Topological Feature Vectors for Chatter Detection in Turning Processes

Melih C. Yesilli
Department of Mechanical Engineering
Michigan State University
yesillim@egr.msu.edu

Firas A. Khasawneh
Department of Mechanical Engineering
Michigan State University
khasawn3@egr.msu.edu

Andreas Otto
Institute of Physics
Chemnitz University of Technology
andreas.otto@physik.tu-chemnitz.de

Abstract

Machining processes are most accurately described using complex dynamical systems that include nonlinearities, time delays and stochastic effects. Due to the nature of these models as well as the practical challenges which include time-varying parameters, the transition from numerical/analytical modeling of machining to the analysis of real cutting signals remains challenging. Some studies have focused on studying the time series of cutting processes using machine learning algorithms with the goal of identifying and predicting undesirable vibrations during machining referred to as chatter. These tools typically decompose the signal using Wavelet Packet Transforms (WPT) or Ensemble Empirical Mode Decomposition (EEMD). However, these methods require a significant overhead in identifying the feature vectors before a classifier can be trained. In this study, we present an alternative approach based on featurizing the time series of the cutting process using its topological features. We utilize support vector machine classifier combined with feature vectors derived from persistence diagrams, a tool from persistent homology, to encode distinguishing characteristics based on embedding the time series as a point cloud using Takens embedding. We present the results for several choices of the topological feature vectors, and we compare our results to the WPT and EEMD methods using experimental time series from a turning cutting test. Our results show that in most cases combining the TDA-based features with a simple Support Vector Machine (SVM) yields accuracies that either exceed or are within the error bounds of their WPT and EEMD counterparts.

1 Introduction

Chatter is characterized by excessive vibrations that can negatively affect the surface finish or shorten the lifetime of cutting tools during machining operations such as turning and milling. Consequently, identification and mitigation of chatter has become a prominent research topic in recent decades. Some of the challenges associated with chatter identification is that it is a complex phenomenon that depends on several factors including the dynamic properties of the tool and the workpiece. Therefore, as these properties vary during the cutting process, the results of predictive models become invalid, thus necessitating a data-based approach for more reliable chatter detection. Motivated by this goal, many studies in the literature have focused on extracting chatter features from signals obtained using sensors mounted on the cutting center. Most of these studies are based on analyzing the spectrum of force or acceleration signals often in combination with machine learning techniques.

For example, Thaler et al. utilized short time Fourier transform with quadratic discriminant analysis to detect chatter in a band sawing process [1]. However, the two most common methods for analyzing

cutting signals are the Wavelet Packet Transform (WPT) and the Empirical Mode Decomposition (EMD). Chen and Zheng used WPT to isolate the part of the signal carrying chatter markers, and they utilized it to compute both time and frequency domain features [2] that can be used as feature vector for machine learning. Similarly, Ji et al. used EMD to decompose the acceleration signal from the tool vibration into Intrinsic Mode Functions (IMFs) [3]. They then identified the IMFs that carry chatter information called the informative IMFs and they used them to define feature vectors for chatter identification. Liu et al. applied EMD to servo motor current signal to compute the energy and the kurtosis of the informative IMFs as features for chatter [4]. They then used Support Vector Machine (SVM) algorithm to train a classifier for chatter detection. SVM is the most widely used machine learning algorithm for chatter classification from sensory signals. Some less commonly used methods include logistic regression [5], deep belief network [6], back propagation neural network [7], and Hidden Markov Model [8, 9]. Although WPT and EEMD are widely used for detecting chatter, these methods have some limitations that preclude them from being adopted as general chatter detection tools. To elaborate, it was recently shown in [10] that identifying appropriate feature vectors using these two methods is signal-dependent and it requires skilled operators. Further [10] also showed that the accuracy of the trained classifier is highly sensitive to changes in the dynamic properties of the tool-workpiece system.

Another promising tool for generating feature vectors for chatter detection comes from a new field with many mature computational tools: Topological Data Analysis (TDA) [11, 12, 13, 14]. TDA, and more specifically persistent homology, provides a quantifiable way for describing the topological features in a signal [15]. Specifically, by embedding the sensory signal into a point cloud, it is then possible to use persistent homology to produce a multiscale summary of the topological features of the signal thus enabling the analysis of the underlying dynamical system. The homology classes that correspond to the embedded signal are often reported using a planar diagram that shows how long each topological feature persisted. The application of TDA tools to machining dynamics has only been recently explored [16, 17, 18]. Specifically, Refs. [19, 17] show that maximum persistence—a single number from the persistence diagram—can be used to ascertain the stability of simulated data from a stochastic turning model. In Ref. [18], the authors incorporated more information from the persistence diagram by extracting 5 features including Carlsson coordinates [20] and the maximum persistence, see Section 1.1 for more details on featurizing persist ence diagrams. The resulting feature vector in combination with SVM was used to train a chatter classifier, and it was applied to simulated deterministic and stochastic turning data with success rates as high as 97% in the deterministic case.

However, despite the active work in the literature on featurizing persistence diagrams, all prior studies on chatter detection with TDA have utilized only a small fraction of the persistence diagram for constructing a feature vector. Further, these publications only studied simulated signals and no sensory signals from actual cutting tests have been tested. Therefore, this work aims to collect and summarize state-of-the-art featurization tools for persistence diagrams, and apply them for the first time for chatter classification using actual experimental signals obtained from an accelerometer mounted on the cutting tool during a turning process. The methods that we investigate for featurizing the resulting persistence diagrams and classifying chatter time series include persistence landscapes [21], Carlsson coordinates [20], persistence images [22], an example kernel method [23], and path signatures of persistence landscapes [24].

The paper is organized as follows. In Section 1.1 we provide a brief literature survey of featurization methods for persistence diagrams. In Section 2 we describe the experimental apparatus as well as how the data was processed and tagged. Section 3 provides the needed background material on the tools we use from TDA. The method we use for featurizing the persistence diagrams and training a classifier is explained in Section 4. Section 5 presents the classification results and runtime comparisons, while the concluding remarks can be found in Section 6.

1.1 An overview for featurization of persistence diagrams

Persistence diagrams can be compared to each other using metrics such as the Wasserstein and Bottleneck distances. This allows creating similarity or distance matrices betwee different persistence diagrams that can be used for machine learning. For example, in previous studies on machine learning applications with TDA, these two distances were utilized as similarity measures between persistence diagrams for classification algorithms [25]. However, working directly with persistence diagrams is difficult in part due to the non-uniqueness of geodesics which leads to non-uniqueness of the Fréchet mean for a collection of diagrams

[26, 27, 28]. This necessitates using methods for extracting feature vectors that live in a Euclidean space from the persistence diagrams, which enables using traditional machine learning tools with topological features. The following paragraph provides a list of some of these methods.

Adcock et al. describe a subring of polynomial functions on persistence diagrams and a convenient system of free generators [20]. However, their construction is not continuous with respect to the bottleneck distance, and it is not applicable to the case of infinite persistence diagrams which are relevant when working with fractal structure. Other tools for extracting feature vectors from persistence diagrams are based on functional summaries [29], which turn persistence diagrams into functions. Perhaps the most common functional summary of persistence diagrams is persistence landscapes [21] which represent persistence diagrams in the form of piecewise continuous functions. Although averages of persistence diagrams are not well-defined, persistence landscapes have well-established statistical properties [30] and their averages and pairwise distances are defined which enables combining them with traditional machine learning tools. Another functional summary of persistence diagrams is persistence images [31, 22], which are closely related to Chen et al. persistence intensity functions [32]. Persistence images are also related to prior works on the size function which predates persistence [33, 34]. These images are obtained by placing Gaussians at each point in the diagram, and then utilizing a histogram of the image for featurization and machine learning.

Another featurization method is based on path signatures of persistence landscapes [24]. Specifically, signatures for n-dimensional paths of bounded variation can be computed and used as features for machine learning [35]. Therefore, by composing persistence landscapes, which can be viewed as one type of persistence path embedding, with path signatures, it is possible to extract mesh-free feature vectors from the underlying persistence diagram. However, path signatures do not come with stability guarantees for all input paths.

Persistence diagrams can also be featurized using a general class of functions called template functions [36]. These functions are only required to be continuous and compactly supported. The specific realization of these functions that we work with in this paper are Chebyshev interpolating polynomials [37].

Kernel-based methods have also been used for featurizing persistence diagrams [23, 38, 39, 40, 41, 42, 43]. However, instead of testing all of the available options, we choose the persistence scale space kernel given by [23] as a representative of this group of methods. This kernel is defined by treating the persistence diagram as a sum of Dirac deltas at each point in the persistence diagram, and using this as the initial condition for a heat diffusion problem. This allows obtaining a closed form solution for this kernel.

2 Experimental Setup and Signal Processing

This section describes the details of the cutting tests 2.1, the data collection, signal tagging 2.3, and time series conditioning 2.2.

2.1 Cutting tests

Figure 1 shows the experimental setup which features a Clasuing-Gamet 33 cm (13 inch) engine lathe instrumented with three accelerometers. Two PCB 352B10 uniaxial accelerometers were used to measure the x and y accelerations of the boring rod, see Fig. 1. These two accelerometers were mounted 3.81 cm (1.5 inch) away from the cutting tool to keep them safe from the cutting debris. Further, a PCB 356B11 triaxial accelerometer was attached to the tool holder with superglue. The acceleration signals were collected with a sample rate of 160 kHz using an NI USB-6366 data acquisition box and Matlab's data acquisition toolbox, see Section 2.2 for more information on the applied signal processing.

The used cutting tool is a 0.04 cm (0.015 inch) radius Titanium nitride coated insert which was attached to an S10R-SCLCR3S boring bar, where the latter is part of the Grizzly T10439 carbide insert boring bar set.

The cutting tests were performed by varying the depth of cut and the cutting speed while holding the stiffness of the boring rod constant. Data was collected from four sets of cutting configurations where each configuration corresponds to a different boring rod stiffness/eigenfrequency. The stiffness of the rod was varied by changing the stickout length of the tool, which is the distance between the heel of the boring rod and the back surface of the tool holder. Four different stickout lengths were considered: 5.08 cm (2 inch), 6.35 cm (2.5 inch), 8.89 cm (3.5 inch) and 11.43 cm (4.5 inch). Increasing the stickout length leads to a stiffer

rod and to higher chatter frequencies since the latter are close to the eigenfrequency of the rod. Similarly, smaller stickout length leads to a more compliant system, and to lower chatter frequencies.
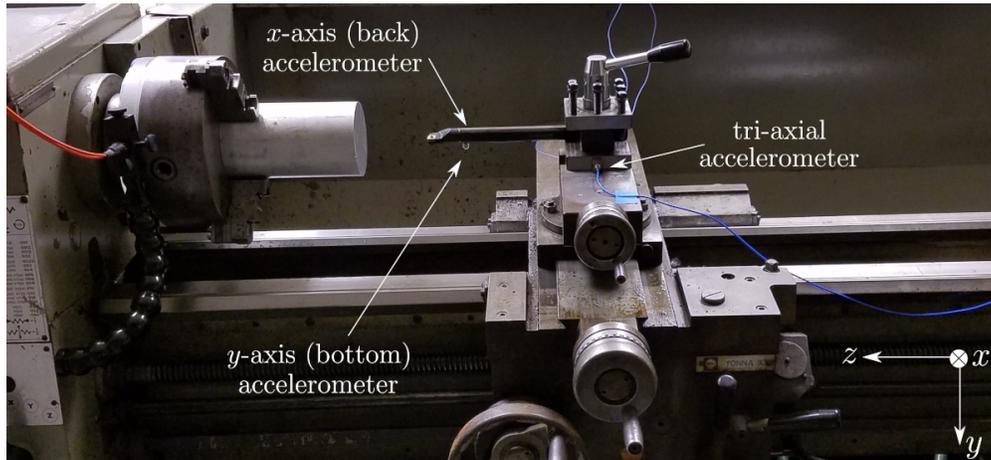

Figure 1: The experimental setup used for collecting cutting data.

2.2 Signal Processing

The experimental data was sampled at the rate of 160 kHz, which is an oversampling by a factor of 16 of the maximum accelerometer measuring frequency of 10kHz. Since no in-line analog filter was used, this oversampling is necessary to avoid aliasing effects by utilizing a digital filter before downsampling the signal to 10kHz. The filter we used is Butterworth low pass filter with order 100 from Matlab's signal processing toolbox. The resulting filtered and downsampled signals were the ones used in tagging the data as described in Section 2.3.

2.3 Data Tagging

Upon examining the acceleration signals, we found that the x-axis signals from the tri-axial accelerometer (see Fig. 1) contained the best Signal-to-Noise Ratio (SNR). The other acceleration signals contained redundant information albeit with lower SNR; therefore, only the x-axis signal from the tri-axial accelerometer was used for tagging the data. Both time and frequency domain information were simultaneously considered when tagging the data. Specifically, in the time domain, we separated the different parts of the raw time series according to amplitude, i.e., portions with large amplitude, low amplitude, and with sudden increase in amplitude. In the frequency domain, frequencies under 5 kHz in the downsampled signal were taken into account.

   We established specific criteria to label the raw data which we describe below. Figure 3a shows that during the same cutting test it was possible for different parts of the time series to be tagged differently. A signal was tagged chatter-free (stable) if it has a low amplitude in both time and frequency domains. These signals have the highest peaks at the spindle rotation frequencies in the Fourier spectrum [44], as shown in Fig. 3b. On the other hand, we characterized intermediate chatter signals as having low amplitude in the time domain but high amplitude in the frequency domain, see Fig. 3c. In contrast, chatter signals have large amplitude in both time and frequency domains, as shown in Fig. 3a and the spectrum superimposed on Figs. 3b–3d. For intermediate chatter and chatter signals, the highest peaks are around the chatter frequencies in the frequency domain. Any case that did not fit with any of the above categories was labeled as unknown. Figure 3d shows that these cases are generally composed of time series that have sudden increase and decrease in amplitude in the time domain combined with low amplitude in the frequency domain.

   The data labeling was verified by spot-checking the tagged data against pictures taken during the cutting tests. Figure 2 shows one such comparison for a sample tagged signal. The figure shows that the surface finish characteristics agree with the assigned tagging, thus confirming the validity of the labeling.
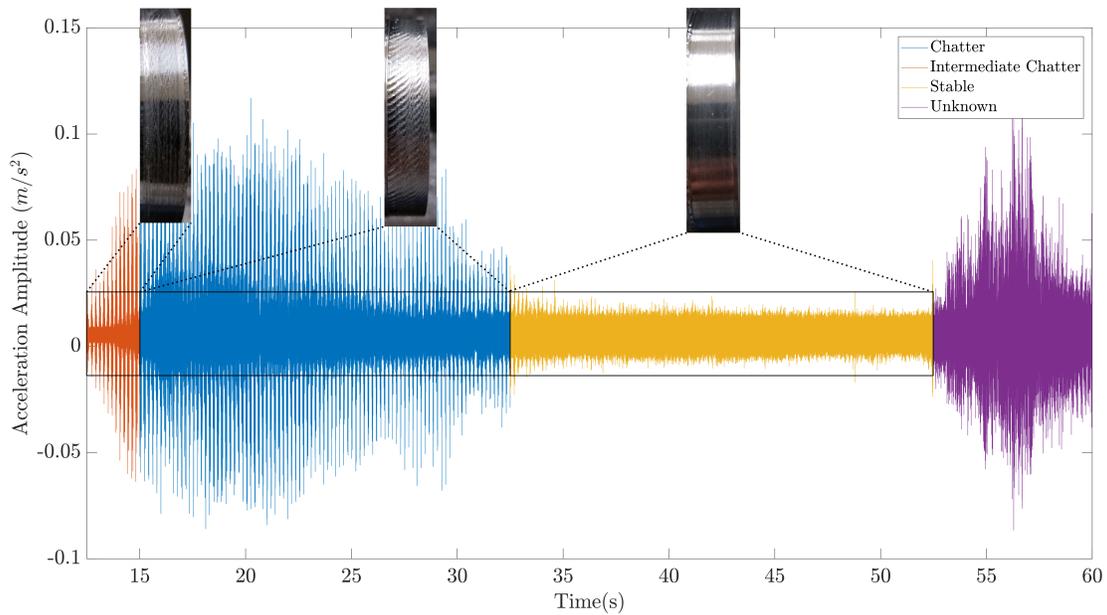
Figure 2: Tagged raw data and corresponding cut surfaces for 5.08 cm (2 inch) 320 rpm and 0.0127 cm depth of cut (0.005 inc) case.
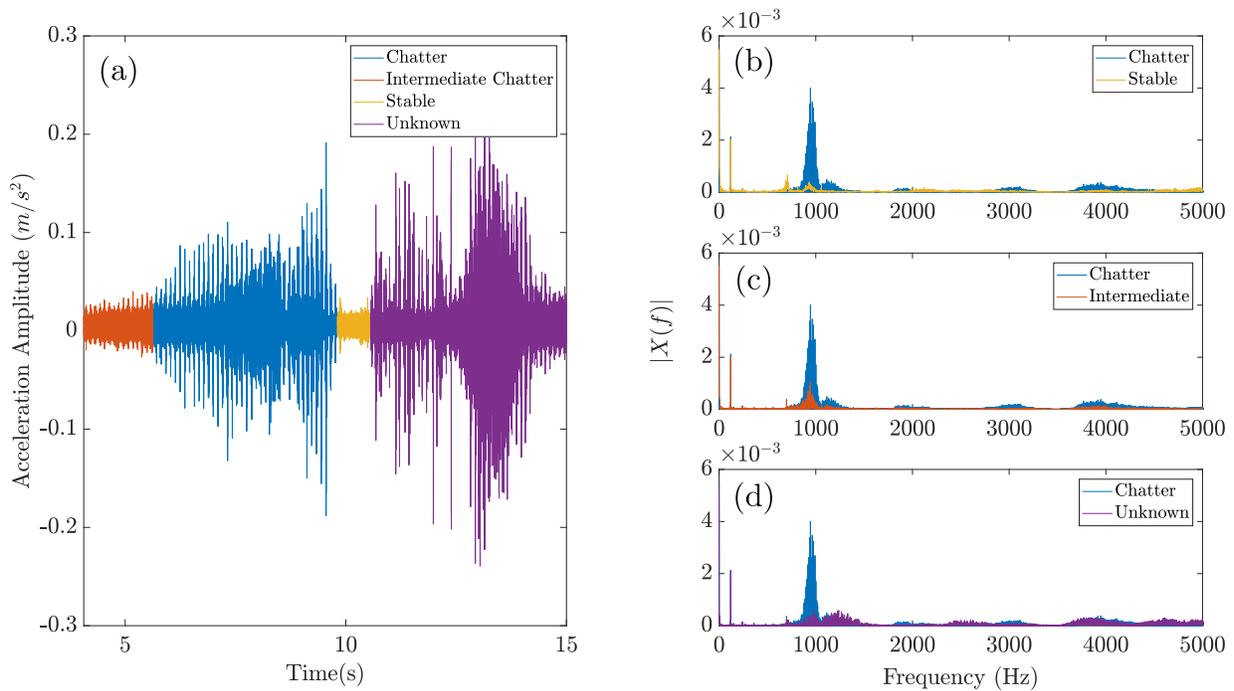

Figure 3: (a) An example of a tagged time series, and a plot of the spectrum of the chatter regime superimposed onto the spectrum of: (b) chatter-free cutting, (c) intermediate chatter, and (d) unknown case. The signal is for turning with a stickout length of 5.08 cm (2 inch), 570 rpm, and 0.00508 cm depth of cut(0.002 inch).

5

Stickout length (cm (inch))	Stable	Mild chatter	Chatter	Total
5.08 (2)	17	8	11	36
6.35 (2.5)	7	4	3	14
8.89 (3.5)	7	2	2	11
11.43 (4.5)	13	4	5	22

Table 1: The number of tagged datasets for each stickout case.

3 Topological Data Analysis

Topological Data Analysis (TDA) extracts information from the data by quantifying its shape and structure. One of the main tools of TDA is persistent homology. Specifically, in this paper we study the time series by embedding them using delay reconstruction and then applying 1-D persistent homology to obtain an information-rich summary of the shape of the resulting ambient space. Features are then extracted from the persistence diagrams and used for machine learning. This section briefly describes the main concepts of 1-D persistent homology, but we defer a more thorough treatment to references in the literature such as [11, 12, 13, 14, 45].

3.1 Simplicial complexes

Geometrically, a k-simplex σ is a set V of $k+1$ vertices whose dimension is given by $\dim(\sigma) = k$. Some examples include the 0-simplex which is a point, the 1-simplex which is an edge, the 2-simplex which is a triangle, and so on, see Fig. 4 for examples. A simplicial complex K is a set of simplices $\sigma \subseteq V$ that satisfies specific inclusion relations. Specifically, if $\sigma \in K$, then all the lower dimensional component simplices $\sigma' \subset \sigma$, called the faces of σ, are also in K. For example, if a 2-simplex (triangle) is in a simplicial complex K, then so are the corresponding 1-simplices (edges of the triangle) as well as all the nodes 0-simplices (nodes of the triangle). The dimension of the resulting simplicial complex is $\dim(K) = \max_{\sigma \in K} \dim(\sigma)$, which is the largest dimension of its simplices. If we restrict the simplicial complex to its simplices of at most order n, then we obtain the n-skeleton of K denoted by $K^{(n)}$, i.e., $K^{(n)} = \{\sigma \in K \mid \dim(\sigma) \leq n\}$. Let $G = (V, E)$ be an undirected graph where V and E are the vertices and edges, respectively. We can then construct the clique or the flag complex

$$K(G) = \{\sigma \subseteq V \mid uv \in E \text{ for all } u \neq v \in \sigma\},$$

which is the 1-skeleton of the graph with simplices added, if available.

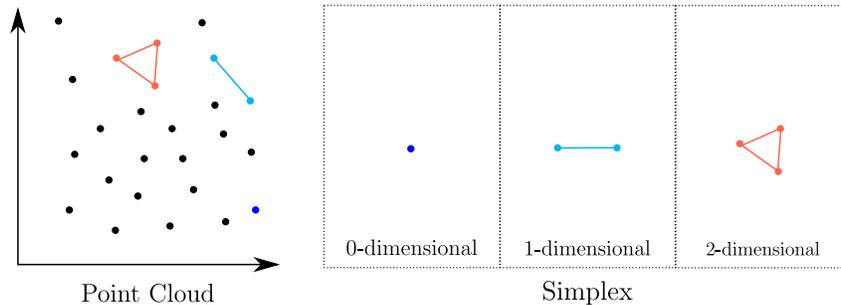
Figure 4: Formation of simplicial complexes from point cloud.

3.2 Homology

Assume that a simplicial complex K is fixed, then the corresponding homology groups can be utilized to quantify the holes of the structure in different dimensions. For instance, the rank of the 0 dimensional homology group $H_0(K)$ is the number of connected components (dimension 0). The rank of the 1 dimensional

homology group $H_1(K)$ is the number of loops (dimension 1), while the rank of $H_2(K)$ is the number of voids (dimension 2), and so on. We construct the homology groups using boundary operators which are linear transformations that are defined below.

To describe boundary operators, we first define the oriented simplicial complex as the simplicial complex obtained by an ordered set of vertices $\sigma = [v_1, \ldots, v_k]$. Permuting two indices in σ gives the opposite simplex according to $[v_1, \ldots, v_i, \ldots, v_j, \ldots, v_k] = -[v_1, \ldots, v_j, \ldots, v_i, \ldots, v_k]$. Now let $\{\alpha_\sigma\}$ be coefficients in a field $F = \mathbb{Z}_2$. Then the n-skeleton of K, $K^{(n)}$, can be used as a generating set of the F-vector space $\Delta_n(K)$. In this representation, any element of $\Delta_n(K)$ can be written as a linear combination $\sum_{\sigma \in K^{(n)}} \alpha_\sigma \sigma$, and addition of elements is accomplished by adding their coefficients. A finite formal sum of the n-simplices in K is called an n-chain, and the group of all n-chains is the nth chain group $\Delta_n(K)$, which is a vector space.

We are now ready to define boundary operators: Given a simplicial complex K and the field $\mathbb{F}$, the boundary map $\partial_n : \Delta_n(K) \to \Delta_{n-1}(K)$ is

$$\partial_n([v_0, \ldots, v_n]) = \sum_{i=0}^{n} [v_0, \ldots, \hat{v}_i, \ldots, v_n],$$

where $\hat{v}_i$ denotes the absence of element v_i from the set. This linear transformation maps any n-simplex to the sum of its codimension 1 faces. Geometrically, the boundary operator gives the boundary of a chain.

By combining boundary operators, we obtain the chain complex

$$\ldots \xrightarrow{\partial_{n+1}} \Delta_n(K) \xrightarrow{\partial_n} \ldots \xrightarrow{\partial_1} \Delta_1(K) \xrightarrow{\partial_0} 0,$$

with the well-known result that the composition of any two subsequent boundary operators is zero, i.e., $\partial_n \circ \partial_{n+1} = 0$. An n-chain $\alpha \in \Delta_n(K)$ is a cycle if $\partial_n(\alpha) = 0$; it is a boundary if there is an $n+1$-chain β such that $\partial_{n+1}(\beta) = \alpha$. Define the kernel of the boundary map ∂_n using $Z_n(K) = \{c \in \Delta_n(K) : \partial_n c = 0\}$, and the image of ∂_{n+1} $B_n(K) = \{c \in \Delta_n(K) : c = \delta_{n+1} c', c' \in \Delta_{n+1}(K)\}$. Consequently, we have $B_k(K) \subseteq Z_k(K)$. Therefore, we define the nth homology group of K as the quotient group $H_n(K) = Z_n(K)/B_n(K)$. In this paper, we only need 0- and 1-dimensional persistent homology. In the case of 0-dimensional homology, there is a unique class in $H_0(K)$ for each connected component of K. For 1-dimensional homology, there is one homology class in $H_1(K)$ for each hole in the complex.

3.3 Persistent homology

Homology is extremely useful for studying the structure of a simplicial complex. However, we are often interested in studying the structure of a changing simplicial complex. For example, assume we have a point cloud $P \subset \mathbb{R}^m$ which results from embedding the time series into $\mathbb{R}^m$, e.g., using delay reconstruction. For each point $p \in P$, let $B(p, r)$ be the ball centered at p and of radius r. For each choice of the radius r we can build a different simplicial complex where, for example, the intersection of any two balls adds an edge, the intersection of three balls adds a triangle, and higher dimensional analogs are added similarly. For a specific choice of r, the resulting simplicial complex of the union of all the balls $\bigcup_{p \in P} B(p, r)$ is well-approximated with Rips complexes which are the clique complexes of the filtration function given by

$$f(\sigma) = \max_{\{u,v\} \subseteq \sigma} d(u, v),$$

where $d(u, v)$ is the distance between vertices u and v, and the distance between a vertex and itself is zero. Notice that f is a real-valued function on the simplicies of K such that $f(\tau) \leq f(\sigma)$ for all $\tau \leq \sigma$ simplices in K. If we let $\{r_1 < r_2 < \ldots < r_\ell\}$ be the set of the sorted range of ball radii, then the filtration of K with respect to f is the ordered sequences of its subcomplexes

$$\emptyset \subseteq K(r_1) \subseteq K(r_2) \subseteq \cdots \subseteq K(r_\ell) = K.$$

The sublevel set of K corresponding to r is defined as

$$K(r) = \{\sigma \in K \mid f(\sigma) \leq r\}, \tag{1}$$

where each of the resulting $K(r)$ is a simplicial complex, and for any $r_1 \leq r_2$, we have $K(r_1) \subseteq K(r_2)$. The filtration of K enables the investigation of the topological space under multiple scales of the output value of the filtration function f.

The main idea behind persistent homology is to watch how the homology changes over the course of a given filtration. Fix a dimension n, then for a given filtration

$$K_1 \subseteq K_2 \subseteq \cdots \subseteq K_N$$

we have a sequence of maps on the homology

$$H_n(K_1) \rightarrow H_n(K_2) \rightarrow \cdots \rightarrow H_n(K_N).$$

We say that a class $[\alpha] \in H_n(K_i)$ is born at i if it is not in the image of the map $H_n(K_{i-1}) \rightarrow H_n(K_i)$. The same class dies at j if $[\alpha] \neq 0$ in $H_n(K_{j-1})$ but $[\alpha] = 0$ in $H_n(K_j)$.

This information can be used to construct a persistence diagram X as follows. A class that is born at i and dies at j is represented by a point in $\mathbb{R}^2$ at (i, j). The collection of the points in the persistence diagram, therefore, give a summary of the topological features that persist over the defined filtration. We denote the number of the off-diagonal points in the persistence diagram by $|X|$. See the example of Fig. 5 for $n = 1$ and point cloud data.
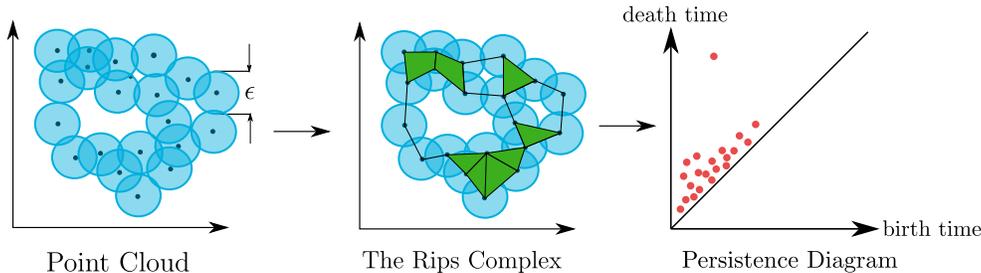

Figure 5: The Rips complex.

4   Method

The method that we develop for chatter detection using topological features can be summarized using Fig. 6. The parts of the pipeline related to data collection, processing, and labeling were described in Sections 2.1–2.3, respectively. In this section, we describe the rest of the steps shown in Fig. 6.

Recall that the cutting tests are composed of four different stickout configurations, and that each configuration includes a different number of time series that correspond to different labels, rotational speeds, and depths of cut. Therefore, the time series are grouped with respect to these three parameters and they are normalized to have zero mean and unit variance. This normalization reduces the effect of large feature values on smaller ones [46].

The next step is to split long time series into smaller pieces to reduce the computation time needed for finding the delay reconstruction parameters (see Section 4.1), and for obtaining the persistence diagrams (see Section 3.3). Upon finding the appropriate embedding parameters, the data is embedded using delay reconstruction, also known as Takens' delay embedding, see Section 4.1. The resulting point cloud is then used to compute the corresponding persistence diagrams using Ripser package for Python. Sections 4.2–4.6 describe different methods in the literature for featurizing the resulting diagrams to obtain a feature vector in Euclidean space that can be used with existing machine learning tools such as SVM. We mainly featurize and use the resulting 1-dimensional persistent homology H_1 except when we use the template function approach where we also utilize the 0-dimensional persistence H_0 [36].
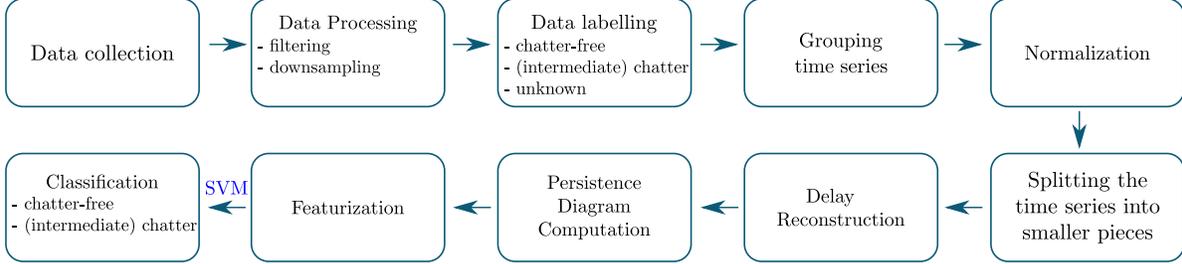
8

Figure 6: Persistence diagram computation steps

4.1 Delay Reconstruction

Takens' theorem lays a theoretical framework for studying deterministic dynamical systems [47]. It states that in general we can obtain an embedding of the attractor of a deterministic dynamical system from a one-dimensional recording of a corresponding trajectory. This embedding is a smooth map $\Psi : M \to N$ between the manifolds M and N that diffeomorphically maps M to N.

Specifically, assume we have an observation function $\beta(\mathbf{x}) : M \to \mathbb{R}$, where for any time $t \in \mathbb{R}$ the point $\mathbf{x}$ lies on an m-dimensional manifold $M \subseteq \mathbb{R}^d$. While in practice we do not have the the flow of the system for an time $t \in \mathbb{R}$ given by $\phi^t(\mathbf{x}) : M \times \mathbb{R} \to M$, the observation function implicitly captures the time evolution information according to $\beta(\phi^t(x))$, typically in the form of the one-dimensional, discrete and equi-spaced time series $\{\beta_n\}_{n \in \mathbb{N}}$.

Takens' theorem states that by choosing an embedding dimension $d \geq 2m + 1$, where m is the dimension of a compact manifold M, and a time lag $\tau > 0$, then the map $\Phi_{\phi,\beta} : M \to \mathbb{R}^d$ given by

$$\Phi_{\phi,\beta} = (\beta(\mathbf{x}), \beta(\phi(\mathbf{x})), \dots, \beta(\phi^{d-1}(\mathbf{x})))$$
$$= (\beta(\mathbf{x}_t), \beta(\mathbf{x}_{t+\tau}, \beta(\mathbf{x}_{t+2\tau}, \dots, \beta(\mathbf{x}_{t+(d-1)\tau}))),$$

is an embedding of M, where ϕ^{d-1} is the composition of ϕ $d-1$ times and $\mathbf{x}_t$ is the value of $\mathbf{x}$ at time t.

For noise-free data of infinite precision any time lag τ can be used; however, in practice, the choice of τ can influence the resulting embedding. In this paper τ was found by using the method of Least Median of Squares (LMS) [48] combined with the magnitude of the Fast Fourier Transform (FFT) of the signal. Specifically, we obtain the FFT spectrum and we identify the maximum significant frequency in the signal using LMS. We then use Nyquist's sampling criterion to choose the delay value according to the inequality described in [49]. This approach yielded reasonable delay values in comparison to the standard mutual information function approach [50] where the mutual information function is plotted for several values of τ and the first dip in the plot indicates the τ value to use. This is because (1) the mutual information function is not guaranteed to have a minimum, thus leading to a failed selection of τ and, (2) the identification of the first *true* dip, if it exists, is not easy to automate especially in non-smooth plots.

The embedding dimension $d \in \mathbb{N}$ is computed by using the False Nearest Neighbor (FNN) approach [51, 52]. In the FNN approach, the delay reconstruction is applied to the time series using increasing dimensions. The distances between neighboring points in one dimension are re-computed when the points are embedded into the next higher dimension. By keeping track of the percent of points that appear to be neighbors in a low dimension but are farther apart in a higher dimension (termed false neighbors), it is possible to identify a threshold that indicates that the attractor has been sufficiently unfolded. Applying FNN to all of the time series yielded the values in the range $d \in \{1, 2, \dots, 10\}$, depending on the time series being reconstructed. Upon identifying τ and d for each time series, delay reconstruction was used to embed the signal into a point cloud $P \subseteq \mathbb{R}^d$. The shape of the resulting point cloud was then quantified using persistence, as described in Section 3. We study three different methods to extract features from the resulting persistence diagrams as we show in Sections 4.2–4.6.

4.2 Persistence Landscapes

Persistence landscapes are functional summaries of persistence diagrams [53]. They are obtained by rotating persistence diagrams by 45° clockwise, and drawing isosceles right triangles for each point in the rotated

diagram [29], see Fig. 7 where the landscape functions are represented by λ_k. Given a persistence diagram, we define the piecewise linear functions [53]

$$g_{(b,d)}(x) = \begin{cases} 0 & \text{if } x \notin (b,d) \\ x - b & \text{if } x \in (b, \frac{b+d}{2}] \\ -x + d & \text{if } x \in (\frac{b+d}{2}, d) \end{cases}$$

where b and d correspond to birth and death times, respectively. Figure 7 shows that there are several landscape functions $\lambda_k(x)$ indexed by the subscript $k \in \mathbb{N}$. For example, the first landscape function $\lambda_1(x)$ is obtained by connecting the topmost values of all the functions $g_{(b_i, d_i)}(x)$ [53]. If we connect the second topmost components of $g_{(b,d)}(x)$, we obtain the second landscape function λ_2. The other landscape functions are obtained similarly. Note that the landscape functions are also piecewise linear functions.
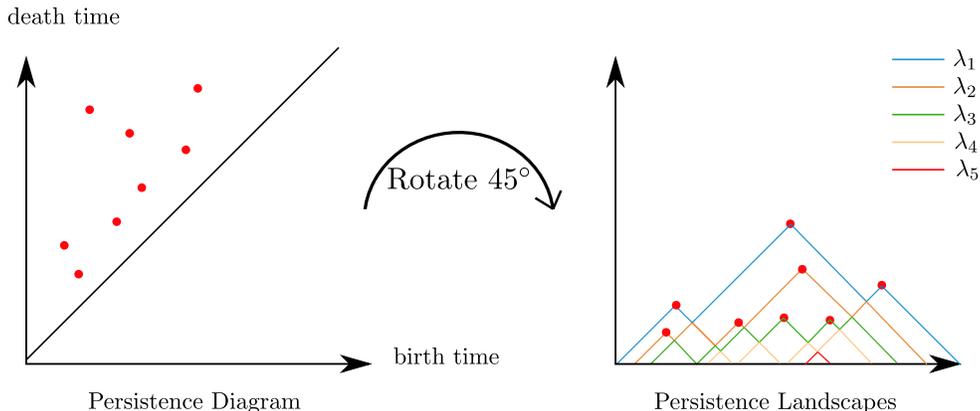

Figure 7: A schematic showing the process of obtaining the landscape functions from a persistence diagram.

4.2.1 Featurization of persistence landscapes

The persistence landscapes—$\lambda_k(x)$ where x corresponds to the birth time—were computed using the persistence diagrams obtained from each of the embedded acceleration signals. Although these persistence landscapes can be utilized to featurize the persistence diagram, there is no one way to define these features. In this paper, we choose to extract a feature vector from the persistence landscapes by defining (a) a set of length $|\mathbb{K}|$ of the landscapes $\{\lambda_k\}_{k \in \mathbb{K}}$ where $\mathbb{K} \subset \mathbb{N}$ to work with, and (b) for the kth landscape, a mesh of non-empty, distinct birth times $\mathbf{b}_k = \{x_i \in \mathbb{R}\}$ where the corresponding values of the death times $\mathbf{d}_k = \{\lambda_k(x_i) \mid x_i \in \mathbf{b}_k\}$ constitute the entries of the feature vector for the kth landscape. We can then combine features from all $|\mathbb{K}|$ landscapes to obtain the full feature vector $\mathbf{d} = \{\mathbf{d}_k\}_{k \in \mathbb{K}}$ that can be used with the machine learning algorithms.

Although the choice of $\mathbb{K}$, the set of landscapes to use, can be optimized using cross validation for example, in this study we set $\mathbb{K} = \{1, 2, \ldots, 5\}$ since it gives good results for our data. Similarly, the mesh may also be optimized in a similar way; however, this a more difficult task due to the infinite domain of $\mathbf{b}$, so we define our mesh as follows and as shown in Fig. 8.

Let $\lambda_{i,j}$ be the ith landscape corresponding to the jth persistence diagram from a training set in a supervised learning setting, see Fig. 8. Fix i and overlay the chosen landscape functions corresponding to all of the persistence diagrams in the training set. Now project all the points that define the linear pieces of each of the landscape functions onto the birth axis. Sort the projected points in ascending order and remove duplicates. The resulting set of points is our mesh $\mathbf{b}_i$ with length $|\mathbf{b}_i|$ for the ith landscape. We can repeat the same process to get the feature vector for all the $|\mathbb{K}|$ landscapes and construct the overall feature vector $\mathbf{b}$. We emphasize that a separate mesh is computed for each selected landscape number, and that the number of features will generally vary for each landscape function. Now to pull the features out of a given landscape function, we need to evaluate that function at the mesh points. Computationally, we efficiently accomplish that using piecewise linear interpolation functions.

10

Upon extracting the features from the persistence landscapes, we construct a feature matrix which represents all the tagged feature vectors. For instance, Table 2 shows an example feature matrix obtained from the first and second landscapes corresponding to each of the n persistence diagrams in the training set. This table shows data with two labels: 0 for no chatter, and 1 for chatter. It also denotes each feature with $y_{i,j}^{b_i}$ where $i \in \{1,3\}$ is the landscape number, $j \in \{1,2,\ldots,n\}$ is the corresponding persistence diagram number, while the superscript $b_i \in \{1,2,\ldots,|\mathbf{b}_i|\}$ is the feature number corresponding to the ith landscape. These feature matrices can then be used with SVM for example to train a classifier.
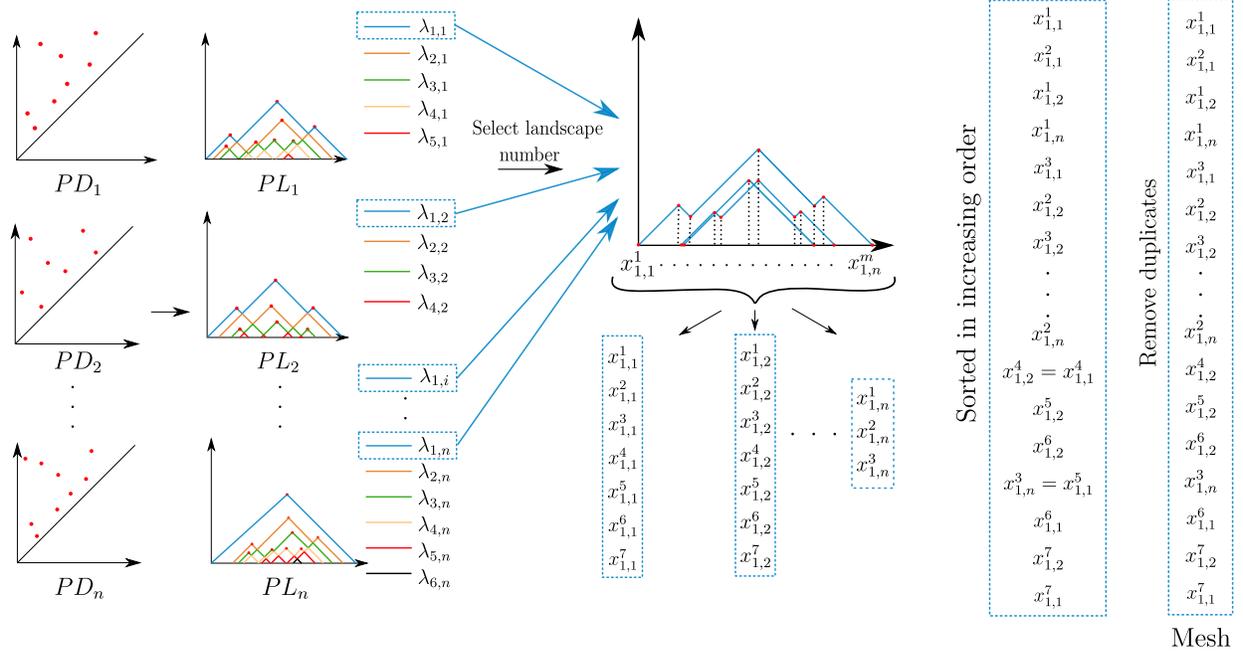
Figure 8: Persistence landscape feature extraction.

Table 2: Feature matrix for persistence landscapes λ_1 and λ_3 corresponding to persistence diagrams X_1 through X_n. The entries in the cells are the values of each of the features $y_{i,j}^{b_i}$ where i is the persistence landscape number, j is the persistence diagram number, while b_i is the feature number corresponding to the ith landscape.

Persistence Diagrams	Label	λ_1				λ_3			
X_1	1	$y_{1,1}^1$	$y_{1,1}^2$	$\cdots$	$y_{1,1}^{|\mathbf{b}_1|}$	$y_{3,1}^1$	$y_{3,1}^2$	$\cdots$	$y_{3,1}^{|\mathbf{b}_3|}$
X_2	0	$y_{1,2}^1$	$y_{1,2}^2$	$\cdots$	$y_{1,2}^{|\mathbf{b}_1|}$	$y_{3,2}^1$	$y_{3,2}^2$	$\cdots$	$y_{3,2}^{|\mathbf{b}_3|}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
X_n	1	$y_{1,n}^1$	$y_{1,n}^2$	$\cdots$	$y_{1,n}^{|\mathbf{b}_1|}$	$y_{3,n}^1$	$y_{3,n}^2$	$\cdots$	$y_{3,n}^{|\mathbf{b}_3|}$

For our data, we computed the persistence diagrams and the corresponding first five persistence landscapes for each stickout case. We then randomly split the resulting landscapes into a training set (67%) and test set (33%), created feature matrices for each of the first five landscapes separately, and used SVM with 'rbf' kernel to obtain and test a classifier. We repeated the split-train-test process 10 times, and every time new meshes were computed from the training sets and these same meshes were used with the corresponding test sets. The mean accuracy and the standard deviation of the classification computed from 10 iterations individually using each of the first 5 landscapes can be found in Table 8 in the appendix. In the results section we utilize the results with the highest accuracy for each of the stickout cases from Table 8 when comparing the different TDA-based featurization methods.

4.3 Persistence Images

Persistence images are another functional summary of persistence diagrams [22, 29]. The first step in converting a persistence diagram $X = \{(b_i, d_i) \mid i \in \{1, 2, \ldots, |X|\}\}$ to persistence images is to define the linear transformation

$$T(b_i, d_i) = (b_i, d_i - b_i) = (b_i, p_i) \tag{2}$$

which transforms the persistence diagram from the birth-death coordinates to birth-lifetime coordinates. Let $D_k(x, y) : \mathbb{R}^2 \to \mathbb{R}$ be the normalized symmetric Gaussian centered at (b_k, p_k) with standard deviation σ according to

$$D_k(x, y) = \frac{1}{2\pi\sigma^2} e^{-[(x-b_k)^2 + (y-p_k)^2]/2\sigma^2}. \tag{3}$$

It was shown in [54] that the persistence images method is not very sensitive to σ, which we set to 1 in this study. We also define a weighting function for the points in the persistence diagram $W(k) = W(b_k, p_k) : (b_k, p_k) \in T(X) \to \mathbb{R}$ according to

$$W(k) = W(b_k, p_k) = \begin{cases} 0 & \text{if } p_k \leq 0; \\ \frac{p_k}{b} & \text{if } 0 < p_k < b; \\ 1 & \text{if } p_k \geq b. \end{cases} \tag{4}$$

Note that this is not the only possible weighting function, but it satisfies the requirements needed to guarantee the stability of persistence images [22]: it vanishes along the horizontal axis, is continuous, and is piecewise differentiable. Now define the integrable persistence surface

$$S(x, y) = \sum_{k \in T(X)} W(k) \, D_k(x, y). \tag{5}$$

The surface S can be reduced to a finite dimensional vector by defining a grid over its domain and then assigning to each box (or pixel) in this grid the integral of the surface over that pixel. For example, the value over the i, j pixel in the grid is given by

$$I_{i,j}(S) = \iint S \, dx dy, \tag{6}$$

where the integral is performed over that entire pixel. The persistence image corresponding to the underlying persistence diagram X is the collection of all of the resulting pixels.

4.3.1 Featurization of persistence images

Persistence images can be used for support vector machine classification [55]. The corresponding feature vector is obtained from persistence images by concatenating the pixel values typically either row-wise or column-wise. The dimension of the resulting vector depends on the choice of the pixel size, i.e., the resolution of the persistence image. For example, let $I_{i,j}$ be a pixel in the persistence image, then a persistence image of size 100×100 pixels is represented by the matrix

$$\begin{bmatrix} I_{1,1} & I_{1,2} & \cdots & I_{1,100} \\ \vdots & \ddots & & \\ I_{100,1} & & & I_{100,100} \end{bmatrix},$$

and a typical feature vector is obtained by concatenating the entries of this matrix row-wise as shown by the rows in Table 3. The table shows a feature matrix where each persistence diagram is labeled either 0 or 1, and the corresponding feature vector is shown using entries of the form $I_{i,j}^k$, where $k \in \{1, 2, \ldots, n\}$ is the persistence diagram index while i, j are row and column numbers, respectively, in the image.

We use Python's PersistenceImages package to featurize the cutting signals, and then randomly split the resulting images into 75%-25% train-test sets. The persistence images have boundaries depending on lifetime and birth time ranges, and we trained a classifier using SVM and the 'rbf' kernel for two different pixel sizes: 0.01 and 0.1. The training and testing results for each different stickout case are available in Table 9 of the appendix. When we compare the classification accuracy for persistence images to the other featurization methods, we choose the best results from this table for each cutting configuration.

Table 3: Feature matrix for persistence images. The notation $I_{i,j}^k$ for each feature uses k to indicate the persistence diagram index, while i, j are row and column numbers, respectively, in the persistence image.

Persistence Diagrams	Label	Persistence Image									
X_1	1	$I_{1,1}^1$	$\cdots$	$I_{1,100}^1$	$I_{2,1}^1$	$\cdots$	$I_{2,100}^1$	$\cdots$	$I_{100,1}^1$	$\cdots$	$I_{100,100}^1$
X_2	0	$I_{1,1}^2$	$\cdots$	$I_{1,100}^2$	$I_{2,1}^2$	$\cdots$	$I_{2,100}^2$	$\cdots$	$I_{100,1}^2$	$\cdots$	$I_{100,100}^2$
$\vdots$	$\vdots$	$\ddots$									
X_n	1	$I_{1,1}^p$	$\cdots$	$I_{1,100}^p$	$I_{2,1}^p$	$\cdots$	$I_{2,100}^p$	$\cdots$	$I_{100,1}^p$	$\cdots$	$I_{100,100}^p$

4.4 Carlsson Coordinates

Another method for featurizing persistence diagrams is Carlsson's four Coordinates [20] with the addition of the maximum persistence [18], i.e, the highest off-diagonal point in the persistence diagram. The basic idea of Carlsson's coordinates is to utilize polynomials that (1) respect the inherent structure of the persistence diagram, and (2) that are defined on the persistence diagrams' off-diagonal points. Specifically, these polynomials must be able to accommodate persistence diagrams with different numbers of off-diagonal points since the persistence diagrams can vary in size even if the original datasets are of equal size. Further, the output of the coordinates must not depend on the order in which the off-diagonal points of a persistence diagram were stored. The resulting features can be computed directly from a persistence diagram X according to

$$
\begin{aligned}
f_1(X) &= \sum b_i(d_i - b_i), \\
f_2(X) &= \sum (d_{\max} - d_i)(d_i - b_i), \\
f_3(X) &= \sum b_i^2(d_i - b_i)^4, \\
f_4(X) &= \sum (d_{\max} - d_i)^2(d_i - b_i)^4, \\
f_5(X) &= \max\{(d_i - b_i)\}
\end{aligned}
$$

where $d_{\max}$ is the maximum death time, b_i and d_i are, respectively, the ith birth and death times, and the summations and maximum are each taken over all the points in X.

In order to utilize Carlsson coordinates, we compute the persistence diagrams from the embedded accelerometer signals and randomly split the data into training (67%) and testing (33%) sets. We then computed all five coordinates for each diagram, and utilized SVM to train a classifier. The feature vectors that we tested were all $\sum_{i=1}^{5} \binom{5}{i}$ combinations of these coordinates, where the term inside the summation is 5 choose i. This revealed which combination of features yielded the highest accuracy in each iteration. The classification results for all of the different feature vectors are reported in Tables 11–12 in the appendix. However, in the results section we utilize the feature vectors that yielded the highest accuracy when we compare the classification results of Carlsson coordinates to the other featurization methods.

4.5 Kernels for Persistence Diagrams

In addition to featurization methods, many kernel methods have also been developed for machine learning on persistence diagrams [23, 38, 39, 40, 41, 42, 43]. As an example, we choose to work with the kernel introduced by Reininghaus et al. [23] which is defined for two persistence diagrams X and Y according to

$$
\kappa_\sigma(X, Y) = \frac{1}{8\pi\sigma} \sum_{z_1 \epsilon X, z_2 \epsilon Y} \exp\left(-\frac{||z_1 - z_2||^2}{8\sigma}\right) - \exp\left(-\frac{||z_1 - \hat{z}_2||^2}{8\sigma}\right), \tag{7}
$$

where if $z = (x, y)$, then $\hat{z} = (y, x)$, and σ is a scale parameter for the kernel that can be used to tune the approach. for this study we investigated two values for this parameter: $\sigma = 0.2$ and $\sigma = 0.25$.

Given either a training or a testing set $\{X_i\}_{i=1}^N$ of labeled persistence diagrams, and using Eq. (7), we can define the kernel matrix

$$
\boldsymbol{\kappa}_\sigma = \begin{bmatrix} \kappa_\sigma(X_1, X_1) & \kappa_\sigma(X_1, X_2) & \dots & \kappa_\sigma(X_1, X_N) \\ \vdots & \ddots & & \\ \kappa_\sigma(X_N, X_1) & & & \kappa_\sigma(X_N, X_N) \end{bmatrix}.
$$

Note that given two persistence diagrams X and Y whose number of points is $|X|$ and $|Y|$, respectively, then the corresponding kernel $\kappa_\sigma(X, Y)$ can be computed in $\mathcal{O}(|F|\cdot|G|)$ time [23]. Therefore the computation time for kernel methods is generally high as shown in Table 6 can complicate optimizing the tuning parameter σ. To emphasize the effect of the computational complexity, in this paper, the long runtime for the 5.08 cm (2 inch) stickout case caused by its large number of samples has forced us to report the corresponding classification results for a smaller number of iterations than the other stickout cases and the other featurization approaches.

For our data, we performed a 67%/33% train/test split of the labeled persistence diagrams. For each of the training and testing sets, we precomputed the corresponding kernel matrices and used Python's LibSVM [56] for classification. For almost all but the 5.08 cm (2 inch) stickout case where only 1 iteration was used, we repeated the split-train-test process 10 times and recorded the average and the standard deviation of the resulting accuracies. The resulting classification accuracies are reported in Table 5. Note that [23] describes another approach for training a classifier based on a measuring the distances between two kernels in combination with a k-Nearest Neighbor (k-NN) algorithm. However we do not explore this alternative method in this work, and only perform the computations using the kernel matrix and the LibSVM library.

4.6 Persistence Paths' Signatures

Persistence paths' signatures are a recent addition to featurization tools for persistence diagrams [24]. Let $\gamma : [a, b] \to \mathbb{R}^d$ be the piecewise differentiable path given by

$$\gamma(t) = \gamma_t = [\gamma_t^1, \gamma_t^2, \ldots, \gamma_t^d], \tag{8}$$

where each $\gamma_t^i = \gamma^i(t)$ is a continuous function with $t \in [a, b]$. The first, second, and third signatures, respectively, can be defined according to the iterated integrals [35]

$$S(\gamma)_{a,t}^i = \int_a^t d\gamma_s^i = \gamma_t^i - \gamma_a^i, \qquad (a < s < t); \tag{9a}$$

$$S(\gamma)_{a,t}^{i,j} = \int_a^t S(\gamma)_{a,s}^i d\gamma_s^k = \int_a^t \int_a^s d\gamma_r^i d\gamma_s^j, \qquad (a < r < s < t); \tag{9b}$$

$$S(\gamma)_{a,t}^{i,j,\ldots,k} = \int_a^t S(\gamma)_{a,s}^{i,j,\ldots,k-1} d\gamma_s^k = \int_a^t \ldots \int_a^{t_2} d\gamma_{t_1}^i \ldots d\gamma_{t_k}^k, \qquad (a < t_1 < t_2 < \ldots < t). \tag{9c}$$

Other signatures are defined similarly, although, the computational cost significantly increases beyond the third level of signatures. The resulting path signatures can be used in classification algorithms as features. Looking back at persistence landscape functions in Section 4.2, we see that the kth landscape function $\lambda_k(t)$ can be written as a two-dimensional path

$$\gamma_t(\lambda_k(t)) = [t, \lambda_k(t)].$$

Therefore, we can obtain signatures from persistence landscapes and use them as features in machine learning algorithms [24]. In this paper we use up to the second level path signatures. Specifically, let $\lambda_{r,i}$ be the rth persistence landscape corresponding to the ith persistence diagram. Then the signatures that we use from the rth landscape function are given by $\mathbf{S}(\gamma_t(\lambda_r(t))) = [S_{r,i}^1, S_{r,i}^2, S_{r,i}^{1,1}, S_{r,i}^{1,2}, S_{r,i}^{2,1}, S_r^{2,2}]$.

By incorporating higher order signatures, or signatures from more landscape functions we can construct a longer feature vector for classification. For example, Table 4 shows the second level feature vectors computed using the first and second landscape functions for n persistence diagrams.

In our experiment, we train a classifier using 75% of the data, and we test using the remaining 25%. We construct a feature vector that includes up to the second path signatures for each of the first five landscape functions. Table 14 shows the classification accuracies for each configuration and for each landscape function. The best results in this table were used to compare the path signatures method to the other featurization procedures in Table 5.

Table 4: Feature matrix for path signatures for n persistence diagrams and using the first λ_1 and second λ_2 persistence landscapes.

Diagrams	Label	λ_1						λ_2					
X_1	1	$S_{1,1}^1$	$S_{1,1}^2$	$S_{1,1}^{1,1}$	$S_{1,1}^{1,2}$	$S_{1,1}^{2,1}$	$S_{1,1}^{2,2}$	$S_{2,1}^1$	$S_{2,1}^2$	$S_{2,1}^{1,1}$	$S_{2,1}^{1,2}$	$S_{2,1}^{2,1}$	$S_{2,1}^{2,2}$
X_2	0	$S_{1,2}^1$	$S_{1,2}^2$	$S_{1,2}^{1,1}$	$S_{1,2}^{1,2}$	$S_{1,2}^{2,1}$	$S_{1,2}^{2,2}$	$S_{2,2}^1$	$S_{2,2}^2$	$S_{2,2}^{1,1}$	$S_{2,2}^{1,2}$	$S_{2,2}^{2,1}$	$S_{2,2}^{2,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
X_n	1	$S_{1,n}^1$	$S_{1,n}^2$	$S_{1,n}^{1,1}$	$S_{1,n}^{1,2}$	$S_{1,n}^{2,1}$	$S_{1,n}^{2,2}$	$S_{2,n}^1$	$S_{2,n}^2$	$S_{2,n}^{1,1}$	$S_{2,n}^{1,2}$	$S_{2,n}^{2,1}$	$S_{2,n}^{2,2}$

5 Results and Discussion

This section presents the classification accuracies for all the methods that are introduced in Section 4 and compares them to the results in Ref. [10] which uses the Wavelet Packet Transform (WPT) and the Ensemble Empirical Mode Decomposition (EEMD). The latter two methods are used for comparison since they are some of the currently most prominent methods for chatter identification using supervised learning. Table 5 also includes the classification results obtained using a new TDA approach, which is not included in Section 4, based on template functions [36]. The classification results are summarized in Table 5 where for each cutting configuration the best classification results for each method are included. In this table, the best accuracy for each dataset is highlighted in green. Further, methods whose accuracy are within one standard deviation of the best result in the same category are highlighted in blue.

Table 5 shows that the WPT approach yields the highest classification accuracy for the 5.08 cm (2 inch) and the 6.35 cm (2.5 inch) stickout cases. However, we also see that for the 5.08 cm (2 inch) case, three of the TDA-based methods (persistence landscapes, template functions, and Carlsson coordinates) yield comparable results, i.e., accuracies that are within 1 standard deviation of the WPT results, see [10]. In fact, for this cutting configuration the average accuracy using persistence landscapes is only 1% lower than its WPT counterpart.

For the 6.35 cm (2.5 inch) stickout case we point out that the number of time series is small in comparison to the other cutting configurations. Specifically, for this case, less than 10 time series were used as the test set, see Table 1. Therefore, the 100% classification accuracy using WPT for this case does not represent a robust result. Nevertheless, Table 5 shows that the TDA methods based on template functions, Carlsson coordinates, and persistence paths yield better results than EEMD—a leading approach for chatter detection.

For the 8.89 cm (3.5 inch) case, Carlsson coordinates method yields the highest mean accuracy of 93%, placing ahead of both WPT and EEMD. Further, the other TDA-based method for this cutting configuration score classification accuracies of at least 83%.

The EEMD method gives the highest mean accuracy for the 11.43 cm (4.5 inch) stick out case with an accuracy of 79.1%. However, both template functions and persistence paths yield mean accuracies of 78.1% and 70%, which are within one standard deviation of the mean accuracy of EEMD. In fact, template functions' accuracy of 78.1% is within 1% of its EEM counterpart.

Table 5: Comparison of results for each method where WPT is the Wavelet Packet Transform, and EEMD stand for Ensemble Empirical Mode Decomposition.

Stickout Length cm (inch)	Persistence Landscapes	Persistence Images	Template Functions	Carlsson Coordinates	Kernel Method	Persistence Paths	WPT	EEMD
5.08 (2)	92.3 %	76.1 %	89.6 %	88.7 %	74.5 %*	83.0 %	93.9 %	84.2 %
6.35 (2.5)	70.2 %	66.7 %	87.7 %	86.3 %	58.9 %	84.2 %	100.0 %	78.6 %
8.89 (3.5)	84.8 %	83.5 %	84.3 %	93.0 %	87.0 %	85.9 %	84.0 %	90.7 %
11.43 (4.5)	66.3 %	66.4 %	78.1 %	65.9 %	59.3 %	70.0 %	78.8 %	79.1 %

*This result belongs to only the first iteration for the 5.08 cm (2 inch) stickout case.

Runtime is another criterion for comparing the different feature extraction methods. In Table 6 we compare the runtime for each of the methods. For the TDA-based methods, the numbers reported in the table include the total runtime required for classification including: (1) obtaining the persistence diagrams, (2) obtain features or computing kernels, and (3) training and testing the corresponding classifier. These comparisons were performed using a Dell Optilex 7050 desktop with Intel Core i7-7700 CPU and 16.0 GB RAM.

Table 6: Comparison of time (seconds) required to obtain results for each method. WPT denotes the Wavelet Packet Transform, while EEMD indicates Ensemble Empirical Mode Decomposition.

Stickout Length cm (inch)	Persistence Landscapes	Persistence Images	Template Functions	Carlsson Coordinates	Kernel Method	Persistence Paths	WPT	EEMD
5.08 (2)	98582	85601	84364	84352	1466747	118403	116	14540
6.35 (2.5)	118391	23930	23756	23752	153759	30563	37	3372
8.89 (3.5)	25527	11437	11322	11320	71553	14401	5	1583
11.43 (4.5)	80462	37966	37623	37619	542378	48958	7	3096

Table 6 shows that feature extraction and classification with WPT method is the fastest. However, the runtime reported for the WPT (and the EEMD) method in the table does not tell the full story. Specifically, both WPT and EEMD methods require the identification of informative wavelet packets or intrinsic mode functions before the classification step. These informative wavelet packets are obtained by manually performing frequency domain analysis for each cutting configuration. Therefore, for each stickout case at least a portion of the dataset must be manually analyzed to identify the informative wavelet packets [10], but the runtime reported in Table 6 does not include the significant amount of time involved in this effort. Further the WPT and EEMD methods use code that has been highly optimized, whereas the TDA-based methods are still under active research with huge future potential for improved parallelization and optimization.

Table 6 also shows that the full persistence pipeline can include large runtimes. For example, kernel methods are computationally the most expensive which is why we split the data into training and test sets three times to obtain their mean classification score. Even then, for the 5.08 cm (2 inch) case, we only provide the result of the first iteration due the associating computational difficulties. Carlsson coordinates and template functions are the fastest TDA-based methods, followed by persistence images, persistence landscapes, and persistence paths.

Despite the relatively long computation time for persistence-based methods, we note that after obtaining the persistence diagrams, they can be saved and used in multiple TDA-based classification methods. Therefore, once these diagrams are computed, the time required for featurization and classification would be a fraction of the the ones reported in Table 6. Another point we wish to emphasize is that the most computationally expensive step is that of training a classifier. Once a classifier is trained, which can be done offline, the effort in classifying incoming streams of data is much smaller because a much smaller set of persistence diagrams and features are needed. This opens the possibility for exploring in-situ chatter detection using TDA-based methods especially with properly optimized algorithms.

6 Conclusion

Two of the most common methods for chatter detection are based on extracting features from the time series using the WPT and the EEMD methods. However, even after obtaining a tagged time series, utilizing either of these methods requires the analysis of at least a subset of the available time series to identify which parts of the signal's decomposition are the most informative when training a classifier. In addition to requiring highly trained individuals to perform this critical step, both WPT and EEMD are then locked into a small set of informative packets or IMFs which limits the transfer learning ability of the resulting classifiers [10].

In contrast to the WPT and EEMD methods, we use persistence homology—the most prominent analysis tool from TDA—to obtain a summary of the persistent topological features of the data. These are based on the global structure of the point cloud embedding of the acceleration signals in a turning experiment; therefore, upon obtaining a persistence diagram, there is no manual work involved in selecting the features from the persistence diagram. Since working directly with the resulting persistence diagrams is difficult, we investigated using the leading tools for feature extraction from persistence diagrams. The featurization methods that we studied are based on persistence landscapes, persistence images, Carlsson Coordiantes, a kernel method, template functions, and persistence paths' signatures. The resulting features are then combined with an SVM algorithm for training a classifier. The classification results are then computed from multiple split-train-test sets, and the resulting mean accuracies as well as the corresponding standard deviation are recorded for each method and for each cutting configuration.

Tables 5 and 6summarize the classification accuracy and the average runtime of all the TDA-based tools as well as their WPT and EEMD counterparts. The table shows that overall TDA-based methods yield high classification accuracies that often match or exceed the traditional classification methods. In terms of the classification accuracy across the different cutting configurations we note the Carlsson coordinates and the template functions approaches yield the best accuracies and have the smallest runtime in comparison to the other TDA tools.

In two out of the three cases where either WPT or EEMD yielded the best accuracies, we note that the corresponding error bars enclose the results obtained from TDA-based methods. Specifically, Table 5 shows that for the 5.08 cm (2 inch) case, persistence landscapes, template functions and Carlsson coordinates yield accuracies that are off by 2%, 5%, and 5.5%, respectively, from the WPT result. Similarly, for the 11.43 cm (4.5 inch) case, template functions and persistence paths are within 1% and 10% of the EEMD result. For the 8.89 cm (3.5 inch) stickout case, Carlsson coordinates yield the highest accuracy scoring 93% with tight error bounds that do not enclose the EEMD accuracy of 90.7%. Table 5 also shows that WPT yields an accuracy of 100% (with a standard deviation of zero) for the 6.35 cm (2.5 inch) stickout case; however, as pointed out in Section 5 and Table 1, the size of the test set for this cutting configuration is too small which casts some doubts about the robustness of this result. Nevertheless, for this case both template functions and Carlsson coordinates still yield at least 86% classification accuracy.

As for runtime comparisons, Table 6 shows that WPT is the fastest followed by EEMD; however, the reported time for these two methods does not include the substantial time involved in identifying and extracting the informative packets in WPT and the informative IMFs in EEMD. The TDA approach completely bypasses this step, and so there is no overhead that needs to be added to the runtime of these methods. Out of all the TDA methods, Carlsson coordinates and template functions are the fastest, followed by followed by, in ascending order, persistence images, persistence landscapes, persistence paths, and kernel methods. Note that these runtimes do not involve any parallel computing or optimization, which we anticipate will dramatically decrease the runtime especially for the kernel method.

To summarize, we show that persistence features are appropriate for chatter detection in cutting processes. These features have the potential to lower the barrier to entry when tagging cutting signals as chatter or chatter free because no manual pre-processing is needed before extracting and using the features in the persistence diagram. We also note that after obtaining a classifier, the time required for the classification of new incoming data will be greatly reduced, thus opening the door for future implementation of TDA methods in-situ for chatter detection and mitigation.

References

[1] T. Thaler, P. Potočnik, I. Bric, and E. Govekar, "Chatter detection in band sawing based on discriminant analysis of sound features," *Applied Acoustics*, vol. 77, pp. 114–121, mar 2014.

[2] G. S. Chen and Q. Z. Zheng, "Online chatter detection of the end milling based on wavelet packet transform and support vector machine recursive feature elimination," *The International Journal of Advanced Manufacturing Technology*, vol. 95, pp. 775–784, nov 2017.

[3] Y. Ji, X. Wang, Z. Liu, H. Wang, L. Jiao, D. Wang, and S. Leng, "Early milling chatter identification by improved empirical mode decomposition and multi-indicator synthetic evaluation," *Journal of Sound and Vibration*, vol. 433, pp. 138–159, oct 2018.

[4] H. Liu, Q. Chen, B. Li, X. Mao, K. Mao, and F. Peng, "On-line chatter detection using servo motor current signal in turning," *Science China Technological Sciences*, vol. 54, pp. 3119–3129, nov 2011.

[5] L. Ding, Y. Sun, and Z. Xiong, "Early chatter detection based on logistic regression with time and frequency domain features," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE, jul 2017.

[6] Y. Fu, Y. Zhang, H. Qiao, D. Li, H. Zhou, and J. Leopold, "Analysis of feature extracting ability for cutting state monitoring using deep belief networks," *Procedia CIRP*, vol. 31, pp. 29–34, 2015.

[7] M. Lamraoui, M. Barakat, M. Thomas, and M. E. Badaoui, "Chatter detection in milling machines by neural network classification and feature selection," *Journal of Vibration and Control*, vol. 21, pp. 1251–1266, aug 2013.

[8] Z. Han, H. Jin, D. Han, and H. Fu, "ESPRIT- and HMM-based real-time monitoring and suppression of machining chatter in smart CNC milling system," *The International Journal of Advanced Manufacturing Technology*, vol. 89, pp. 2731–2746, dec 2016.

[9] F.-Y. Xie, Y.-M. Hu, B. Wu, and Y. Wang, "A generalized hidden markov model and its applications in recognition of cutting states," *International Journal of Precision Engineering and Manufacturing*, vol. 17, pp. 1471–1482, nov 2016.

[10] M. C. Yesilli, F. A. Khasawneh, and A. Otto, "On transfer learning for chatter detection in turning using wavelet packet transform and empirical mode decomposition," *arXiv:1905.01982*, 2019.

[11] R. Ghrist, "Barcodes: The persistent topology of data," *Builletin of the American Mathematical Society*, vol. 45, pp. 61–75, 2008. Survey.

[12] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society*, vol. 46, pp. 255–308, Jan. 2009. Survey.

[13] H. Edelsbrunner and J. L. Harer, *Computational topology: an introduction.* American Mathematical Society, 2009.

[14] S. Y. Oudot, *Persistence theory: from quiver representations to data analysis*, vol. 209 of *AMS Mathematical Surveys and Monographs.* American Mathematical Society, 2015.

[15] M. Robinson, *Topological Signal Processing.* Springer-Verlag Berlin Heidelberg, 1 ed., 2014.

[16] F. A. Khasawneh and E. Munch, "Stability of a stochastic turning model using persistent homology." In submission, 2014.

[17] F. A. Khasawneh and E. Munch, "Chatter detection in turning using persistent homology," *Mechanical Systems and Signal Processing*, vol. 70-71, pp. 527–541, 2016.

[18] F. A. Khasawneh, E. Munch, and J. A. Perea, "Chatter classification in turning using machine learning and topological data analysis," *IFAC-PapersOnLine*, vol. 51, no. 14, pp. 195–200, 2018.

[19] F. A. Khasawneh and E. Munch, "Stability determination in turning using persistent homology and time series analysis," in *Proceedings of the ASME 2014 International Mechanical Engineering Congress & Exposition, November 14-20, 2014, Montreal, Canada*, 2014. Paper no. IMECE2014-40221.

[20] A. Adcock, E. Carlsson, and G. Carlsson, "The ring of algebraic functions on persistence bar codes," *Homology, Homotopy and Applications*, vol. 18, no. 1, pp. 381–402, 2016.

[21] P. Bubenik, "Statistical topological data analysis using persistence landscapes," *Journal of Machine Learning Research*, vol. 16, pp. 77–102, 2015.

[22] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier, "Persistence images: A stable vector representation of persistent homology," *J. Mach. Learn. Res.*, vol. 18, pp. 218–252, Jan. 2017.

[23] J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt, "A stable multi-scale kernel for topological machine learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[24] I. Chevyrev, V. Nanda, and H. Oberhauser, "Persistence paths and signature features in topological data analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[25] C. Li, M. Ovsjanikov, and F. Chazal, "Persistence-based structural recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1995–2002, 2014.

[26] Y. Mileyko, S. Mukherjee, and J. Harer, "Probability measures on the space of persistence diagrams," *Inverse Problems*, vol. 27, no. 12, p. 124007, 2011.

[27] K. Turner, Y. Mileyko, S. Mukherjee, and J. Harer, "Fréchet means for distributions of persistence diagrams," *Discrete & Computational Geometry*, vol. 52, no. 1, pp. 44–70, 2014.

[28] E. Munch, K. Turner, P. Bendich, S. Mukherjee, J. Mattingly, and J. Harer, "Probabilistic fréchet means for time varying persistence diagrams," *Electron. J. Statist.*, vol. 9, pp. 1173–1204, 2015.

[29] E. Berry, Y.-C. Chen, J. Cisewski-Kehe, and B. T. Fasy, "Functional summaries of persistence diagrams,"

[30] F. Chazal, B. T. Fasy, F. Lecci, A. Rinaldo, and L. Wasserman, "Stochastic convergence of persistence landscapes and silhouettes," in *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, (New York, NY, USA), pp. 474:474–474:483, ACM, 2014.

[31] S. Chepushtanova, T. Emerson, E. Hanson, M. Kirby, F. Motta, R. Neville, C. Peterson, P. Shipman, and L. Ziegelmeier, "Persistence images: An alternative persistent homology representation," *arXiv preprint arXiv:1507.06217*, 2015.

[32] Y.-C. Chen, D. Wang, A. Rinaldo, and L. Wasserman, "Statistical analysis of persistence intensity functions," *arXiv preprint arXiv:1510.02502*, 2015.

[33] P. Donatini, P. Frosini, and A. Lovato, "Size functions for signature recognition," in *Vision Geometry VII* (R. A. Melter, A. Y. Wu, and L. J. Latecki, eds.), SPIE, oct 1998.

[34] M. Ferri, P. Frosini, A. Lovato, and C. Zambelli, "Point selection: A new comparison scheme for size functions (with an application to monogram recognition)," in *ACCV*, 1998.

[35] I. Chevyrev and A. Kormilitzin, "A primer on the signature method in machine learning,"

[36] J. A. Perea, E. Munch, and F. A. Khasawneh, "Approximating continuous functions on persistence diagrams using template functions,"

[37] L. Fox and I. B. Parker, *Chebyshev Polynomials in Numerical Analysis.* London: Oxford Univ. Press, 1968.

[38] R. Kwitt, S. Huber, M. Niethammer, W. Lin, and U. Bauer, "Statistical topological data analysis - a kernel perspective," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett, and R. Garnett, eds.), pp. 3052–3060, Curran Associates, Inc., 2015.

[39] Q. Zhao and Y. Wang, "Learning metrics for persistence-based summaries and applications for graph classification,"

[40] G. Kusano, Y. Hiraoka, and K. Fukumizu, "Persistence weighted gaussian kernel for topological data analysis," in *International Conference on Machine Learning*, pp. 2004–2013, 2016.

[41] G. Kusano, K. Fukumizu, and Y. Hiraoka, "Kernel method for persistence diagrams via kernel embedding and weight factor," 2017.

[42] M. Carrire, M. Cuturi, and S. Oudot, "Sliced wasserstein kernel for persistence diagrams," *arXiv:1706.03358*, 2017.

[43] G. Kusano, "Persistence weighted gaussian kernel for probability distributions on the space of persistence diagrams," 2018.

[44] T. Insperger, B. P. Mann, T. Surmann, and G. Stépán, "On the chatter frequencies of milling processes with runout," *International Journal of Machine Tools and Manufacture*, vol. 48, pp. 1081–1089, aug 2008.

[45] E. Munch, "A user's guide to topological data analysis," *Journal of Learning Analytics*, vol. 4, pp. 47–61, jul 2017.

[46] S. Theodoridis and K. Koutroumbas, "Feature selection," in *Pattern Recognition*, pp. 261–322, Elsevier, 2009.

[47] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980* (D. Rand and L.-S. Young, eds.), vol. 898 of *Lecture Notes in Mathematics*, pp. 366–381, Springer Berlin Heidelberg, 1981.

[48] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American Statistical Association*, vol. 79, pp. 871–880, dec 1984.

[49] M. Melosik and W. Marszalek, "On the 0/1 test for chaos in continuous systems," *Bulletin of the Polish Academy of Sciences Technical Sciences*, vol. 64, pp. 521–528, sep 2016.

[50] A. M. Fraser and H. L. Swinney, "Independent coordinates for strange attractors from mutual information," *Phys. Rev. A*, vol. 33, pp. 1134–1140, Feb 1986.

[51] M. B. Kennel, R. Brown, and H. D. I. Abarbanel, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Physical Review A*, vol. 45, pp. 3403–3411, mar 1992.

[52] H. D. I. Abarbanel, T. A. Carroll, L. M. Pecora, J. J. Sidorowich, and L. S. Tsimring, "Predicting physical variables in time-delay embedding," *Physical Review E*, vol. 49, pp. 1840–1853, mar 1994.

[53] P. Bubenik and P. Dłotko, "A persistence landscapes toolbox for topological statistics," *Journal of Symbolic Computation*, vol. 78, pp. 91–114, jan 2017.

[54] M. Zeppelzauer, B. Zieliński, M. Juda, and M. Seidl, "A study on topological descriptors for the analysis of 3d surface texture," *Computer Vision and Image Understanding*, vol. 167, pp. 74–88, feb 2018.

[55] S. Chepushtanova, T. Emerson, E. Hanson, M. Kirby, F. Motta, R. Neville, C. Peterson, P. Shipman, and L. Ziegelmeier, "Persistence images: An alternative persistent homology representation,"

[56] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Appendices

A   Expressions for persistence paths' signatures

Let the kth landscape functions be $\lambda_k(x) = y$ where x and y represent coordinates along the birth time and the persistence axes. Since the persistence landscapes are piecewise linear functions, we can write them in closed form in terms of the nodes $\{x_i, y_i\}_{i=1}^n$ that define the boundaries of each of their linear pieces according to

$$\lambda_k(t) = \begin{cases} \frac{y_{i+1}-y_i}{x_{i+1}-x_i}t + \frac{y_{i+1}(x_{i+1}-x_i)+x_{i+1}(y_i-y_{i+1})}{x_{i+1}-x_i}, & \text{for } i \in \{1, 2, \ldots, n\} \text{ and } t \in [x_1, x_n], \\ 0 & \text{otherwise .} \end{cases} \tag{10}$$

The corresponding path is given by

$$P = [P_t^1, P_t^2] = \left[t, \frac{y_{i+1} - y_i}{x_{i+1} - x_i}t + \frac{y_{i+1}(x_{i+1} - x_i) + x_{i+1}(y_i - y_{i+1})}{x_{i+1} - x_i} \right], \tag{11}$$

and its differential is

$$dP = [dP_t^1, dP_t^2] = [dt, \frac{y_{i+1} - y_i}{x_{i+1} - x_i}dt]. \tag{12}$$

Using the above definitions, we can derive general expressions for the first and second level signatures, respectively, according to

$$S_1 = \int_{x_1}^{x_n} dP_t^1 = \int_{x_1}^{x_n} dt = x_n - x_1 \tag{13a}$$

$$S_2 = \int_{x_1}^{x_n} dP_t^2 = \int_{x_1}^{x_2} \frac{y_2 - y_1}{x_2 - x_1}dt + \ldots + \int_{x_i}^{x_{i+1}} \frac{y_{i+1} - y_i}{x_{i+1} - x_i}dt + \ldots + \int_{x_{n-1}}^{x_n} \frac{y_n - y_{n-1}}{x_n - x_{n-1}} \tag{13b}$$

The equations for the second level signatures is provided in Table 7.

Table 7: Closed-form expressions for the first and second level signature paths for landscape functions.

Signature	Equation
S_1	$\displaystyle \int_{x_1}^{x_n} dP_t^1 = \int_{x_1}^{x_n} dt = x_n - x_1$
S_2	$\displaystyle \int_{x_1}^{x_n} dP_t^2 = \int_{x_1}^{x_2} \frac{y_2 - y_1}{x_2 - x_1} dt + \ldots + \int_{x_i}^{x_{i+1}} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} dt + \ldots + \int_{x_{n-1}}^{x_n} \frac{y_n - y_{n-1}}{x_n - x_{n-1}}$
$S_{1,1}$	$\displaystyle \int\int dP_t^1 dP_t^1 = \int_{x_1}^{x_n} \left(\int_{x_1}^{t_2} dt_1 \right) dt_2 = \int_{x_1}^{x_n} (t_2 - x_1) dt_2 = \left(\frac{t_2^2}{2} - x_1 t_2 \right) \Bigg|_{t_2=x_1}^{t_2=x_n}$
$S_{1,2}$	$\displaystyle \int\int dP_t^1 dP_t^2 = \int_{x_1}^{x_n} \left(\int_{x_1}^{t_2} dt_1 \right) dP_t^2 dt_2 = \int_{x_1}^{x_2} (t_2 - x_1) \frac{y_2 - y_1}{x_2 - x_1} dt_2 + \ldots + \int_{x_i}^{x_{i+1}} (t_2 - x_1) \frac{y_{i+1} - y_i}{x_{i+1} - x_i} dt_2 + \ldots + \int_{x_{n-1}}^{x_n} (t_2 - x_1) \frac{y_n - y_{n-1}}{x_n - x_{n-1}} dt_2$
$S_{2,1}$	$\displaystyle \int\int dP_t^2 dP_t^1 = \int_{x_1}^{x_n} \left(\int_{x_1}^{t_2} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} dt_1 \right) dt_2 = \int_{x_1}^{x_n} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \left(\int_{x_1}^{t_2} dt_1 \right) dt_2 + \ldots + \int_{x_i}^{x_{i+1}} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \left(\int_{x_i}^{t_2} dt_1 \right) dt_2 + \ldots + \int_{x_{n-1}}^{x_n} \frac{y_n - y_{n-1}}{x_n - x_{n-1}} \left(\int_{x_n}^{t_2} dt_1 \right) dt_2$ $\displaystyle = \int_{x_1}^{x_2} \frac{y_2 - y_1}{x_2 - x_1} \left(\int_{x_1}^{t_2} dt_1 \right) dt_2 + \ldots$
$S_{2,2}$	$\displaystyle \int\int dP_t^2 dP_t^2 = \int_{x_1}^{x_n} \left(\int_{x_1}^{t_2} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} dt_1 \right) \frac{y_{i+1} - y_i}{x_{i+1} - x_i} dt_2 = \int_{x_1}^{x_n} \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right)^2 \left(\int_{x_1}^{t_2} dt_1 \right) dt_2$ $\displaystyle = \int_{x_1}^{x_2} \left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 \left(\int_{x_1}^{t_2} dt_1 \right) dt_2 + \ldots + \int_{x_i}^{x_{i+1}} \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right)^2 \left(\int_{x_i}^{t_2} dt_1 \right) dt_2 + \ldots + \int_{x_{n-1}}^{x_n} \left(\frac{y_n - y_{n-1}}{x_n - x_{n-1}} \right)^2 \left(\int_{x_n}^{t_2} dt_1 \right) dt_2$

B Classification results

Table 8: Persistence Landscape Results with SVM classifier. Landscape number column represents which landscapes was used to extract features.

Landscape Number	2 inch		2.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	92.3 % ± 5.2 %	92.8 % ± 3.9 %	70.2 % ± 11.1 %	74.9 % ± 10.3 %
2	74.5 % ± 1.1 %	74.8 % ± 1.1 %	63.0 % ± 4.2 %	63.1 % ± 2.1 %
3	73.7 % ± 1.7 %	75.2 % ± 0.9 %	65.1 % ± 7.8 %	62.1 % ± 3.8 %
4	74.3 % ± 0.9 %	74.9 % ± 0.4 %	63.5 % ± 6.5 %	62.9 % ± 3.2 %
5	73.9 % ± 2.0 %	75.1 % ± 1.0 %	62.8 % ± 6.5 %	63.2 % ± 3.2 %
Landscape Number	3.5 inch		4.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	84.8 % ± 8.1 %	80.7 % ± 4.2 %	63.6 % ± 4.6 %	65.4 % ± 2.3 %
2	80.0 % ± 7.6 %	83.2 % ± 4.0 %	64.9 % ± 6.0 %	64.7 % ± 3.0 %
3	84.3 % ± 4.0 %	80.9 % ± 2.1 %	64.1 % ± 2.7 %	65.1 % ± 1.4 %
4	83.9 % ± 6.7 %	81.1 % ± 3.5 %	64.4 % ± 4.5 %	65.0 % ± 2.3 %
5	83.9 % ± 6.5 %	81.1 % ± 3.4 %	66.3 % ± 4.8 %	64.0 % ± 2.4 %

Table 9: Persistence Images Results with SVM classifier.

Image Resolution	2 inch		2.5 inch	
	Test Set	Training Set	Test Set	Training Set
0.01	76.1 % ± 2.9 %	74.2 % ± 1.0 %	65.5 % ± 4.5 %	62.3 % ± 1.5 %
0.10	74.8 % ± 3.7 %	74.7 % ± 1.2 %	66.7 % ± 10.1 %	61.9 % ± 3.5 %
Image Resolution	3.5 inch		4.5 inch	
	Test Set	Training Set	Test Set	Training Set
0.01	81.2 % ± 9.8 %	82.4 % ± 3.3 %	62.3 % ± 6.7 %	65.6 % ± 2.2 %
0.10	83.5 % ± 12.0 %	81.6 % ± 4.1 %	66.4 % ± 6.9 %	64.2 % ± 2.3 %

Table 10: Template Function Results with SVM classifier.

Persistence Diagram Dimension	2 inch	2.5 inch	3.5 inch	4.5 inch
	Test Set			
0	82.3 % ± 1.6 %	82.6 % ± 4.3 %	78.7 % ± 10.6 %	66.6 % ± 5.7 %
1	89.6 % ± 1.2 %	87.7 % ± 6.0 %	84.3 % ± 7.6 %	78.1 % ± 3.7 %

Table 11: 2 inch and 2.5 inch stickout case Carlsson Coordinates results obtained with SVM

Feature Combination	2 inch		2.5 inch	
	Test Set Score	Training Set Score	Test Set Score	Training Set Score
1	$74.3\% \pm 2.4\%$	$82.8\% \pm 0.8\%$	$79.1\% \pm 4.0\%$	$94.9\% \pm 1.2\%$
2	$73.4\% \pm 2.1\%$	$85.9\% \pm 0.7\%$	$82.1\% \pm 6.1\%$	$96.0\% \pm 1.4\%$
3	$82.0\% \pm 1.7\%$	$82.9\% \pm 0.8\%$	$82.8\% \pm 4.6\%$	$86.8\% \pm 2.4\%$
4	$73.8\% \pm 1.8\%$	$77.6\% \pm 0.8\%$	$86.3\% \pm 5.5\%$	$86.1\% \pm 2.7\%$
5	$87.4\% \pm 1.8\%$	$87.1\% \pm 0.8\%$	$82.8\% \pm 4.8\%$	$82.1\% \pm 2.3\%$
1-2	$73.4\% \pm 1.9\%$	$96.7\% \pm 0.7\%$	$78.6\% \pm 6.1\%$	$98.9\% \pm 1.1\%$
1-3	$79.8\% \pm 2.3\%$	$91.2\% \pm 1.2\%$	$80.9\% \pm 3.7\%$	$94.7\% \pm 1.2\%$
1-4	$75.4\% \pm 1.9\%$	$86.1\% \pm 1.1\%$	$81.1\% \pm 3.7\%$	$94.7\% \pm 1.2\%$
1-5	$84.3\% \pm 2.3\%$	$91.2\% \pm 1.4\%$	$81.4\% \pm 1.4\%$	$94.7\% \pm 1.2\%$
2-3	$75.6\% \pm 2.6\%$	$93.1\% \pm 1.1\%$	$80.7\% \pm 5.8\%$	$96.0\% \pm 1.1\%$
2-4	$73.7\% \pm 2.7\%$	$87.4\% \pm 0.8\%$	$82.3\% \pm 5.8\%$	$95.7\% \pm 1.2\%$
2-5	$77.6\% \pm 2.2\%$	$91.7\% \pm 1.0\%$	$82.1\% \pm 5.8\%$	$95.7\% \pm 1.2\%$
3-4	$81.4\% \pm 1.2\%$	$87.3\% \pm 0.6\%$	$83.0\% \pm 4.7\%$	$86.8\% \pm 2.4\%$
3-5	$88.7\% \pm 1.4\%$	$91.7\% \pm 1.0\%$	$83.0\% \pm 4.7\%$	$86.8\% \pm 2.4\%$
4-5	$86.5\% \pm 1.6\%$	$90.8\% \pm 1.0\%$	$86.3\% \pm 5.4\%$	$86.1\% \pm 2.7\%$
1-2-3	$73.9\% \pm 2.1\%$	$98.5\% \pm 0.8\%$	$78.4\% \pm 6.1\%$	$98.6\% \pm 1.1\%$
1-2-4	$72.9\% \pm 1.9\%$	$95.8\% \pm 0.9\%$	$78.4\% \pm 6.1\%$	$98.6\% \pm 1.1\%$
1-2-5	$74.4\% \pm 2.4\%$	$97.1\% \pm 1.0\%$	$78.1\% \pm 6.4\%$	$98.6\% \pm 1.1\%$
1-3-4	$79.9\% \pm 2.3\%$	$93.1\% \pm 0.8\%$	$82.1\% \pm 3.8\%$	$94.8\% \pm 1.1\%$
1-3-5	$82.5\% \pm 2.5\%$	$92.7\% \pm 1.0\%$	$82.3\% \pm 3.8\%$	$94.8\% \pm 1.1\%$
1-4-5	$83.6\% \pm 2.0\%$	$94.0\% \pm 0.6\%$	$82.3\% \pm 4.1\%$	$94.7\% \pm 1.2\%$
2-3-4	$75.9\% \pm 2.4\%$	$93.1\% \pm 1.2\%$	$80.5\% \pm 5.5\%$	$95.2\% \pm 1.5\%$
2-3-5	$76.8\% \pm 2.6\%$	$93.9\% \pm 1.1\%$	$80.2\% \pm 5.7\%$	$95.2\% \pm 1.5\%$
2-4-5	$76.1\% \pm 2.7\%$	$91.3\% \pm 0.8\%$	$81.6\% \pm 5.8\%$	$95.3\% \pm 1.3\%$
3-4-5	$85.6\% \pm 1.8\%$	$92.4\% \pm 1.2\%$	$83.0\% \pm 4.7\%$	$86.8\% \pm 2.4\%$
1-2-3-4	$73.8\% \pm 2.0\%$	$98.2\% \pm 0.9\%$	$79.1\% \pm 6.0\%$	$98.4\% \pm 1.1\%$
1-2-3-5	$73.9\% \pm 2.1\%$	$98.5\% \pm 0.8\%$	$78.8\% \pm 6.2\%$	$98.4\% \pm 1.1\%$
1-2-4-5	$73.4\% \pm 2.0\%$	$96.5\% \pm 0.9\%$	$78.8\% \pm 6.2\%$	$98.4\% \pm 1.1\%$
1-3-4-5	$82.1\% \pm 2.2\%$	$93.6\% \pm 0.8\%$	$82.3\% \pm 4.1\%$	$94.7\% \pm 1.1\%$
2-3-4-5	$77.3\% \pm 2.4\%$	$93.7\% \pm 0.8\%$	$81.6\% \pm 5.5\%$	$94.6\% \pm 1.4\%$
1-2-3-4-5	$73.9\% \pm 2.0\%$	$98.1\% \pm 0.8\%$	$79.3\% \pm 6.3\%$	$98.3\% \pm 1.1\%$

Table 12: 3.5 inch and 4.5 inch stickout case Carlsson Coordinates results obtained with SVM

Feature Combination	3.5 inch		4.5 inch	
	Test Set Score	Training Set Score	Test Set Score	Training Set Score
1	81.3 % ± 5.9 %	91.1 % ± 2.6 %	64.4 % ± 5.7 %	84.0 % ± 1.8 %
2	81.3 % ± 4.4 %	97.1 % ± 2.1 %	62.4 % ± 5.1 %	90.3 % ± 2.5 %
3	90.4 % ± 5.8 %	91.4 % ± 3.0 %	63.6 % ± 4.5 %	67.4 % ± 2.4 %
4	83.5 % ± 5.8 %	86.4 % ± 1.4 %	65.6 % ± 4.7 %	68.2 % ± 2.4 %
5	93.0 % ± 4.4 %	95.7 % ± 2.6 %	65.3 % ± 5.1 %	64.5 % ± 2.6 %
1-2	83.5 % ± 5.8 %	100.0 % ± 0.0 %	65.3 % ± 4.7 %	99.6 % ± 0.4 %
1-3	83.0 % ± 5.3 %	97.1 % ± 1.8 %	64.4 % ± 5.2 %	81.5 % ± 2.2 %
1-4	81.7 % ± 6.4 %	90.9 % ± 1.8 %	60.3 % ± 3.1 %	84.2 % ± 2.4 %
1-5	83.0 % ± 6.0 %	92.1 % ± 2.3 %	63.7 % ± 6.0 %	79.8 % ± 2.5 %
2-3	83.5 % ± 5.8 %	98.9 % ± 1.1 %	61.0 % ± 4.1 %	91.2 % ± 1.5 %
2-4	80.9 % ± 4.8 %	96.8 % ± 2.3 %	63.4 % ± 4.0 %	91.3 % ± 2.2 %
2-5	81.7 % ± 4.3 %	97.3 % ± 2.2 %	63.1 % ± 4.4 %	87.5 % ± 2.0 %
3-4	91.3 % ± 5.8 %	96.8 % ± 1.5 %	63.2 % ± 5.4 %	70.2 % ± 2.8 %
3-5	90.4 % ± 5.8 %	93.1 % ± 2.4 %	64.4 % ± 4.7 %	67.1 % ± 3.2 %
4-5	91.3 % ± 6.2 %	96.3 % ± 2.5 %	65.9 % ± 5.1 %	67.8 % ± 2.2 %
1-2-3	83.5 % ± 5.8 %	100.0 % ± 0.0 %	64.2 % ± 4.4 %	99.6 % ± 0.4 %
1-2-4	83.5 % ± 5.8 %	100.0 % ± 0.0 %	64.2 % ± 4.4 %	99.6 % ± 0.4 %
1-2-5	83.5 % ± 5.8 %	100.0 % ± 0.0 %	65.3 % ± 4.3 %	99.3 % ± 0.6 %
1-3-4	83.5 % ± 6.1 %	95.5 % ± 1.4 %	60.7 % ± 4.4 %	83.3 % ± 1.9 %
1-3-5	83.0 % ± 5.7 %	97.1 % ± 2.3 %	63.1 % ± 3.7 %	79.4 % ± 1.7 %
1-4-5	83.0 % ± 6.0 %	91.6 % ± 2.3 %	61.0 % ± 3.9 %	82.4 % ± 2.3 %
2-3-4	83.5 % ± 6.1 %	98.4 % ± 1.5 %	62.7 % ± 3.4 %	90.3 % ± 2.0 %
2-3-5	83.5 % ± 5.8 %	98.6 % ± 1.1 %	61.9 % ± 3.7 %	89.0 % ± 1.7 %
2-4-5	80.9 % ± 4.8 %	97.1 % ± 2.1 %	64.6 % ± 3.5 %	89.0 % ± 2.2 %
3-4-5	91.3 % ± 5.8 %	96.8 % ± 1.5 %	63.4 % ± 5.3 %	69.1 % ± 2.7 %
1-2-3-4	83.5 % ± 5.8 %	100.0 % ± 0.0 %	64.1 % ± 4.1 %	99.2 % ± 0.6 %
1-2-3-5	83.5 % ± 5.8 %	100.0 % ± 0.0 %	63.9 % ± 4.0 %	99.2 % ± 0.6 %
1-2-4-5	83.5 % ± 5.8 %	100.0 % ± 0.0 %	64.2 % ± 4.4 %	99.2 % ± 0.6 %
1-3-4-5	83.5 % ± 6.1 %	96.1 % ± 1.8 %	61.7 % ± 4.0 %	82.3 % ± 1.8 %
2-3-4-5	83.0 % ± 6.0 %	98.2 % ± 1.7 %	63.2 % ± 3.2 %	89.4 % ± 2.1 %
1-2-3-4-5	83.5 % ± 5.8 %	100.0 % ± 0.0 %	64.2 % ± 4.2 %	99.2 % ± 0.6 %

Table 13: Kernel Method Results with LibSVM package

	2 inch	2.5 inch	3.5 inch	4.5 inch
Kernel Scale (σ)	Test Set Score and Deviation			
0.2	*	*	30.4 % ± 6.2 %	*
0.25	74.5 %**	58.9 % ± 28.5 %	87.0 % ± 3.6 %	59.3 % ± 9.6 %

*These results are not available due to high computational time.
**This result belongs to first iteration for 2 inch stickout case.

Table 14: Path signature method results obtained with SVM classifier. Landscape number represent which landscapes was used to compute signatures.

Landscape Number	2 inch		2.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	83.0 % ± 2.9 %	84.8 % ± 0.9 %	82.7 % ± 5.3 %	82.9 % ± 2.0 %
2	79.2 % ± 3.1 %	80.2 % ± 1.0 %	84.2 % ± 3.5 %	80.4 % ± 1.4 %
3	78.6 % ± 1.8 %	79.4 % ± 0.7 %	79.1 % ± 2.9 %	80.8 % ± 1.1 %
4	79.3 % ± 2.5 %	79.5 % ± 0.8 %	80.6 % ± 5.6 %	78.6 % ± 2.3 %
5	0.0 % ± 0.0 %	0.0 % ± 0.0 %	80.9 % ± 5.1 %	78.8 % ± 1.2 %
Landscape Number	3.5 inch		4.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	81.2 % ± 6.3 %	82.6 % ± 2.2 %	70.0 % ± 6.4 %	71.4 % ± 2.2 %
2	82.9 % ± 7.2 %	81.8 % ± 2.4 %	64.1 % ± 5.4 %	67.0 % ± 2.4 %
3	81.2 % ± 10.5 %	82.2 % ± 3.8 %	67.7 % ± 6.6 %	65.8 % ± 2.3 %
4	85.9 % ± 4.7 %	80.8 % ± 1.6 %	64.5 % ± 4.2 %	65.4 % ± 1.2 %
5	82.4 % ± 9.1 %	82.0 % ± 3.1 %	64.8 % ± 6.9 %	65.4 % ± 2.1 %