Topology-Based Signal Separation

V. Robins

Department of Applied Mathematics
Research School of Physical Sciences and Engineering
The Australian National University
ACT 0200 Australia

N. Rooney and E. Bradley

Department of Computer Science
University of Colorado
Boulder, CO 80309-0430 USA

April 1, 2003

In review, Chaos.

Abstract

Traditional noise-filtering techniques are known to significantly alter features of chaotic data. In this paper, we present a noncausal topology-based filtering method that is effective in removing additive, uncorrelated noise from time-series data. Signal-to-noise ratios and Lyapunov exponent estimates are dramatically improved following the removal of the identified noisy points.
Lead paragraph:

Traditional linear or Fourier-based schemes for removing noise are unsuitable for filtering chaotic signals because they remove all modes in some interval of the frequency spectrum. Since chaos is characterized by a broad frequency spectrum, this process inevitably destroys part of the dynamically relevant signal. The dynamical systems community has developed a variety of nonlinear filtering methods that exploit the characteristic state-space geometry of these systems in order to remove noise without disturbing the signal. This paper proposes a nonlinear filtering method that is based on the characteristic topology of an attractor of a continuous-time dynamical system. In particular, such objects are known to be perfect—that is, they contain no isolated points. In practice, the numerical representation of an orbit fails to be perfect because of finite sampling and/or additive noise. We use a variable-resolution topological analysis to find and remove any points. Experiments with clean Lorenz data contaminated with various forms of noise show highly encouraging results: the topology-based filter removes 96-100% of the noisy points, improving the signal-to-noise ratio (SNR) from \( \approx 20dB \) to more than \( 50dB \), with a false-positive rate of 1.4-2.5%. Estimates of the largest Lyapunov exponent in these data are dramatically improved following the removal of the identified noisy points: the noisy data has \( \lambda \)s on the order of 10-100, while \( \lambda \)s of the filtered data are 2-4. Data from a laboratory apparatus—a parametrically forced pendulum—showed equally encouraging results, though of course one cannot quantify percentages or SNR in real-world data. The key feature of the method proposed here is its identification of separation of scale. Since separation of scale is fundamental to many other forms of signal that one might be interested in untangling, this method is by no means limited to dynamical systems—or to filtering applications.
1 Introduction

Removing noise from chaotic data is highly problematic. Chaotic behavior is both broad band and sensitively dependent on system state, so traditional filtering schemes—which simply remove all signal in some band of the power spectrum—can alter important features of the dynamics, and in a significant manner. Many authors have recognized this; see [37] for a good synopsis. A variety of schemes, many of which are reviewed very nicely in Chapter 7 of [1], have been proposed for working around this limitation. A few of these rely on variations of traditional linear filters[20, 26, 28, 29, 35], but the majority use nonlinear approaches. Specific techniques vary, depending on how much is known about the problem at hand, but the basic idea is to exploit the fact that deterministic dynamics evolves upon smooth submanifolds of state space. One family of noise-reduction methods is based on local approximations to these submanifolds. Farmer and Sidorowich[14], for instance, use the stable and unstable manifolds of the dynamical system, via forward- and backward-time simulation and an averaging scheme, to reduce additive noise. Kostelich[18] uses a different geometric property of the dynamics, linearizing around saddle points, where orbits are recurrent. There are many other approaches in this family; see, for example, [16, 19]. Another family of methods projects the noisy vectors onto carefully chosen subspaces, iterating the procedure until the results settle down to what is presumably the true dynamics[8, 9, 33, 36]. If one has measurements of more than one state variable, this process can be streamlined[17]; if one has the full state vectors and also knows the dynamics, one can avoid many of the mathematical gymnastics described above, along with their attendant numerical sensitivities[5, 12]. Probability and statistics can also be useful when one knows the dynamics, as described in [21].

The success of these methods amply demonstrates that the characteristic geometry of the state space of a dynamical system can be a useful basis for a dynamically meaningful noise-filtering scheme. Filtering approaches that rest upon topology can be equally powerful, but have seen far less investigation. State-space attractors (that is, \( \omega \)-limit sets) of continuous-time dynamical systems are, in theory, perfect sets\(^1\). A perfect set is one with no isolated points: \( \text{i.e.} \), no points \( x \) for which there is a \( \delta > 0 \) such that no other point from the set lies in a ball of radius \( \delta \) about \( x \). In practice, the numerical representation of an orbit fails to be perfect because of finite sampling and/or additive noise. If one can work around the sampling effects and still effectively identify the isolated points, then, one can reasonably conclude that those points were noisy—and remove them. This idea is the topic of this paper.

\(^{1}\)since a point in the attractor either lies on an orbit trajectory (and is therefore the limit of nearby points in that trajectory) or is the limit point of a sequence of trajectories.
2 Approach and Evaluation

The broad field of topology is generally concerned with the features of an object that are invariant under deformations that stretch and twist but never tear or glue the object. The ideas and techniques described in this paper are based in the subset of that field that addresses connectedness, the topological concept that captures the notion of continuity of a space. An object—more formally, a subspace—is connected if it cannot be decomposed into two non-empty open sets. Note that these two sets are open in the subspace topology (i.e., the intersection of an open set from the parent space with the subset of points in the subspace). Because experimental data are not infinite in quantity or precision, however, we cannot simply use the traditional topological definitions of connectedness to find and remove isolated points. Rather, we must reformulate those notions, as described in [30, 31, 32], to fit the discrete nature of the underlying space (e.g., the space of floating-point numbers on a computer, or the space of measurements made by a sensor that has one millivolt of precision). The roots of this approach lie in Cantor’s early work, which defines two points as epsilon connected if they are joined by an epsilon chain: a finite sequence of points $x_0 \ldots x_N$ that are separated by distances of epsilon or less: $|x_i - x_{i+1}| < \epsilon$. This provides an easy way to make explicit the finite precision of real data, to formulate useful definitions of topological properties that make sense at variable resolutions, and to deduce the topology of the underlying set from the limiting behavior of those properties.

In this paper, we use several of the fundamental quantities defined in [30, 31, 32], most importantly the number $C(\epsilon)$ of the epsilon-connected components in a set and the number $I(\epsilon)$ of epsilon-isolated points. An epsilon-connected component is a maximal epsilon-connected subset; an epsilon-isolated point is an epsilon-component consisting of a single point. As established in [31], one can compute $C$ and $I$ for a range of epsilon values, and deduce the topological properties of the underlying set—in this case, the true, underlying orbit of the dynamical system—from the patterns in the $C$ and $I$ curves. Figure 1 demonstrates the basic ideas. The point-set data shown in part (a) form a single epsilon-connected component for $\epsilon > \epsilon^*$, where $\epsilon^*$ is the largest inter-point spacing, as shown in Figure 1(b). In this case, $C(\epsilon) = 1$. If $\epsilon$ is slightly less than $\epsilon^*$, $C(\epsilon) = 2$; the corresponding epsilon-components are shown in part (c). As $\epsilon$ shrinks further, successively closer point pairs are resolved, and $C(\epsilon)$ increases, eventually flattening out at $C(\epsilon) = N$ for sufficiently small values of $\epsilon$, where $N$ is the number of points in the data set. The precise manner of that increase depends upon the connectedness properties and fractal dimension of the underlying set, as well as the distribution of data points over the object; this is described briefly in [32] and covered in more depth in [30]. Briefly speaking, $C(\epsilon) = 1$ for $\epsilon > \epsilon^*$ if the underlying set is connected, then rises smoothly and sharply with decreasing $\epsilon$ because smaller $\epsilon$ values allow successively closer point pairs to be resolved. This behavior is shown in Figure 1(d). If the underlying set is a totally disconnected fractal, like the set in part (e) of the Figure, $C(\epsilon)$ rises in a stair-step fashion because of the scaling of the gaps in the set. The number of epsilon-isolated points is closely related to $C$. A point becomes epsilon-isolated when $\epsilon$ decreases past the distance to its nearest neighbor. If the data approximate a perfect set, $I(\epsilon)$ behaves in a similar
Figure 1: Computing connectedness: (a) point-set data and (b) the minimal spanning tree whose edges connect nearest neighbors in that data. If $\epsilon > \epsilon^*$—the largest inter-point gap in the set—all of the points are $\epsilon$-connected (that is, the number of connected components $C = 1$); if $\epsilon$ is slightly less than $\epsilon^*$, the set contains two connected components, as shown in part (c). The behavior of $C$ as a function of $\epsilon$ reflects the topology of the underlying set—the object of which these points are samples. If that set is connected, $C(\epsilon)$ will fall off smoothly with decreasing $\epsilon$, as successively closer point pairs are resolved; see part (d). If the set is a disconnected fractal, as in part (e), $C(\epsilon)$ falls off in a stair-step fashion because of the scaling of the gaps in the data.
manner to $C(\epsilon)$ for small $\epsilon$; in particular $\lim_{\epsilon \to 0} C(\epsilon) = \lim_{\epsilon \to 0} I(\epsilon) = N$, where $N$ is the number of points in the data set. (This result was proved by M. Penrose for the case $N \to \infty$[27].) For larger values of $\epsilon$, a perfect set will have $I(\epsilon) = 0$. Note that $I(\epsilon) \leq C(\epsilon)$ for all $\epsilon$. If $I(\epsilon) = 0$ and $C(\epsilon) > 0$, this implies the existence of some number of distinct connected components in the data.

The computer implementation of these calculations relies on constructs from discrete geometry called 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 Figure 1(b) for an example. To construct the MST, one starts with any point in the set and its nearest neighbor, adds the closest point, and repeats until all points are in the tree\(^2\). The nearest-neighbor graph or 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. Once these graphs are constructed, computing $C$ and $I$ is easy: one simply counts edges. $C(\epsilon)$, for example, is one more than the number of MST edges that are longer than $\epsilon$, and $I(\epsilon)$ is the the number of NNG edges that are longer than $\epsilon$. 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 $\epsilon$s is captured in their edge lengths.

These reformulations and algorithms allow one to assess the state-space topology of a dynamical system—a fundamental and meaningful property—even though the orbits involved are quantized in space and time by the finite resolution of sensors and computers. Figure 2 shows MST-based connectedness results for the canonical Lorenz system. The $C$ and $I$ curves exhibit the classic smooth, sharp falloff that indicates connectedness and perfectness, respectively. This behavior is affected by the sampling: the falloff is less sharp for data that are sparsely sampled and/or non-uniform. As explained above, $C(\epsilon)$ and $I(\epsilon)$ results are virtually identical for the connected sets that we study in this paper. The latter carries the information that we want, as we are looking for lapses in perfectness, so we only plot $I(\epsilon)$ hereafter. Incidentally, the MST and NNG constructs contain other information that is useful for the analysis of dynamical systems. Their branching structure can be used to identify orbit types[39] and discontinuities in bubble-chamber tracks[40], as discussed further at the end of this paper, and they are widely used in the kinds of clustering tasks that arise in pattern recognition[13] and computer vision[3].

As mentioned before, continuous-time dynamical system orbits are, in theory, perfect sets, and so any isolated points on such an orbit are an aberration. Noise is one potential cause of this; consider an intermittent glitch in a sensor (or an error in an algorithm)

\(^2\)Our implementation actually employs Prim’s algorithm[11], which is a bit more subtle. It begins with any vertex as the root and grows the MST in stages, 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.
Figure 2: Topology of the Lorenz attractor: (a) trajectory reconstructed via delay-coordinate embedding from the $x$ component of a 9000-step 4th order Runge-Kutta integration of the Lorenz equations, starting from the initial condition $(x, y, z) = (-11, -12, 45)$, with $a = 16$, $r = 50$, $b = 4$, and $\Delta t = 0.0005$. Embedding parameters were $m = 7$ and $\tau = 0.05$; in this 2D rendering, $x(t+\tau)$ is plotted against $x(t)$. The MST of this trajectory, shown in part (b), is visually indistinguishable from the orbit shown in part (a) because of the small time step. The $C(\epsilon)$ and $I(\epsilon)$ plots in parts (c) and (d) indicate that the underlying orbit is connected and perfect.
that adds noise to some subset of points in the orbit. In order to explore this effect, we added noise to the time-series data from Figure 2, redid the embedding, repeated the topological analysis, and observed the effects on the $I(\varepsilon)$ plots. Figure 3 shows a representative set of results. The spanning tree clearly brings out the displacement of the noisy points from the rest of the orbit: if the magnitude of the noise 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, which creates an extra shoulder on the $I$ (and $C$) curves. Of course, if the magnitude of the noise is small compared to that spacing, the associated MST edges will not be unusually long, and so the $I$ and $C$ curves will not have an extra shoulder. Figure 4 demonstrates this, showing $I(\varepsilon)$ plots for different magnitudes and types of noise. In particular, part (a) of the Figure shows the effects of constant noise, where a fixed value ($\pm n$) was added to roughly 1% of the original time-series points. The width of the shoulder increases with $n$, reflecting the wider distribution of MST edge lengths that results when the noise-added points are embedded. Different types of noise—e.g., uniform, where we added some number between $n$ and $m$, for different $[n,m]$, or Gaussian, with different $\sigma$s and $\bar{z}$s—make the shape of the shoulder more irregular, but do not change its general shape. In all cases, the larger the noise, the wider the shoulder. Again, this makes sense: larger noise values push points further from their original position on the true orbit, which means that they will remain $\epsilon$-isolated until $\epsilon$ is quite large. This is the genesis of the claim above about the effects of ‘small’ noise on the MST. Of course, the direction of the noise vector also matters. If a noise-added point happens to fall near another point—e.g., if the noise vector is along the trajectory, rather than in a transverse direction—the associated MST edge will not distinguish it from the non-noisy points. There are some ways around this, as described later in this paper. Taken together, the noisy data sets in Figure 4 are representative of a broad class of potential disturbances; they contain instances of the pathological, difficult-to-detect cases mentioned above—small noise magnitudes and along-the-orbit noise—and so they provide a useful set of test cases for this paper. Incidentally, we use embedded data in these examples, rather than the full state-space trajectories produced by the integrator, because we were interested in evaluating the utility of these techniques for experimental data, and dynamical systems are rarely observable in practice.

The obvious differences between the $I(\varepsilon)$ curves in Figures 2 and 3 suggest a topology-based filtering scheme. Specifically, a breakpoint in the falloff region of the curve (indicated by $\varepsilon_n$ in Figure 3), followed by a second hump, indicates that there is a scale separation in the data set. That is, the MST edge length distribution has two peaks, one below $\varepsilon_n$ and one above it. Such a separation of scale can arise if two processes are at work in the data—such as signal and noise. If the noise is large compared to the sample spacing of the data, we can take advantage of that scale separation in order to disentangle the two. The breakpoint $\varepsilon_n$, in this case, can be interpreted as approximating the maximum edge length of the MST of the non-noisy data. At that value, most of

---

3Note that any or all of the coordinates of a given embedded point may include noise, so points do not simply move $\pm n$ along one axis of the reconstruction space.

4In what control theory calls an observable system, one can either measure or deduce values for every state variable.
Figure 3: The effects of noise upon attractor topology. Each point in the original Lorenz time series from the previous Figure was perturbed, with probability 0.01, by a value of ±1.0, which is roughly 2% of the overall width of the reconstructed attractor. This noise-added data was then embedded to obtain the trajectory shown in (a). The minimal spanning tree of this set, pictured in (b), clearly shows the noisy points, as does the $I(\epsilon)$ plot in part (c), where the noise adds a shoulder to the curve of Figure 2(d). $\epsilon_n$ can be interpreted as approximating the maximum edge length of the MST of the non-noisy data.
Figure 4: Effects of noise distribution upon $I(\epsilon)$. (a) Constant noise: $\pm x_{\text{noise}} = n$ was added to each $x$ in the original Lorenz time series, with probability 0.01, for various $n$. (b) Uniform random noise: $\pm x_{\text{noise}} \in [n, m]$ was added to each $x$, with probability 0.01, for various $n$ and $m$. All values in $[n, m]$ were chosen with equal probability. (c) Gaussian noise: $\pm x_{\text{noise}}$ was added to each $x$, with probability 0.01. The mean and standard deviation of $x_{\text{noise}}$ was varied as shown. In all cases, noise adds recognizable shoulders to the plots.
Table 1: Effects of noise on the filtering algorithm. Noise parameters as in previous Figure; pruning length = 0.3 throughout. Signal-to-noise ratio $SNR = 20 \log_{10} \frac{\text{signal}}{\text{noise}}$.

<table>
<thead>
<tr>
<th>Noise Type</th>
<th>Parameters</th>
<th>SNR of Noisy Data (dB)</th>
<th>% Noisy Points Removed</th>
<th>SNR of Filtered Data (dB)</th>
<th>% Non-Noisy Points Removed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>0.5</td>
<td>23.8</td>
<td>100</td>
<td>$\infty$</td>
<td>2.1</td>
</tr>
<tr>
<td>Constant</td>
<td>1</td>
<td>23.2</td>
<td>98.0</td>
<td>56.9</td>
<td>1.4</td>
</tr>
<tr>
<td>Constant</td>
<td>2</td>
<td>22.1</td>
<td>97.7</td>
<td>54.7</td>
<td>2.1</td>
</tr>
<tr>
<td>Constant</td>
<td>3</td>
<td>23.7</td>
<td>99.4</td>
<td>68.2</td>
<td>1.9</td>
</tr>
<tr>
<td>Uniform</td>
<td>[0.5 4]</td>
<td>22.3</td>
<td>100</td>
<td>$\infty$</td>
<td>2.5</td>
</tr>
<tr>
<td>Uniform</td>
<td>[2 4]</td>
<td>23.2</td>
<td>98.7</td>
<td>60.8</td>
<td>2.1</td>
</tr>
<tr>
<td>Uniform</td>
<td>[2 6]</td>
<td>23.7</td>
<td>100</td>
<td>$\infty$</td>
<td>2.1</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$\sigma = 1$, $\bar{x} = 3$</td>
<td>23.5</td>
<td>100</td>
<td>$\infty$</td>
<td>2.1</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$\sigma = 1.75$, $\bar{x} = 2.25$</td>
<td>23.6</td>
<td>96.0</td>
<td>51.3</td>
<td>2.0</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$\sigma = 2$, $\bar{x} = 4$</td>
<td>23.5</td>
<td>97.9</td>
<td>56.9</td>
<td>2.0</td>
</tr>
</tbody>
</table>

The noisy points—and few of the regular points—are $\epsilon$-isolated. One can easily discard the noisy points by pruning the appropriate edges of the MST: those that are longer than the breakpoint value $\epsilon_n$. Figure 5 shows the results of this filtering technique, as applied to the data from Figure 3. Using $\epsilon = 0.3$ (slightly above the $\epsilon_n$ breakpoint from Figure 3) to prune the minimal spanning tree, this method removed 534 of the 545 noisy points and 150 of the 7856 non-noisy points. This translates to 98.0% success with a 1.9% false positive rate, and a signal-to-noise ratio or SNR ($20 \log_{10} \frac{\text{signal}}{\text{noise}}$) reduction from 23.2$dB$ to 56.9$dB$. These rates vary slightly for different types and amounts of noise, but the success and false-positive percentages remain close to 100% and 0%, respectively, and the SNR of the filtered data is substantially larger than that of the noisy data; see Table 1. These are promising numbers—comparable to or better than the existing filtering schemes described in the introduction. Incidentally, the exact details of the pruning algorithm are somewhat more subtle than is implied above because not all noisy points are terminal nodes of the spanning tree. Thus, an algorithm that simply deletes all points whose connections to the rest of the tree are longer than the pruning length can sever connections to other points, or clusters of points. This is an issue if one noisy point creates a ‘bridge’ to another noisy point and only one of the associated MST edges is longer than $\epsilon_n$. 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. Lastly, note that while noise was added to each point in the scalar time-series data with probability 0.01, each of those data points is a coordinate of $m$ points in an $m$-dimensional embedding, so roughly $m\%$ of the points in the embedded trajectory are noisy.

Another way to evaluate this filtering method is to use dynamical invariants: e.g., to compare the Lyapunov exponent $\lambda$ of the original, noise-added, and filtered trajec-
Figure 5: Topology-based filtering of the Lorenz data: the original data—cf, Figure 2(a)—is shown at the top and the noise-added version of that data from Figure 3(a) appears in the middle. The filtered data in the bottom image was obtained by pruning the MST of Figure 3(b) using $\epsilon = 0.3$—a value just above the $\epsilon_n$ breakpoint in part (c) of Figure 3. Using this pruning value, the topology-based filtering algorithm removes 98% of the noisy points, with a false-positive rate of 1.9%.
<table>
<thead>
<tr>
<th>Noise Type</th>
<th>Parameters</th>
<th>$\lambda$ of Noisy Data</th>
<th>$\lambda$ of Filtered Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>0.5</td>
<td>8.7</td>
<td>3.2</td>
</tr>
<tr>
<td>Constant</td>
<td>1</td>
<td>35.5</td>
<td>3.2</td>
</tr>
<tr>
<td>Constant</td>
<td>2</td>
<td>82.2</td>
<td>3.4</td>
</tr>
<tr>
<td>Constant</td>
<td>3</td>
<td>81.3</td>
<td>2.9</td>
</tr>
<tr>
<td>Uniform</td>
<td>[0.5 4]</td>
<td>68.9</td>
<td>2.8</td>
</tr>
<tr>
<td>Uniform</td>
<td>[2 4]</td>
<td>81.9</td>
<td>4.7</td>
</tr>
<tr>
<td>Uniform</td>
<td>[2 6]</td>
<td>90.7</td>
<td>3.0</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$\sigma = 1$, $\bar{x} = 3$</td>
<td>73.5</td>
<td>2.9</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$\sigma = 1.75$, $\bar{x} = 2.25$</td>
<td>68.2</td>
<td>2.5</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$\sigma = 2$, $\bar{x} = 4$</td>
<td>78.2</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Table 2: Effects of noise and filtering on the Lyapunov exponent, as calculated by Wolf’s algorithm. Noise type and parameters as in Figure 4. The $\lambda$ of the original trajectory is discussed in the text.

Table 2 shows these results, calculated with an IDL implementation of Wolf’s algorithm[38]. As is well known, noise not only increases the dimension of an orbit, but also affects its $\lambda$. This effect, which is described nicely in [6, 7], is abundantly clear from the third column of the table. Ideally, filtering out the noise would reduce $\lambda$ back to the original value. In practice, all of these calculations—and the comparison to the $\lambda$ of the original trajectory—are somewhat problematic, as numerical algorithms for calculating $\lambda$ are notoriously sensitive to orbit length, initial conditions, and other dynamical, algorithmic, and computational parameters. For instance, if the initial conditions of the trajectory used to generate Figure 2(a) are varied slightly—$\pm 0.1$ on each of the three state variables $x$, $y$, and $z$—the calculated $\lambda$ ranges from 0.125 to 0.422, in spite of the temporal averaging performed by Wolf’s algorithm. If the initial condition is fixed at the values used in Figure 2 and the orbit length is varied from 5000 to 30000, the calculated $\lambda$ ranges from 0.11 to 0.86. The $\lambda$ of the embedded trajectory—which should be equal to the original $\lambda$ if the embedding dimension is adequate—varies as shown in Table 3. Since the dimension of the Lorenz system is three, the Whitney/Mañe/Takens conditions require that $m \geq 7$, in the worst case, for a successful embedding, though Casdagli et al. suggest that $m \geq d_A$, where $d_A$ is the box-counting dimension, is sufficient[34]. The $\lambda$ values in the Table do indeed appear to settle out—around $m = 6$ or 7—but to a value that is significantly lower than those calculated for the full $xyz$ trajectory. All of this variation makes it hard to know what to compare the values in the rightmost column of Table 2 against. Furthermore, removing points affects the $\lambda$ calculation. Wolf’s algorithm proceeds by identifying a point’s nearest neighbor, then following the point pair until their spacing exceeds a heuristic threshold, then renormalizing by finding a new near neighbor\textsuperscript{5}, and repeating to the end of the trajectory. Removing points from that trajectory will necessarily force the near-neighbor search to return artificially distant results, with unpredictable effects upon the calculated $\lambda$. To explore this effect, we

\textsuperscript{5}In the direction of the vector of the last separation
randomly removed 700 of the points in the trajectory of Figure 2—the same number that were removed, on the average, in our pruning experiments—and re-ran the calculation. Over ten such trials, $\lambda$ ranged from 1.9 to 2.7, so it appears that removing points, in general, raises $\lambda$. All of these experiments make it clear that the error bars of Wolf’s algorithm, especially as applied to data sets with pointwise gaps, are substantial. (This is not a new result, nor a unique feature of this algorithm; [6] discusses this issue, comparing and contrasting several $\lambda$-calculation algorithms.) In this context, it appears that our filtering method works very well, though not perfectly: the $\lambda$ values for the filtered data are dramatically smaller than those for the noisy data, and not hugely different from the range of “true” $\lambda$ values. This corroborates the success/false positive percentages given in the previous paragraph, and confirms that this topology-based filtering scheme is indeed effective at removing noise without disturbing the dynamics.

Removing noise that one has artificially added to a trajectory is a useful first test, but it is certainly not the intended use of this filtering method. The real applications are sensor data from physical experiments\(^6\). Figure 6 shows an instance of such an application: an experimental data set from a parametrically forced pendulum. The sampling rate in this data set was much faster than the highest frequency of the dynamics, but the angle sensor was intermittently noisy, and so the data set contains some outlying points, which are clearly visible in the minimal spanning tree in part (b). Because of the periodic boundary conditions in the projection, the reconstructed attractor appears to be disconnected, and so the MST contains a dozen or so long edges that span the apparent gaps in the attractor. Note that while these edges are much longer than the others, their endpoints are not isolated, and so they will not cause the filtering algorithm to generate false positives. The $I(\epsilon)$ curve, too, is somewhat more complicated than in the Lorenz examples: it has a well-defined shoulder running from $\log \epsilon \approx -1.5$ to $\approx -0.5$.

\(^6\)The problems caused by floating-point arithmetic limitations are comparatively minor on modern machines.
Figure 6: Experimental driven pendulum data. (a) 99800 measurements of the bob angle $\theta$, sampled every 265 microseconds by an optical encoder with a resolution of 0.7 degree, and embedded in $[0,2\pi)^2$ with $\tau = 0.0265$ seconds. (b) MST and (c) $I(\epsilon)$ of the embedded pendulum trajectory. In these 2D renderings, $\theta(t + \tau)$ is plotted against $\theta(t)$. The long edges in the MST that span the apparent gaps in the attractor are an artifact of the periodic boundary conditions, as are the stair-steps in the $I(\epsilon)$ plot for $\log \epsilon > -0.5$. 

15
followed by a stair-step falloff for \( \log \epsilon > -0.5 \). As described earlier in this paper, this kind of pattern can be an indication that the set is a disconnected fractal. In this case, however, it too is simply a side effect of the periodic boundary conditions: those stair steps reflect the spurious gap-spanning edges.

Filtering these data as in the Lorenz example, we delete all points that are \( \epsilon \)-isolated, using a \( \log \epsilon \) value of -1.4, which is slightly above the breakpoint \( \epsilon_n \) of the \( I(\epsilon) \) curve shown in part (c). The results are shown in Figure 7. Obviously, we cannot give percentage comparisons here, as we do not know which of the original points are noisy. A visual comparison of Figures 6 and 7, however—especially the MST close-ups—suggests that most of the noisy points have been removed. Moreover, the absence of a shoulder on the \( I(\epsilon) \) plot in Figure 7(d) suggests that the filtered trajectory is continuous and perfect—as it should be, given that the pendulum is a continuous-time dynamical system, and that the sampling interval is small. Note, too, that the edges spanning the spurious gaps in the projected attractor are removed by the filtering scheme. All in all, these results are quite promising.

Oversampling turns out to be critical to the implementation of this approach, for a variety of reasons whose roots lie in the mathematics of continuity. Undersampling destroys perfectness. In sparsely sampled data, for example, MST edges not only skip over unsampled chunks of the attractor, but can even jump crosswise from one attractor thread to another instead of connecting points along the orbit. A related issue is our filtering model: we simply remove the point; we do not deduce where it ‘should be’ and move it in that direction. There are several obvious solutions to these problems, all of which move towards the geometry-based schemes outlined in the first paragraph of this paper—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. There are other ways to leverage continuity, both in identifying noisy points and reconstructing where they should have been. Any displacement transverse to the orbit, for example, is a clear suspect for noise. Zahn[40] exploits this to prune noisy points from bubble chamber data, using the local structure of the minimal spanning tree to identify points that are not “on the main tracks.” In an even earlier application, Clark and Miller[10] use MSTs to link sequences of spark-chamber images by iteratively removing the short, terminal ‘hairs’ on the tree. Incorporating temporal information may also be helpful; our current scheme only uses the geometry and topology of the data, disregarding which points were close in time—which can be an effective indication of continuity (and violations thereof). Adding time as an additional dimension in the distance metric used in the MST would address this. Note, however, that spatiotemporal continuity only holds for flows, so this would not be useful if one is working with data from a map. We are working out a sensible formalization of all of these ideas within our computational topology framework, and will report upon their results in a future paper. In the meantime, this preliminary implementation works quite well, and it is generally far easier to oversample a physical system than to fix the source of the noise.

Quantization has complicated and interesting effects upon this scheme. Data quantization implies distance metric quantization, so MST edges can only take on discrete
Figure 7: Filtered pendulum data, obtained by removing all $\epsilon$-isolated points from the data in the previous Figure, with $\log \epsilon = -1.4$ chosen just above the first breakpoint of the associated $I(\epsilon)$ curve.
lengths. Moreover, in raster images, noise does not move points around, as it does in the examples in this paper; rather, it simply reshares pixels. The computer vision community[3] distinguishes these two kinds of noise as “distortion” and “salt and pepper,” respectively. Because the metric used in the MST captures distances between points, it is more effective at detecting the latter than the former. These issues are discussed in more depth in a companion paper written for the computer science community[4].

3 Conclusion

The filtering method introduced in this paper exploits the fundamental topological properties of continuous-time dynamical systems in order to find and remove noisy points from a data set. ō-limit sets of these systems are, in theory, perfect sets—that is, they contain no isolated points. In practice, however, the numerical representation of an orbit fails to be perfect because of finite sampling and/or additive noise. The approach described here uses a variable-resolution approach to computational topology to work around the sampling effects and effectively identify the isolated, noisy points—and then remove them. Experiments with clean Lorenz data contaminated with various forms of noise showed encouraging results: the topology-based filter removed 96-100% of the noisy points, improving the signal-to-noise ratio or SNR $\left(20 \log_{10} \frac{\text{signal}}{\text{noise}}\right)$ from $\approx 20 dB$ to more than $50 dB$, with a false-positive rate of 1.4-2.5%. Data from a laboratory apparatus—a parametrically forced pendulum—showed equally encouraging results, though of course one cannot quantify noise percentages or ratios in real-world data.

There have been a few other topology-based approaches to dynamical systems analysis. Our use of the minimal spanning tree as a data structure was inspired by Yip’s work on automated phase-portrait analysis[39]. Mishaikow et al.[24] use algebraic topology to construct a symbol dynamics from trajectory data. This neatly fineses the noise issue by using a coarser representation, 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. Muldoon et al. compute the homology of embedded attractors[25]. This approach does not address noise directly, but does describe how to build a triangulation from time-series data, and then computes many of the same topological quantities that are used in our broader work. Mees[22] uses tesselations to find the dimension of the state space, along with information about folds and branches in the attractor, leading to a method for reconstruction of models of the dynamics. Mindlin and Gilmore also use topological techniques—templates and knots, in particular—to model the stretching/squeezing dynamics on a chaotic attractor[23]. Lastly, approaches that are based on unstable periodic orbits[2, 15] could also be viewed as topological.

The key feature of the method proposed here is its identification of separation of scale: if the magnitude of the noise is greater than the inter-point spacing, the variable-
resolution topological analysis described in this paper will bring that out. Conversely, noise that is of the order of the separation between the points in the data set, or less, does not produce an $\epsilon$-isolated point, and thus cannot be identified or removed by this method. Filtering schemes have different weaknesses, but all are vulnerable to small noise magnitudes. Like most others, too, this method works only for additive noise, not dynamic noise, which is coupled back into system evolution (e.g., an error in an ODE solver, which affects the initial condition for the next step). Like most of the other methods described in the dynamical systems literature, this scheme is noncausal, in the sense that its algorithms require both past and future values of the trajectory. As it is currently implemented, this method requires the data to be oversampled, but we are working on modifications that remove that restriction, and that selectively add geometric knowledge to the algorithm so it can be more intelligent about detecting and adjusting noisy points. Separation of scale is fundamental to many other forms of signal that one might be interested in untangling, so this method is by no means limited to dynamical systems—or to filtering applications.

Acknowledgements

The IDL implementation of Wolf’s algorithm was written by Ricardo Mantilla of the University of Colorado. Wheeler Ruml of PARC offered useful comments on a draft of this document.

References


