UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

CONTINUOUS AND ADAPTIVE CARTOGRAPHIC GENERALIZATION OF RIVER NETWORKS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

MOSHE GUTMAN
Norman, Oklahoma
2012

CONTINUOUS AND ADAPTIVE CARTOGRAPHIC GENERALIZATION
OF RIVER NETWORKS

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Dr. Chris Weaver, Chair

Dr. S. Lakshmivarahan

Dr. Dean Hougen

Dr. May Yuan

Dr. Meijun Zhu

Acknowledgments

I wish to thank my advisor Dr. Chris Weaver for his support and encouragement during this project. I am also very grateful to my committee members—Dr. S. Lakshmivarahan, Dr. Dean Hougen, Dr. May Yuan, and Dr. Meijun Zhu—for their time and advice. Finally, a big thank you to my family and friends.

Table of Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Research path . . . 3
- 1.3 Contributions . . . 7
- 1.4 Overview . . . 9

2 Cartographic generalization **13**
- 2.1 Digitization . . . 14
- 2.2 Generalization . . . 15
- 2.3 Simplification, Smoothing, and Refinement . . . 16
  - 2.3.1 Simplification . . . 17
  - 2.3.2 Smoothing . . . 20
  - 2.3.3 Refinement/Pruning . . . 24
- 2.4 Wavelets in cartography . . . 25
  - 2.4.1 Smoothing . . . 25
  - 2.4.2 Other uses for wavelets . . . 28
- 2.5 Multi-resolution databases . . . 30

3 Wavelets **32**
- 3.1 Wavelet concepts . . . 33
- 3.2 Orthogonal wavelets . . . 35
  - 3.2.1 Naming conventions . . . 35
  - 3.2.2 Mask and scaling function . . . 35
  - 3.2.3 Orthogonal systems and expansions . . . 37
  - 3.2.4 Multiresolution analysis . . . 40
  - 3.2.5 *dbN* wavelets . . . 43
- 3.3 Biorthogonal wavelets . . . 45
  - 3.3.1 Naming conventions . . . 45
  - 3.3.2 Masks and scaling functions . . . 45
  - 3.3.3 Convergence of the approximations . . . 51
- 3.4 Discrete Wavelet Transform . . . 51
  - 3.4.1 Elementary signal operations . . . 52

3.4.2 Low-pass and High-pass filters 54

4 Dimensional wavelets and IMP representations 56
4.1 Dimensional scaling functions 57
4.2 IMP functions and signals 59
4.3 Operations with IMP functions and signals 61
4.3.1 Main results 62
4.3.2 Wavelet decomposition of IMP functions 66
4.4 Oversampling level and expansion coefficients 67
4.5 Cascade Algorithm 70

5 Wavelet-based smoothing of linear features 71
5.1 Transformation of a tributary to the IMP form 72
5.1.1 Great-circle distance formula 72
5.1.2 Tributary parametrization 73
5.1.3 Transformation to the IMP form 75
5.2 Wavelet decomposition of a tributary 78
5.2.1 Oversampling 79
5.2.2 Wavelets details of T 80
5.2.3 Wavelet details thresholding 80
5.2.4 Summary of the decomposition stage 81
5.3 Tributary synthesis 82
5.3.1 Finding level J 82
5.3.2 Interpolated frame $\hat{\mathbf{c}}$ 82
5.3.3 Summary of the synthesis stage 83
5.3.4 Preservation of endpoints and continuous smoothing of tributaries 84
5.4 Tributary smoothing analysis 88

6 A method for river network generalization 102
6.1 Preprocessing stage 103
6.1.1 De-cycling of a river network graph 104
6.1.2 Strahler number determination 105
6.1.3 Extraction of tributaries 107
6.1.4 Tributary decomposition 109
6.2 Generalization stage 109
6.2.1 Tributary synthesis 110
6.2.2 Generalized tree formation and restoration of cycles 110
6.2.3 Smooth pruning 113

7 Implementation **117**
7.1 Dataset source . . . 118
7.1.1 Retrieval and preprocessing of the dataset . . . 120
7.2 Graph generation . . . 120
7.2.1 Edges and nodes . . . 121
7.2.2 Tributary extraction . . . 122
7.3 System data structures . . . 122
7.4 Database generation . . . 124
7.4.1 Spline interpolated tributaries . . . 124
7.4.2 Cubic spline interpolation . . . 125
7.4.3 Decomposition . . . 126
7.5 Rendering . . . 127
7.6 Performance . . . 129
7.7 Evaluation . . . 131
7.7.1 Setup . . . 131
7.7.2 Results . . . 133
7.7.3 Analysis . . . 134

8 Conclusions **137**
8.1 Automated smoothing . . . 137
8.2 Future work . . . 140
8.2.1 Research . . . 140
8.2.2 Implementation . . . 141

A Evaluation material **149**

List of Tables

3.1 Biorthogonal family (5–3) 47

3.2 Biorthogonal family (9–3) 47

3.3 Biorthogonal family (13–3) 47

3.4 Biorthogonal family (9–7) (FBI) 48

5.1 Sample tributaries T_1, T_2, T_3 88

5.2 Maximal deviation of tributaries T_1, T_2, T_3 94

7.1 Size of database $\mathcal{D}$ vs Initial accuracy ε_0 130

7.2 Rendering performance for various scales 130

7.3 Indicate how responsive the National Map Viewer is (Task 1.1) . . 133

7.4 Indicate how responsive the Vermont Viewer is (Task 1.2) 133

7.5 Which mode do you prefer? (Task 2.1) 133

7.6 Which experience do you prefer? (Task 3.1) 133

7.7 Which mode do you prefer? (Task 4.1) 133

7.8 Which map viewer did you find to be the most responsive? 134

7.9 Which map viewer's "zoom" feature was easier to use? 134

List of Figures

1.1 Coastline of Italy with and without gaps 5

2.1 A polyline and spline digitization of *Bowknot Bend* along the *Green River* in Grand County, Utah. 15

2.2 Simplification operator 17

2.3 Smoothing operator 17

2.4 Refinement operator 17

2.5 Illustration of Douglas–Peucker algorithm 18

2.6 Fort Irwin terrain wavelet approximations 29

3.1 Transformation of a signal using wavelets 32

3.2 *db*1 scaling function 37

3.3 *db*2 scaling function 37

3.4 Primary scaling function φ for the FBI (9–7) wavelet family . . . 49

3.5 Dual scaling function $\tilde{\varphi}$ for the FBI (9–7) wavelet family 49

4.1 Inverse mirror periodic (IMP) representation of $x(t)$ 59

4.2 Dilated and translated dual scaling function 67

5.1 Tributary T 73

5.2 The longitude component x of T with its baseline 76

5.3 The latitude component y of T with its baseline 76
5.4 Function $x^*(t)$ 77
5.5 Function $y^*(t)$ 77
5.6 Transformed polyline $\big(x^*(t), y^*(t)\big)$, $0 \le t \le \gamma_T$ 78
5.7 Transformed polyline $\big(x^*(t), y^*(t)\big)$, $t \in \mathbb{R}$ after extension 79
5.8 Tributary T_1 and a portion of tributary T_3 showing a spiral A . . 89
5.9 Illustration of progressive smoothing of a tributary 90
5.10 Tributary T_1 and its smoothing T_1^ε 91
5.11 Tributary T_2 and its smoothing T_2^ε 91
5.12 Tributaries T_3 and T_3^ε at the spiral feature A for $\varepsilon_1 = 10^{-3}$ 92
5.13 Tributaries T_3 and T_3^ε at the spiral feature A for $\varepsilon_2 = 10^{-4}$ 92
5.14 Tributaries T_3 and T_3^ε at the spiral feature A for $\varepsilon_3 = 10^{-5}$ 93
5.15 Maximal deviation of tributary T_1 versus ε 95
5.16 Maximal deviation of tributary T_2 versus ε 95
5.17 Maximal deviation of tributary T_3 versus ε 96
5.18 Deviation of T_3^ε for $\varepsilon_1 = 10^{-3}$ 98
5.19 Deviation of T_3^ε for $\varepsilon_2 = 10^{-4}$ 99
5.20 Deviation of T_3^ε for $\varepsilon_3 = 10^{-5}$ 99
5.21 Deviation of T_3 at the spiral feature A 100
5.22 Deviation vs tributary length for $\varepsilon_1 = 10^{-3}$ 101
5.23 Deviation vs tributary length for $\varepsilon_2 = 10^{-4}$ 101
5.24 Deviation vs tributary length for $\varepsilon_3 = 10^{-5}$ 101

6.1 Illustration of the UndoCycles algorithm 104
6.2 A sample river network with Strahler stream numbers 106
6.3 Tributaries in a river network 108

6.4 Restoring river network connectedness after generalization 111
6.5 Illustration of the FormTree algorithm 112
6.6 Smooth pruning 114
6.7 Renderings of the Vermont river network near the river junction B at four different map scales s. 116

7.1 Vermont hydrography map 119
7.2 Data structure for the `Graph` class 121
7.3 Data structure for the `Node` class 121
7.4 Data structure for the `Edge` class 122
7.5 Data structure for the `Tributary` class 123
7.6 Data structure for the `RangeArray` class 123
7.7 Data structure for the `WrappingArray` class 124
7.8 Function f and its linear interpolation L 128

A.1 Screenshot of our viewer 150
A.2 Questionnaire – Page 1 151
A.3 Questionnaire – Page 2 152
A.4 Questionnaire – Page 3 153
A.5 Questionnaire – Page 4 154
A.6 Lab setup for evaluation study 155
A.7 Institutional Review Board Approval Letter 156

List of Algorithms

2.1 The Douglas–Peucker algorithm 19

2.2 The Fourier approximation algorithm 22

5.1 The TributaryDecomposition algorithm 81

5.2 The TributarySynthesis algorithm 83

6.1 The UndoCycles algorithm 105

6.2 The Strahler algorithm 106

6.3 The GetTributaries algorithm 108

6.4 The FormTree algorithm 111

7.1 The SplineEvaluation algorithm 127

Abstract

The focus of our research is on a new automated smoothing method and its applications. Traditionally, the application of a smoothing method to a collection of polylines produces a new smoothed dataset. Although the new dataset was derived from the original dataset, it is stored independently. Since many smoothing methods are slow to execute, this is a valid trade-off. However, this greatly increases the data storage requirements for each new smoothing. A consequence of this approach is that interactive map systems can only offer maps at a discrete set of scales. It is desirable to have a fast enough method that would support the reuse of a single base dataset for on-the-fly smoothing for the production of maps at *any* scale.

We were able to create a framework for the automated smoothing of river networks based on the following major contributions:

- A wavelet–based method for polyline smoothing and endpoint preservation
- Inverse Mirror Periodic (IMP) representation of functions and signals, and dimensional wavelets
- Smoothing of features that does not change abruptly between scales

- Features are pruned in a continuous manner with respect to scale
- River network connectedness is maintained for all scales
- Reuse of a base geographic dataset for all scales
- Design and implementation of an interactive map viewer for linear hydrographic features that renders in subsecond time

We have created an interactive map that can smoothly zoom to any region. Numerical experiments show that our wavelet-based method produces cartographically appropriate smoothing for tributaries. The system is implemented to view hydrographic data, such as the USGS National Hydrography Dataset (NHD). The map demonstrates that a wavelet–based approach is well suited for basic generalization operations. It provides smoothing and pruning that is continuously dependent on map scale.

Chapter 1

Introduction

1.1 Motivation

Maps are representations of the world around us. When cartographers make a map, it is their job to select which features appear on that map. This selection process is a combination of both art and science. Researchers are looking for ways to formalize this process with the purpose of identifying the elemental operations in map making. The creation of a map is a tedious and time consuming activity. With the advent of computers, it is natural to attempt to automate it.

In order to automate this process, the data should be available in digital form. A geographic information system (GIS) is a system to capture, store, and display digital geographic data. A GIS typically represents rivers, boundaries, and cities as collections of polylines, polygons, and points. It is the task of an automated system to extract a relevant subset of this data to produce a usable map.

Automated map creation involves the application of several generalization operators. This includes smoothing, simplification, and pruning. Smoothing is the extraction of a general trend in polyline data. Simplification is the reduction

in the number of vertices in a polyline. Pruning is the selection of features that are relevant for a given map.

The focus of our research is on a new automated smoothing method and its applications. Traditionally, the application of a smoothing method to a collection of polylines produces a new smoothed dataset. Although the new dataset was derived from the original dataset, it is stored independently. Since many smoothing methods are slow to execute, this is a valid trade-off. However, this greatly increases the data storage requirements for each new smoothing. A consequence of this approach is that interactive map systems can only offer maps at a discrete set of scales. It is desirable to have a fast enough method that would support the reuse of a single base dataset for on-the-fly smoothing for the production of maps at *any* scale. This would counteract the need for increased data storage requirements.

There are several existing methods for smoothing: the Douglas–Peucker algorithm, Perkal's ε–circle method, Fourier transform method, Gaussian smoothing, and others. However, none of them satisfy our desired requirements.

Recently wavelets have emerged as a new mathematical tool with wide practical applications. Addison [1] provides a review of various applications of wavelets to fluids, data compression, medicine, finance, geophysics, and other areas. They have also been applied to cartographic issues, such as smoothing. However, the research in wavelet-based cartographic smoothing was inconclusive. In particular, researchers have indicated that features were not smoothed in a cartographically appropriate manner. For example, some generalizations produced self-intersections. Others smoothed the feature unevenly, see [3]. Some algorithms were slow, required excessive storage, and as a result were not scalable to larger datasets.

A major stumbling block for the generalization of linear feature networks (road networks, river networks, etc.) is the drift that occurs after smoothing. While not an issue when smoothing a single linear feature, this becomes problematic when smoothing a collection of linear features that should remain connected. A desirable characteristic of any smoothing method is the preservation of some critical points. In particular, if the endpoints are not preserved then gaps may appear between adjacent linear features.

We hypothesize that by using wavelets we can achieve the following:

- Smoothing of linear features in a cartographically appropriate manner
- Generalization of linear feature networks
- Reuse of a single base dataset to produce maps at any scale
- A fast execution time suitable for on-the-fly smoothing

1.2 Research path

In 2008, we had the opportunity to work on a project that required the development of an interactive map viewer. When rendering a map, it became immediately clear that this was a challenging task. We had to deal with labelling, pruning, and point-reduction. In particular, we implemented the Douglas–Peucker algorithm to simplify highways, borders, and rivers. This was necessary to reduce rendering time to allow for acceptable interactivity. It also demonstrated the deficiencies of this approach. Namely, each simplified layer had to be stored separately which led to a significant increase in the dataset size. Also, the algorithm was quite slow.

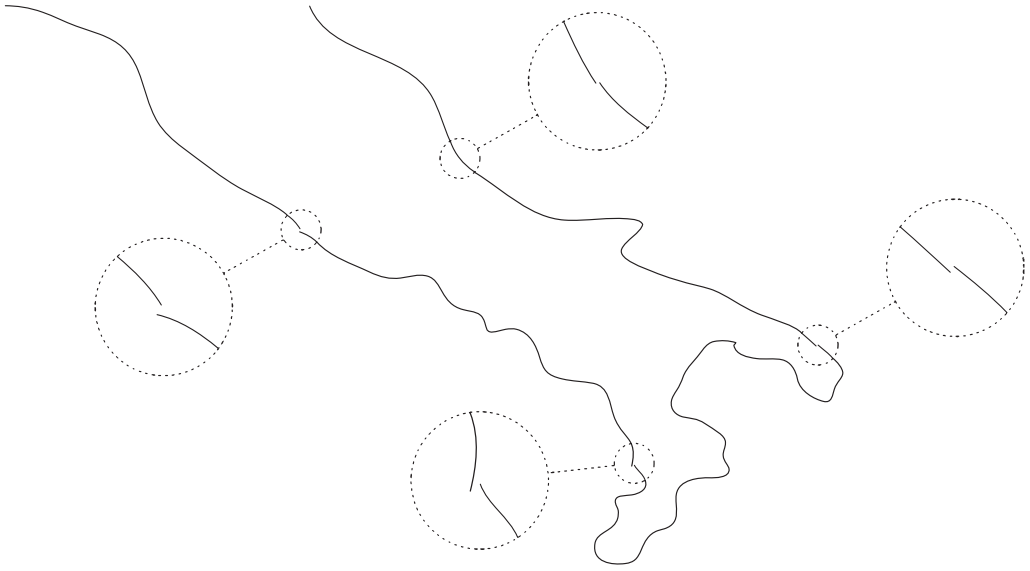
We researched current efforts in multi-scale databases and their visualization. Such a database is designed to support the production of multiple scale maps from a single base dataset. Around the same time, we learned that wavelets are a mathematical theory based on multiresolution analysis. As a result, we thought that wavelets could be applied to solve the multi-scale map generation problem.

Since wavelet theory is a difficult concept to grasp, our intention was to use wavelets as a "black box" from a toolkit. During the literature review, we found that many researchers were working in this area. Many of them also used wavelet toolkits to carry out their experiments. However, their results had limited applicability [3]. This could be due to their use of simple wavelet families.

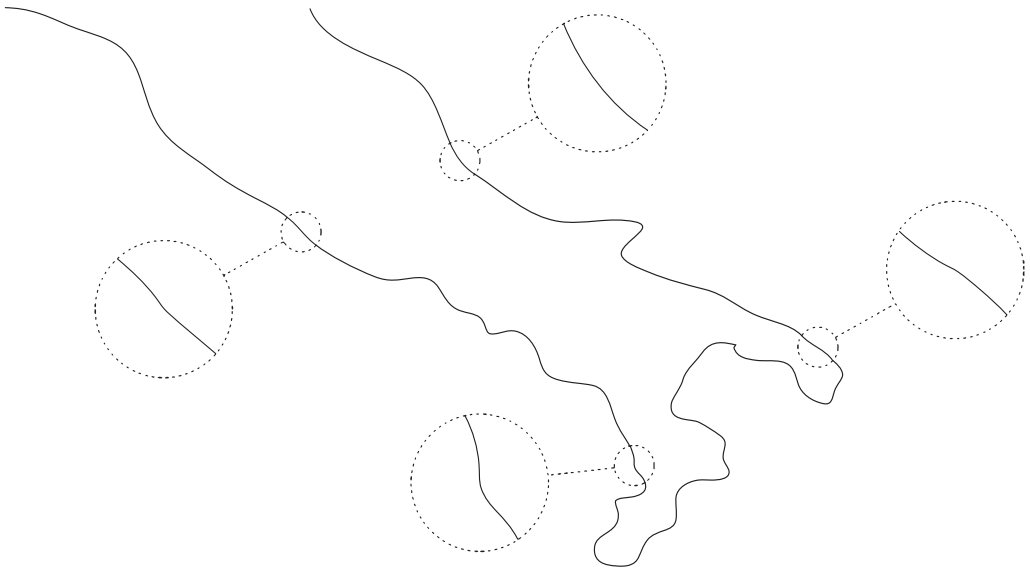
Additionally, a significant difficulty in linear feature network generalization is the *feature drift problem*. After smoothing, a feature's shape has changed and some drift or movement from the original feature has occurred. When multiple features need to be joined, like when a stream joins a river, the smoothed features may no longer connect.

A promising solution to this problem was presented in the paper by Li, et al. [32] (discussed later in this chapter). They show that using standard-wavelet-smoothing, for Italy's coastline, produced gaps between adjacent linear features. The authors proposed a remedy based on the construction of a special family of wavelets that preserve linear feature endpoints, see Figures 1.1(a) and 1.1(b). However, their computationally expensive method requires significant storage overhead for each linear feature. As a result, the size of the dataset must be reduced to fit within memory constraints. This makes their method impractical to use for real-world geographic datasets. A practical solution to endpoint preservation remained unresolved.

During our investigation of moving-average-based smoothing we realized that



(a)



(b)

Figure 1.1: (a) Smoothed coastline of Italy with gaps between adjacent segments, (b) Smoothed coastline of Italy with an endpoint preservation method [32]. Reprinted with permission from Elsevier.

when certain functions are averaged by symmetric kernels their endpoints are preserved. This led us to believe that similar results could be achieved with wavelets too.

As a result, we have devised a method based on three major components:

1. Functions are represented in special symmetric and periodic form. We call such representations *Inverse Mirror Periodic* (IMP) form.

2. Biorthogonal wavelet families with symmetric scaling functions.

3. The chosen biorthogonal wavelet family is scaled appropriately to the endpoints of the function. We call such wavelet families *dimensional wavelets* because they reflect the length of the linear feature they approximate.

To justify this method, it was necessary to prove that dimensional wavelet families acting upon IMP functions would achieve the desired effect of preserving the endpoints. Additionally, we devised a wavelet-based algorithm that smooths linear features in a manner continuously dependent on an accuracy parameter. This constitutes the main theoretical study of our research.

Having developed a theoretical basis for linear feature smoothing. We applied it to a real-world river network system. Such a dataset was available from the United States Geological Survey (USGS), which maintains the National Hydrography Dataset:

> The National Hydrography Dataset (NHD) is the surface water component of The National Map. The NHD is a digital vector dataset used by geographic information systems (GIS). It contains features such as lakes, ponds, streams, rivers, canals, dams and streamgages. These data are designed to be used in general mapping and in the analysis of surface-water systems [64].

The National Map project contains an interactive viewer for the NHD [65]. It provides for several discrete map scales to view the data. However, discrete map scales make it difficult for a user to relate objects from one scale to another. This makes for a poor user experience. The viewer is image tile based—this means that each scale shows different pre-rendered images. Such an approach requires a massive amount of storage for the images, and it is impossible to display the map at intermediate scales.

To demonstrate the capability of our method, we decided to implement a Vermont hydrography viewer. An issue that remained to be solved was the preservation of network connectedness after smoothing. For example, a child tributary that flows into a parent tributary (in between endpoints) may become disconnected after smoothing. It was solved by devising an algorithm that uses the hierarchy of tributaries to snap the children back to their parents.

The resulting viewer has been evaluated by a group of participants who provided their opinion on the performance of the Vermont viewer. They also compared their experience with the National Map viewer. Generally, the participants preferred our viewer.

1.3 Contributions

As we have already mentioned, the smoothing of polylines by wavelets introduces some undesirable effects. In particular, two adjacent polylines may become disconnected after this procedure. Figure 1.1(a) shows such gaps between adjacent segments. The issue is how to preserve the endpoints of the polylines after the smoothing operation.

In our research we concentrate on the generalization of hydrography flowline

river networks. These types of networks present a wide variety of linear feature shapes to be smoothed. Other linear features, such as roads, will be considered in future work. Accordingly, we present our research in terms of *tributaries* and river networks, even that it may have other applications.

Another obstacle to overcome is the possible violation of river network connectedness. In other words, tributaries may become detached from the rivers they flow into after smoothing. Such a map would not be cartographically accurate.

We also have to resolve how to fluidly transition between maps at different scales. This is needed for implementing continuous zoom-operations in an interactive map viewer. Interactivity implies that such a viewer should perform all these actions in subsecond time.

We were able to create a framework for the automated smoothing of river networks based on the following major contributions:

- A wavelet–based method for polyline smoothing and endpoint preservation
- Inverse Mirror Periodic (IMP) representation of functions and signals, and dimensional wavelets
- Smoothing of features that does not change abruptly between scales
- Features are pruned in a continuous manner with respect to scale
- River network connectedness is maintained for all scales
- Reuse of a base geographic dataset for all scales
- Design and implementation of an interactive map viewer for linear hydrographic features that renders in subsecond time

1.4 Overview

Chapter 2 is devoted to cartographic issues. We discuss digitization, simplification, smoothing, refinement and review them. We present the Douglas–Peucker algorithm, which is the most popular method for reducing vertices in a polyline. Although it is not a smoothing method, it is often used as one because of its simplicity. We also consider several smoothing algorithms: Perkal's ε–circle method, Fourier transform method, and Gaussian smoothing. Next we look at the use of wavelets in cartography. Wavelets are used for smoothing, terrain modeling, and satellite image compression.

Chapter 3 is a brief tutorial on wavelets and a compilation of reference materials. Since our research expands standard wavelet theory and since wavelet conventions are not standardized, it is appropriate to present a unified wavelet theory description. We start with basic wavelet concepts, including masks and scaling functions. We introduce orthogonal systems and multi-resolution analysis. We discuss the structure of orthogonal and biorthogonal wavelet families. Then, we give examples of biorthogonal wavelet families with symmetric masks. The transition between wavelet coefficients is described in signal processing terms. This topic is the Discrete Wavelet Transform, which includes the definition of low and high pass filters.

Chapter 4 examines our modifications to the wavelet method. We introduce *dimensional wavelets* which are specially scaled biorthognal wavelets tuned to the size of their underlying linear features. We also introduce the Inverse Mirror Periodic (IMP) representation for functions and signals. Our main results in this chapter are theorems in section 4.3.1. They show that IMP functions and signals retain their IMP structure under transforms by dimensional wavelets.

Dimensional wavelets along with IMP functions provide the theoretical basis for our system.

Chapter 5 shows how to apply dimensional wavelets to smooth linear features. It shows how to transform tributaries to IMP form, and how to decompose them into wavelet details. The TributarySynthesis algorithm describes how to use the wavelet details to obtain smoothed tributaries. Our main result is a theorem (5.3.4) that proves that the algorithm preserves the endpoints of the tributaries and smooths them in a continuous manner with respect to map scale. We also provide a detailed evaluation of the algorithm as it applies to various tributaries.

We chose three tributaries of various length. For each tributary (T_1, T_2, and T_3) we smoothed them for three different accuracy levels and plotted the results. We demonstrate that our method can smooth a variety of tributary shapes in a cartographically appropriate manner. In particular, tributary T_3 contained a complicated spiral-like feature that was still smoothed appropriately.

An important quantitative measure of the accuracy of the approximation is the deviation of the smoothed tributary from the original. We show the deviation of tributary T_3 along a portion of its length. Our analysis of 5,322 tributaries shows a slight correlation between maximal deviation and tributary length.

Chapter 6 describes our method for the generalization of river networks. It is based on an application of our method for linear feature generalization. This approach consists of two stages: the preprocessing stage and the generalization stage. The preprocessing stage divides the river network into a collection of tributaries. This stage involves decycling the network in order to prioritize the edges of the graph by their Strahler numbers. A *tributary* is defined as the maximal linear feature with the same Strahler number. Each tributary is decomposed into wavelet details. Some additional metadata related to the tributary's hierarchy is

also stored in a database. A complete description of the database is provided in Chapter 7.

The generalization stage uses two functions $\varepsilon(s)$ and $\sigma(s)$. When a user requests a map of a region, the system infers the scale s of the requested map. The first function $\varepsilon(s)$ relates the accuracy of the smoothing to s, and the second function $\sigma(s)$ relates the amount of pruning to s. The processing of the user's request involves the determination of tributaries that should be shown on the map, their prioritization, their wavelet-based smoothing, and their pruning. Our algorithms smooth and prune tributaries in a manner that is continuous with respect to the map's scale.

Chapter 7 discusses an implementation of the system. The goal of the implementation is to produce an interactive viewer for hydrographic data. The viewer illustrates the practicality of the wavelet-based smoothing framework. We used the Vermont flowline hydrography as a base dataset. This dataset was retrieved from the USGS NHD website. The next step was to use a spatial index to build an in-memory graph that represented the river network. At this stage we implemented the UndoCycles, Strahler, and GetTributaries algorithms. The system data structures are also presented in this chapter.

The TributaryDecomposition algorithm was implemented according to our dimensional wavelet theory that is described in Chapter 5. Smooth functions require fewer wavelet details to represent them accurately, than non-smooth ones. Accordingly, our implementation involves an additional step of spline interpolating the tributary polylines. This also produces a better representation of the actual physical feature.

Our renderer uses the TributarySynthesis and FormTree algorithms. Additional steps are taken to improve its rendering speed. As a result the system

executes in subsecond time.

A study was conducted to evaluate the responsiveness of the implemented viewer and to compare it to the National Map Viewer. The evaluation study material is attached in Appendix A. Overwhelmingly, the participants indicated that our viewer was more responsive than the USGS National Map viewer. The participants preferred our smooth pruning strategy, as well as multi-touch track-pads. Additionally, they indicated that our viewer was easier to use for zooming operations.

Chapter 8 summarizes our results and provides conclusions. In the future we plan to extend our system so that it will take into account other map layers, support progressive data transfer over the Internet, and allow for the viewing of large geospatial datasets. Additionally, we will look into algorithms that provide more flexible generalization options.

Chapter 2

Cartographic generalization

The focus of our research is on a new automated smoothing method. One of its intended applications is for the smoothing of geographic linear features. This chapter provides a brief background on cartographic issues to provide context for our method.

Maps are a visual way to present geographic data. Typically, geographic data is stored in a Geographic Information System (GIS). Such a system encodes geographic features as collections of numerical values. A GIS can capture, analyze, and display geographic information. For a general introduction to cartography and GIS see [27, 15, 33].

For the sake of readability, we refer to linear features and their networks as tributaries and river systems. However, it could be applicable to other linear features including borders, shorelines, and roads. These applications are not addressed in this dissertation, but are a subject of future research. In a GIS a linear feature is represented by a polyline. A *polyline* is a sequence of coordinate points (vertices) that are connected by line segments. It is important to notice that the polyline serves as an approximation of the original geographic feature.

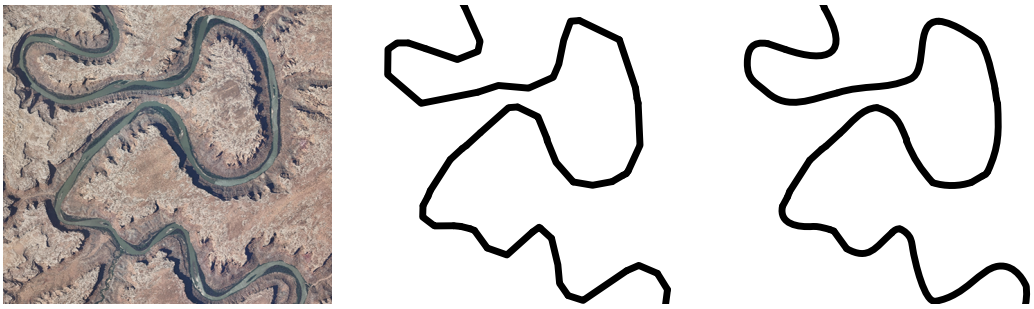
In this chapter, we discuss cartographic digitization and generalization. We consider the simplification, smoothing, and refinement generalization operators. We present the Douglas–Peucker algorithm, which is the most popular method for reducing vertices in a polyline. Although it is not a smoothing method, it is often used as one because of its simplicity. We also consider several smoothing algorithms: Perkal's ε–circle method, Fourier transform method, and Gaussian smoothing. We also review how wavelets are applied in GIS.

Current research efforts for automated multi-scale map production are directed at reusing a baseline geographic dataset [42]. Traditionally, a separate database is maintained for each required map scale. This means that the same feature is represented multiple times. When features need to be updated, it is necessary to update each database accordingly. This can be a time consuming and error prone job. Therefore, such an approach could become unmanageable for more than a few map scales. Wavelets provide a natural way to reuse a single dataset for multi-scale map production. Multi-resolution databases are reviewed at the end of this chapter.

2.1 Digitization

To represent geographic features in a GIS, it is necessary to digitize them [27, 15]. The digital data can be obtained in a variety of ways: surveying, high-resolution GPS [8], Light Detection And Ranging (LiDAR) [10], and the digitization of existing paper maps. Merwade [40] describes an automated method for delineating hydrological features from aerial photography.

Figure 2.1(a) shows an aerial photograph [58] of the *Snake River* in Grand County, Utah. Its polyline representation is shown in Figure 2.1(b). This poly-



(a) Aerial photograph (b) Polyline representation (c) Spline representation

Figure 2.1: A polyline and spline digitization of *Bowknot Bend* along the *Green River* in Grand County, Utah.

line represents the center flowline of the river. Additionally, Figure 2.1(c) shows a cubic spline interpolation of the polyline. It can be seen that the spline interpolation provides a more realistic representation of the original river.

2.2 Generalization

According to Stanislawski [56],

> A principal objective of cartographic generalization is reduction of content and detail of geospatial data in a manner that appropriately portrays remaining features at smaller scales.

Plazanet [44] describes generalization as the process of selecting the key geographic features to maintain, while removing unimportant features when producing a map at a targeted scale.

McMaster and Shea [38], describe twelve generalization operators: aggregation, amalgamation, classification, collapse, displacement, enhancement, exaggeration, merge, refinement, simplification, smoothing, and typification. Other generalization models are discussed by Li [34].

2.3 Simplification, Smoothing, and Refinement

At this time our system only employs three generalization operators: simplification, smoothing, and refinement. It applies these operators in a manner that is continuous with respect to the scale of the map. Examples of these operators are shown in Figures 2.2, 2.3, and 2.4.

According to McMaster and Shea [38], simplification (point reduction) is the process of reducing the number of coordinate points (polyline vertices) of a feature while retaining its original character. The benefits of simplification include reduced plotting time and reduced storage. However, it can introduce distortions and some important details of the feature can be lost. Figure 2.2 illustrates a reduction in the number of vertices in the polyline, however the overall shape is preserved.

The smoothing operator relocates the original coordinate points, with the intention of "capturing only the most significant trends of the line," [38]. While the original linear feature is a polyline, smoothing produces a smooth line that resembles the polyline. Figure 2.3 illustrates a smoothing of the original polyline. Notice that several of the vertices have shifted their location.

The refinement (pruning) operator reduces the number of displayed features on a map. This is done to reduce map clutter and increase its clarity. For example, a small scale map should show only major rivers, while a large scale map may show every single stream. Figure 2.4 illustrates how only the major network features remain after refinement.

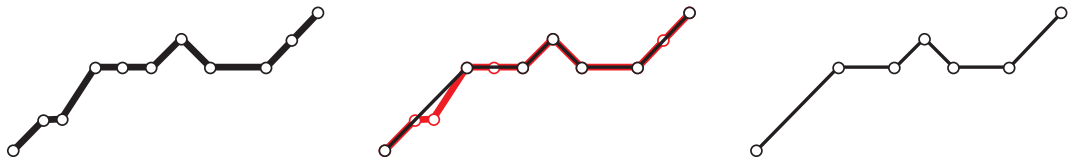


Figure 2.2: Simplification operator

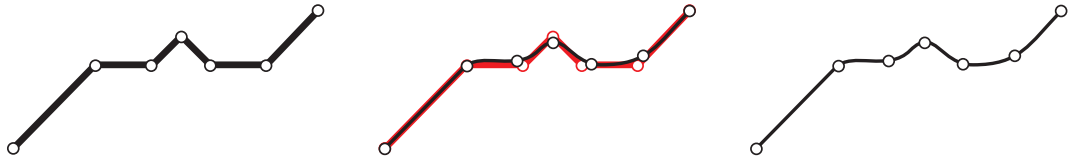


Figure 2.3: Smoothing operator

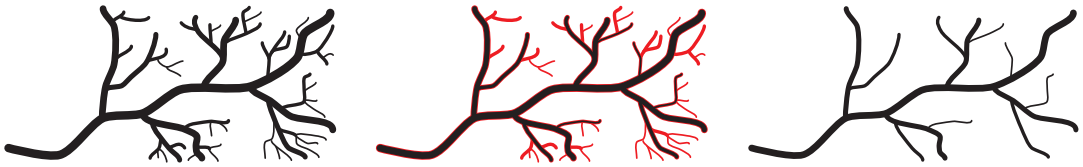


Figure 2.4: Refinement (pruning) operator

2.3.1 Simplification

Li [34] states that point reduction algorithms have been studied intensely. The most popular among them is the algorithm by Douglas and Peucker [16], also known as Ramer's Algorithm [47]. While the algorithm does reduce the number of points, it also introduces several undesirable artifacts. These artifacts include self-intersection, cross-intersection, shape distortion, and starting point dependency. Attempts have been made to rectify these issues, such as the Li–Openshaw algorithm [35]. Raposo [49] provides an analysis of several other point reduction algorithms. For example, Guilbert [23] describes a method to detect

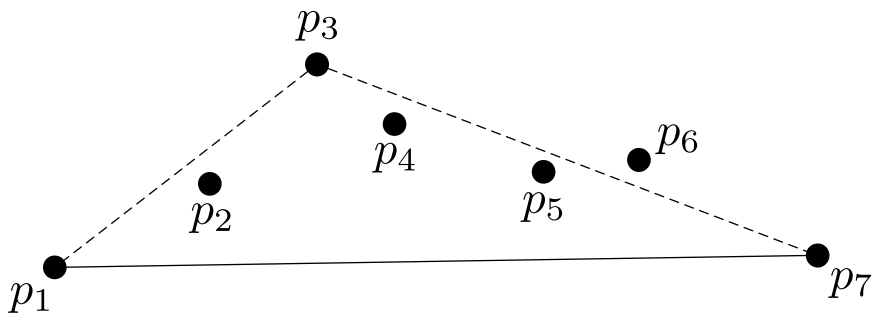


Figure 2.5: Illustration of Douglas–Peucker algorithm

critical points of a polyline feature. These critical points are used for simplification. Shi, Cheung [53], Cromley, and Campbell [13] evaluate the performance of such algorithms.

Douglas–Peucker algorithm

A popular method for line generalization is the Douglas–Peucker algorithm. Consider for example a sequence P of seven points p_i, where $i = 1, 2, 3, \ldots, 7$. According to the algorithm, we connect the points p_1 and p_7 by a straight line segment, as shown in Figure 2.5.

We compute the distances from the remaining points p_2, p_3, p_4, p_5, p_6 to the segment $\overline{p_1 p_7}$. If each distance is smaller than accuracy $\varepsilon > 0$, then the algorithm is stopped. The segment $\overline{p_1 p_7}$ becomes the generalization of the original sequence. Otherwise, choose the point farthest from $\overline{p_1 p_7}$. In the figure, this is p_3. Connect p_1 to p_3, and p_3 to p_7. Now rerun the algorithm on both segments $\overline{p_1 p_3}$ and $\overline{p_3 p_7}$. In this example, the algorithm stops after this iteration because the accuracy is satisfied. The generalized polyline consists of segments $\overline{p_1 p_3}$ and $\overline{p_3 p_7}$.

Input: Polyline $P = \{p_1, p_2, \ldots, p_N\}$ and accuracy ε
Output: Simplified polyline P^ε

```
1  Let P^ε ← P
2  if N ≤ 2 return P^ε
3  Connect points p_1 and p_N
4  for i = 2 to N − 1 do
5     Compute the distance d_i from point p_i to the segment p_1p_N
6  end for
7  Find any d_j ← MAX(d_i) where 2 ≤ i ≤ N − 1
8  if d_j < ε then
9     return P^ε
10 else
11    Call Douglas–Peucker({p_1, ..., p_j}, ε)
12    Call Douglas–Peucker({p_j, ..., p_N}, ε)
13 end if
```

Algorithm 2.1: The Douglas–Peucker algorithm

The accuracy affects the number of points that are eliminated from the original polyline. Choosing a small accuracy ε results in a better approximation to the original polyline. One desirable outcome of the algorithm is endpoint preservation. In the example above, the endpoints p_1 and p_7 always remain the endpoints of any simplified polyline.

This recursive algorithm has time complexity $\mathcal{O}(n^2)$. A detailed analysis of the performance of the algorithm as well as an enhancement is described by Hershberger [28] and Jenks [29]. Additionally, a parallel implementation of the algorithm is detailed by Vaughan et al. [66]. The disadvantage of this generalization method is its long execution time. Ramos et al. [48] propose an efficient method to store multiple generalized versions of the geographic data obtained by running the Douglas-Peucker algorithm. This is useful when implementing an interactive multi-scale system.

2.3.2 Smoothing

The smoothing operator approximates the original polyline f by a smooth curve. The smoothed curve does not necessarily pass through all the vertices of the polyline. Generally, the level of smoothing depends on the desired accuracy $\varepsilon \geq 0$. It is desirable that the smoothed curves f^{ε} satisfy $f^{\varepsilon} \to f$ as $\varepsilon \to 0$. The accuracy is typically dependent on the map's scale. This is justified by Li and Openshaw's [35] "natural principle":

> ... for a given scale of interest, all details about the spatial variation of geographical objects beyond certain physical limitations are unable to be presented and can be neglected.

Perkal's ε–circle method

In 1965, Perkal [43] proposed an ε–circle method for smoothing borders of polygonal areas. This method can also be applied to polylines. Given a polyline, a circle of diameter ε is rolled along the line. By tracing the center of the circle, we obtain a smoothed version of the polyline. Clearly, small features are neglected in this approach. Also, as $\varepsilon \to 0$ the smoothed curves approach the original polyline. However, research by Christensen [12] shows that the computer implementation of this method is difficult. Note that this algorithm does not preserve the endpoints of the original curve.

Fourier transform method

The Fourier transform provides another smoothing method [7, 44].

Consider the space $L^2[0, \pi]$ of square integrable functions. For such functions

we define the inner product $\langle f, g\rangle$ and the norm by

$$\langle f, g\rangle = \int_0^\pi f(t)g(t)\,dt, \quad \|f\| = \sqrt{\int_0^\pi |f(t)|^2\,dt}.$$

A system of functions $\{u_n\}_{n=1}^\infty \in L^2[0,\pi]$ is called an orthonormal basis if

$$\|u_n\| = 1, \quad n \in \mathbb{N}, \quad \langle u_n, u_m\rangle = 0, \quad m \neq n,$$

and

$$f = \sum_{n=1}^\infty c_n u_n, \quad \text{where} \quad c_n = \langle f, u_n\rangle$$

for any $f \in L^2[0,\pi]$.

Approximations for f are defined as partial sums of the above series

$$f_N = \sum_{n=1}^N c_n u_n.$$

An example of an orthonormal basis in $L^2[0,\pi]$ is $u_n(t) = \sqrt{\frac{2}{\pi}}\sin(nt)$, $n = 1, 2, 3, \ldots$

It follows from the orthonormality of the basis u_n that

$$\|f\|^2 = \sum_{n=1}^\infty c_n^2 \quad \text{and} \quad \|f_N\|^2 = \sum_{n=1}^N c_n^2. \tag{2.1}$$

The identities in (2.1) are known as *Parseval's identities* [6].

An efficient way to approximate functions using the Fourier series method is to only use the Fourier coefficients $c_n = \langle f, u_n\rangle$ with the highest magnitudes. The level of approximation (smoothing) is controlled by the accuracy parameter ε. This is summarized in the following algorithm:

Input: Function f and accuracy $\varepsilon > 0$
Output: Approximation f^{ε}

1 Given f compute its Fourier coefficients $c_n = \langle f, u_n \rangle,\ n = 1, 2, 3, \ldots$
2 $c_{sorted} \leftarrow$ Sort the coefficients in descending order of their magnitude
3 Compute $\|f\|^2 = \sum_{n=1}^{\infty} c_n^2$
4 Choose first M coefficients from c_{sorted} such that

$$\sum_{i=1}^{M} c_{sorted,i}^2 \geq (1 - \varepsilon)\|f\|^2$$

5 Compute the approximation $f_M = \sum_{i=1}^{M} c_{sorted,i}\, u_{sorted,i}$
6 **return** $f^{\varepsilon} \leftarrow f_M$

Algorithm 2.2: The FOURIER approximation algorithm

The error of the approximation produced by the FOURIER approximation algorithm can be bounded by $error = \|f - f^{\varepsilon}\|^2 \leq \varepsilon\|f\|^2$. The efficiency of this algorithm can be improved if the function f is properly extended outside the interval $[0, \pi]$. A method for such an extension is described in section 4.2. A direct implementation of this algorithm yields an $\mathcal{O}(n^2)$ time complexity. However, a Fast Fourier Transform (FFT) implementation yields an $\mathcal{O}(n \log n)$ time complexity [46].

Gaussian smoothing

Gaussian smoothing is discussed by Fdez-Valdivia et al. [17]. The smoothed curves are used to find points with the greatest curvature. Such points are considered to be important features of the original curve. This information is used to reduce the number of points in the linear feature in a scale appropriate manner. A similar approach is used by Thapa [61]. Rosin [50] subdivides a curve based on curvature, and then performs Gaussian smoothing on each segment.

Gaussian smoothing is a weighted moving average method. Let $f(t),\ t \in \mathbb{R}$

be a function, and $\varepsilon > 0$ be an accuracy parameter. This parameter controls the level of smoothing. The smoothed function f^ε is defined by:

$$f^\varepsilon(t) = \int_{-\infty}^{\infty} f(z)U(t-z,\varepsilon)\,dz,$$

where the Gaussian $U(t,\varepsilon)$ is given by

$$U(t,\varepsilon) = \frac{1}{2\sqrt{\pi\varepsilon}} e^{-\frac{t^2}{4\varepsilon}}.$$

This gives an accuracy dependent smoothing for f, since $f^\varepsilon \to f$ as $\varepsilon \to 0$. Further information can be found in the book by Greenberg [21].

Other smoothing methods

Nöllenburg et al. [41] present a polyline morphing algorithm. In this approach, polylines are smoothed by a smoothing method at a fixed number of scales. Then, to achieve an approximation at any intermediate scale, the precomputed smoothed curves are appropriately interpolated.

Burghardt [9] and Guilbert [22, 24] use a snake model to achieve smoothing. A snake is defined as a B-spline that has minimal energy. Energy is defined as the sum of internal and external energies. The internal energy is a measure of the sinuosity (deviation from a straight line) of the curve, and the external energy is a measure of the distance of the curve to neighboring map objects.

Saux [51] uses B-spline curves to approximate coastlines. The algorithm selects representative vertices of the original polyline. Then, a B-spline with minimal curvature is selected to approximate the polyline.

These approaches require repeated minimization procedures to obtain differ-

ent levels of smoothing. Therefore, they are computationally expensive algorithms that are not well-suited for on-demand smoothing at multiple scales.

2.3.3 Refinement/Pruning

McMaster [38] gives the following definition for refinement:

> In many cases, where like features are either too numerous or too small to show to scale, no attempt should be made to show all the features. Instead, a selective number and pattern of the symbols are depicted. Generally, this is accomplished by leaving out the smallest features, or those which add little to the general impression of the distribution, but can be accomplished by using a representative pattern of the symbols... Though the overall initial features are thinned out, the general pattern of the features is maintained.

Töpfer and Pillewizer [62] propose a Radical Law for feature selection. The law can be stated as

$$n_f = n_a \sqrt{\frac{M_a}{M_f}},$$

where n_f is the number of objects which can be shown at the target scale, n_a is the number of objects shown on the source map, M_a is the scale denominator of the source map, and M_f is the scale denominator of the target map.

Stanislawski [56, 55] uses the drainage area of a river network to determine how to prune the network. This metric is used to assign priority to tributaries of the river system. Their research is used to support automated generalization of the USGS National Hydrography Dataset.

Assigning priority to tributaries is essential for pruning operations. Horton [36] proposed an algorithm for such prioritization. Another popular stream ordering method is based on Strahler numbers [20]. While any prioritization can be used, we selected Strahler numbers in our implementation to assign priority

to tributaries. The selection of one prioritization method over another was not essential our research on smoothing.

2.4 Wavelets in cartography

Recently, wavelets have emerged as a new mathematical tool with wide practical applications. Addison [1] provides a review of various applications of wavelets to fluids, data compression, medicine, finance, geophysics, and other areas. A mathematical foundation is detailed in Chapters 3 and 4.

2.4.1 Smoothing

Wavelets have cartographic applications as well. Balboa and López [3] analyze the effectiveness of wavelets used for linear feature smoothing. The authors use Haar wavelets to generalize linear features. They evaluate the method's performance using mathematical measures such as:

- Percentage change in number of coordinates
- Percentage change in the standard deviation of the number of coordinates per 100 meters of the line
- Percentage change in angularity with sign
- Percentage change in angularity in absolute value
- Total areal difference in absolute value
- Percentage change in the number of inflection points

To establish a baseline for the quantification, Balboa and López also measure the results of generalization by the Douglas–Peucker algorithm. The authors' results show that generalizations using Haar wavelets poorly maintain the original position and geometry of the linear feature. However, the general tendency (or trend) of the polyline is preserved. The method performs poorly because it depends on the homogeneity of the polyline. The homogeneity of the polyline refers to how similar in terms of sinuosity different segments of the line are. The method simplifies different parts of the polyline in a non-uniform fashion. Furthermore, this can cause crossings.

The authors discuss that by using the above measures, it is possible to classify segments of the polyline as sinuous or straight. This classification can then be used as an input parameter to a generalization algorithm. The authors conclude that the comparison between the wavelet approach and the Douglas–Peucker algorithm was mixed. Some measures, such as the standard deviation of the number of coordinates, performed better under the wavelet methods. However, the angularity measures were better under the Douglas–Peucker algorithm. According to Balboa and López, the wavelet method is useful for trend extraction used in strong scale reductions, segmentation of the polyline, and the classification of the curve based on sinuosity. However, Fritsch and Lagrange [19] conclude that a wavelet-based algorithm provides acceptable smoothing of linear features.

Shu, et al. [54] equate the linear feature generalization problem to data *denoising*. They state that noises are unimportant details in the data that can be removed. The discussion centers on various techniques for wavelet coefficient thresholding. The authors use Töpfer's Radical Law of feature selection to determine which coefficients to keep at each wavelet dilation level. The results of the experiments show that few coefficients need to be kept in order to give a good

approximation of the original linear feature. For these experiments the authors used $db4$ wavelets [63]. This family provides a smoother approximation than Haar or $db2$ wavelet families.

Li et al. [32] use B-spline wavelets subjected to certain constraints. Constrained B-spline wavelets have the added property that they can preserve certain key points from the original base data. This is important when generalizing a linear feature that should be connected to an adjacent linear feature.

Approximations using non-constrained wavelets are subject to geographic location drifts. Figure 1.1(a) illustrates the use of non-constrained B-spline wavelets. It shows gaps between independently generalized linear features of Italy's coastline. Figure 1.1(b) shows how constrained B-spline wavelets solve this problem. This method resolves the important issue of endpoint preservation by using a matrix to perform a minimization routine. However, this technique does not scale well for more than a few linear features. The method requires the storage of a matrix associated with each linear feature. Additionally, the minimization is computationally expensive, which makes the method unsuitable for on-demand applications.

Hahmann et al. [25] shows how to generalize polygonal features while preserving their areas. Wang and Zhang [67] use wavelets to simplify NURBS (non-uniform rational B-spline) curves. Their method preserves the endpoints of the smoothed curve by nullifying certain rows in the matrix that represents the wavelet transform. Finkelstein and Salesin [18] describe an application of wavelets for smoothing. Their method also uses a matrix-based approach. Such an approach is computationally expensive and requires significant storage overhead for each linear feature. This makes their method impractical to use with real-world geographic datasets. Chieppa et al. [11] present a similar method.

2.4.2 Other uses for wavelets

Terrain modelling

McArthur et al. [37] use various families of wavelets to generalize raw digital terrain elevation data (DTED). The goal is to create a hierarchically structured multiple-levels-of-detail database for use in applications such as flight simulation. A hierarchy enables efficient rendering of the terrain so that only relevant local details need to be updated.

Bjørke and Nilsen [5] use 2D wavelets to generate simplified terrain models. The authors investigate the effects of wavelet coefficient thresholding on the generalization. It is shown that wavelet methods are an efficient way to represent the terrain. However, artifacts can be introduced when thresholds are set too high. The authors note that artifacts can be removed from flat surfaces, such as lakes and oceans, by masking the area and eliminating it from the generalization routine. In their experiments, the authors use biorthogonal average interpolating 2D wavelets. Zhang et al. [71, 72] use wavelets to generalize large terrain models for an interactive 3D viewer.

Tate et al. [60] investigate the effectiveness of wavelets and regression methods for noise elimination of LiDAR digital surface models. They conclude that both techniques are effective, however the regression method produces better results.

Satellite Image compression

Wu et al. [69, 68] describe the design of an online viewer for high-resolution digital orthophotos (satellite imagery). The system applies a 2D Haar wavelet transform to the photos. It allows for progressively higher resolution approximations. Therefore, the client can efficiently download images from the server

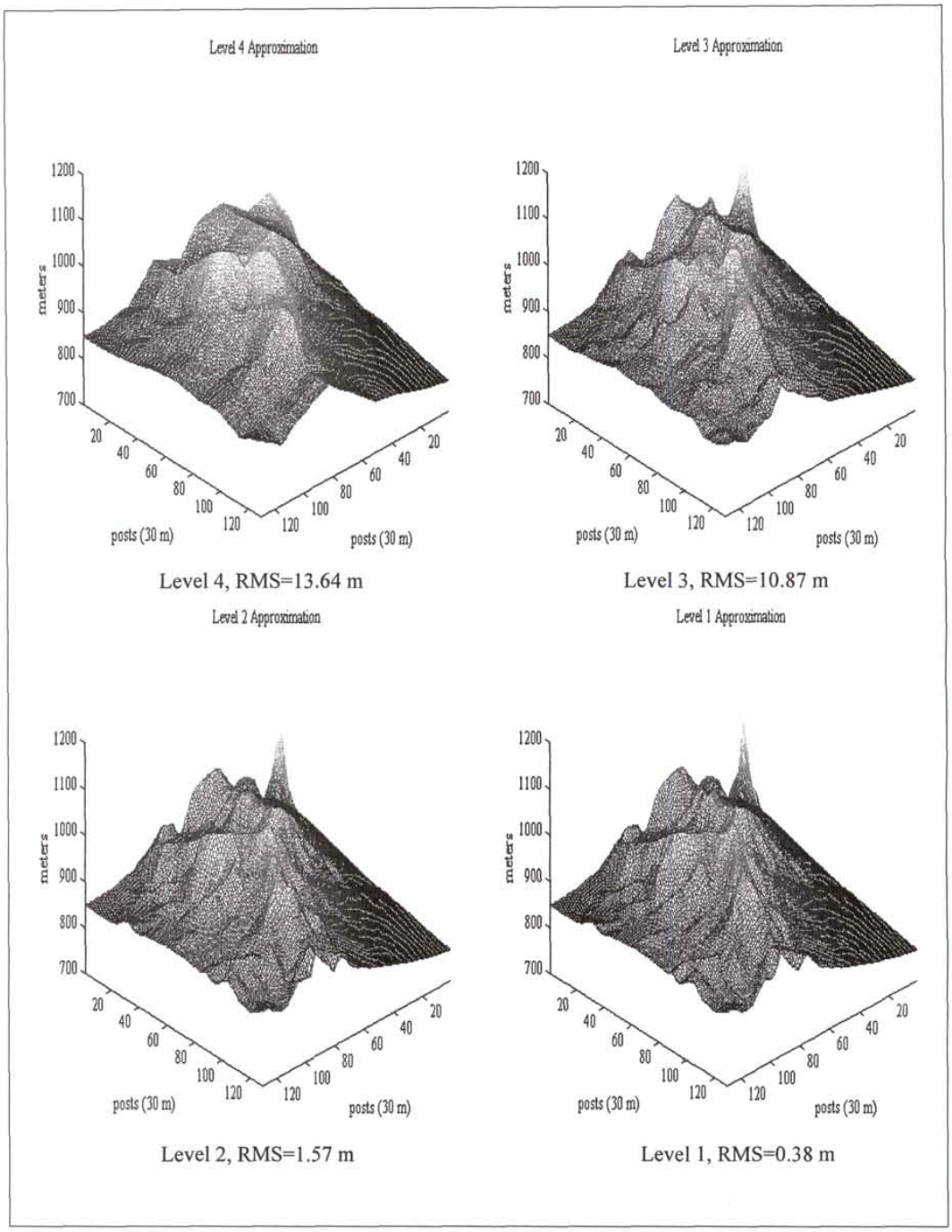


Figure 2.6: Fort Irwin terrain wavelet approximations [37]. Reproduced with permission from the American Society for Photogrammetry & Remote Sensing, Bethesda, MD.

by receiving only the wavelet coefficients associated with the relevant map scale. The client keeps track of what data has already been received to prevent unnecessary downloading of the same data. Wu et al. [69] do recognize that Daubechies wavelets would achieve better compression, but to keep processing time low Haar wavelets were chosen instead.

A major bottleneck in the system is the amount of bandwidth available for transmitting data to the client. To achieve even more transmission savings, the system employs three additional compression techniques: quantization, run-length coding, and Huffman coding. The quantization technique is used to set the number of bits used to represent a coefficient. For example, larger magnitude coefficients use 8-bits, and smaller magnitude coefficients use 4-bits. This technique does not drastically affect the reconstruction of the photo because larger magnitude coefficients dominate the approximation. The authors' prototype achieves compression ratios of 10:1 for the 0.5m/pixel photos in one example case. The authors conclude that their technique achieves efficient, high-performance viewing of image data over low-bandwidth networks.

2.5 Multi-resolution databases

A GIS multi-resolution database is used to store a single base geographic dataset in such a way that multiple scale maps can be derived from it. Bertolotto and Egenhoffer [4] review various challenges with regard to progressive transmission and the storage of such databases.

Sester and Brenner [52] describe a framework for a multi-resolution database. They break down cartographic generalization into a sequence of elementary generalization operators. These operators are applied to the base data to produce

maps at the desired scale. When a user requests a new map, only the relevant change operations are applied to the current map. The goal of their approach is to reduce the redundancy of the transmitted data.

Hamid et al. [26] propose a progressive transmission technique that reduces the data transmission time by eliminating topological checks for certain vertices. Antoniou et al. [2] discuss the difficulties in the transmission of vector data. They propose a tile-based system for sending data only in the client's viewport. Yang et al. [70] investigate efficient transmission of vector data over the Internet. They propose a rule-based point reduction method that maintains topological consistency during progressive data transmission. Ramos et al. [48] describes an efficient way to store the multiple simplified polylines that result from execution of the Douglas–Peucker algorithm.

Chapter 3

Wavelets

Wavelets are a mathematical tool to approximate functions. They can also be viewed from a signal processing perspective, in which an input signal is processed by a wavelet transformation into an output signal. Depending on the type of wavelet transform and the selection of wavelet parameters, different output waveforms can be achieved. We use wavelets to process our input polyline tributary into a smoothed output tributary as shown in Figure 3.1.

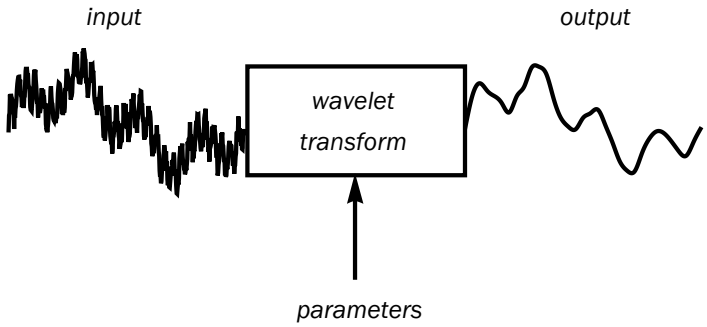


Figure 3.1: Transformation of a signal using wavelets

For a general discussion of wavelets see Daubechies [14]. Our presentation follows the conventions in the book by Urban [63].

This chapter provides a brief tutorial on wavelets and gives a compilation of reference materials. Since our research expands standard wavelet theory, it is appropriate to present a unified wavelet theory description. We start with basic wavelet concepts, including masks and scaling functions. We introduce orthogonal systems and multi-resolution analysis.

Next, we discuss the structure of orthogonal and biorthogonal wavelet families. Then, we give examples of biorthogonal wavelet families with symmetric masks. The Discrete Wavelet Transform (DWT) establishes how decomposition coefficients at different expansion levels are related. This relationship is usually described using signal processing terminology. Accordingly, we define the low and high pass filters used in DWT.

3.1 Wavelet concepts

Given function $f(t),\ t \in \mathbb{R}$, its wavelet decomposition consists of a sequence of progressively finer *frames* (approximations) $f_j(t),\ t \in \mathbb{R},\ j = 0, 1, \ldots$ The frames f_j approach f as $j \to \infty$. Any frame f_j is a combination of dilated and translated versions $\varphi_{j,k},\ k \in \mathbb{Z}$ of a fixed *scaling function* $\varphi(t),\ t \in \mathbb{R}$. Function φ has a finite support, i.e. it vanishes outside of a bounded interval $I \subset \mathbb{R}$.

The difference $d_j(t) = f_{j+1}(t) - f_j(t),\ t \in \mathbb{R}$, between the finer frame f_{j+1} and the preceding coarser frame f_j, is called the *details*. Consequently, producing a finer frame f_{j+1} (better approximation for f) from a coarser frame f_j amounts to adding the details to it: $f_{j+1} = f_j + d_j$. Similarly, the difference between any two frames f_J and f_K with $J > K$ is the sum of the corresponding details, or

$f_J(t) = f_K(t) + d_K(t) + d_{K+1}(t) + \cdots + d_{J-1}(t),\ t \in \mathbb{R}$. We can say that the fine frame f_J is the sum of the coarse frame f_K and all the details corresponding to the *expansion levels* between K and $J-1$. The advantage of working with the details, rather than the frames, is that the details become small for finer expansion levels. Small values can be disregarded, which allows for higher compression rates.

Any frame f_j is a combination of dilated and translated versions $\varphi_{j,k},\ k \in \mathbb{Z}$ of the scaling function φ. Similarly, the details can be represented as a combination of dilated and translated versions $\psi_{j,k},\ k \in \mathbb{Z}$ of a fixed *wavelet* function $\psi(t),\ t \in\ \mathbb{R}$. These combinations are written as

$$f_j(t) = \sum_{k\in\mathbb{Z}} c_{j,k}\varphi_{j,k}(t), \quad \text{and} \quad d_j(t) = \sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(t), \quad t \in \mathbb{R}.$$

The main idea of the wavelet method is to work with the *coefficients* $c_{j,k}$ and $d_{j,k}$, rather than with the corresponding frames f_j and the details d_j. Given a frame f_j, its sequence of coefficients $c_{j,k},\ k \in \mathbb{Z}$ is denoted by $\mathbf{c}_j$. Thus $c_{j,k} = (\mathbf{c}_j)_k,\ k \in \mathbb{Z}$. Similarly, the sequence of the wavelet coefficients $d_{j,k}$ is denoted by $\mathbf{d}_j$. Coefficient sequences $\mathbf{c}_j$ and $\mathbf{d}_j$ are called *signals*, since they are treated using signal processing methods.

Wavelet transforms are categorized by *class* and further categorized by *family*. There are several wavelet classes. Our system is based on biorthogonal wavelets, which are a generalization of orthogonal wavelets. In the next sections, we give detailed descriptions of orthogonal wavelets, biorthogonal wavelets, the Discrete Wavelet Transform.

3.2 Orthogonal wavelets

3.2.1 Naming conventions

Before we begin our discussion of orthogonal wavelets, it is important to point out there are multiple notations commonly used to refer to a wavelet family. We will follow the notation used by Urban [63], in which the families are denoted by $db1, db2, db3, \ldots, dbN, \ldots$ This is equivalent to the alternative notation $DAUB2$, $DAUB4$, $DAUB6$, $\ldots, DAUB2N, \ldots$ used by Daubechies [14] and Press, et al. [46].

3.2.2 Mask and scaling function

Each family of orthogonal wavelets dbN is defined by the mask $\mathbf{a}$ and the scaling function φ. A *mask* is a finite sequence

$$\mathbf{a} = (a_0, a_1, a_2, \ldots, a_{2N-1}),$$

where N is the number appearing in the dbN designation of the family. The length of the mask $\mathbf{a}$ is $2N$.

The *scaling function* φ is defined by the equation

$$\varphi(t) = \sum_{k=0}^{2N-1} a_k \varphi(2t-k), \quad t \in \mathbb{R}. \tag{3.1}$$

This equation is called the *refinement equation* because φ is defined through a refined version of itself. Also note that the coefficients of the refinement equation are from the mask $\mathbf{a}$.

Additionally, the scaling function φ is normalized by

$$\sum_{k\in\mathbb{Z}} \varphi(t-k) = 1, \quad t \in \mathbb{R}. \tag{3.2}$$

With this normalization, the scaling function φ satisfying (3.1) is unique, [63, Prop. 2.6].

For example, a commonly referenced orthogonal wavelet family is the *Haar* wavelet family. It is also known as *db*1. Its mask is

$$\mathbf{a} = (a_0, a_1) = (1, 1),$$

and its refinement equation is

$$\varphi(t) = a_0\varphi(2t-0) + a_1\varphi(2t-1) = \varphi(2t) + \varphi(2t-1), \quad t \in \mathbb{R}. \tag{3.3}$$

In this case, there is an explicit solution for the refinement equation (3.3)

$$\varphi^{Haar}(t) = \begin{cases} 1, & 0 \le t < 1, \\ 0, & \text{otherwise.} \end{cases}$$

Now consider the *db*2 orthogonal wavelet family. Its mask is

$$\mathbf{a} = (a_0, a_1, a_2, a_3) = (0.68301, 1.18301, 0.31698, -0.18301). \tag{3.4}$$

More precise values of a_k for *db*2 can be found in Urban [63]. The corresponding

refinement equation for $db2$ is

$$\varphi(t) = a_0\varphi(2t-0) + a_1\varphi(2t-1) + a_2\varphi(2t-2) + a_3\varphi(2t-3), \quad t \in \mathbb{R}. \quad (3.5)$$

For orthogonal wavelet families dbN with $N \geq 2$ it is proved that it is impossible to find an explicit closed-form scaling function φ [14]. However, we can still compute the values of φ by using the Cascade Algorithm that is presented in section 4.5. Our smoothing method uses this algorithm.

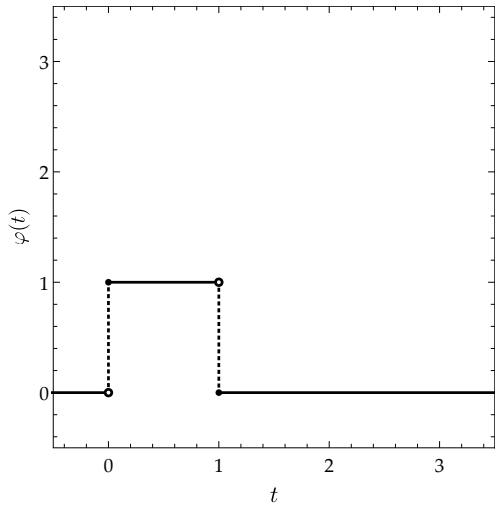


Figure 3.2: $db1$ scaling function

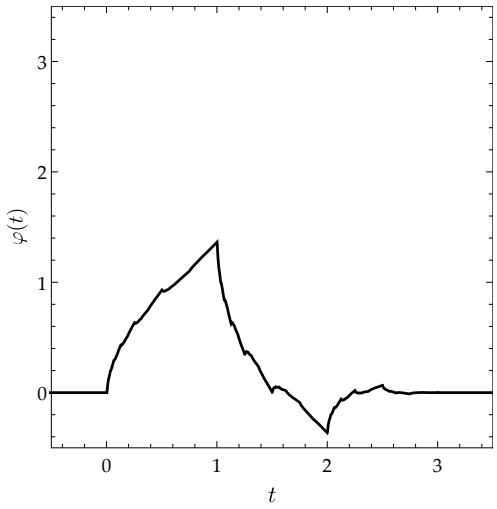


Figure 3.3: $db2$ scaling function

3.2.3 Orthogonal systems and expansions

To continue our discussion of orthogonal wavelet families it is necessary to define some operations.

The *inner product* (dot product) of two vectors $\vec{b}$ and $\vec{c}$ in $\mathbb{R}^2$ is defined by

$$\vec{b} \cdot \vec{c} = \langle b_1, b_2 \rangle \cdot \langle c_1, c_2 \rangle = b_1c_1 + b_2c_2.$$

The *norm* of $\vec{b}$ is defined by

$$\|\vec{b}\| = \sqrt{\vec{b} \cdot \vec{b}} = \sqrt{b_1^2 + b_2^2}.$$

The coordinate vectors $\{\vec{i},\ \vec{j}\}$ form an orthonormal system in $\mathbb{R}^2$, since

$$\vec{i} \cdot \vec{j} = \langle 1, 0\rangle \cdot \langle 0, 1\rangle = 0, \quad \|\vec{i}\| = \sqrt{1^2 + 0^2} = 1, \quad \|\vec{j}\| = \sqrt{0^2 + 1^2} = 1.$$

Let $\vec{b} \in \mathbb{R}^2$. If

$$\vec{b} = s_1\vec{i} + s_2\vec{j},$$

then we can find the coefficients s_1 and s_2 as follows. Take the dot product of $\vec{b}$ with $\vec{i}$. This gives

$$\vec{b} \cdot \vec{i} = (s_1\vec{i} + s_2\vec{j}) \cdot \vec{i} = s_1\vec{i} \cdot \vec{i} + s_2\vec{j} \cdot \vec{i} = s_1\|\vec{i}\|^2 = s_1,$$

because of the orthogonality of $\vec{i}$ and $\vec{j}$.

Thus $s_1 = \vec{b} \cdot \vec{i}$. Similarly $s_2 = \vec{b} \cdot \vec{j}$. Therefore, writing $\vec{b} \cdot \vec{i} = \langle \vec{b}, \vec{i}\rangle$ and $\vec{b} \cdot \vec{j} = \langle \vec{b}, \vec{j}\rangle$ we get the expansion

$$\vec{b} = \langle \vec{b}, \vec{i}\rangle\vec{i} + \langle \vec{b}, \vec{j}\rangle\vec{j}. \tag{3.6}$$

Now suppose that functions $\{w_k,\ k \in \mathbb{Z}\}$ form an orthonormal system in $L^2(\mathbb{R})$. This means that the inner products are

$$\langle w_k, w_l\rangle = \int_{-\infty}^{\infty} w_k(t)w_l(t)dt = 0, \quad k, l \in \mathbb{Z},\ k \neq l$$

and the norms are

$$\|w_k\| = \sqrt{\int_{-\infty}^{\infty} |w_k(t)|^2 dt} = 1, \quad k \in \mathbb{Z}.$$

Suppose that

$$f(t) = \sum_{k\in\mathbb{Z}} s_k w_k(t), \quad t \in \mathbb{R}. \tag{3.7}$$

Then, taking the inner product of f with w_k, we get $s_k = \langle f, w_k\rangle,\ k \in \mathbb{Z}$, because of the orthonormality of the system $\{w_k,\ k \in \mathbb{Z}\}$. Thus expansion (3.7) can be written as

$$f(t) = \sum_{k\in\mathbb{Z}} \langle f, w_k\rangle w_k(t), \quad t \in \mathbb{R}, \tag{3.8}$$

similarly to the $\mathbb{R}^2$ example (3.6).

If f does not allow the representation (3.7), then

$$f \neq \sum_{k\in\mathbb{Z}} \langle f, w_k\rangle w_k.$$

However, this sum can be considered an approximation of f. To say it more precisely, we need to introduce the notion of a *span* of a system of functions. The span of the system of functions $\{w_k\}$ is defined by

$$span\{w_k,\ k \in \mathbb{Z}\} = \Big\{ \sum_k c_k w_k \Big\},$$

where the sum is over finitely many elements. Now we define the closure of the span $\overline{span}$ as all the limits of the elements of the *span* in $L^2(\mathbb{R})$.

Let the subspace M be defined by

$$M = \overline{span}\{w_k,\ k \in \mathbb{Z}\} \subset L^2(\mathbb{R}),$$

and the approximation f_M be

$$f_M = \sum_{k\in\mathbb{Z}} \langle f, w_k\rangle w_k.$$

We say that f_M is the *projection* of f onto M.

3.2.4 Multiresolution analysis

We define dilated (compressed/stretched) and translated (shifted) versions of the scaling function φ by

$$\varphi_{j,k}(t) = \sqrt{2}^j \varphi(2^j t - k), \quad j,k \in \mathbb{Z}, \quad t \in \mathbb{R}. \tag{3.9}$$

For $j = 0$ we have

$$\varphi_{0,0}(t) = \varphi(t), \quad \varphi_{0,1}(t) = \varphi(t-1), \quad \varphi_{0,-5}(t) = \varphi(t+5),$$

and so on. For $j = 1$ we have

$$\varphi_{1,0}(t) = \sqrt{2}\,\varphi(2t), \quad \varphi_{1,1}(t) = \sqrt{2}\,\varphi(2t-1), \quad \varphi_{1,-5}(t) = \sqrt{2}\,\varphi(2t+5),$$

and so on.

The coefficients of the mask $\mathbf{a}$ in (3.4) are chosen in such a way that, for a fixed $j \in \mathbb{Z}$, the system $\{\varphi_{j,k},\ k \in \mathbb{Z}\}$ is orthogonal in $L^2(\mathbb{R})$, i.e. $\langle \varphi_{j,k}, \varphi_{j,l}\rangle = 0,\ k,l \in \mathbb{Z}$, if $k \neq l$. The normalization factor $\sqrt{2}^j$ makes the system orthonormal in $L^2(\mathbb{R})$, i.e $\|\varphi_{j,k}\| = 1,\ k \in \mathbb{Z}$.

For each $j \in \mathbb{Z}$ let the subspace $S_j \subset L^2(\mathbb{R})$ be defined by

$$S_j = \overline{span}\{\varphi_{j,k},\ k \in \mathbb{Z}\}. \tag{3.10}$$

where the *span* means that we consider all possible *finite* sums

$$f(t) = \sum_k s_{j,k}\varphi_{j,k}(t), \quad t \in \mathbb{R}$$

for some coefficients $s_{j,k}$. For example, for $j = 0$

$$\begin{aligned} f(t) &= 3.5\varphi_{0,-1}(t) - 2.2\varphi_{0,1}(t) + 17\varphi_{0,2}(t) \\ &= 3.5\varphi(t+1) - 2.2\varphi(t-1) + 17\varphi(t-2) \in span\{\varphi_{0,k},\ k \in \mathbb{Z}\}. \end{aligned}$$

The bar over the *span* (the closure in $L^2(\mathbb{R})$) means that we can also take *infinite* sums

$$f = \sum_{k\in\mathbb{Z}} s_{j,k}\varphi_{j,k}$$

as long as the resulting function f is in $L^2(\mathbb{R})$.

Let $f \in S_0$. This means that the function f can be represented as

$$f(t) = \sum_{k\in\mathbb{Z}} s_{0,k}\varphi_{0,k}(t) = \sum_{k\in\mathbb{Z}} s_{0,k}\varphi(t-k), \quad t \in \mathbb{R} \tag{3.11}$$

for some coefficients $s_{0,k} \in \mathbb{R},\ k \in \mathbb{Z}$.

The scaling function $\varphi \in S_0$. Rewriting the refinement equation (3.1) using (3.9) we get

$$\varphi(t) = \sum_{k=0}^{2N-1} a_k \frac{1}{\sqrt{2}}\sqrt{2}\,\varphi(2t-k) = \sum_{k=0}^{2N-1} \frac{a_k}{\sqrt{2}}\varphi_{1,k}(t), \quad t \in \mathbb{R}.$$

Therefore $\varphi \in S_1$ in addition to $\varphi \in S_0$.

In the same way we argue that any function $f \in S_0$ can also be represented by functions $\varphi_{1,k}(x),\ k \in \mathbb{Z}$. This means that any such function $f \in S_1$. That is, $S_0 \subset S_1$. Extending this reasoning to arbitrary $j \in \mathbb{Z}$, we get $S_j \subset S_{j+1},\ j \in \mathbb{Z}$. This is called the *multiresolution analysis* [63, definition 2.3].

The conclusion $S_j \subset S_{j+1},\ j \in \mathbb{Z}$ can be restated by saying that any function f that can be represented by *coarse* functions $\varphi_{j,k},\ k \in \mathbb{Z}$

$$f = \sum_{k \in \mathbb{Z}} s_{j,k} \varphi_{j,k} \tag{3.12}$$

can also be represented by *finer* functions $\varphi_{j+1,k},\ k \in \mathbb{Z}$,

$$f = \sum_{k \in \mathbb{Z}} s_{j+1,k} \varphi_{j+1,k}. \tag{3.13}$$

The exact transition between the coefficients $s_{j,k}$ and $s_{j+1,k}$ is handled best using the signal processing methods described in section 3.4.

The above description can be summarized by stating that any function f_j from S_j (i.e., of the form (3.12)) can also be represented in the form (3.13) in S_{j+1} with no loss of information. However, if one chooses an arbitrary $f_{j+1} \in S_{j+1}$, then its representation (*frame*) f_j using only the coarser functions $\varphi_{j,k}$ *may* cause a loss of information. We say that such a representation in S_j will lack the *details* d_j which are defined by

$$d_j(t) = f_{j+1}(t) - f_j(t), \quad t \in \mathbb{R}. \tag{3.14}$$

Therefore we have $f_{j+1} = f_j + d_j$. That is, the finer representation consists of the coarser representation and the details.

We would like to represent the details d_j in a form similar to (3.12), so that they can be handled by a signal processing method.

3.2.5 dbN wavelets

Consider a dbN family with mask $\mathbf{a}$, and scaling function φ. The mask $\mathbf{a}$ is extended periodically to $\mathbb{Z}$ with the period $2N$.

The *wavelet* function ψ for this family is defined by

$$\psi(t) = \sum_{k=0}^{2N-1} b_k \varphi(2t-k), \quad t \in \mathbb{R}, \tag{3.15}$$

where

$$b_k = (-1)^k a_{1-k}, \quad 0 \le k \le 2N-1, \tag{3.16}$$

[63, section 5.5, proposition 5.1].

Therefore, the wavelet for the $db1$ (Haar) family is

$$\psi^{Haar}(t) = a_1 \varphi^{Haar}(2t) - a_0 \varphi^{Haar}(2t-1) = \varphi^{Haar}(2t) - \varphi^{Haar}(2t-1), \quad t \in \mathbb{R}.$$

The wavelet for $db2$ family is

$$\psi(t) = a_1\varphi(2t) - a_0\varphi(2t-1) + a_3\varphi(2t-2) - a_2\varphi(2t-3), \quad t \in \mathbb{R}.$$

The definition of the wavelet ψ is chosen so that $\psi \perp \varphi$, in $L^2(\mathbb{R})$, in other words $\langle \psi, \varphi \rangle = 0$. This justifies the *orthogonal* wavelet name.

Just as in (3.9) we define

$$\psi_{j,k}(t) = \sqrt{2}^j \psi(2^j t - k), \quad j,k \in \mathbb{Z}, \quad t \in \mathbb{R}. \tag{3.17}$$

For each $j \in \mathbb{Z}$ let the subspace $W_j \subset L^2(\mathbb{R})$ be defined by

$$W_j = \overline{span}\{\psi_{j,k},\ k \in \mathbb{Z}\}. \tag{3.18}$$

Recall that the details d_j represent the "difference" between the frames f_{j+1} and f_j. It can be shown that $d_j \in W_j$,

$$d_j(t) = \sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(t) = \sum_{k\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle \psi_{j,k}(t), \quad t \in \mathbb{R}. \tag{3.19}$$

Note that the coefficients $d_{j,k} \in \mathbb{R}$ and the details functions d_j, use the same letter d, [63, section 5.4.1].

Since $f_{j+1} = f_j + d_j$, we can write

$$\begin{aligned} f_{j+1}(t) &= f_j(t) + d_j(t) \\ &= f_{j-1}(t) + d_{j-1}(t) + d_j(t) = \cdots = f_K(t) + \sum_{l=K}^{j} d_l(t),, \quad t \in \mathbb{R}, \end{aligned} \tag{3.20}$$

[63, section 5.2.1].

This means that the finest approximation f_{j+1} of f is represented by a very coarse approximation f_K plus all the details d_l accumulated while transitioning from the finest level $j+1$ to the coarsest level K.

In principle, the representation (3.20) of f_{j+1} by the sum of the details does not contain any new information about f_{j+1}, nor does it lose any information. The point of such a representation is that details are usually very small and can be ignored. This allows for significant storage compression of the approximations of f. More importantly, it gives an efficient and fast method for computing various approximations of f.

3.3 Biorthogonal wavelets

Orthogonal wavelets are too restrictive for many applications. For our generalization algorithm we want the tributary endpoints to remain fixed after simplification. To accomplish this we need wavelets based on symmetric masks with an odd length. The orthogonal wavelets do not have symmetric masks, so the wavelet framework has to be modified by replacing orthogonality constraints with more flexible biorthogonality constraints, see equation (3.27).

3.3.1 Naming conventions

Urban [63, section 2.7] classifies biorthogonal wavelets by a pair of positive integers d and $\tilde{d}$ with $d + \tilde{d}$ being an even number. The integers d and $\tilde{d}$ are related to the smoothness of the scaling functions; where bigger values refer to increased smoothness. Another notation for a biorthogonal wavelet family is pair $(\tilde{N}, N)$, where N is the length of the primary mask $\mathbf{a}$, and $\tilde{N}$ is the length of the dual mask $\tilde{\mathbf{a}}$. There may be several different biorthogonal wavelet families described by the same classification.

3.3.2 Masks and scaling functions

A biorthogonal wavelet family is defined by two finite masks $\mathbf{a}$ and $\tilde{\mathbf{a}}$. Additionally we require the masks to be symmetric and odd in length

$$\mathbf{a} = (a_{-M}, ..., a_0, ..., a_M), \quad \tilde{\mathbf{a}} = (\tilde{a}_{-\tilde{M}}, ..., \tilde{a}_0, ..., \tilde{a}_{\tilde{M}}). \tag{3.21}$$

Accordingly, we have two scaling functions: the primary scaling function φ, and the dual scaling function $\tilde{\varphi}$ satisfying the refinement equations

$$\varphi(t) = \sum_{k=-M}^{M} a_k \varphi(2t-k), \quad \text{and} \quad \tilde{\varphi}(t) = \sum_{k=-\tilde{M}}^{\tilde{M}} \tilde{a}_k \tilde{\varphi}(2t-k), \quad t \in \mathbb{R}, \tag{3.22}$$

and the normalization conditions

$$\sum_{k\in\mathbb{Z}} \varphi(t-k) = 1, \quad \text{and} \quad \sum_{k\in\mathbb{Z}} \tilde{\varphi}(t-k) = 1, \quad t \in \mathbb{R}. \tag{3.23}$$

Since the masks are symmetric, that is $a_{-k} = a_k$ and $\tilde{a}_{-k} = \tilde{a}_k$, the scaling functions are even (see Theorem 4.3.1). This means that $\varphi(-t) = \varphi(t)$ and $\tilde{\varphi}(-t) = \tilde{\varphi}(t)$ for any $t \in \mathbb{R}$.

Tables 3.1 - 3.4 show the coefficients for the masks $\mathbf{a}$ and $\tilde{\mathbf{a}}$ for various biorthogonal wavelet families with symmetric masks. For example, for the wavelet family (5–3) the primary mask $\mathbf{a}$ has 3 non-zero coefficients

$$\mathbf{a} = (a_{-1}, a_0, a_1) = (0.5, 1.0, 0.5),$$

and the dual mask $\tilde{\mathbf{a}}$ has 5 non-zero coefficients

$$\tilde{\mathbf{a}} = (\tilde{a}_{-2}, \tilde{a}_{-1}, \tilde{a}_0, \tilde{a}_1, \tilde{a}_2) = (-0.25, 0.50, 1.50, 0.50, -0.25).$$

Because of a different normalization adopted by Daubechies [14], our mask coefficients differ by a factor of $\sqrt{2}$ from the values given there. The (9–7) biorthogonal wavelet family described in Table 3.4 is also known as the FBI wavelet, because it is used by the Federal Bureau of Investigation to compress fingerprint images [30]. The graphs of the scaling functions φ and $\tilde{\varphi}$ for the FBI (9–7) biorthogonal

wavelet family are shown in Figures 3.4 and 3.5.

Table 3.1: Biorthogonal family (5–3)

k	a_k	$\tilde{a}_k$
0	1.0	1.50
±1	0.5	0.50
±2		-0.25

Table 3.2: Biorthogonal family (9–3)

k	a_k	$\tilde{a}_k$
0	1.0	1.406250
±1	0.5	0.593750
±2		-0.250000
±3		-0.093750
±4		0.046875

Table 3.3: Biorthogonal family (13–3)

k	a_k	$\tilde{a}_k$
0	1.0	1.3671875
±1	0.5	0.6328125
±2		-0.240234375
±3		-0.15234375
±4		0.06640625
±5		0.01953125
±6		-0.009765625

Table 3.4: Biorthogonal family (9–7) (FBI)

k	a_k	$\tilde{a}_k$
0	1.11508705245689	1.205898036472
±1	0.59127176311341	0.533728236886
±2	-0.05754352622794	-0.156446533058
±3	-0.09127176311391	-0.033728236886
±4		0.053497514822

The wavelet coefficients are defined by

$$b_k = (-1)^k \tilde{a}_{1-k}, \quad k = -\tilde{M}+1, ..., \tilde{M}+1,$$
$$\tilde{b}_k = (-1)^k a_{1-k}, \quad k = -M+1, ..., M+1, \tag{3.24}$$

[63, equation 5.16]. Note that the primary wavelet coefficients b_k are defined using the dual mask $\tilde{\mathbf{a}}$. The primary and the dual wavelets are defined by

$$\psi(t) = \sum_{k=-\tilde{M}+1}^{\tilde{M}+1} b_k \varphi(2t-k), \quad \text{and} \quad \tilde{\psi}(t) = \sum_{k=-M+1}^{M+1} \tilde{b}_k \tilde{\varphi}(2t-k), \quad t \in \mathbb{R}. \tag{3.25}$$

For example, if a symmetric mask $\mathbf{a}$ has length $N = 5$ ($M = 2$), and

$$\mathbf{a} = (a_{-2}, a_{-1}, a_0, a_1, a_2) = (a_2, a_1, a_0, a_1, a_2),$$

then

$$\tilde{\mathbf{b}} = (b_{-1}, b_0, b_1, b_2, b_3) = (-a_2, a_1, -a_0, a_1, -a_2).$$

Note that the wavelet mask $\tilde{\mathbf{b}}$ has the same length 5 as $\mathbf{a}$, but it is *not* symmetric, since its indices run from -1 to 3.

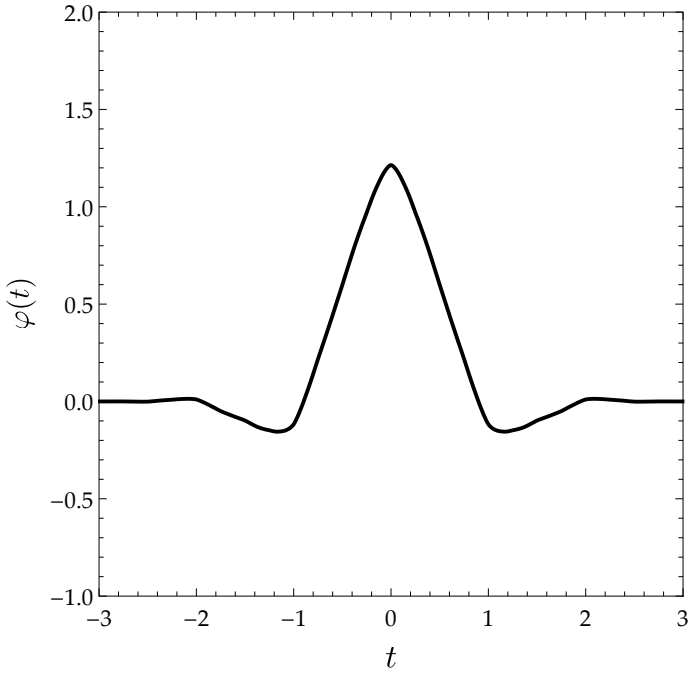


Figure 3.4: Primary scaling function φ for the FBI (9–7) biorthogonal wavelet family

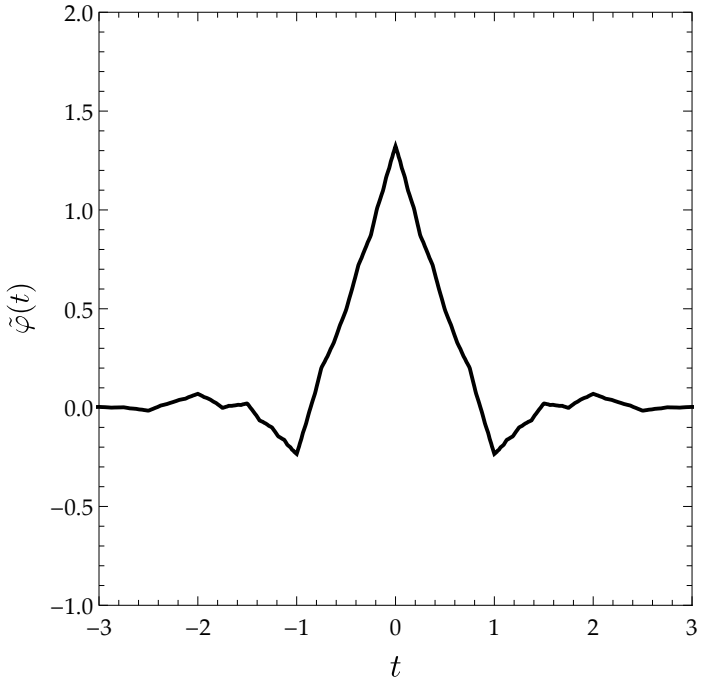


Figure 3.5: Dual scaling function $\tilde{\varphi}$ for the FBI (9–7) biorthogonal wavelet family

The translated and dilated versions of the scaling functions and the wavelets are defined by

$$\varphi_{j,k}(t) = \sqrt{2}^j \varphi(2^j t - k), \quad \psi_{j,k}(t) = \sqrt{2}^j \psi(2^j t - k), \quad j,k \in \mathbb{Z}, \quad t \in \mathbb{R}. \quad (3.26)$$

The dual scaling functions and the wavelets are defined similarly. Let $\langle \cdot, \cdot \rangle$ be the inner product in $L^2(\mathbb{R})$. The biorthogonality conditions [63, sections 2.7 and 5.3] are

$$\langle \psi_{j,k}, \tilde{\psi}_{l,m} \rangle = \delta_{j,l}\delta_{k,m}, \quad \langle \varphi_{j,k}, \tilde{\varphi}_{j,m} \rangle = \delta_{k,m}, \quad (3.27)$$

where $\delta_{k,m} = 0$ if $k \neq m$, and $\delta_{k,m} = 1$ if $k = m$.

Then the expansions of the frames and the details can be written as

$$f_j(t) = \sum_{k \in \mathbb{Z}} c_{j,k} \varphi_{j,k}(t), \quad (3.28)$$

where $c_{j,k} = \langle f, \tilde{\varphi}_{j,k} \rangle,\ j,k \in \mathbb{Z}$, and

$$d_j(t) = \sum_{k \in \mathbb{Z}} d_{j,k} \psi_{j,k}(t) = \sum_{k \in \mathbb{Z}} \langle f, \tilde{\psi}_{j,k} \rangle \psi_{j,k}(t), \quad t \in \mathbb{R}, \quad (3.29)$$

[63, sections 5.3 and 5.4]. The multiresolution equation is

$$\begin{aligned} f_J(t) = f_{J-1}(t) + d_{J-1}(t) = f_{J-2}(t) + d_{J-2}(t) + d_{J-1}(t) = \cdots \\ = f_K(t) + \sum_{j=K}^{J-1} d_j(t), \quad t \in \mathbb{R}. \end{aligned} \quad (3.30)$$

It describes the decomposition of a fine level frame f_J into a coarse level frame f_K and the details d_j for all levels between K and $J-1$.

3.3.3 Convergence of the approximations

According to the theory of biorthogonal wavelets [63, section 5.3], we can estimate the $L^2(\mathbb{R})$ norm of the difference $f_J - f_K$ in terms of the detail coefficients of f. We have

$$f_J(t) - f_K(t) = \sum_{j=K}^{J-1} d_j(t) = \sum_{j=K}^{J-1} \sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(t), \quad t \in \mathbb{R}, \tag{3.31}$$

and

$$c_\psi \sum_{j=K}^{J-1} \sum_{k\in\mathbb{Z}} |d_{j,k}|^2 \leq \left\| \sum_{j=K}^{J-1} \sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k} \right\|^2_{L^2(\mathbb{R})} \leq C_\psi \sum_{j=K}^{J-1} \sum_{k\in\mathbb{Z}} |d_{j,k}|^2, \tag{3.32}$$

where the constants c_ψ and C_ψ depend only on the biorthogonal wavelet family. This means that they do not depend on a particular function f.

3.4 Discrete Wavelet Transform

The central part of the wavelet transform method is a fast transition between coarse and fine level *coefficients* $c_{j,k}$ for the frames $f_j(x)$, and for *coefficients* $d_{j,k}$ for the details $d_j(x)$. This is usually explained in signal processing terms [39, 57].

Let

$$\mathbf{u} = (u_0, u_1, \ldots, u_{2^N-1})$$

be a discrete signal of length 2^N. We assume that the signal is somehow extended beyond its range of indices, that is, for all $k \in \mathbb{Z}$. One typical extension method is by zero padding. Then signal $\mathbf{u}$ becomes

$$\mathbf{u}_{zero} = (\ldots, 0, 0, 0, u_0, u_1, \ldots, u_{2^N-1}, 0, 0, 0, \ldots).$$

Another common way to extend $\mathbf{u}$ is periodically:

$$\mathbf{u}_{periodic} = (\dots, u_0, u_1, \dots, u_{2^N-1}, u_0, u_1, \dots, u_{2^N-1}, u_0, u_1, \dots, u_{2^N-1}, \dots).$$

In our application the signal is extended by inverse periodic mirroring (explained in Chapter 4). However, in this section there are no restrictions on the extension of the signal $\mathbf{u}$.

3.4.1 Elementary signal operations

First, we define some elementary operations for arbitrary signals $\mathbf{u}$ and $\mathbf{v}$, [63, section 5.4].

Mirror

$$(\mathbf{u}^{\updownarrow})_k = u_{-k}, \quad k \in \mathbb{Z}. \tag{3.33}$$

For example

$$\mathbf{u} = (\dots, u_{-1}, \underbrace{u_0}_{k=0}, u_1, u_2, \dots), \quad \mathbf{u}^{\updownarrow} = (\dots, u_2, u_1, \underbrace{u_0}_{k=0}, u_{-1}, \dots).$$

That is, the pivot for the mirroring operation is the element at the index $k = 0$, which retains its position.

Downsampling

$$(\downarrow \mathbf{u})_k = u_{2k}, \quad k \in \mathbb{Z}, \tag{3.34}$$

For example

$$\mathbf{u} = (\dots, u_{-2}, u_{-1}, u_0, u_1, u_2, \dots), \quad \downarrow \mathbf{u} = (\dots, u_{-2}, \underbrace{u_0}_{k=0}, u_2, \dots).$$

Upsampling

$$(\uparrow \mathbf{u})_k = \begin{cases} u_m, & k = 2m, \\ 0, & k = 2m+1 \end{cases}, \quad k \in \mathbb{Z}, \tag{3.35}$$

For example

$$\mathbf{u} = (\dots, u_{-2}, u_{-1}, u_0, u_1, u_2, \dots), \quad \uparrow \mathbf{u} = (\dots, u_{-2}, 0, u_{-1}, 0, \underbrace{u_0}_{k=0}, 0, u_1, 0, u_2, \dots).$$

Dot product

Given signals $\mathbf{u}$ and $\mathbf{v}$, their dot product $\mathbf{u} \cdot \mathbf{v}$ is the number defined by

$$\mathbf{u} \cdot \mathbf{v} = \sum_{m \in \mathbb{Z}} u_m v_m, \quad k \in \mathbb{Z}. \tag{3.36}$$

Convolution

Given signals $\mathbf{u}$ and $\mathbf{v}$, their convolution $\mathbf{u} * \mathbf{v}$ is a new signal, whose components are defined by

$$(\mathbf{u} * \mathbf{v})_k = \sum_{m \in \mathbb{Z}} u_m v_{k-m}, \quad k \in \mathbb{Z}. \tag{3.37}$$

For example, assume that

$$\mathbf{u} = (\dots, u_{-2}, u_{-1}, u_0, u_1, u_2, \dots), \quad \mathbf{v} = (v_{-1}, v_0, v_1).$$

That is, all other elements of $\mathbf{v}$ are zeros. Then

$$(\mathbf{u} * \mathbf{v})_0 = u_{-1}v_1 + u_0v_0 + u_1v_{-1}, \quad (\mathbf{u} * \mathbf{v})_1 = u_0v_1 + u_1v_0 + u_2v_{-1},$$

and so on. In other words, the convolution component $(\mathbf{u} * \mathbf{v})_k$ is the dot product of the signal $\mathbf{u}$ with the mirrored signal $\mathbf{v}^{\updownarrow}$ that is shifted k places to the right.

3.4.2 Low-pass and High-pass filters

Let $\mathbf{u}$ be a signal. Let $\mathbf{a}$ and $\tilde{\mathbf{a}}$ be the masks associated with a biorthogonal wavelet family. If the masks are symmetric, then $\mathbf{a} = \mathbf{a}^{\updownarrow}$, and $\tilde{\mathbf{a}} = \tilde{\mathbf{a}}^{\updownarrow}$. Recall that the wavelet masks $\mathbf{b}$ and $\tilde{\mathbf{b}}$ were defined in (3.24), and they are not symmetric.

Define the Low-pass filter L_-, and its dual L_+ by

$$L_-\mathbf{u} = \frac{1}{\sqrt{2}} \downarrow (\mathbf{u} * \tilde{\mathbf{a}}^{\updownarrow}), \tag{3.38}$$

$$L_+\mathbf{u} = \frac{1}{\sqrt{2}} (\uparrow \mathbf{u}) * \mathbf{a}. \tag{3.39}$$

Define the High-pass filter H_-, and its dual H_+ by

$$H_-\mathbf{u} = \frac{1}{\sqrt{2}} \downarrow (\mathbf{u} * \tilde{\mathbf{b}}^{\updownarrow}), \tag{3.40}$$

$$H_+\mathbf{u} = \frac{1}{\sqrt{2}} (\uparrow \mathbf{u}) * \mathbf{b}, \tag{3.41}$$

as in Urban [63, section 5.4].

The Discrete Wavelet Transform (DWT) establishes the following transition

rules between the frame and the detail signals

$$\mathbf{c}_{j-1} = L_-\mathbf{c}_j, \quad \mathbf{d}_{j-1} = H_-\mathbf{c}_j, \quad \mathbf{c}_j = L_+\mathbf{c}_{j-1} + H_+\mathbf{d}_{j-1}. \tag{3.42}$$

The last equation implies that

$$\mathbf{c}_j = L_+\mathbf{c}_{j-1} + H_+\mathbf{d}_{j-1} = L_+L_-\mathbf{c}_j + H_+H_-\mathbf{c}_j = (L_+L_- + H_+H_-)\mathbf{c}_j,$$

or $L_+L_- + H_+H_- = Identity$. Therefore the DWT algorithm gives lossless reconstruction of the data. The rules (3.42) are algebraic forms of the refinement equations (3.22), [63, section 5.4].

Chapter 4

Dimensional wavelets and IMP representations

This chapter describes our new method for smoothing linear features. It constitutes the main theoretical contribution of the dissertation. It is based on three major components:

1. Functions represented in special symmetric and periodic form. We call such representations *Inverse Mirror Periodic* (IMP) form.

2. Biorthogonal wavelet families with symmetric scaling functions

3. Appropriate scaling of the chosen biorthogonal wavelet to the endpoints of the function. We call such wavelet families *dimensional wavelets* because they reflect the length of the linear feature they approximate.

Our main results are the theorems and their proofs in section 4.3. They state that IMP functions and signals retain their IMP structure under wavelet transformations.

First, we introduce our dimensional scaling functions, and IMP functions and signals. Then we prove our main results. The Discrete Wavelet Transform (DWT) provides a method for transitioning between expansion levels. However, to start this process, it is necessary to know frame expansion coefficients for a fine expansion level. We call such a level the *oversampling level*. We show that the determination of the expansion coefficients at this level amounts to sampling of the function. Finally, we describe the Cascade algorithm, which is used to compute values of the scaling and wavelet functions.

4.1 Dimensional scaling functions

Let φ be a scaling function, and $h > 0$. Define

$$\varphi^{(h)}(t) = \varphi\left(\frac{t}{h}\right), \quad t \in \mathbb{R}. \tag{4.1}$$

We call $\varphi^{(h)}$ a dimensional scaling function because it is scaled by length h.

Then, according to the refinement equation (3.22)

$$\varphi^{(h)}(t) = \varphi\left(\frac{t}{h}\right) = \sum_{k=-M}^{M} a_k \varphi\left(2\frac{t}{h} - k\right) = \sum_{k=-M}^{M} a_k \varphi^{(h)}(2t - kh), \quad t \in \mathbb{R}. \tag{4.2}$$

By the normalization condition (3.23)

$$\sum_{k\in\mathbb{Z}} \varphi^{(h)}(t - kh) = \sum_{k\in\mathbb{Z}} \varphi\left(\frac{t - kh}{h}\right) = \sum_{k\in\mathbb{Z}} \varphi\left(\frac{t}{h} - k\right) = 1, \quad t \in \mathbb{R}. \tag{4.3}$$

Using the substitution $t = y + k$, and the normalization condition (3.23) we get

$$\int_{\mathbb{R}} \varphi(t)dt = \sum_{k\in\mathbb{Z}} \int_k^{k+1} \varphi(t)dt = \sum_{k\in\mathbb{Z}} \int_0^1 \varphi(y+k)dy = \int_0^1 \left[\sum_{k\in\mathbb{Z}} \varphi(y+k)\right] dy = 1.$$

The substitution $y = t/h$ gives

$$\int_{\mathbb{R}} \varphi^{(h)}(t)dt = \int_{\mathbb{R}} \varphi\left(\frac{t}{h}\right) dt = h \int_{\mathbb{R}} \varphi(y)dy = h. \tag{4.4}$$

Similar conclusions are held for the function $\tilde{\varphi}^{(h)}$. To summarize, functions $\varphi^{(h)}$ and $\tilde{\varphi}^{(h)}$ are the scaling functions satisfying modified refinement equations

$$\varphi^{(h)}(t) = \sum_{k=-\tilde{M}}^{\tilde{M}} a_k \varphi^{(h)}(2t - kh), \quad \tilde{\varphi}^{(h)}(t) = \sum_{k=-\tilde{M}}^{\tilde{M}} \tilde{a}_k \tilde{\varphi}^{(h)}(2t - kh), \quad t \in \mathbb{R}, \tag{4.5}$$

the normalization conditions

$$\sum_{k=-M}^{M} \varphi^{(h)}(t - kh) = 1, \quad \sum_{k=-\tilde{M}}^{\tilde{M}} \tilde{\varphi}^{(h)}(t - kh) = 1, \quad t \in \mathbb{R}, \tag{4.6}$$

and

$$\int_{\mathbb{R}} \varphi^{(h)}(t)dt = h, \quad \int_{\mathbb{R}} \tilde{\varphi}^{(h)}(t)dt = h. \tag{4.7}$$

The translated and dilated versions of the new scaling functions and the wavelets are defined by

$$\varphi_{j,k}^{(h)}(t) = \frac{\sqrt{2}^j}{\sqrt{h}} \varphi^{(h)}(2^j t - kh), \quad t \in \mathbb{R}, \quad j, k \in \mathbb{Z}, \quad \text{etc.} \tag{4.8}$$

The direct verification shows that they satisfy the biorthogonality conditions (3.27). Therefore all the results of section 3.3 remain valid for the subfamily of

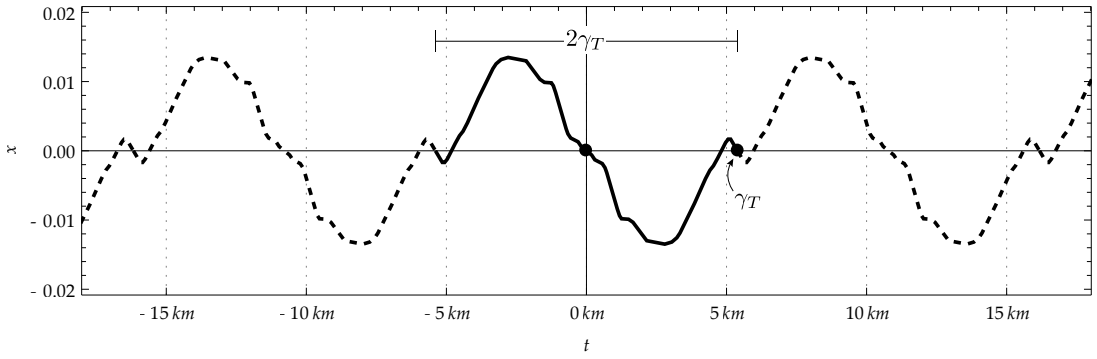


Figure 4.1: Inverse mirror periodic (IMP) representation of $x(t)$

the dimensional scaling functions and wavelets $\varphi^{(h)}, \tilde{\varphi}^{(h)}$ and $\psi^{(h)}, \tilde{\psi}^{(h)}$.

4.2 IMP functions and signals

Definition 4.2.1. Function $f(t),\ t \in \mathbb{R}$ is called Inverse Mirror Periodic (IMP) of length γ, if

1. $f(0) = f(\gamma) = 0$.

2. $f(-t) = -f(t)$ for any $t \in \mathbb{R}$.

3. $f(\gamma - t) = -f(\gamma + t)$ for any $t \in \mathbb{R}$.

In other words, an IMP function f of length γ is odd with respect to $t = 0$, and with respect to $t = \gamma$. Accordingly, $f(t+2\gamma) = f(\gamma+(t+\gamma)) = -f(\gamma-(t+\gamma)) = -f(-t) = f(t),\ t \in \mathbb{R}$. So, such a function is 2γ periodic.

Given a function $f(t)$ defined for $0 \leq t \leq \gamma$ with $f(0) = f(\gamma) = 0$, we can extend it for all $t \in \mathbb{R}$ as an IMP function by defining $f(-t) = -f(t)$ for $-\gamma \leq t < 0$, and then extending it as a 2γ periodic function (see Figure 4.1). The restriction of f to $[0, \gamma]$ is called the *representative part* of f.

Definition 4.2.2. A discrete signal $\mathbf{c} = (\ldots, c_0, c_1, \ldots, c_{M-1}, c_M, \ldots)$ is called Inverse Mirror Periodic (IMP) of length M, if

1. $c_0 = c_M = 0$.

2. $c_{-k} = -c_k$ for any $k \in \mathbb{Z}$.

3. $c_{M-k} = -c_{M+k}$ for any $k \in \mathbb{Z}$.

We can say that an IMP signal of length M is odd with respect to $k = 0$ and $k = M$. Furthermore, $c_{k+2M} = c_{M+(k+M)} = -c_{M-(k+M)} = -c_{-k} = c_k$ for any $k \in \mathbb{Z}$, so it is a $2M$ periodic signal. Notice that an IMP signal of length M is actually defined for any $k \in \mathbb{Z}$.

Let $\mathbf{c} = (c_0, c_1, \ldots, c_{M-1})$ be a signal of length M with $c_0 = 0$. We can extend it to an IMP signal defined for all $k \in \mathbb{Z}$ by defining $c_{-k} = -c_k$ for $k = 1, 2, \ldots, M-1$, $c_{-M} = 0$, and then extending it as a $2M$ periodic signal for all $k \in \mathbb{Z}$. The part of $\mathbf{c}$ with the indexes from 0 to $M-1$ is called the *representative part* of $\mathbf{c}$.

Definition 4.2.3. It is convenient to describe and handle IMP signals by using the *wrapping array* notation. Let $\mathbf{c} = (0, c_1, \ldots, c_{M-1})$ be a signal of length M. Then its IMP extension is the wrapping array (also denoted by $\mathbf{c}$)

$$\mathbf{c} = \overleftarrow{\{}\, 0, c_1, \ldots, c_{M-1} \overrightarrow{\}}.$$

In this case we say that the wrapping array $\mathbf{c}$ has length M, even though it is defined for all indices $k \in \mathbb{Z}$.

4.3 Operations with IMP functions and signals

The next theorem shows that the scaling function associated with a symmetric mask is symmetric (even).

Theorem 4.3.1. *If a mask* **a** *is symmetric with an odd length, then the corresponding scaling function* φ *is even, i.e.* $\varphi(t) = \varphi(-t)$ *for any* $t \in \mathbb{R}$.

Proof. By definition, the scaling function φ satisfies the refinement equation

$$\varphi(t) = \sum_{k=-M}^{M} a_k \varphi(2t-k), \quad t \in \mathbb{R}, \tag{4.9}$$

and it is normalized by

$$\sum_{k\in\mathbb{Z}} \varphi(t-k) = 1, \quad t \in \mathbb{R}. \tag{4.10}$$

With this normalization the scaling function φ satisfying (4.9) is unique, [63, Proposition 2.6].

Let $\Phi(t) = \varphi(-t)$. Use substitution $k = -m$, and the symmetry condition $a_{-m} = a_m$ to get

$$\begin{aligned}\Phi(t) = \varphi(-t) &= \sum_{k=-M}^{M} a_k \varphi(-2t-k) = \sum_{k=-M}^{M} a_k \Phi(2t+k) \\ &= \sum_{m=-M}^{M} a_{-m} \Phi(2t-m) = \sum_{m=-M}^{M} a_m \Phi(2t-m), \quad t \in \mathbb{R}. \end{aligned} \tag{4.11}$$

Thus

$$\Phi(t) = \sum_{k=-M}^{M} a_k \Phi(2t-k), \quad t \in \mathbb{R}. \tag{4.12}$$

Furthermore, using $m = -k$

$$\sum_{k\in\mathbb{Z}} \Phi(t-k) = \sum_{k\in\mathbb{Z}} \varphi(-t+k) = \sum_{m\in\mathbb{Z}} \varphi(-t-m) = 1, \quad t \in \mathbb{R}$$

by (4.10) with t being replaced by $-t$ ((4.10) is valid for *any* t). Thus Φ is another solution of (4.9) that satisfies the normalization condition (4.10). The uniqueness implies that

$$\varphi(t) = \Phi(t) = \varphi(-t), \quad t \in \mathbb{R}.$$

□

4.3.1 Main results

Theorem 4.3.2. *Let L_- and L_+ be the low-pass filters associated with a biorthogonal wavelet family with symmetric masks $\mathbf{a}$ and $\tilde{\mathbf{a}}$. Suppose that $\mathbf{c}$ is an IMP signal of length $2N$. Then signal $\mathbf{c}_- = L_-\mathbf{c}$ is an IMP signal of length N. The signal $\mathbf{c}_+ = L_+\mathbf{c}$ is an IMP signal of length $4N$.*

Proof. Recall that

$$L_-\mathbf{c} = \frac{1}{\sqrt{2}} \downarrow (\mathbf{c} * \tilde{\mathbf{a}}^{\updownarrow}).$$

First, we show that signal $\mathbf{u} = \mathbf{c} * \tilde{\mathbf{a}}^{\updownarrow}$ is IMP of length $2N$. Since $\tilde{\mathbf{a}}$ is a symmetric mask, we have $\tilde{\mathbf{a}} = \tilde{\mathbf{a}}^{\updownarrow}$. Also $\tilde{\mathbf{a}}$ is a finite mask with nonzero indices in the range $[-\tilde{M}, \tilde{M}]$. Therefore

$$(\mathbf{u})_k = (\mathbf{c} * \tilde{\mathbf{a}})_k = \sum_{m\in\mathbb{Z}} c_m \tilde{a}_{k-m} = \sum_{m\in\mathbb{Z}} c_m \tilde{a}_{m-k} = \sum_{m=k-\tilde{M}}^{k+\tilde{M}} c_m \tilde{a}_{m-k}.$$

That is, the mask $\tilde{\mathbf{a}}$ is slid k elements along $\mathbf{c}$, so that $\tilde{a}_0$ is aligned with c_k. Then we compute the dot product of these two vectors. Thus, using $n = -m$ and the fact that $\mathbf{c}$ is an IMP signal, we get

$$u_{-k} = \sum_{m=-k-\tilde{M}}^{-k+\tilde{M}} c_m \tilde{a}_{m+k} = \sum_{n=k-\tilde{M}}^{k+\tilde{M}} c_{-n} \tilde{a}_{-n+k} = -\sum_{n=k-\tilde{M}}^{k+\tilde{M}} c_n \tilde{a}_{n-k} = -u_k$$

for any $k \in \mathbb{Z}$. In particular, $u_0 = 0$.

Now, using $n = m + 4N$ and $c_{n-4N} = c_n$

$$u_{k+4N} = \sum_{m=-k+4N-\tilde{M}}^{-k+4N+\tilde{M}} c_m \tilde{a}_{m+4N-k} = \sum_{n=-k-\tilde{M}}^{-k+\tilde{M}} c_{n-4N} \tilde{a}_{n-k} = \sum_{n=-k-\tilde{M}}^{-k+\tilde{M}} c_n \tilde{a}_{n-k} = u_k.$$

Therefore $\mathbf{u}$ is $4N$ periodic. This implies

$$u_{2N-k} = -u_{-2N+k} = -u_{2N+k}, \quad k \in \mathbb{Z},$$

and, consequently, $u_{2N} = 0$. Thus $\mathbf{u}$ is an IMP signal of length $2N$.

The downsampling operator $\downarrow \mathbf{u}$ removes the odd indexes in $\mathbf{u}$. Therefore we still have $(\downarrow \mathbf{u})_{-k} = -(\downarrow \mathbf{u})_k$, since $\mathbf{u}$ is IMP. Also we see that $\downarrow \mathbf{u}$ is periodic with half the period of $\mathbf{u}$. That is $\downarrow \mathbf{u}$ is an IMP signal of length N.

A similar argument shows that $\mathbf{c}_+ = L_+\mathbf{c}$ is an IMP signal of length $4N$. □

The next theorem shows that any frame of an IMP function is an IMP signal, provided that the scaling functions are suitably chosen. Then we prove that its inverse is also true.

Theorem 4.3.3. *Let f be an IMP function of length h, and $\varphi^{(h)}$, $\tilde{\varphi}^{(h)}$ be the scaling functions associated with a biorthogonal wavelet family with symmetric*

masks. Fix an integer $j \geq 1$. Then the frame coefficients $c_{j,k},\ k \in \mathbb{Z}$ form an IMP signal of length $M = 2^j$.

Proof. By the definition

$$c_{j,k} = \left\langle f, \tilde{\varphi}_{j,k}^{(h)} \right\rangle = \int_{\mathbb{R}} f(t)\tilde{\varphi}_{j,k}^{(h)}(t)\,dt.$$

We have

$$c_{j,k} = \int_{\mathbb{R}} f(t)\tilde{\varphi}_{j,k}^{(h)}(t)\,dt = \int_{\mathbb{R}} f(t)\frac{\sqrt{2}^j}{\sqrt{h}}\tilde{\varphi}^{(h)}(2^j t - kh)\,dt$$
$$= \int_{\mathbb{R}} f(t)\frac{\sqrt{2}^j}{\sqrt{h}}\tilde{\varphi}\left(2^j\frac{t}{h} - k\right)\,dt, \quad (4.13)$$

and

$$c_{j,-k} = \int_{\mathbb{R}} f(t)\tilde{\varphi}_{j,-k}^{(h)}(t)\,dt = \int_{\mathbb{R}} f(t)\frac{\sqrt{2}^j}{\sqrt{h}}\tilde{\varphi}^{(h)}(2^j t + kh)\,dt$$
$$= \int_{\mathbb{R}} f(t)\frac{\sqrt{2}^j}{\sqrt{h}}\tilde{\varphi}\left(2^j\frac{t}{h} + k\right)\,dt. \quad (4.14)$$

Let $y = -t$. Then, using $f(-y) = -f(y)$ and $\tilde{\varphi}(-t) = \tilde{\varphi}(t)$ we get

$$c_{j,-k} = \int_{\mathbb{R}} f(-y)\frac{\sqrt{2}^j}{\sqrt{h}}\tilde{\varphi}\left(-2^j\frac{y}{h} + k\right)\,dy$$
$$= -\int_{\mathbb{R}} f(y)\frac{\sqrt{2}^j}{\sqrt{h}}\tilde{\varphi}\left(2^j\frac{y}{h} - k\right)\,dy = -c_{j,k}. \quad (4.15)$$

In particular, $c_{j,-0} = -c_{j,0}$. Therefore, $c_{j,0} = 0$. Now we prove that the coefficients $c_{j,k}$ are $2M$ periodic in k, that is $c_{j,2M+k} = c_{j,k}$ for any $k \in \mathbb{Z}$. Using

$t = y + 2h$, $M = 2^j$ and the $2h$ periodicity of f

$$c_{j,2M+k} = \int_{\mathbb{R}} f(y+2h)\frac{\sqrt{2}^j}{\sqrt{h}}\tilde{\varphi}\left(2^j\frac{y+2h}{h} - 2M - k\right)\,dy$$
$$= \int_{\mathbb{R}} f(y)\frac{\sqrt{2}^j}{\sqrt{h}}\tilde{\varphi}\left(2^j\frac{y}{h} - k\right)\,dy = c_{j,k}. \quad (4.16)$$

Thus,

$$c_{j,M-k} = -c_{j,-(M-k)} = -c_{j,-M+k} = -c_{j,2M+(-M+k)} = -c_{j,M+k}.$$

This equality with $k = 0$ also shows that $c_{j,M} = 0$. Therefore the coefficients $c_{j,k},\ k \in \mathbb{Z}$ form an IMP signal. □

Theorem 4.3.4. *Let $\mathbf{c}_j$ be an IMP signal of length $M = 2^j$. Let $h > 0$ and*

$$\varphi^{(h)}_{j,k}(t) = \frac{\sqrt{2}^j}{\sqrt{h}}\varphi^{(h)}(2^j t - kh), \quad j,k \in \mathbb{Z}, \quad t \in \mathbb{R}. \quad (4.17)$$

Then function

$$f_j(t) = \sum_{k\in\mathbb{Z}} c_{j,k}\varphi^{(h)}_{j,k}(t), \quad t \in \mathbb{R} \quad (4.18)$$

is an IMP function of length h.

Proof. Using $m = -k$ we obtain

$$f_j(-t) = \sum_{k\in\mathbb{Z}} c_{j,k}\varphi^{(h)}_{j,k}(-t) = \sum_{k\in\mathbb{Z}} c_{j,k}\frac{\sqrt{2}^j}{\sqrt{h}}\varphi\left(-2^j\frac{t}{h} - k\right)$$
$$= -\sum_{k\in\mathbb{Z}} c_{j,-k}\frac{\sqrt{2}^j}{\sqrt{h}}\varphi\left(2^j\frac{t}{h} + k\right) = -\sum_{m\in\mathbb{Z}} c_{j,m}\frac{\sqrt{2}^j}{\sqrt{h}}\varphi\left(2^j\frac{t}{h} - m\right)$$
$$= -\sum_{m\in\mathbb{Z}} c_{j,m}\varphi^{(h)}_{j,m}(t) = -f_j(t). \quad (4.19)$$

Also, using $M = 2^j$, $m = k - 2M$, and $c_{j,m+2M} = c_{j,m}$ we get

$$f_j(t+2h) = \sum_{k\in\mathbb{Z}} c_{j,k}\varphi_{j,k}^{(h)}(t+2h) = \sum_{k\in\mathbb{Z}} c_{j,k}\frac{\sqrt{2}^j}{\sqrt{h}}\varphi\left(2^j\frac{t+2h}{h} - k\right)$$
$$= \sum_{k\in\mathbb{Z}} c_{j,k}\frac{\sqrt{2}^j}{\sqrt{h}}\varphi\left(2^j\frac{t}{h} + 2M - k\right) = \sum_{m\in\mathbb{Z}} c_{j,m+2M}\frac{\sqrt{2}^j}{\sqrt{h}}\varphi\left(2^j\frac{t}{h} - m\right)$$
$$= \sum_{m\in\mathbb{Z}} c_{j,m}\frac{\sqrt{2}^j}{\sqrt{h}}\varphi\left(2^j\frac{t}{h} - m\right) = \sum_{m\in\mathbb{Z}} c_{j,m}\varphi_{j,m}^{(h)}(t) = f_j(t), \quad (4.20)$$

so $f_j(t+2h) = f_j(t)$ for any $t \in \mathbb{R}$. Thus, $f_j(h-t) = -f_j(-h+t) =$
$-f_j(h+t), \quad t \in \mathbb{R}$. Therefore f_j is an IMP function of length h. □

4.3.2 Wavelet decomposition of IMP functions

The wavelet decomposition of an IMP function f has a very special structure. Fix an approximation level N. It is shown in section 4.4 how to compute the coefficients of its frame $\mathbf{c}_N$. According to Theorem 4.3.3, the frame $\mathbf{c}_N$ of f is an IMP signal of length 2^N, the frame $\mathbf{c}_{N-1}$ is an IMP signal of length 2^{N-1}, and so on. According to (3.42) it takes N consecutive applications of the filters L_- and H_- to completely decompose $\mathbf{c}_N$, that is to reach the IMP signal $\mathbf{c}_0 = \overleftarrow{\{\,0\,\}}^{\rightarrow}$ of length 1. This signal is identically equal to zero.

The result of the decomposition of f is N detail signals $\mathbf{d}_j,\ j = 0, ..., N-1$. Since the mask $\tilde{\mathbf{b}}$ is not symmetric, the signals $\mathbf{d}_j$ are not IMP. However, $\mathbf{d}_j$ is a periodic signal of period 2^{j+1}. The decomposition process is reversed by N consecutive applications of the operation $\mathbf{c}_j = L_+\mathbf{c}_{j-1} + H_+\mathbf{d}_{j-1}$, which perfectly reconstructs (synthesizes) the frame $\mathbf{c}_N$ from the details.

4.4 Oversampling level and expansion coefficients

To use the DWT for an IMP function f of length h, we first have to compute the signal $\mathbf{c}_N$ for some frame f_N. Similarly to (3.28), the frame coefficients $c_{N,k}$ are defined by $c_{N,k} = \langle f, \tilde{\varphi}^{(h)}_{N,k} \rangle = \int_{\mathbb{R}} f(t) \tilde{\varphi}^{(h)}_{N,k}(t)\, dt$. The evaluation of integrals is a computationally expensive operation. However, we show in this section that a sample of the function f at a fine mesh on $[0, h]$ is approximately equal to the signal $\mathbf{c}_N$ (up to a constant factor). We call such a fine mesh the *oversampling mesh* for f.

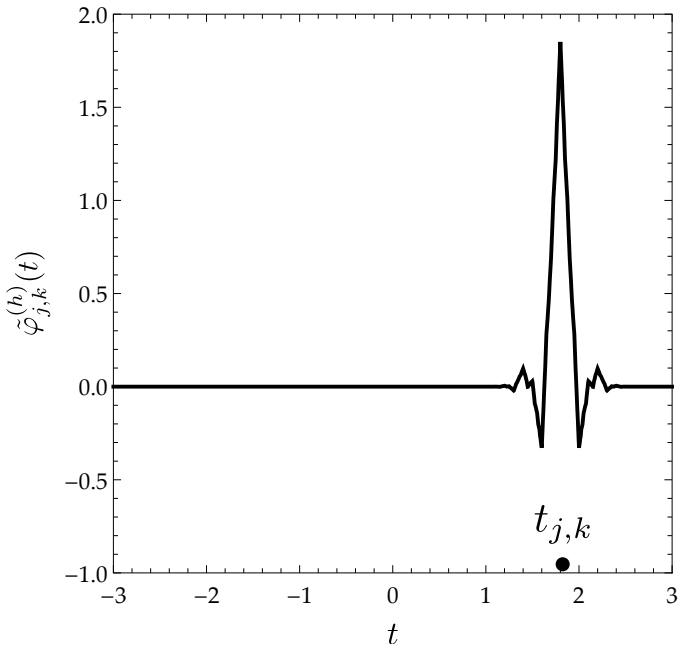


Figure 4.2: Dilated and translated dual scaling function $\tilde{\varphi}^{(h)}_{j,k}$ for the FBI (9–7) biorthogonal wavelet family

By (4.7), (4.8), and using $y = 2^j t - kh$ we get

$$\int_{\mathbb{R}} \tilde{\varphi}^{(h)}_{j,k}(t) dt = \frac{\sqrt{2}^j}{\sqrt{h}} \int_{\mathbb{R}} \tilde{\varphi}^{(h)}(2^j t - kh) dt = \frac{1}{\sqrt{h}\sqrt{2}^j} \int_{\mathbb{R}} \tilde{\varphi}^{(h)}(y) dy = \frac{\sqrt{h}}{\sqrt{2}^j}. \quad (4.21)$$

Since the mask $\tilde{\mathbf{a}}$ defining $\tilde{\varphi}$ is symmetric and of an odd length, then $\tilde{\varphi}$ is an even

function, see Theorem 4.3.1. The scaling function $\tilde{\varphi}$ has a bounded support in $\mathbb{R}$ which is symmetric about the point $t = 0$, [63, propositions 2.11, 2.13]. Therefore the support of $\tilde{\varphi}_{j,k}^{(h)}$ is centered at the point $t_{j,k}$ such that $2^j t_{j,k} - kh = 0$. That is, $t_{j,k} = \frac{kh}{2^j}$. This is illustrated in Figure 4.2. The length of the support of $\tilde{\varphi}_{j,k}^{(h)}$ shrinks to zero, as $j \to \infty$.

Let j be sufficiently large. If a function f is continuous, then it changes very little over the small support of $\tilde{\varphi}_{j,k}^{(h)}$. Thus $f(t) \approx f(t_{j,k})$ for any $t \in \mathit{supp}\ \tilde{\varphi}_{j,k}^{(h)}$. By (4.21)

$$\left\langle f, \tilde{\varphi}_{j,k}^{(h)} \right\rangle = \int_{\mathbb{R}} f(t)\tilde{\varphi}_{j,k}^{(h)}(t)dt \approx \int_{\mathbb{R}} f\left(\frac{kh}{2^j}\right)\tilde{\varphi}_{j,k}^{(h)}(t)dt = \frac{\sqrt{h}}{\sqrt{2}^j} f\left(\frac{kh}{2^j}\right). \quad (4.22)$$

The frame f_j is given by $f_j = \sum_{k\in\mathbb{Z}} c_{j,k}\varphi_{j,k}^{(h)}$. Since the scaling functions $\varphi^{(h)}, \tilde{\varphi}^{(h)}$ form a biorthogonal system, we conclude that $c_{j,k} = \langle f, \tilde{\varphi}_{j,k}^{(h)} \rangle$. Thus, according to (4.22), for a sufficiently large j

$$c_{j,k} = \langle f, \tilde{\varphi}_{j,k}^{(h)} \rangle \approx \frac{\sqrt{h}}{\sqrt{2}^j} f\left(\frac{kh}{2^j}\right), \quad k \in \mathbb{Z}. \quad (4.23)$$

Let us examine this conclusion more closely. The support of the dimensionless dual scaling function $\tilde{\varphi}$ is $[-\tilde{M}, \tilde{M}]$, where $\tilde{M}$ is defined by the dual mask $\tilde{\mathbf{a}}$, see (3.21). Accordingly, the support of $\tilde{\varphi}^{(h)}$ is $[-\tilde{M}h, \tilde{M}h]$, and

$$\mathit{supp}\ \tilde{\varphi}_{j,k}^{(h)} = \left[\frac{kh}{2^j} - \tilde{M}\frac{h}{2^j}, \frac{kh}{2^j} + \tilde{M}\frac{h}{2^j}\right]. \quad (4.24)$$

It is convenient to define $h_j = h/2^j$. The geometric meaning of h_j is the subinterval length (the *mesh resolution*) of the mesh on interval $[0, h]$ associated with

the dilation level j. Then the mesh is extended to $\mathbb{R}$. We have

$$supp\ \tilde{\varphi}_{j,k}^{(h)} = \left[(k-\tilde{M})h_j, (k+\tilde{M})h_j\right], \tag{4.25}$$

and (4.23) becomes

$$c_{j,k} \approx \sqrt{h_j} f(kh_j), \quad k \in \mathbb{Z}. \tag{4.26}$$

Therefore, for a sufficiently large j the computation of the expansion coefficients $c_{j,k}$ is just the sampling of the function f at the mesh corresponding to the level j.

Notice, that the approximation sign in (4.26) can be replaced by the equality sign if the function f is constant over the support of $\tilde{\varphi}_{j,k}^{(h)}$. Furthermore, by Theorem 4.3.1, $\tilde{\varphi}$ is an even function. Therefore, if f is a linear function, then the approximation in (4.22) is an equality. Thus equation (4.26) also becomes an equality.

We can therefore conclude the following. Let $N \in \mathbb{N}$ be so large that the derivative f' can be assumed to be nearly constant over any interval of length $2\tilde{M}h_N$, where $h_N = h/2^N$. Then function f is well approximated by linear functions over any such interval, and it is acceptable to compute the coefficients $c_{N,k}$ according to formula

$$c_{N,k} = \sqrt{h_N} f(kh_N), \quad k \in \mathbb{Z}. \tag{4.27}$$

Practically, the value for N is determined experimentally based on the application's performance.

4.5 Cascade Algorithm

The Cascade algorithm is used to compute values of scaling functions and wavelets at *diadic points*. Diadic points are points of the form $k/2^j,\ j,k \in \mathbb{Z}$. Combined with an interpolation this gives us a way to compute the scaling functions and wavelets for any $t \in \mathbb{R}$.

The idea of the Cascade algorithm is to run the unit signal $\mathbf{e} = (\dots, 0, 1, 0, \dots)$ through the inverse wavelet transform with all the details set equal to zero. Note that signal $\mathbf{e}$ is not an IMP signal. If the dual low-pass filter L_+ is applied to $\mathbf{e}$, then the result is the scaling function φ. If the dual high-pass filter H_+ is applied to $\mathbf{e}$, then the result is the wavelet ψ, [14, section 6.5].

The Cascade algorithm for the scaling function φ is the following

1. Define the unit signal $\mathbf{e}$ by $e_0 = 1$, and $e_k = 0$ for any $k \neq 0$. That is $\mathbf{e} = (\dots, 0, 1, 0, \dots)$.

2. Choose a sufficiently large $K \in \mathbb{N}$. Let $\mathbf{c} = \mathbf{e}$. Repeat the following computation K times: $\mathbf{c} \leftarrow L_+\mathbf{c}$.

3. Let $h_K = \dfrac{1}{2^K}$. According to (4.27),

$$\varphi\left(\frac{k}{2^K}\right) = \varphi(k\,h_K) = \frac{1}{\sqrt{h_K}}\,c_k, \quad k \in \mathbb{Z}. \tag{4.28}$$

The graphs of the scaling functions φ and $\tilde{\varphi}$ for the FBI (9–7) biorthogonal wavelet family are shown in Figures 3.4 and 3.5. They were obtained by using the Cascade algorithm.

Chapter 5

Wavelet-based smoothing of linear features

This chapter describes how to to smooth linear features using dimensional wavelets . Although our method can be applied to any multidimensional polyline of finite length, in our application we are focused on smoothing the centerline of a tributary.

Given a tributary T, our goal is to decompose it into a sequence of wavelet details. First, we transform it into IMP form. Then, the TributaryDecomposition algorithm obtains the wavelet details. At the synthesis stage, these details are used to construct smoothed versions T^ε of T that depend on an accuracy parameter ε.

Our main result, Theorem 5.3.4, asserts that T^ε and T have the same endpoints, and that T^ε approximates T continuously with respect to ε.

We also provide a detailed evaluation of the algorithm as it applies to various tributaries. We chose three tributaries of various length. For each tributary (T_1, T_2, and T_3) we smoothed them for three different accuracy levels and plotted

the results. We demonstrate that our method can smooth a variety of tributary shapes in a cartographically appropriate manner. In particular, tributary T_3 contains a complicated spiral-like feature that the algorithm smooths appropriately.

A quantitative measure of the accuracy of the approximation is the deviation of the smoothed tributary from the original. We show the deviation of tributary T_3 along a portion of its length. Our analysis of 5,322 tributaries shows a slight correlation between maximal deviation and tributary length.

5.1 Transformation of a tributary to the IMP form

Before we can decompose T into a sequence of wavelet details, we need to represent T in Inverse Mirror Periodic (IMP) form. The first step is to parametrize the tributary by its length. To accomplish this, we use the Great-circle distance formula to compute the distance between points on the tributary. A more precise method can also be used.

5.1.1 Great-circle distance formula

To compute the distance between two points on the surface of a sphere, we use the Great-circle distance formula. Let (λ_s, ϕ_s) and (λ_f, ϕ_f) be the longitude and latitude of a start point and final point, respectively. Let their differences be $\Delta\lambda$ and $\Delta\phi$. Let $\Delta\sigma$ be the (spherical) angular difference, or central angle. It can be computed from the haversine formula:

$$\Delta\sigma = 2\arcsin\left(\sqrt{\sin^2\frac{\Delta\phi}{2} + \cos\phi_s\cos\phi_f\sin^2\frac{\Delta\lambda}{2}}\right).$$

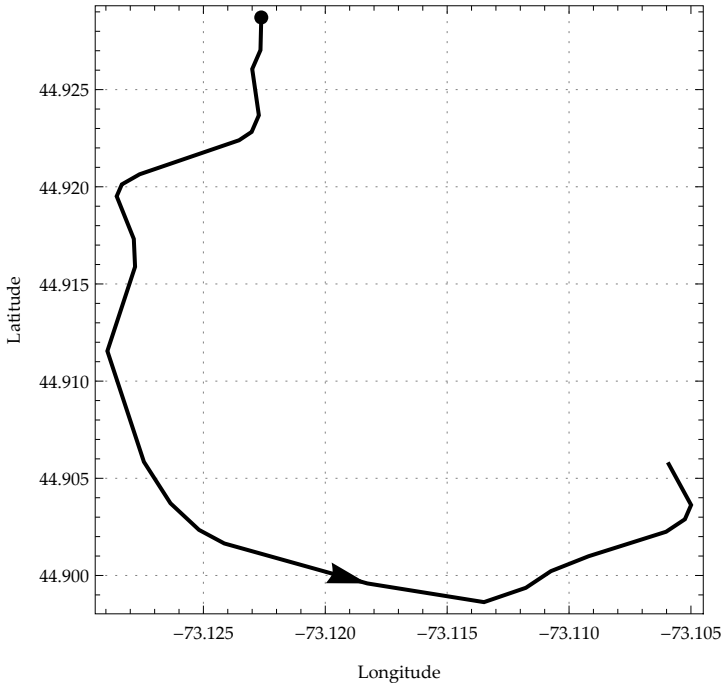


Figure 5.1: Tributary T

For $\Delta\sigma$ given in radians, the arc length distance d between these two points is

$$d = R\Delta\sigma,$$

where R is the radius of the sphere. The average radius for a spherical approximation of Earth is $6,371.01$ km.

5.1.2 Tributary parametrization

Let $T = \{p_0, p_1, \ldots, p_M\}$ be a tributary with a flow direction, where $p_i = (x_i,\, y_i)$ are the vertices of the polyline, see Figure 5.1. Here x_i, y_i are the longitude and latitude coordinates of the vertex p_i. Assume that the vertices are arranged in the same order as the direction of the flow in T. Next, we parametrize T by its length.

First we compute the length l_i of each segment $\overline{p_{i-1}p_i}$, for $i = 1, \ldots, M$ using the Great-circle distance formula. Let $l_0 = 0$. Then the total length of the tributary is

$$\gamma_T = \sum_{i=0}^{M} l_i.$$

Let t_i be the distance between p_0 and p_i along the polyline, that is

$$t_i = \sum_{k=0}^{i} l_k.$$

This way, each vertex p_i is associated with its parameter value $t_i,\ 0 \leq t_i \leq \gamma_T$. Therefore, the positions of the vertices are given by

$$\begin{aligned} p_0 &= \big(x(0),\, y(0)\big), \\ p_1 &= \big(x(t_1),\, y(t_1)\big), \\ &\vdots \\ p_M &= \big(x(\gamma_T),\, y(\gamma_T)\big). \end{aligned}$$

Now our goal is to define the point on the polyline that corresponds to any particular value of t, where $0 \leq t < \gamma_T$. Given t, we find t_i such that $t_i \leq t < t_{i+1}$. Define $x(t)$ and $y(t)$ by

$$x(t) = x(t_i) + \frac{t - t_i}{t_{i+1} - t_i}\Big(x(t_{i+1}) - x(t_i)\Big), \quad y(t) = y(t_i) + \frac{t - t_i}{t_{i+1} - t_i}\Big(y(t_{i+1}) - y(t_i)\Big).$$

In other words, the point $\big(x(t),\, y(t)\big)$ is on the segment $\overline{p_i p_{i+1}}$, in a position proportional to its distance from p_i.

The result of this procedure is the parametrization $T(t) = \big(x(t),\, y(t)\big),\ 0 \leq t \leq \gamma_T$, where γ_T is the length of T. If point $p \in T$, and t is the distance from p_0

to p along the polyline, then $T(t) = p$, In particular, if t_m is the distance along the polyline between the vertices p_0 and p_m, then $T(t_m) = p_m$.

5.1.3 Transformation to the IMP form

Now that we have obtained a parametrization of tributary T as $\big(x(t),\, y(t)\big),\ 0 \leq t \leq \gamma_T$, we can continue with its transformation to the IMP form. Since both the longitude component $x(t)$ and the latitude component $y(t)$ are treated in the same way, we will just discuss $x(t)$.

The slope and the x-intercept of the *baseline* (line connecting the endpoints of $x(t)$) are computed by

$$a_x = \frac{1}{\gamma_T}\Big(x(\gamma_T) - x(0)\Big), \quad b_x = x(0).$$

Next, the baseline is subtracted from the $x(t)$, giving $x^*(t)$ defined by

$$x^*(t) = x(t) - a_x t - b_x, \quad 0 \leq t \leq \gamma_T.$$

Then $x^*(0) = x^*(\gamma_T) = 0$. Figures 5.2 and 5.3 show the x and y components of T together with their baselines. Figures 5.4 and 5.5 show the transformed components $x^*(t)$ and $y^*(t)$.

Geometrically, this means that the transformed polyline $\big(x^*(t), y^*(t)\big)$, $0 \leq t \leq \gamma_T$ is a closed loop with the start and the end points at the origin as shown in Figure 5.6.

Now we extend $x^*(t)$ to the values of the parameter t in the interval $[-\gamma_T, 0]$

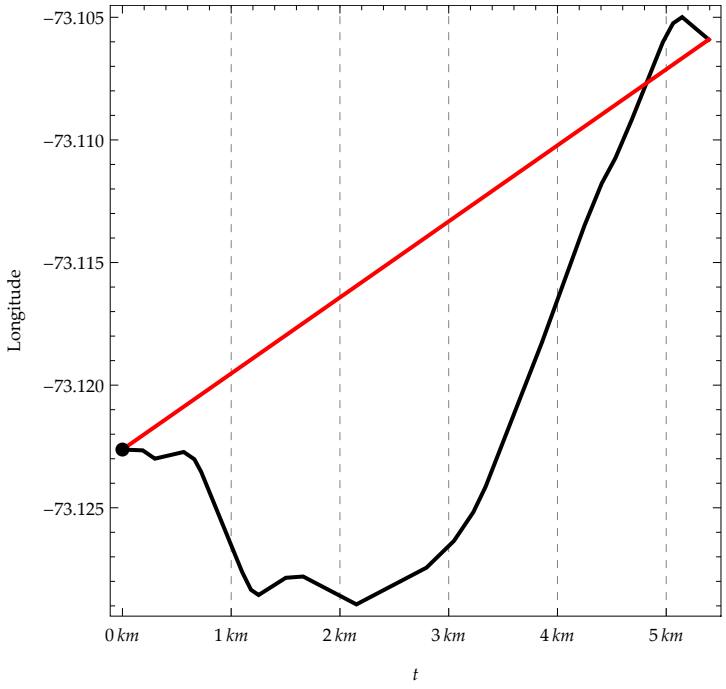


Figure 5.2: The longitude component x of T with its baseline

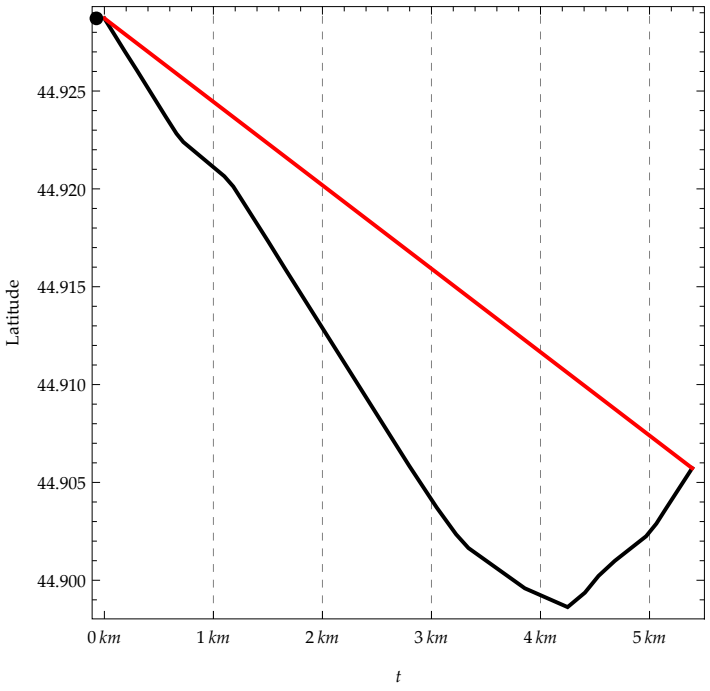


Figure 5.3: The latitude component y of T with its baseline

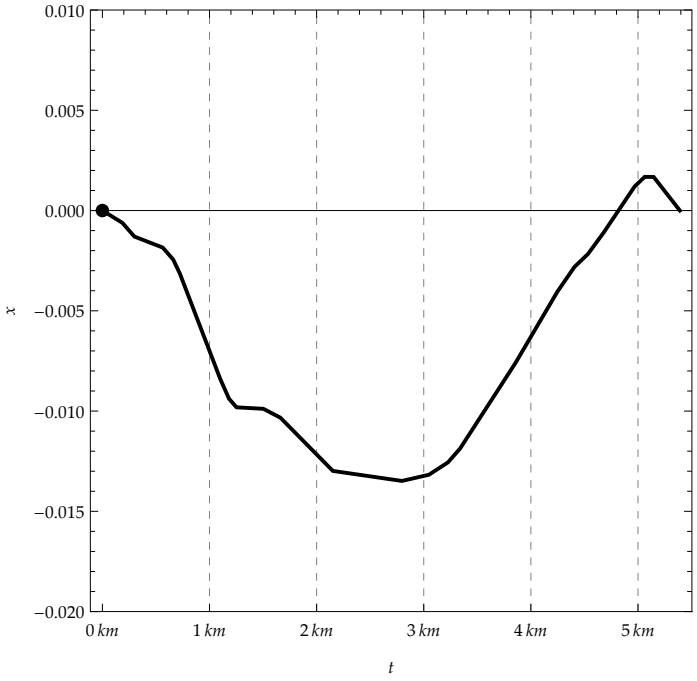


Figure 5.4: Function $x^*(t)$

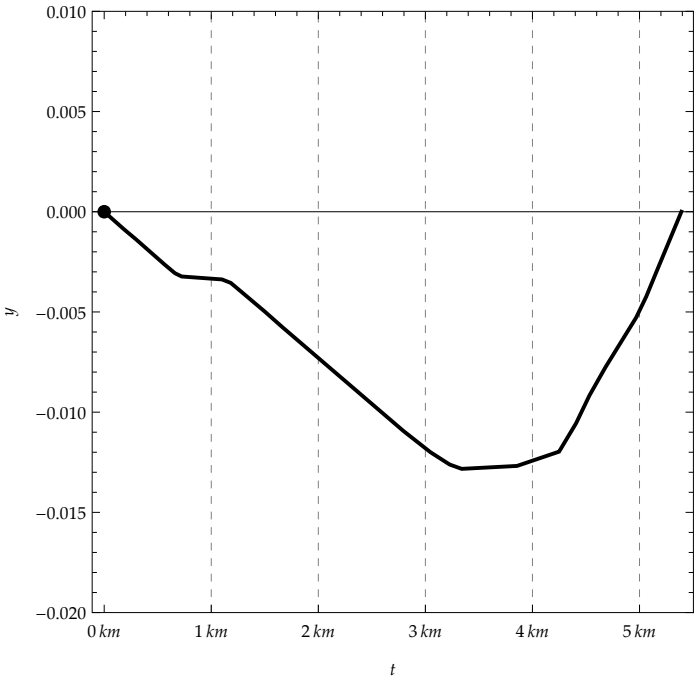


Figure 5.5: Function $y^*(t)$

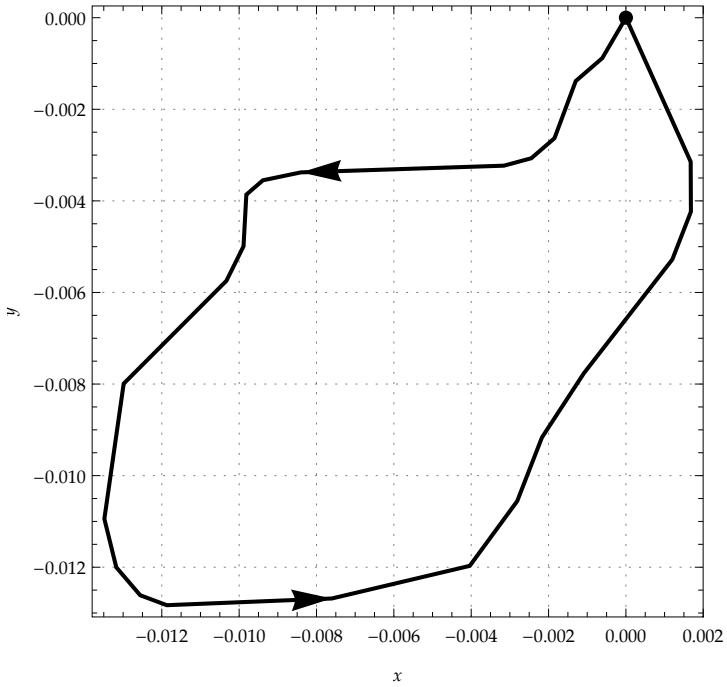


Figure 5.6: Transformed polyline $\big(x^*(t), y^*(t)\big),\ 0 \le t \le \gamma_T$

by inverse mirroring. This means that

$$x^*(t) = -x^*(-t), \quad -\gamma_T \le t \le 0.$$

Finally, extend $x^*(t)$ periodically with the period $2\gamma_T$ for all $t \in \mathbb{R}$. The transformed polyline $\big(x^*(t), y^*(t)\big),\ t \in \mathbb{R}$ is now extended to become a figure-8 shaped loop, see Figure 5.7. Figure 4.1 shows IMP-extended $x^*(t)$ by itself.

5.2 Wavelet decomposition of a tributary

After the tributary T is represented in IMP form $\big(x^*(t), y^*(t)\big),\ t \in \mathbb{R}$, our next step is to determine its wavelet details. First, we determine the oversampling mesh for T. Then we sample x^* at the oversampling mesh to obtain the wavelet frame coefficients at the oversampling level.

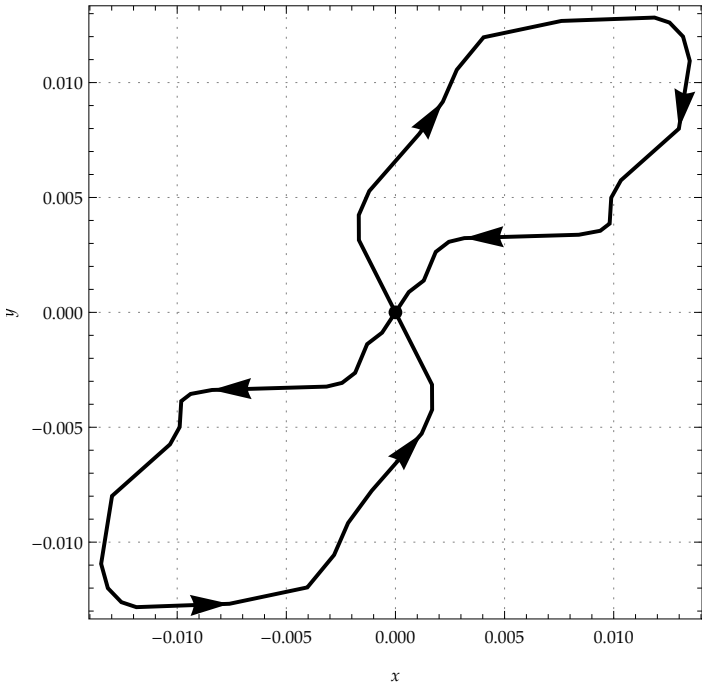


Figure 5.7: Transformed polyline $\big(x^*(t), y^*(t)\big),\ t \in \mathbb{R}$ after extension

5.2.1 Oversampling

The representative part of the IMP function x^* is defined on the interval $[0, \gamma_T]$. The wavelet framework (section 4.4) requires that the interval be divided into 2^N subintervals. Furthermore, we require that these subintervals should satisfy the inequality $\delta/2 \leq h_N < \delta$, where $\delta > 0$ is the bound on the mesh resolution. This is needed to ensure that all tributaries are smoothed similarly. In our system, the value of δ is determined experimentally.

We define the *mesh resolution* h_N as the distance between sampling points. The *mesh size* 2^N is the number of sampling intervals. The number N and the mesh resolution h_N are determined by

$$N = \left\lfloor \log_2 \left(\frac{\gamma_T}{\delta} \right) \right\rfloor + 1, \quad \text{and} \quad h_N = \frac{\gamma_T}{2^N}. \tag{5.1}$$

Note that the mesh size and mesh resolution depend on the length γ_T of the tributary.

To obtain the wavelet frame signal $\mathbf{c}_N$ for x^* at the oversampling level N, we sample x^* at the mesh points $kh_N,\ k = 0, 1, \ldots, 2^N - 1$. This means that we let

$$\mathbf{c}_N = \overleftarrow{\{}\, c_{N,0}, c_{N,1}, \ldots, c_{N,2^N-1} \,\overrightarrow{\}},$$

where $c_{N,k} = \sqrt{h_N}\ x^*(kh_N),\ k = 0, 1, \ldots, 2^N - 1$; see formula (4.27).

5.2.2 Wavelets details of T

According to section 4.3.2, the wavelet frame signal $\mathbf{c}_N$ can be decomposed into N detail signals $\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}$. To find them, we perform N consecutive applications of the filters L_- and H_-, by letting j decrease from $j = N$ to $j = 1$

$$\mathbf{c}_{j-1} = L_-\mathbf{c}_j \quad \text{and} \quad \mathbf{d}_{j-1} = H_-\mathbf{c}_j.$$

We retain the coefficients $d_{j-1,k}$ with $-\tilde{M} \le k \le 2^{j-1} + \tilde{M}$ in $\mathbf{d}_{j-1}$ using $\tilde{M}$ from the dual mask $\tilde{a}$; see (3.21), (3.24). At the same time, we compute the magnitudes

$$d_{j-1}^2 = \sum_k |d_{j-1,k}|^2,$$

in which the sum is over the retained coefficients in $\mathbf{d}_{j-1}$.

5.2.3 Wavelet details thresholding

We would like to discard details that have a small effect on the tributary approximation. Let $\varepsilon_0 \ge 0$ be the initial accuracy. According to the theory of

biorthogonal wavelets, we can estimate the $L^2(\mathbb{R})$ norm of the difference between two frames in terms of the detail coefficients, see (3.32).

We discard details $\mathbf{d}_j$ that do not affect the initial accuracy by finding the smallest integer J_x such that $0 \leq J_x < N-1$ and

$$\sum_{j=J_x+1}^{N-1} d_j^2 \leq \gamma_T \varepsilon_0^2. \tag{5.2}$$

We normalize the error estimate (5.2) by the length of the tributary γ_T so that the approximations are comparable for different length tributaries. An analysis of this normalization is given in section 5.4. Details satisfying (5.2) can be discarded since their influence on the reconstruction of x is within the specified accuracy ε_0. In other words, keep the details $\mathbf{d}_j$ for $0 \leq j \leq J_x$ and discard the rest.

5.2.4 Summary of the decomposition stage

The complete description of the decomposition stage is given by the TRIBUTARY-DECOMPOSITION algorithm.

Input: Parametric representation $T(t) = \big(x(t),\, y(t)\big),\ 0 \leq t \leq \gamma_T$ of the tributary, oversampling mesh bound $\delta > 0$, and initial accuracy $\varepsilon_0 \geq 0$.

Output: Detail signals $\mathbf{d}_j^x,\ j = 0, 1, \ldots J_x$, and $\mathbf{d}_j^y,\ j = 0, 1, \ldots J_y$ of finite length for the x and y components of the tributary T; and parameters $a_x, b_x, a_y, b_y, \gamma_T, J_x, J_y$.

1 Compute the oversampling mesh level N and its mesh resolution h_N by (5.1)
2 Transform the x component of T into an IMP function x^* (section 5.1.3)
3 Retain the slope a_x, and the x-intercept b_x
4 Find the wavelet frame signal $\mathbf{c}_N$ (section 5.2.1)
5 Find the details signals $\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}$ (section 5.2.2)
6 Find the magnitudes $d_0^2, \ldots, d_{N-1}^2$
7 Retain details $\mathbf{d}_0, \ldots, \mathbf{d}_{J_x}$ that affect the initial accuracy ε_0 (section 5.2.3)
8 Repeat lines 2–7 for the y component of T

Algorithm 5.1: The TRIBUTARYDECOMPOSITION algorithm

5.3 Tributary synthesis

The goal of the Tributary synthesis stage is to construct a smoothed tributary T^ε that has the same endpoints as T and is continuously dependent on the accuracy ε. This is achieved by finding the frames $\mathbf{c}_J$ and $\mathbf{c}_{J+1}$ such that $\mathbf{c}_J$ *does not* satisfy the accuracy requirement, and $\mathbf{c}_{J+1}$ *does* satisfy it. The smoothed tributary T^ε is constructed from the frame $\hat{\mathbf{c}}$ that interpolates the frames $\mathbf{c}_J$ and $\mathbf{c}_{J+1}$.

5.3.1 Finding level J

Let the accuracy parameter ε satisfy $\varepsilon \geq \varepsilon_0$. We would like to disregard the details $\mathbf{d}_j$ for the x component of T that do not affect the required accuracy ε.

We find the smallest integer J such that $0 \leq J < J_x$ and

$$\sum_{j=J+1}^{J_x} d_j^2 \leq \gamma_T(\varepsilon^2 - \varepsilon_0^2) < \sum_{j=J}^{J_x} d_j^2$$

is satisfied. This implies that $d_J^2 > 0$.

However, if $d_{J_x}^2 > \gamma_T(\varepsilon^2 - \varepsilon_0^2)$, then let $J = J_x$. In other words, the finest level details are already non-negligible.

One other possibility is if $\sum_{j=0}^{J_x} d_j^2 \leq \gamma_T(\varepsilon^2 - \varepsilon_0^2)$. Then we let $x^\varepsilon(t) = a_x t + b_x$, $0 \leq t \leq \gamma_T$. In this case, all the details are negligible.

5.3.2 Interpolated frame $\hat{\mathbf{c}}$

Now we use the DWT procedure to construct the output frame and the smoothed x component of T. To find $\mathbf{c}_{J-1}$ we perform $J-1$ consecutive applications of the filters L_+ and H_+, by letting j increase from $j = 1$ to $j = J-1$, where

$\mathbf{c}_j = L_+\mathbf{c}_{j-1} + H_+\mathbf{d}_{j-1}$ and $\mathbf{c}_0 = \overleftarrow{\{}\, 0 \,\overrightarrow{\}}$. The result is the frame signal $\mathbf{c}_{J-1}$.

Now we define the interpolated frame $\hat{\mathbf{c}}$ by

$$\hat{\mathbf{c}} = L_+\mathbf{c}_{J-1} + \frac{1}{d_J^2}\left(\sum_{j=J}^{J_x} d_j^2 - \gamma_T(\varepsilon^2 - \varepsilon_0^2)\right) H_+\mathbf{d}_{J-1}. \tag{5.3}$$

The smoothed component $x^{*,\varepsilon}$ of the tributary T is given by

$$x^{*,\varepsilon}(t) = \sum_k \hat{c}_k \varphi_{J,k}^{(h)}(t), \quad 0 \le t \le \gamma_T, \quad h = \gamma_T, \tag{5.4}$$

where k is such that $t \in supp\, \varphi_{J,k}^{(h)}$.

Finally we reconstruct x^ε by

$$x^\varepsilon(t) = x^{*,\varepsilon}(t) + a_x\, t + b_x, \quad 0 \le t \le \gamma_T.$$

5.3.3 Summary of the synthesis stage

The formal description of the synthesis stage is

Input: The output of the TributaryDecomposition algorithm and accuracy parameter $\varepsilon \ge \varepsilon_0$.
Output: Smoothed tributary $T^\varepsilon = \left(x^\varepsilon,\, y^\varepsilon\right)$

1. Find level J for the x component of T (section 5.3.1)
2. Find interpolated frame $\hat{\mathbf{c}}$ and the smoothed component x^ε of T (section 5.3.2)
3. Find level J for the y component of T (section 5.3.1)
4. Find interpolated frame $\hat{\mathbf{c}}$ and the smoothed component y^ε of T (section 5.3.2)

Algorithm 5.2: The TributarySynthesis algorithm

5.3.4 Preservation of endpoints and continuous smoothing of tributaries

In this section we present our main result, Theorem 5.3.4 proves that tributary T and its smoothing T^ε have the same endpoints. Furthermore, smoothed tributaries T^ε change continuously with respect to the accuracy parameter ε.

Definition 5.3.1. The *distance* $d(P, Q)$ between two points P and Q on Earth can be found by the Great-circle distance formula (section 5.1.1).

Definition 5.3.2. Let T and $\hat{T}$ be two tributaries defined over the same interval $0 \le t \le \gamma$. The *deviation* between them at $t \in [0, \gamma]$ is defined by $d\big(T(t), \hat{T}(t)\big)$.

Definition 5.3.3. The *maximal deviation* E between two tributaries of length γ is defined by

$$E(T, \hat{T}) = \max_{0 \le t \le \gamma} d(T(t), \hat{T}(t)).$$

Theorem 5.3.4. *Let T be a tributary. Suppose that $\varepsilon \ge \varepsilon_0$, and the smoothed tributary $T^\varepsilon = \big(x^\varepsilon,\, y^\varepsilon\big)$ has been obtained using the* TributarySynthesis algorithm. Then

1. The tributaries T and T^ε have the same endpoints.

2. The smoothed tributaries T^ε are continuously dependent on ε. That is

$$E(T^\tau, T^\varepsilon) = \max_{0 \le t \le \gamma_T} d(T^\tau(t), T^\varepsilon(t)) \to 0, \quad \text{as} \quad \tau \to \varepsilon.$$

Proof. Let $T = (x, y)$ and $T^\varepsilon = (x^\varepsilon, y^\varepsilon)$. According to the Tributary decomposition and synthesis algorithms we have

$$x(t) = x^*(t) + a_x t + b_x, \quad y(t) = y^*(t) + a_y t + b_y, \quad 0 \le t \le \gamma_T, \tag{5.5}$$

and

$$x^{\varepsilon}(t) = x^{*,\varepsilon}(t) + a_x t + b_x, \quad y^{\varepsilon}(t) = y^{*,\varepsilon}(t) + a_y t + b_y, \quad 0 \le t \le \gamma_T, \tag{5.6}$$

where x^* and y^* are IMP functions that are defined for any $t \in \mathbb{R}$.

Part I. By Theorem 4.3.3 the frame $\mathbf{c}_j$ of x^* is an IMP signal of length 2^j. The same Theorem shows also that $\mathbf{c}_{j-1}$ is an IMP signal of length 2^{j-1}. By Theorem 4.3.2 signal $L_+\mathbf{c}_{j-1}$ is IMP of length 2^j. According to DWT $\mathbf{c}_j = L_+\mathbf{c}_{j-1} + H_+\mathbf{d}_{j-1}$, section 3.4. Therefore the signal $H_+\mathbf{d}_{j-1} = \mathbf{c}_j - L_+\mathbf{c}_{j-1}$ is an IMP signal of length 2^j. Note that $\mathbf{d}_{j-1}$ is not an IMP signal itself.

Therefore $\hat{\mathbf{c}} = L_+\mathbf{c}_{j-1} + \alpha H_+\mathbf{d}_{j-1}$ is an IMP signal of length 2^j for any $\alpha \in \mathbb{R}$. In particular, the frame $\hat{\mathbf{c}}$ constructed in section 5.3.2 is IMP of length 2^J. Now it follows from Theorem 4.3.4 that $x^{*,\varepsilon}$ is an IMP function of length $h = \gamma_T$. Since any IMP function of length γ_T is equal to zero at $t = 0$ and at $t = \gamma_T$, we conclude that $x^{*,\varepsilon}(0) = x^{*,\varepsilon}(\gamma_T) = 0$. By (5.6) $x^{\varepsilon}(0) = x^{*,\varepsilon}(0) + b_x = x(0)$, since $b_x = x(0)$. Similarly $y^{\varepsilon}(0) = y(0)$. Thus $T^{\varepsilon}(0) = T(0)$.

Concerning the other endpoint of T, we have $x^{\varepsilon}(\gamma_T) = x^{*,\varepsilon}(\gamma_T) + a_x\gamma_T + b_x = x(\gamma_T)$, since $a_x = (x(\gamma_T) - x(0))/\gamma_T$. Similarly $y^{\varepsilon}(\gamma_T) = y(\gamma_T)$, and $T^{\varepsilon}(\gamma_T) = T(\gamma_T)$. In conclusion, $T(0) = T^{\varepsilon}(0)$, and $T(\gamma_T) = T^{\varepsilon}(\gamma_T)$, which proves the first part of the Theorem.

Part II. To prove the continuity of T^{ε} with respect to ε, it is enough to establish that the components of the smoothed tributaries satisfy $x^{\tau} \to x^{\varepsilon}$ and $y^{\tau} \to y^{\varepsilon}$ as $\tau \to \varepsilon$. More precisely, let

$$\|x^{\tau} - x^{\varepsilon}\|_{\infty} = \max_{0 \le t \le \gamma_T} |x^{\tau}(t) - x^{\varepsilon}(t)|.$$

Then we want to show that $\|x^\tau - x^\varepsilon\|_\infty \to 0$, as $\tau \to \varepsilon$. Convergence for the y-components is done similarly, so we continue only with the x-components of the tributaries.

Our task is further simplified by using (5.6), which gives

$$\|x^\tau - x^\varepsilon\|_\infty = \|x^{*,\tau} - x^{*,\varepsilon}\|_\infty.$$

So Part II of the Theorem would be established if we are able to show that $\|x^{*,\tau} - x^{*,\varepsilon}\|_\infty \to 0$, as $\tau \to \varepsilon$.

We continue the proof by considering various cases appearing in the TributarySynthesis algorithm, which can happen for different choices of $\varepsilon \geq \varepsilon_0$. First, suppose that

$$\sum_{j=J+1}^{J_x} d_j^2 < \gamma_T(\varepsilon^2 - \varepsilon_0^2) < \sum_{j=J}^{J_x} d_j^2$$

for some J such that $0 \leq J < J_x$. Then the above inequality is satisfied for any τ sufficiently close to ε. By (5.3) the corresponding frames $\hat{\mathbf{c}}^\tau$ and $\hat{\mathbf{c}}^\varepsilon$ satisfy

$$\hat{\mathbf{c}}^\tau - \hat{\mathbf{c}}^\varepsilon = \frac{\gamma_T}{d_J^2}\big(-\tau^2 + \varepsilon^2\big) H_+ \mathbf{d}_{J-1}.$$

Therefore

$$|\hat{c}_k^\tau - \hat{c}_k^\varepsilon| = \frac{\gamma_T}{d_J^2}|\varepsilon^2 - \tau^2||(H_+\mathbf{d}_{J-1})_k|. \tag{5.7}$$

By (5.4)

$$|x^{*,\tau}(t) - x^{*,\varepsilon}(t)| \leq \sum_k |\hat{c}_k^\tau - \hat{c}_k^\varepsilon||\varphi_{J,k}^{(h)}(t)|, \quad 0 \leq t \leq \gamma_T,$$

where k is such that $t \in supp\, \varphi_{J,k}^{(h)}$. For any $t \in \mathbb{R}$ there are at most $2M$ indices

$k \in \mathbb{Z}$ such that $t \in supp\, \varphi^{(h)}_{J,k}$, where M is defined by the primary mask $\mathbf{a}$. Thus we can estimate $\|x^{*,\tau} - x^{*,\varepsilon}\|_\infty \le C|\tau - \varepsilon|$ for some positive constant C which is independent of $t \in \mathbb{R}$. Since $\|x^\tau - x^\varepsilon\|_\infty = \|x^{*,\tau} - x^{*,\varepsilon}\|_\infty$ the continuity of x^ε with respect to ε follows.

Now consider the case $\sum_{j=J+1}^{J_x} d_j^2 = \gamma_T(\varepsilon^2 - \varepsilon_0^2)$. Then $\hat{\mathbf{c}}^\varepsilon = L_+\mathbf{c}_{J-1} + H_+\mathbf{d}_{J-1}$, which means that $\hat{\mathbf{c}}^\varepsilon = \mathbf{c}_J$. If $\tau \to \varepsilon+$, i.e. $\tau > \varepsilon$, then the above argument is applicable with no change. If $\tau \to \varepsilon-$, then (discounting trivial cases)

$$\hat{\mathbf{c}}^\tau = L_+\mathbf{c}_J + \frac{1}{d_{J+1}^2}\left(\sum_{j=J+1}^{J_x} d_j^2 - \gamma_T(\tau^2 - \varepsilon_0^2)\right) H_+\mathbf{d}_J = L_+\mathbf{c}_J + \frac{\gamma_T}{d_{J+1}^2}\big(\varepsilon^2 - \tau^2\big) H_+\mathbf{d}_J. \tag{5.8}$$

Therefore

$$x^{*,\tau}(t) = \sum_m (L_+\mathbf{c}_J)_m \varphi^{(h)}_{J+1,m}(t) + \frac{\gamma_T}{d_{J+1}^2}\big(\varepsilon^2 - \tau^2\big) \sum_m (H_+\mathbf{d}_J)_m \varphi^{(h)}_{J+1,m}(t)$$

for any $0 \le t \le \gamma_T$. Observe that we have the following equivalent representations for x^*_J

$$x^{*,\varepsilon}(t) = x^*_J(t) = \sum_k c_{J,k}\varphi^{(h)}_{J,k}(t) = \sum_m (L_+\mathbf{c}_J)_m \varphi^{(h)}_{J+1,m}(t), \quad 0 \le t \le \gamma_T.$$

Therefore

$$x^{*,\varepsilon}(t) - x^{*,\tau}(t) = \frac{\gamma_T}{d_{J+1}^2}\big(\varepsilon^2 - \tau^2\big) \sum_m (H_+\mathbf{d}_J)_m \varphi^{(h)}_{J+1,m}(t), \quad 0 \le t \le \gamma_T,$$

and the continuity follows. Other cases can be treated similarly. □

Comment. To plot the smoothed tributary according to (5.4) we have to be able to evaluate the scaling function $\varphi(t)$, $t \in \mathbb{R}$. This is accomplished by using

the Cascade algorithm with interpolation (section 4.5).

5.4 Tributary smoothing analysis

Here we present an evaluation of how our system smooths tributaries. An implementation of our system is described in Chapter 7. The goal of this analysis is to confirm that our method preserves tributary endpoints, smooths continuously with respect to the accuracy parameter, and produces cartographically acceptable generalizations.

We have selected three tributaries from the Vermont river system to use in our evaluation. These tributaries vary in length and sinuosity. They can be found in Table 5.1. This table shows the start and end positions (longitude, latitude) of the tributaries, their length γ_T, the oversampling level N, and the mesh resolution h_N.

The system's parameters were set with an initial accuracy $\varepsilon_0 = 0$, and an oversampling mesh bound $\delta = 50$ meters. This means that the mesh resolution h_N will always be between 25 meters and 50 meters. For example, T_1 has a mesh size of $2^N = 2^6 = 64$ sampling intervals, and a mesh resolution $h_N = \gamma_T/2^N = 1.809/64 = 0.02827$ km $= 28.27$ meters.

Table 5.1: Sample tributaries T_1, T_2, T_3

	Start	End	γ_T	N	h_N
T_1	(-73.5041°, 43.9720°)	(-73.5035°, 43.9581°)	1.809 km	6	28.27 m
T_2	(-72.4709°, 43.5426°)	(-72.3906°, 43.5266°)	10.537 km	8	41.16 m
T_3	(-73.4022°, 43.5699°)	(-73.3298°, 44.5313°)	142.526 km	12	34.80 m

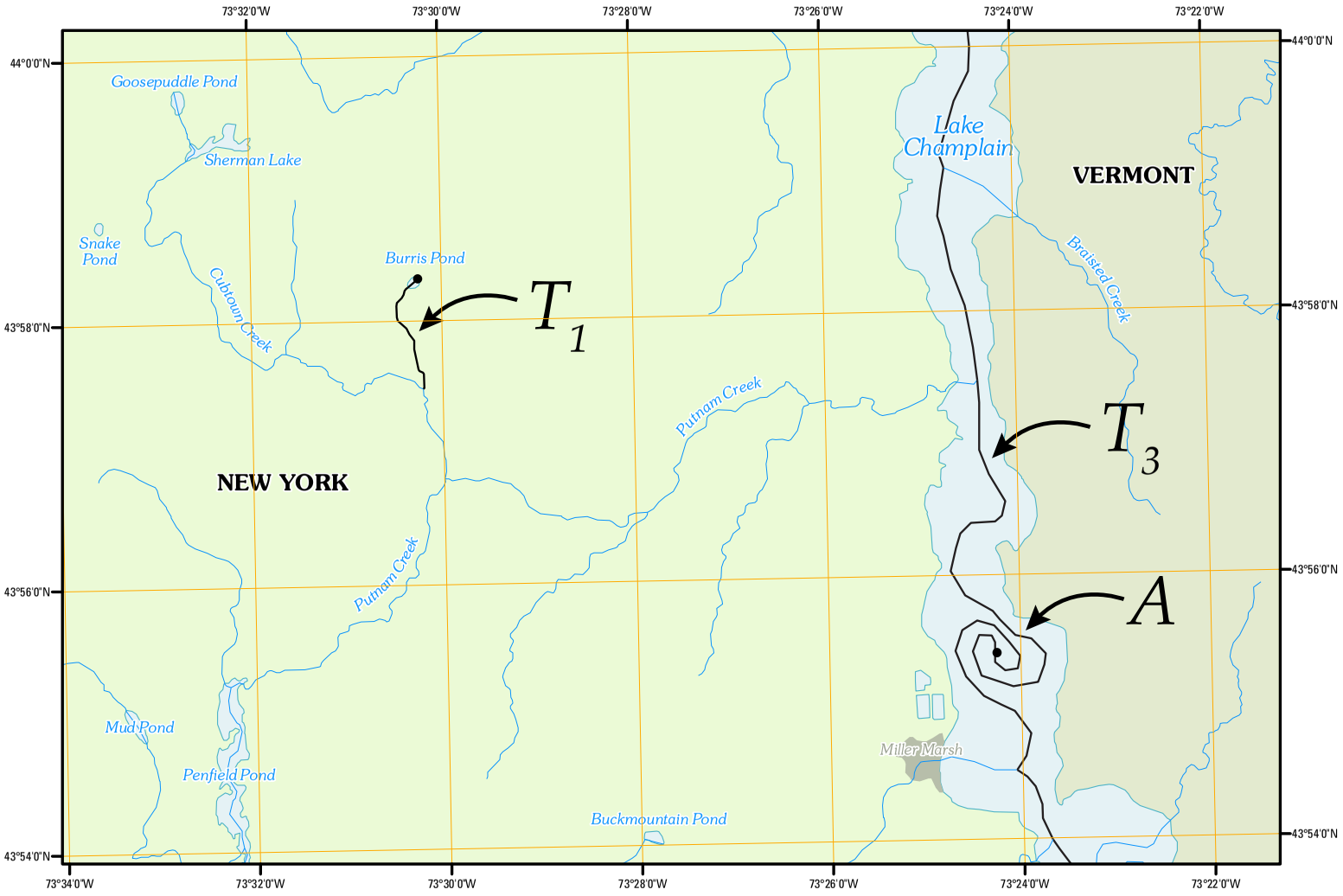


Figure 5.8: Tributary T_1 and a portion of tributary T_3 showing a spiral feature A

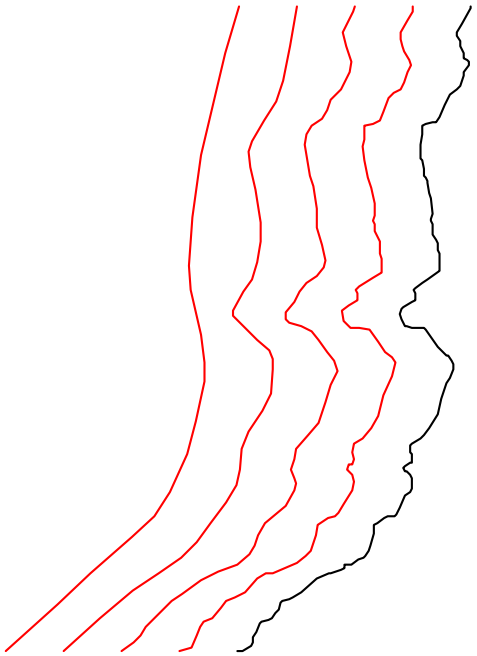


Figure 5.9: Illustration of progressive smoothing of a tributary

Figure 5.8 shows tributary T_1 in New York near *Burris Pond* and a portion of tributary (flowline) T_3 in *Lake Champlain*. The spiral feature A in T_3 is located in *Lapham Bay* at $(-73.4041°, 43.9235°)$. It is located 64.65 km from the start of the tributary T_3. This tributary flows from the south to the north.

To obtain optimal smoothing results by the wavelet method, it is advantageous to have the original tributary to be smooth. This means that we require x and y of T to be twice continuously differentiable functions. We accomplish this by computing a natural spline that goes through the vertices of the polyline. The spline interpolation procedure is described in section 7.4.2. Such a spline provides a more natural representation of the tributary.

Figure 5.9 shows a progression of smoothing for a tributary. The original tributary is rightmost in the figure. The smoothed curves correspond to decreasing values of the accuracy parameter ε (from left to right).

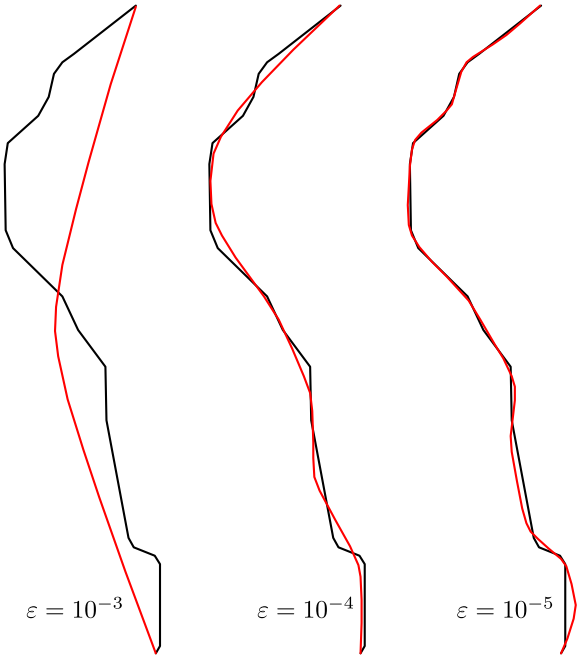


Figure 5.10: Tributary T_1 and its smoothing T_1^ε

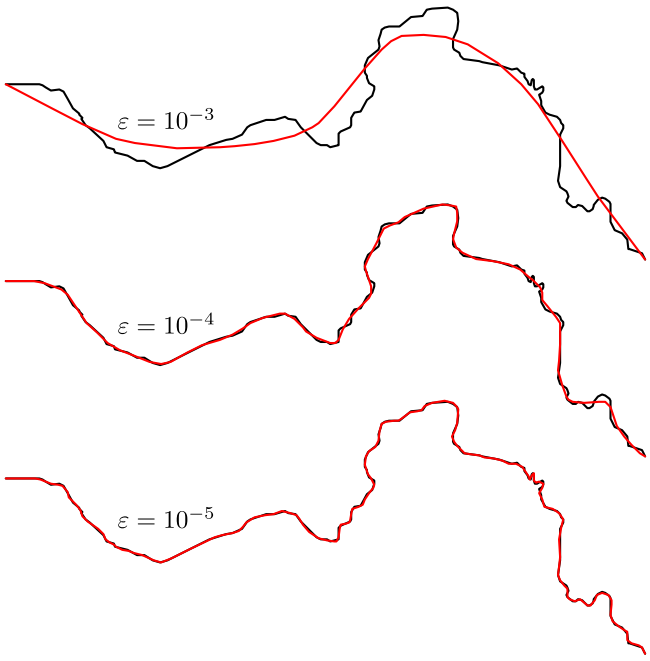


Figure 5.11: Tributary T_2 and its smoothing T_2^ε

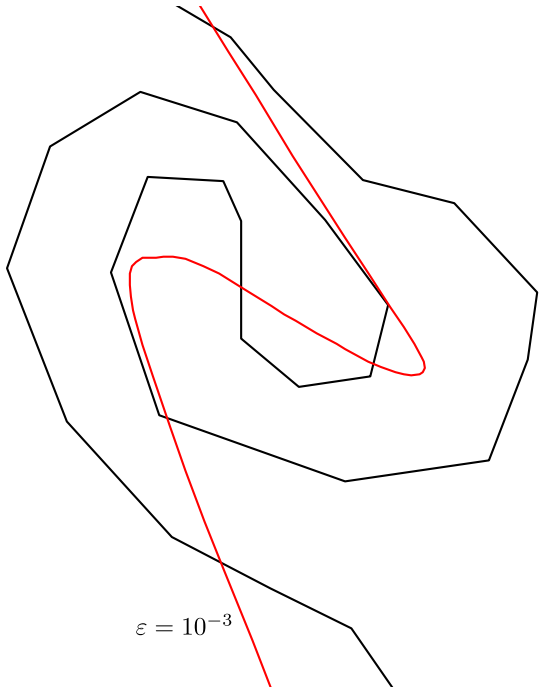


Figure 5.12: A portion of tributary T_3 and T_3^{ε} at the spiral feature A

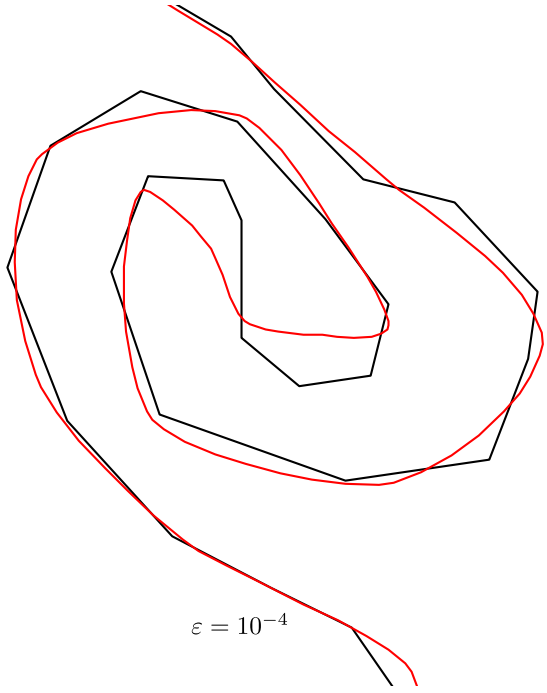


Figure 5.13: A portion of tributary T_3 and T_3^{ε} at the spiral feature A

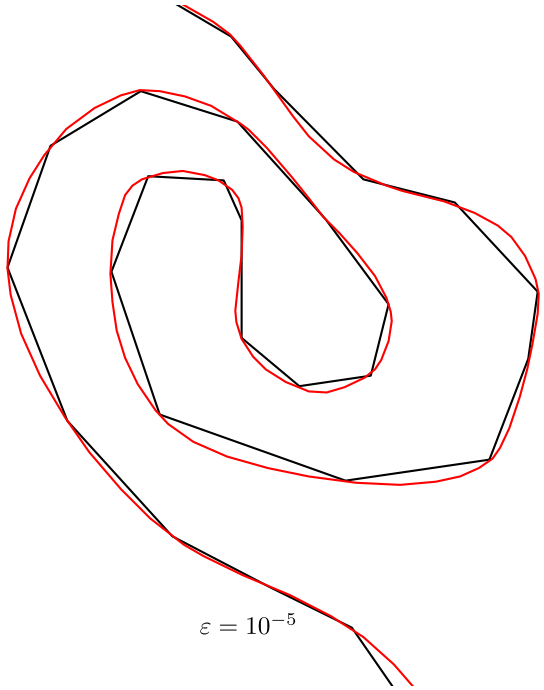


Figure 5.14: A portion of tributary T_3 and T_3^{ε} at the spiral feature A

Figures 5.10–5.14 show the *positions* of the smoothed tributaries with respect to the original polyline. The smoothed tributaries are shown for the accuracy parameters $\varepsilon = 10^{-3}$, $\varepsilon = 10^{-4}$, and $\varepsilon = 10^{-5}$.

Figures 5.10 and 5.11 show tributaries T_1, T_2 and their smoothing. The figures display the entire extent of the tributaries. We can observe that the endpoints are preserved in the smoothed representations. In other words, T_1 and T_1^{ε} have the same endpoints. The same is also true for T_2 and T_2^{ε}. This confirms the design of the wavelet algorithm and Theorem 5.3.4.

Figures 5.12–5.14 show a portion of tributary (flowline) T_3 in *Lake Champlain* and its smoothing. Note that the smoothing is done for the entire 142.526 km length of the tributary, but the figures show only a 1.5 km portion. Figure 5.14 shows that the smoothed tributary T_3^{ε} for $\varepsilon = 10^{-5}$ passes through the vertices of the original polyline T_3. Recall that T_3^{ε} is designed to approximate the *spline* and

not the polyline. The spline is not shown in Figure 5.14, but it is indistinguishable from T_3^{ε} for $\varepsilon = 10^{-5}$.

To evaluate the quality of the approximation by the system, we define the maximal deviation $E(\varepsilon)$ by

$$E(\varepsilon) = \max_{0 \le t \le \gamma_T} d\big(T(t),\ T^{\varepsilon}(t)\big),$$

where $d(P,\ Q)$ is the distance between points P and Q on Earth. This distance can be computed by using the haversine formula (section 5.1.1). Note that the maximal deviation $E(\varepsilon)$ is computed along the entire length of the tributary. Table 5.2 shows the maximal deviation $E(\varepsilon)$ for the sample tributaries T_1, T_2, T_3 for accuracies $\varepsilon_1 = 10^{-3}$, $\varepsilon_2 = 10^{-4}$, and $\varepsilon_3 = 10^{-5}$.

Table 5.2: Maximal deviation of tributaries T_1, T_2, T_3

	$E(\varepsilon_1)$	$E(\varepsilon_2)$	$E(\varepsilon_3)$
T_1	198.8 m	30.5 m	11.2 m
T_2	303.3 m	60.5 m	21.7 m
T_3	867.8 m	158.9 m	28.7 m

Figures 5.15–5.17 display the maximal deviation for tributaries T_1, T_2, and T_3 as functions of the accuracy parameter ε. Note that in these figures, the values of ε are shown in logarithmic scale. The figures confirm that ε controls the smoothing in a consistent manner.

Figures 5.18–5.20 provide more detailed information about the deviation of T_3 from T_3^{ε}. The deviation is shown for the portion of the tributary from 50 km to 70 km from its start. Recall that the spiral feature A is located at 64.65 km from the start of the tributary. Its range is highlighted in the figures. We observe in Figure 5.18 that for, ε_1, the deviation at this feature is about 500 m. The high peaks between 50 km and 58 km correspond to similar spiral features located

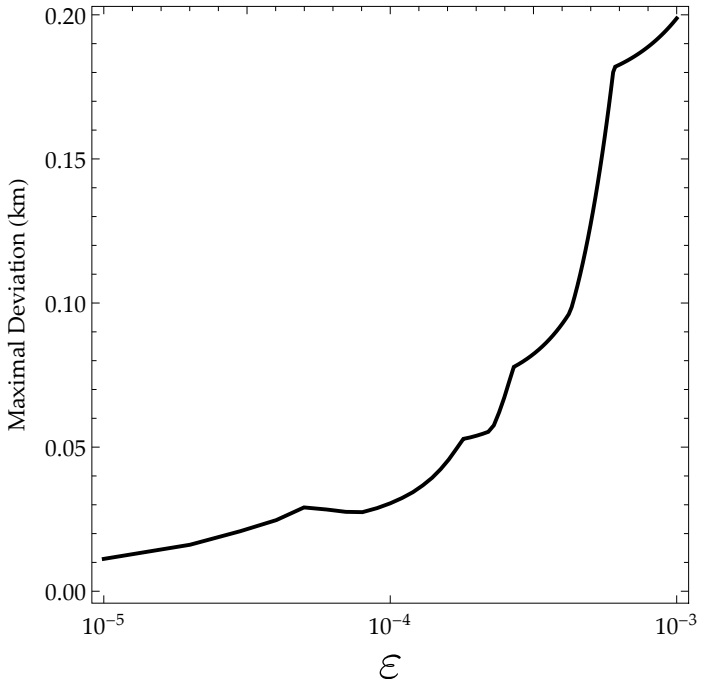


Figure 5.15: Maximal deviation of tributary T_1 versus ε

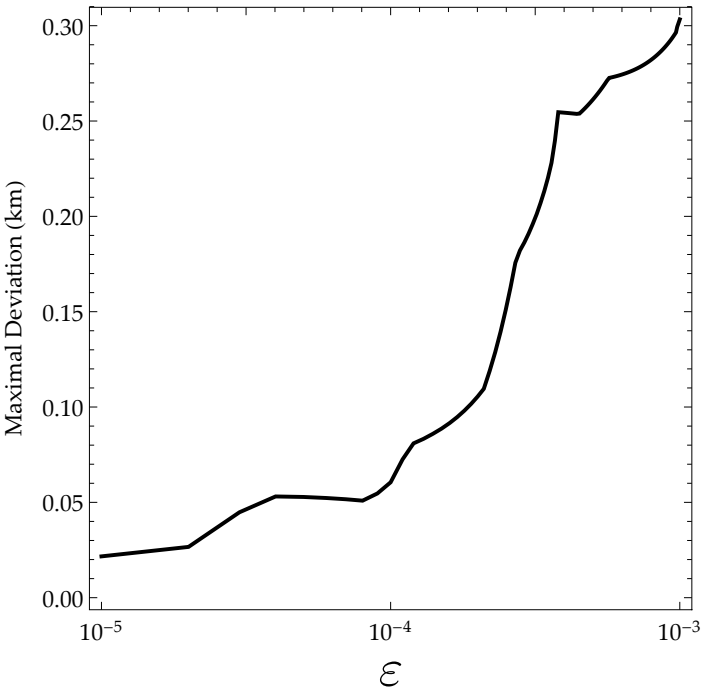


Figure 5.16: Maximal deviation of tributary T_2 versus ε

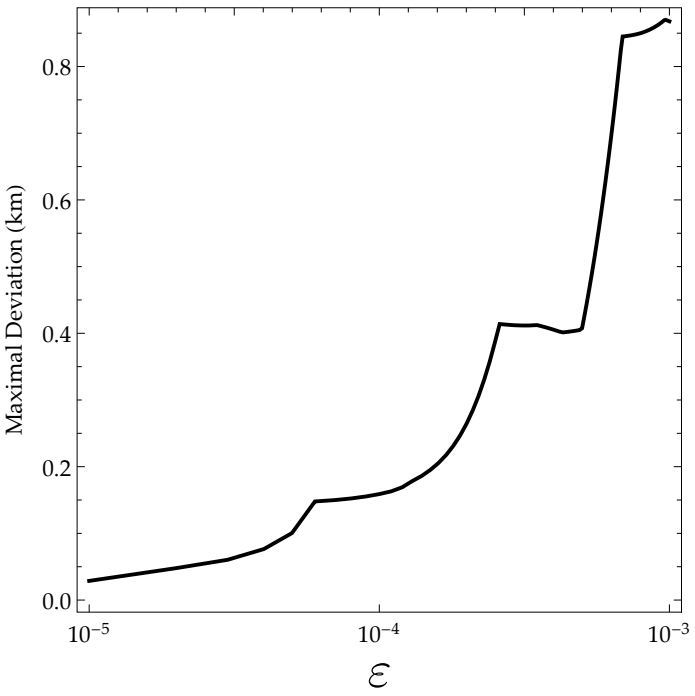


Figure 5.17: Maximal deviation of tributary T_3 versus ε

south of feature A.

Figure 5.19 shows a deviation at the spiral feature A that is less than 150 m for ε_2. Figure 5.20 shows that at A the deviation is less than 10 m for ε_3. This deviation information is presented in a different form in Figure 5.21. The green color indicates the deviation at the point is less than 10% of the maximal deviation stated in Table 5.2.

The complete Vermont river system contains 5,322 tributaries. Section 6.1.3 describes how they are obtained. Figures 5.22–5.24 show the maximal deviation $E(\varepsilon)$ of a tributary T versus its length γ_T. The experiments were conducted for ε_1, ε_2, and ε_3. For clarity, we have split the tributaries into short (0–7 km) and long (7–60 km) groups. The figures show a slight correlation between tributary length and the maximal deviation $E(\varepsilon)$. This correlation is more pronounced in the short group.

Evaluation summary

- The algorithms for tributary decomposition and synthesis perform as designed. They preserve the endpoints of the tributary. Figures 5.10 and 5.11 show tributaries T_1, T_2 and their smoothing. The figures display the entire extent of the tributaries. We can observe that the endpoints are preserved in the smoothed representations. This confirms the design of the wavelet algorithm and Theorem 5.3.4.

- The algorithm shows the ability to smooth a variety of curve shapes. Figures 5.12–5.14 show spiral feature A along tributary T_3 and its smoothing. The algorithm achieves a full range of smoothing for this complex shape.

- The smoothed tributaries show a continuous dependency on the accuracy parameter ε.

- The accuracy parameter ε controls the maximal deviation of the tributary. Figures 5.15–5.17 show that a decrease in ε correlates to a nearly linear decrease in the maximal deviation.

- Figures 5.18–5.20 show that the algorithm achieves smaller deviation in less sinuous sections of the tributary. A topic of future research is to make the deviation more uniform across the entire tributary.

- The experimental data shows a correlation between maximal deviation and tributary length. This correlation is more pronounced for short tributaries and less pronounced for long tributaries. This is illustrated in Figures 5.22–5.24. A topic of future research is to eliminate this correlation.

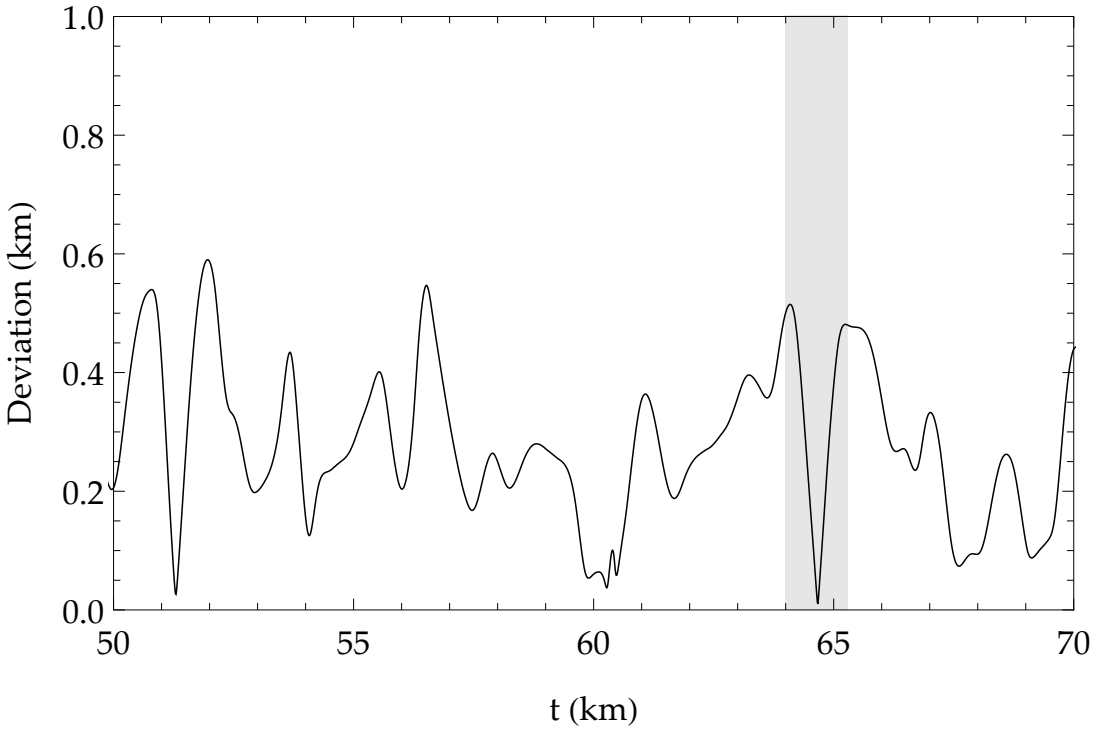


Figure 5.18: Deviation of T_3^{ε} for $\varepsilon_1 = 10^{-3}$

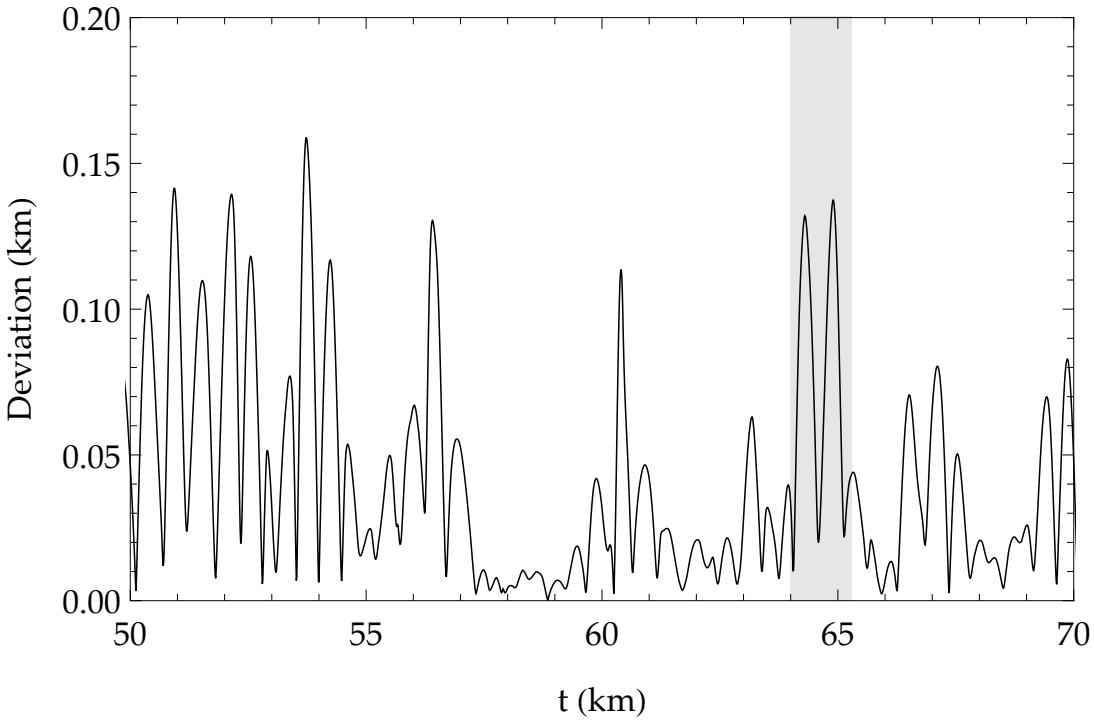


Figure 5.19: Deviation of T_3^ε for $\varepsilon_2 = 10^{-4}$

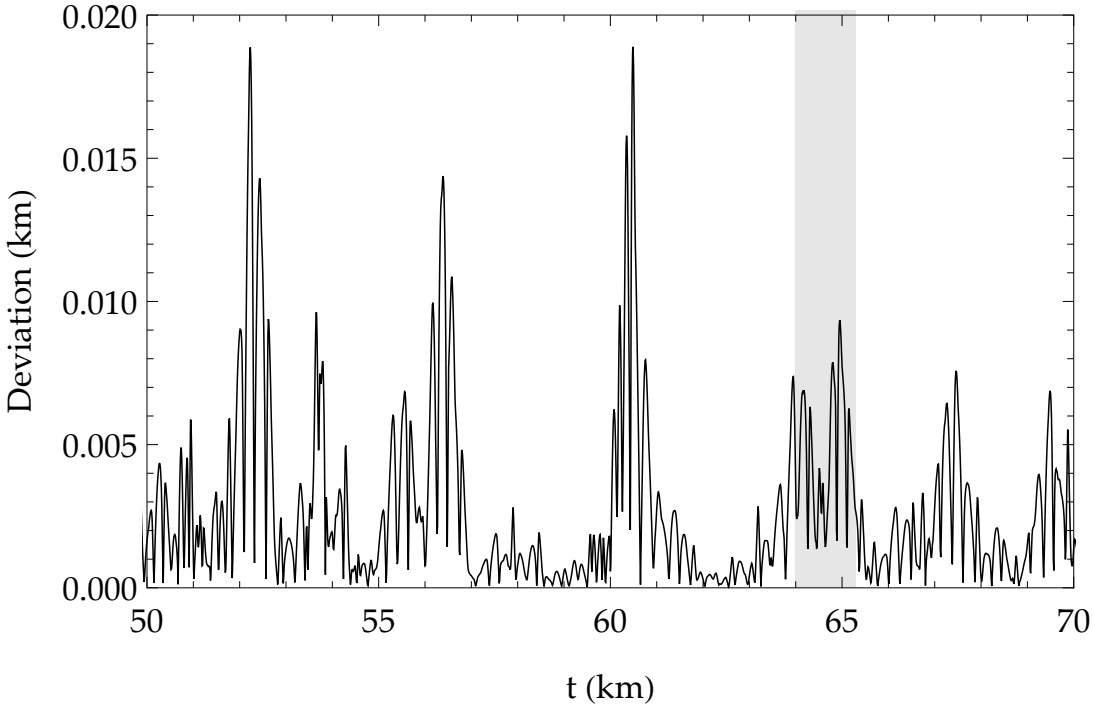


Figure 5.20: Deviation of T_3^ε for $\varepsilon_3 = 10^{-5}$

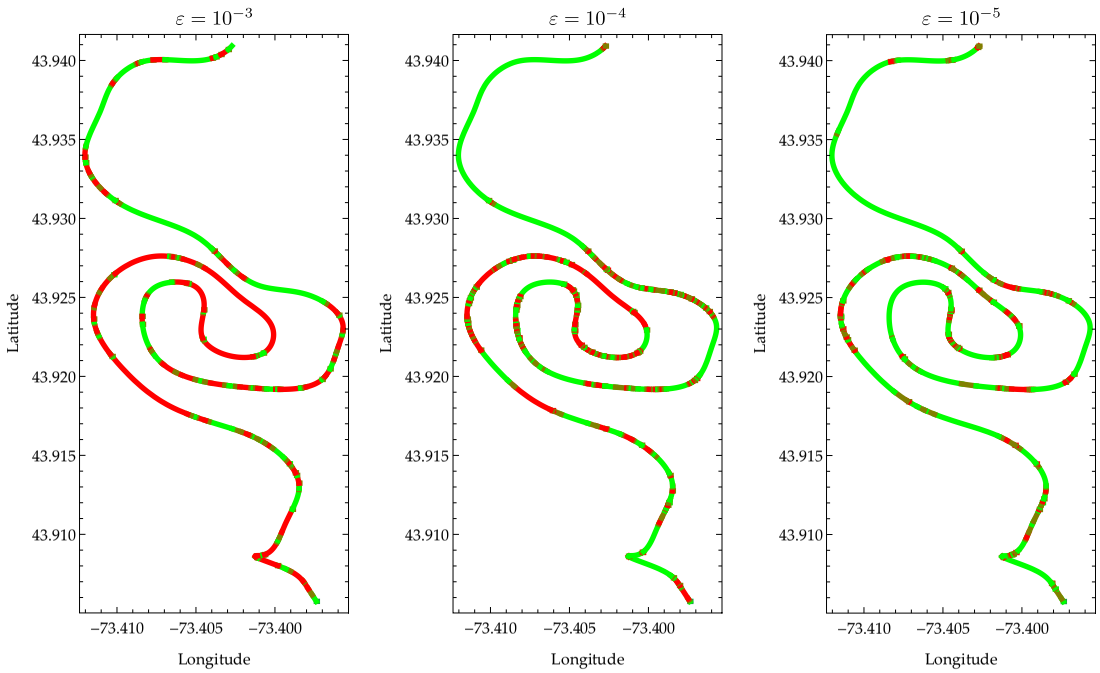


Figure 5.21: A portion of tributary T_3^{ε} at the spiral feature A. The green color indicates that the deviation at the point is less that 10% of the maximal deviation $E(\varepsilon)$. The smoothing more accurately represents the original tributary for smaller ε. This is shown by having more green color.

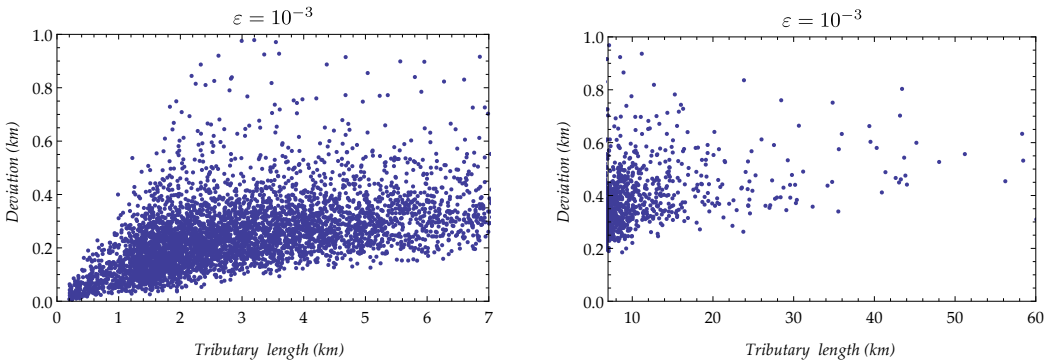


Figure 5.22: Deviation vs tributary length for $\varepsilon_1 = 10^{-3}$

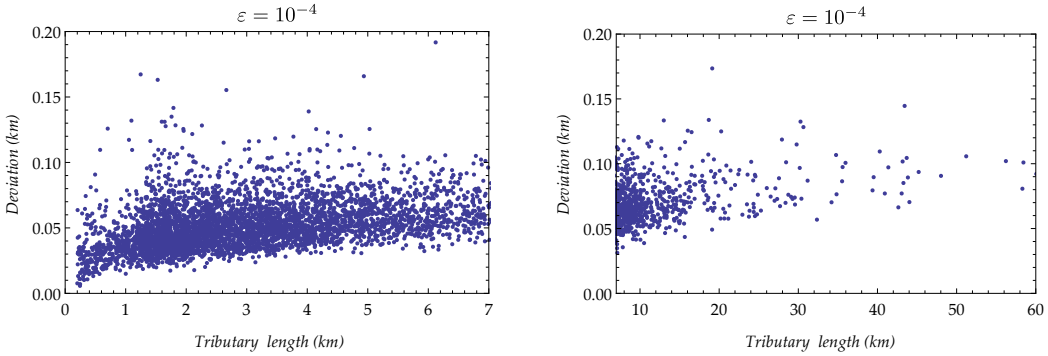


Figure 5.23: Deviation vs tributary length for $\varepsilon_2 = 10^{-4}$

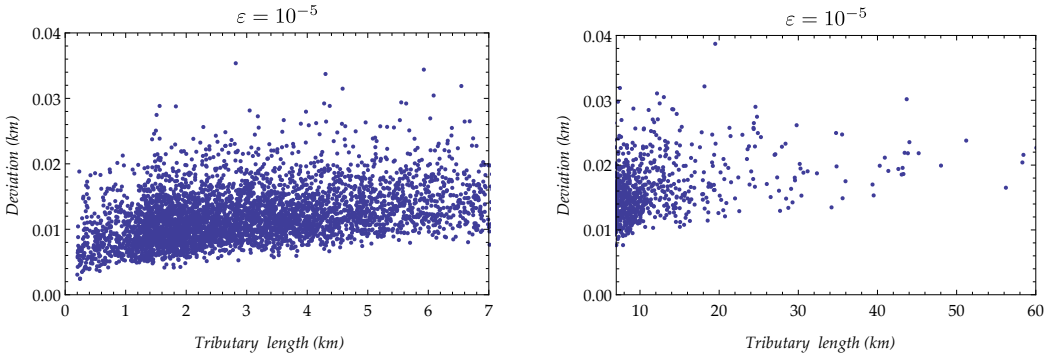


Figure 5.24: Deviation vs tributary length for $\varepsilon_3 = 10^{-5}$

Chapter 6

A method for river network generalization

In the previous chapter we discussed our method for smoothing a single tributary. Now we describe a method to generalize a collection of tributaries that form a river network.

Our river network generalization system consists of two stages: the preprocessing stage and the generalization stage. The preprocessing stage is performed to create a special database $\mathcal{D}$. The generalization stage uses this database to produce a generalized river network for the requested geographic region $\mathcal{B}$.

The preprocessing stage takes the input geographic data and performs the following steps:

- Remove network cycles
- Assign Strahler numbers to network edges
- Extract tributaries according to their Strahler numbers
- Decompose each tributary

- Record parent-child tributary relationships
- Store the collection of tributaries in database $\mathcal{D}$

A complete description of the implemented database is provided in Chapter 7.

The model smooths and prunes tributaries in a manner that is continuous with respect to the map's scale.

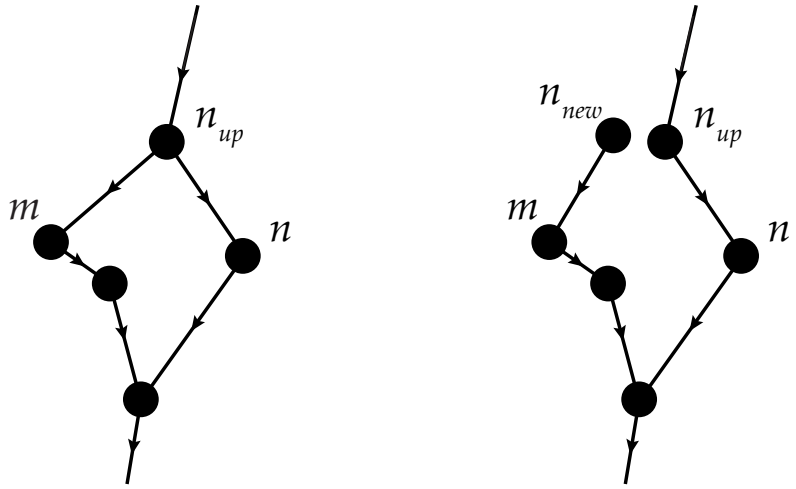
The generalization stage performs the following steps:

- Extract a collection of tributaries $\mathcal{T}_{\mathcal{B}} \in \mathcal{D}$ that intersect region $\mathcal{B}$
- Synthesize the tributaries in the collection
- Restore the connectedness of the graph
- Render the tributaries (section 7.5)

The generalization stage uses two functions $\varepsilon(s)$ and $\sigma(s)$. When a user requests a map of a geographic region $\mathcal{B}$, the system infers the scale s of the requested map. The first function $\varepsilon(s)$ relates the accuracy of the smoothing to s, and the second function $\sigma(s)$ relates the amount of pruning to s. The processing of the user's request involves the determination of tributaries that should be shown on the map, their prioritization, their wavelet-based smoothing, and their pruning.

6.1 Preprocessing stage

The goal of the preprocessing stage is to take the input river network data and create a database $\mathcal{D}$ of tributaries to support real-time generalization and high performance rendering.



(a) Cycle in graph P (b) Resolved cycle in graph G

Figure 6.1: Removing cycles from a graph according to the UndoCycles algorithm. Nodes n_{up} and n_{new} are distinct, but have the same coordinates.

We represent the input river network data as a directed graph $\mathcal{G}$. Each node in $\mathcal{G}$ may have upstream and downstream edges. If a node has no upstream edges, it is a *source*. If a node has no downstream edges, it is a *mouth*. It is important to note that graph $\mathcal{G}$ may contain more than one connected component, and it may contain cycles.

Our method for extracting tributaries from $\mathcal{G}$ relies on assigning Strahler numbers to the edges of the graph. The algorithm for such an assignment requires that the graph be a tree. A tree is a graph where each node does not have more than one downstream edge.

6.1.1 De-cycling of a river network graph

We remove the cycles in $\mathcal{G}$ by applying the UndoCycles algorithm, see Figure 6.1. It performs a breadth-first search of the graph and marks nodes as *visited*. If an already *visited* node n_{up} is encountered again, then a cycle is detected. This

means that the node n_{up} is connected to at least two downstream edges e_1 and e_2. To remove the cycle, we duplicate the node n_{up} as a new node n_{new} having the same coordinates. We then disconnect the edge e_1 from n_{up} and connect it to n_{new}. This algorithm produces a tree graph $\mathcal{G}_{tree}$ that represents a river network with no cycles. The pseudocode for UndoCycles is shown in Algorithm 6.1.

Input: River network graph $\mathcal{G}$
Output: Graph $\mathcal{G}_{tree}$
1 **for each** mouth node n in $\mathcal{G}$ **do**
2 Traverse the upstream nodes of n in a breadth-first manner.
3 **if** a node n_{up} is **not** *visited* **then**
4 Mark n_{up} as *visited*
5 **else**
6 Since n_{up} is already visited, it means that it is connected to at least one more downstream node m besides n.

  To remove the cycle we do the following:
7 (a) Create a duplicate node n_{new} with the same coordinates as n_{up}
8 (b) Disconnect n_{up} from m
9 (c) Connect n_{new} to m
10 (d) Mark n_{new} as *visited*
11 **end if**
12 Continue the traversal until all the nodes are *visited*
13 **end for**

Algorithm 6.1: The UndoCycles algorithm

6.1.2 Strahler number determination

Stream ordering algorithms are used to prioritize branches of a river network. There are several stream ordering algorithms, but the most popular amongst them is Strahler's Stream Ordering Algorithm [59]. An efficient implementation of this algorithm was discovered by Gleyzer, et. al. [20] in 2004.

For instance, in Figure 6.2, the flow of the river moves from left to right. The mouth of the river network is the rightmost point. The Strahler algorithm

assigns the Strahler number $S = 1$ to all streams that begin at a source. The six sources are located at the beginning point of the edges labeled with a 1. The algorithm assigns $S = 2$ to edges where two or more $S = 1$ edges merge. In general, when two edges with the same Strahler number S merge, then the downstream edge will have the Strahler number $S + 1$.

Input: River network directed acyclic graph $\mathcal{G}$
Output: A labeled directed acyclic graph $\mathcal{G}$

1 For each *source* node assign a Strahler number $S = 1$ to its downstream edge
2 Let m be a node with all labeled upstream edges and an unlabeled downstream edge e_{down}.
3 Let S be the maximum of the Strahler numbers assigned to the upstream edges of m.
4 Let $count$ be the number of upstream edges of m with the Strahler number S.
5 **if** $count \geq 2$ **then**
6 Strahler number of e_{down} is $S + 1$
7 **else**
8 Strahler number of e_{down} is S
9 **end if**
10 Repeat until all the edges are labeled

Algorithm 6.2: The STRAHLER algorithm

The UNDOCYCLES algorithm produces a directed acyclic graph $\mathcal{G}_{tree}$. For simplicity we will refer to it as just $\mathcal{G}$. The next step is to assign Strahler num-

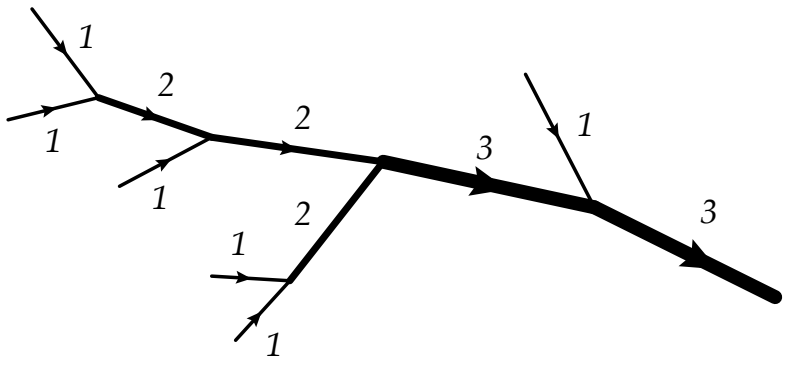


Figure 6.2: A sample river network with Strahler stream numbers

bers to the edges of $\mathcal{G}$. This is done by applying the STRAHLER algorithm to each connected component of $\mathcal{G}$. The pseudocode for STRAHLER is shown in Algorithm 6.2.

6.1.3 Extraction of tributaries

Our method smooths linear features of $\mathcal{G}$. These linear features, which we call *tributaries*, are extracted according to their Strahler number. The idea of the tributary extraction algorithm GETTRIBUTARIES (Algorithm 6.3) is to chain edges of the graph with the same Strahler number. As the algorithm traverses the graph, it inspects upstream edges of the current node and decides if the tributary chain is to be extended upstream, or a new tributary chain is to be started.

Let $\mathcal{G}$ be a river network directed acyclic graph labeled with Strahler numbers. The GETTRIBUTARIES algorithm is applied to each mouth node in $\mathcal{G}$. The result is a collection of tributaries $\mathcal{T}$ of the river network $\mathcal{G}$.

Let T_1 and T_2 be two tributaries. If T_2 flows into T_1, then T_1 is the *parent* of T_2 and T_2 is a *child* of T_1. The parent-child relationship between tributaries can be derived from the network graph $\mathcal{G}$. Each tributary T retains a reference to its parent tributary. This reference includes a unique identifier for the parent tributary. Let A be the point where T_2 merges into T_1. The percentage *fractionJoint* of the length from the start of the parent tributary T_1 to A is recorded in the child tributary T_2. The *fractionJoint* is used in the generalization stage to restore the connectedness of the graph.

If a tributary T_{new} was created as a result of a split in the network graph $\mathcal{G}$ (see UNDOCYCLES algorithm), then we retain in T_{new} the unique identifier of the

tributary T of which it was split from. In addition, we also record the percentage of the length from the the start of T to the split point.

The pseudocode for GetTributaries is shown in Algorithm 6.3. An example of tributaries extracted from a network graph is shown in Figure 6.3.

Input: A tributary T with n as its most upstream node
Output: A collection of tributaries $\mathcal{T}$

```
for each upstream edge e of node n do
    if e_Strahler = T_Strahler then
        Chain e to T
        Call GetTributaries(T)
    else
        Create an empty tributary T_new and add it to 𝒯
        Chain e to T_new
        Call GetTributaries(T_new)
    end if
end for
return 𝒯
```

Algorithm 6.3: The GetTributaries algorithm

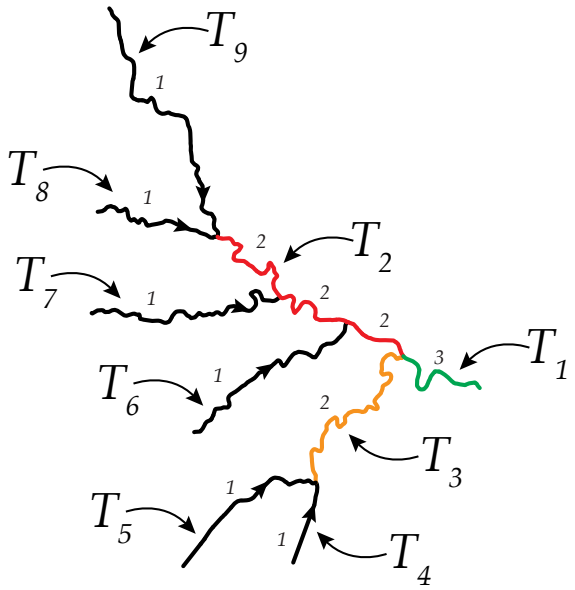


Figure 6.3: Tributaries in a river network

6.1.4 Tributary decomposition

Each extracted tributary in $\mathcal{T}$ is processed by the TributaryDecomposition algorithm described in section 5.2.4, with an initial accuracy parameter $\varepsilon_0 \geq 0$. The output of the algorithm along with the other tributary attributes is stored in database $\mathcal{D}$. The attributes of a tributary that are stored in database $\mathcal{D}$ are described in section 7.3.

6.2 Generalization stage

The generalization stage uses database $\mathcal{D}$, created during the preprocessing stage, and two functions relating the map scale to the map appearance. The first function $\varepsilon(s)$ relates the map scale to the generalization accuracy. Let $\varepsilon \geq \varepsilon_0$ be the accuracy of the approximation of tributaries on a given map. Note that the accuracy depends on the map scale. The map scale s is expressed as a fraction. For example, a map with scale $s = 1 : 10,000$ represents each $10,000$ centimeters (100 meters) on Earth with 1 centimeter on the map.

For an $s = 1 : 10,000$ scale map, we may accept an accuracy of $\varepsilon = 40$ meters on Earth. For a smaller scale map $s = 1 : 100,000$, the corresponding accuracy may be $\varepsilon = 400$ meters. To be consistent with the preprocessing stage, we choose $\varepsilon(s_{max}) = \varepsilon_0$, where s_{max} is the largest expected map scale.

The second function $\sigma(s)$ associates each scale s with a threshold Strahler number to be used for displaying the tributaries. For example, we may decide that only the tributaries with a Strahler number of 3 or higher are to be displayed on maps with scale $1 : 100,000$ or smaller. Both functions $\varepsilon(s)$ and $\sigma(s)$ are determined by experimentation to provide users with the most satisfactory map appearance.

The generalization stage consists of the following steps: First, the tributaries visible in the requested map are extracted from the database, and pruned according to $\sigma(s)$. Then they are synthesized with an accuracy of $\varepsilon(s)$. Next, the tributaries are repositioned on the map using the tree restoration algorithm to restore the network connectedness. Finally, the de-cycling algorithm is reversed to restore the network cycles.

6.2.1 Tributary synthesis

The user supplies a bounding box $\mathcal{B}$ of a geographic region of interest along with the map scale s. We define the collection of tributaries $\mathcal{T}_{\mathcal{B}}$ as all the tributaries $T \in \mathcal{D}$ that intersect the bounding box $\mathcal{B}$, and such that their Strahler numbers $T_{Strahler}$ satisfy $T_{Strahler} \geq \lfloor \sigma(s) \rfloor$.

For each tributary $T \in \mathcal{T}_{\mathcal{B}}$ use the TributarySynthesis algorithm from section 5.3.3 with accuracy $\varepsilon = \varepsilon(s)$ to construct the smoothed tributary T^{ε}.

6.2.2 Generalized tree formation and restoration of cycles

The collection of tributaries obtained in the previous step does not necessarily preserve the connectedness of the original river network, as shown in Figure 6.4. To reestablish the network connectedness, we use the FormTree algorithm.

The FormTree algorithm uses the collection $\mathcal{T}_{\mathcal{B}}$ of tributaries. The algorithm generates a dependency graph $\mathcal{Q}$ that describes the parent-child relationship between the tributaries in $\mathcal{T}_{\mathcal{B}}$. The algorithm assigns higher priorities to parents than to their children.

Let tributary T_2 be a child of parent tributary T_1. The point A where T_2 merges with T_1 is located at $fractionJoint \times \gamma_{T_1}$ along T_1. The value of

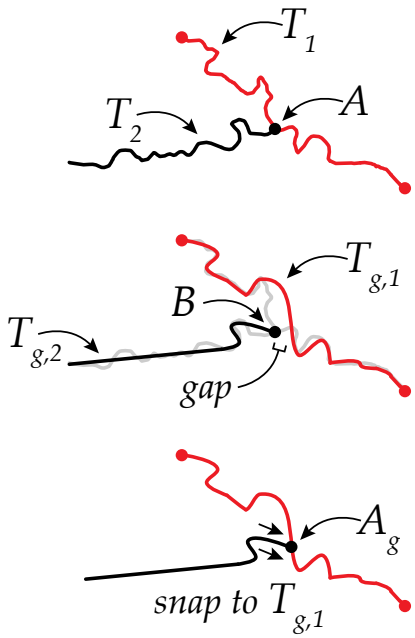


Figure 6.4: Restoring river network connectedness after generalization

$fractionJoint$ is stored with the child tributary T_2 in $\mathcal{D}$. During the tree formation process, point A_g is located at $fractionJoint \times \gamma_{T_{g,1}}$ along $T_{g,1}$. Let B (the same position as A) be the end point of tributary T_2. To reconnect (snap) the generalized tributary $T_{g,2}$ to $T_{g,1}$, we simply translate $T_{g,2}$ by vector $\overrightarrow{BA_g}$.

Input: Collection of generalized tributaries $\mathcal{T}_{\mathcal{B}}$
Output: Collection of connected generalized tributaries $\mathcal{T}_{g,\mathcal{B}}$

1 Generate parent-child dependency graph $\mathcal{Q}$
2 Let $T_{g,1}$ be the tributary with the highest priority in $\mathcal{T}_{\mathcal{B}}$
3 Let $T_{g,2}$ be a child of $T_{g,1}$
4 Let A_g be the point located at $fractionJoint \times \gamma_{T_1}$ along $T_{g,1}$
5 Let B be the end point of $T_{g,2}$
6 Translate $T_{g,2}$ by $\overrightarrow{BA_g}$ and add it to $\mathcal{T}_{g,\mathcal{B}}$
7 Repeat for every child and sub-child of $T_{g,1}$ in hierarchical order
8 Repeat from Line 2 starting with the next highest priority in $\mathcal{T}_{\mathcal{B}}$

Algorithm 6.4: The FormTree algorithm

The final step of the generalization stage is the restoration of cycles. The

parent-child relationship between tributaries was introduced in section 6.1.3. According to this definition, child tributaries merge (flow) into their parents. To avoid confusion, we describe river network splits by introducing the ancestor-descendant relationship.

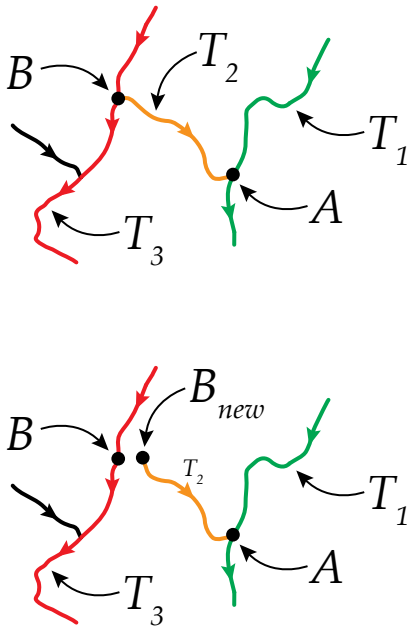


Figure 6.5: Original river network before generalization showing a split at B (top). Generalized river network after FormTree algorithm, but before FormCycles algorithm (bottom).

Let B be a point along tributary T_3 where the flow splits into two, see Figure 6.5 (top). The start point of tributary T_2 is at B. Tributary T_3 is the *ancestor* of T_2. Tributary T_2 is the *descendant* of T_3.

During the decycling process, point B was artificially doubled. The new point B_{new} became the new starting point of tributary T_2. After the FormTree algorithm the tributaries $\mathcal{T}_{g,\mathcal{B}}$ are generalized and form a tree. However, the position of B_{new} may not coincide with B anymore. For clarity, we will omit the g subscript from the discussion. It is also omitted from Figure 6.5 (bottom).

Our goal is to reattach T_2 to T_3. The FormCycles algorithm uses the collection $\mathcal{T}_{g,\mathcal{B}}$ of generalized tributaries. The algorithm generates a dependency graph $\mathcal{R}$ describing the ancestor-descendant relationship between the tributaries in $\mathcal{T}_{g,\mathcal{B}}$. The algorithm assigns higher priorities to ancestors than to their descendants. Then, according to these priorities, the descendant tributaries are reattached to their ancestors.

Reattachment is achieved by a linear transformation. In Figure 6.5, we reattach T_2 to T_3 by transforming it in such a way that point B_{new} overlaps B, but point A remains in place. Let γ be the length of tributary T_2. Then a parametric representation of T_2 is $\big(x(t),\, y(t)\big),\ 0 \leq t \leq \gamma$. The transformed tributary T_2' is given by

$$\vec{T}_2'(t) = \vec{T}_2(t) + \left(1 - \frac{t}{\gamma}\right) \overrightarrow{B_{new}B}.$$

Under this transformation, when $t = 0$, we have

$$\vec{T}_2'(0) = \vec{T}_2(0) + \overrightarrow{B_{new}B} = B_{new} + \overrightarrow{B_{new}B} = B.$$

On the other hand, when $t = \gamma$ we have

$$\vec{T}_2'(\gamma) = \vec{T}_2(\gamma) = A.$$

6.2.3 Smooth pruning

Pruning (refinement) is an essential element of cartographic generalization. Our collection of tributaries $\mathcal{T}$ is prioritized by Strahler numbers. Given a map scale s, we define $\mathcal{T}_{\mathcal{B}}$ as the collection of all the tributaries $T \in \mathcal{T}$ that are visible in $\mathcal{B}$ and satisfy the condition $T_{Strahler} \geq \lfloor \sigma(s) \rfloor$.

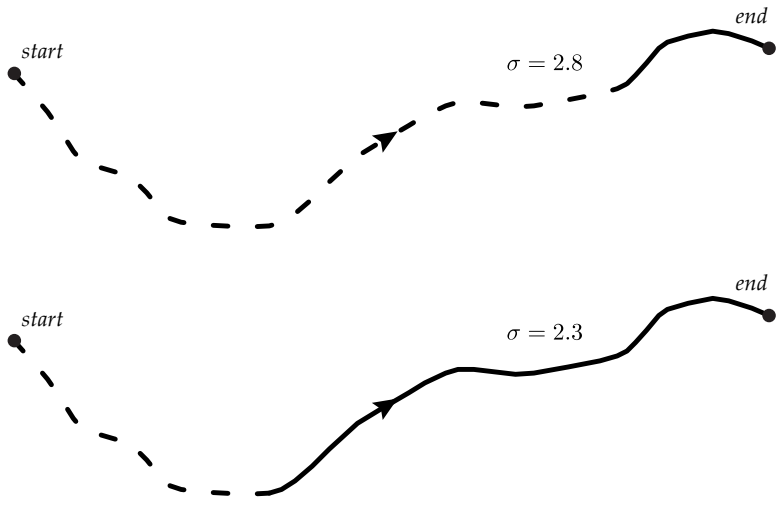


Figure 6.6: Smooth pruning of a tributary with Strahler number $S = 2$. For $\sigma = 2.8$, only 20% of the tributary is displayed. For $\sigma = 2.3$, 70% of the tributary is displayed.

However, under this pruning strategy a small change in scale s can cause a sudden appearance or disappearance of a tributary. This does not provide a good user experience (see Chapter 7). To remedy this problem, we introduce a new *smooth pruning* strategy.

Under this strategy, all tributaries in $\mathcal{T}_{\mathcal{B}}$ with $T_{Strahler} \geq \lfloor \sigma(s) \rfloor + 1$ are displayed for their entire length. However, if the Strahler number of the tributary satisfies

$$\lfloor \sigma(s) \rfloor \leq T_{Strahler} < \lfloor \sigma(s) \rfloor + 1,$$

then only a portion of tributary T is displayed. More precisely, if $0 \leq t \leq \gamma_T$ then the point $T(t)$ is shown only if

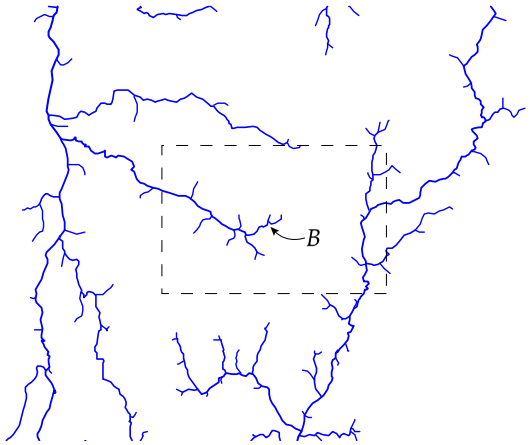
$$t \geq (\sigma(s) - \lfloor \sigma(s) \rfloor)\gamma_T.$$

In our implementation, functions $\sigma(s)$ and $\varepsilon(s)$ are defined by

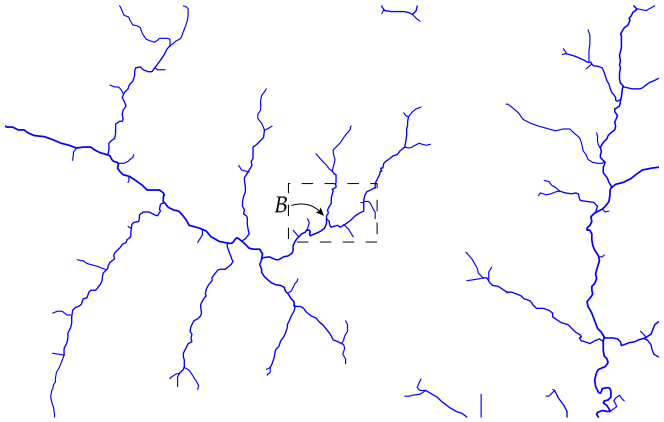
$$\sigma(s) = \frac{4.5}{\sqrt{s}} \quad \text{and} \quad \varepsilon(s) = \frac{5 \times 10^{-10}}{s},$$

where s is the map scale. We have also tested different functions and concluded that these functions give adequate results.

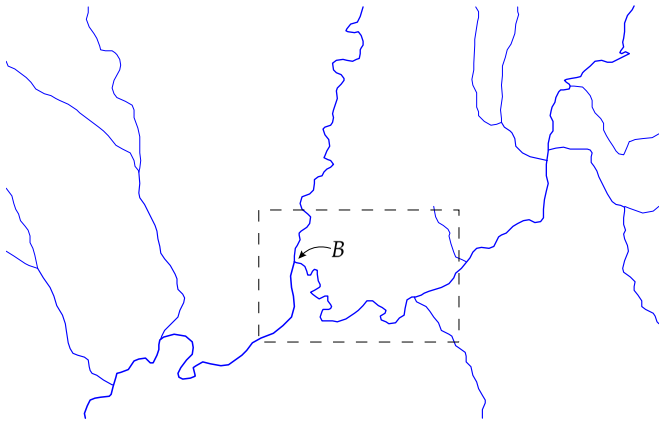
Figure 6.7 shows a portion of the Vermont river network, as displayed by our system at four different map scales. The point B indicates the junction of the *Winooski River* and the *Kingsbury Branch*, located at (-72.4538°, 44.2830°). These figures show the result of applying of smooth pruning and continuous smoothing to the network.



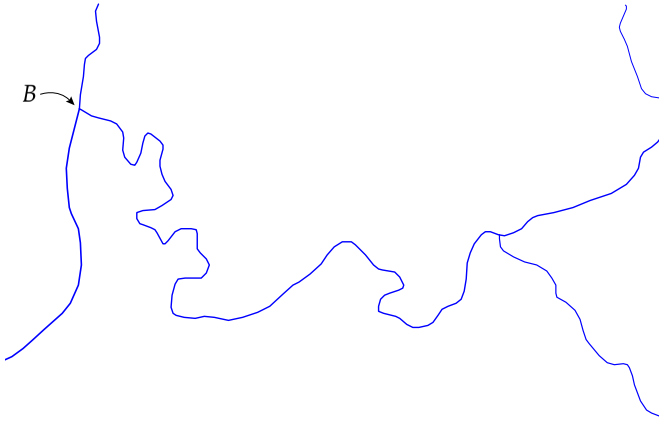
(a) $s = 1 : 1,350,360$



(b) $s = 1 : 451,749$



(c) $s = 1 : 67,453$



(d) $s = 1 : 20,385$

Figure 6.7: Renderings of the Vermont river network near the river junction B at four different map scales s.

Chapter 7

Implementation

In this chapter we discuss the implementation of our river network generalization method. The goal of our implementation is to produce an interactive viewer for hydrographic data. The viewer illustrates the practicality of our wavelet-based smoothing framework. We used the Vermont flowline hydrography as our base dataset. This dataset comes from the United States Geological Survey (USGS) National Hydrography Dataset (NHD) website. The next step was to use a spatial index to build an in-memory graph that represented the river network. At this stage we implemented the UndoCycles, Strahler, and GetTributaries algorithms. The system data structures are also presented in this chapter.

The TributaryDecomposition algorithm was implemented according to our dimensional wavelet theory described in Chapter 5. Smooth functions require fewer wavelet details to represent them accurately, than non-smooth ones. Accordingly, our implementation adds a step to spline interpolation to the tributary polylines. This also produces a more natural representation of the actual physical feature.

Our renderer uses the TributarySynthesis and FormTree algorithms.

Additional steps are taken to improve its rendering speed. As a result the system executes in subsecond time.

We have conduced a study to evaluate the responsiveness of the implemented viewer and to compare it to the National Map Viewer. The design and details of the study are presented in section 7.7. The evaluation study material is attached in Appendix A. The participants indicated that our viewer was more responsive than the USGS National Map viewer. The participants preferred our smooth pruning strategy, as well as multi-touch trackpads. They indicated that our viewer was also easier to use for zooming operations.

7.1 Dataset source

We used the hydrography of the State of Vermont to evaluate the implementation of our system. It was obtained online from the USGS National Hydrography Dataset (NHD) [64]. The hydrography is available in three resolutions: local (1:5,000), high (1:24,000), and medium (1:100,000). We chose the medium resolution dataset to keep the computational time and memory requirements within manageable limits.

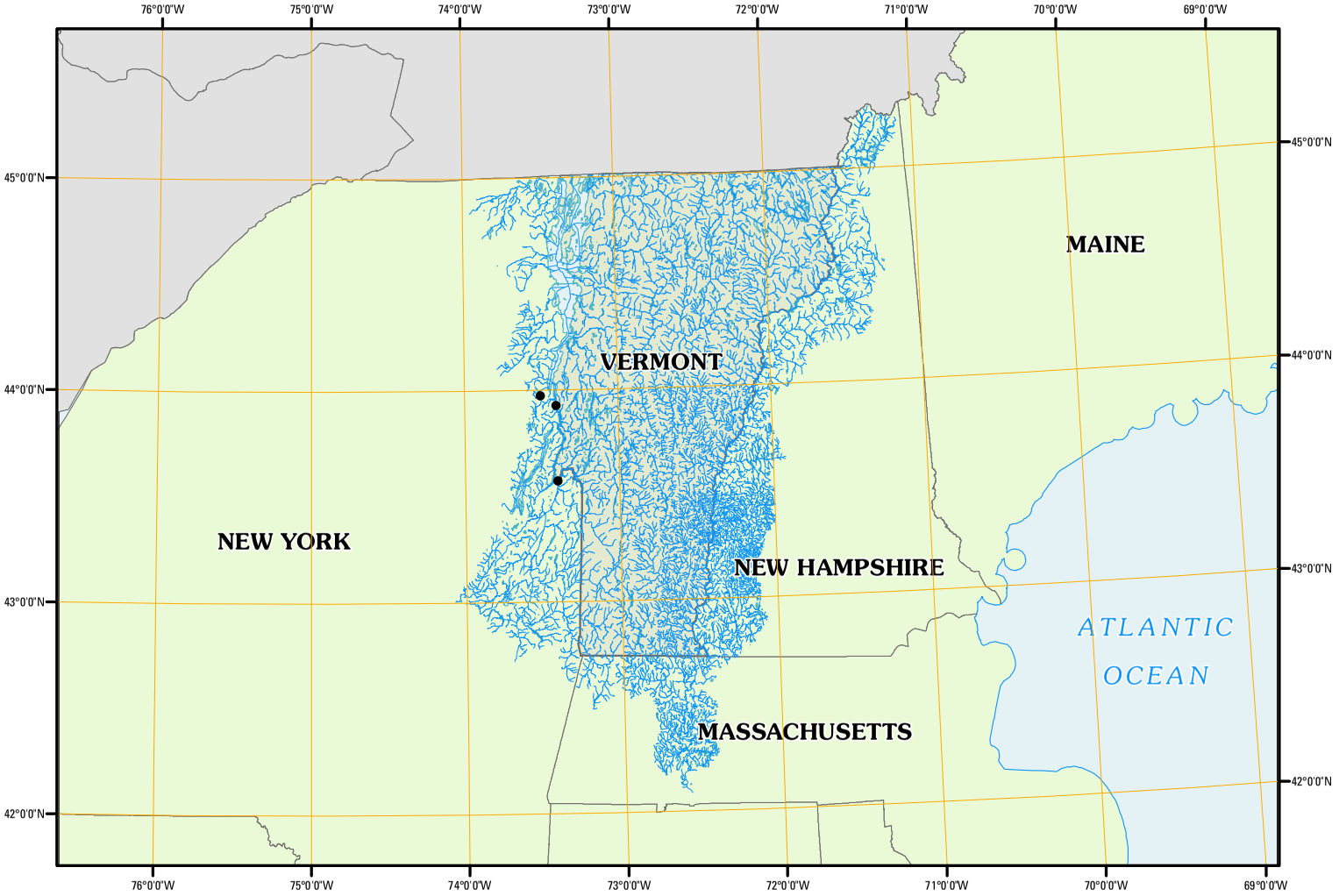


Figure 7.1: Vermont hydrography map

7.1.1 Retrieval and preprocessing of the dataset

We have taken the following steps to obtain the shapefile (a standard GIS data storage format):

- Download the Vermont dataset from `http://nhd.usgs.gov/`.
- Import the *Hydrography* dataset into a new map in *ArcMap*.
- Uncheck all the layers except *NHDFlowLine*
- Create a new definition query for *NHDFlowline* with the following SQL statement `"FlowDir" = 1` . This ignores all the features that do not participate in the hydrography flow graph
- Export the selected layer to a new SHP file
  - Note: Select *Export Shapes in View*, not *All Shapes*
- Name the output shapefile file `vermont.shp`

In Figure 7.1, the retrieved Vermont dataset is shown on a map. We observe that the density of the river system makes it difficult to see the major waterways. We also see that the hydrography system extends beyond Vermont's state borders. The current presentation is unsuitable for most maps. A map should prioritize major waterways and hide minor streams.

7.2 Graph generation

To apply our wavelet-based method (described in Chapter 6), the river system must be represented as a directed graph $\mathcal{G}$. The `vermont.shp` shapefile is a sequence of *records*. Each record is a sequence of vertices that forms the polyline. Our goal is to generate the directed graph $\mathcal{G}$ from these records.

7.2.1 Edges and nodes

To generate the directed graph $\mathcal{G}$, we first create a new `Graph` object (Figure 7.2). Then we add an `Edge` (Figure 7.4) for each record in the shapefile. We set the unique identifier of the edge equal to the shapefile record number. We also set the length of the edge. The start `Point` and end `Point` are set to the start and end vertices of the polyline.

Now that we have our list of edges, we would like to connect them together through nodes. To achieve this, we examine each `Edge`. We want to see if a `Node` (Figure 7.3) exists that matches the edge's start position. If there is a match, we set the current edge's **StartPosition** to be equal to that `Node`. If no `Node` exists at that position, then a new one is created. The same procedure is performed for the edge's **EndPosition**. When a `Node` is assigned to an `Edge` we also update the **Upstream** and **Downstream** arrays. At the end of the procedure, we obtain an in-memory object graph that represents the river network.

Graph class

Edges	`Edge[]`	A list of all the edges in the graph
Nodes	`Node[]`	A list of all the nodes in the graph

Figure 7.2: Data structure for the `Graph` class

Node class

Position	`Point`	The location of the node
IsVisited	`bool`	Indicates if the node was visited
Upstream	`Edge[]`	A list of upstream edges to this node
Downstream	`Edge[]`	A list of downstream edges from this node

Figure 7.3: Data structure for the `Node` class

Edge class		
ID	`int`	The unique identifier for this edge
StartPosition	`Point`	The location of the start of the edge
EndPosition	`Point`	The location of the end of the edge
StartNode	`Node`	The node at the start of the edge
EndNode	`Node`	The node at the end of the edge
Length	`double`	The length of the edge in kilometers
Strahler	`int`	The Strahler number of the edge

Figure 7.4: Data structure for the `Edge` class

7.2.2 Tributary extraction

To perform tributary extraction, we need to assign Strahler numbers to the edges of the graph. The STRAHLER algorithm (section 6.1.2) requires that the graph be a directed acyclic graph (tree). To do this, we apply the UNDOCYCLES algorithm (section 6.1.1) to each mouth `Node` in the graph. This may result in several connected components. Once the Strahler numbers are assigned we can extract tributaries by using the GETTRIBUTARIES algorithm (section 6.1.3). The resulting collection of tributaries is $\mathcal{T}$.

7.3 System data structures

The main concept of our method is *tributary*. The system uses the `Tributary` object to represent it, as shown in Figure 7.5. The tributary collection $\mathcal{T}$ is converted to a collection of `Tributary` objects that form database $\mathcal{D}$. At this point the following `Tributary` attributes are set: **ID**, **Name**, **Start**, **End**, **Strahler**, **ParentID**, **AncestorID**, **FractionJoint**, **FractionJointStart**, **TopRight**, **BottomLeft**, and **Length**.

The preprocessing stage uses frame signals $\mathbf{c}$ and detail signals $\mathbf{d}$. According

Tributary class		
ID	`int`	The unique identifier for the tributary
Name	`String`	The geographic name of the tributary
Start	`Point`	The starting position of the tributary
End	`Point`	The ending position of the tributary
Strahler	`int`	The Strahler number of the tributary
ParentID	`int`	The ID of the parent tributary
AncestorID	`int`	The ID of the ancestor tributary
FractionJoint	`double`	Percentage of the length of the parent tributary where this tributary joins it
FractionJointStart	`double`	Percentage of the length of the ancestor tributary where this tributary splits away from it
TopRight	`Point`	The top right corner of the bounding box
BottomLeft	`Point`	The bottom left corner of the bounding box
Length	`double`	The length of the tributary in kilometers
JX	`int`	The finest level for the x–component
JY	`int`	The finest level for the y–component
DetailsX	`RangeArray[]`	The wavelet details for the x–component
DetailsY	`RangeArray[]`	The wavelet details for the y–component
SlopeX	`double`	The slope for the x–component
SlopeY	`double`	The slope for the y–component
InterceptX	`double`	The intercept for the x–component
InterceptY	`double`	The intercept for the y–component

Figure 7.5: Data structure for the `Tributary` class

RangeArray class		
Data	`double[]`	The internal array of data
StartIndex	`int`	The starting index of the range
EndIndex	`int`	The ending index of the range
`double` **GetDataAtIndex**(`int`)		Retrieves data at the requested index

Figure 7.6: Data structure for the `RangeArray` class

WrappingArray class	
Data `double[]`	The internal array of data
`double` **GetDataAtIndex**(`int`)	Retrieves data at the requested index in IMP form. The index can be negative and/or beyond the range of the internal array.

Figure 7.7: Data structure for the `WrappingArray` class

to Chapter 4, frame signals $\mathbf{c}$ are IMP signals. Theoretically, they are infinite signals, but practically it is only necessary to store their representative part. We use the `WrappingArray` object (Figure 7.7) to store such IMP data.

The detail signals $\mathbf{d}$ are stored using the `RangeArray` object (Figure 7.6). The `RangeArray` object extends an `Array` object to allow it to be defined from a starting index to an ending index. These indices can have effectively any integer value.

7.4 Database generation

The first step in database $\mathcal{D}$ generation is to replace the polyline representation of $T \in \mathcal{T}$ with a natural cubic spline representation. To achieve this, we use the polyline vertices as the knots of the natural cubic spline interpolation. We use the spline interpolated tributaries as the input to the TributaryDecomposition algorithm.

7.4.1 Spline interpolated tributaries

Smooth functions require fewer terms to approximate than functions which have discontinuities and corners. The x and y components of a tributary T are piece-

wise linear functions which are continuous, but not differentiable at the vertices. The same is true for the IMP representation components x^* and y^*.

Let $x_s^*(t),\ 0 \leq t \leq \gamma_T$ be the natural cubic spline interpolant (through the polyline vertices) of $x^*(t)$. Thus $(x_s^*)''(0) = (x_s^*)''(\gamma_T) = 0$. Let it be extended to all $t \in \mathbb{R}$ as an IMP function. Since $(x_s^*)(0) = (x_s^*)(\gamma_T) = 0$ we conclude that $x_s^*(t),\ t \in \mathbb{R}$ is a twice continuously differentiable function. To simplify the notation we will write x^* for x_s^*, and y^* for y_s^*. Since the components are splines, their values can be easily computed for any $0 \leq t \leq \gamma_T$.

The result is the tributary representation by IMP. It is a continuous, and twice continuously differentiable (smooth) function. An immediate benefit of this representation is a reduction in the number of wavelet coefficients needed to be stored.

7.4.2 Cubic spline interpolation

Kincaid [31] offers a complete discussion of spline interpolation methods. We present a summary here.

Suppose that we are given mesh points $t_k,\ k = 0, 1, \ldots, M$, and the corresponding values x_k at these points. The natural cubic spline $S(t),\ t \in [t_0, t_M]$ interpolation of these mesh points is a twice continuously differentiable function with $S''(t_0) = S''(t_M) = 0$ (the natural spline condition), satisfying $S(t_k) = x_k,\ k = 0, 1, ..., M$. In addition, over any subinterval $[t_k, t_{k+1}]$, the spline $S(t)$ is required to be a cubic polynomial. In other words, $S(t)$ is a piecewise cubic function which has continuous first and second derivatives on $[t_0, t_M]$.

The key to finding $S(t)$ is to determine the values of the second derivatives of $S(t)$, $z_k = S''(t_k)$, at each mesh point t_k. These values satisfy the following

tridiagonal linear system of equations

$$\begin{pmatrix} u_1 & h_1 & & & & & \\ h_1 & u_2 & h_2 & & & & \\ & h_2 & u_3 & h_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & h_{M-3} & u_{M-2} & h_{M-2} & \\ & & & & h_{M-2} & u_{M-1} & \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{M-2} \\ z_{M-1} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{M-2} \\ v_{M-1} \end{pmatrix},$$

where

$$h_k = t_{k+1} - t_k, \quad u_k = 2\big(h_k + h_{k-1}\big), \quad b_k = \frac{6}{h_k}\Big(x_{k+1} - x_k\Big), \quad v_k = b_k - b_{k-1},$$

$k = 1, \ldots, M-1$. Such a tridiagonal system can be efficiently solved using a Gaussian elimination method.

Evaluation of the spline function $S(t)$

Given any particular value of $t \in [t_0,\, t_M]$, our goal is to find the value of the spline function at t. The pseudocode for SplineEvaluation is shown in Algorithm 7.1. This algorithm also computes the values $S'(t)$ and $S''(t)$.

7.4.3 Decomposition

The spline interpolated components of a tributary T provide the input to the TributaryDecomposition algorithm described in section 5.2.4. Using the outputs, we can assign `Tributary` object attributes: **JX**, **JY**, **DetailsX**, **DetailsY**, **SlopeX**, **SlopeY**, **InterceptX**, **InterceptY**. All the tributaries are stored in the database $\mathcal{D}$, and this completes the preprocessing stage.

Input: Sequences t_k, x_k, z_k, $k = 0, 1, \ldots, M$ and $t \in [t_0, t_M]$
Output: $S(t)$, $S'(t)$, $S''(t)$

1 Find k such that $t_k \leq t \leq t_{k+1}$.

2 Compute

$$h_k = t_{k+1} - t_k, \quad A_k = \frac{1}{6h_k}\Big(z_{k+1} - z_k\Big), \quad B_k = \frac{z_k}{2},$$

and

$$C_k = -\frac{h_k}{6}z_{k+1} - \frac{h_k}{3}z_k + \frac{1}{h_k}\Big(x_{k+1} - x_k\Big).$$

3 Compute

$$\begin{aligned} S(t) &= x_k + (t - t_k)\Big[C_k + (t - t_k)[B_k + (t - t_k)A_k]\Big], \\ S'(t) &= C_k + 2(t - t_k)B_k + 3(t - t_k)^2 A_k, \\ S''(t) &= 2B_k + 6(t - t_k)A_k. \end{aligned}$$

4 **return** $S(t)$, $S'(t)$, $S''(t)$

Algorithm 7.1: The SplineEvaluation algorithm

7.5 Rendering

During interaction with the viewer, a user requests a map of a region $\mathcal{B}$. From the region and the window size we can determine the scale s of the map. Section 6.2 describes the actions in the generalization stage: tributary synthesis, graph restoration, and smooth pruning. These steps produce the collection of connected generalized tributaries $\mathcal{T}_{g,\mathcal{B}}$ to be rendered.

We compute points that are visible in region $\mathcal{B}$ along each generalized tributary $T_g \in \mathcal{T}_{g,\mathcal{B}}$. The points are spaced 1 pixel apart in the view window. Since the rendering of lines is a time-consuming operation, it is necessary to reduce the number of drawing operations.

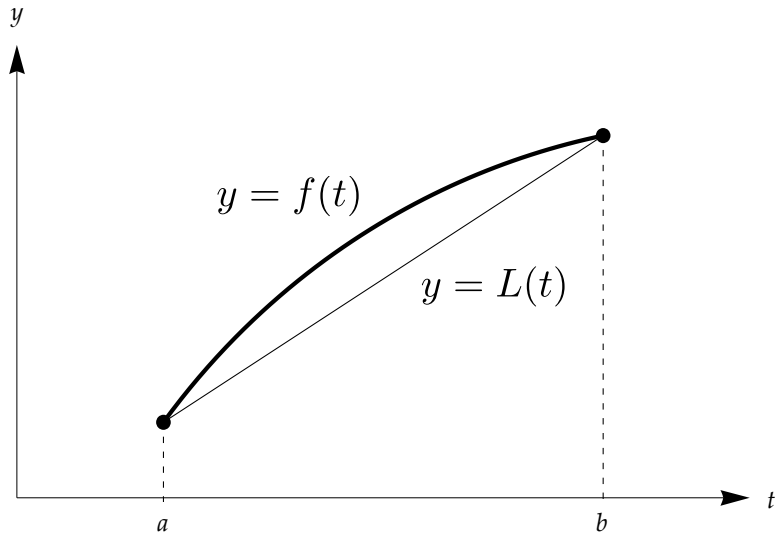


Figure 7.8: Function f and its linear interpolation L

A basic drawing operation is the DrawLine function. It draws a straight line between two points, with a specified color and thickness. We call points supplied to the DrawLine function the *drawing points*.

The points forming a tributary are computed 1 pixel apart. However, if the tributary is nearly a straight line on a certain interval, it is much more efficient to supply only two drawing points (the beginning and the end of the interval). To select the drawing points we use the following inequality that describes the maximal deviation of a curve away from its secant line.

Let f be a twice continuously differentiable function on an interval $[a, b]$, and L be the linear function joining the points $\big(a, f(a)\big)$ and $\big(b, f(b)\big)$, as shown in Figure 7.8. It is established in Prenter [45] that the following Lagrange error estimate is valid

$$\max_{a \le t \le b} |f(t) - L(t)| \le \frac{(b-a)^2}{8} \max_{a \le t \le b} |f''(t)|. \tag{7.1}$$

Accordingly, suppose that we want the deviation $\max_{a \le t \le b} |f(t) - L(t)|$ to be less than the pixel tolerance tol. So we require

$$\frac{(b-a)^2}{8} \max_{a \le t \le b} |f''(t)| \le tol.$$

Then the length of the interval should satisfy

$$|b-a| \le \sqrt{\frac{8 \times tol}{\max_{a \le t \le b} |f''(t)|}}. \tag{7.2}$$

Suppose that $T_g(t_0) = \big(x(t_0),\, y(t_0)\big)$ is a drawing point P. We proceed by incrementing m. If t_m is such that inequalities

$$|t_m - t_0| \le \sqrt{\frac{8 \times tol}{\max_{t_0 \le t \le t_m} |x''(t)|}}, \quad |t_m - t_0| \le \sqrt{\frac{8 \times tol}{\max_{t_0 \le t \le t_m} |y''(t)|}} \tag{7.3}$$

are satisfied, but an inequality in (7.3) with t_m replaced with t_{m+1} is not satisfied, then $Q = T_g(t_m)$ is the next drawing point.

The drawing points form a subset of all the points in the fine mesh.

7.6 Performance

In this section, we evaluate the size of the database $\mathcal{D}$ and the rendering time. The size of the `vermont.shp` shapefile is 9.71 MB. The preprocessing stage transforms it into the database $\mathcal{D}$. The size of $\mathcal{D}$ depends on the initial accuracy parameter ε_0, supplied to the TributaryDecomposition algorithm (section 5.2). Table 7.1 shows the size of $\mathcal{D}$ for various values of ε_0.

An initial accuracy parameter should not be increased beyond $\varepsilon_0 = 10^{-5}$ because this would result in an unacceptable loss of accuracy during the tributary

Table 7.1: Size of database $\mathcal{D}$ vs Initial accuracy ε_0

ε_0	Size of $\mathcal{D}$
0	26.3 MB
10^{-7}	26.3 MB
10^{-6}	26.2 MB
10^{-5}	22.3 MB

Table 7.2: Rendering performance for various scales

Scale s	# of tributaries	Synthesis	Rendering	Total time
1:4,000,000	12	0.005 s	0.008 s	0.020 s
1:2,000,000	59	0.021 s	0.043 s	0.071 s
1:1,000,000	174	0.050 s	0.091 s	0.150 s
1:500,000	326	0.097 s	0.104 s	0.209 s
1:300,000	152	0.093 s	0.074 s	0.174 s
1:200,000	108	0.054 s	0.067 s	0.128 s
1:100,000	163	0.116 s	0.055 s	0.178 s
1:50,000	85	0.100 s	0.045 s	0.153 s
1:25,000	85	0.150 s	0.028 s	0.185 s
1:10,000	58	0.129 s	0.025 s	0.161 s

synthesis process. Even though the size of database $\mathcal{D}$, for $\varepsilon_0 = 10^{-5}$, is about 2.3 times bigger than the original shapefile, it contains all the multiresolution information necessary for smoothing the tributaries at all scales.

Table 7.2 provides rendering performance information depending for various map scales. The window size for these experiments was fixed at 1024×768 pixels. We conducted the experiments on a 2.0 GHz Intel® Core i7-2635QM Processor notebook computer.

The table shows that at small scales, most tributaries are pruned away. At large scales, the geographic region $\mathcal{B}$ is small so it contains few tributaries. The rendering time directly correlates with the number of tributaries visible in the window. The synthesis time increases with the map scale because a larger scale implies a smaller value for the accuracy ε. This means that more wavelet details

are needed to fulfill the accuracy requirement.

The total time reflects the amount of time needed to load the tributaries from database $\mathcal{D}$, and to perform graph restoration, synthesis, and rendering. It is the total time from view request to final display. The table shows that the system performs in less than 0.209 seconds for any map scale. This allows for an interactive user experience.

7.7 Evaluation

7.7.1 Setup

We performed a study of our viewer (Vermont Viewer) to evaluate its performance and to compare it to the USGS National Map Viewer [65]. The University of Oklahoma Institutional Review Board (IRB) reviewed the study; Figure A.7 documents the approval of our protocol. Ten participants from the School of Computer Science and Department of Geography responded to our recruitment e-mail.

The physical setup for the study was a computer lab setting, as shown in Figure A.6. Our viewer was installed and opened on an iMac® desktop computer with a mouse and a multi-touch trackpad (Magic Trackpad™). The viewer included several buttons that affected the rendering. A screenshot of the viewer is shown in Figure A.1. Additionally, the website for the National Map Viewer was opened in a web browser window.

Participants were asked to complete several tasks as detailed in the questionnaire. All tasks involved interaction with the viewers. The complete questionnaire is shown in Figures A.2–A.5. It consists of multiple choice questions as well

as several open ended questions. On average, the study took 20 minutes for a participant to complete.

Here is a brief description of the tasks:

Task 1.1 Use the National Map Viewer to find Points A, B and C
Determine the flow directions of the rivers near the points
Evaluate the responsiveness of the viewer

Task 1.2 Use the Vermont Viewer to find Point A
Evaluate the responsiveness of the viewer

Task 2.1 Use the Vermont Viewer to find Point B
Zoom in and out on Point B using Mode 2.1 and Mode 2.2
Describe which mode is preferable

Task 3.1 Use the Vermont Viewer to find Point C
Zoom in and out on Point C using the mouse and multi-touch trackpad
Describe which input device is preferable

Task 4.1 Use the Vermont Viewer to find Point D
Zoom in and out on Point D using Mode 4.1 and Mode 4.2
Describe which mode is preferable

Summary Evaluate the responsiveness of both viewers
Evaluate which zoom feature was easier to use
Suggest improvements to the viewer
Provide comments

7.7.2 Results

Tables 7.3–7.9 display aggregated results for the multiple choice questions:

Table 7.3: Indicate how responsive the National Map Viewer is (Task 1.1)

Too Slow	Slow	Did not notice	Fast Enough	Fast
0%	30%	0%	60%	10%

Table 7.4: Indicate how responsive the Vermont Viewer is (Task 1.2)

Too Slow	Slow	Did not notice	Fast Enough	Fast
0%	0%	0%	60%	40%

Table 7.5: Which mode do you prefer? (Task 2.1)

I prefer 2.1	2.1 is a little better than 2.2	I don't prefer one over the other	2.2 is a little better than 2.1	I prefer 2.2
20%	20%	0%	30%	30%

Table 7.6: Which experience do you prefer? (Task 3.1)

I prefer the mouse wheel	The mouse wheel is a little better than the trackpad	I don't prefer one over the other	The trackpad is a little better than the mouse wheel	I prefer the trackpad
20%	0%	10%	50%	20%

Table 7.7: Which mode do you prefer? (Task 4.1)

I prefer 4.1	4.1 is a little better than 4.2	I don't prefer one over the other	4.2 is a little better than 4.1	I prefer 4.2
20%	10%	30%	10%	30%

Table 7.8: Which map viewer did you find to be the most responsive?

The National Map viewer was the most responsive	The National Map viewer was a little more responsive than the Vermont viewer	I don't prefer one over the other	The Vermont viewer was a little more responsive than the National Map viewer	The Vermont viewer was the most responsive
0%	10%	10%	10%	70%

Table 7.9: Which map viewer's "zoom" feature was easier to use?

The National Map viewer was easier to zoom	The National Map viewer was a little easier to zoom than the Vermont viewer	I don't prefer one over the other	The Vermont viewer was a little easier to zoom than the National Map viewer	The Vermont viewer was easier to zoom
0%	20%	10%	30%	40%

7.7.3 Analysis

Task 1.1 was designed to evaluate the responsiveness of the National Map Viewer. Task 1.2 was designed to evaluate the responsiveness of the Vermont Viewer. The participants indicated that the National Map Viewer has sufficient responsiveness. However, they found the Vermont Viewer to be more responsive. 30% of the participants indicated that the National Map Viewer was "slow". Only 1 participant indicated that the viewer was "fast". On the other hand, all participants found the Vermont viewer to be either "fast enough" or "fast". It should be noted that the National Map Viewer is a website, and the Vermont Viewer is a locally installed application. Consequently, our viewer does not have to wait for the download of data.

Task 2.1 was designed to evaluate two different pruning strategies. Map mode 2.1 utilized the *smooth pruning* strategy, described in section 6.2.3. Map mode 2.2

utilized the *discrete pruning* strategy. When participants were asked to evaluate two map modes in Task 2.1, the preference was mixed. However, a closer analysis of the participants' responses to the open ended questions for Task 2.1 shows that some participants were not responding to the intent of the task. The goal of the task was to compare two different pruning strategies. In the *discrete pruning* strategy, the entire length of the tributary is displayed once a threshold zoom level is achieved. On the other hand, in the *smooth pruning* strategy the tributaries appear gradually.

All participants who did understand the intent of the task preferred *smooth pruning*. Participants who preferred mode 2.2, indicated that they chose that mode because it shows more rivers. One participant stated, "For the same amount of scrolling, 2.2 shows more branches of rivers than 2.1." We conclude that the task was not properly designed to evaluate the pruning strategies. A better task would explicitly state that the purpose is to evaluate pruning strategies.

Task 3.1 was designed to evaluate two different input devices for performing zooming operations. Participants were asked to compare their use of a mouse wheel to a multi-touch trackpad. The results of Task 3.1 show a strong preference for the multi-touch trackpad for performing zooming operations. One female participant indicated that she preferred the mouse wheel. We observed that she had difficulty using the multi-touch trackpad because of her long nails. A participant who is a guitar player preferred the mouse wheel as well because the trackpad could not detect his callused fingers.

Task 4.1 was designed to evaluate the participants preference for the smoothing of linear features for two different accuracy parameters ε. The synthesis part of the generalization stage is described in section 6.2.1. We conclude that participants did not have a preference for either approach. An important implication for

the implementation of the system is that users may find slightly lower accuracy maps to be acceptable. Therefore, it is possible to decrease the rendering time by using fewer wavelet details.

Finally, we asked the participants to compare the National Map Viewer to the Vermont Viewer to summarize their experiences. They were asked to evaluate the responsiveness of the viewers and the ease-of-use in zooming operations.

In the open ended questions, participants suggested improvements to the Vermont Viewer, including a double-click interaction to zoom in, a search box, additional map layers, labels of features, a click-to-center interaction, and to display the map scale.

After the participants completed all the tasks, they were asked to summarize their experiences. Overwhelmingly, 70% of the participants indicated that the "Vermont viewer was the most responsive" when compared to the National Map Viewer. Additionally, 70% of the participants indicated that the Vermont viewer was either a "little easier" or "easier" to use for zooming operations. We conclude that our method is useful for designing a map viewer that supports continuous zoom operations.

Chapter 8

Conclusions

8.1 Automated smoothing

We were able to create a framework for the automated smoothing of river networks based on the following major contributions:

- A wavelet–based method for polyline smoothing and endpoint preservation
- Inverse Mirror Periodic (IMP) representation of functions and signals, and dimensional wavelets
- Smoothed features do not abruptly change between scales
- Features are pruned in a continuous manner with respect to scale
- River network connectedness is maintained for all scales
- Reuse of a base geographic dataset for all scales
- Design and implementation of an interactive map viewer for linear hydrographic features that renders in subsecond time

The creation of a map is a tedious and time consuming activity. Researchers are looking for ways to formalize this process with the purpose of identifying the elemental operations and automating them.

Our system demonstrates that a wavelet-based approach is well suited for basic generalization operations. It provides smoothing and pruning that is continuously dependent on map scale. The size of database $\mathcal{D}$, that contains all the multi-resolution information, is of the same order as the original dataset. The system works at a speed that allows for fluid interactions with the map.

Smoothing algorithms produce a smoothed version of a tributary that is continuously dependent on some accuracy parameter. Examples include Perkal's ε–circle method, Fourier transform method, and Gaussian smoothing. However, these methods have deficiencies that make them ineffective. Wavelet-based methods are multi-resolution by design. Additional effort is required to ensure that endpoints do not move and smoothing is done continuously. Our algorithm fulfills these requirements.

The use of wavelets for smoothing has not previously produced satisfactory results. This is because methods for endpoint preservation required large storage and long execution time. In Chapters 3 and 4, we developed a method for efficient wavelet-based smoothing that preserves the endpoints of a tributary. These chapters describe our method for merging wavelets with IMP representations.

Our contribution is to represent a tributary in Inverse Mirror Periodic (IMP) form. Next we introduced dimensional wavelets. Such wavelets are appropriately scaled to the size of the tributaries they approximate.

Numerical experiments presented in Chapter 5 show that our wavelet method produces cartographically appropriate smoothing for tributaries. The experiments were conducted for more than 5,000 tributaries from the Vermont dataset.

They show a slight correlation between the accuracy of the approximation and the length of the tributary.

The method is able to correctly smooth complex geographic features such as a spiral. As expected, straight segments are approximated with less deviation than sinuous ones.

The smoothing of linear features introduces feature drifts. While our method preserves the endpoints, the intermediate points can shift. To produce a cartographically correct generalization of a river network, it is necessary to preserve the network connectedness.

In Chapter 6, we described our method that solves this problem. The process involves removing cycles from the network, prioritizing the segments according to their Strahler numbers, and extracting tributaries. Then each tributary is decomposed into wavelet details as in Chapter 5.

Pruning is a generalization operator that is applied to reduce map clutter. In Chapter 6, we described discrete pruning based on tributary priority. We have also developed a *smooth pruning* strategy.

When the user requests a map of a region $\mathcal{B}$, the window size infers the scale s. Functions $\varepsilon(s)$ and $\sigma(s)$ determine the accuracy and the pruning level. The tributaries that are visible in $\mathcal{B}$ are synthesized to the required accuracy $\varepsilon(s)$ and displayed according to the pruning function $\sigma(s)$. In our *smooth pruning* strategy, the tributaries smoothly come in and out of view depending on the scale.

It is desirable to have a fast enough method that would support the reuse of a single base dataset for on-the-fly smoothing for the production of maps at *any* scale. Our implementation shows that the interactive map renders views in subsecond time. We have determined experimentally that the FBI (9–7) biorthogonal

wavelet family provides the best compromise between quality of approximation and computation time.

Chapter 7 describes details of the implementation and presents an evaluation study of the viewer. The results of the evaluation show that participants found our viewer to be more responsive than the National Map Viewer. They noted that it was easier to use for zooming operations. The participants also indicated their preference for smooth pruning, rather than discrete pruning. Also, they preferred zooming operations with the multi-touch trackpad, as opposed to the mouse wheel.

8.2 Future work

8.2.1 Research

Our wavelet-based smoothing method may be useful for solving the following problems:

- **Rivers as polygons** — For certain scales a river is better represented by a polygon rather than a polyline. Its banks have to be generalized in a coordinated manner to avoid self-intersection. Additionally, the width of the river should be accounted for.

- **Additional generalization operators** — In addition to smoothing and pruning, other generalization operators should be considered. For example, the *exaggeration* operator, overemphasizes important features such as inlets. The rules for exaggeration are different than for smoothing.

- **Interaction with other layers** — For a map to be useful, it should

contain additional layers. The generalization algorithms should take these other layers into account. For example, a road that follows a coastline should always remain on land. Another example is that a city should always appear on the proper bank of a river.

8.2.2 Implementation

The interactive map viewer can be extended to include additional features:

- **Accuracy based on location in addition to scale** — The accuracy of a generalization can be based on other factors such as mouse position. For example, as the mouse pans over a region, the region is rendered with higher detail.

- **Progressive data transfer over the Internet** — As it was mentioned in section 2.5, modern viewers work in a web environment. Wavelets provide a natural mechanism for progressive data transfer.

- **Viewing large geospatial datasets** — Our implementation only views the Vermont dataset. Additional research is necessary to extend the viewer to support large datasets such as the entire US National Hydrography Dataset.

- **Multitouch interaction** — Multitouch interfaces are becoming increasingly common, and users expect to access data through them. Such devices provide new ways to interact with maps in a continuous manner.

Bibliography

[1] P. S. Addison, *The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in Science, Engineering, Medicine and Ainance*. Institute of Physics Publishing, 2002.

[2] V. Antoniou, J. Morley, and M. Haklay, "Tiled Vectors: A Method for Vector Transmission over the Web," in *Web and Wireless Geographical Information Systems*, ser. Lecture Notes in Computer Science, J. Carswell, A. Fotheringham, and G. McArdle, Eds. Springer Berlin / Heidelberg, 2009, vol. 5886, pp. 56–71.

[3] J. L. G. Balboa and F. J. A. López, "Frequency filtering of linear features by means of wavelets. A method and an example," *The Cartographic Journal*, vol. 37, no. 1, pp. 39–49, 2000.

[4] M. Bertolotto and M. J. Egenhofer, "Progressive transmission of vector map data over the World Wide Web," *GeoInformatica*, vol. 5, no. 4, pp. 345–373, 2001.

[5] J. T. Bjørke and S. Nilsen, "Wavelets applied to simplification of digital terrain models," *International Journal of Geographical Information Science*, vol. 17, no. 7, pp. 601–621, 2003.

[6] M. L. Boas, *Mathematical Methods in the Physical Sciences*. Wiley, 2006.

[7] C. Boutoura, "Line generalization using spectral techniques," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 26, no. 3, pp. 33–48, 1989.

[8] J. Brasington, B. T. Rumsby, and R. A. Mcvey, "Monitoring and modelling morphological change in a braided gravel-bed river using high resolution GPS-based survey," *Earth Surface Processes and Landforms*, vol. 25, no. 9, pp. 973–990, Aug. 2000.

[9] D. Burghardt, "Controlled line smoothing by snakes," *GeoInformatica*, vol. 9, no. 3, pp. 237–252, 2005.

[10] M. E. Charlton, A. R. G. Large, and I. C. Fuller, "Application of airborne LiDAR in river environments: the River Coquet, Northumberland, UK," *Earth Surface Processes and Landforms*, vol. 28, no. 3, pp. 299–306, 2003.

[11] L. Chieppa, M. Fiorentino, A. E. Uva, and G. Monno, "Unified interactive wavelet approach for 2D sketch segmentation and editing," *International Journal of Shape Modeling*, vol. 16, no. 1-2, pp. 39–56, 2010.

[12] A. H. J. Christensen, "Line generalization by waterlining and medial-axis transformation. Successes and issues in an implementation of Perkal's proposal," *The Cartographic Journal*, vol. 37, no. 1, pp. 19–28, 2000.

[13] R. G. Cromley and G. M. Campbell, "Integrating quantitative and qualitative aspects of digital line simplification," *The Cartographic Journal*, vol. 29, no. 1, pp. 25–30, 1992.

[14] I. Daubechies, *Ten Lectures on Wavelets*, ser. CBMS-NSF regional conference series in applied mathematics. Society for Industrial and Applied Mathematics, 1992.

[15] B. E. Davis, *GIS: A Visual Approach*. Onword Press/Thomson Learning, 2001.

[16] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, vol. 10, no. 2, pp. 112–122, Dec. 1973.

[17] J. Fdez-Valdivia, J. A. Garcia, and M. Garcia-Silvente, "Simplifying cartographic boundaries by using a normalized measure of ambiguity," *Computers & Geosciences*, vol. 22, no. 6, pp. 607–623, Jul. 1996.

[18] A. Finkelstein and D. H. Salesin, "Multiresolution curves," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 261–268.

[19] E. Fritsch and J. Lagrange, "Spectral representations of linear features for generalisation," in *Spatial Information Theory A Theoretical Basis for GIS*, ser. Lecture Notes in Computer Science, A. Frank and W. Kuhn, Eds. Springer Berlin / Heidelberg, 1995, vol. 988, pp. 157–171.

[20] A. Gleyzer, M. Denisyuk, A. Rimmer, and Y. Salingar, "A fast recursive GIS algorithm for computing Strahler stream order in braided and nonbraided networks," *Journal of the American Water Resources Association*, vol. 40, no. 4, pp. 937–946, 2004.

[21] M. D. Greenberg, *Foundations of Applied Mathematics*. Prentice-Hall, 1978.

[22] E. Guilbert and E. Saux, "Cartographic generalisation of lines based on a B-spline snake model," *International Journal of Geographical Information Science*, vol. 22, no. 8, pp. 847–870, 2008.

[23] E. Guilbert, "Line decomposition based on critical points detection," in *AGILE Conf.*, ser. Lecture Notes in Geoinformation and Cartography, M. Sester, L. Bernard, and V. Paelke, Eds. Springer, 2009, pp. 369–385.

[24] E. Guilbert and H. Lin, "B-Spline curve smoothing under position constraints for line generalisation," *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems GIS'06*, p. 3, 2006.

[25] S. Hahmann, B. Sauvage, and G. Bonneau, "Area preserving deformation of multiresolution curves," *Computer Aided Geometric Design*, vol. 22, no. 4, pp. 349–367, 2005.

[26] A. A. Hamid, M. Ahmed, and Y. Helmy, "Enhanced progressive vector data transmission for Mobile Geographic Information Systems (MGIS)," in *Innovations and Advances in Computer Sciences and Engineering*, T. Sobh, Ed. Springer Netherlands, 2010, pp. 61–66.

[27] F. J. Harvey, *A Primer of GIS: Fundamental Geographic and Cartographic Concepts*. Guilford Press, 2008.

[28] J. Hershberger and J. Snoeyink, "An $\mathcal{O}(n \log n)$ Implementation of the Douglas-Peucker Algorithm for Line Simplification," in *Proceedings of the 10th Annual Symposium on Computational Geometry*. ACM Press, 1994, pp. 383–384.

[29] G. F. Jenks, "Geographic logic in line generalization," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 26, no. 1, pp. 27–42, 1989.

[30] D. W. Kammler, *A First Course in Fourier Analysis*, 1st ed. Prentice-Hall, 2000.

[31] D. R. Kincaid and E. W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, ser. Pure and Applied Undergraduate Texts. American Mathematical Society, 2002.

[32] D. Li, K. Qin, and H. Sun, "Curve modeling with constrained B-spline wavelets," *Computer Aided Geometric Design*, vol. 22, no. 1, pp. 45–56, Jan. 2005.

[33] Z. Li, *Algorithmic Foundation of Multi-scale Spatial Representation*. CRC Press, 2007.

[34] ——, "Digital map generalization at the age of enlightenment: A review of the first forty years," *The Cartographic Journal*, vol. 44, no. 1, pp. 80–93, 2007.

[35] Z. Li and S. Openshaw, "A natural principle for the objective generalization of digital maps," *Cartography and Geographic Information Science*, vol. 20, no. 1, pp. 19–29, 1993.

[36] R. Mazur and H. W. Castner, "Horton's ordering scheme and the generalisation of river networks," *The Cartographic Journal*, vol. 27, no. 2, pp. 104–112, 1990.

[37] D. E. McArthur, R. W. Fuentes, and V. Devaragan, "Generation of hierarchical multiresolution terrain databases using wavelet filtering," *Photogrammetric Engineering & Remote Sensing*, vol. 66, no. 3, p. 287, Mar. 2000.

[38] R. B. McMaster and K. S. Shea, *Generalization in Digital Cartography*, ser. Resource Publications in Geography. Association of American Geographers, 1992.

[39] A. Mertins and D. A. Mertins, *Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications*. New York, NY, USA: John Wiley & Sons, Inc., 1999.

[40] V. M. Merwade, "An automated GIS procedure for delineating river and lake boundaries," *Transactions in GIS*, vol. 11, no. 2, pp. 213–231, 2007.

[41] M. Nöllenburg, D. Merrick, A. Wolff, and M. Benkert, "Morphing polylines: A step towards continuous generalization," *Computers, Environment and Urban Systems*, vol. 32, no. 4, pp. 248–260, 2008.

[42] P. Oosterom, "Research and development in geo-information generalisation and multiple representation," *Computers, Environment and Urban Systems*, vol. 33, no. 5, pp. 303–310, Sep. 2009.

[43] J. Perkal, "An attempt at objective generalization," *Michigan Inter-University Community of Mathematical Geographers*, 1965.

[44] C. Plazanet, J. G. Affholder, and E. Fritsch, "The importance of geometric modeling in linear feature generalization," *Cartography and Geographic Information Science*, no. 4, pp. 291–305, 1995.

[45] P. M. Prenter, *Splines and Variational Methods*, ser. Wiley Classics Library. Wiley, 1989.

[46] W. H. Press, *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, ser. FORTRAN Numerical Recipes. Cambridge University Press, 1992.

[47] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, vol. 1, pp. 244–256, 1972.

[48] J. Ramos, C. Esperança, and E. Clua, "A progressive vector map browser for the web," *Journal of the Brazilian Computer Society*, vol. 15, no. 1, pp. 35–48, 2009.

[49] P. Raposo, "Scale-Specified Automated Map Line Simplification by Vertex Clustering on a Hexagonal Tessellation," Master's thesis, The Pennsylvania State University, 2011.

[50] P. L. Rosin, "Non-parametric multi-scale curve smoothing," *Proc SPIE Conf Applications of Artificial Intelligence XI Machine Vision and Robotics*, vol. 1964, no. 6, pp. 66–77, 1993.

[51] E. Saux, "B-spline curve fitting: Application to cartographic generalization of maritime lines," in *8th International Conference on Computer Graphics and Visualization (GraphiCon'98)*, 1998, pp. 196–203.

[52] M. Sester and C. Brenner, "A vocabulary for a multiscale process description for fast transmission and continuous visualization of spatial data," *Computers & Geosciences*, vol. 35, no. 11, pp. 2177–2184, 2009.

[53] W. Shi and C. Cheung, "Performance evaluation of line simplification algorithms for vector generalization," *The Cartographic Journal*, vol. 43, no. 1, pp. 27–44, 2006.

[54] H. Shu, C. Qi, and C. Li, "Wavelet-based line simplification," in *Geoinformatics 2006: Geospatial Information Science*, J. Gong and J. Zhang, Eds., vol. 6420, no. 1. SPIE, 2006, pp. 64 200W–1.

[55] L. Stanislawski and S. Savino, "Pruning of hydrographic networks: A comparison of two approaches," in *14th ICA/ISPRS Workshop on Generalisation and Multiple Representation*, Paris, 2011.

[56] L. V. Stanislawski, "Feature pruning by upstream drainage area to support automated generalization of the United States National Hydrography Dataset," *Computers, Environment and Urban Systems*, vol. 33, no. 5, pp. 325–333, Sep. 2009.

[57] H. G. Stark, *Wavelets and Signal Processing: An Application-Based Introduction*. Springer, 2005.

[58] State of Utah. 2009 National Agriculture Imagery Program (NAIP) 1 Meter Orthophotography. [Online]. Available: http://gis.utah.gov/

[59] A. N. Strahler, "Quantitative analysis of watershed geomorphology," *American Geophysical Union Transactions*, vol. 38(6), pp. 912–920, 1957.

[60] N. Tate, C. Brunsdon, M. Charlton, A. Fotheringham, and C. Jarvis, "Smoothing/filtering LiDAR digital surface models. Experiments with LOESS regression and discrete wavelets," *Journal of Geographical Systems*, vol. 7, no. 3, pp. 273–290, 2005.

[61] K. Thapa, "Critical points detection and automatic line generalisation in raster data using zero-crossings," *The Cartographic Journal*, vol. 25, no. 1, pp. 58–68, 1988.

[62] F. Töpfer and W. Pillewizer, "The principles of selection," *The Cartographic Journal*, vol. 3, no. 1, pp. 10–16, 1966.

[63] K. Urban, *Wavelet Methods for Elliptic Partial Differential Equations*. Oxford University Press, 2009.

[64] USGS. (2012) National Hydrography Dataset. [Online]. http://nhd.usgs.gov/.

[65] ——. (2012) The National Map Viewer. [Online]. http://nationalmap.gov/.

[66] J. Vaughan, D. Whyatt, and G. Brookes, "A parallel implementation of the Douglas–Peucker line simplification algorithm," *Software Practice and Experience*, vol. 21, no. 3, pp. 331–336, 1991.

[67] W. Wang and Y. Zhang, "Wavelets-based NURBS simplification and fairing," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 5-8, pp. 290–300, 2010.

[68] J. Wu, "Wavelet-Based Multiresolution Data Representations for Scalable Distributed GIS Services," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.

[69] J. Wu, K. Amaratunga, and R. Chitradon, "Design of a distributed interactive online GIS viewer by wavelets," *Journal of Computing in Civil Engineering*, vol. 16, no. 2, pp. 115–123, 2002.

[70] B. Yang, R. Purves, and R. Weibel, "Efficient transmission of vector data over the Internet," *International Journal of Geographical Information Science*, vol. 21, no. 2, pp. 215–237, 2007.

[71] L. Q. Zhang, C. J. Yang, S. H. Liu, Y. C. Ren, D. L. Liu, and X. P. Rui, "Effective techniques for interactive rendering of global terrain surfaces," *IEEE Geoscience and Remote Sensing Letters*, vol. 2, no. 2, pp. 215–219, Apr. 2005.

[72] L. Zhang, C. Yang, X. Tong, and X. Rui, "Visualization of large spatial data in networking environments," *Computers & Geosciences*, vol. 33, no. 9, pp. 1130–1139, 2007.

Appendix A

Evaluation material

The explanation and analysis of our river network viewer evaluation study is provided in Chapter 7. Figure A.1 shows a screenshot of our river network viewer. It displays buttons that are referred to in the questionnaire. Figures A.2–A.5 show the questionnaire for the study. Figure A.6 shows the lab setup.

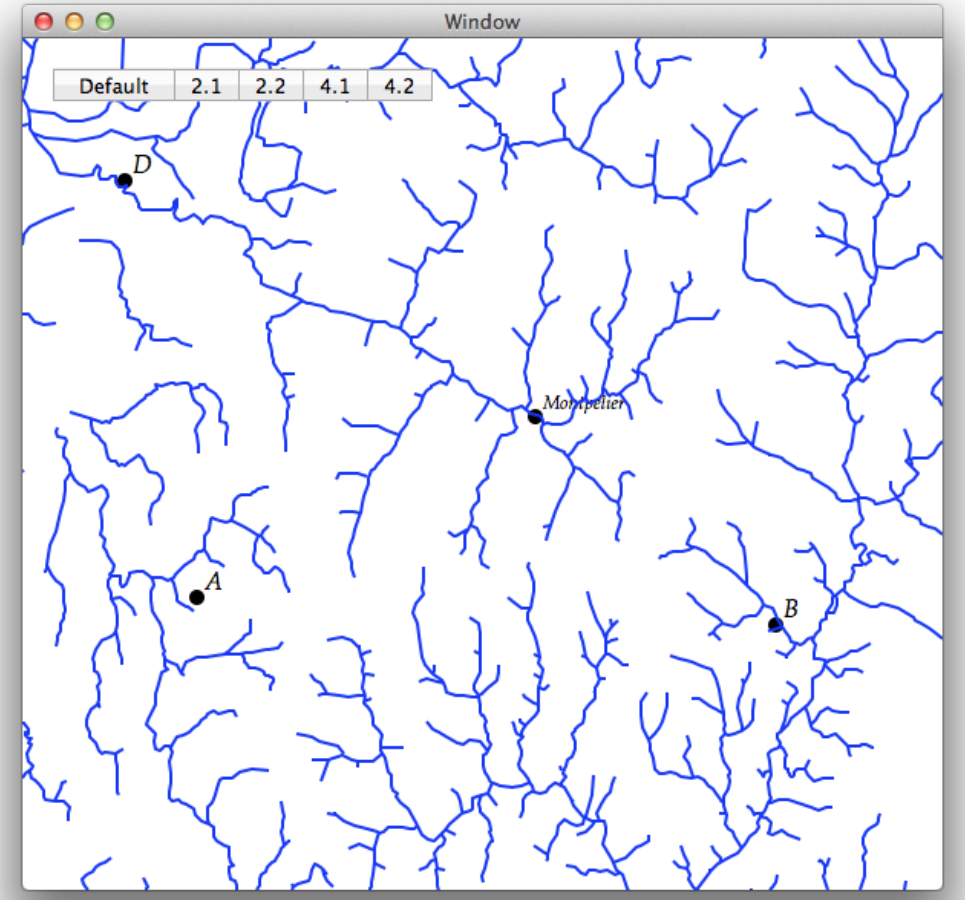


Figure A.1: Screenshot of our viewer

River Network Map Viewer Questionnaire

Participant # _____

Experiment 1

Task 1.1

1. Open the National Map viewer
2. Find Point A (44.050, -73.121) *(latitude, longitude)* by using the search box
3. Zoom in on Point A
4. Find the nearest river to Point A and write down the flow direction:

5. Find Point B (44.018, -72.187) by using the search box
6. Zoom in on Point B
7. Find the nearest river to Point B and write down the flow direction:

8. Find Point C (43.048, -72.374) by using the search box
9. Zoom in on Point C
10. Find the nearest river to Point C and write down the flow direction:

11. Indicate how responsive this viewer is?

○	○	○	○	○
Too Slow	Slow	Didn't notice	Fast Enough	Fast

Figure A.2: Questionnaire – Page 1

Task 1.2

1. Open the Vermont map viewer
2. Find Point A (44.050, -73.121) on the map.
3. Zoom in on Point A
4. Indicate how responsive this viewer is?

○	○	○	○	○
Too Slow	Slow	Didn't notice	Fast Enough	Fast

Experiment 2

Task 2.1

1. Open the Vermont map viewer
2. Find Point B (44.018, -72.187) on the map.
3. Press button 2.1
4. Zoom in and out on Point B
5. Press button 2.2
6. Zoom in and out on Point B again
7. Describe the differences between the two map modes. You may try both ways again.

8. Which mode do you prefer?

○	○	○	○	○
I prefer 2.1	2.1 is a little better than 2.2	I don't prefer one over the other	2.2 is a little better than 2.1	I prefer 2.2

Figure A.3: Questionnaire – Page 2

Experiment 3

Task 3.1

1. Open the Vermont map viewer
2. Find Point C (43.048, -72.374) on the map.
3. Zoom in and out on Point C by using the Mouse Wheel
4. Zoom in and out on Point C by using the Magic Trackpad
5. Describe the differences between the two. You may try both ways again.

6. Which experience do you prefer?

○	○	○	○	○
I prefer the mouse wheel	The mouse wheel is a little better than the trackpad	I don't prefer one over the other	The trackpad is a little better than the mouse wheel	I prefer the trackpad

Experiment 4

Task 4.1

1. Open the Vermont map viewer
2. Find Point D (44.531, -73.237) on the map.
3. Press button "4.1"
4. Zoom in and out on Point D
5. Press button "4.2"
6. Zoom in and out on Point D
7. Describe the differences between the two modes. You may try both ways again.

8. Which mode do you prefer?

○	○	○	○	○
I prefer 4.1	4.1 is a little better than 4.2	I don't prefer one over the other	4.2 is a little better than 4.1	I prefer 4.2

Figure A.4: Questionnaire – Page 3

Summary

1. Which map viewer did you find to be the most responsive?

○	○	○	○	○
The National Map viewer was the most responsive	The National Map viewer was a little more responsive than the Vermont viewer	I don't prefer one over the other	The Vermont viewer was a little more responsive than the National Map viewer	The Vermont viewer was the most responsive

2. Which map viewer's "zoom" feature was easier to use?

○	○	○	○	○
The National Map viewer was easier to zoom	The National Map viewer was a little easier to zoom than the Vermont viewer	I don't prefer one over the other	The Vermont viewer was a little easier to zoom than the National Map viewer	The Vermont viewer was easier to zoom

3. What improvements do you suggest for the Vermont viewer?

4. Do you have any comments?

Figure A.5: Questionnaire – Page 4

Figure A.6: Lab setup for evaluation study

Institutional Review Board for the Protection of Human Subjects

Approval of Initial Submission – Exempt from IRB Review – AP01

Date: February 29, 2012 **IRB#:** 0526

Principal Investigator: Moshe Gutman **Approval Date:** 02/29/2012

Exempt Category: 2 – Educational tests, surveys, interviews, or observation

Study Title: River Network Map Viewer

On behalf of the Institutional Review Board (IRB), I have reviewed the above-referenced research study and determined that it meets the criteria for exemption from IRB review. To view the documents approved for this submission, open this study from the *My Studies* option, go to *Submission History*, go to *Completed Submissions* tab and then click the *Details* icon.

As principal investigator of this research study, you are responsible to:

- Conduct the research study in a manner consistent with the requirements of the IRB and federal regulations 45 CFR 46.
- Request approval from the IRB prior to implementing any/all modifications as changes could affect the exempt status determination.
- Maintain accurate and complete study records for evaluation by the HRPP Quality Improvement Program and, if applicable, inspection by regulatory agencies and/or the study sponsor.

If you have questions about this notification or using iRIS, contact the IRB @ 405-325-8110 or irb@ou.edu.

Cordially,

Lara Mayeux, Ph.D.

Vice Chair, Institutional Review Board

Figure A.7: Institutional Review Board Approval Letter