TopoLines: Topological Smoothing for Line Charts

Ashley Suh* University of South Florida
Christopher Salgado† University of South Florida
Mustafa Hajij‡ University of South Florida
Paul Rosen§ University of South Florida

ABSTRACT

We present TopoLines, a method for smoothing line charts by leveraging techniques from Topological Data Analysis. Common methods used to smooth line charts, such as rank filters, convolutional filters, frequency domain filters, and subsamples, optimize on various properties of the data—many of which are not an ideal representation for people. We hypothesize TopoLines to be more visually accurate to the original graph when compared to conventional filtering techniques. Our approach uses a merge tree to capture the most topologically significant events of a scalar function and remove noise with a user-controlled simplification. In this paper, we briefly discuss and compare these frequently used filtering methods and TopoLines.

Keywords: Topological Data Analysis, Information Visualization, user studies, graph drawing.

1 INTRODUCTION

Line Charts are commonly used today for a variety of applications: visualizing stock trends, mapping weather changes, radio wave detection, etc. While significant increases in data availability allow users to create large-scale graphs, the ability to effectively represent scalar functions under varying simplification thresholds can result in the loss of key features.

Commonly used filtering methods can lack an intuitiveness when smoothing noise. For example, subsampling by removing every other point in a dataset results in unstable simplification, while others may ignore key features—such as important critical points—dependent on the simplification threshold and technique.

Applications of Topological Data Analysis (TDA) have often been used to reduce noise in data while capturing the most significant features. We propose a method for leveraging the noise reduction capabilities of TDA as a substitute for current filtering methods. We use merge trees as a model for capturing features and smoothing line charts.

Prior work has been done in the study of merge trees as a topological simplification tool [2], but we are not aware of work done as an application of smoothing for line charts.

2 FILTERING METHODS FOR LINE CHARTS

In this section we discuss four different classes for filtering a signal. Each of these classes preserves particular properties of the input in the simplified output. Each offers adjustable simplification parameters, which are approach dependent.

2.1 Rank Filters

Rank filters are nonlinear filters that order values within a neighborhood window and select a value from that set. The *Median Filter* takes the neighbor and selects the median value. The level of smoothing can be increased or decreased by enlarging or shrinking the window, respectively.

2.2 Convolutional Filters

Convolutional are a stencil based method, where a series of weights are applied to the neighborhood of values. The *Gaussian Filter* is commonly used in convolutional signal and image processing [4]. It weights the input function using a standard Gaussian distribution. The user may increase or decrease the simplification level by adjusting the standard deviation of the Gaussian.

2.3 Frequency Domain Filters

Frequency domain filtering converts the scalar data into a frequency domain representation, via wavelets or Fourier transform. Our approach uses a bandpass or, more specifically, a *low-pass filter*. The approach uses a Discrete Fourier Transform and zeros out high frequency values before converting back. Increasing or decreasing the level of smoothing is done by modifying the range of frequencies, which are zeroed out.

2.4 Subsampling

Likely the most common choice, due to its ease of implementation, *subsampling* simply selects a subset of input points. Between selected points, interpolation is used. We used regular subsampling and linear interpolation. Simplification is increased by sampling fewer points.

3 TOPOLINES: TOPOLOGICALLY SMOOTHED LINES

3.1 Merge Tree

3.2 Branch Decomposition

To extract the topological features of a scalar function for our simplification process, we use techniques from Topological Data Analysis, specifically the merge tree [3].

Our process is shown in Figure 1. Given an initial curve (Figure 1(a)), a merge tree is calculated on the set of points used to construct the line (Figure 1(b)). Critical points are paired using branch decomposition [5]. Each pair of critical points is assigned a value called *persistence*, which is the difference in value between the 2 critical points. This value is used for choosing which critical points to remove.

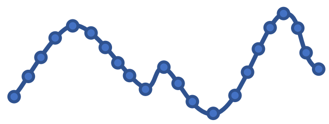
The user specifies a threshold, and all critical points with persistence below that value are removed from the tree. The signal is reconstructed by using isotonic regression [1] between all remaining critical points. Isotonic regression finds a curve which minimizes error in a least-squared sense.

The isotonic regression does not ensure C^1-continuity. Therefore, the points of the isotonic regression are smoothed using a Gaussian filter.

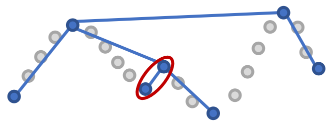
4 RESULTS

To analyze the effectiveness of TopoLines, we numerically evaluate line errors produced by TopoLines, subsampling, low-pass filter, median filter, and Gaussian filters. To compare simplifications, we compute the function $g(x) = f(x) - f'(x)$, where $f(x)$ is the input

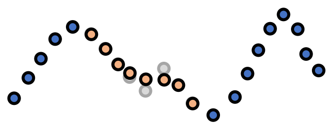
*e-mail: asuh@mail.usf.edu
†e-mail: csalgado1@mail.usf.edu
‡e-mail: mhajij@usf.edu
§e-mail: prosen@usf.edu



(a) Example Line Chart



(b) Merge Tree on Line with critical points to remove selected in red.



(c) Resulting point set after isotonic regression and smoothing.

Figure 1: Smoothing process for an input signal.

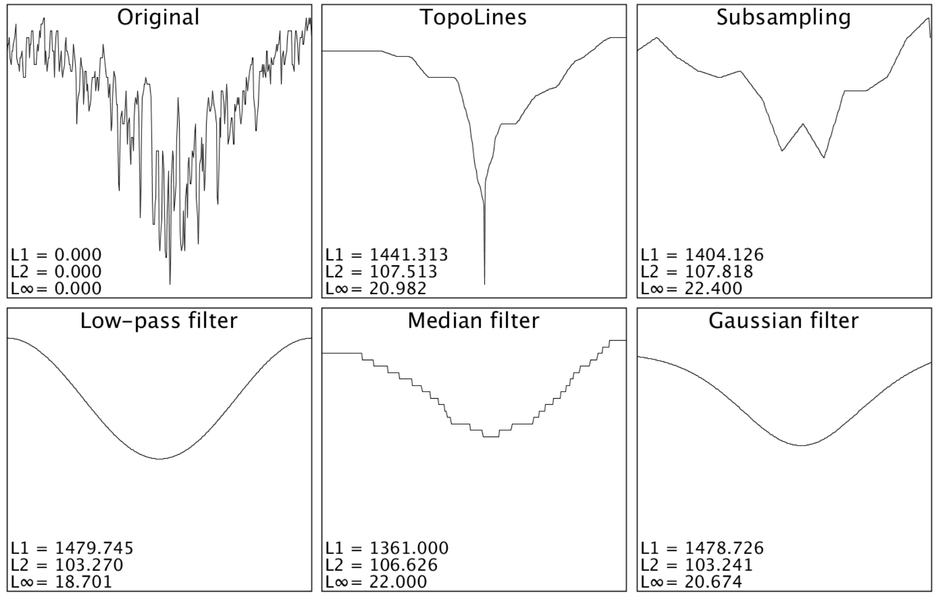


Figure 2: Climate dataset [6] with similar error levels. TopoLines preserves the global minimum as the highest persistence feature.

function and $f'(x)$ is the output. Next, we measure each l^1,l^2, and l^∞ norms. Each filtering method has an associated set of errors, which can be increased or decreased dependent on the simplification threshold provided by the user.

The l^1 error calculates the summation of errors all absolute values $l^1 = \sum g(x)$. The l^2 error measures the Euclidean Distance between the original scalar function and the simplified signal, $l^2 = \sqrt{\sum g(x)^2}$. Finally, the l^∞ error returns the absolute maximum over all $l^\infty = \max |g(x)|$. When choosing a simplification threshold we attempt to match errors between methods.

4.1 Visual Analysis

We provide examples of two datasets simplified at similar error measurements.

We hypothesize that TopoLines provides a visually accurate representation of the original data at any error interval. To test this hypothesis, we choose varying error levels between the five methods and provide an example of two line charts to compare against the original graph.

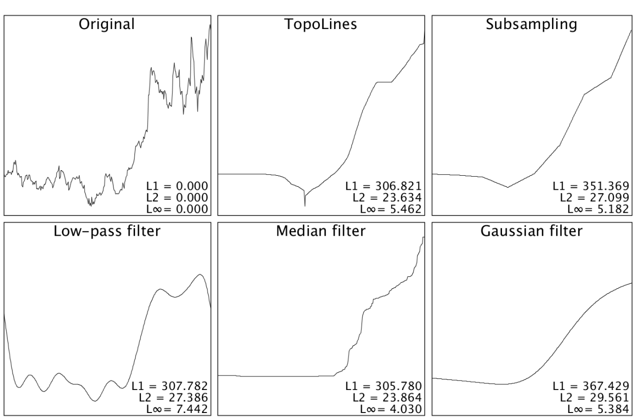


Figure 3: Intel stock dataset, from Google Finance, after smoothing.

One of the most compelling advantages of this approach is the ability to preserve large spikey features during the simplification process over the field. In both examples, shown in Figure 2 and Figure 3, TopoLines retains the global minimum while smoothing low persistence noise.

5 Discussion

While our claim that TopoLines provides a more accurate representation for the smoothing of line charts, we believe a user study is both necessary and helpful in understanding how users interpret data at the intersection of scalar functions. In addition to providing a numerical analysis of TopoLines, we will also conduct a user study through Amazon's Mechanical Turk to measure users' preference for TopoLines when compared to other filtering techniques.

Acknowledgments

This work was supported in part by a grant from National Science Foundation (IIS-1513616).

References

[1] R. E. Barlow. Statistical inference under order restrictions; the theory and application of isotonic regression. Technical report, 1972.

[2] K. Beketayev, D. Yeliussizov, D. Morozov, G. H. Weber, and B. Hamann. Measuring the distance between merge trees.

[3] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry: Theory and Applications*, 24(3):75–94, 2003.

[4] S. K. Kopparapu and M. Satish. Identifying optimal gaussian filter for gaussian noise removal. In *2011 Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*, pp. 126–129, Dec 2011. doi: 10.1109/NCVPRIPG.2011.34

[5] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *Proc. IASTED Conference on Visualization, Imaging, and Image Processing*, pp. 452–290. Citeseer, 2004.

[6] A. H. Young, K. R. Knapp, A. Inamdar, W. Hankins, and W. B. Rossow. The international satellite cloud climatology project h-series climate data record product. *Earth System Science Data*, 10(1):583593, 2018. doi: 10.5194/essd-10-583-2018