Intelligent Data Analysis 6 (2004) 1–11 IDA185 1
IOS Press

Topology and intelligent data analysis

V. Robins[a], J. Abernethy[b], N. Rooney[b] and E. Bradley[b,*]
[a]*Department of Applied Mathematics, Research School of Physical Sciences, The Australian National University, Canberra, Australia*
[b]*Department of Computer Science, University of Colorado, Boulder, CO, USA*

Received 15 November 2003
Revised 31 December 2003
Accepted March 2004

Abstract. A broad range of mathematical techniques, ranging from statistics to fuzzy logic, have been used to great advantage in intelligent data analysis. Topology – the fundamental mathematics of shape – has to date been conspicuously absent from this repertoire. This paper shows how topology, properly reformulated for a finite-precision world, can be useful in intelligent data analysis tasks.

1. Introduction

Topology is the fundamental descriptive machinery for shape. Putting its ideas into real-world practice, however, is somewhat problematic, as traditional topology is an infinite-precision notion, and real data are both limited in extent and quantized in space and time. The field of *computational topology* grew out of this challenge [5,7]. Among the formalisms in this field is the notion of *variable-resolution topology*, where one analyzes the properties of the data – e.g., the number of components and holes, and their sizes – at a variety of different precisions, and then deduces the topology from the limiting behavior of those curves. This framework, which was developed by one of the authors of this paper (Robins) [18–20], turns out to be an ideal tool for intelligent data analysis.

Our approach to assessing connectedness and components in a data set has its roots in the work of Cantor. We define two points as *epsilon* (ε) *connected* if there is an ε-chain joining them; all points in an *ε-connected set* can be linked by an ε-chain. For the purposes of this work, we use several of the fundamental quantities introduced in [19]: the number $C(\varepsilon)$ and maximum diameter $D(\varepsilon)$ of the ε-connected components in a set, as well as the number $I(\varepsilon)$ of ε-isolated points – ε-components that consist of a single point. As demonstrated in [18,20], one can compute all three quantities for a *range* of ε values and deduce the topological properties of the underlying set from their limiting behavior. If the underlying set is connected, the behavior of C and D is easy to understand. When ε is large, all points in the set are ε-connected and thus it has one ε-component ($C(\varepsilon) = 1$) whose diameter $D(\varepsilon)$ is the maximum diameter of the set. This situation persists until ε shrinks to the *largest* interpoint spacing,

*Corresponding author: Elizabeth Bradley, University of Colorado, Department of Computer Science, Boulder, CO 80309-0430, USA. Tel.: +1 303 492 5355; Fax: +1 303 492 2844; E-mail: Elizabeth.Bradley@Colorado.EDU.

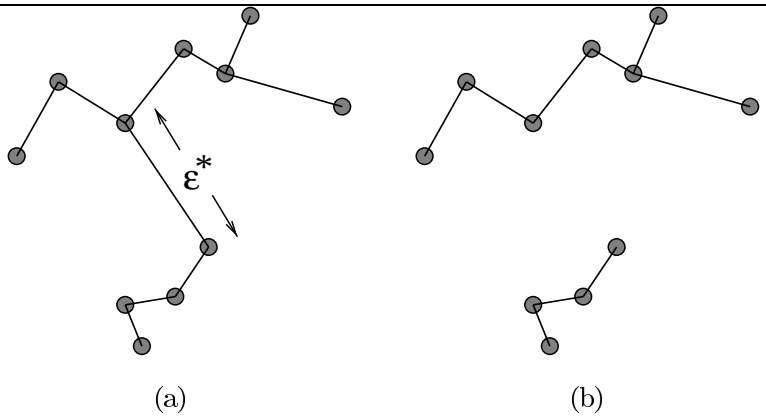


Fig. 1. Computing connectedness: (a) The *minimal spanning tree* (MST) whose edges connect nearest neighbors in a data set. (b) This set contains one ε-connected component if $\varepsilon > \varepsilon^*$; if ε is slightly less than ε^*, the number of ε-connected components is two, and so on.

at which point $C(\varepsilon)$ jumps to two and $D(\varepsilon)$ shrinks to the larger of the diameters of the two subsets, and so on.

When ε reaches the *smallest* interpoint spacing, every point is an ε-connected component, $C(\varepsilon) = I(\varepsilon)$ is the number of points in the data set, and $D(\varepsilon)$ is zero. If the underlying set is a disconnected fractal, the behavior is similar, except that C and D exhibit a stair-step behavior with changing ε because of the scaling of the gaps in the data. When ε reaches the largest gap size in the middle-third Cantor set, for instance, $C(\varepsilon)$ will double and $D(\varepsilon)$ will shrink by 1/3; this scaling will repeat when ε reaches the next-smallest gap size, and so on. All of these results are derived, explained, and demonstrated in detail in [19].

Our computer implementation of these connectedness calculations relies on constructs from discrete geometry: the minimal spanning tree (MST) and the nearest neighbor graph (NNG). The former is the tree of minimum total branch length that spans the data; see Fig. 1(a) for an example. To construct the MST, one starts with any point in the set and its nearest neighbor, adds the closest point to this pair, and repeats until all points are in the tree. This is Prim's algorithm which more formally, grows the MST by adding at each stage an edge (x, y) and vertex y to the tree if (x, y) is minimal among all edges where x is in the tree and y is not. The NNG is a directed graph that has an edge from x_A to x_B if x_B is the nearest neighbor of x_A. To construct it, one starts with the MST and keeps the shortest edge emanating from each point. Both algorithms may be easily implemented in R^d; the computational complexity of the MST is $O(N^2)$ in general and $O(N \log N)$ in the plane, where N is the number of data points. To compute C and I from these graphs, one simply counts edges. $C(\varepsilon)$, for example, is one more than the number of MST edges that are longer than ε, and $I(\varepsilon)$ is the number of NNG edges that are longer than ε. Diameters $D(\varepsilon)$ of ε components are then found using standard computational geometry techniques. Note that one must count NNG edges with multiplicity, since x_A being x_B's nearest neighbor does not imply that x_B is x_A's nearest neighbor (i.e., if a third point x_C is even closer to x_A). Note, too, that the MST and NNG need only be constructed once; all of the C and I information for different εs is captured in their edge lengths. Finally, to identify ε-components, one simply removes edges that are longer than ε.

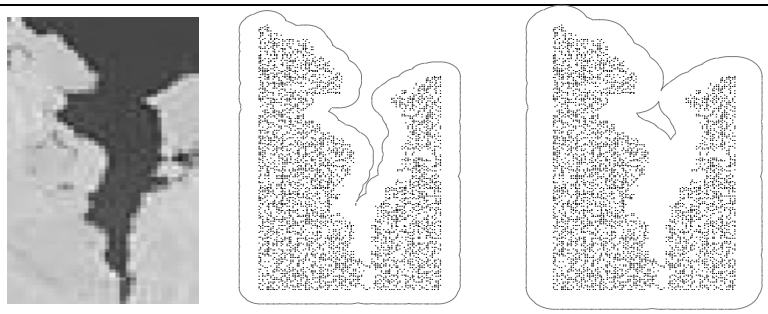
These trees and the information encoded in their edges are extremely useful in intelligent data analysis. A vertical jump in $C(\varepsilon)$, for instance, occurs at an ε value that corresponds to the size of a gap in the dataset. $D(\varepsilon)$ is of obvious utility in describing the sizes of objects, and $I(\varepsilon)$ can be used to filter out

noise. All of these quantities mesh well with experimental reality: the $C(\varepsilon)$ plot for a satellite image of the Arctic ice pack, for instance, captures exactly how many ice floes will be resolved if the instrument has a precision of 10 m, 5 m, 1 m, etc. The examples in Section 2 expand upon some of these ideas. MSTs can also be used to aggregate points into structures, and so they have been widely used in the kinds of clustering tasks that arise in pattern recognition [8] and computer vision [4]. Their branching structure can also be exploited – e.g., to identify orbit types in dynamical systems [25] or to find discontinuities in bubble-chamber tracks [26]. Clustering and coherent-structure extraction are not the only applications of the MST; additive noise creates small transverse 'hairs' on these trees, and so filtering out those points is simply a matter of pruning the associated edges. Section 3 covers this idea in more depth.

Another fundamental topological property is the number, size, and shape of the *holes* in an object. One way to characterize this is via homology. This branch of topology describes structure using algebraic groups that reflect the connectivity of an object in each dimension. The rank of each homology group is called the *Betti number*, b_k. The zeroth-order Betti number, b_0, counts the number of connected components. Geometric interpretations of the other Betti numbers depend on the space the object lives in: in 2D, b_1 is the number of holes, while in 3D b_1 is (roughly speaking) the number of tunnels and b_2 is the number of enclosed voids. See our website [1] for some images that make these definitions more concrete. The definition of the homology groups requires a discrete geometric representation of the object, e.g., a triangulation of subsets of 2D, and simplicial complexes in three or more dimensions. See Munkres [13] for further details.

In order to put these hole-related ideas into computational practice, we use the α-shape algorithm developed by Edelsbrunner [10]. The basic idea is to "fatten" the data by forming its α-neighborhood (a union of balls of radius α centered at each data point). When α is large (on the order of the diameter of the data) the α-neighborhood is a single, convex connected blob. As α decreases, the α-neighborhood shrinks and more shape detail is resolved. When α is just small enough that a ball of radius α can fit inside the data without enclosing any data points, a hole is created in the α-neighborhood. For very small α, the individual data points are resolved. The implementation of the α-shape algorithm is based on the Voronoi diagram and its dual Delaunay triangulation, commonly used constructions from computational geometry. The Betti numbers are computed from triangulations that capture the topology of the coarse-grained data at the different α-resolutions. These calculations of the Betti numbers are non-trivial in general, but fast algorithms have been developed for α-shapes of 2D and 3D data [6], and associated software is available on the world-wide web [2].

As in the connectedness discussion above, one can compute the number of holes in an α-neighborhood of a data set while varying α, and then use that information to deduce the topological properties of the underlying set [17,18]. There is one important difference, however. The *geometry* of the set can create holes in the α-neighborhoods, even if the set contains no holes. This effect is demonstrated in Fig. 2. Mathematically, this problem can be resolved by incorporating information about how the set maps inside its α-neighborhood. This leads to the definition of a *persistent Betti number*, which was introduced in [17]: for $\varepsilon < \alpha, b_k(\varepsilon, \alpha)$ is the number of holes in the α-neighborhood for which there are corresponding holes in the ε-neighborhood (equivalently, the number of holes in the ε-neighborhood that do not get filled in by forming the fatter α-neighborhood). These persistent Betti numbers are well defined for sequences of complexes that provide successively better approximations to a manifold [17] and are computable using linear algebra techniques. Recently, Edelsbrunner and collaborators have made a similar definition of persistent Betti number specifically for α-shapes, and devised an incremental algorithm for their quantity [9].



(a) (b) (c)

Fig. 2. Computing holes: α-neighborhoods can be used to detect whether the Arctic icepack image in part (a) contains a bay, and, if so, how wide its mouth is. (b) and (c) show α-neighborhoods of the set of white (ice) pixels in (a) for a few interesting α values.

While the non-persistent holes effect makes it difficult to make a correct diagnosis of the underlying topology, it has important implications for intelligent data analysis because it gives us geometric information about the embedding of the set in the plane. This can be very useful in the context of coherent structure extraction. Consider a narrow bay in an icepack, as shown in Fig. 2.

In this case, $b_1(\alpha)$ computed for the set of white (ice) pixels would be zero for α smaller than half the width of the bay, zero for α larger than the largest radius of its interior, and one in between – that is, where an α ball fits inside the bay, but not through its mouth. Note that the α value where the spurious hole first appears is exactly the half-width of the entrance to the bay. If there were a *hole* in the ice, rather than a bay, $b_1(\alpha)$ would be a step function: zero when α is greater than the largest radius of the hole and one when it is less. This technique for finding and characterizing coherent structures whose defining properties involve holes and gaps of various shapes – ponds, isthmuses, channels, tunnels, etc. – is potentially quite useful in intelligent data analysis.

Note that the α-shapes algorithm was written for floating-point data, not digital-image data. We emphasize that the variable-resolution topology measures are not dependent on the numerical context. More efficient implementations for pixel- and voxel-based images need to be developed. The examples following in Section 2 form a proof-of-concept.

2. Coherent structure extraction

In this section, we present three examples that demonstrate how to use our variable-resolution topology techniques to find coherent structures in aerial images of sea ice.[1] Scientists look for several things in these kinds of images: open water or "leads," ice floes, and melt ponds, all of which are visible in Fig. 3. Climatology studies that track the seasonal evolution of these coherent structures are a major current thrust of Arctic science, but they require an automated way of analyzing large numbers of images. Traditional image-processing and machine-learning tools can help with tasks like this, to a point – e.g., edge-finding, contrast enhancement, etc. – but topology-based methods are an even better solution. The following paragraphs treat three examples in detail: finding a lead through an icepack, distinguishing regions of different ice/water concentration, and studying how the number and size of melt ponds are distributed in sea ice.

[1] courtesy of D. Perovich from CRREL.

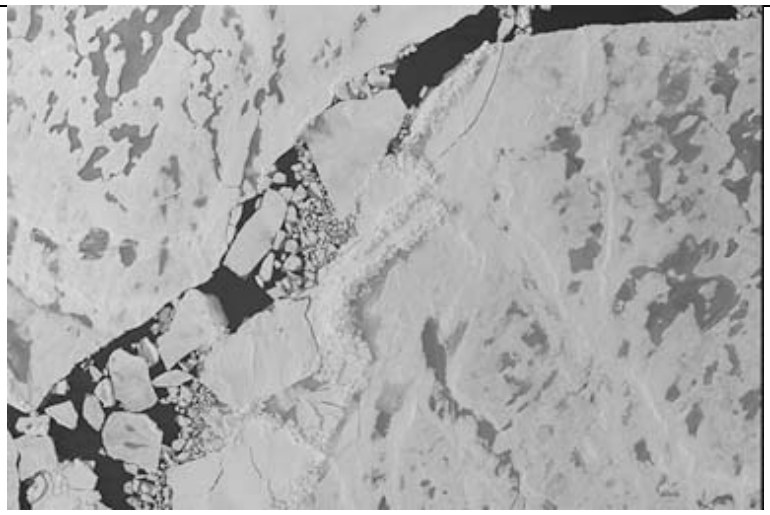


Fig. 3. The arctic ice pack is made up of open water, ice, and melt ponds, which image as black, white, and grey, respectively. Studying the climatology of this system – the seasonal evolution of the different kinds of *coherent structures* – is a major current thrust of Arctic science.

The size, shape, and position of floes and leads is of obvious interest to travelers in the polar regions. Finding a path for a ship through a complicated region of the ice pack, in particular, is a common practical problem, and the variable-resolution computational topology techniques described in Section 1 are quite useful in doing so. Consider a 20 m-wide ship that must traverse the patch of ocean depicted in Fig. 4(a), from left to right. Solving this problem requires assessing the holes in the ice, so we threshold the data and discard the dark water pixels, then apply the α-shape algorithm with a radius $\alpha = 10$. The results, shown in Fig. 4(b), identify all regions that are at least 20 m wide. The next step is to use standard computational geometry techniques [16] on these regions to ascertain which ones touch both sides of the image and then determine which is the shortest.[2] Parts (c) and (d) of the Figure show the effects of raising α beyond 10 m, demonstrating the relationship between the channel shape and the holes in the α-neighborhood – e.g., successively resolving the various tight spots, or identifying regions of water that are at least n meters from ice.

More generally, this variable-resolution analysis can be used to assess the mix of water and ice in a region – not just the relative area fractions (which can be done by counting black and white pixels), but also the distribution of sizes of the regions involved – or to find and characterize regions of different ice concentration. Depending on the task at hand, one can analyze either the water pixels (where α-holes are α-size ice floes) or the ice pixels, where holes are leads. When the ice is close to solid, as in Fig. 5(a), the narrow water channel is resolved as a small α-hole in the set of ice pixels, similar to the lead in the previous paragraph. When the image contains large areas of open water and a few floes, as in part (c), an α-shape analysis of the water pixels resolves a few large holes over a wide range of αs – after a band of smaller αs where the α-neighborhoods are filling in the various gaps and bumps in the floes and creating spurious holes, as discussed in conjunction with Figs 2 and 4. The middle image is more interesting; the wide distribution of floe sizes translates to a wide α range where a small change in that parameter

[2]Note that this development assumes a circular ship; for oblong ships, one should use an α value that is the largest chord of the hull.

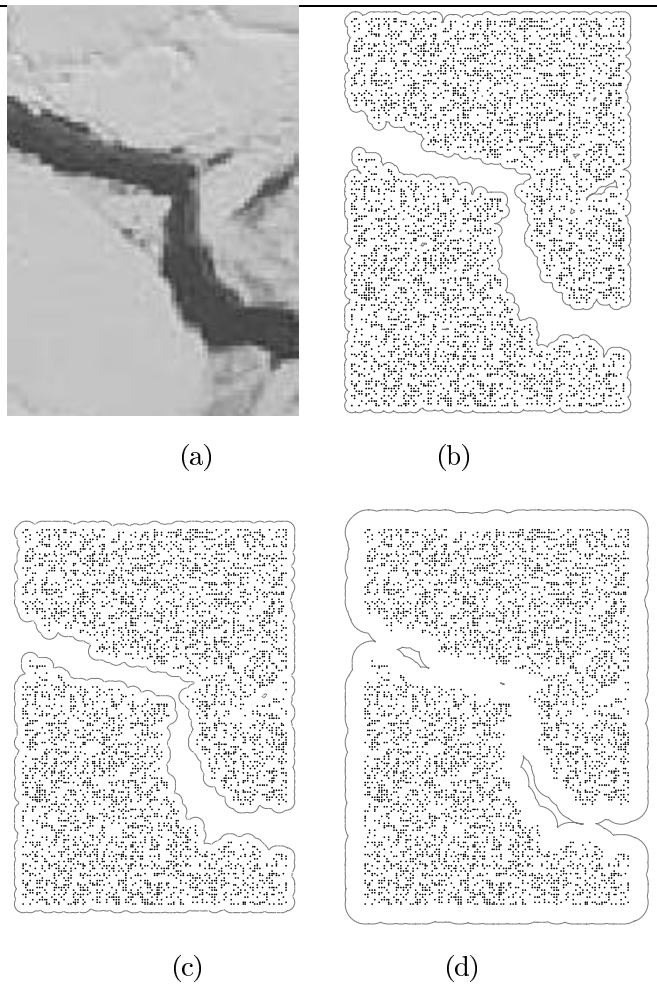
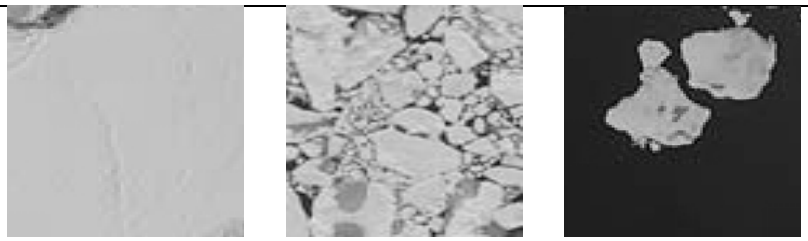


Fig. 4. Using α-shapes to find a path through an icepack: (b), (c), and (d) show three different α-neighborhoods of the set of ice pixels in (a).

resolves a few new holes in the set of water pixels (i.e., floes). All of these results accurately reflect important properties of the data.

The temporal evolution of the albedo of Arctic sea ice is of great importance to climate modelers because of its role in heat transfer. A key factor governing the sea-ice albedo in summer is the size distribution of the melt ponds, which appears to take the form of a power law[15]. We can easily automate these distribution calculations using α-shapes. The first step is to threshold the data so that the 'nonpond' points (both the ice and the water) are black; the second is to look for holes in that data. The results, shown in Fig. 6(a), corroborate the power-law distribution hypothesis quite nicely; after an initial region below about $\log \alpha = 0.3$, where changes in this parameter are resolving promontories and bays in the melt ponds, the log-log plot is fairly linear, with a least squares fit to the slope giving -1.7. Part (b) reproduces the corresponding figure from [15] for comparison. In that paper the slope was estimated as $-3/2$, which is within around 10% of our result. We can obtain similar curves using the connectedness tools described in Section 1: e.g., computing a minimal spanning tree of the 'pond' points in the image, as in Fig. 6(c), and then extracting the connected components and computing their



(a) (b) (c)

Fig. 5. Using α-shapes to distinguish different ice/water distributions: (a) almost-solid ice (b) a lead filled with various-size floes (c) open water with a few big floes. An α-shapes analysis of these images yields a direct measure of the morphology of the ice and water.

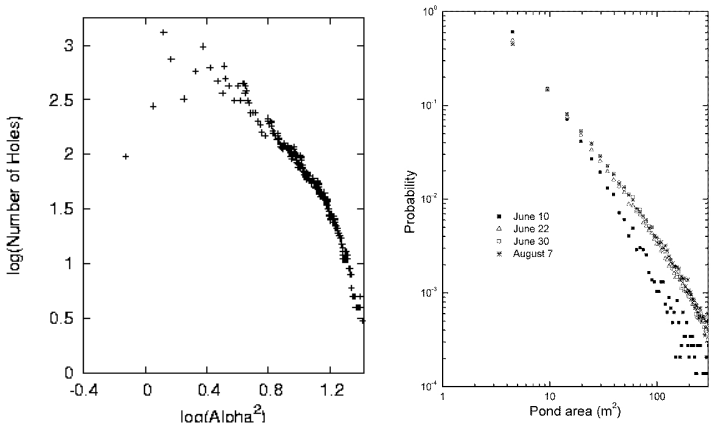
areas and/or diameters.

Our next step will be to combine these notions of multiple-resolution topology with some ideas from spatial statistics and artificial intelligence in order to handle the ill-defined nature of real coherent structures and to recognize the spatial and temporal patterns in which those structures move, evolve, and interact. Spatial statistics are useful in this application because computational topology produces aggregate measures over a whole dataset. Adding a sliding window to the process, and manipulating its geometry intelligently – an approach that we term *topospatial analysis* – allows one to distinguish sub-regions where those measures are different. Artificial intelligence (AI) techniques are useful because coherent structures are not crisp. The Gulf Stream, for instance, is a hot-water jet extending from the eastern seaboard of the US into the Atlantic. It does not have a natural, well-defined boundary. Rather, the temperature at its edges falls off smoothly, and so any threshold-based definition is necessarily somewhat arbitrary. Moreover, coherent structures are much easier to recognize than to describe, let alone define in any formal way. The AI community has developed a variety of representations and techniques for problems like these, and these solutions mesh well with topospatial analysis.

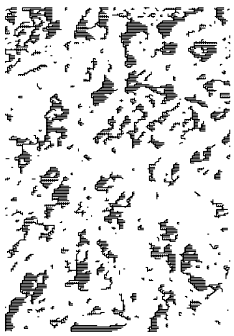
3. Filtering

Real-world data invariably contain noise of one form or another. Traditional linear filters are useful in removing some kinds of noise, but not all. Chaotic behavior, for instance, is both broad band and sensitively dependent on system state, so linear or Fourier-based filters – which simply remove all signal in some band of the power spectrum – can alter important features of the dynamics, and in a significant manner [24]. In cases like this, topology-based filtering can be a much better solution. As mentioned in Section 1, noise often manifests as isolated points (e.g., from an intermittent glitch in a sensor), and variable-resolution topology is an effective way to identify these points. Figure 7 demonstrates the basic ideas.

The spanning tree clearly brings out the displacement of the noisy points from the rest of the orbit, both in direction and in edge length. Specifically, if the noise magnitude is large compared to the inter-point spacing, the edges joining the noisy points to the rest of the tree are longer than the original edges. (Of course, if the magnitude of the noise is *small* compared to that spacing, the associated MST edges will not be unusually long, and the noisy points are not so obvious.) This kind of *separation of scale* often arises when two processes are at work in the data – e.g., signal and noise. Because variable-resolution topology is good at identifying scale separation, we can use it to identify and remove the noisy points.



(a) (b)



(c)

Fig. 6. Computational topology and melt ponds: (a) An α-shapes analysis of the ponds in an icepack image. (b) Corresponding figure from [15] (c) Connected components in melt-pond data.

The first step is to look for a second peak in the edge-length distribution of the MST (equivalently, a shoulder in the plot $I(\varepsilon)$ of the number of isolated points as a function of ε). The maximum edge length of the MST of the *non*-noisy data occurs in the interval between the two peaks (or at the location of the $I(\varepsilon)$ breakpoint). At that value, most of the noisy points – and few of the regular points – are ε-isolated. The next step is to identify and remove all MST edges that exceed that length. Note that an ε-isolated noisy point will lose all its connections to the MST and the edge-pruning procedure will successfully remove the point. If the noisy point was a terminal node of the MST, the rest of the tree structure is unaffected. A slight complication arises if a long edge creates a "bridge" between relevant subsections of the tree. In this case, deleting points connected to the tree by the identified MST edge will discard useful data.[3] Our approach in this case is simply to remove the edge and work with a disconnected version of the tree. This works well, since the quantity of interest in our filtering algorithm is $I(\varepsilon)$, not $C(\varepsilon)$.

Part (c) of the Figure shows the results; in this case, the topology-based filter removed 534 of the 545

[3]We found that this situation rarely occurs in practice if the sampling rate of the data is sufficient.

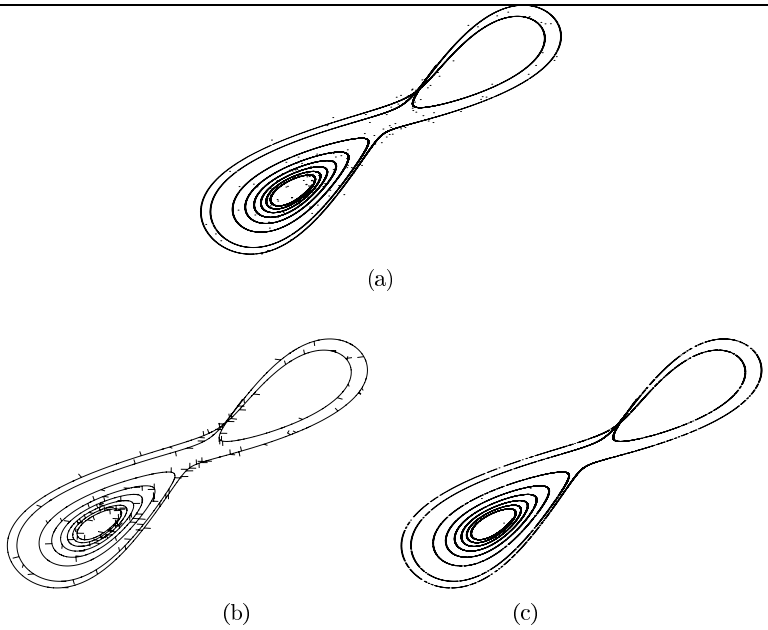


Fig. 7. The effects of noise upon data topology. The dataset in part (a) is a canonical example in dynamical systems – the Lorenz attractor – each point of which was perturbed, with probability 0.01, by a small amount of additive noise. The MST in part (b) clearly shows these noisy points. Part (c) shows the same data set after processing with a topology-based filter that prunes the 'hairs' off the MST.

noisy points and 150 of the 7856 non-noisy points. This translates to 98.0% success with a 1.9% false positive rate. These are promising results – better than any linear filter, and comparable to or better than the noise-reduction techniques used by the dynamical systems community[3]. Increasing the pruning length, as one would predict, decreases the false-positive rate; somewhat less intuitively, though, larger pruning lengths do *not* appear to significantly affect the success rate – until they become comparable to the length scales of the noise. As described in [21], these rates vary slightly for different types and amounts of noise, but remain close to 100% and 0%; even better, the topology-based filter does not disturb the dynamical invariants like the Lyapunov exponent.

While topology-based filtering is very effective, it does come with some caveats. For example, the method simply *removes* noisy points; it does not deduce where each one 'should be' and move it in that direction. There are several obvious solutions to this, all of which add a bit more geometry into the recipe – e.g., averaging the two points on either side of the base of the edge that connects an isolated point to the rest of the trajectory. Moreover, noise does not move points around in raster images; rather, it simply reshades pixels. The computer vision community distinguishes these two kinds of noise as "salt and pepper" and "distortion," respectively. Because the metric used in the MST captures distances between points, it is more effective at detecting the latter than the former.

4. Conclusion

The mathematical framework of variable-resolution topology has tremendous potential for intelligent data analysis. It is, among other things, an efficient way to automate the process of finding and characterizing coherent structures. Figure 6(b), for instance, was constructed by hand-processing roughly

2000 images, which required roughly 50 working days [14]; Figure 6(a) required only a few minutes of CPU time. Because it reveals separation of scale, variable-resolution topology can also be useful in noise removal. In a series of numerical experiments, a topology-based filter removed close to 100% of the noisy points from dynamical system data sets, with a false-positive rate of only a few percent. Separation of scale is fundamental to many other processes whose results one might be interested in untangling, so this method is by no means limited to filtering applications – or to dynamical systems.

There have been a few other topology-based approaches to filtering. Mischaikow et al. [12], for example, use algebraic topology to construct a coarse-grained representation of the data. This effectively finesses the noise issue, and thus constitutes a form of filtering. Rather than use algebraic topology to construct a useful coarse-grained representation of the dynamics, our approach uses *geometric* topology to remove noisy points while working in the original space, which allows us to obtain much finer-grained results.

A tremendous volume of papers on techniques for reasoning about the structure of objects has appeared in various subfields of the computer science literature. Very few of these papers focus on distilling out the topology of the coherent structures in the data. Those that do either work at the pixel level (e.g., the work of Rosenfeld and collaborators [22]) or are hand-crafted for a particular application (e.g., using wavelets to find vortices in images of sea surface temperature [11]), and all are constrained to two or three dimensions. Our framework is algorithmically much more general, and it works in arbitrary dimension. Many geographic information systems [23] and computer vision [4] tools incorporate simple topological operations like adjacency or connectedness; these, too, generally only handle 2D gridded data, and none take varying resolution into account – let alone exploit it.

References

[1] http://www.cs.colorado.edu/∼lizb/topology.html.
[2] http://www.alphashapes.org.
[3] H. Abarbanel, *Analysis of Observed Chaotic Data*, Springer, 1995.
[4] D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, 1982.
[5] M. Bern and D. Eppstein, Emerging challenges in computational topology, 1999.
[6] C. Delfinado and H. Edelsbrunner, An incremental algorithm for Betti numbers of simplicial complexes on the 3-sphere, *Computer Aided Geometric Design* **12** (1995), 771–784.
[7] T. Dey, H. Edelsbrunner and S. Guha, Computational topology, in: *Advances in Discrete and Computational Geometry*, B. Chazelle, J. Goodman and R. Pollack, eds, American Math. Society, Princeton, NJ, 1999.
[8] R. Duda and P. Hart, *Pattern Classification*, Wiley, New York, 1973.
[9] H. Edelsbrunner, D. Letscher and A. Zomorodian, Topological persistence and simplification, in: *IEEE Symposium on Foundations of Computer Science*, 2000, pp. 454–463.
[10] H. Edelsbrunner and E. Mücke, Three-dimensional alpha shapes, *ACM Transactions on Graphics* **13** (1994), 43–72.
[11] H. Li, Identification of coherent structure in turbulent shear flow with wavelet correlation analysis, *Transactions of the ASME* **120** (1998), 778–785.
[12] K. Mischaikow, M. Mrozek, J. Reiss and A. Szymczak, Construction of symbolic dynamics from experimental time series, *Physical Review Letters* **82** (1999), 1144–1147.
[13] J. Munkres, *Elements of Algebraic Topology*, Benjamin Cummings, 1984.
[14] D. Perovich, Personal communication.
[15] D. Perovich, W. Tucker and K. Ligett, Aerial observations of the evolution of ice surface conditions during summer, *Journal of Geophysical Research: Oceans* **10.1029/2000JC000449** (2002).
[16] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
[17] V. Robins, Towards computing homology from finite approximations, *Topology Proceedings* **24** (1999), 503–532.
[18] V. Robins, *Computational Topology at Multiple Resolutions*, PhD thesis, University of Colorado, June 2000.
[19] V. Robins, J. Meiss and E. Bradley, Computing connectedness: An exercise in computational topology, *Nonlinearity* **11** (1998), 913–922.

[20] V. Robins, J. Meiss and E. Bradley, Computing connectedness: Disconnectedness and discreteness, *Physica D* **139** (2000), 276–300.
[21] V. Robins, N. Rooney and E. Bradley, Topology-based signal separation, *Chaos*, in review; see www.cs.colorado.edu/∼lizb/publications.html.
[22] P. Saha and A. Rosenfeld, The digital topology of sets of convex voxels, *Graphical Models* **62** (2000), 343–352.
[23] S. Shekhar, M. Coyle, B. Goyal, D. Liu and S. Sarkar, Data models in geographic information systems, *Communications of the ACM* **40** (1997), 103–111.
[24] J. Theiler and S. Eubank, Don't bleach chaotic data, *Chaos* **3** (1993), 771–782.
[25] K. Yip, *KAM: A System for Intelligently Guiding Numerical Experimentation by Computer*, Artificial Intelligence Series. MIT Press, 1991.
[26] C. Zahn, Graph-theoretical methods for detecting and describing Gestalt clusters, *IEEE Transactions on Computers* **C-20** (1971), 68–86.