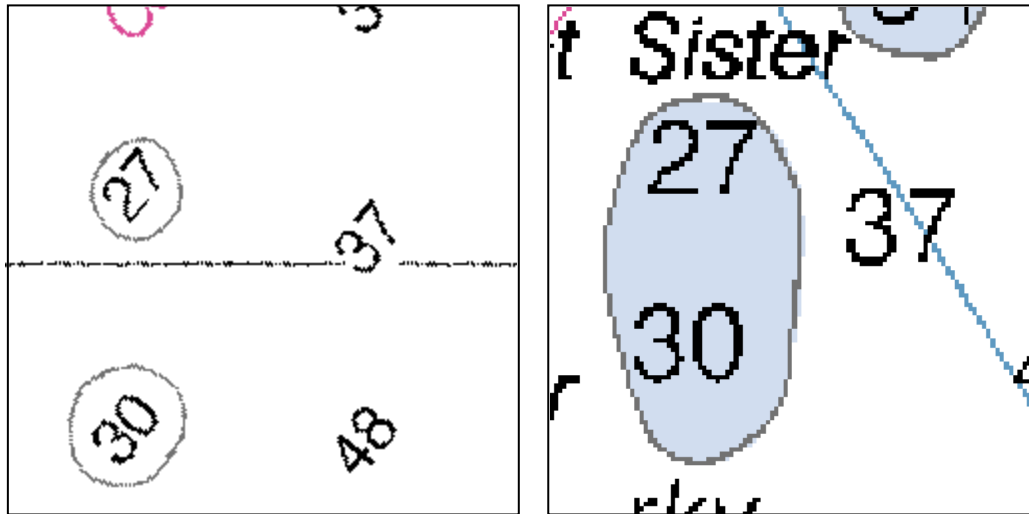


Figure 3-3: Sample of aggregation of two polygons

The left sub-figure is a selected area from the 1:40,000 scale raster chart, 13274; the right sub-figure is the same area from the 1:80,000 scale raster chart, 13286. In the left sub-figure, there are two 30 foot polygon contours; in the right sub-figure, these two polygons are aggregated into one large 30 foot polygon contour.

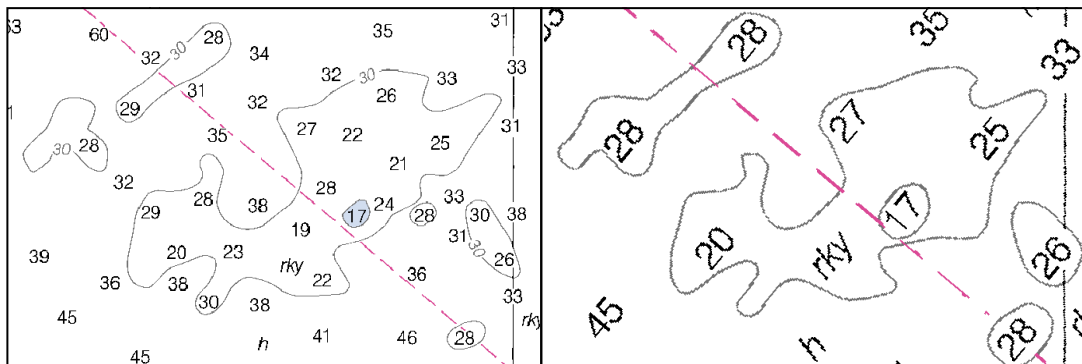


Figure 3-4: Sample of aggregation of a group of simple polygon contours

The left sub-figure is a selected area from the 1:20,000 scale raster chart, 13283; the right sub-figure is the same area of the 1:40,000 scale raster chart, 13274. In the left sub-figure, there are six polygon contours with a depth of 30 feet. In the right sub-figure, there are just four polygon contours of 30 feet. Four of these contours are aggregated into two larger polygons.

3.1.1.1 Aggregate Operator Method

There are four cases for the aggregate operator. The first is aggregating a polyline with a polygon contour (Figure 3-5, One). The second case is aggregating two polygon contours (Figure 3-5, Two). The third case is aggregating two groups of contours, where each group has a set of

polygon contours inside it (Figure 3-5, Three). The fourth case is aggregating two contours when they have intersected with each other (Figure 3-5, Four). In this last case, the intersected contour can be either a polygon or a polyline. This case may happen during the generalization process when the step size of the last generalization was too large: the polyline moved a large step, and intersected with the neighbor feature. The biggest difference between the first two cases is the aggregation result. For Case One, the aggregated result is a polyline, but for the second case the aggregation result is a polygon. The conditions are treated in order of complexity; the first and second conditions are relatively simple, the third condition has more steps and is more complicated.

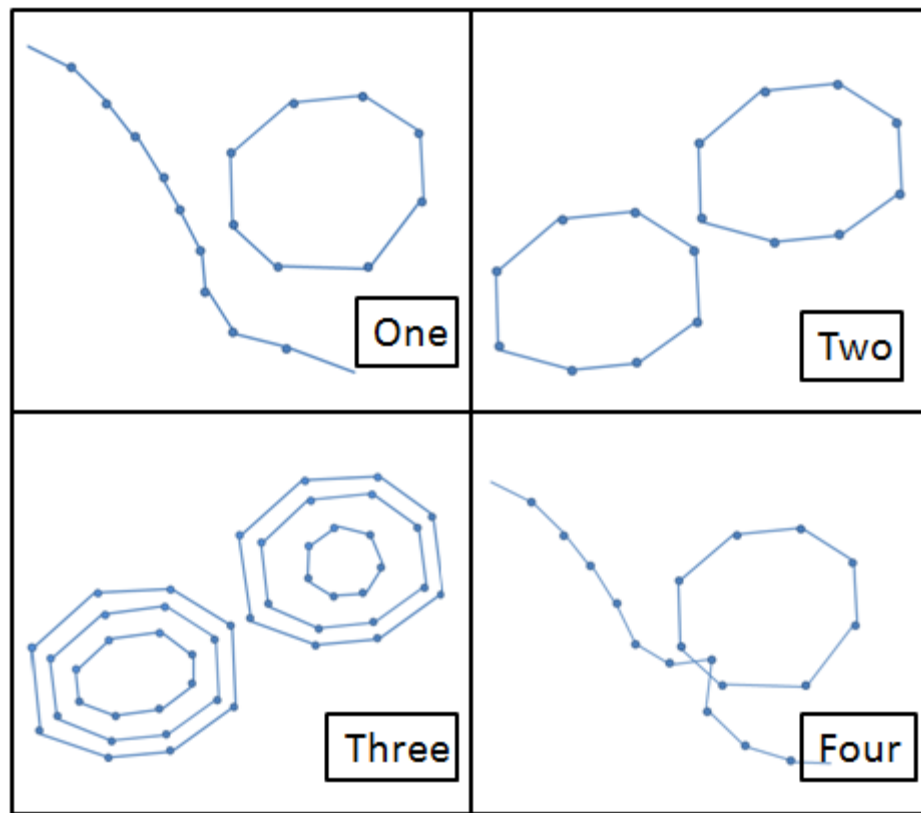


Figure 3-5: The four cases of the aggregate operator

Case One: one polyline and one polygon; Case Two: two polygons; Case Three: two polygon groups; Case Four: aggregation when two contours intersect. These four cases represent different possible conditions in aggregation; the detail explanation of how each case is implemented is in section 3.1.1.2.

3.1.1.2 Aggregate Operator Implementation

3.1.1.2.1 Aggregate Operator Case One Implementation

The basic goal of the aggregate operator is to find two supporting segments that connect the two features, and then to remove the segments of the two features between the contours. For example, the brown lines in Figure 3-6A are the supporting lines for these two features; the green line in Figure 4-6B is the aggregated result of those two features.

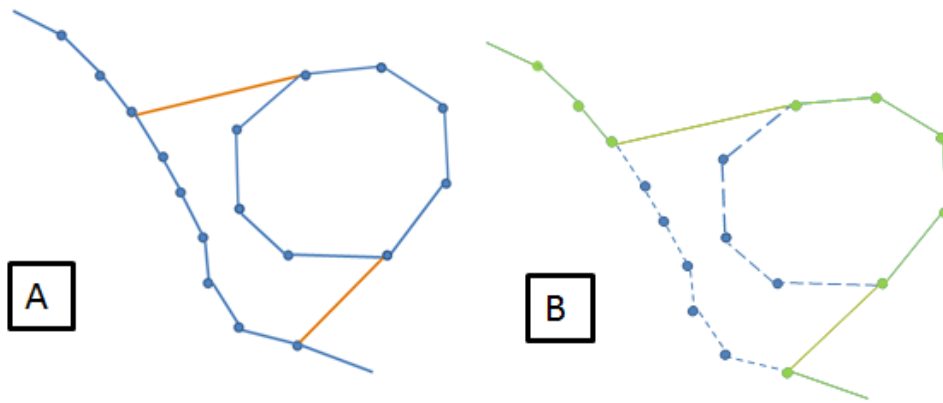


Figure 3-6: Two steps of aggregate operator in Case One

A: find two supporting segments; B: connect the remaining part of the two features and the supporting segments. The aggregate operator here finds the supporting segments (the brown lines in the left figure) and combines them with the selected part of polyline and polygon, and forms a new feature (the green line in the right figure). The dashed lines will be deleted.

Pseudo code for aggregating a polyline and a polygon is as follows:

Algorithm 3-1

Input: one polyline and one polygon contour

1. Calculate the minimum distance between the polyline and polygon
2. If the minimum distance is smaller than a threshold
 - 2.1 Find the proper supporting segments for the polyline-polygon case
 - 2.2 Combine the supporting segments with the rest of the features
3. End If

Pseudo code for finding supporting segments is as follows:

Algorithm 3-2

Input: one polyline and one polygon contour.

1. Find two points $P_{polygon}$ and P_{line} (Figure 3-7) of the polygon and polyline that are closest to each other.
2. Get the starting index S_1 and ending index E_1 of the selected section of the polygon.
 S_1 's index is the index of $P_{polygon}$ plus N, E_1 's index is the index of $P_{polygon}$ minus M (point index increases clockwise from an arbitrary point, and should be considered modeling the number of points in the polygon).
3. Get the starting index S_2 and ending index E_2 of the selected section of the polyline. Calculate the minimum distance from each vertex of the polyline to the polygon. The vertex which has minimum distance smaller than a threshold will be selected. For example in Figure 3-7, the segment from index S_2 to index E_2 is the selected segment of the polyline. Then line S_1S_2 and line E_1E_2 will be the supporting segments of this polyline and polygon.
4. Check if line S_1S_2 and line E_1E_2 intersect with the original polygon contour. If they do, the point closet to the intersection point will be the new starting or ending point of the polygon contour, and the algorithm will use the new starting and ending point as the S_1 and E_1 index (Figure 3-8). Then the new line S_1S_2 and E_1E_2 will be the valid supporting lines.
5. Connect the valid supporting lines and remaining segments of the polyline and the polygon. Delete the original polyline and polygon. This new polyline will be the aggregated result (Figure 3-9).

Note: The value of N and M are discussed in Appendix A.1.

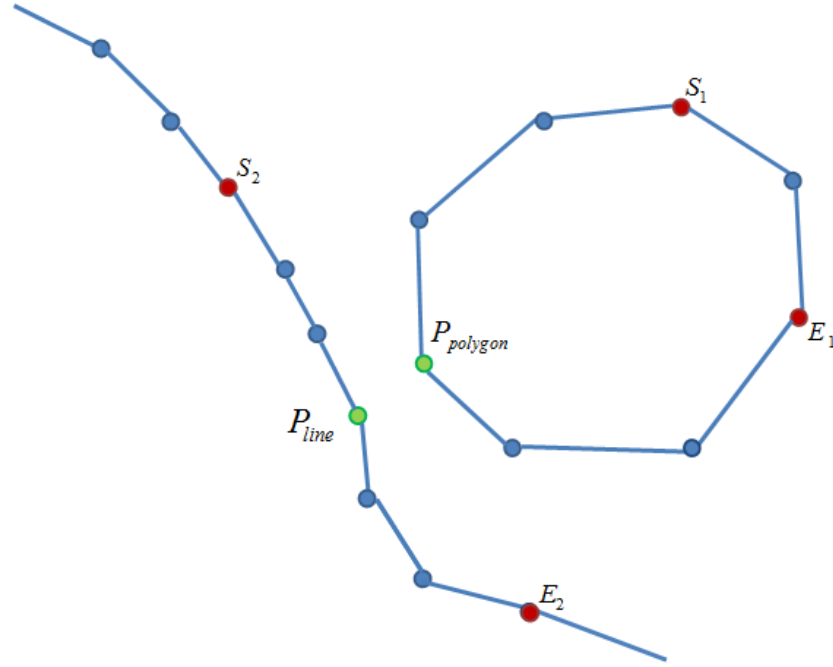


Figure 3-7: Finding the starting and ending index for the selected segment of the polyline and polygon

S_1 is the starting index and E_1 is the ending index of the selected segment of the polygon;
 S_2 is the starting index and E_2 is the ending index of the selected segment of the polyline.
This step finds the starting and ending indices of the selected segment.

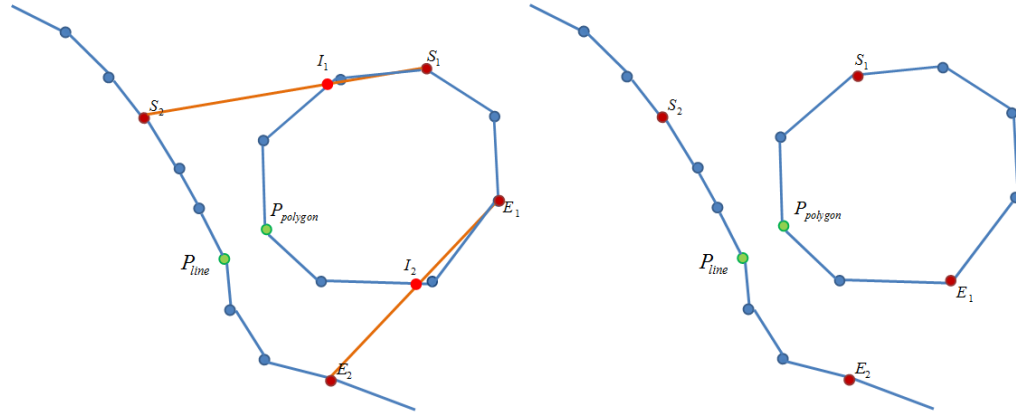


Figure 3-8: Check if line S_1S_2 and line E_1E_2 intersect with the polygon

If the line S_1S_2 and line E_1E_2 intersect with the polygon (left), calculate the intersection points I_1 and I_2 , then (right) the new starting index S_1 and ending index E_1 will be shifted to the closest index to the intersection points.

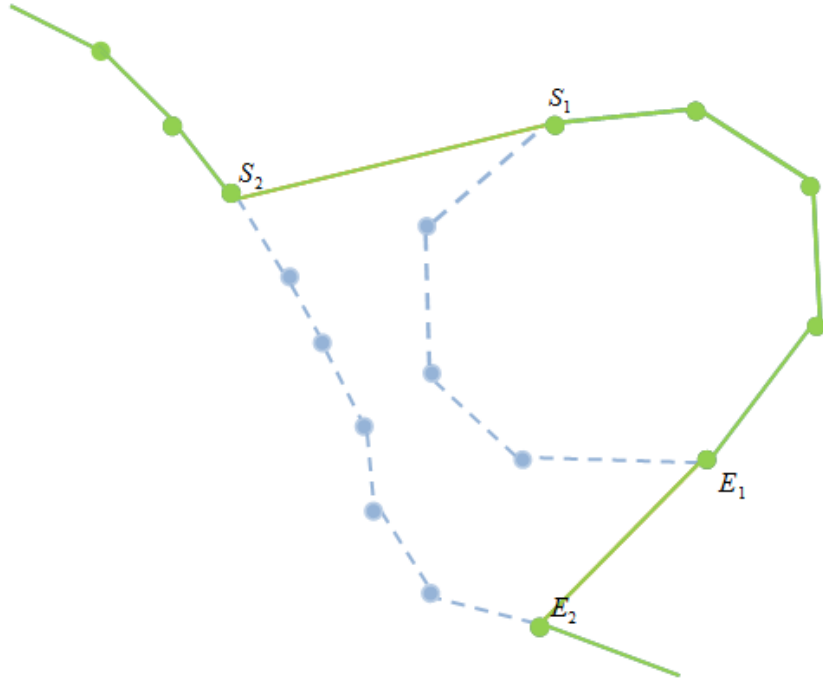


Figure 3-9: Aggregate operator Case One result

The remaining segments of the polyline are all the parts of the polyline except the section from index S_2 to index E_2 . The remaining segment of the polygon is from index S_1 to index E_1 . The green line is the aggregated result.

3.1.1.2.2 Aggregate Operator Case Two Implementation

The second case is to aggregate two polygon contours (Figure 3-10). The method is similar to Case One, but as there are two polygon features, the intersection checking steps should be used for both polygons. In addition, another constraint is added to the supporting segment: the angle formed by the supporting line and the neighbor segment of either polygon should be obtuse (Figure 3-11). If that angle is acute, there will be numerical blunders in the subsequent calculation of the energy equation.

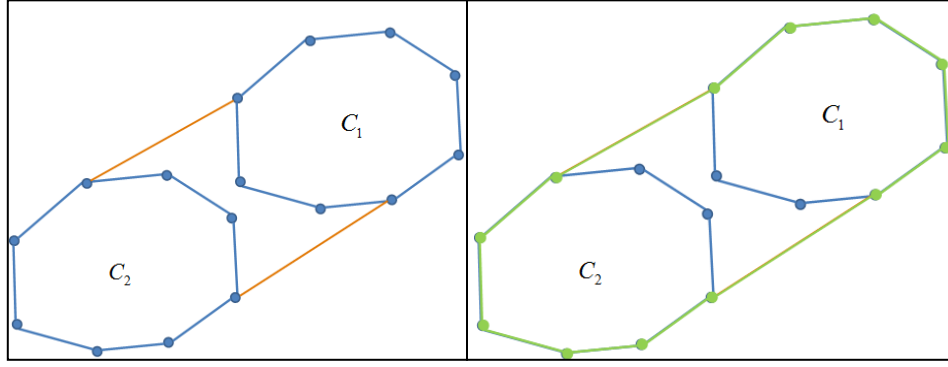


Figure 3-10: Aggregation Case Two

The left sub-figure shows the first step: find two supporting segments; the right sub-figure shows the second step: connecting the remaining part of the two features and the supporting lines.

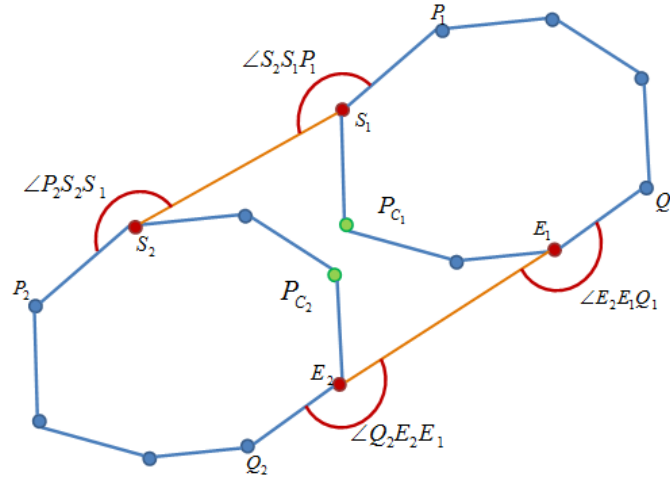


Figure 3-11: Angles formed by the supporting line and neighbor segment should be larger than 90 degrees

The points S_2 and E_2 are the starting and ending indices of the selected segment of the polygon on the left side, points S_1 and E_1 are the starting and ending indices of the selected segment of the polygon on the right side. P_2, P_1, Q_2 , and Q_1 are the neighbor points of S_2, S_1, E_2 , and E_1 respectively. Angles $\angle S_2S_1P_1, \angle P_2S_2S_1, \angle Q_2E_2E_1, \angle E_2E_1Q_1$ should all be larger than 90 degrees. P_{c_1} and P_{c_2} are the two closet points on polygon C_1 and C_2 .

Pseudo code for aggregating two polygons is as follows:

Algorithm 3-3

Input: two polygon contours.

1. Calculate the minimum distance between the two polygons.
2. If the minimum distance is smaller than a threshold
 - 2.1 Find the proper supporting segments (one on each polygon) for the two polygons condition (Algorithm 3-4).
 - 2.2 Combine the supporting lines with the remainder of the points from each polygon.
3. End If

Note: The value of the threshold is discussed in Appendix A.8.

Pseudo code for finding supporting segments for the two polygons is as follows:

Algorithm 3-4

Input: two polygon contours.

1. Find the two points P_{C_1} and P_{C_2} of the polygon on the right side and the polygon on the left side that are closest to each other by calculating distances between all points in the two polygons.
2. Find the starting point S_2 and ending point E_2 of the selected section of the left side polygon C_2 .
 - 2.1 If the total number of vertices of the left polygon is larger than 20
 - 2.1.1 S_2 's index is the index of P_{C_2} minus a constant value N_1 , E_2 's index is the index of P_{C_2} plus N_1 (point index increases clockwise from an arbitrary point).

2.2 Elseif the total number of vertices is less than or equal to 20

2.2.1 S_2 's index is the index of P_{C_2} minus a constant value N_2 , E_2 's

index is the index of P_{C_2} plus N_2

2.3 End If

3. Repeat step 2 for polygon C_2

4. If line S_1S_2 or line E_1E_2 intersect with the left polygon, replace point S_2 or E_2 with the point on the left polygon nearest the intersection points.

5. If the updated line S_1S_2 or line E_1E_2 intersect with the right polygon, replace point S_1 or E_1 with the points that are closest to the intersection points.

6. If angle $\angle S_2S_1P_1$ or $\angle P_2S_2S_1$ is acute

6.1 While any of $\angle S_2S_1P_1$ or $\angle P_2S_2S_1$ is acute

6.1.1 If $\angle S_2S_1P_1$ is acute, shift point S_1 by one index clockwise.

6.1.2 If $\angle P_2S_2S_1$ is acute, shift point S_2 by one index counter clockwise.

6.2 End While

7. If angle $\angle E_2E_1Q_1$ or $\angle Q_2E_2E_1$ is acute

7.1 While any of $\angle E_2E_1Q_1$ or $\angle Q_2E_2E_1$ is acute

7.1.1 if $\angle E_2E_1Q_1$ is acute, shift point E_1 by one index counter clockwise,

7.1.2 if $\angle Q_2E_2E_1$ is acute, shift point E_2 by one index clockwise.

7.2 End While

8. Repeat step 4-7 until connecting lines are valid ($\angle S_2 S_1 P_1$, $\angle P_2 S_2 S_1$, $\angle E_2 E_1 Q_1$ and $\angle Q_2 E_2 E_1$ are larger than 90 degree, and $S_2 S_1$, $E_2 E_1$ are not intersecting with either polygon).
9. Connect the valid supporting lines and remaining segments of the two polygons. Delete the original two polygons. This new larger polygon will be the aggregated result (Figure 3-12).

Note: The reasons for the values of N_1 and N_2 , and why the value of 20 were chosen are discussed in Appendix A.2.

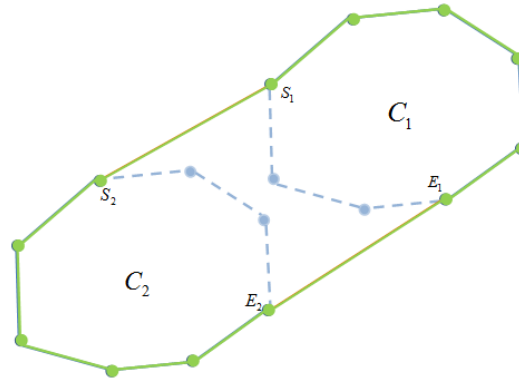


Figure 3-12: Aggregate operator Case Two result

The remaining segment of polygon C_2 is from point S_2 counter clockwise to E_2 . The remaining segment of polygon C_1 is from point S_1 clockwise to point E_1 . The green line is the aggregated result of Case Two.

3.1.1.2.3 Aggregate Operator Case Three Implementation

Case Three is similar to Case Two except that the sequence of desired contours here must be considered as a group and processed in sequence. The pseudo-code is:

Algorithm 3-5

Input: two groups of polygon contours G_1 and G_2 , each group with several contours inside each other.

1. Select the current target features as the outer-most polygons of each group.

2. While there are more polygon contours inside the current aggregated feature
 - 2.1 Aggregate the most exterior polygon of each group with the method of Case Two, and find a new aggregate polygon (green circle in the right figure of Figure 3-13).
 - 2.2 Change the current target polygon contour to the interior neighbor polygon contours of the previously aggregated polygons (Figure 3-14 and Figure 3-15).
- 3 End While

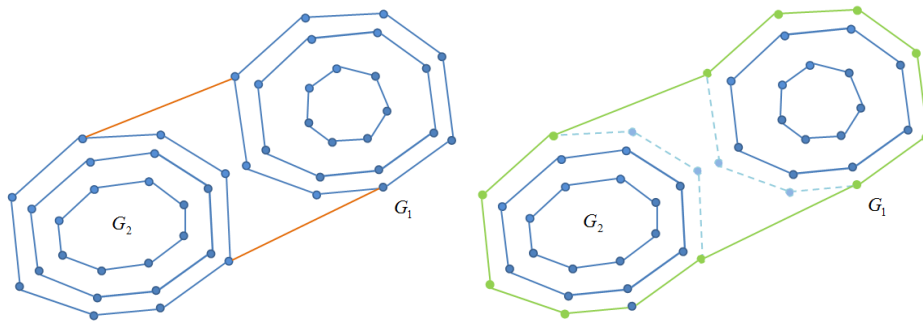


Figure 3-13: Aggregate operator Case Three step one
Start with the most exterior two polygon contours; apply the same aggregation method as in Case Two.

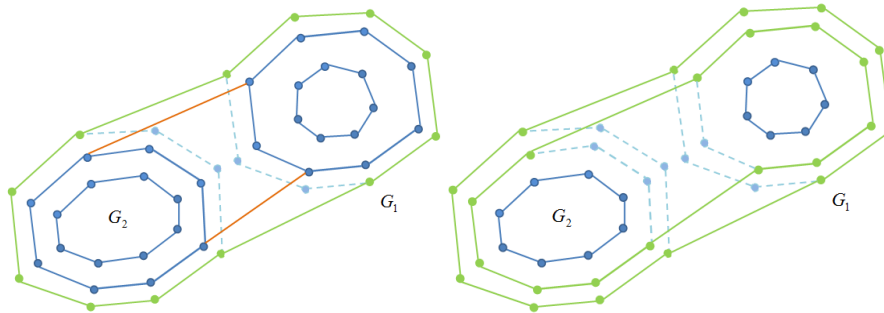


Figure 3-14: Aggregate operator Case Three step two
Process the inner polygon contours, using the method of Case Two.

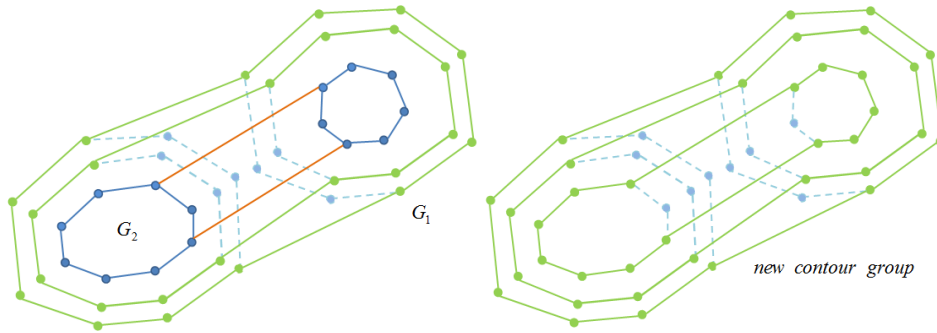


Figure 3-15: Aggregate operator Case Three result

Finally, when there are no more polygon contours inside the current target polygon, the aggregation ends. The new contour group is the aggregate result of this case.

3.1.1.2.4 Aggregate Operator Case Four Implementation

Case Four is when two polygons have already intersected with each other (Figure 3-16). The algorithm attempts to avoid this, but this condition can occur when, before exaggeration, the distance between two polygons is larger than the minimum distance (defined in (19)), this distance maintains suitable distance between two lines that human eyes can distinguish), but, after one exaggeration, the exaggeration step-size might be larger than their previous distance at some vertices, causing the two polygons to intersect. The method to solve this situation is similar to Case Two, but instead of finding the closest vertices, the intersected vertices are found:

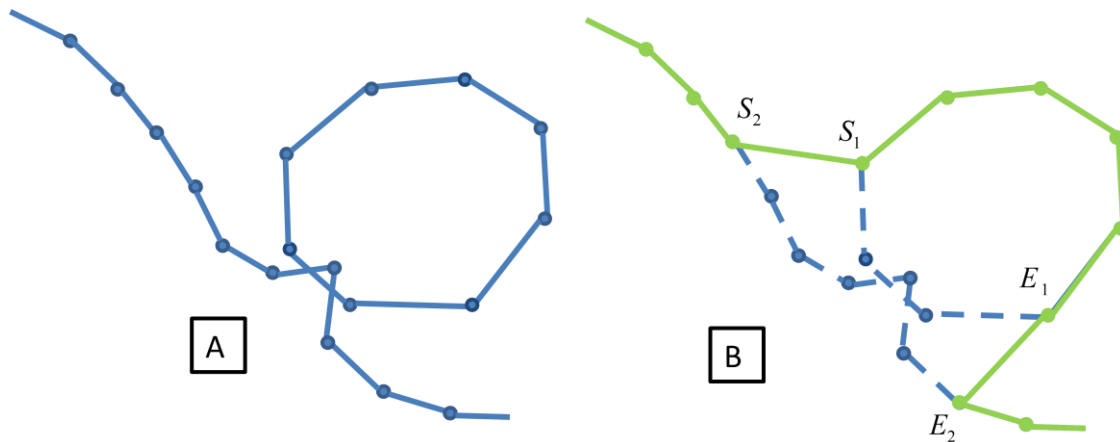


Figure 3-16: Two steps of aggregate operator in Case Four

A: find two supporting segments; B: connecting the remaining part of the two features and the supporting segments. The green polyline is the aggregated result. The dashed lines are the deleted segments. This case is similar to the Case One. Only the step of finding supporting segments is different.

Pseudo code for aggregating polyline and polygons when they intersect is as follows:

Algorithm 3-6

Input: two contours.

1. Detect the intersection points of the two contours. Use the end vertices of the segment where the intersection point at as the point with minimum distance as defined in the previous three cases (points S_2 , S_1 , E_2 , and E_1 in Figure 3-16).
2. If the input data are one polyline and one polygon
 - 2.1 Find the proper supporting segments (segment S_1S_2 in Figure 3-16) for the polyline-polygon condition using Algorithm 3-2.
 - 2.2 Combine the supporting lines with the rest of the features (the green line covered polygon in Figure 3-16) using Algorithm 3-2.
3. Elseif the input data are two polygon contours
 - 3.1 Find the proper supporting segments for the polygon-polygon condition using Algorithm 3-2.
 - 3.2 Combined the supporting lines for the polygon-polygon condition using Algorithm 3-2.
4. End If

3.1.1.2.5 Two Polylines

In this thesis, when two polylines are within a minimum distance or intersect, an algorithm is used to make sure that they will not be aggregated. Instead, one of them will be deleted or they will be forced to separate, thereby maintaining a minimum distance between these two polylines. This “minimum distance maintain” method is illustrated in Algorithm 3-7.

3.1.2 Exaggerate Operator

Figure 3-17 shows a bay inlet which is exaggerated during generalization. If it is not exaggerated, this feature will disappear when the scale becomes smaller. As this feature is significant to the map user, it should be maintained.

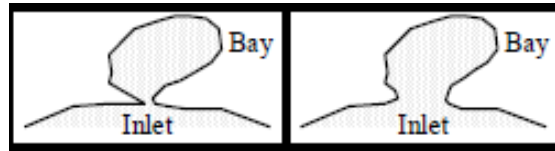


Figure 3-17: Sample spatial transformation of exaggerate operator

In the left sub-figure the entrance of the inlet is relatively narrow; in the right sub-figure, the entrance is enlarged, such that it is not closed if the whole contour shrinks (Shea and McMaster, 1989).

In manual generalization, exaggeration is a tool that is used to maintain significant features.

Figures 3-18 to 3-20 show examples of cartographic use of exaggeration.

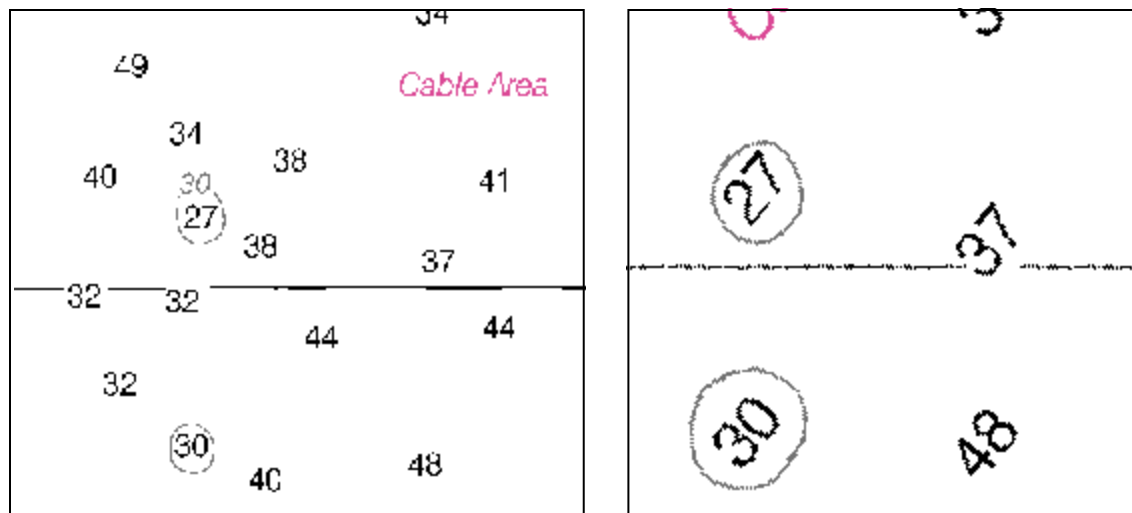


Figure 3-18a: Sample exaggeration of simple polygon contour

The left sub-figure is a selected area from the 1:20,000 scale raster chart, 13283, the right sub-figure is the same area from the 1:40,000 scale raster chart, 13274. In the left sub-figure, there are two 30 foot polygon contours, in the right sub-figure, these two 30 foot contours increase their size, and still keep their 30 foot depth. The reason why these two 30 foot contours in the right sub-figure need to be exaggerated is because as the scale decreases, the chart display area for the same real geographic feature also decreases. The right sub-figure in Figure 3-18a looks like the right sub-figure of Figure 3-18b, as the scale changes from 1:20,000 to 1:40,000, the display area shrinks to $\frac{1}{4}$ of the left figure. If these two polygon contours retain their same spatial extent, they would be too small to have numbers written inside them.

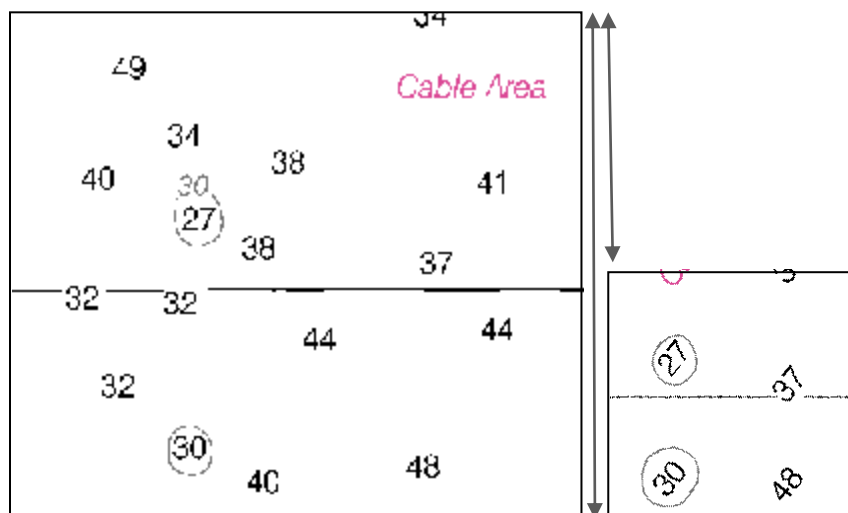


Figure 3-18b: Sample of exaggeration of simple polygon contour

The left sub-figure is a selected area from the 1:20,000 scale raster chart, 13283; the right sub-figure is the same area from the 1:40,000 scale raster chart, 13274, at the correct display scale. This example shows how exaggeration is used during generalization.

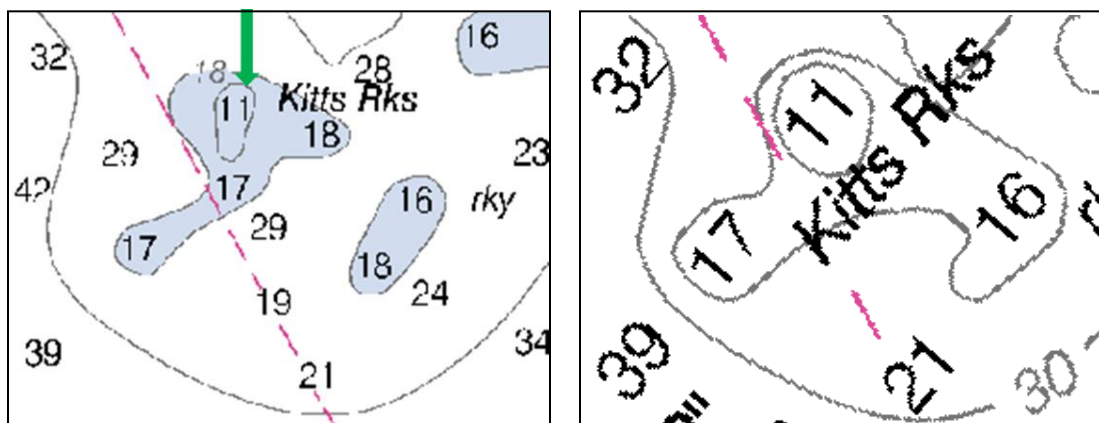


Figure 3-19: Sample exaggeration of simple polygon contour

The left sub-figure is a selected area from the 1:20,000 scale raster chart, 13283; the right sub-figure is the same area from the 1:40,000 scale raster chart, 13274. In addition to the two 18 foot contours being aggregated with each other, the 12 foot polygon contour (around the sounding number “11”, indicated by the green arrow) increases in size from the 1:20,000 scale chart to 1:40,000 scale chart. This example shows that during generalization, aggregation and exaggeration are often used together.

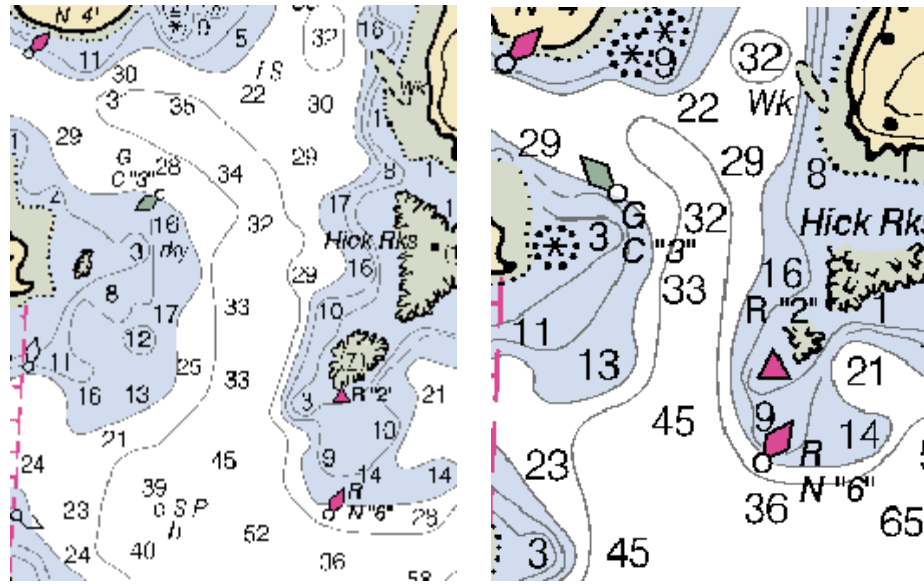


Figure 3-20: Sample exaggeration of one complex long polyline

The left figure is a selected area from the 1:10,000 scale raster chart, 13283_2; the right figure is the same area of the 1:20,000 scale raster chart, 13283_1. In the right figure, the length of the bay shape decreases, the polyline moves to the deeper side, but the width of this bay shape did not decrease significantly; the width of this bay shape is exaggerated, such that the user can recognize this inlet feature better.

In this study, the exaggeration operator is used on both polygon and polyline contours. In nautical charts, polygon contours usually represent a knoll on the seafloor or a depression. In the former there is always one or more shallow peaks inside the polygon (Figure 3-18). During the generalization process, if the polygon gradually becomes smaller and is eventually removed, this shallow depth value will be deleted, and becomes an unrepresented hazard to navigation. The polygon contour should therefore be exaggerated during generalization. The exaggeration of polylines is primarily used to maintain a minimum display distance between adjacent vertices of the polyline itself, such that this polyline will not be intersect with itself during generalization (Figure 3-20). The geographic meanings of that kind of feature are usually inlet or bay features. They are important features to mariners, as they usually are used as traffic routes for ships. By maintaining the minimum distance of these parts of the polyline contours, these important navigational features will be retained and strengthened.

3.1.2.1 Exaggerate Operator Method

There are two methods for exaggeration: one to exaggerate a polygon contour, expanding its shape, the other to exaggerate a section of a long complex polyline. The exaggeration of a polygon contour is implemented by adding an extra force to the force equation ((9) in Chapter II), and forming a new equation ((11) in Chapter II). The exaggeration method for a complex polyline works to maintain a minimum distance from the polyline itself. The exaggeration for a polyline is implemented by using an algorithm specifically for this condition (Algorithm 3-7).

For the exaggeration operator, this extra force keeps the polygon contour growing outwards. As introduced in section 2.3.2 of Chapter II, this new extra “balloon” force is defined as:

$$F_{balloon} = b \vec{n}_j \quad (22)$$

where \vec{n}_j is the unit vector normal to the curve at point P_j (Figure 3-21) and b is the amplitude of this force (Cohen and Cohen, 1993). The details of this calculation are covered in the next section.

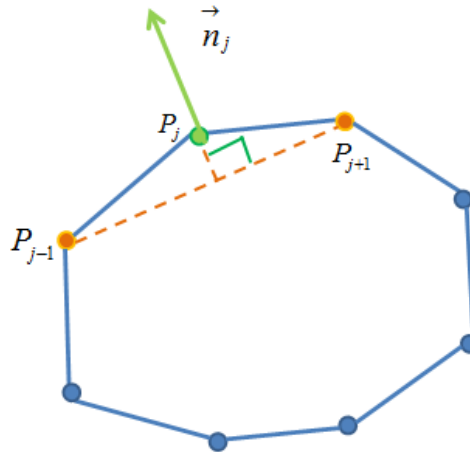


Figure 3-21: Illustration of balloon force at one vertex of a polygon contour

The balloon force \vec{n}_j at vertex P_j is perpendicular to the line $P_{j-1}P_{j+1}$.

3.1.2.2 Exaggerate Operator Implementation

3.1.2.2.1 Polygon Exaggerate Operator Implementation

The polygon exaggerate operator is implemented by adding the balloon force to the total forces:

$$\nabla E_{\text{int}} + \nabla E_{\text{ext}} + F_{\text{balloon}} \quad (23)$$

The force is applied on both the x and y coordinate of each point for all points on polygon:

$$F_{\text{balloon}} = (px, py) \quad (24)$$

For each point (x_j, y_j) , the force (px_j, py_j) is:

$$px_i = b \cdot n_{ix} \quad (25)$$

$$py_i = b \cdot n_{iy} \quad (26)$$

n_{ix}, n_{iy} are the unit normal in x, y direction, which are defined as:

$$n_{ix} = \frac{(x_{i+1} - x_{i-1})}{\sqrt{(x_{i+1} - x_{i-1})^2 + (y_{i+1} - y_{i-1})^2}} \quad (27)$$

$$n_{iy} = \frac{(y_{i+1} - y_{i-1})}{\sqrt{(x_{i+1} - x_{i-1})^2 + (y_{i+1} - y_{i-1})^2}} \quad (28)$$

By adding px_i and py_i to each original vertex (x_i, y_i) , the polygon contour will be exaggerated (Figure 3-22).

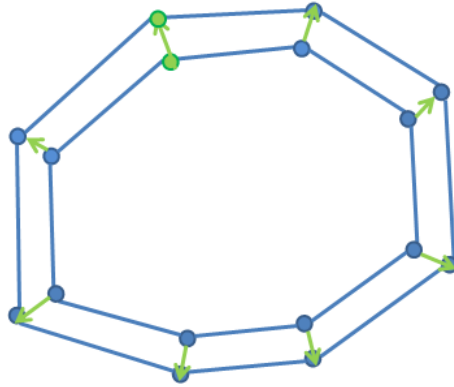


Figure 3-22: Illustration of a polygon contour after exaggeration

Each vertex is given a balloon force, it is moved outward, and the total polygon has been exaggerated. The balloon force points towards the outside of the polygon. By adding this force, the polygon is expanded, and the shape is exaggerated.

3.1.2.2.2 Polyline Exaggeration Operator Implementation

Some sections of one polyline can get very close to each other during the generalization process (Figure 3-23), and this will further lead to a self-intersection problem. So these close sections will be exaggerated during the generalization, they will not be moved until their position in the next iteration is not going to move them closer.

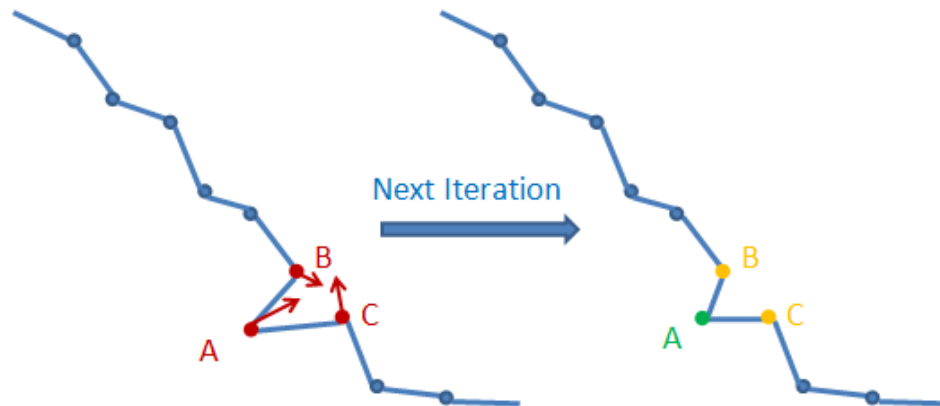


Figure 3-23: Illustration of when polyline exaggeration is needed

The total force at vertices A, B and C are in the directions shown by the red arrows; if B and C move in the direction shown by the red arrow, they will get too close. In order to maintain a legible distance between segment AB and segment BC, in the next iteration, vertex A keeps moving, vertices B and C stay stationary.

In contrast to the polygon exaggeration operator, polyline exaggeration is not implemented by adding an extra force. It is implemented by identifying the segments that are too close to each other, then avoid moving these points in the next iteration, so that they will not get closer, and such that these segments are exaggerated. The pseudo code is as follows:

Algorithm 3-7

Input: One polyline contour; and this polyline contour prior to adding the B-spline snake step size (polyline contours in two different states, refer to section 2.3.1).

1. Iterate through all points of the polyline
 - 1.1 Calculate the distance between each point and all the other points on this polyline except itself and its neighbor points.
 - 1.2 Record the indices of the points for which the distance to the current point is smaller than a threshold (the threshold is set to the average value of all distances between the polygon vertices and their neighbor points).
 - 1.3 Replace the recorded points' coordinate values with the coordinates before adding the previous B-snake step size (such that the minimum distance between different parts of the polygon is maintained).
2. End Iterate

3.2 Auxiliary Functions

This section illustrates all important auxiliary functions used in the workflow, that have not been mentioned in the operator section. The purpose of these functions is explained and pseudo code is shown.

3.2.1 B-spline Snake Calculation

The B-spline snake calculation function is the implementation of the B-spline snake. The

explanation of B-spline snake and how to approximate the gradient of the curvature have been described in Chapter 2 ((5) to (10) and (14) to (15)). Algorithm 3-8 shows the pseudo code of this B-spline calculation.

Algorithm 3-8

Input: one control points polyline that approximates the input contour line, and the original contour line.

1. Iterate through all points of the control points polyline (the i th point has coordinates (x_i, y_i)).
 - 1.1 Calculate the curvature k of the current point (see (5)).
 - 1.2 Calculate the gradient value $g_{k,x}, g_{k,y}$ of the curvature at this point in x and y directions (see (15)).
 - 1.3 Calculate the step-size $\delta_{i,x}, \delta_{i,y}$ of just the internal energy: $\delta_{i,x} = k \cdot g_{k,x} \cdot \beta$, $\delta_{i,y} = k \cdot g_{k,y} \cdot \beta$ (the first part of (10) in Chapter II) in x and y directions.
 - 1.4 If $\delta_{i,x} < 15$ or $\delta_{i,y} < 15$ (See Appendix A.7)
 - 1.4.1 Add $\delta_{i,x}$ or $\delta_{i,y}$ to the coordinate of current point: $x_i' = x_i + \delta_{i,x}$ or $y_i' = y_i + \delta_{i,y}$.
 - 1.5 Else
 - 1.5.1 The new coordinate of the current iterating vertices is the average of the coordinate of its two neighbor points: $x_i' = \frac{x_{i-1} + x_{i+1}}{2}$, $y_i' = \frac{y_{i-1} + y_{i+1}}{2}$.
 - 1.6 End If
 - 1.7 Calculate the gradient $g_{d,x}$ and $g_{d,y}$ of the external energy at this current vertex in x and y directions ((21) and (22)).
 - 1.8 If the current point is on the shoaler side of the original contour
 - 1.8.1 Coefficient $C_o = 1$
 - 1.9 Else
 - 1.9.1 Coefficient $C_o = 0$

```

1.10 End If
1.11 Calculate the step size  $\delta_{e,x}, \delta_{e,y}$  of the external energy:  $\delta_{e,x} = g_{d,x} \cdot c_o, \delta_{e,y} = g_{d,y} \cdot c_o$ ,
      in x and y directions.
1.12 If  $\delta_{e,x} < 15$  or  $\delta_{e,y} < 15$  (See Appendix A.7)
      1.12.1 Add  $\delta_{e,x}$  or  $\delta_{e,y}$  to the coordinates of this current iterating vertex:
            
$$x_i' = x_i + \delta_{e,x} \text{ or } y_i' = y_i + \delta_{e,y}$$

1.13 Else Maintain the current location of the point
1.14 End If
2. End Iterate

```

3.2.2 Preprocess Polygon Contour Function

The preprocess polygon contour function cleans the polygon contours before they are generalized. The raw polygon contour data from the ENC have indices that randomly increase clockwise and counter-clockwise, which will cause a problem in polygon aggregation. The raw polygon data have a duplicated point at the beginning of each polygon, and this duplicated point will lead to noise in the exaggeration result. Therefore, the duplicated points are deleted, and the sequences of the indices of all polygons are set to increase clockwise.

The pseudo code is as follows:

Algorithm 3-9

Input: one polygon contour selected from the ENC data with x, y coordinates and depth value at each point.

1. Delete the duplicated point at the beginning of the polygon.
2. If the indices increase counter clockwise, reverse the indexing direction of the polygon.
3. Add points to the polygon, if the distance between neighboring points is larger than six (See Appendix A.5), adding at most one point to each segment.

3.2.3 Maintain Minimum Distance Between Neighbor Contours Function

The maintain minimum distance between neighbor contours function is used to maintain a minimum distance between two neighbor contours. It is used when there is more than one contour being generalized at the same time. A minimum distance between two neighbor contours is maintained such that the neighbor contours can be distinguished by human eyes. The contours can be either polyline contours or polygon contours. The pseudo code is as follows:

Algorithm 3-10

Input: two contours and the minimum value the user wants these two contours to maintain.

1. Calculate the distance between each pair of points on the two contours.
2. If the distance between any two points is less than the minimum value N (See Appendix A.8)
 - 2.1 Move the contour with deeper depth value to the deep direction with a step size of the minimum value that neighbor contours should maintain minus the current distance.
3. End If

3.2.4 Polygon Group Intersection Prevention Function

The polygon group intersection prevention function is used to prevent, any intersection in groups of concentric polygon contours. In a concentric polygon group, after exaggeration, some inner polygon might intersect with its neighbor polygon. This algorithm is used to prevent this situation from happening, and to also maintain a minimum distance M between the concentric polygons. The pseudo code is as follows:

Algorithm 3-11

Input: a group of concentric polygons and the minimum distance, M , that the user want to maintain between the neighbor concentric polygons.

1. Iterate through all polygons in this group of concentric polygons(sequence does not matter)
 - 1.1 Calculate the distance between the current polygon and its neighbor polygons. If the current polygon is at a boundary (inner boundary or outer boundary), it has just one neighbor.
 - 1.2 If the distance is smaller than the minimum value M (See Appendix A.9)
 - 1.2.1 Call Algorithm 3-10 (maintain minimum distance).
 - 1.3 End If
2. End Iterate

3.2.5 Polyline Self Intersection Removal Function

The polyline self intersection removal algorithm is used after each aggregation or exaggeration step to detect and delete any self intersection of the newly aggregated or exaggerated contour. This step is important in the generalization because it keeps the generalization result stable during the generalization. The pseudo code is as follows:

Algorithm 3-12

Input: one control points polyline (the aggregated or exaggerated result from a previous step).

1. Iterate through all line segments on the contour
 - 1.1 Check if the line segment $S_i S_{i+1}$ (S_i, S_{i+1} are the starting and ending vertex of current iterating line segment) and its neighbor points intersects with any other segment of this contour; record the intersection segments $S_i S_{i+1}$'s first vertex's index i in a list A and the segment $I_j I_{j+1}$'s first vertex's index j in a list B (I_j, I_{j+1} are the line segment intersected points, j is the index of the point closest intersection point).
 - 1.2 If list A is not empty
 - 1.2.1 Delete the points with the indices between i and j in step 1.1.
 - 1.2.2 The new contour is made of the remaining vertices.

- 1.3 End If
2. End Iterate

3.3 Workflow

In this section, five workflows for different generalization scenarios are illustrated. They are listed from the simplest scenario to the most complex one. The scenarios include:

1. The simplest condition: one contour line generalization, which is how one contour line is smoothed and simplified and is prevented from moving to the shoaler side of the original position.
2. One polyline and several polygon contours: the polyline contour aggregates with the polygon contours when it is generalized, which is a more complex.
3. Similar to scenario 2, except that the polygons are also exaggerated, as is sometimes required.
4. Exaggerations of multiple concentric polygon contours, with aggregation.
5. A combination of scenarios 1 and 3, which is close to the full generalization problem.

3.3.1 Workflow for Single Polyline Contour Generalization

If the generalization object is just one line, which is the simplest case of generalization, the workflow is as follows:

Algorithm 3-13

Input: one polyline contour with (x, y) coordinates, and the number R of iterations (see Appendix A.10).

1. Repeat (repeat R times)
 - 1.1 Apply the simplification operator: represent this contour as a B-spline curve with about 80% (see Appendix A.3) of the original curve points (section 2.3.4.1 of

Chapter II).

1.2 Apply the smoothing and exaggeration operators. Set the internal and external energy terms (section 2.3.3.1 of Chapter II).

1.3 Solve the energy equation. Calculate the step size of the current contour, and move the current contour to the next step (section 2.3.1 of Chapter II).

2. End Repeat

3.3.2 Workflow for One Polyline and Multiple Polygons On One Side

The one polyline and multiple polygons scenario is more complex than the first case; it contains two types of contour features: a polyline and polygon contours. Simplification, smoothing, shoal-biasing, and the aggregate operators will be added in this generalization process. In this scenario, when approximating the original contour with B-spline, a least square method may be used, if too many control points are specified.

Algorithm 3-14

Input: A set of polyline and polygon contours, and the number R of iterations (see Appendix A.10).

1. Represent all polyline contours as B-spline curves with 80% or less of the original curve points (section 2.3.4.1 of Chapter II).
2. Preprocess polygon contours: equally distribute the points on the polygon contour by distance, and add points to the polygon, such that the distance between each point is smaller than 1/100 of the distance between the furthest two points of that polygon.
3. Repeat R times
 - 3.1 Calculate the distance between the current snake position and all other polygons in the data, and find the feature that is closest to the current snake, and the closest approach distance for that feature.

3.2 If the closest approach distance is smaller than a threshold (4 here due to the line thickness observed) indicating that the two features are too close and need to be aggregated, and this feature has not been aggregated before:

3.2.1 Mark the closest feature as having been aggregated.

3.2.2 Find the two segments that connect the current curve and closest feature.

3.2.3 Add the two segments and the remaining part of the closest feature into the current snake (Algorithm 3-1).

3.3 End If

3.4 If the distance between any two neighbor points on the current snake is larger than a suitable threshold (1/60 of the total length of current contour was chosen empirically):

3.4.1 Add points to all segments where two original points are too far away from each other (similar but opposite to the process of Algorithm 2-1, add extra points to segments where start and end points are too far from each other).

3.5 End If

3.6 If the distance of any two neighbor points on the current snake curve is smaller than a suitable threshold (1/600 of the current snake length was chosen empirically):

3.6.1 Find the set of all points that are within the threshold distance of their neighbors.

3.6.2 Find the sub-set of the set found in step 3.6.1 of all groups of at least three consecutive points in sequence.

3.6.3 Delete the first point in each group of three continuous points.

3.7 End If

3.8 Calculate the step size of the current snake, and move the current snake to the next step (Algorithm 3-8).

4 End Repeat

Note: See Appendix A.4 to A.6 for detail of the parameters in this function.

3.3.3 Workflow for A Group of Polygon Contours Exaggeration and Aggregation

The group of polygon contours exaggeration and aggregation workflow deals with the scenario of generalization of a group of polygon contours. As illustrated by the cartographer's manual process of generalization results (Figure 3-4 and 3-18), in order to maintain shallow features on the chart, all polygon contours must be exaggerated on a smaller scale chart. As they expand their size, close pairs of polygon contours are aggregated into one polygon. This workflow implements this process, and the workflow is as follows:

Algorithm 3-15

Input: a group of simple polygon contours.

1. Preprocess all polygon contours, such that all of their indices are increasing clockwise (Algorithm 3-9)
2. Iterate through all polygons:
 - 2.1 Exaggerate current polygon (Algorithm 3-7).
 - 2.2 Apply Algorithm 3-6 to correct any self intersections
 - 2.3 Check if this polygon intersects with or is too close to any other polygons in this polygon array. If there is intersection
 - 2.3.1 Store the index number of the pairs of intersection polygons, and close polygons.
 - 2.3.2 If there are close polygons, but no intersection polygons
 - 2.3.2.1 Aggregate the close polygons with the current polygon one by one using Algorithm 3-5.
 - 2.3.2.2 Check if the aggregated polygon has any self intersection, and delete the self intersections if there are any (Algorithm 3-12)

```

2.3.3 If there are intersected polygons but no close polygons
    2.3.3.1 Aggregate the intersected polygons with the current polygon one by one
            with the aggregate function for two intersected polygons (Algorithm 3-
            6).
    2.3.3.2 Check if the aggregated polygon has any self intersection, and delete the
            self intersection if there is any (Algorithm 3-12).
2.3.4 Else there are both intersected polygons and close polygons
    2.3.4.1 Aggregate the close polygons with method in step 2.3.2
    2.3.4.2 Aggregate the intersected polygons with method in step 2.3.3.
2.3.5 End if
2.4 End If
3. End Iterate

```

3.3.4 Workflow for Concentric Polygon Contours Exaggeration and Aggregation

The concentric polygon contours exaggeration and aggregation workflow deals with the scenario when the generalization object is a complex set of multiple polygon contours and multiple sets of concentric polygons with various depth values.

Algorithm 3-16

Input: Multiple polygons and multiple sets of concentric polygon contours, and number R of iterations (see Appendix A.10).

1. Preprocess all polygon contours using Algorithm 3-9, such that the indices are increasing clockwise.
2. Divide polygons into two groups: concentric polygons in one group, and simple polygons in another.
3. Repeat R times

- 3.1 Maintain a minimum distance between the most outside polygon and its neighbor by using Algorithm 3-10.
- 3.2 Aggregate neighbor concentric polygon groups with Algorithm 3-5.
- 3.3 Aggregate the polygon and the outer-most polygon of the concentric polygon groups with Algorithm 3-3 and 3-4.
- 3.4 Delete any self intersections of the aggregated polygons using Algorithm 3-12.
- 3.5 Maintain a minimum distance between the outer-most polygon and its neighbor by using Algorithm 3-10.
- 3.6 Aggregate neighbor single polygons if they are close (Algorithm 3-3 and 3-4).
- 3.7 Delete any self intersections of the aggregated polygons.
- 3.8 Maintain a minimum distance between any polygon contour and its neighbor (Algorithm 3-10).
- 3.9 Exaggerate all polygons one by one, for each polygon; if there is self intersection
 - 3.9.1 Delete the self intersection if it exists (Algorithm 3-12).
4. End Repeat

3.3.5 Workflow for Polyline and Polygon Contour Exaggeration and Aggregation

The polyline and polygon contour exaggeration and aggregation scenario is different from the second scenario (workflow 3.3.2) because not only does the line aggregate with the polygon during generalization, but the polygons themselves exaggerate their shape and aggregate with neighbor polygons when they get too close. The workflow is as follows:

Algorithm 3-17

Input: One long complex polyline contour and multiple simple polygon contours, and the number R of iterations (see Appendix A.10).

1. Preprocess all polygon contours using Algorithm 3-9.

2. Approximate the polyline contours with B-spline Snakes with about 80% (see Appendix A.3) of the original curve points, reduce the points on the polyline contour.
3. Repeat R times
 - 3.1 Calculate the step size of the polyline, and move the polyline to the next step.
 - 3.2 Delete any self intersection using Algorithm 3-12.
 - 3.3 Call workflow 3.3.3, exaggerate and aggregate all the single polygon contours.
 - 3.4 If any polygon contours are too close to the polyline or if any polygon contours intersect with the polyline
 - 3.4.1 If there are polygon contours close to the polyline and no intersected polygon
 - 3.4.1.1 Aggregate the line with the polygon using Algorithm 3-1 and 3-2.
 - 3.4.2 Elseif there are polygons intersecting with the polyline but no polygon detected close to the polyline
 - 3.4.2.1 Aggregate the line with the polygon with Algorithm 3-1 and 3-2.
 - 3.4.3 Elseif there are polygons intersecting with the polyline and polygons close to the polyline
 - 3.4.3.1 Aggregate close polylines and polygons with the method in step 3.4.1.
 - 3.4.3.2 Aggregate intersecting polylines and polygons with the method in step 3.4.2.
 - 3.4.4 End If
4. End Repeat

3.4 Summary

This chapter illustrates algorithms that implemented exaggeration and aggregation operators, and auxiliary functions that are used in the generalization process. This chapter

illustrates five scenarios that might occur in generalization process, and the workflow of how they can be simulated with the functions and operators developed in this thesis work. The next chapter shows one example of each these five scenarios and how the cartographer's manual generalization results compare.

CHAPTER IV

TEST EXAMPLES

In this chapter, five test examples are shown to illustrate how the B-spline Snake method is used to generalize contour features. All of the examples progress while keeping the shoal-biased rule enforced. Test scenario one is a simple line example, showing how one single line is gradually simplified and smoothed. Test scenario two is an example of one simple polyline contour and a set of polygon contours. The polyline is generalized with the same behavior as in example 1, and meanwhile the polygons are being aggregated with the polyline. Test scenario three is an example of using the exaggeration operator on a group of polygon contours. During the generalization, each polygon exaggerates its shape, and neighbor contours are aggregated into one large polygon when they get close. Test scenario four is an example of a set of polygon contours, the difference is that there are concentric polygon contours (algorithm 3-15 in chapter III) whereas previous scenarios only deal with simple polygons. This scenario occurs in manual generalization too, and the process is different than scenario three. Test scenario five is an example of a long, complicated polyline and a set of simple polygon contours. The polyline contour is simplified and smoothed while the polygons are exaggerated. The polygons and polylines aggregate each other in the generalization process. The ENC data used in the following scenarios are downloaded from NOAA website (NOAA, NOS website).

4.1 Test Scenario One: One Line Example

Scenario one shows the generalization results of one contour line. The simplification operator, smoothing operator and shoal-bias constraint are tested in this example. Figure 4-1 shows the shape of the original ENC input contour on a background of a 1:20,000 scale raster

chart. Figure 4-2 shows the same ENC contour on a 1:80,000 scale chart, where the same depth contour has changed its position. It has moved to the deeper side, and become smoother. Figure 4-3 to Figure 4-6 show the intermediate stages of the generalization result.

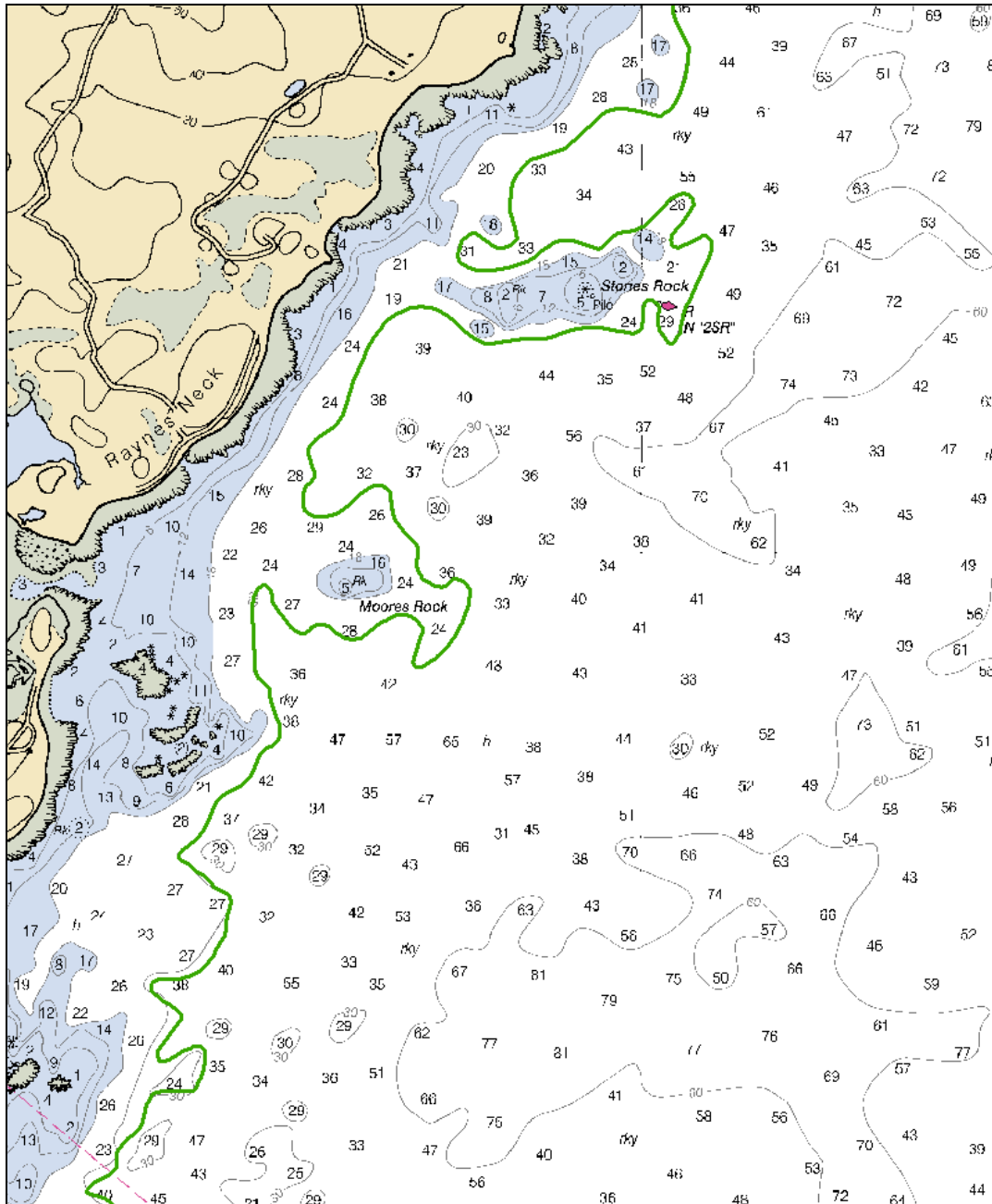


Figure 4-1: Original input with background of raster chart 13283

This polyline is selected from the ENC US5NH02M of Portsmouth Harbor, NH; the background chart is paper chart 13283 (scale 1:20,000). The selected ENC 60 foot contour overlays with the 60 foot contour line in this background raster chart.

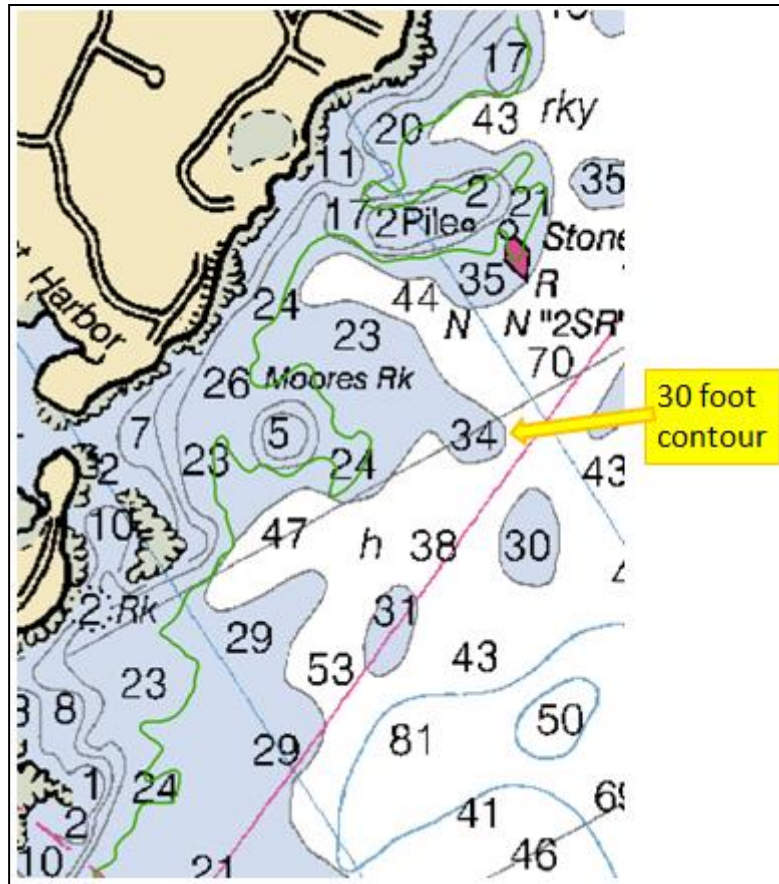


Figure 4-2: Original input on raster chart 13278

Green contour is the original input 60 foot contour, the yellow arrow points to the contour which is the same depth on the 1:80,000 scale raster chart 13278.

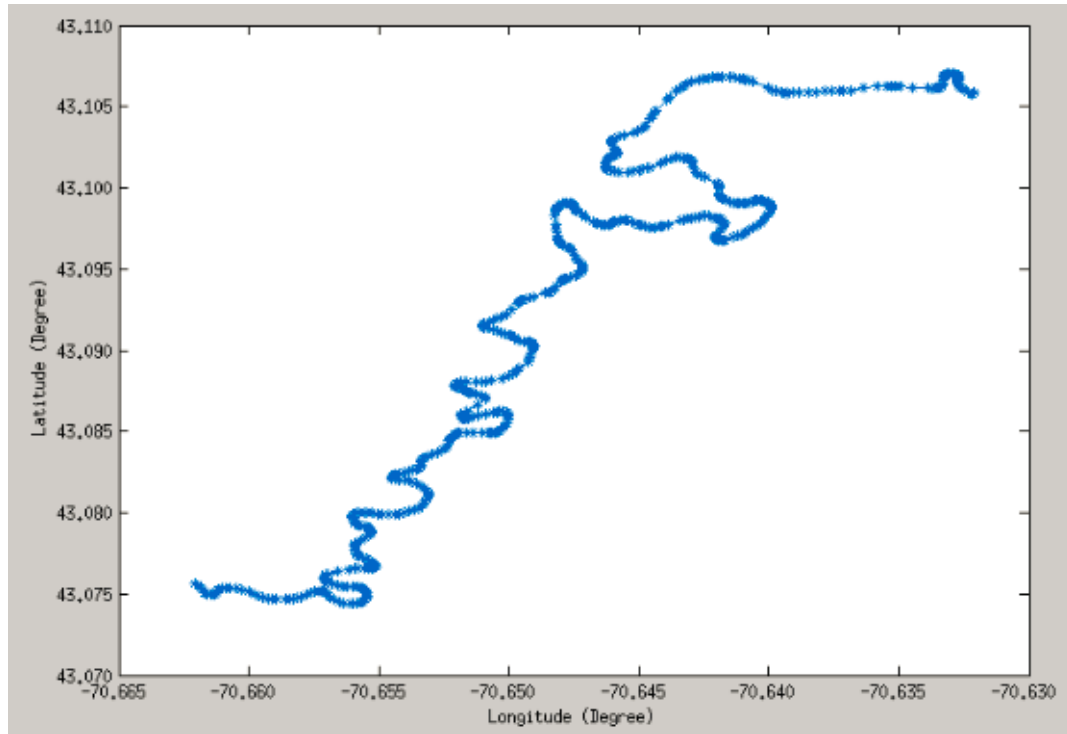


Figure 4-3: Original input contour polyline with a star symbol at each vertex

This figure shows the shape of the original input contour in this test scenario. The original contour has 787 points. In the next step, the original contour is represented by a B-spline curve. This decreases the total points that need to be stored, and increases the processing speed.

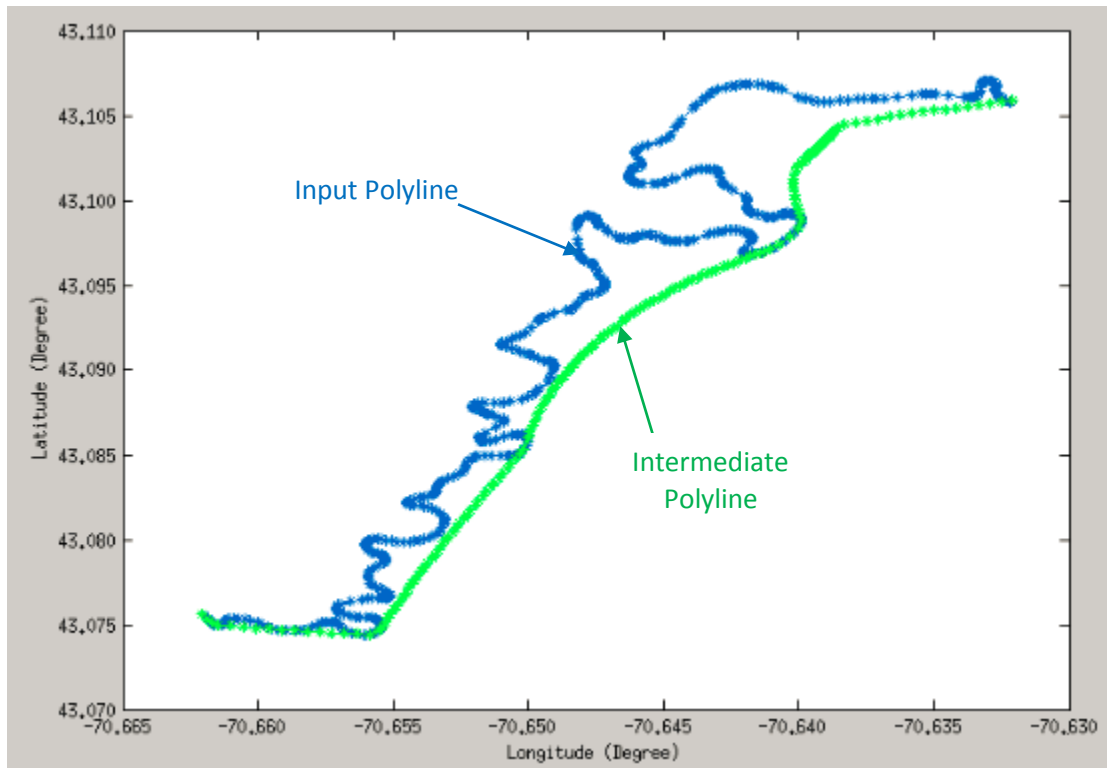


Figure 4-4: Intermediate stage of the generalization process

The blue curve is the result of B-spline simplification, which now contains only 607 points. The green curve is the intermediate stage of the generalization. The generalized curve is smoother than the input, and contains less detail. The curve has shifted to the deeper side of the input, which obeys the shoal-bias principle.

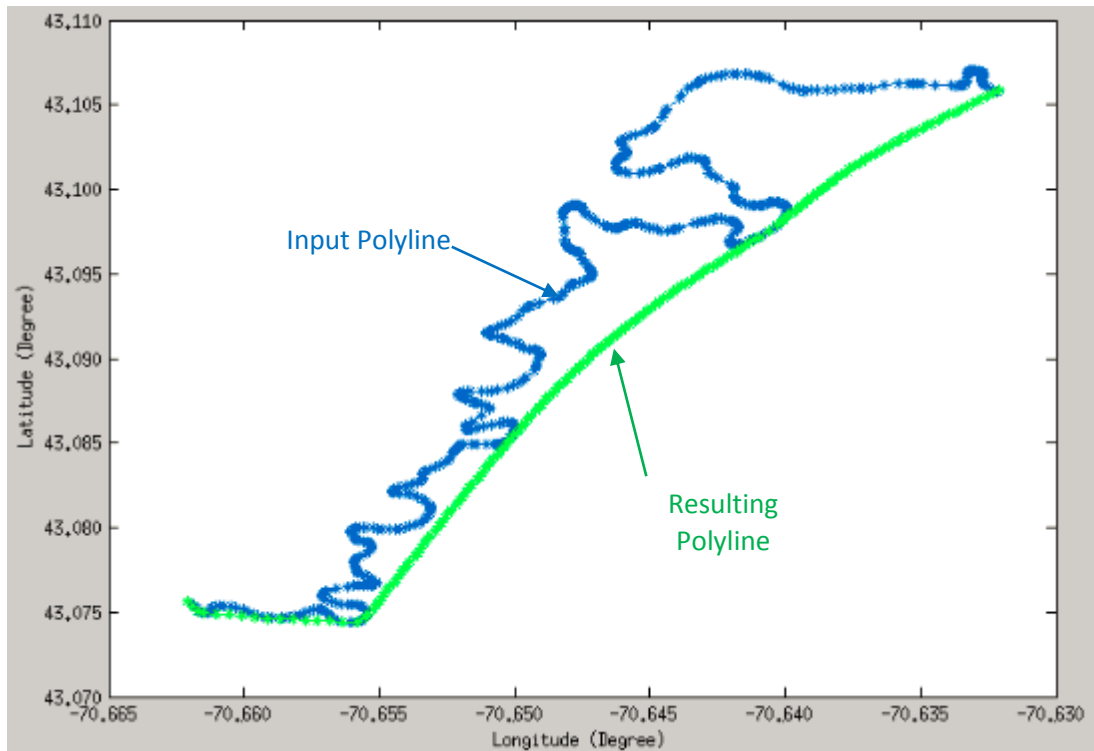


Figure 4-5: Generalization result of one simple polyline contour generalization

The green polyline is the generalization result after smoothing and simplifying. Compared to the result of cartographer's manual process of generalization in Figure 4-2, the algorithm provides a similar result in that the new curve is simpler and smoother than the original input, and has been shifted to the deeper side of original curve.

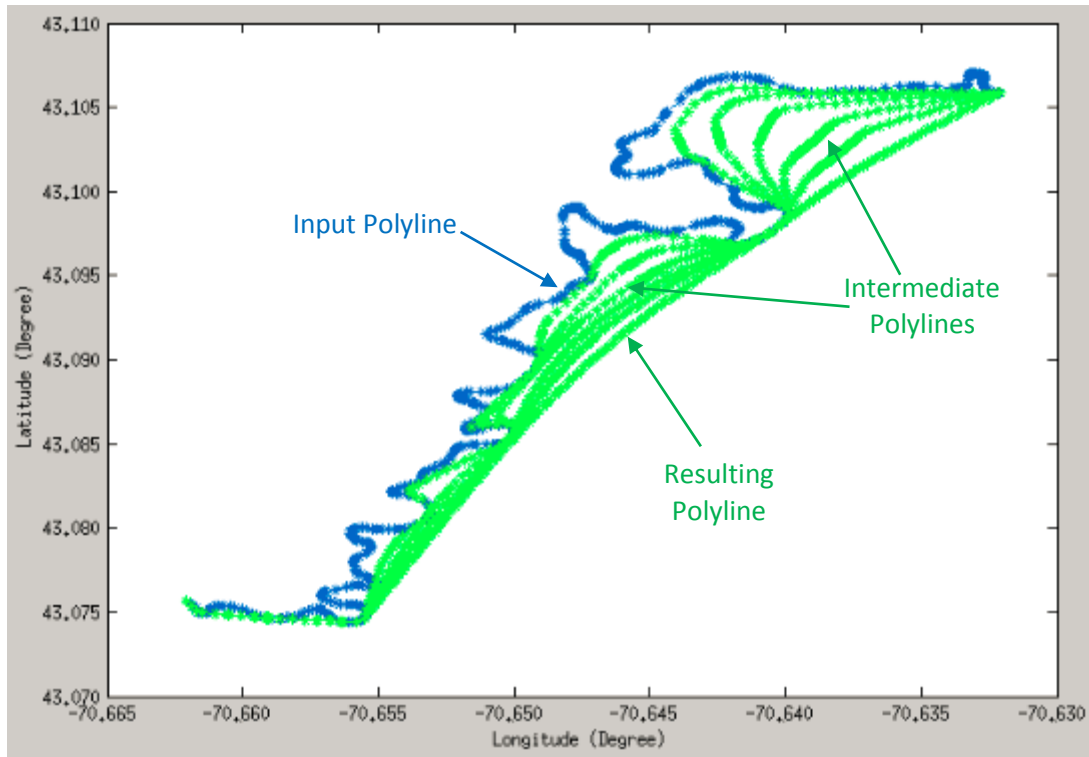


Figure 4-6: Original input line versus gradual generalization results

The blue line is the original input contour; the green curves are the middle stages of gradual generalization.

The result of this algorithm (Figure 4-6) is similar to what the cartographer did manually (Figure 4-1). The polyline contour is smoothed and simplified, and the new polyline moves to the deeper side of the original input polyline contour.

4.2 Test Scenario Two: Polyline Contour and Polygon Contour Aggregation

Scenario two shows the generalization results of one polyline contour aggregating with a set of polygon contours. Figure 4-7 shows the original input contours: one polyline contour and nine polygon contours. All nine polygon contours are on the deeper side of the polyline contour.

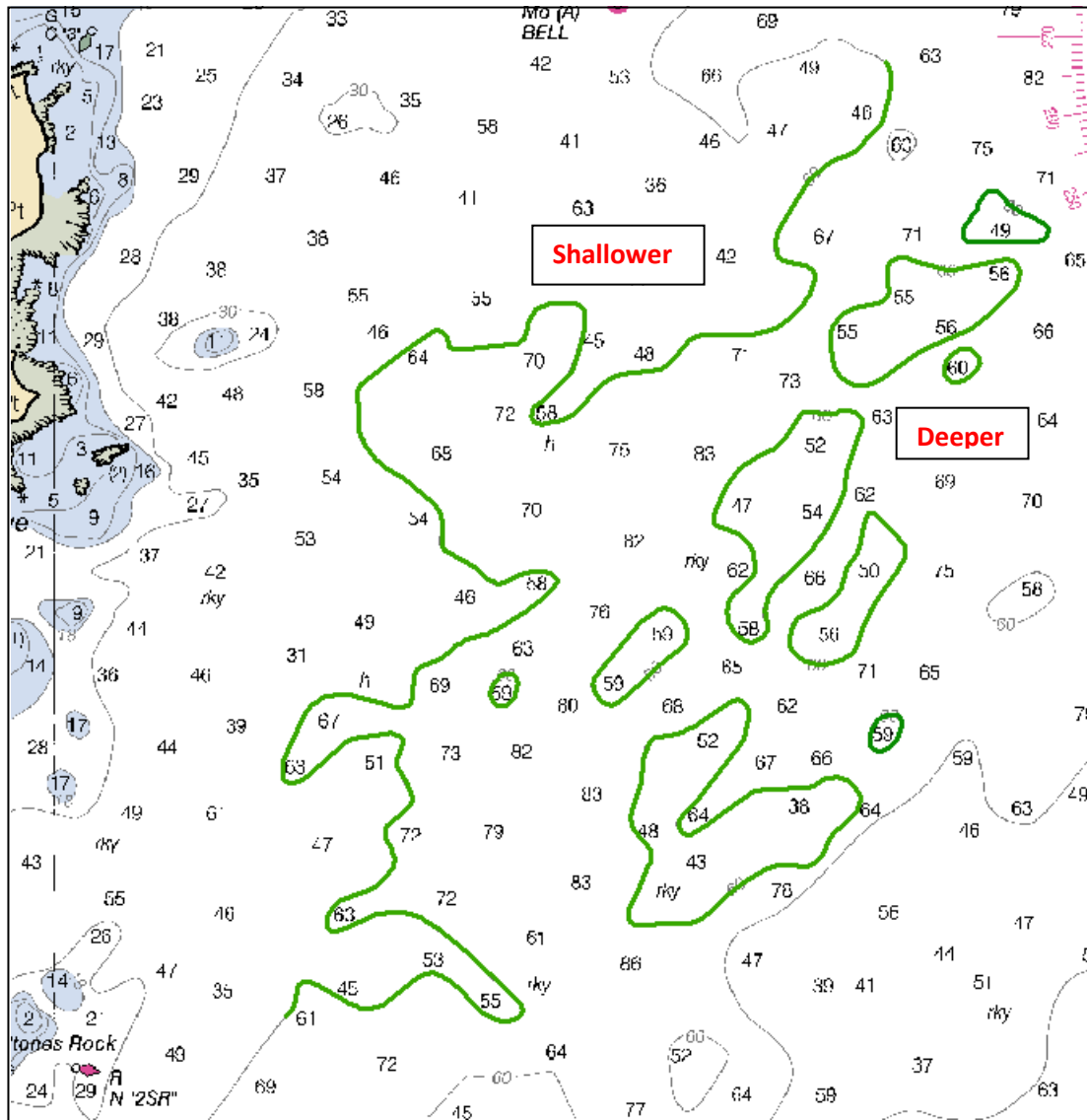


Figure 4-7: Original input contours with background of raster chart

Background is the 1:20,000 scale raster chart 13283. The green contours are the selected 60 foot contours from the ENC US5NH02M of Portsmouth Harbor, NH. The selected ENC 60 foot contour overlays with the 60 foot contour line in this background raster chart. In this test example, the aggregation operator is tested for how the polyline contour aggregates the other nine polygon contours.

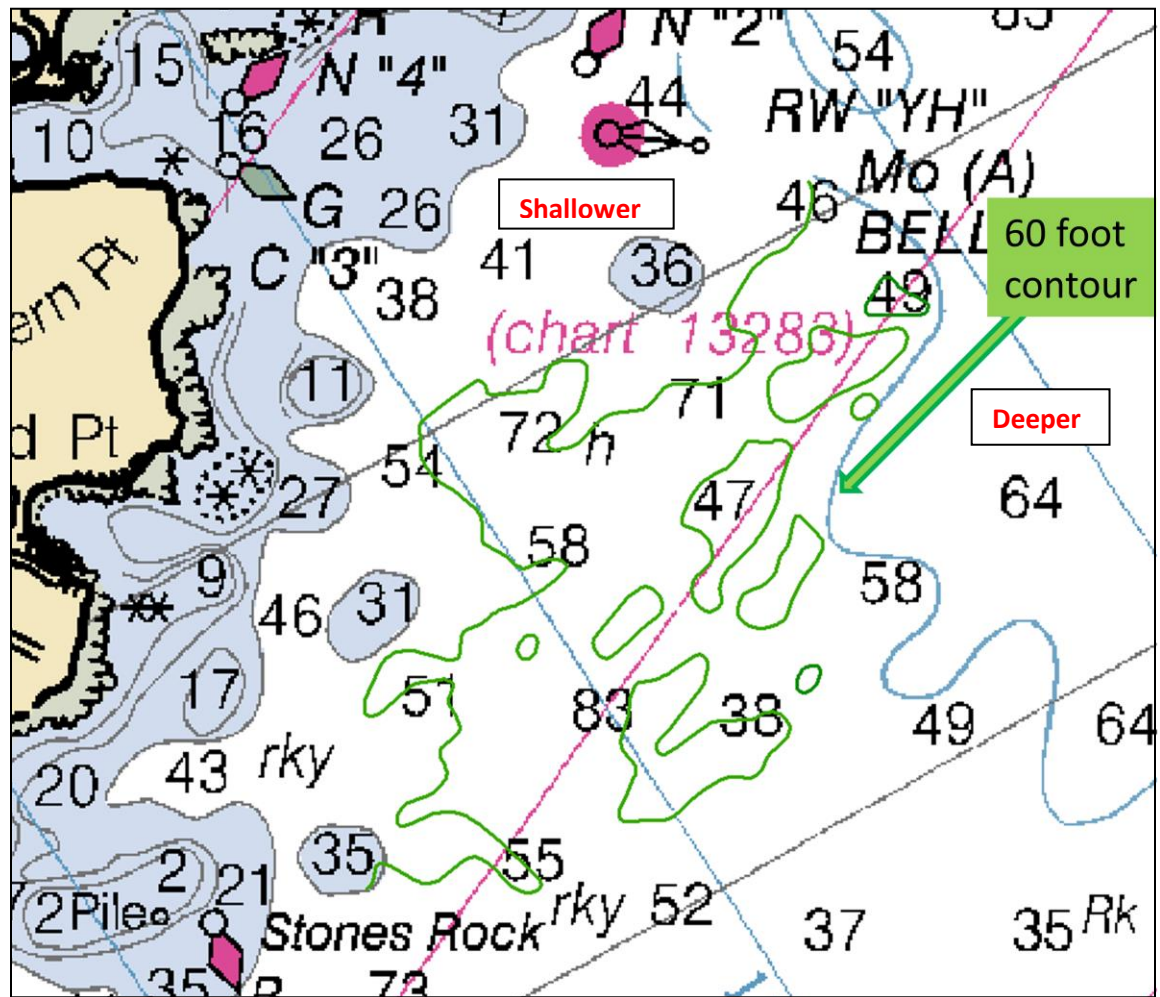


Figure 4-8: The input contours on the background of raster chart

Background is the 1:80,000 scale raster chart 13286. The green contours are the input 60 foot ENC polyline and polygon contours; the blue contour indicated by the green arrow is the 60 foot contour on this 1:80,000 scale raster chart. In this figure, cartographers generalized by deleting all of the 60 foot green contours, which are on the 1:20,000 scale chart, and aggregated all polygon contours with the polyline contour, such that the generalized result is only one polyline contour (the blue curve (not the thinner blue straight line) indicated by the green arrow).

Figure 4-8 shows the cartographers' manual process of generalization. The polygon contours are deleted, and a new polyline is created. This manual process can be simulated as a polyline aggregating with polygon contours when it is moved to the deeper side during generalization. This example shows the generalization results of using the aggregation, line smoothing, and simplification operators. Figure 4-9 to Figure 4-12 show the contours at progressive iterations through the generalization process.

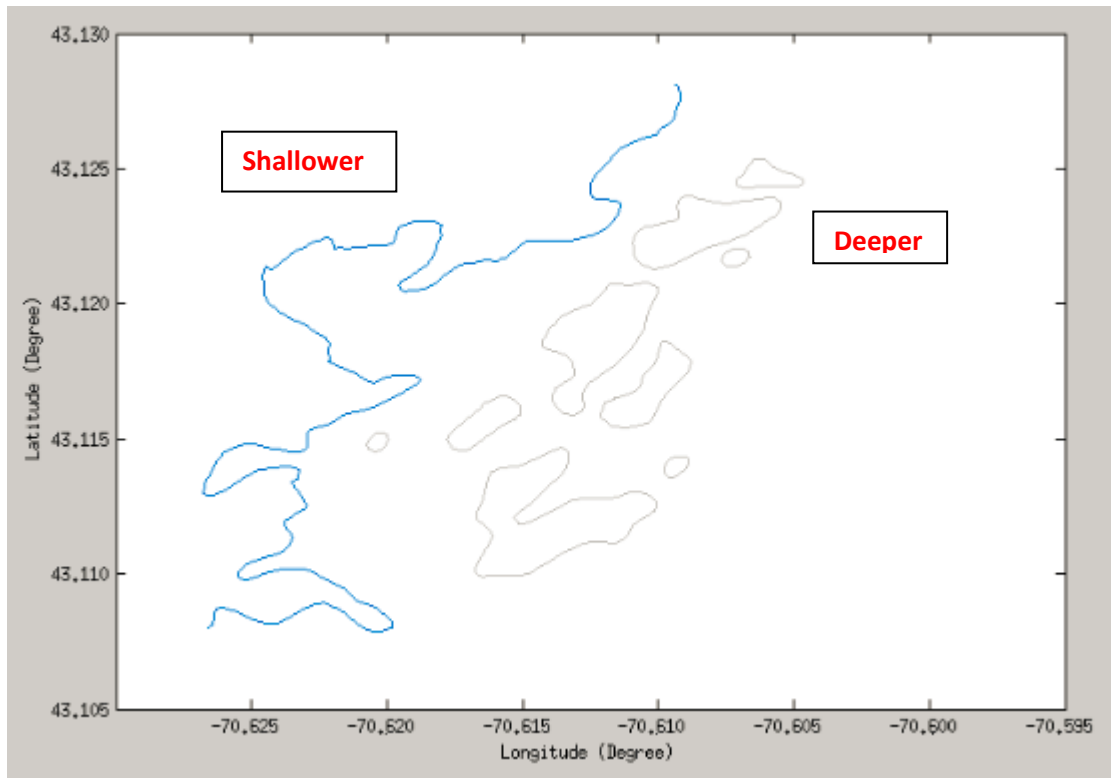


Figure 4-9: Original input of one polyline and nine polygon contours

Both the blue polyline contour and the grey contours are of 60 foot depth. The polygon contours are located seaward of the polyline contour. In this example, the blue polyline contour will aggregate the grey polygon contours.

The progressions of the generalization algorithm are shown in Figure 4-9 to Figure 4-12. Figure 4-10 shows an early stage of generalization. The light blue target contour has been generalized from the original dark blue line, but has yet to encounter any of the other closed contours. Figures 4-11 to 4-12 illustrate the situation where a number of contours have been aggregated into the target polyline contour as it has been increasingly generalized. The shape of the contours being aggregated can be seen as jumping seaward when the polyline contour encounters the landward-most point of each contour. After generalization (Figure 4-12), all of the contours are aggregated, the result is a smooth contour that maximizes the outer hull of all of the contours, while the segments between the promontories are smoothed.

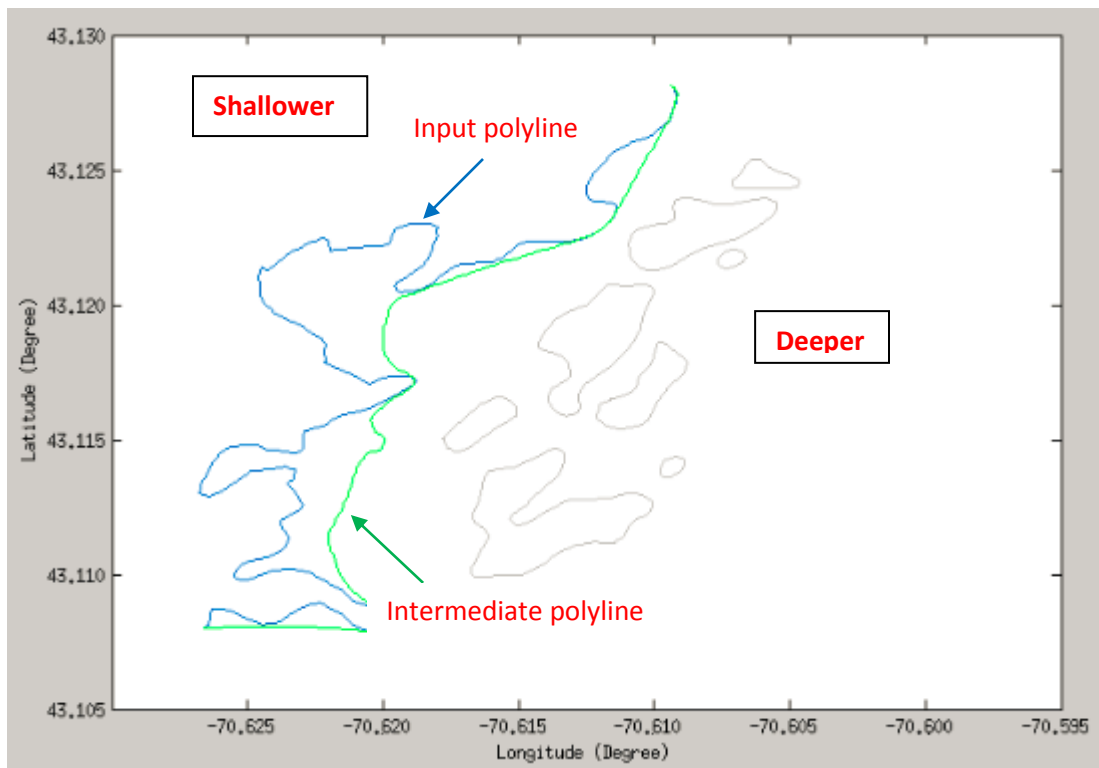


Figure 4-10: An early stage of generalization at about 10% of the whole process
 No polygon contours have been encountered. Only the polyline contour is generalized.

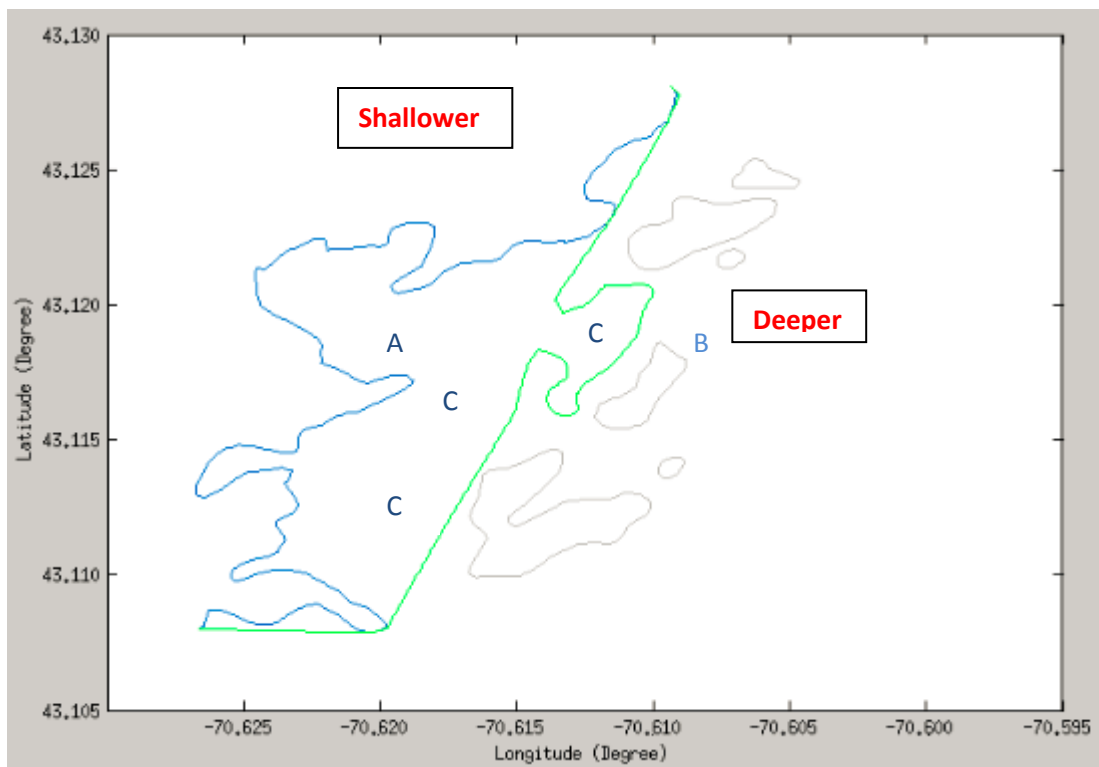


Figure 4-11: Intermediate stage of generalization at about 30% of the whole process

The generalized result after two polygon contours have been aggregated (A and B). The seaward shape of the contour currently being aggregated (B) has been preserved. Three polygons have been aggregated (C). The polygon contour is moving towards the deeper side, and aggregates polygon features when they are close.

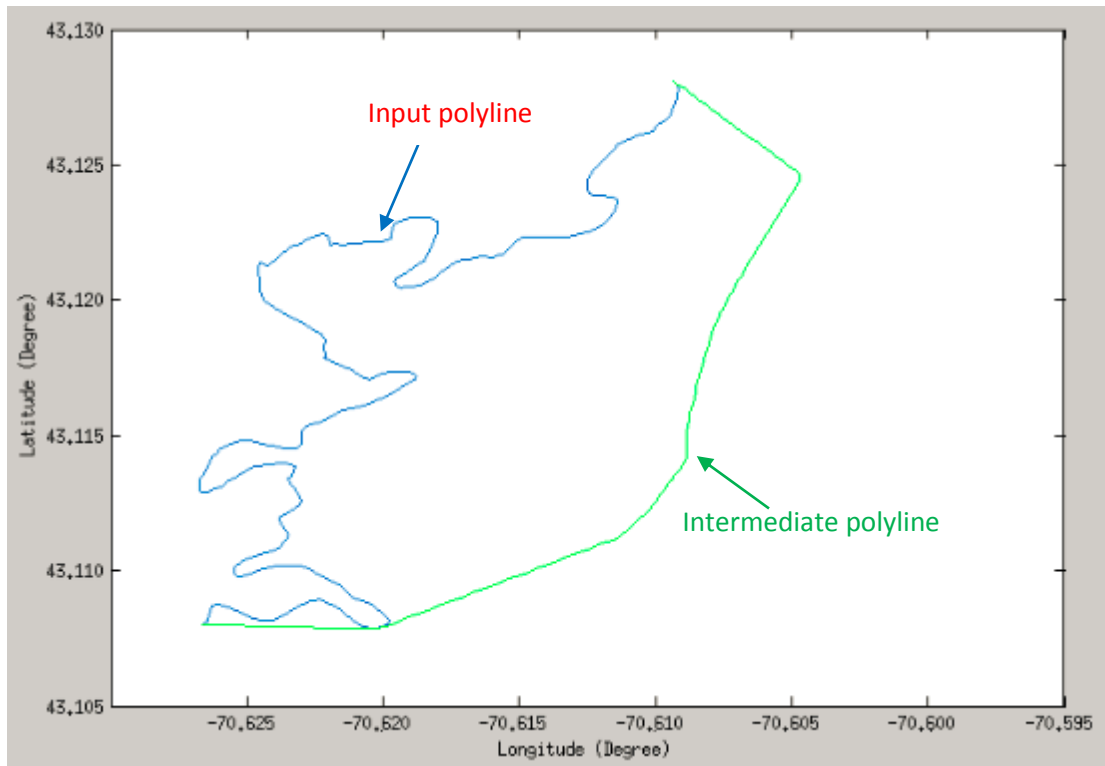


Figure 4-12: The final state of generalization

All of the polygon contours have been aggregated and generalized, such that the result (at much lower scale) preserves only the outer promontories of the originals, with smooth transitions between them. All polygons are deleted, such that only one polyline remains. This polyline is located seaward of where the most seaward points of the old polygons were.

The polygon features are not exaggerating their own size or aggregating with each other in this test example.

The result of this algorithm (Figure 4-12) is similar to what the cartographer did manually (Figure 4-8). The polygon contours are all deleted, and the new polyline moves to the deeper side of all previous polygon contours.

Aggregation

The green contours are the selected 30 foot polygon contours from the ENC US5NH02M of Portsmouth Harbor, NH. Background is raster chart 13283 of the same area. The selected ENC 30 foot contour overlays with the 30 foot contour line in this background raster chart.

Figure 4-13 shows the input contour data on the background of a 1:20,000 scale raster chart. The green contours are the selected polygon contours from the ENC of Portsmouth Harbor, NH. There are 12 polygon contours in the input. Similar to the previous test scenarios, the ENC data has the same shape as the polygon contours on the 1:20,000 scale raster chart, 13283 (Figure 4-13).

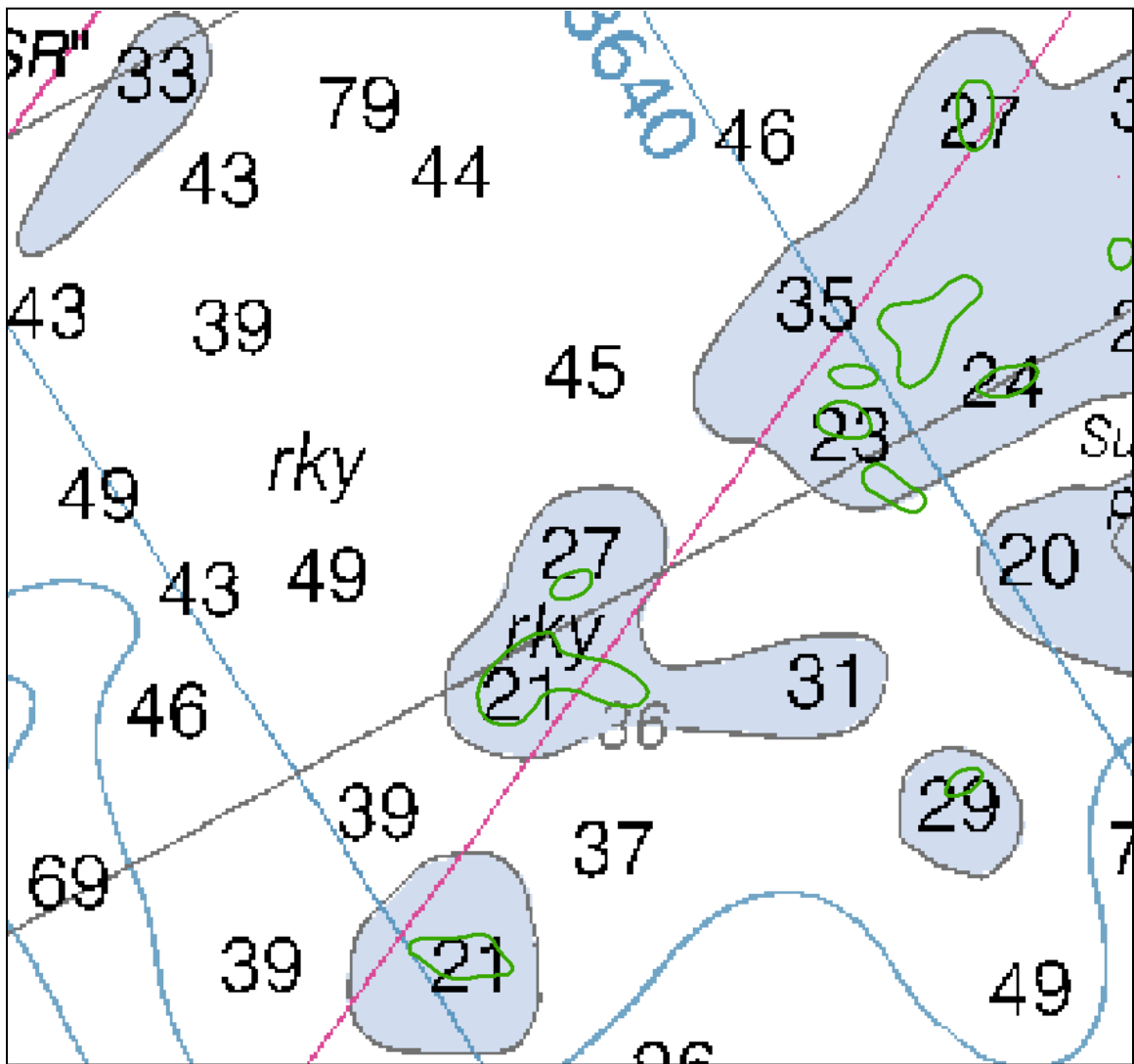


Figure 4-14: Input polygon contours on a background of the raster chart 13286

The green contours are the input contours (selected from ENC); the background is raster chart 13286 with scale of 1:80,000. The five grey polygon contours with blue fill inside are the corresponding contours on raster chart 13286 of Portsmouth Harbor, NH. Compared to the green contours, the grey contours (except the one on the upper left corner) are larger and each grey polygon covers several green polygon contours. This figure shows how the cartographers exaggerate each polygon and aggregate neighbor polygon when they get too close.

All polygons have the same depth value of 30 feet. During the generalization, all contours can be aggregated into one large polygon. Figure 4-15 shows the input data. Figure 4-16 to Figure 4-19 show the generalization process.

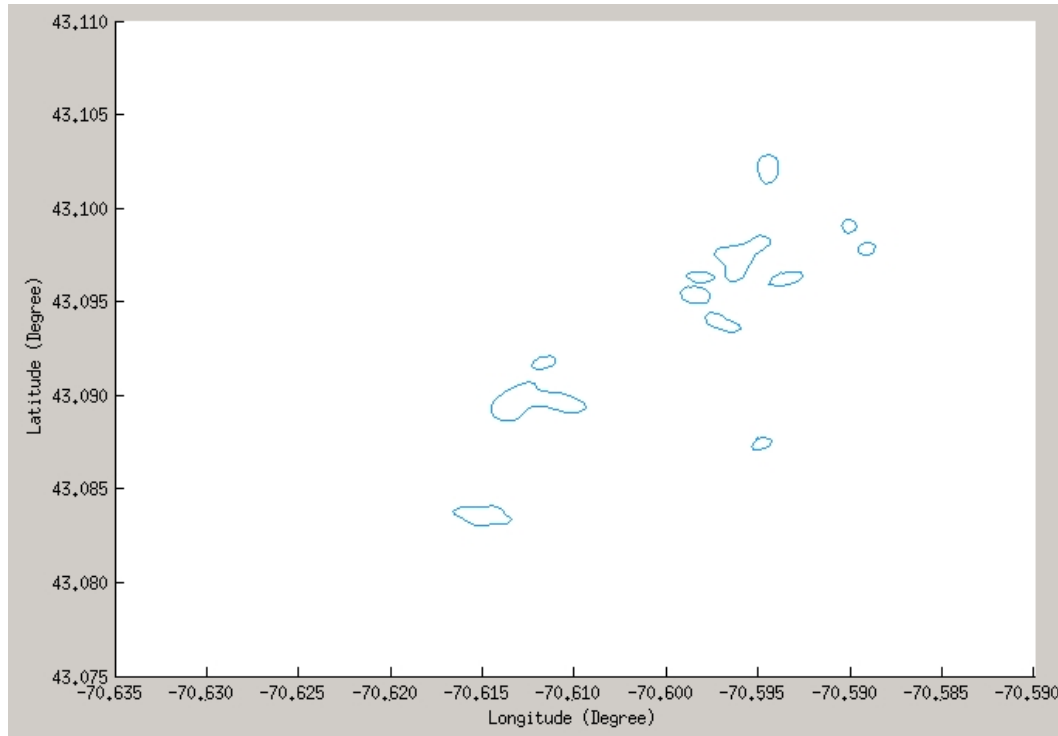


Figure 4-15: Input of a set of simple polygon contours

All polygon contours have depth value of 30 feet. Polygons of the same depth will be aggregated in the following generalization process.

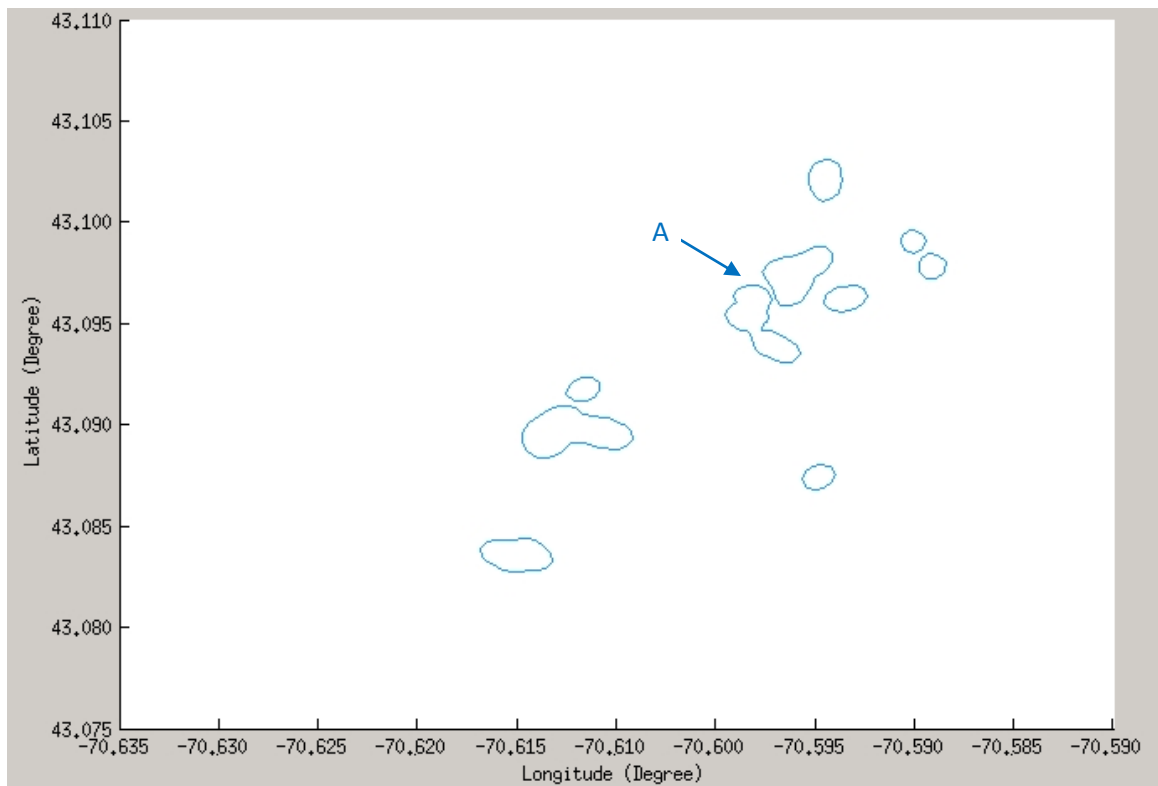


Figure 4-16: Intermediate stage of the generalization process at 18th iteration

In the generalization process, all polygon contours have been exaggerated; two polygon contours have been aggregated (arrow A).

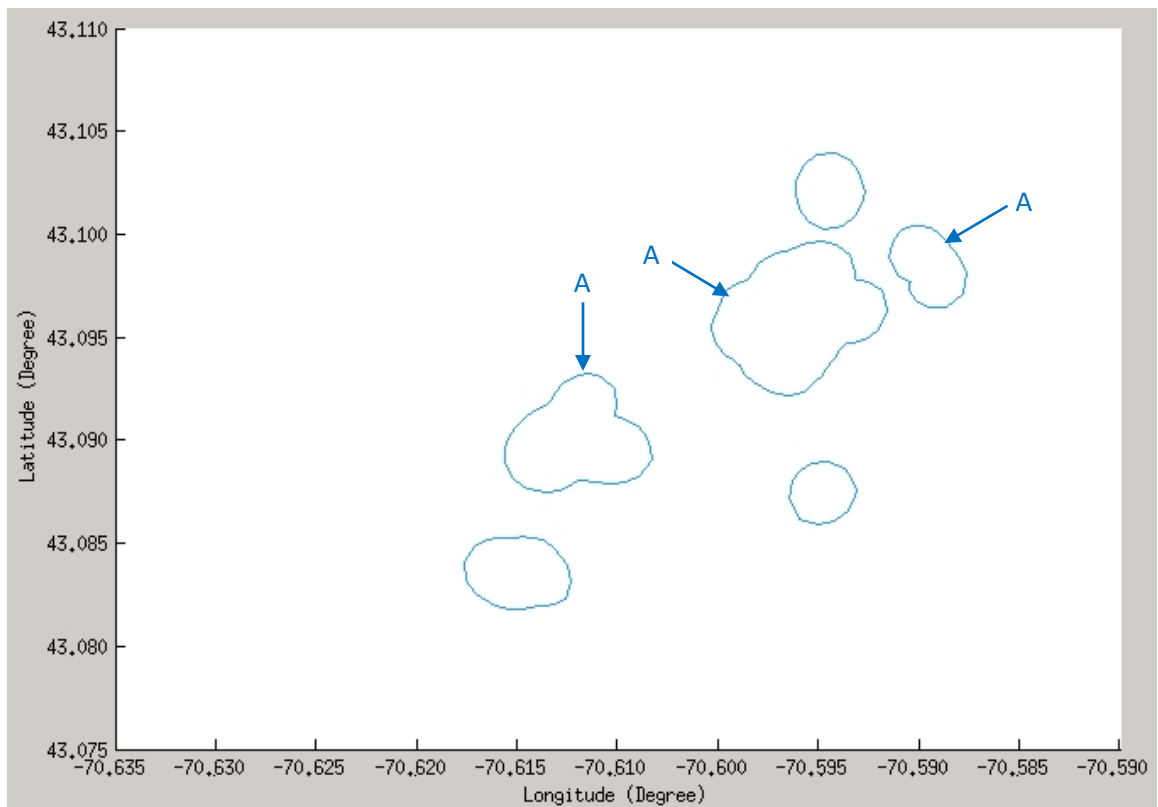


Figure 4-17: Intermediate stage of the generalization process at 68th iteration

All polygon contours are exaggerating; two more polygon contours have been aggregated (arrows A). The aggregation process is gradual, and no self intersection has happened in the generalization process.

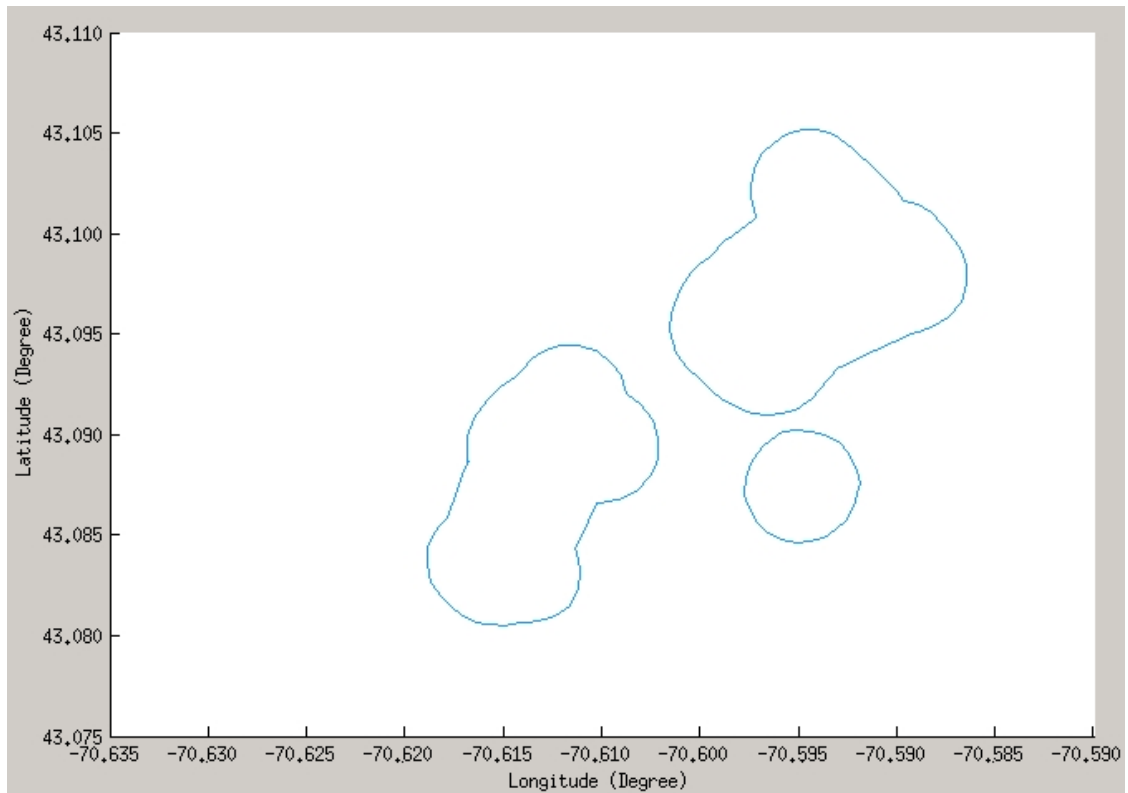


Figure 4-18: Intermediate stage of the generalization process at 140th iteration

All polygon contours are exaggerating; five polygon contours have been aggregated, resulting in two polygons. As the generalization goes further, the size of the polygons increases, although they look relatively large in this figure, but as their scale decreases, so the actual size that they will be displayed at in the chart is smaller.

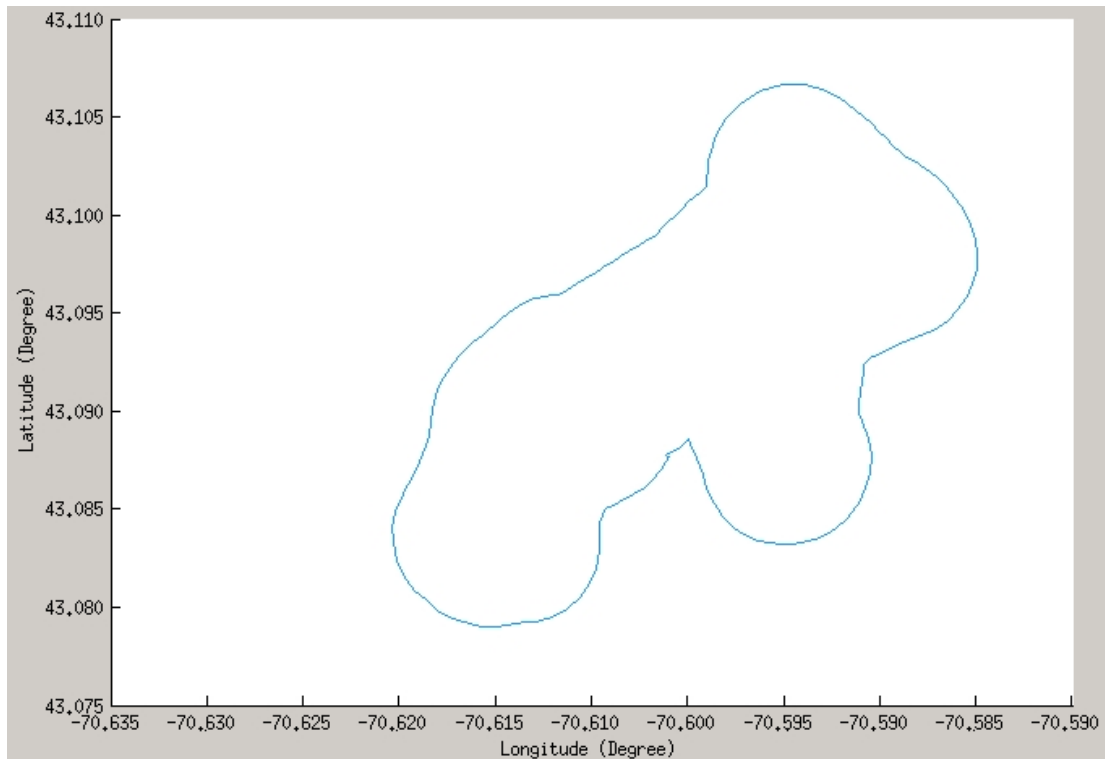


Figure 4-19: Generalization result of test scenario 3 at 244th iteration

All polygon contours with 30 foot depth are aggregated into one polygon. The aggregation stops when all contours at the same depth are aggregated into one polygon. The stopping condition can be altered to use other criterion according to different generalization purposes.

The intermediate status result in Figure 4-18 is very similar to the cartographers' manual process of generalization in Figure 4-14: all polygons are exaggerated, and neighbor polygons are aggregated into one large polygon. The generalization stops when the distance between the 60 foot contour (green) and the one large 30 foot contour (blue) reaches the minimum neighbor distance. The final generalization result in figure 4-19 is a further generalization based on the intermediate status of Figure 4-18. All polygons with 30 foot depth are aggregated into one.

4.4 Test Scenario Four: Concentric Polygon Contours Exaggeration and Aggregation

This section shows the generalization result of a group of polygon contours where some of the polygons are concentric polygon contours. The operators used in this example are the same as in test scenario 3, but the geometry type of the input polygon features is different. In this case, the contour set is not just a group of simple polygon contours, but includes concentric polygon contours. The algorithms for aggregating simple polygon contours and aggregating concentric contours are different (as illustrated in Algorithm 3-3).

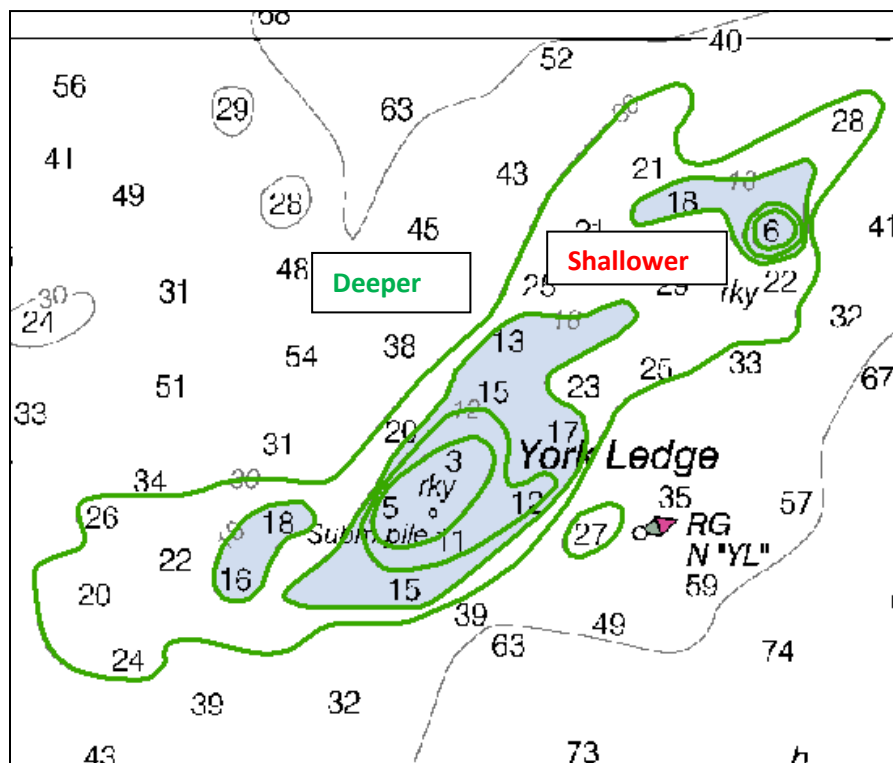


Figure 4-20: Input polygon contours on a background of raster chart 13283

Nine polygon contours are selected from the ENC US5NH02M of Portsmouth Harbor area, NH; eight of them are concentric contours. This example tests the aggregation operator for the concentric polygon contours group.

In this test scenario, some of these contours have the same depth value, while some have different depth values. During the generalization, each polygon contour exaggerates and increases

its size. When two neighbor contours get too close, and if they have same depth value, they will be aggregated. Figure 4-20 shows the original input data with the background of the 1:20,000 scale raster chart (13283).

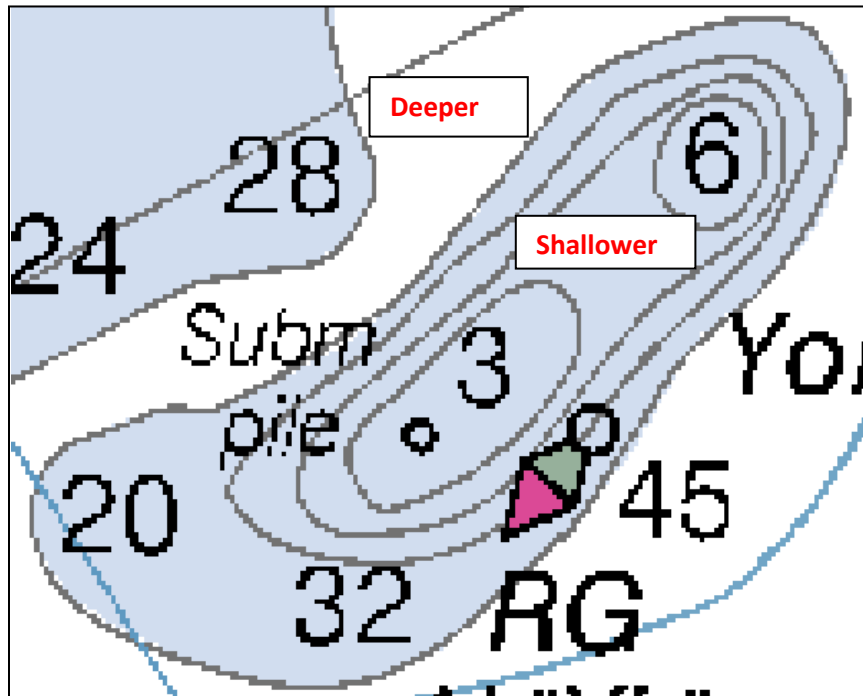


Figure 4-21: Selected area on raster chart 13286

These are the contours of the same area as Figure 4-21 on a 1:80,000 scale chart using manual generalization. Compared to Figure 4-20, all contours have smoother and simplified shapes. Contours with same depth are aggregated into one polygon contour.

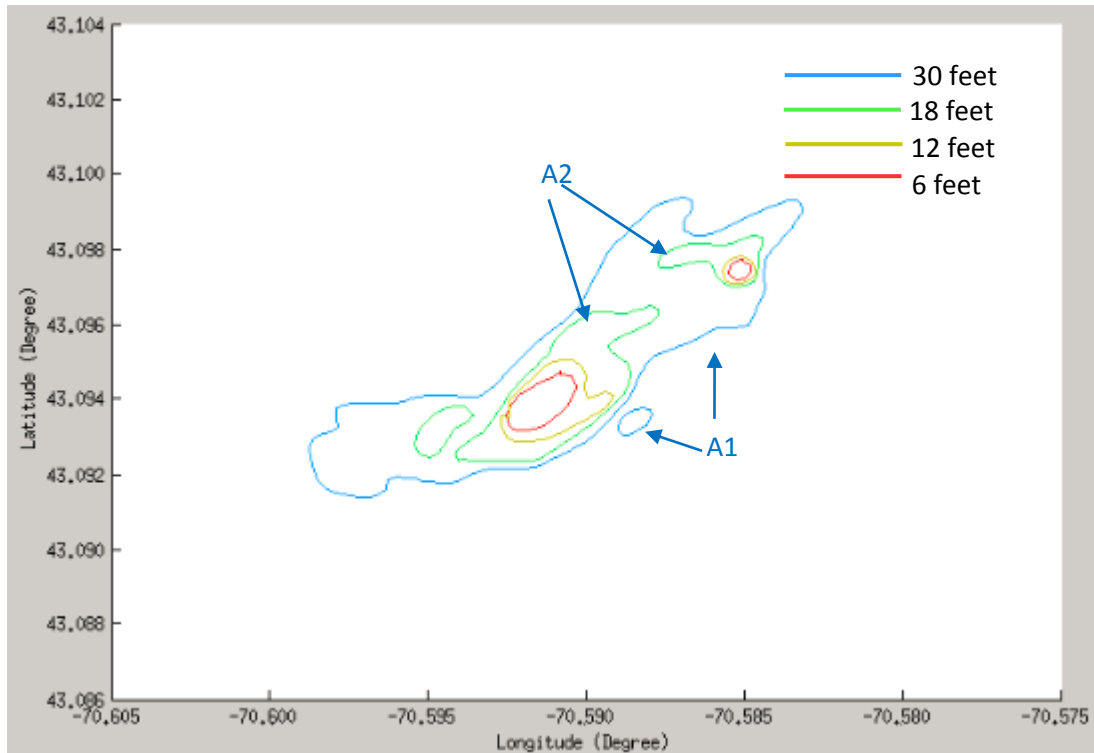


Figure 4-22: Input concentric contours in depth coded color at iteration 0

The input contours have varied depths from 30 foot to 6 foot. In the following generalization process, only contours of the same depth can be aggregated, while all contours are exaggerated. The green polygons (A2) and the blue polygons (A1) will be aggregated in Figure 4-23.

During the generalization, all contours will be exaggerated. If two neighbor contours get too close, and they have the same depth value, they will be aggregated; if they have different depth value, they will not be aggregated, and a minimum distance will be maintained between them.

Figure 4-23 to Figure 4-24 shows the gradual generalization process.

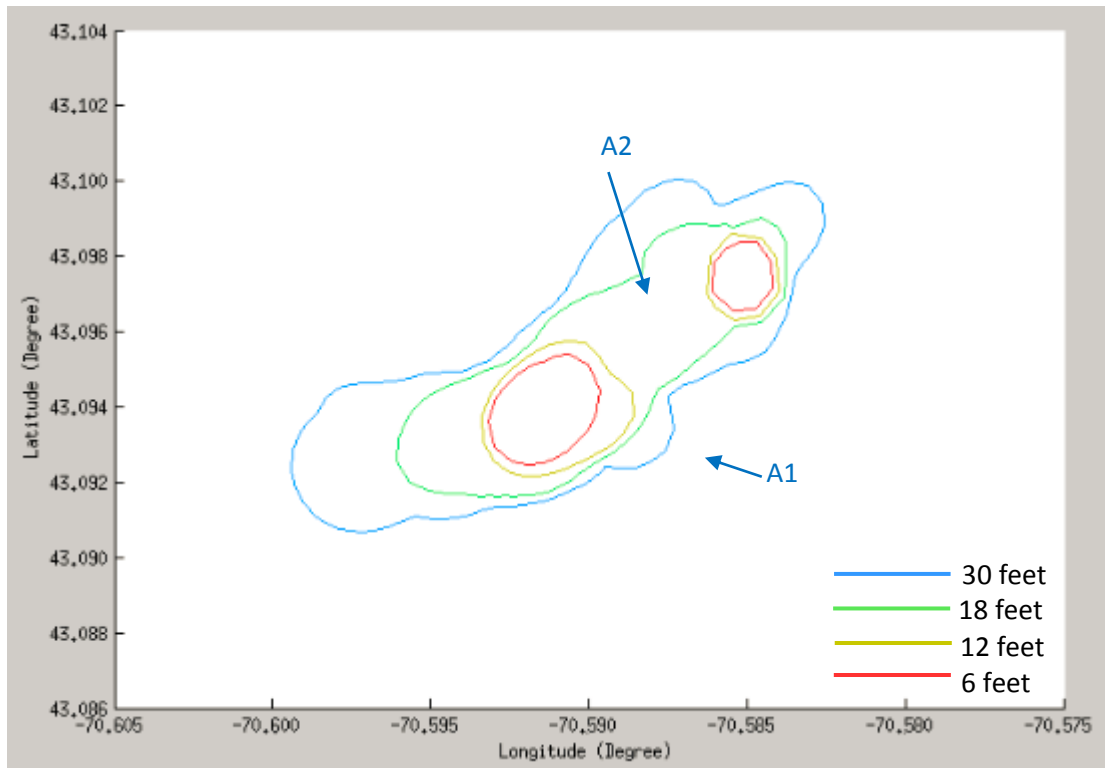


Figure 4-23: Intermediate stage of the generalization process at 44th iteration
The two 30 foot contours aggregated in to one large polygon (A1). Two 18 foot contours are also aggregated into one large polygon (A2). All polygons are exaggerated.

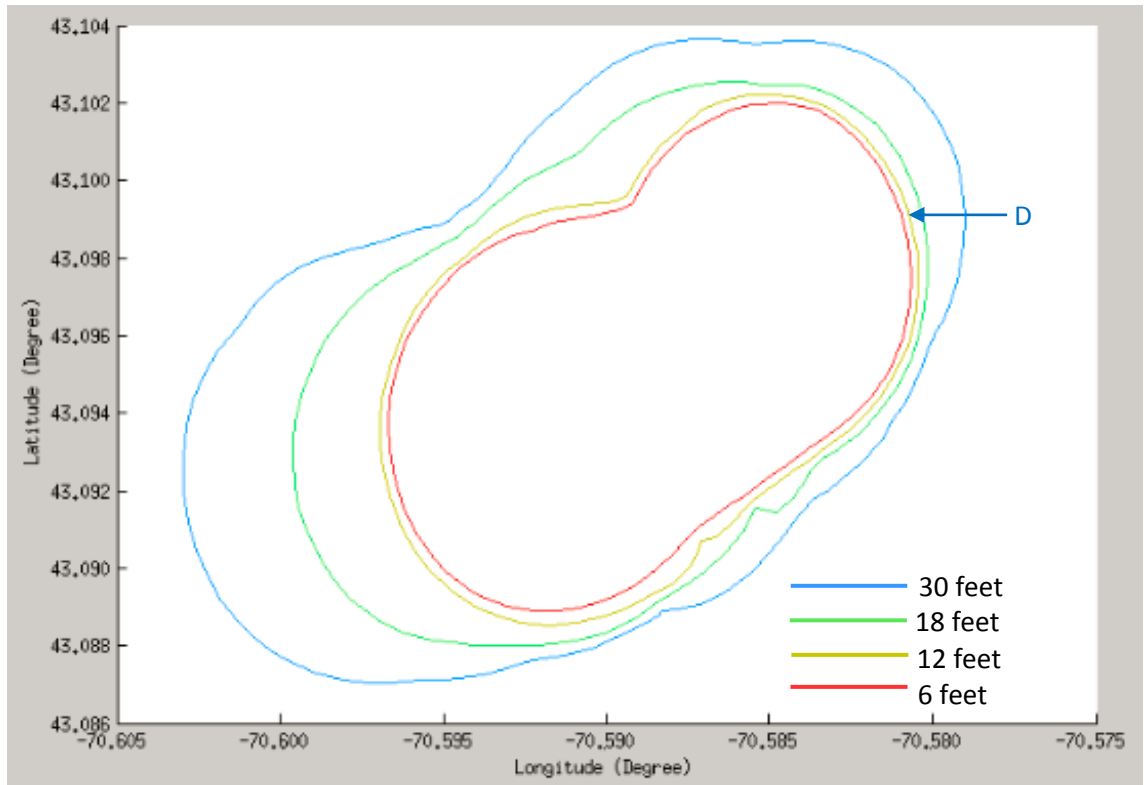


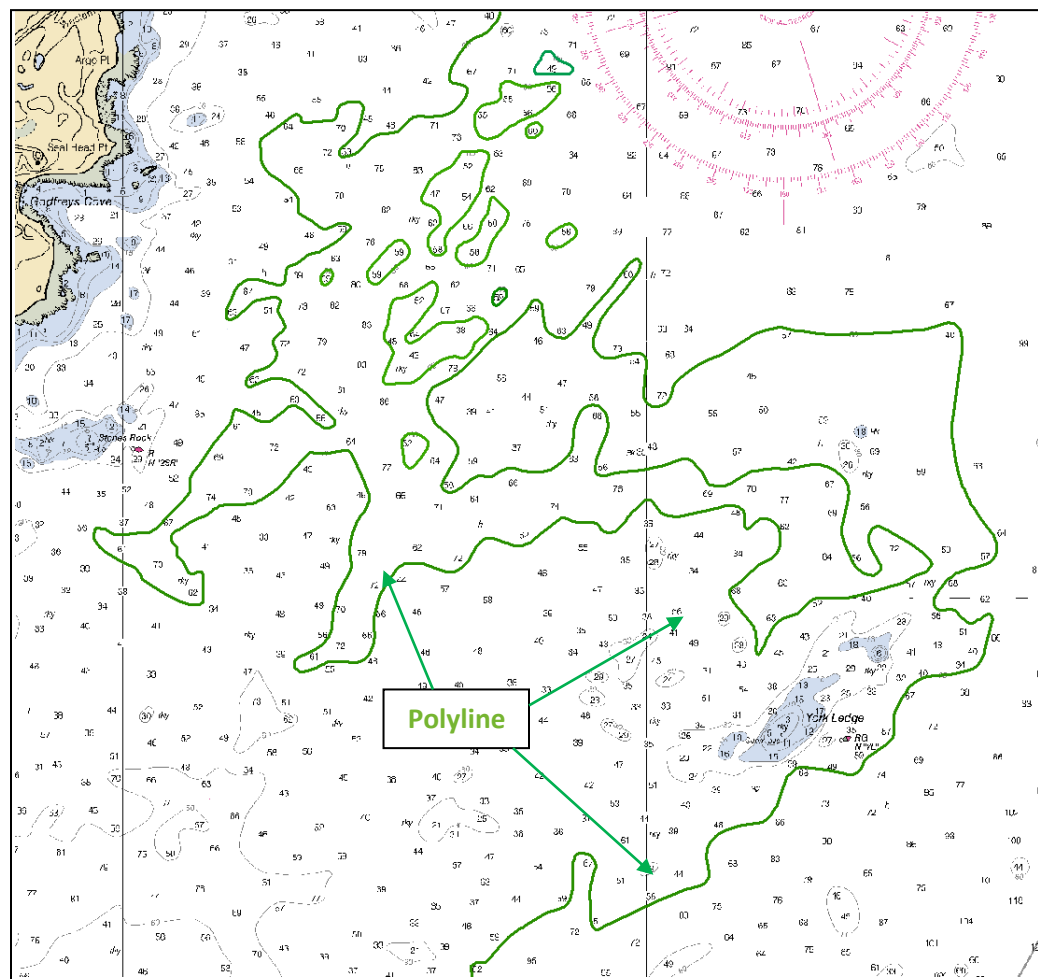
Figure 4-24: Generalization result of test scenario four at 251st iteration

All contours with same depth are aggregated; minimum distances between contours with different depth are maintained (arrow D). The generalization stops when a particular iteration number is reached. If no stopping point is set, the contours can keep exaggerating to infinitely large size. For different generalization purposes, other stopping criteria can be set.

Compared with Figure 4-21, the generalization result of this algorithm (Figure 4-24) is similar to the cartographers' manual process of generalization. The outline of the polygon contour is simplified and smoothed, the shapes of the contours are enlarged, and contours with the same depth are aggregated.

4.5 Test scenario Five: Polyline and Polygon Contours Exaggeration and Aggregation

This section shows the generalization result of one complex long polyline and a set of single polygon contours. It is a scenario combining all previously discussed operators and the input data is more complex than previous scenarios. It is closer to a real generalization problem. All contours have the same depth value of 60 feet. Figure 4-26 shows the input data. Figure 4-28 to Figure 4-34 shows the generalization process.



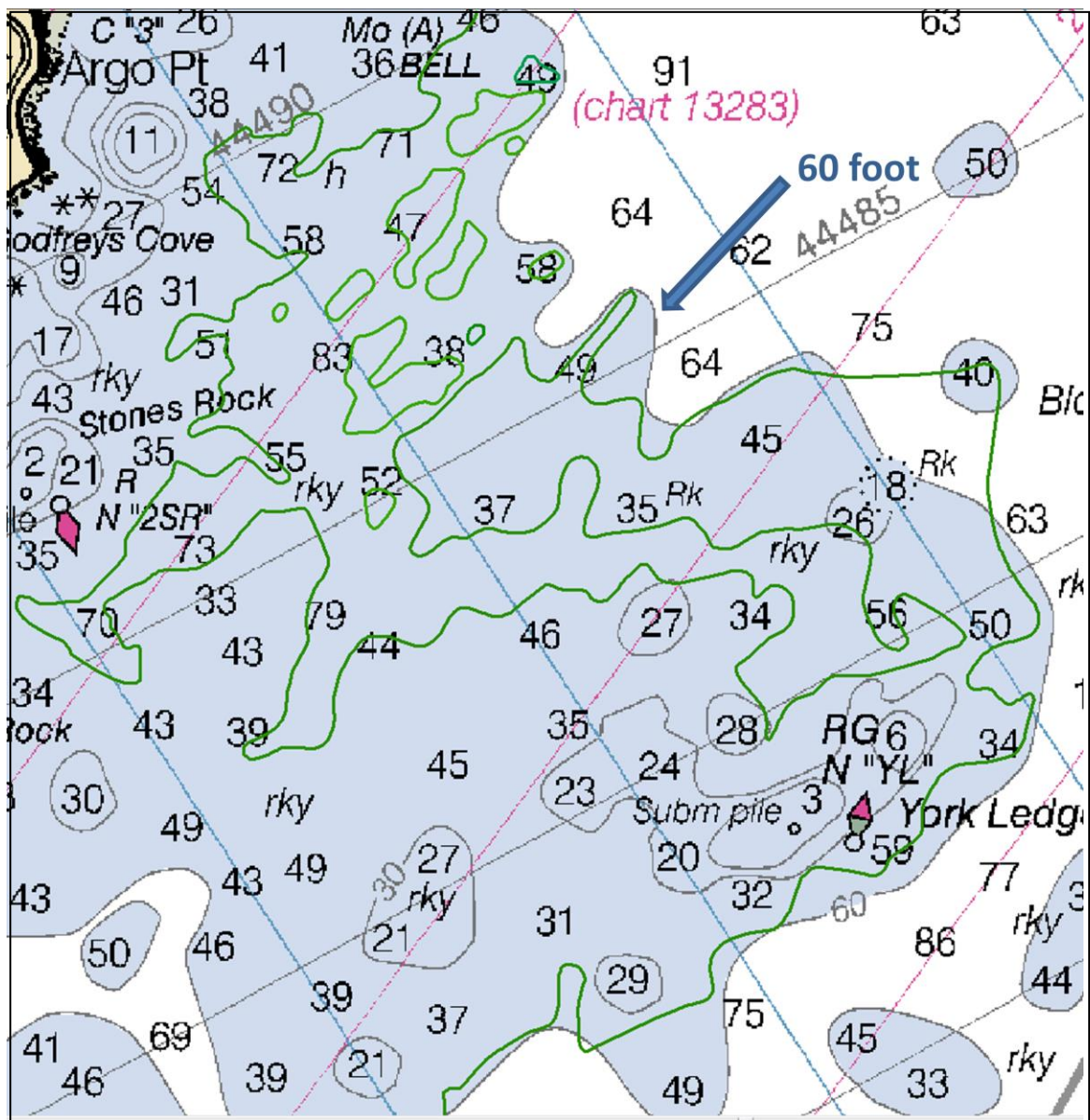


Figure 4-26: Input data on the background of 1:80,000 scale raster chart

The grey line (indicated by the blue arrow) is the 60 foot contour on the 1:80,000 scale raster chart 13278. The 60 feet contour of 1:80,000 scale raster chart is on the deeper side of all input polyline and polygon contours, and its shape is much more simplified and smoothed. Note: the pink label "chart 13283" means for detail information at this area, refer to chart 13283.

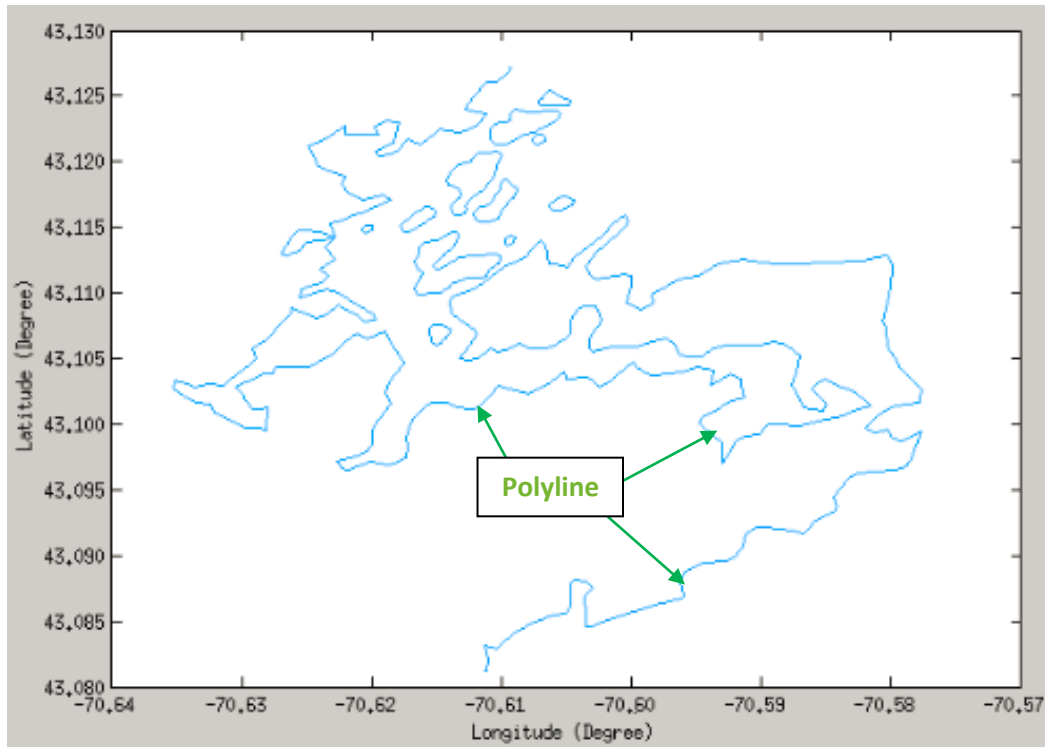


Figure 4-27: Initial input data at iteration 0

In this test scenario, the input data is one long polyline and a set of single polygon contours. The polygons are all on the deeper side of the polyline contour. The depth value of all these contours is 60 foot. In this example, all polygon contours will exaggerate their shape, and aggregate with each other when they are close. The polyline will move to the deeper side, and aggregate the polygon features when it gets close to the polygon contours.

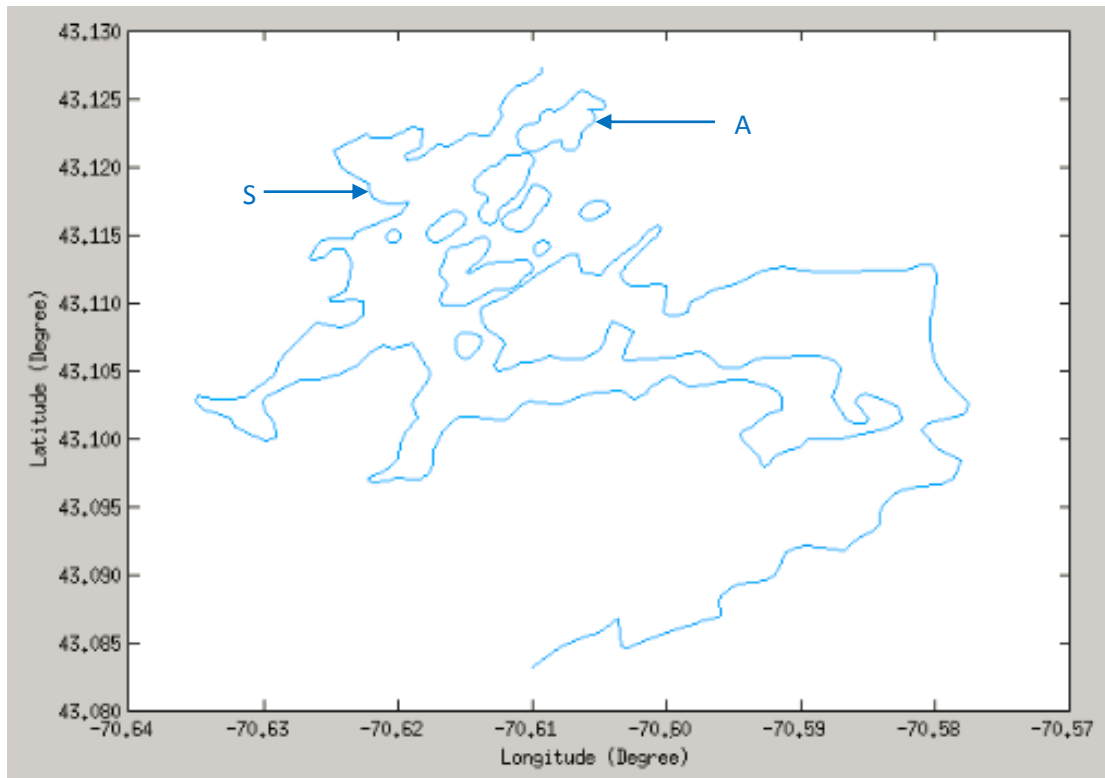


Figure 4-28: Intermediate stage of the generalization process at 11th iteration

The polyline contour is smoothed and simplified and moved to the deeper side (S); all polygon contours exaggerate their shape; the neighbor polygon contours are aggregated (A).

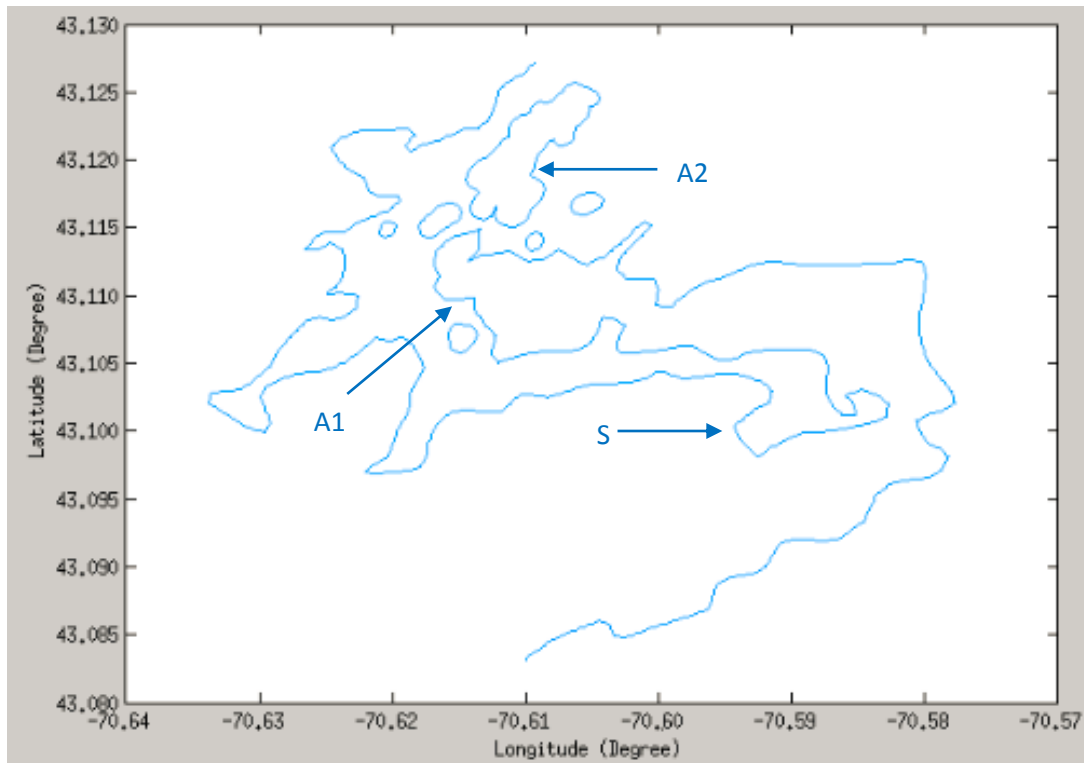


Figure 4-29: Intermediate stage of the generalization process at 17th iteration

In this test scenario, aggregation not only occurs between pairs of polygon contours, but also during the generalization, when a polyline moves too close to a polygon (A1). A2 shows two polygons' aggregation. Besides aggregation, all polygon contours are exaggerating their size, and the polyline is smoothed and simplified (S).

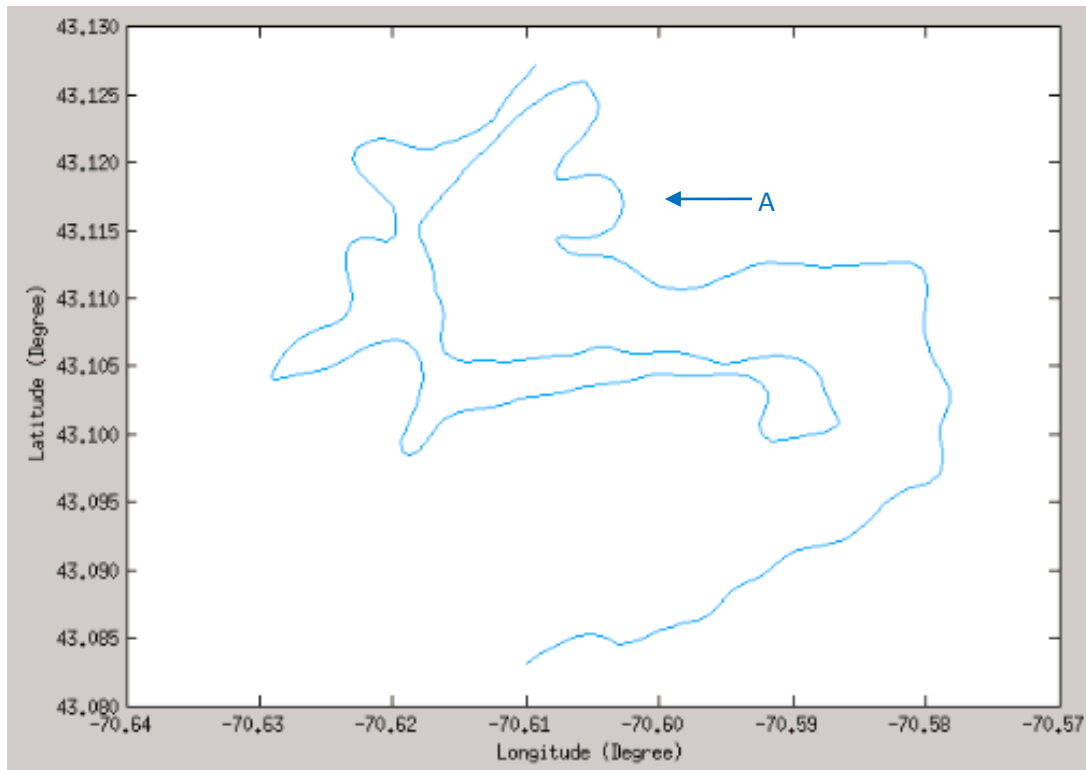


Figure 4-30: Intermediate stage of the generalization process at 150th iteration

Further aggregation of the polyline and polygon. Polygons are aggregated (A). The generalization process consists only of smoothing, simplifying and moving the polyline contour in the sea-ward direction from this iteration forward.

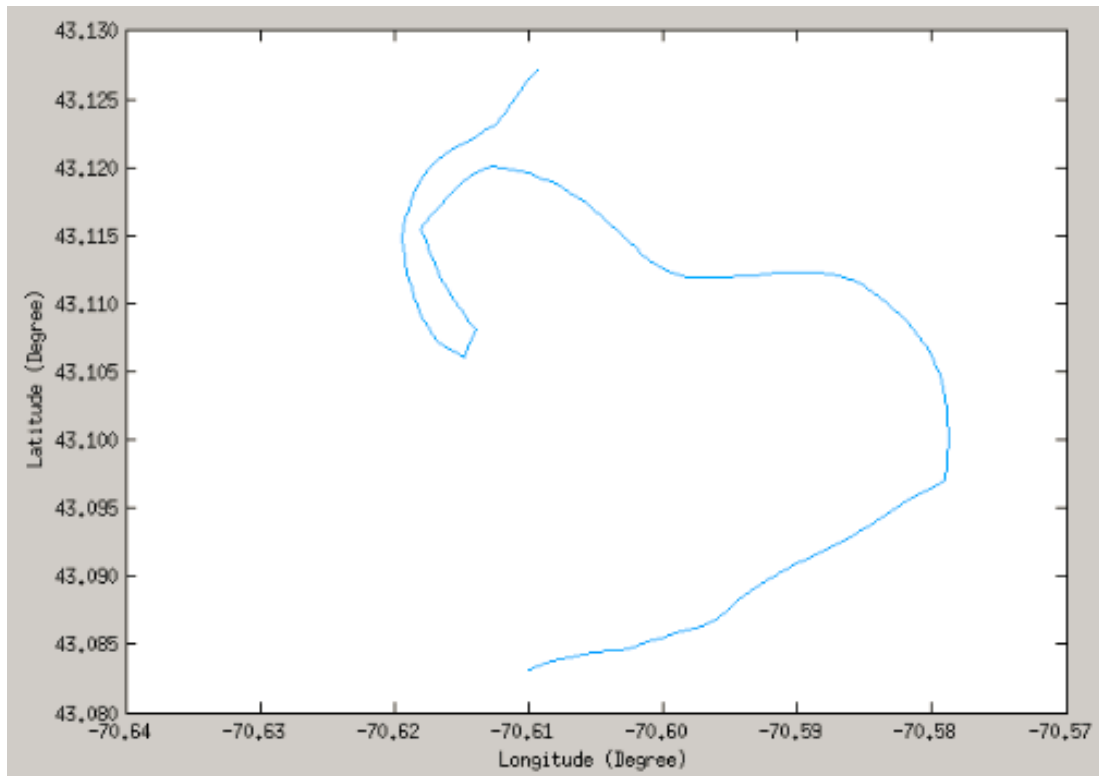


Figure 4-31: Intermediate stage of the generalization process at 900th iteration

After all polygon contours are aggregated and deleted, the generalization is only applying the simplification, smoothing and shoal-bias operators. The polyline gets smoother and simpler.

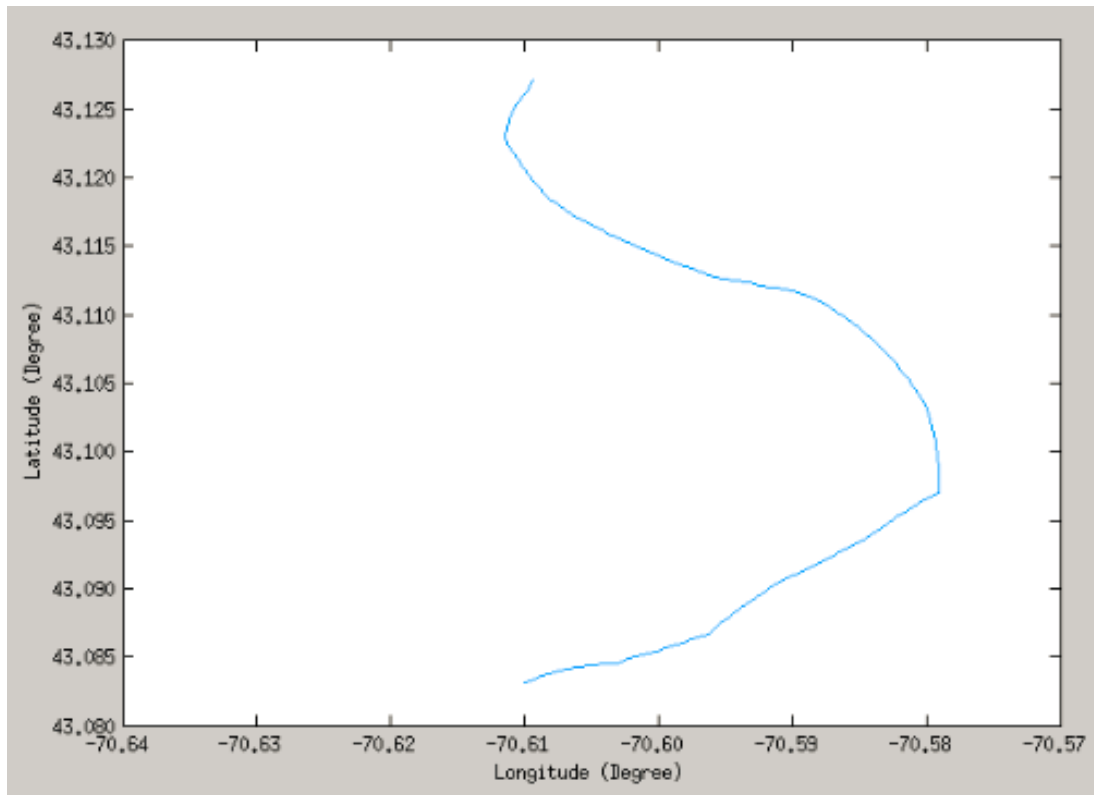


Figure 4-32: Generalization result at 1600th iteration

The polyline is smoothed and simplified and moved to the deeper side of original input contours. Similar to the previous example case, the generalization stops when an iteration number reached.

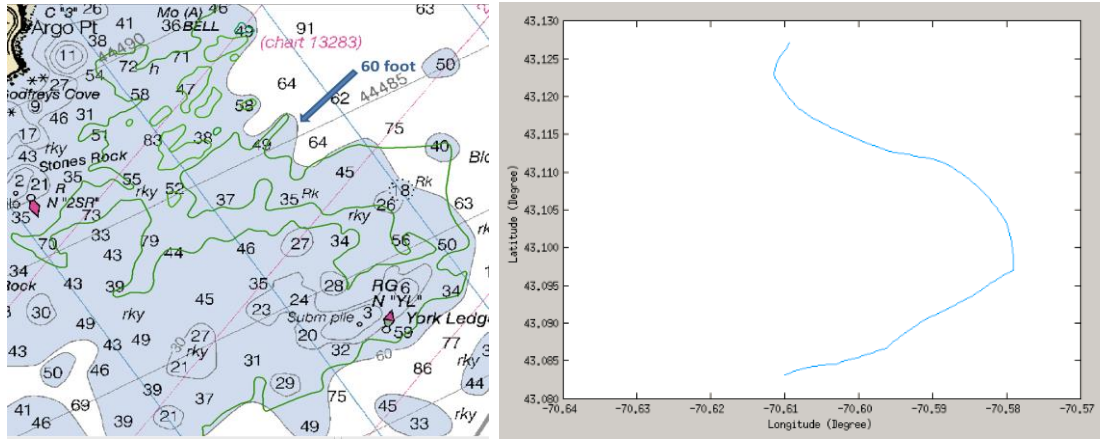


Figure 4-33: Generalization result comparison

The left sub-figure is the same as Figure 4-26, the grey polyline pointed by the blue arrow in the left sub-figure is the generalized 60 foot contour by a cartographer; the right sub-figure is the same as Figure 4-32, the blue line in the right sub-figure is the generalized contour using the algorithms developed in this thesis.

Figure 4-33 shows, compared to the cartographers' manual process of generalization result in Figure 4-27, the algorithm's result simulates the generalization process of deleting all polygon contours and moving the polyline to the deeper side of input contours.

CHAPTER V

DISCUSSION AND FUTURE WORK

5.1 Discussion

This thesis has demonstrated several algorithms implementing the smooth, simplify, exaggeration, and aggregation operators, and obeying the shoal-bias constraint of chart contour generalization. It developed workflows to combine these operators to create gradual generalization of a group of contours, such that there are continuous intermediate stages from the start of the generalization to the final result of generalization. This study is useful because current generalization processes only produce results at certain scales, and there are no representations available for intermediate scales. This study provided a framework and implementation methods to generate gradual continuous generalization with five different contour combination scenarios. The contour features from these five scenarios are generalized from large scale to small scale without creating any intermediate features that require special processing to resolve.

In the smoothing and simplification operators, the starting and ending points keep their position, and only interior vertices of polyline contours are moved. This is acceptable because, in real charts, a polyline contour's start and end points are usually on the edge of the chart. When doing the generalization, it is reasonable to keep those two points stable. They need to match up with the extension of the contour of the adjacent chart.

In this study, there is no clear stopping point for the generalization. This means that if the iteration is continued, the polygon contours can be exaggerated to an infinitely large polygon. In a future study, a stopping criterion could be set to the area of the polygon with respect to the desired scale value and the distance to the neighbor polyline features. A scale value could be added to the stopping criterion. The minimum distance between neighbor features can be

calculated from the visible threshold of distance between two lines divided by the scale.

During the generalization process, points are constantly added to the contours in order to maintain the accuracy of aggregation, exaggeration and smoothing operators. However, at the end of the generalization, the contours can be examined, and the redundant points can be deleted, such that the total number of points is reduced after the generalization.

The operators developed in this study can be applied to different input contours, and the B-spline Snake method, aggregation, and exaggeration operators create result similar to manual generalization. The different scenarios represent some conditions that might occur in chart generalization. However, there are more complex conditions in chart generalization, and more operators and workflows that stimulate more generalization scenarios can be developed based on this study to address them.

5.2 Future Work

Future research on this topic could include the following options:

- The further study can develop a deletion operator for the features that need to be deleted in the generalization process, as the deletion is more of a model generalization instead of the graphic generalization this thesis is focused on. Beside, a workflow for the complex input feature conditions can also be created. In this study, only five examples are tested, but there are other conditions which might occur in real chart generalization that have not been included in this study: the complex condition when a polyline, a group of concentric polygon contours, and several polygon contours on both shallow and deep sides of the polyline contour.
- There might be a relationship between the parameters and the geographic size of input data, which could be investigated further. The functions developed here use parameters that are set to certain numbers that work best for the test data used here. However, the parameters have

not been developed for universal usage, so for some other input contour data that are significantly different than the test data used here (much longer polyline contour, or much larger polygon contour), the parameters in the functions might vary.

- The curvature calculation in this study does not calculate with the second derivative, but in future work, the second derivative of the contour B-spline curve could be used to calculate the curvature, such that the curvature value will be more accurate.
- How to relate the exact scale number with the iteration step: this study only generates the continuous intermediate status of the generalization; however, it did not indicate which scale those intermediate states are. Future study could focus on making the calculation more scale related.

LIST OF REFERENCE

- D. Burghardt. 2005. "Controlled Line Smoothing by Snakes." *GeoInformatica* 9:3, Pages 237-252.
- L. D. Cohen, Issac Cohen, 1993. "Finite Element Methods for Active Contour Models and Balloons for 2D and 3D Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-15 November.
- D. H. Douglas, T. K. Peucker. 1973. "Algorithm for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature." *The Canadian Cartographer*.
- E. Guilbert, and Hui Lin. 2007. "Isobathymetric Line Simplification with Conflict Removal Bases on a B-spline Snake Model." *Marine Geodesy*, 20 :169-195.
- E. Guilbert, E. Saux. 2008. "Cartographic Generalisation of Lines Based on a B-spline Snake Model." *International Journal of Geographical Information Science*, Vol 22, No. 8, August, 2008
- E. Guilbert, E. Saux, and M. Daniel. 2006. "Conflict Removal between B-spline Curves for Isobathymetric Line Generalization Using a Snake Model." *Cartography and Geographic Information Science*, Vol. 33, No.1 Pages 37-52.
- L. Harrie. 2001. "An Optimisation Approach to Cartographic Generalisation." Doctoral Thesis, Department of Survey, Lund Institute of Technology, Lund University.
- M. Kass, A. Witkin, and D. Terzopoulos, 1987. "Snakes: active Contour Models", *International Journal of Computer Vision*, 321-331.
- Z. Li, S. Openshaw. 1993, "A Natural Principle for the Objective Generalization of Digital Maps." *Cartography and Geographic Information Systems*, Vol. 20, No. 1.
- W. A. Mackaness, Anne Ruas. 2007. *Generalisation of Geographic Information: Cartographic Modeling and Applications*.
- W. Peng. 2000. "Database Generalization: Concepts, Problems, and Operations." *International Archives of Photogrammetry and Remote Sensing*. Vol. XXXIII, Part B4, Amsterdam 2000.
- R. B. McMaster, K. S. Shea. 1992. "Generalization on Digital Cartography."
- NOAA, 1996. "Technical Issues in NOAA's Nautical Chart Program." National Academy Press, Washington, DC, 1996.
- NOAA, 1997. "The Nautical Chart User's Manual" U.S. Department of Commerce, National Oceanic and Atmospheric Administration (NOAA), National Oceanic Service, Washington, DC, 1997.
- NOAA, NOS website http://www.nauticalcharts.noaa.gov/mcd/learn_diffRNC_ENC.html

- K. Stuart Shea. 1988. "Cartographic Generalization." NOAA Technical Report NOS 127 CGS 12.
- K. Stuart Shea, Robert B. McMaster. 1989. "Cartographic Generalization in a Digital Environment: When and How to Generalize".
- S. Steiniger, S. Meier. 2004. "A Technique for Line Smoothing and Displacement in Map Generalisation." ICA Commission on Generalisation and Multiple Representation Workshop, Leicester, Aug. 2004.
- Z. Wang, J-C Muller. 1998. "Line Generalization Based on Analysis of Shape Characteristics" Cartography and Geographic Information Systems, Vol. 25, No.1.
- J. Mark Ware. 2003. "Automated map generalization with multiple operators: a simulated annealing approach." INT. J. Geographic Information Science, Volume 17, No. 8, Pages 743-769.
- C. Xu, and J. L. Prince, 1998. "Snakes, shapes, and Gradient vector Flow" IEEE Transactions on Image Processing, 7(3) 359-369.

APPENDICES

APPENDIX A

PARAMETER DISCUSSION

A.0 Distance Unit Definition

The unit of the distance used in this thesis is not meters or feet. The original coordinates of input contours are decimal degrees in latitude and longitude, and they are processed before used in the calculation in this thesis work. The conversion and the unit of the numbers used in this thesis are defined as follows:

Example input coordinate: (-70.6094, 43.1143), the coordinate is decimal degree in geographic coordinates, and the unit here is degrees. Then multiply both x and y coordinate by 100,000, so the coordinate becomes (-706094, 431143), and the unit is 1/100,000 degree. All the following calculations use these coordinates. If there are no specific notifications of the unit, all the numbers and parameters of distance discussed in this thesis are using the unit of 1/100,000 degree.

A.1 Parameter in Algorithm 3-2

In algorithm 3-2, in the pseudo code for finding supporting segments, S_1 's index is calculated by the index of $P_{polygon}$ plus N , E_1 's index is calculated by the index of $P_{polygon}$ minus M . N and M are parameters used to determine the index of the starting and ending vertex S_1 and E_1 of the supporting segment on the polygon. As all polygons' indices are set to increase clockwise, and the polygon is to the right of polyline, the start index S_1 is always on the clockwise side of vertex $I_{polygon}$, and E_1 is on the counter clockwise side of vertex $I_{polygon}$. The N and M are set to create a small distance between S_1 and E_1 . Basically, S_1 is defined as $I_{polygon}$

plus a small number, and S_1 is defined as $I_{polygon}$ minus a small number, however, the end vertex might be within the range of these small number, so the end situations are processed separately by assigning the end vertex (length of the polygon) or start vertex (1) to S_1 and E_1 . The N and M are defined as follows:

$I_{polygon}$ is the index of the vertex in the polygon that is closest to the polyline;

len is the total number of vertices in the polygon;

If $I_{polygon} < len - 3$

$$N = 3, S_1 = I_{polygon} + 3$$

Else

$$S_1 = 1$$

If $I_{polygon} > 2$

$$M = 2, E_1 = I_{polygon} - 2$$

Else

$$E_1 = len$$

All polygon contours that have been used in the polyline-polygon sample test scenarios have total numbers of vertices are mostly in the range of 20 to 50. In addition, in the preprocessing for the polygon before generalization starts, vertices are processed such that they are mostly evenly distributed, meaning that 2-3 points are about 4% to 10% of the total length of the polygon. The purpose to finding support segments for aggregation is to create two appropriate segments that connect the polyline and polygon, such that the aggregated feature will not have sharp angles, or intersect with itself. After testing with several other numbers, 2-3 samples turned out to best generate the support segments. They are small steps along the polygon contour, such that the basic shape of the polygon can be maintained. If N and M are too small, for example one sample, then the distances between S_1 and E_1 will be just two samples, that is too close, and the aggregation result will look unnatural; If N and M are set to too large a number, then line S_1S_2 and E_1E_2 will intersect with the polygon, so S_1 and E_1 will still be

replaced with the intersection points.

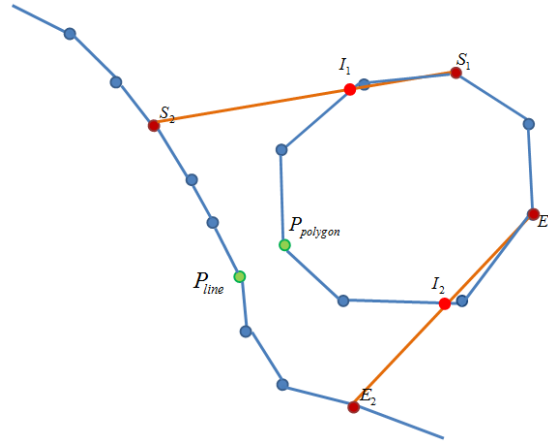


Figure A.1: Location of vertices in algorithm 3-2

A.2 Parameter in Algorithm 3-4

In Algorithm 3-4, in the pseudo code for finding supporting segments for the aggregation of two polygons, if the total number of vertices of the left polygon is larger than 20, S_2 's index is the index of P_{C_2} minus N_1 , E_2 's index is the index of P_{C_2} plus N_1 ; otherwise, if the total number of vertices is less than 20, S_2 's index is the index of P_{C_2} minus N_2 , E_2 's index is the index of P_{C_2} plus N_2 . N_1 , N_2 are defined as follows:

$I_{polygon}$ is the index of the vertex on the first polygon that is closest to the other polygon, len is the total number of vertices on the left polygon.

If $len > 20$

$$N_1 = 10, S_2 = I_{polygon} - 10, E_2 = I_{polygon} + 10$$

ElseIf $len \leq 20$

$$N_2 = 3, S_2 = I_{polygon} - 3, E_2 = I_{polygon} + 3$$

As in the two polygon aggregation test scenario, the input polygon contours are mostly of

medium size, with total numbers of vertices mostly in the range of 50 to 140. Therefore an offset of 10 samples is used here, for reasons similar to those of selection of offset in section A.1. However, in this test scenario, there are a few small polygon contours that have less than 20 vertices, and for these polygon contours, as they are considered small polygons, an offset of three samples is used, for reasons similar to those in section A.1 for smaller polygons.

The reason why 20 is chosen here to separate the polygons is as follows: most input polygons are in the size of 50 to 140 vertices, there are a few small polygons that have less than 16 vertices, and here 20 is used as a threshold to separate the polygons.

A.3 Parameter in Algorithm 3-13

In step two of Algorithm 3-13, a new contour is created that has total vertices of about 80% of the original curve points. There are two reasons why 80% is picked: one is because the purpose of this step is to simplify the contour, so the fewer vertices the better. The second reason is if there are too few vertices, the new curve will not be able to depict the basic shape and some of the significant detail of the original curve. Several other percentages (50% to 100%) were tested, but a figure of 80% yielded the best result, depicted the basic shape and the significant detail of the original with fewer vertices, and it is a relatively subjective process to determine whether the basic shape and significant detail is well depicted or not. For interpretation purposes, the number of samples closest to 80% of the total is used as a sample where a fractional sample count would be practical.

A.4 Point Adding Parameter in Algorithm 3-14

In step 2 of Algorithm 3-14, points are added to the polygon, such that the distance between each point is smaller than 1/100 of the distance between the furthest two points of that closed polygon. The reason for choosing this number is because the purpose of adding points to the polygon is to make the polygon have more evenly distributed vertices, such that in the following aggregation step, when finding the support segment, if the distance between neighbor vertices is very large and uneven, then the result of aggregation will look unnatural (Figure A.4).

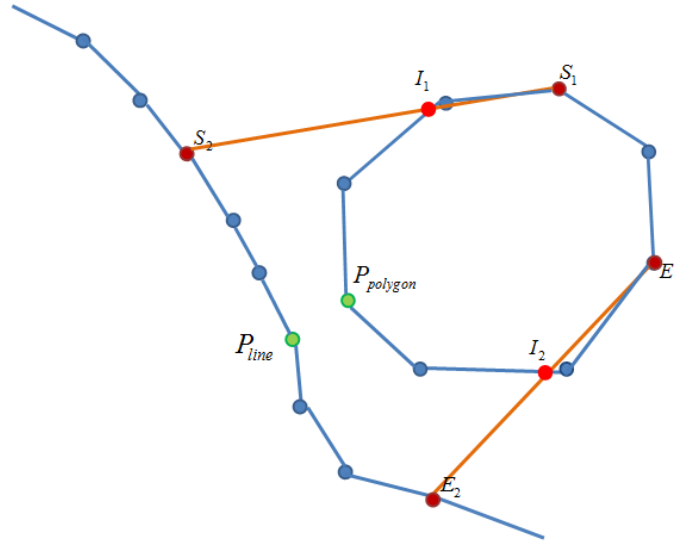


Figure A.4: Location of vertices in Algorithm 3-14

E_2 is two points from $P_{polygon}$, and S_1 is two points from $P_{polygon}$ too, the support segment S_1S_2 and E_1E_2 should ideally be same length and look symmetric to each other. However as the vertices of polygon are not evenly distributed, E_1 is much closer to $P_{polygon}$, and E_1E_2 is much shorter than S_1S_2 , such that the aggregation result looks unnatural.

The value of 1/100 is used because it is a small value, which makes sure that the polygon get enough vertices on it. Similar to the choice of 80% in Appendix A.4, the 1/100 here is not an absolute value. If the distance between the furthest two points of that closed polygon is 120, then

the distance between each point can be rounded to two, to allow for integer calculation. Other values were tested, but $1/100$ was found empirically to give the most useful result.

In step 3.4, $1/60$ of the length of the current contour is used as the threshold of whether new points should be added to the contour. $1/60$ is chosen empirically from tests with values ranging from $1/100$ to $1/10$. When the value is too small, there are too few points added, so the line will not have enough more points; if the value is too large, too many points are added, which are not necessary for the calculation.

A.5 Neighbor Points Parameter in Algorithm 3-14 and 3-9

In step nine of Algorithm 3-14, the threshold of the distance between two neighbor points on the snake curve is set to be $1/600$ of the current curve length. Similar to the $1/100$ in A.4, the $1/600$ is the lower bound of the neighbor points distance, at $1/6$ of the $1/100$. That is, if there are six points in a normal neighbor vertices segment. That is considered too crowded and points need to be deleted. Six is used based on experiments with different numbers, and generates the best and most stable results: points are deleted and the basic shape of the line is still maintained; no large change of the shape occurred during the generalization.

A.6 Delete Points Parameter in Algorithm 3-14 and 2-1

In step 9.2 and 9.3 of Algorithm 3-14, and step 3.1 of Algorithm 2-1, three continuous points are deleted if their neighbor distances are all smaller than $1/600$ of the total contour length.

Since the purpose of the deletion is to reduce the redundant points in the segments which have six more vertices than a normal segment's vertices (as in section A.5), a target of these samples were chose as half of this range. Other values in the range 1 to 10 were tested, but three

samples gives, empirically, the most stable results during generalization. If a larger number is used, the shape of the contour will be changed more, and the deformation will look unnatural. If value is set too small, the deletion will be not effective enough. .

A.7 Parameter in Algorithm 3-8

In Algorithm 3-8, step 1.4 and 1.10, a value of 15 is used as a threshold to determine whether a step size is used or not. Tests have been done with values ranging from 5 to 50. When using a small threshold value such as 5, very few of the stepsizes that were calculated with the gradient method were used, and the polyline was not generalized in a smooth and shoal-biased fashion. Most points were moved to the middle of its adjacent two points (step 1.5), which is meaningless. However, the gradient method is an approximate way to solve the energy equation; there are numerical spikes in the stepsize value during the calculation. If the threshold is set to a large value like 50, these spikes are used, and the polyline has zig-zag shape during the generalization. In order to delete these spikes and meanwhile maintain a smoothing and shoal-biased generalization result, after tests with different value, 15 turned out to generate the most desirable smooth, shoal-biased result while no spikes occurred during the generalization.

A.8 Parameter in Algorithm 3-10

The minimum distance N in step 2 is set to 10. In principle, the minimal distance should be related to the scale and width of minimal visible line size. Here, in empirical tests, values from 5 to 20 were tested, and 10 yielded a subjectively good result: lines can be seen as separate lines. However, in further studies, this value can be set to a more mathematically and geographically accurate value.

A.9 Parameter in Algorithm 3-11

The minimum distance M in step 1.2 is set to 19. Similar to section A.8, this value 19 is chosen by empirical tests. As in section A.8, the minimum distance here ideally should be calculated by using the scale and minimum visible line width on the screen, however, after tests with values ranging from 5 to 30, 19 gives a good result (19 is the minimum value that ensures that neighbor polygons can be seen as separate polygons). Similar to section A.8, in further studies, this number should be set to a more mathematically and geographically accurate value.

A.10 Parameter in Algorithm 3-13, 3-14, 3-16, 3-17

The iteration number R in Algorithms 3-13, 3-14, 3-16, and 3-17 is set to a large number (usually larger than 1500), so that the iteration can keep going on. The algorithms are stopped manually when the contours are generalized to a level that no significant changes will happen, for example at Algorithm 3-13, that means the single input polyline contour cannot be further smoothed, all curvatures are smoothed, or the curvatures should be kept for shoal-bias purposes. At Algorithm 3-14, that means all polygon contours are aggregated, and the polyline contour is much smoothed, and there are no more curves to be smoothed, or curves should be kept for shoal-biased reasons. For Algorithm 3-16, that means all polygons with same depth have been aggregated, and all left polygons are all very smooth. For Algorithm 3-17, that means all polygons are exaggerated and aggregated, and the polyline cannot be further smoothed.