TOPOLOGICAL DATA ANALYSIS AND MACHINE LEARNING FRAMEWORK FOR STUDYING TIME SERIES AND IMAGE DATA

By

Melih Can Yesilli

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Mechanical Engineering – Doctor of Philosophy

2022

ABSTRACT

TOPOLOGICAL DATA ANALYSIS AND MACHINE LEARNING FRAMEWORK FOR STUDYING TIME SERIES AND IMAGE DATA

By

Melih Can Yesilli

The recent advancements in signal acquisition and data mining have revealed the importance of data-driven tools for analyzing signals and images. The availability of large and complex data has also highlighted the need for investigative tools that provide autonomy, noise-robustness, and efficiently utilize data collected from different settings but pertaining to the same phenomenon. State-of-the-art approaches include using tools such as Fourier analysis, wavelets, and Empirical Mode Decomposition for extracting informative features from the data. These features can then be combined with machine learning for clustering, classification, and inference. However, these tools typically require human intervention for feature extraction, and they are sensitive to the input parameters that the user chooses during the laborious but often necessary manual data pre-processing. Therefore, this dissertation was motivated by the need for automatic, adaptive, and noise-robust methods for efficiently leveraging machine learning for studying images as well as time series of dynamical systems. Specifically, this work investigates three application areas: chatter detection in manufacturing processes, image analysis of manufactured surfaces, and tool wear detection during titanium alloys machining. This work's novel investigations are enabled by combining machine learning with methods from Topological Data Analysis (TDA), a relatively recent field of applied topology that encompasses a variety of mature tools for quantifying the shape of data.

First, this study experimentally shows for the first time that persistent homology (or persistence) from TDA can be used for chatter classification with accuracies that rival existing detection methods. Further, the efficient use of chatter data sets from different sources is formulated and studied as a transfer learning problem using experimental turning and

milling vibration signals. Classification results are shown using comparisons between the TDA pipeline developed in this dissertation and prominent methods for chatter detection.

Second, this work describes how to utilize TDA tools for extracting descriptive features from simulated samples generated using different Hurst roughness exponents. The efficiency of the feature extraction is tested by classifying the surfaces according to their roughness level. The resulting accuracies show that TDA can outperform several traditional feature extraction approaches in surface texture analysis. Further, as part of this work, adaptive threshold selection algorithms are developed for Discrete Cosine Transform, and Discrete Wavelet Transform to bypass the need for subjective operator input during surface roughness analysis. Both experimental and synthetic data sets are used to test the effectiveness of these two algorithms. This study also discusses a TDA-based framework that can potentially provide a feasible approach for building an automatic surface finish monitoring system.

Finally, this work shows that persistence can be used for tool condition monitoring during titanium alloys machining. Since, in these processes, the cutting tools typically fracture catastrophically before the gradual tool wear reaches the maximum tool life criteria, the industry uses very conservative criteria for replacing the tools. An extensive experiment is described for relating wear markers in various sensor signals to the tool condition at different stages of the tool life. This work shows how, in this setting, TDA provides significant advantages in terms of robustness to noise and alleviating the need for an expert user to extract the informative features. The obtained TDA-based features are compared to existing state-of-the-art featurization tools using feature-level data fusion. The temporal location of the most representative tool condition features is also studied in the signals by considering a variety of window lengths preceding tool wear milestones.

To my beloved family

ACKNOWLEDGEMENTS

First, I owe my deepest gratitude to my parents, Günay and İsmail, for their endless support throughout my entire life. My brother, Semih, has been one of my best friends. I would not get through this journey without his support and encouragement.

I would like to thank my advisor Dr. Firas Khasawneh for his support and mentorship over the past four years. His guidance turned me into an independent researcher, and I have learned a lot from him over the past four years. I enjoyed our weekly meetings, where we brainstormed about a new problem and lost track of time. I thank my committee members, Dr. Elizabeth Munch, Prof. Brian Feeny, and Dr. Zhaojian Li, for their guidance and support over the last four years. I am also grateful for the patience and feedback of my collaborators Dr. Andreas Otto, Prof. Brain Mann, Dr. Yang Guo, and Prof. Patrick Kwon. I also would like to thank my labmates, Audun Myers and Max Chumley, for their collaboration and contribution.

Finally, this research would not be possible without the financial support of the National Science Foundation (NSF) and Michigan State University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Machine learning has become of the trending areas during the last decades. One area where machine learning has been especially useful is dynamical systems. Data-driven analysis of dynamical systems has grown increasingly popular due to enhancements in measuring devices and data mining. A wide variety of signal processing tools is combined with machine learning to study dynamical systems. However, there is still a need to explore new approaches because the-state-of-the art has some drawbacks specific to applications. This study aims to combine machine learning with Topological Data Analysis (TDA) tools to create new investigative methods to study dynamical systems.

One example of complex dynamical systems is machining processes, including nonlinearities, time delays, and stochastic effects. One of the challenging problems is detecting the occurrence of chatter characterized by a large amplitude of vibrations of the cutting tool. Consequently, identification and mitigation of chatter have become prominent research topics in recent decades. Some of the challenges associated with chatter identification are that it 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 [4, 5, 6, 7, 8]. The two most common methods for analyzing cutting signals are the Wavelet Packet Transform (WPT) and the Empirical Mode Decomposition (EMD). However, these methods have limitations that preclude them from being adopted as general chatter detection tools. To elaborate, this study shows that identifying appropriate feature vectors using these two methods is signal-dependent, and

it requires skilled operators [9]. In addition to the contribution to traditional approaches, Topological Data Analysis (TDA) based approach is developed and used for both synthetic and experimental cutting signals to identify chatter in time series [3, 10]. Another novel approach based on similarities between time series has also developed in this study. This approach combines the Dynamic Time Warping (DTW) algorithm and K-Nearest Neighbor algorithm to diagnose chatter [11].

Another challenging problem in complex dynamical system analysis is data-driven model identification. It provides a useful approach for comparing the performance of a device to the simplified model used in the design phase. One of the modern and popular methods for model identification is Sparse Identification of Nonlinear Dynamics (SINDy). Although this approach has been widely investigated in the literature mainly using numerical models, its applicability and performance with physical systems are still a topic of current research [12]. In Chapter 3, SINDy is extended to identify the mathematical model of a complicated physical experiment of a chaotic pendulum with a varying potential interaction. It is also tested using a simulated model of a nonlinear, simple pendulum. The input to the approach is a time series and estimates of its derivatives. While the standard approach in SINDy is to use the Total Variation Regularization (TVR) for derivative estimates, some caveats for using this route are presented in this study. The performance of TVR is benchmarked against other methods for derivative estimation.

In addition to chatter diagnosis and parameter identification, surface texture analysis is also a challenging problem. Surface roughness determines many important surface properties such as adhesion, friction, wear, as well as both thermal and electrical contact conductance [13, 14]. Currently, the most prevalent approach for describing manufactured surfaces uses statistical point summaries that contain a small fraction of the surface information. These point summary representations are not robust (they are too dependent on the direction of measurement and on noise). They are also not amenable to mathematically linking the surface texture to the physics of the generating process. Therefore, there is a need for

alternative compact descriptions of the often complex and possibly fractal manufactured surfaces [15, 16, 17, 18, 19, 20, 21]. The new objects used for describing these surfaces must be easily visualized, robust to noise, have well-defined metrics, are capable of representing the surface texture at multiple scales, and have the ability to leverage machine learning and other computational and statistical tools. Therefore, Chapter 4 implements tools from TDA in surface texture analysis to address the current limitations in surface texture representation; however, the utility of these tools has never been explored in the context of surface texture characterization and analysis.

The last application area investigated in this study is tool wear analysis. Signal decomposition approaches are also widely adopted for tool wear identification and prediction in the literature. However, these methods have similar problems stated in chatter diagnosis. The final decomposition of the signals requires manual preprocessing and input parameters which are dependent on human error. Therefore, building an automatic and adaptive machine learning framework is the current area of research. In Chapter 5, persistence homology from TDA is utilized to build a parameter-free machine learning framework to classify experimental cutting signals based on the wear amount of corresponding cutting tools. In addition to implementing TDA in tool wear analysis, Chapter 5 modifies existing approaches to reduce the number of parameters required from users.

CHAPTER 2

CHATTER DIAGNOSIS USING MACHINE LEARNING

2.1 Literature Review

Turning, boring, milling, and drilling operations constitute a major part of manufacturing processes. One challenging problem that all these processes have in common is the occurrence of large amplitude, and detrimental oscillations called chatter [22, 23, 24]. Since chatter leads to increased tool wear, poor surface finish, and noise, it is extremely important to anticipate and avoid its occurrence. Alternatively, several chatter mitigation techniques, including increasing stiffness in machine tools and active and passive damping techniques, also exist [25]. Efficient methods for the identification of the stability lobes that separate stable cutting and chattering motion [26, 27] can help keep the machine away from chatter via selecting parameters in the safe area below the stability lobes. However, these models often do not account for the effect of the changing dynamics or for highly complex cutting operations. This led to the emergence of in-situ methods for chatter detection based on instrumenting the cutting center with sensors and analyzing the resulting signals [28, 29, 30, 31, 32, 33].

The majority of available in-process methods for chatter identification rely on extracting certain features from the acoustic, vibration, or force signals and comparing them against some predefined markers of chatter [34, 35, 36, 37, 30, 38, 39, 40, 41, 42, 43, 44]. They can be broadly categorized into two groups as shown in Figure 2.1. The most prevailing methods are Wavelet Packet Transforms (WPT) and Empirical Mode Decomposition (EMD) or the Ensemble Empirical Mode Decomposition (EEMD). Generally, such decomposition-based methods for analyzing the cutting signal follow the same procedure. First, the signal is decomposed into different parts using some transformation. Then, the decomposed portions or packets of the signal, which include the relevant information about machine tool chatter, are selected to reconstruct a new signal. These packets are chosen by applying the Fast

Fourier Transform (FFT) to the different parts or packets and choosing the ones that overlap with the known chatter frequencies of the system. Finally, various time and frequency domain features are computed from these packets. In several papers, these features are ranked and are utilized as the input for the machine learning classifiers. Support Vector Machine (SVM) algorithm is the most common classifier used for chatter classification [32, 6, 45, 5, 8, 46, 47]. Other less common classifiers include quadratic discrimination analysis [4], Hidden Markov Model (HMM) [48], generalized HMM [49], and logistic regression [50] (see Figure 2.1).

Wavelet packet decomposition and wavelet transform are widely adopted in machining state monitoring. Chen and Zheng [5] generated feature matrices for chatter classification using wavelet packets whose frequency bands contain the chatter frequency. Yao et al. [32] used the standard deviation and the energy of the decomposition obtained using the Discrete Wavelet Transform and the WPT for chatter detection from acceleration signals in a boring experiment. The energy of the wavelet packets was also utilized in turning experiments with the comparison of different levels of WPT [51, 49]. Ding et al. [50] used wavelet packet entropy as a feature for early chatter detection. In addition to WPT, EMD and EEMD are also often utilized to featurize cutting signals. Ji et al. [6] proposed EMD to both eliminate noise from milling vibration signals and to extract features from informative Intrinsic Mode Functions (IMF). Chen et al. [45] used top-ranked features extracted from the IMFs obtained from EEMD for machining state detection. Li et al. [52] used the energy spectrum of the IMFs as features for chatter detection. The resulting features are ranked by using Fisher Discriminant Ratio (FDR) [45] and, when the number of features is high, recursive feature elimination (RFE) is used to reduce the number of features [5]. Although EMD/EEMD is typically applied to vibration signals, Liu et al. [8] also used EMD to extract features from the servo motor current time series.

In addition to WPT and EMD-based approaches, there are other methods for feature extraction from metal removal processes. For example, Thaler et al. [4] used Short-Time Fourier Transform to extract the frequency domain features of the feed force, acceleration,

and sound pressure signals in band sawing operation. Moreover, the Q-factor and the power spectrum of the signal were used for chatter classification in milling [46]. Cao et al. [53] applied the Hilbert Huang transform to signals reconstructed using only the informative wavelet packets. Lamraoui et al. applied multi-band resonance filtering and envelope analysis to milling vibration signals [54]. Yesilli and Khasawneh combined Fast Fourier Transform (FFT), Power Spectral Density (PSD), and Auto-correlation Function (ACF) with supervised classification algorithms to detect chatter in turning signals. They used the coordinates of the peaks of FFT, PSD, and ACF plots as features in classification algorithms [55]. Fourier Transform is used in signal-based methods for chatter detection, and Liu et al. [56] combined signal-based and model-based methods to build and hybrid method for chatter detection. Variational Mode Decomposition (VMD) is another method for chatter detection. For example, Liu et al. developed a method to automatically select the VMD parameters and extract the corresponding features using signal energy entropy [57].

Chatter detection strategies based on WPT or EEMD require deciding on which informative parts of the signal to use. However, since searching for the informative parts of the decomposition is a multi-step process, these approaches become impractically laborious. Although the time required to obtain the needed WPT and EEMD decompositions is relatively low, choosing the informative decompositions in WPT and EEMD is often not straightforward. This is because the featurization process involves looking into the Fourier spectra and the energy ratio plots for each signal in order to determine the most informative parts of the decomposition. Consequently, only a few cases are often analyzed, and the chosen packets or decompositions are fixed and used for feature extraction for all the subsequent data sets. For example, in the WPT-based approach, the standard procedure is to pick the packets with the highest energy ratio as the most informative part of the decomposition. However, these packets do not necessarily contain the chatter frequency bands, and thus they may not be the most suitable markers for chatter detection [9].

There are also chatter classification methods that do not rely on signal decomposition.

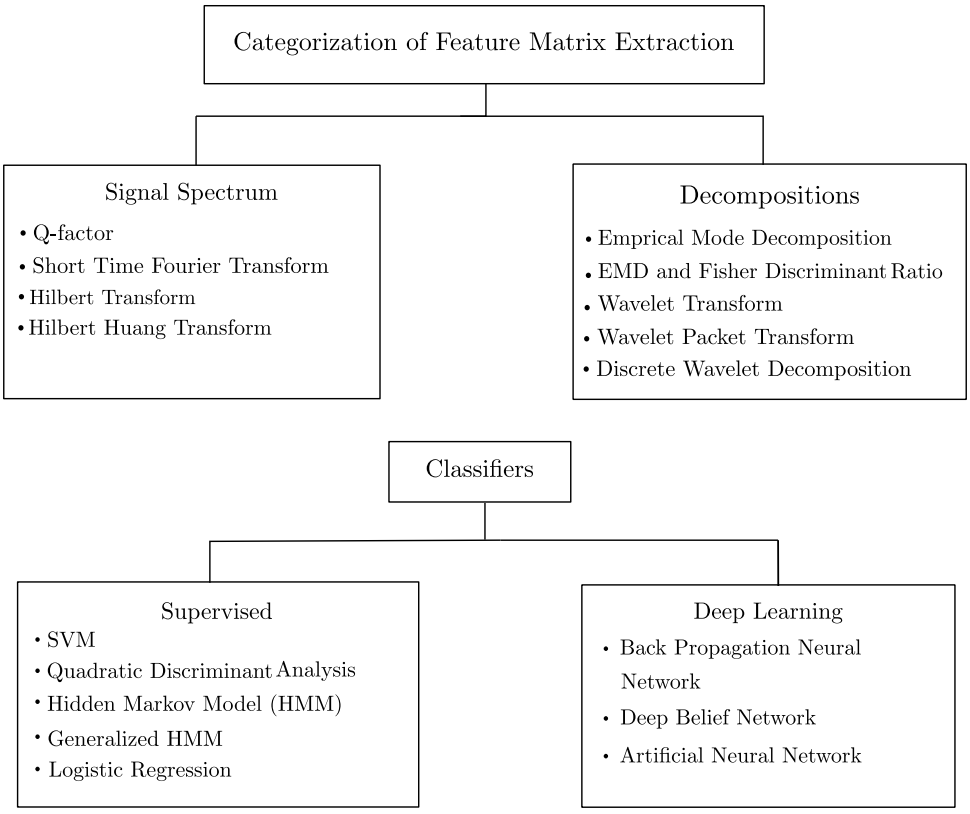


Figure 2.1: Categorization of feature extraction methods and classifiers used for chatter detection.

For example, Tarng et al. utilized unsupervised neural networks with adaptive resonance theory [58]. Tangjitsitcharoen et al. proposed three different parameters which are based on the variance of the cutting force signals to diagnose different cutting states [59]. Fu et al. used a deep belief network with an automatic feature construction model based on unsupervised greedy layer-wise pre-training and supervised fine-tuning to monitor the state of milling processes [60]. Cherukuri et al. used Artificial Neural Network (ANN) on synthetic turning data for chatter classification [61]. However, using ANN (or other black-box machine learning methods) requires large training sets. That amount of data may not always be available, especially in small-batch production processes, which constitute a large portion of discrete manufacturing.

Although prior studies on chatter detection have shown some success, these tools typically share two main limitations: (1) training a classifier requires significant manual pre-processing

of the data, and (2) the trained classifier is sensitive to the differences between the training set and the test set [9]. For example, in the WPT or EEMD method, the signal is decomposed into wavelet packets or IMFs, respectively. The pre-processing requires the selection of the informative wavelet packets or informative IMFs via choosing the packet or IMF that falls within the range of the chatter frequency. These informative packets or informative IMFs are used for extracting frequency and time features, which are often ranked with the Recursive Feature Elimination (RFE) method. Then, an incoming data stream can be classified based on these features and a classification algorithm such as SVM. This means that there are fundamental limitations if the chatter frequencies change significantly, for example, due to changing natural frequencies or changing process parameters. Specifically, Yesilli et al. assessed the transfer learning performance of WPT and EEMD [9]. Further, these methods require a level of skill for feature extraction and classifier training that precludes their wide adoption in chatter detection settings. Therefore, there is a need for an accurate machine learning algorithm for chatter diagnosis that can (1) be easily and automatically applied, and (2) can be computed in a reasonable time. Therefore, this work proposes two novel approaches for chatter detection. These are Topological Data Analysis (TDA) based approach and the approach that utilizes the similarity between time series via Dynamic Time Warping (DTW).

Topological Data Analysis (TDA) [62, 63, 64, 65] is a promising tool for generating feature vectors for chatter detection comes from a new field with many mature computational tools. TDA, and more specifically persistent homology, provides a quantifiable way for describing the topological features in a signal [66]. 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

[67, 68, 69]. Specifically, Reference [68] and [70] 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. Khasawneh et al. incorporated more information from the persistence diagram by extracting 5 features, including Carlsson coordinates ([71]) and the maximum persistence [69], see Section 2.6.4 for more details on featurizing persistence diagrams. In combination with SVM, the resulting feature vector 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. In addition, Yesilli et al. utilized Carlsoon Coordinates and Template Functions [72] to diagnose chatter in milling simulations and show that these two featurization methods are noise-robust [3].

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 are investigated for featurizing the resulting persistence diagrams and classifying chatter time series include persistence landscapes [73], Carlsson coordinates [71], persistence images [74], an example kernel method [75], and path signatures of persistence landscapes [76]. Moreover, the run time for each featurization method is provided, including the runtime for persistence diagram computation, which constitutes most of the total computation time. To reduce the runtime for persistence diagram computation, this study utilizes Bézier curve approximation method [77], greedy permutation [78] and parallel computing.

The second approach proposed in this study is based on combining the K-Nearest Neighbor (KNN) classifier with time series similarity measure: Dynamic Time Warping (DTW)

and Approximate and Eliminate Search Algorithm (AESA). DTW has been used in many application domains including speech recognition ([79, 80, 81, 82, 83]), time series classification ([84, 85, 86]), and signature verification ([87, 88, 89, 90]). In this study, DTW is comp with AESA algorithms to detect chatter in signals obtained from turning experiments.

In machining, natural frequencies of the system shift when cutting configuration parameters such as overhang distance is changed. The chatter frequency, where chatter takes place in the frequency domain, also changes. Since training a classifier on a data set obtained from each new configuration is cumbersome, this study is interested in how a trained classifier on one cutting process can transfer knowledge to the different cutting processes. This general idea is known as transfer learning in the literature. Within the context of machining, it has the potential to provide a methodology for pooling data from different manufacturing settings to more robustly detect chatter. In addition to traditional machine learning, this study also tests the performance of transfer learning in each featurization approach. Classifiers were trained and tested from data gathered from milling and turning experiments. Except for DTW, four different classification algorithms were used for all methods: Support Vector Machine (SVM), Logistic Regression (LR), Random Forest classifier (RF), and Gradient Boosting (GB). K-Nearest Neighbor was used for measuring the performance of the similarity measure technique DTW.

2.1.1 Transfer Learning Approaches for Machining

Several studies focus on chatter detection using deep learning and transfer learning. Cherukuri et al. use synthetic data to train an artificial neural network (ANN) to predict chatter [61]. Postel et al. used a pre-trained Deep Neural Network to predict stability in milling operation [91]. A synthetic data set is used to train the network, and then fine-tuning is performed using the small size of an experimental data set. Unver and Sener used a numerical simulation of milling operation to train AleXNet structure for Convolutional Neural Networks, and they tested the same network on experimental milling data to detect

chatter [92]. In addition to chatter detection, the majority of prior works that apply transfer learning focus on fault detection and tool/machine conditioning instead of chatter detection. Further, these works utilize deep learning algorithms that require a large number of observations [93] and do not provide insight into the signals' most informative features for chatter detection. For instance, Wu et al. used 1D Convolutional Neural Networks (CNN) for fault detection in bearings and gears [94]. They applied two different transfer learning approaches: 1) training and testing a classifier on samples from different working conditions and 2) training on simulation data and testing on experimental data. Li and Liang developed a CNN-based approach to diagnosing severe tool wear, tool breakage, and spindle failure during machining processes [93]. They used two different CNC machines to train and test a classifier in an experiment that took six months to collect the data needed to train the CNN. Kim et al. used Support Vector Regressor to predict the machining power, and they transferred knowledge from machining power models of steel and aluminum to predict the power model of titanium [95]. Mamledesai et al. utilized CNN and transfer learning to monitor tool conditions to help the machinist decide whether to keep using the same tool or replace it [96]. Marei et al. used Convolution Neural Network-based transfer learning to predict tool wear of the carbide cutting tool flank [97]. Another study that includes transfer learning and deep learning is focused on the estimation of force in the milling process using simulation data and experimental data as a source and target domain, respectively [98]. Wang et al. use the pre-trained network VGG19 to identify machining fault types in rolling bearings. They modified the final fully connected layer to reduce the number of network parameters and implement the transfer learning between non-manufacturing data and manufacturing data [99]. Kim et al. proposed another approach that converts cutting force signals into images using a multi-layer recurrence plot (MRP) to estimate the machining quality in laser-assisted micro-milling operation [100]. They used a pre-trained ResNet-18 CNN structure and tested it on the images generated from cutting signals.

Traditional machine learning approaches are also adopted in transfer learning approaches

for machining applications. For instance, Gao et al. implemented extreme vector machines and transfer learning to build a prediction model for remaining tool life [101]. Yesilli et al. combined traditional signal decomposition tools and machine learning algorithms such as support vector machines, random forest classifier, and gradient boosting to detect chatter in experimental turning signals [9]. Fast Fourier Transform, Auto-correlation Function, and Power Spectral Density are also combined with similar machine learning algorithms to identify unstable time series obtained from turning experiments [55]. Shen et al. combined the TrAdaBoost transfer learning algorithm [102] and singular value decomposition-based feature extraction to identify different fault types in a bearing data set [103]. The TrAdaBoost algorithm is also used in tool tip dynamics prediction [104].

2.1.2 Main Contribution

For WPT and EEMD approaches, the resulting informative packets or decompositions may not contain chatter information, especially if the system parameters shift during operation, e.g., due to the movement of the machining center, which may involve changing the overhang distance of the tool and thus the flexibility of the cutting tool. Therefore, in these situations, the classifier is required to categorize signals that may carry different characteristics and chatter features than the ones it was trained on. In other words, the ability of the classifier to achieve *transfer learning* is tested in these situations. However, there have not been any studies on the transfer learning capabilities of WPT and EEMD. Therefore, this study investigates the transfer learning performance of these two approaches to the novel approaches proposed in this study. Further, the common approach for picking the informative packets in WPT is to choose the packets with the highest energy. However, these packets do not necessarily contain the chatter frequency bands, and thus they may not be the most suitable markers for chatter detection [9].

State-of-the-art methods for chatter detection, WPT, and EEMD require intense manual preprocessing and thus have low automation potential ([9, 55]). For example, the frequency

spectrum of the signals obtained from wavelet packets and IMFs must be checked to choose the informative wavelet packets or IMFs. In contrast, the proposed DTW approach eliminates the feature extraction step since it does not depend on signal decomposition but rather relies on computing pairwise distances between time series. All the steps in the DTW approach can be performed automatically. It only needs two input parameters: the number of neighbors required for the KNN classifier and the looseness constant (H) for the AESA algorithm.

Although this study focuses on turning as a use case, the DTW approach is applicable to other machining processes where the data stream is in the form of time series. A comparison of the resulting chatter classification success rates between the DTW approach and other widely used methods in the literature shows that the DTW approach has the highest average accuracy or is within the error band of the highest accuracy in two out of the four cutting configurations for turning experiments.

This study also shows how to drastically reduce the computation time by combining the DTW approach with the Approximate and Eliminate Search Algorithm (AESA) or parallel computing. Although AESA has been widely used in word recognition, pattern recognition, and handwritten character recognition ([105, 106, 107, 108, 109]), it is believed that this work is the first to combine AESA and DTW for analyzing engineering systems. In addition, the results obtained with AESA and parallel computing show that after training a classifier offline, an incoming time series can be labeled in less than two seconds. Therefore, the DTW approach is very conducive to online chatter detection applications.

Previous studies on chatter detection with a TDA-based approach either utilize a small subset of the available featurization methods of persistence diagrams, or they only consider simulated data. Specifically, Reference [68] used maximum persistence lifetime on a simulated stochastic turning model to visually show that persistence diagrams carry chatter information. In Reference [69], the authors used Carlsson Coordinates (in addition to maximum lifetime) as features and logistic regression classifier to distinguish chatter versus chatter-free signals obtained from stochastic and deterministic turning simulations. Reference [70]

used Persistent Homology to visually detect the changes in the behavior of a linearized turning model using a heat map of maximum persistence plotted on top of the spindle speed and the depth of cut space. Reference [3] only used Carlsson Coordinates and Template Functions, where the latter was introduced in [72], for chatter detection in simulated milling signals. In contrast to previous studies on TDA, machine learning, and machining dynamics, this study is the first to consider experimental data. Further, in contrast to the previous studies that used one or two TDA featurization methods, this study compares for the first time the most common featurization methods from TDA. Further, this study also focuses on the classification results using four common classifiers: Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), and Gradient Boosting (GB). It is believed that this study is the first to apply a variety of persistence diagram featurization techniques to experimental machining data sets. Another distinguishing feature of this study is a focus on speeding up persistence computations by leveraging several computational tools such as greedy permutation, Bézier curve approximation, and parallel computing. The described speedups significantly reduce the computational time, thus enabling utilizing the approach described in this work for effective chatter detection.

Another main contribution of this work is to present the first study on using state-of-the-art feature extraction tools to transfer chatter knowledge across turning and milling operations using experimental data. The main goal is to automate chatter detection for different cutting conditions and operations and to reduce the amount of data and time needed to train a classifier [110]. Once a classifier is trained using a given data set, the gained information can be utilized for different operations without needing large and completely new training data sets from the target process.

In contrast to prior works on transfer learning for chatter detection, this work focuses on a large number of feature extraction methods, including WPT, EEMD, FFT, ACF, PSD, as well as two other novel methods that have been proposed in this study in the context of machining: Topological Data Analysis (TDA) methods and similarity-based methods using

Dynamic Time Warping (DTW).

This chapter is organized as follows. Section 2.2 gives the background information of four different classification algorithms that have been widely adopted in this chapter. Section 2.3 explains the feature extraction from WPT and EEMD and presents the results obtained from these approaches. Section 2.4 describes the traditional feature extraction approach using FFT/PSD/ACF and provides the classification results. The first novel approach developed in this study is explained in Section 2.5 and the resulting classification accuracies can be found in the same section as well. The second novel approach developed by this study and the results obtained using this approach are provided in Section 2.6. The third main contribution of this study for chatter diagnosis is the application of transfer learning between different machining operations using various featurization techniques. Section 2.7 provides background information for transfer learning and the results obtained using experimental turning and milling data sets. This chapter uses experimental data set obtained from turning and milling experiments. One can refer to Section A.1 and A.2 for more details about data collection and processing.

2.2 Supervised Classification Algorithms

This section gives background information on the different classifiers used to test the performance of considered feature extraction methods, namely SVM, logistic regression, random forest classification, and gradient boosting.

2.2.1 Support Vector Machine

A Support Vector Machine (SVM) is used to classify the time series by using feature vectors. The Support Vector Machine algorithm is a supervised machine learning technique for finding the optimal hyperplane separating two training data set classes. This hyperplane can then be used to classify the test data. The two dimensional case of a linear SVM is illustrated in Figure 2.2. The feature vectors corresponding to two different classes, e.g.,

chatter (crosses) and no-chatter (circles), form two linearly separable data sets. The optimal hyperplane is selected such that the perpendicular distances from the feature vectors, which are closest to the hyperplane and also called support vectors, are equal. This means that the optimal hyperplane has the largest margin [111]. In general, it can be described by the set of points $\mathbf{x}$ satisfying

$$\mathbf{w} \cdot \mathbf{x} + c = 0, \tag{2.1}$$

and the dashed lines where the support vectors lie on are defined according to

$$f_{\pm 1}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + c = \pm 1. \tag{2.2}$$

Then, the margin of the optimal hyperplane can be denoted as $2/\|\mathbf{w}\|$. The two hyperplanes with Equation 2.2, and therefore the optimal hyperplane from Equation (2.1), can be found by maximizing the distance $2/\|\mathbf{w}\|$ or by minimizing $\|\mathbf{w}\|^2$ with the constraints

$$\begin{aligned} &\mathbf{w} \cdot \mathbf{x} + c \geq +1, \quad \text{and} \\ &\mathbf{w} \cdot \mathbf{x} + c \leq -1. \end{aligned} \tag{2.3}$$

The classification for a feature vector $\mathbf{x}_{\text{test}}$ of the test set can be made by checking the sign of the expression $\mathbf{w} \cdot \mathbf{x}_{\text{test}} + c$, which defines the label for the two classes. For the theory behind multi-class classification with SVM, one can refer to [112]. In some cases, the training data are not separable by a linear hyperplane. In this case, the SVM can be extended to nonlinear classification with the help of kernel functions [113].

2.2.2 Logistic Regression

Logistic regression is a supervised learning classification algorithm that computes the probability of two class labels for a given dependent variables [114]. It is quite similar to linear regression, but its output is divided into two categories [115]. Figure 2.3 illustrates linear and logistic regression on a binary data set. In this figure, $X = \{x_1, x_2, \ldots, x_n\}$ is the set of elements in the feature vector while $Y \in \{0, 1\}$ is the dichotomous outcome variable.

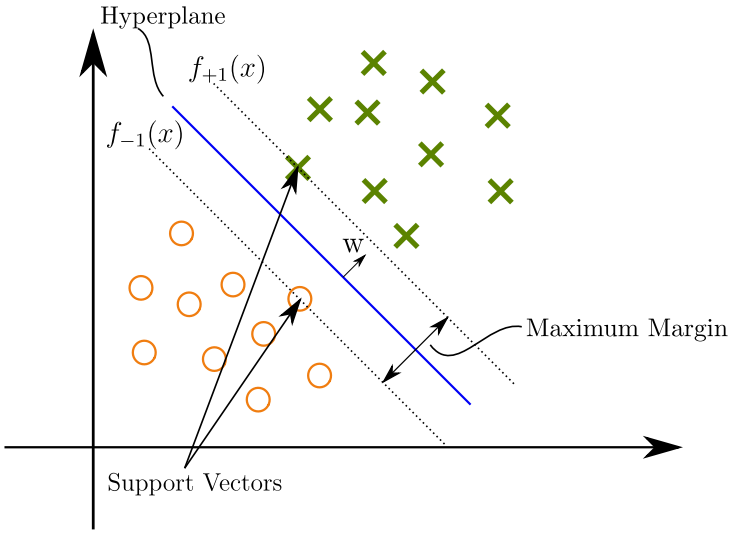


Figure 2.2: Selected optimal hyperplane for linearly separable two class data set.

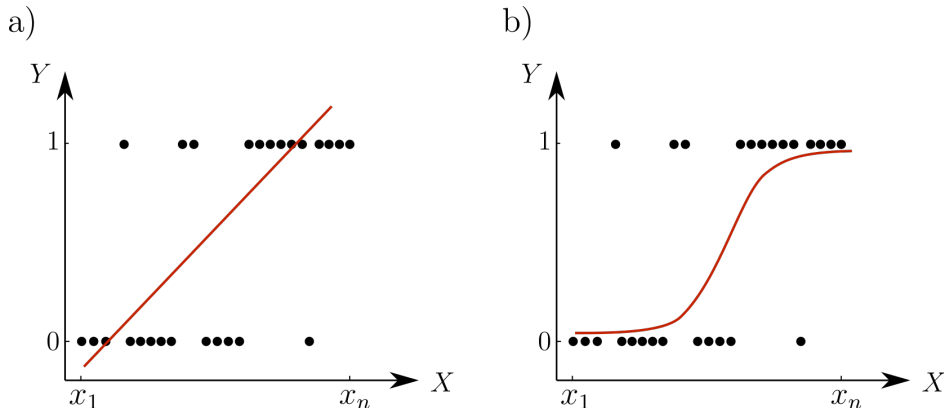


Figure 2.3: Linear (a) and logistic (b) regression onto data set whose output is binary.

For dichotomous output, linear regression can be applied, but the model will not fit well, as shown in Figure 2.3a. There are two main reasons why the linear equation does not explain the relationship between the variables X and Y [116]: (1) the relationship between the variables does not have a linear trend, and (2) the errors are not constant, or they are not normally distributed. However, this problem can be solved by introducing the logit transformation.

Let $\pi(x) = E(Y|x)$ be the expected value of Y given the value of x. The regression model $g(x)$ and the logit transformation $\pi(x)$, respectively, are defined according to [115] in Equation 2.4, where $\pi(x)$ is defined as the sigmoid function for logistic regression in

Equation 2.5. The regression model is expressed as a linear function. However, it is converted into a nonlinear probability function with logit transformation.

$$g(x) = \ln\left(\frac{\pi(x)}{1-\pi(x)}\right) = \beta_0 + \beta_1 x, \tag{2.4}$$

$$\pi(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}, \tag{2.5}$$

Although Equation 2.4 is defined for only one independent input variable x, the model can be further extended to a multivariate version. To assign labels for a given input x, the decision boundary must first be formed. In Figure 2.3b this boundary is the sigmoid function that splits tags 0 and 1. The x values which satisfy $\beta_0 + \beta_1 x = 0$ form the decision boundary [114], and the probability at the boundary, per Equation (2.5), is 0.5. The parameters β_0 and β_1 in the regression model can be identified using maximum likelihood estimators [114].

2.2.3 Random Forest Classification

Ensemble learning uses multiple methods to get higher prediction rates for a given problem. For example, random forest is an ensemble learning method composed of decision trees where the number of these trees is part of the user input to the algorithm [117]. Each decision tree is composed of branch nodes with two branches emanating from each root node; hence, they are called binary trees. The nodes that have no descendants are termed leaf nodes or leafs. Assuming numeric inputs, each branch node corresponds to one variable and its split point, while the leaf nodes correspond to output variables. Figure 2.4 illustrates decision tree classification using two classes (0 and 1) and two input variables (x and y). The first step is to partition the input space of the training set into rectangles (or hyper-rectangles in higher dimensions), in this case, L_1 through L_5. Selecting the partitions is based on making each subset of the training set purer, i.e., with fewer mixed labels, than the training set itself [118]. An impurity function defines the goodness of each partition, see [118] for a discussion on optimum splits. After defining the partitions for the training data set (left graph in Figure 2.4), a tree is formed (right graph in Figure 2.4).

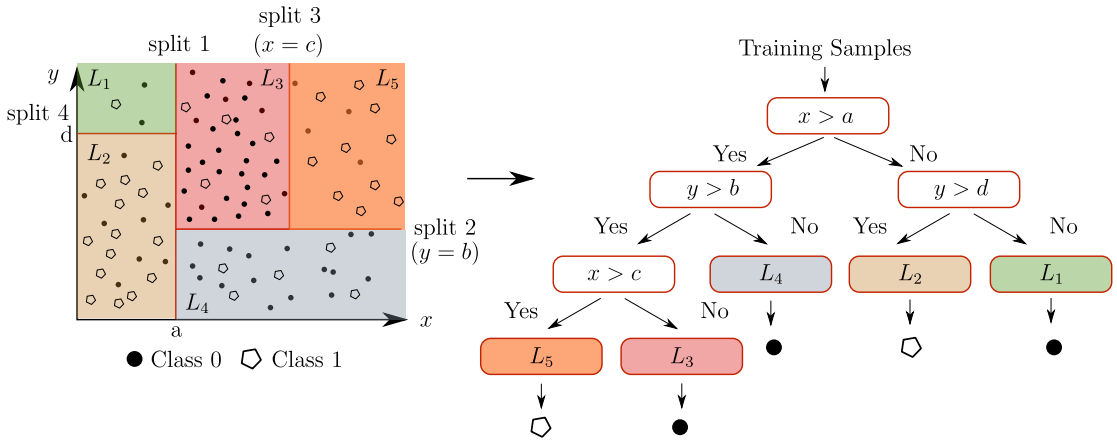


Figure 2.4: Generation of decision tree.

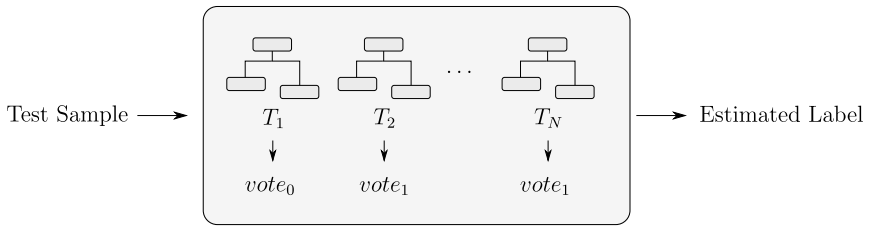


Figure 2.5: Random Forest Classification ($vote_0$:Final decision of the corresponding tree is class 0. $vote_1$:Final decision of the corresponding tree is class 1.).

The branch nodes of the tree correspond to conditions, either on x or on y, such that the samples L_1 through L_5 can be placed in one of the leaf nodes. Each leaf node is then labeled by following the plurality rule [118]: the most frequent labels in any node are assigned as the label for that node. For example in Figure 2.5, leaf nodes L_1, L_3, and L_4 are labeled as class 0, while leafs L_2 and L_5 are labeled as class 1. Given a new input, a tag is generated by traversing the tree starting at the tree's root node. The new input's label is then matched to the leaf node it ends up in within the tree.

In Random forest classification, there are N decisions trees, and each tree votes for a label for a given test sample. The algorithm chooses a number of samples to generate the decision trees, and this is iterated until the desired number of trees is obtained. The estimation for the label of the sample is made with respect to the most frequent votes [119] as shown in Figure 2.5.

2.2.4 Gradient Boosting

Gradient boosting algorithm was introduced by Schapire to answer the question of whether the performance of a single strong learner is equal to the set of weak learner performance [120]. Gradient boosting was proposed as an algorithm that provides more accurate predictions for regression and classification problems by generating new base models, which can be linear models, smooth models, and decision trees [121]. Gradient Boosting aims to correct the previous models by adding new base models to minimize the loss function. When the decision trees are used as new base models, a new decision tree is added after computing the loss function. The new decision tree is generated by parametrizing it so that it can decrease the loss of the existing model. Specifically, the gradient descent is used to minimize the loss function value, and it is applied in functional space since each tree (base learner) can be represented as a function. Gradient boosting algorithm fits the new base models to the negative gradient of the loss function, where the choice of the loss function is user-dependent, to increase the accuracy of the overall model [122].

2.3 Signal Decomposition Based Approach

This section describes two signal decomposition approaches, Wavelet Packet Transform (WPT) and Ensemble Empirical Mode Decomposition (EEMD). These approaches are widely adopted in literature for chatter detection. Recursive Feature Elimination (RFE) is also explained and used to rank features. The approach which utilizes WPT can be divided into four steps, which are summarized in Figure 2.6. The first step is the decomposition of the time series into wavelet packets. This technique from signal processing is especially useful for a high-resolution time-frequency analysis. The motivation for an additional decomposition of the signal is the increase of the signal-to-noise ratio and increasing sensitivity for chatter features [5]. The output of the WPT is wavelet packets. The second step is the selection of the informative packets based on the properties of the wavelet packets and the characteristics of chatter in the considered process. The third step is the feature extraction and its automatic

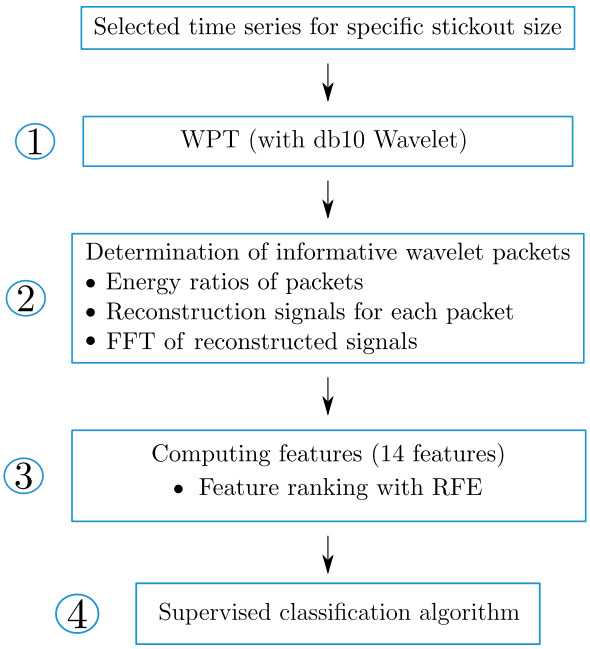


Figure 2.6: Overview of the Wavelet Packet Transform (WPT) method with Recursive Feature Elimination (WPT).

ranking with the RFE method, which is used to distinguish between chatter and chatter-free motion. On the basis of the extracted features, the fourth step is the classification into chatter/chatter-free cases via a Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF) classification, and Gradient Boosting (GB).

The structure of the method that employs EEMD is similar to the WPT-based approach. However, in contrast to the WPT method, the EEMD is used for the decomposition of the original time series, and the output of the EEMD is intrinsic mode functions (IMF) instead of wavelet packets. After the decomposition, the informative IMF is selected, and various features for chatter detection are extracted. The features are automatically ranked via the RFE method, and supervised machine learning algorithms are used to classify them into chatter/chatter-free cases.

2.3.1 Wavelet Packet Transform

The methodology in Reference [5] was followed in this section, and WPT is applied to the time series before feature extraction and classification. The WPT is an extension of the discrete wavelet transform. One level of the discrete wavelet transform decomposes the signal into a low and a high-frequency component by passing it simultaneously through a low and a high pass filter. The properties of the two filters are related to each other and are determined by the chosen wavelet basis. According to [5], the Daubechies orthogonal wavelet db10 is used as the wavelet basis function. The outputs of the low and the high pass filter give the approximation coefficients and detailed coefficients denoted by A_i and D_i, respectively, where the subscript i specifies the level of the decomposition. The resulting signal after the decomposition is called a wavelet packet and can be reconstructed from the approximation or detailed coefficients by using the filter properties [5]. In the discrete wavelet transform, only the output A_i is passed again through both filters to generate two additional outputs AA_{i+1} and AD_{i+1} in the next level.

In contrast, in the WPT approach the output A_i of the low pass filter as well as as the output D_i of the high pass filter are both again low- and high-pass filtered to generate the wavelet packets AA_{i+1}, AD_{i+1}, DA_{i+1} and DD_{i+1} in the next level. This means that the WPT generates 2^k wavelet packets at the kth level, see Figure 2.7 for a schematic of level 3 WPT. In Figure 2.7, for example, DAA_3 denotes the packet in the third level, where in the first and the second level, the low pass filter and in the third level, the high pass filter have been applied. Before passing through the filters in the next level, the signal is downsampled by a factor of two, which increases the frequency resolution. Moreover, since the two resulting wavelet packets contain only one-half of the frequencies of the input data after each decomposition, this downsampling is possible without losing information. As a consequence, the resulting wavelet packets in one level contain only a frequency band, which is mainly distinct from the bands of the other packets. Even if the frequency bands become narrower at each level, the packets contain rich information about the original signal due to

the increase in the frequency resolution. The location of the frequency band is determined by the chronological order of the applied filters, which are used to generate the wavelet packet (cf. Figure 2.7). In the following, the wavelet packets are labeled according to the order of their frequency band, beginning with 1 for the packet with the lowest frequencies ($A \dots A_k$) resulting only from low pass filtering to 2^k for the packet containing the highest frequencies ($D \dots D_k$) resulting from a successive application of the high pass filter.

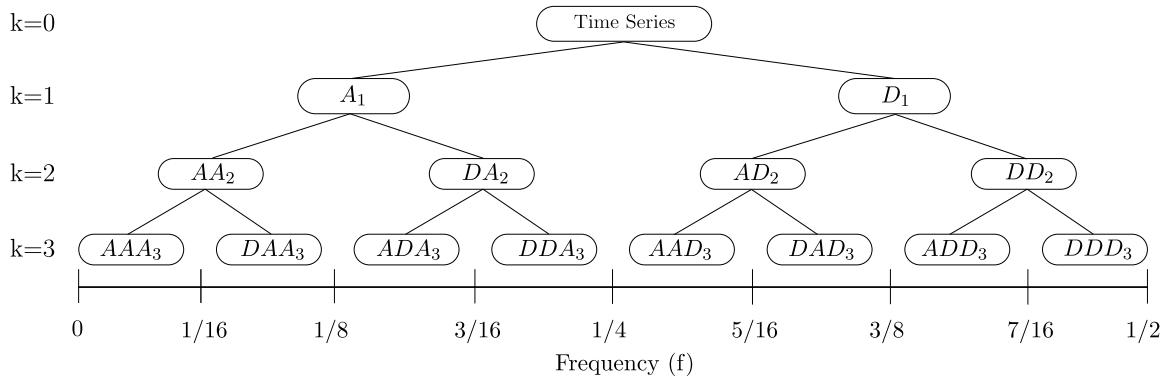


Figure 2.7: 3 Level Wavelet Packet Transform.

2.3.1.1 Selection of Informative Wavelet Packets

The next step is the selection of the informative wavelet packets, which are best suited to distinguish between stable cutting and chattering motion. The criteria for selecting the informative wavelet packets are high signal energy compared to other packets for a good signal-to-noise ratio and significant overlap of the frequency band of the packet with possible chatter frequencies. In this section, the selection of the informative wavelet packets is described for the turning experiment explained in Section A.1.

The identification of the band of chatter frequencies is made by examining the FFT of the signals tagged as stable, intermediate chatter, and chatter (see Section A.1 and A.1.1 for the description of the experimental setup and the labeling of turning experiments). Figure 2.8 shows example time series and the corresponding Fourier spectra for three tagged signals for

the case whose stickout length, rotational rpm, and depth of cut are 5.08 cm (2 inch), 320 rpm, and 0.127 mm (0.005 inch), respectively. The dominant frequencies are low for stable cutting and correspond to the spindle rotation frequency. In addition, there is a significant peak at 120 Hz, which can be found in all measurements and probably comes from an external source. For intermediate chatter and chatter, a significant part of the energy in the signal is contained at high frequencies near 1000 Hz, which is close to the eigenfrequency of the lateral tool vibration. As a consequence, these chatter frequencies become larger for increasing stickout length, and for each of the four different stickout lengths, a different range of chatter frequencies has been identified.

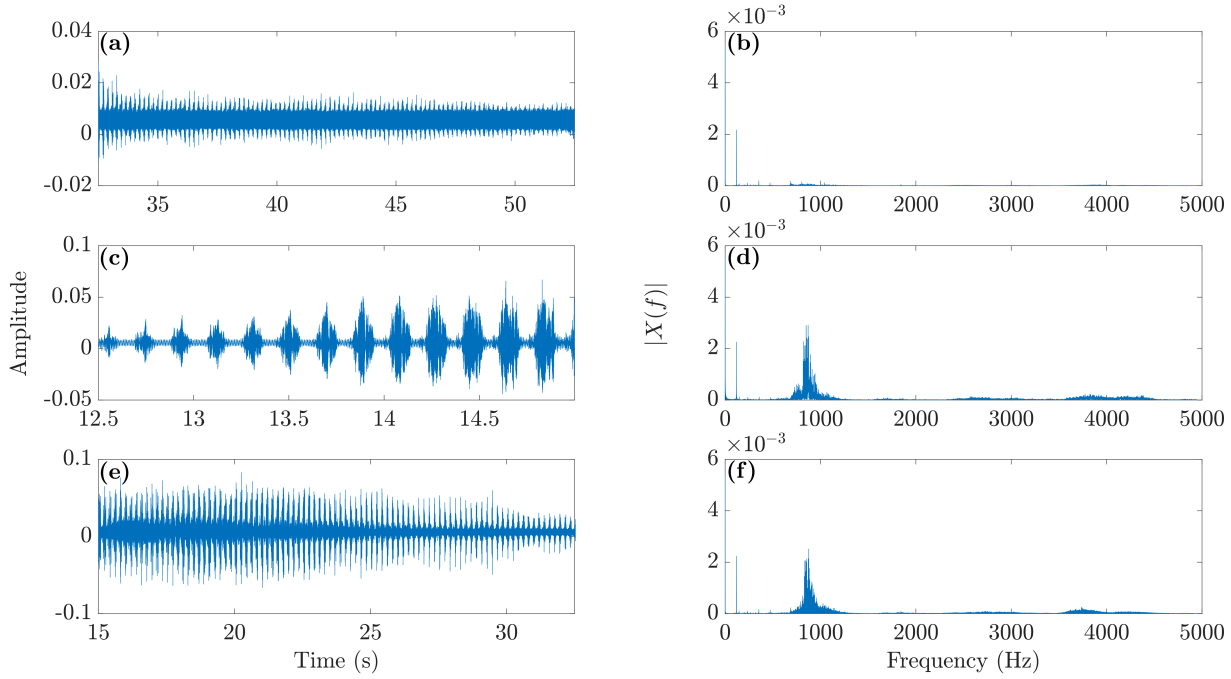


Figure 2.8: Time domain and frequency domain of stable (a,b), intermediate (c,d), and chatter (e,f) regions for overhang distance of 5.08 cm (2 inch), 320 rpm, 0.002 inch depth of cut case of turning experiments.

In order to analyze the properties of the wavelet packets, levels 1, 2, 3, and 4 WPT are obtained from the experimental data. Figure 2.9 shows the resulting level 4 energy ratios of the wavelet packets for two example cases. The energy ratios represent the fraction of energy in each packet relative to the total energy in all the packets. It is obvious from the

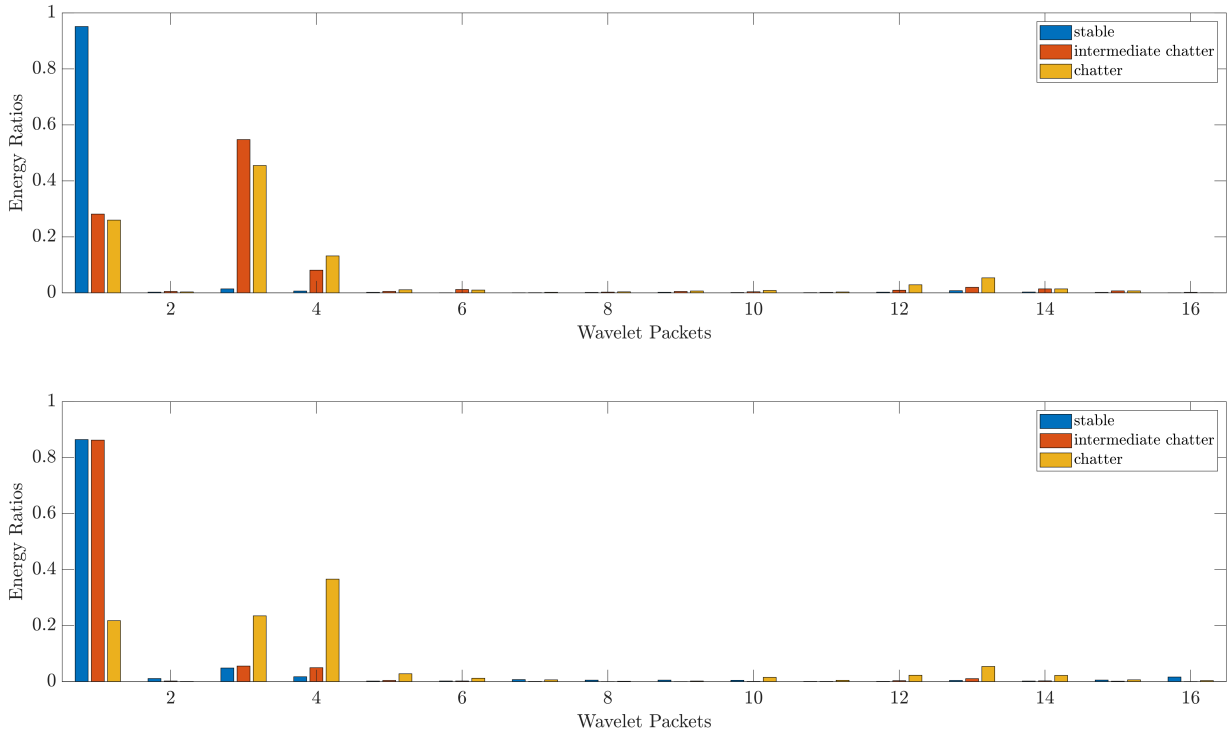


Figure 2.9: Energy ratios of wavelet packets for two cases of turning experiment: (top) 5.08 cm (2 inch) stickout, 320 rpm and 0.0127 cm (0.005 inch) DOC, and (bottom) 5.08 cm (2 inch) stickout, 570 rpm and 0.00508 cm (0.002 inch) DOC. Note the differences in the scale of the vertical axis.

figure that most of the energy is concentrated in the first wavelet for stable cutting. In contrast, the energy is concentrated mainly in the first, third, and fourth wavelet packets for the intermediate chatter and the chatter regions. This is consistent with the behavior of the frequency spectrum of the original data in Figure 2.8 since increasing the number of the wavelet packets corresponds to a higher frequency band.

Upon identifying the wavelet packets whose energy ratios are relatively high with respect to the other packets, the third step is to identify the packets whose spectrum has significant peaks that overlap with the chatter frequencies given in Table 2.1 [5]. Specifically, a time domain signal for each wavelet packet is reconstructed, and the corresponding FFT is obtained for each of the reconstructed signals. For the two examples with stickout length 5.08 cm (2 inch), the frequency spectrum of the reconstructed signals obtained from the first four wavelet packets for the intermediate chatter and chatter regions are provided in Figure 2.10.

Table 2.1: The chatter frequency ranges, the informative wavelet packets, and the informative IMFs corresponding to each overhang distance of the cutting tool for the turning experiments.

Stickout length (cm (inch))	Chatter frequency range (Hz)	Informative wavelet packets	Informative IMF
5.08 (2)	900–1000	Level 1 :1, Level 2: 1, Level 3: 2, Level 4: 3	2
6.35 (2.5)	1200–1300	Level 1 :1, Level 2: 1, Level 3: 3, Level 4: 4	2
8.89 (3.5)	1600–1700	Level 1 :1, Level 2: 2, Level 3: 3, Level 4: 6	1
11.43 (4.5)	2900–3000	Level 1 :2, Level 2: 3, Level 3: 5, Level 4: 10	1

It can be seen that the peaks in the spectrum of the 3rd and 4th wavelet packet overlap with the band of the previously identified chatter frequency (900–1000 Hz, see Table 2.1 and Figure 2.8). Since, for the stickout length 5.08 cm (2 inch), the energy ratios and the amplitudes in the corresponding FFT (see Figure 2.10) are slightly higher in the 3rd wavelet packet than in the 4th wavelet packet, the 3rd packet was chosen as the informative wavelet for chatter detection at level 4 WPT. An overview of the selected informative wavelet packet for each level of the WPT can be found in Table 2.1. For higher stickout length, the dominant chatter frequencies increase, and therefore, in general, a wavelet packet with a higher frequency band is selected as the informative wavelet packet.

Current literature attempted to automate the selection of the informative packets by selecting the packets with the highest energy. However, this study showed that the informative wavelet packet is not necessarily the one with the highest energy because it is important that the range of possible chatter frequencies are in the frequency band of the informative wavelet packet [9]. In fact, often, the first packet has the highest energy ratio. However, its frequency band does not overlap with the chatter frequencies, which are mainly contained in packets with a higher index (cf. Table 2.1).

Since the frequency band of the wavelet packets can be predicted from the WPT tree in Figure 2.7, it is also possible to predict the informative wavelet packet that contains information about chatter frequencies. For example, from the sampling rate of 10 kHz, it follows that the first wavelet packet in level 3 corresponds to the frequency band 0–625 Hz. The upper frequency limits for other packets in level 3 are equal to the corresponding wavelet

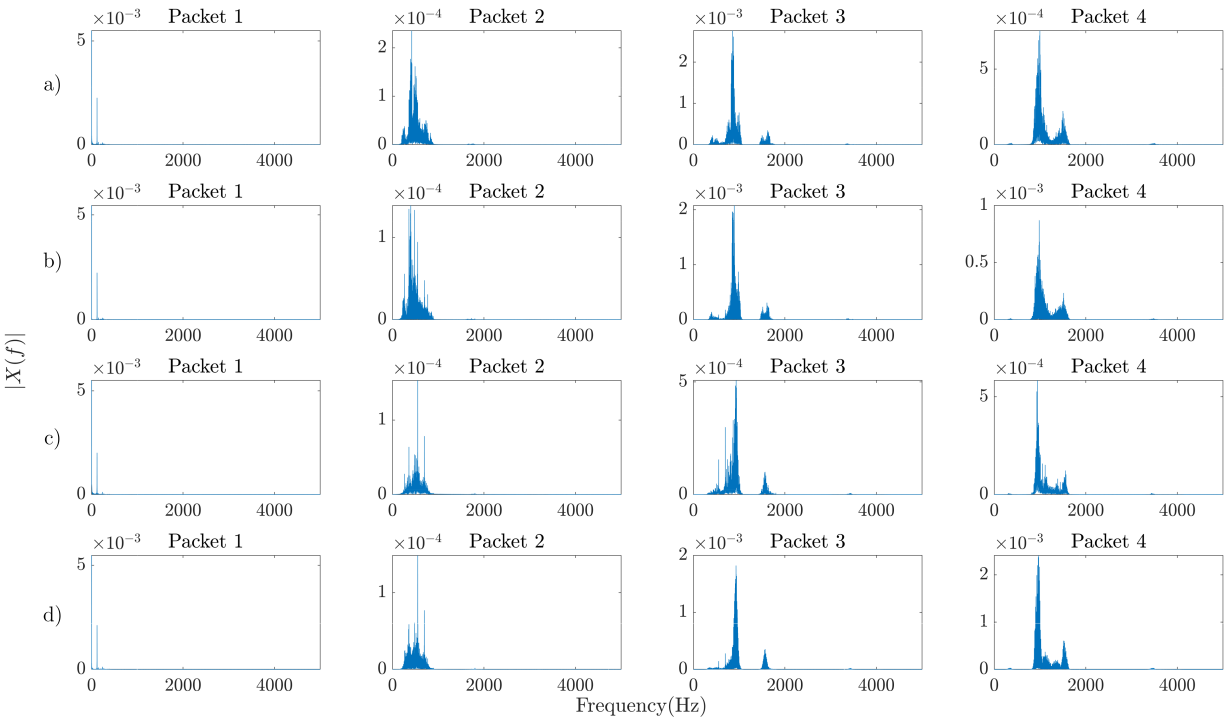


Figure 2.10: The spectrum of the first four wavelet packets of level 4 WPT for intermediate chatter (a),(c) and chatter (b),(d) in the case with 5.08 cm (2 inch) overhang distance. The spindle speed and depth of cut is 320 rpm and 0.0127 cm (0.005 inch) in (a), (b) and 570 rpm and 0.00508 cm (0.002 inch) in (c), (d), respectively.

Table 2.2: Comparison between predicted and selected informative wavelet packet number for all overhang distances cases of the turning experiment. Predicted wavelet packets are decided with by overlapping the chatter frequency with the wavelet packet frequency range obtained from the WPT tree (Figure 2.7) for level 4.

Overhang (Stickout) Distance (cm (inch))	Chatter frequency range (Hz)	Informative wavelet packets (Predicted)	Informative wavelet packets (Selected)
5.08 (2)	900–1000	Level 4: 3-4	Level 4: 3
6.35 (2.5)	1200–1300	Level 4: 4-5	Level 4: 4
8.89 (3.5)	1600–1700	Level 4: 6	Level 4: 6
11.43 (4.5)	2900–3000	Level 4: 10	Level 4: 10

packet number times the upper frequency level of the first wavelet packet (cf. Figure 2.7). Table 2.2 provides the predicted and the selected informative wavelet packets for the level 4 WPT. The selected informative wavelet packets are consistent with the predicted ones for all cases.

2.3.1.2 Recursive Feature Elimination (RFE)

The reconstructed signal from the informative wavelet packet allows the extraction of both frequency domain as well as time domain features for chatter identification. A collection of frequency domain and time domain features, which are taken from Reference [5], are provided in Table 2.3.

Python is used to train a supervised classification algorithm combined with Recursive Feature Elimination (RFE), where in this case maximum of 14 features are available at the level 4 WPT. Recursive feature elimination is an iterative process that eliminates one of the features in each iteration until all the features are removed for classification [5], which means that the number of iterations for RFE equals the number of the considered features. Elimination of features is based on their influence on the classification: the feature with the smallest effect is eliminated in each iteration [123]. In the end, RFE returns a feature ranking list corresponding to one specific training set.

The ranked features are used to generate feature vectors where the first vector contains only the first ranked feature, while each consecutive feature vector adds the subsequent feature in the ranking until all the features are included in the 14th vector at the fourth level of WPT. The classification accuracy is calculated for all 14 feature vectors. In other words, in the first step, only the top-ranked feature is used, and in each further step, the next highest-ranked feature is added to the feature matrix, and the classification accuracy is computed again.

2.3.2 Ensemble Empirical Mode Decomposition

EEMD is based on the Empirical Mode Decomposition (EMD), which is an elementary step in the Hilbert-Huang transform [124]. Similar to WPT, EMD is useful for non-stationary signals since the resulting IMFs contain the time and frequency information of the signal. The main difference in contrast to WPT and other linear decomposition methods is that the expansion bases of EMD are not fixed but are rather adaptive, and they are determined by

Table 2.3: Time domain features ($a_1, \ldots, a_{10}$) and frequency domain features ($a_{11,\ldots,14}$).

Features	
$a_1 = \frac{1}{N}\sum_{m=1}^{N} x_m$ (Mean)	$a_8 = \frac{a_4}{(\frac{1}{N}\sum_{m=1}^{N}\sqrt{\lvert x_m\rvert})^2}$ (Clearance Factor)
$a_2 = \sigma(x_m)$ (Standard Deviation)	$a_9 = \frac{a_3}{\frac{1}{N}\sum_{m=1}^{N}\lvert x_m\rvert}$ (Shape Factor)
$a_3 = \sqrt{\frac{1}{N}\sum_{m=1}^{N} x_m^2}$ (RMS)	$a_{10} = \frac{a_4}{\frac{1}{N}\sum_{m=1}^{N}\lvert x_m\rvert}$ (Impulse Factor)
$a_4 = max(\lvert x_m\rvert)$ (Peak)	$a_{11} = \frac{\sum_{k=1}^{M} f_k^2\lvert X(f_i)\rvert}{\sum_{k=1}^{M}\lvert X(f_k)\rvert}$ (Mean Square Frequency)
$a_5 = \frac{\sum_{m=1}^{N}(x_m-a_1)^3}{(N-1)a_3^3}$ (Skewness)	$a_{12} = \frac{\sum_{k=1}^{M} cos(2\pi f_k\Delta t)\lvert X(f_k)\rvert}{\sum_{k=1}^{M}\lvert X(f_k)\rvert}$ (One Step Auto Correlation Function)
$a_6 = \frac{\sum_{m=1}^{N}(x_m-a_1)^4}{(N-1)a_3^4}$ (Kurtosis)	$a_{13} = \frac{\sum_{k=1}^{M} f_k\lvert X(f_i)\rvert}{\sum_{k=1}^{M}\lvert X(f_k)\rvert}$ (Frequency Center)
$a_7 = \frac{a_4}{a_3}$ (Crest Factor)	$a_{14} = \frac{\sum_{k=1}^{M}(f_k-a_{13})^2\lvert X(f_i)\rvert}{\sum_{k=1}^{M}\lvert X(f_k)\rvert}$ (Standard Frequency)

the data. On the one hand, this means that EMD is a nonlinear decomposition and, on the other hand, it is suitable for analyzing nonlinear and non-stationary data [124].

The algorithm for the decomposition of a given time series $s(t)$ can be described as follows. The first residue $r_0(t)$ is equivalent to the original data, i.e. $r_0(t) = s(t)$. Then the IMFs $c_i(t)$ with $i \geq 1$ are generated from the residues $r_{i-1}(t)$ by repeated application of the so-called sifting process described below. After extracting the ith IMF $c_i(t)$, the next residue is calculated by

$$r_i(t) = r_{i-1}(t) - c_i(t). \tag{2.6}$$

This procedure is repeated until the result of Equation (2.6), that is, the ith residue $r_i(t)$, becomes a monotonic function, and no more IMFs can be extracted. As a result, the decomposition of the original data can be given by

$$s(t) = \sum_{i=1}^{N} c_i(t) + r_N, \tag{2.7}$$

The sifting process for the generation of the ith IMF c_i from the residue r_{i-1} is done via the following iterative scheme. Lower and upper envelopes of the data are generated by using cubic splines for interpolation between the local minima and maxima of the residue, respectively. The mean $m(t)$ of the lower and upper envelope is calculated. The first guess for the IMF is obtained by the difference between the residue r_{i-1} and $m(t)$. Then the first guess for the IMF is treated as the new data, and the sifting process is repeated until a given stoppage criterion is fulfilled. As a consequence of the iteration, the lower and the upper envelopes of the final IMF $c_i(t)$ are nearly symmetric, and the mean of the latter is approximately zero. Moreover, the number of extrema and the number of zero crossings are equal or differ at most by one. IMFs with lower indices correspond to high-frequency bands, while the ones with higher indices correspond to lower frequency bands. These properties of the decomposition make it useful for further data analysis.

However, one major problem with the original EMD is the occurrence of mode mixing, which means that one IMF contains two signals whose frequency bands are totally different, or a signal of a similar scale is observed inside different IMFs whose frequency bands are different [125]. EEMD was developed to solve the mode mixing problem in EMD [126]. Accordingly, Wu and Huang [125] proposed the following steps for EEMD:

1. Create an ensemble from the original data by adding white noise.

2. Decompose each member of the ensemble into IMFs.

3. Compute the ensemble means of the corresponding IMFs.

The added white noise amplitude must not exceed 20% of the standard deviation of the original signal, while the ensemble size for the EEMD can be selected as 200 [45]. The Python package PyEMD with the default stoppage criterion is used for the analysis [127, 128]. The ensemble number and the noise width parameter are set to 200 and 0.2 (20%), respectively.

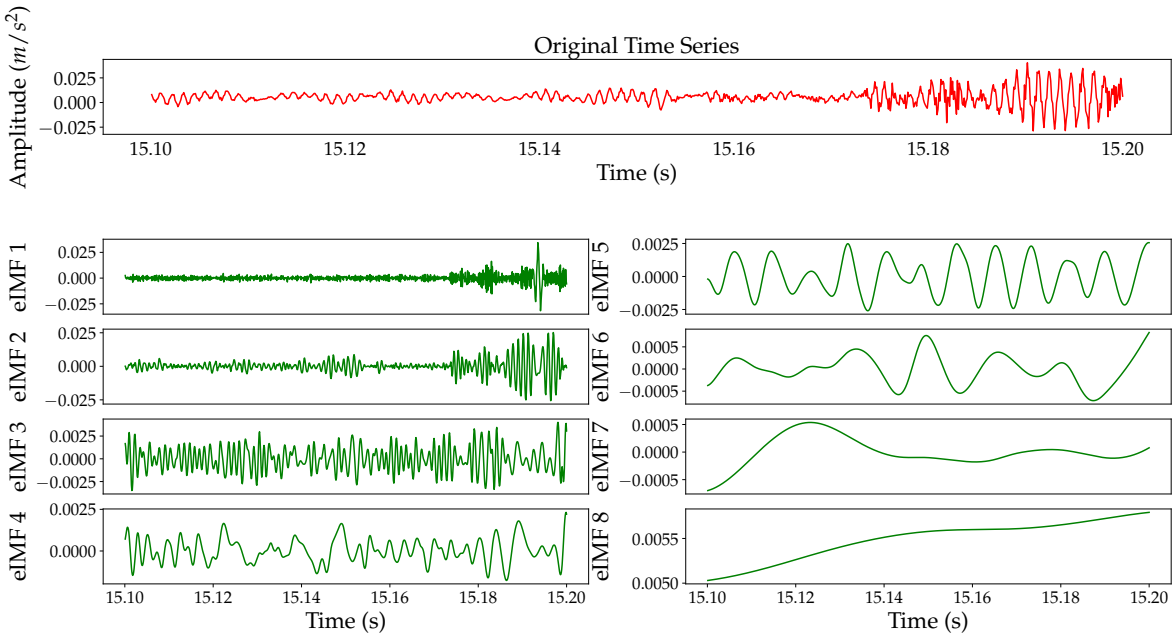


Figure 2.11: The original time series and the corresponding intrinsic mode functions (IMFs) for the case of 5.08 cm (2 inch) stickout, 320 rpm, and 0.0127 cm (0.005 inch) DOC.

2.3.2.1 Selection of Informative Intrinsic Mode Function

In this section, the informative IMF selection is described using the experimental signals from turning experiments explained in Section A.1. In order to obtain features for machine learning from vibration signals using EEMD, the vibration signals are decomposed into IMfs, see Figure 2.11 for an example. For long time series, the computation time is reduced for this step by dividing the signal into shorter segments whose length is approximately 1000 points. The informative IMF selection process is very similar to its WPT counterparts (see Section 2.3.1.1). Specifically, the power spectrum in Figure 2.12 shows that the first IMF includes the high frequency vibrations while higher order IMFs include the low frequency ones. For example, for the 5.08 cm (2 inch) stickout case, the FFT of the second IMF matches the chatter frequency region (900–1000 Hz). Therefore, the second IMF is selected as the informative IMF in this case. The informative IMFs for the other stickout cases are summarized in Table 2.1.

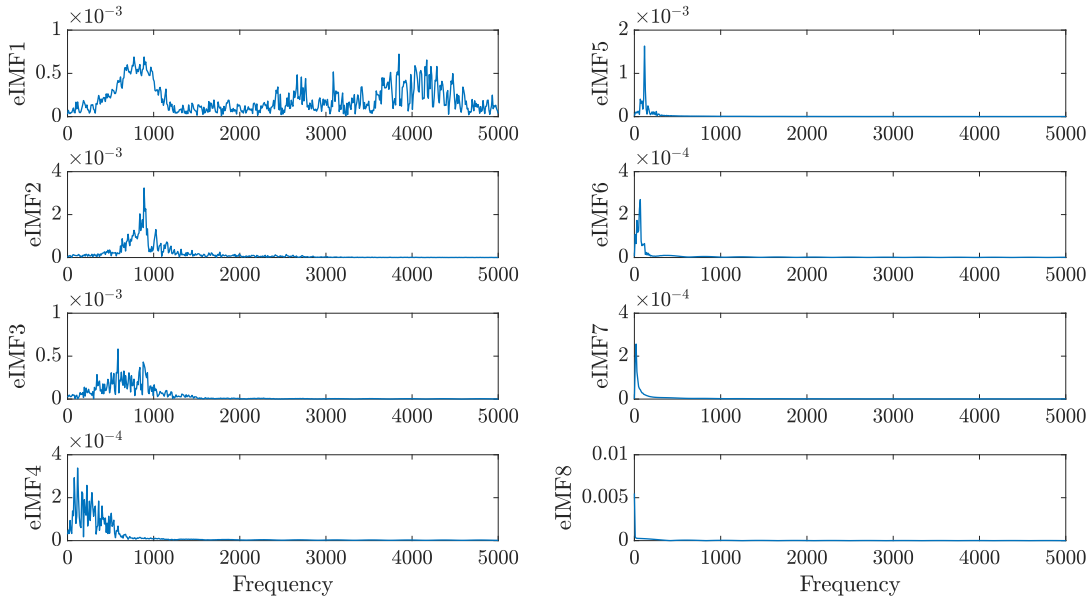


Figure 2.12: The spectrum of each intrinsic mode function (IMF) for the case of 5.08 cm (2 inch) stickout, 320 rpm and 0.0127 cm (0.005 inch) DOC.

2.3.2.2 Feature Extraction Using EEMD

Similar to Chen et al. [45], seven time domain features are extracted from the informative IMF. These features are listed in Table 2.4, and they include the energy ratio, peak to peak value, standard deviation, root mean square, crest factor, as well as skewness, and kurtosis of the signals. The features are computed and then ranked using the Recursive Feature Elimination (RFE) method, which was introduced in Reference [129] and is described in Section 2.3.1.2. The feature matrix for classification is formed starting with the top-ranked feature by itself and then by concatenating, in descending order, the rest of the features one at a time. This results in seven combinations of features, which are then used for classification into chatter and chatter-free cases via four different classifiers similar to the WPT approach (see Section 2.2).

2.3.3 Results

This section shows the classification accuracy for the methods discussed in Section 2.3.1 and 2.3.2 for turning cutting experiments explained in Section A.1. Specifically, Sections

Table 2.4: Time domain features for the intrinsic mode functions $c_i(t_k)$. The parameters t_k and $\bar{c}_i$ represent, respectively, the kth discrete time and the mean of ith IMF.

Feature	Equation
Energy ratio	$f_1 = \frac{\sum_{k=1}^{n} c_i^2(t_k)}{\sum_{i=1}^{I}\sum_{k=1}^{n} c_i^2(t_k)}$
Peak to Peak	$f_2 = max(c_i(t_k)) - min(c_i(t_k))$
Standard Deviation	$f_3 = \sigma(c_i(t_k))$
Root Means Square (RMS)	$f_4 = \sqrt{\frac{1}{n}\sum_{k=1}^{n} c_i^2(t_k)}$
Crest Factor	$f_5 = \frac{max(c_i(t_k))}{f_4}$
Skewness	$f_6 = \frac{\sum_{k=1}^{n}(c_i(t_k)-\bar{c}_i(t_k))^3}{(n-1)f_4^3}$
Kurtosis	$f_7 = \frac{\sum_{k=1}^{n}(c_i(t_k)-\bar{c}_i(t_k))^4}{(n-1)f_4^4}$

2.3.3.1 and 2.3.3.2 show the WPT-based and the EEMD-based results, respectively. The results are obtained by randomly splitting the data from each stickout case into 67% training and 33% testing sets. As described in Sections 2.3.1 and 2.3.2, the features from the informative wavelet packet or informative IMF are extracted, and four different classification algorithms are used for training. Then, each classifier is tested using the corresponding test set. This split-train-test process is repeated 10 times, and the averages and standard deviations of the resulting classification accuracy are tabulated.

2.3.3.1 Wavelet Packet Transform with RFE

In each realization of training data and test data, the feature ranking via RFE is repeated as described in Section 2.3.1.2. Since the training and test sets are different in each realization, ten different rankings of the features are obtained. Figure 2.13 shows the ranking for the 10 iterations where each bar corresponds to a feature whose equation is provided in

Table 4.1. The height of the bar in the figure shows the number of times each feature is ranked for the corresponding rank number. For instance, feature a_{14} (standard frequency) is the feature with the most influence on the classification in all realizations. On the other hand, features a_{11}, a_{12} and a_{13} are ranked second, respectively, in three, four, and three out of ten split-train-test realizations. In general, the features based on the frequency domain are higher ranked than the time domain features.

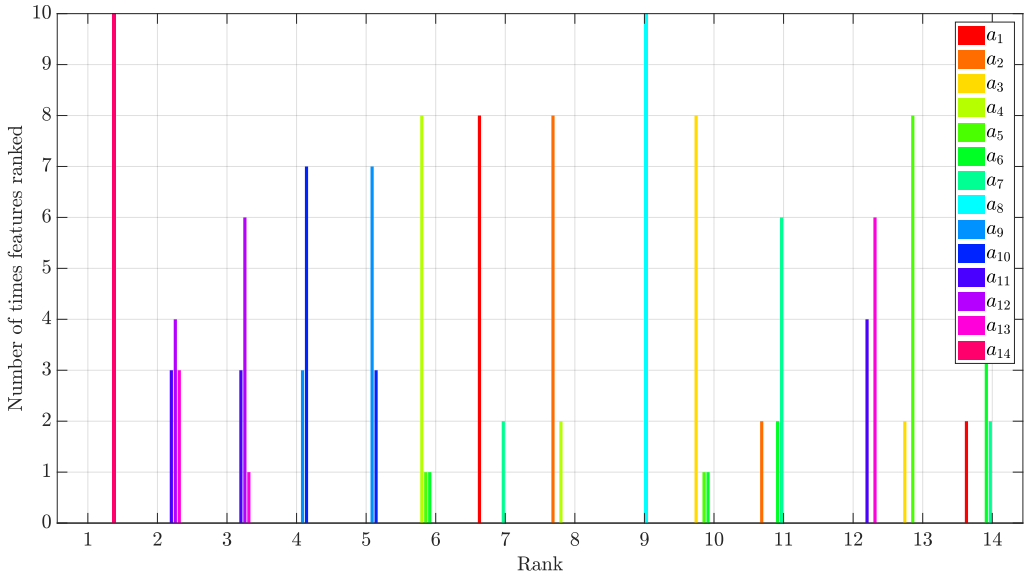


Figure 2.13: Bar plot for feature ranking for 5.08 cm (2 inch) stickout case at level 4 WPT.

The mean and the standard deviation of the accuracy of the classification for the 10 realizations of training and test sets based on the level 4 WPT method are presented in Figure 2.14 for all stickout cases. In this figure, it is seen that when the number of the features is 8 or 10, adding lower ranked features into the feature vector does not affect the result. This shows that RFE ranked the features properly and that lower ranked features do not have an influence on the results.

One difference between the WPT-based approach that is described here and the one described in [5] is that the accuracy of the classifier is investigated using informative wavelet functions computed at each level of the WPT. On average, the level 1 and level 2 WPT leads to better classification results in the test sets than the level 3 and level 4 WPT. This might be

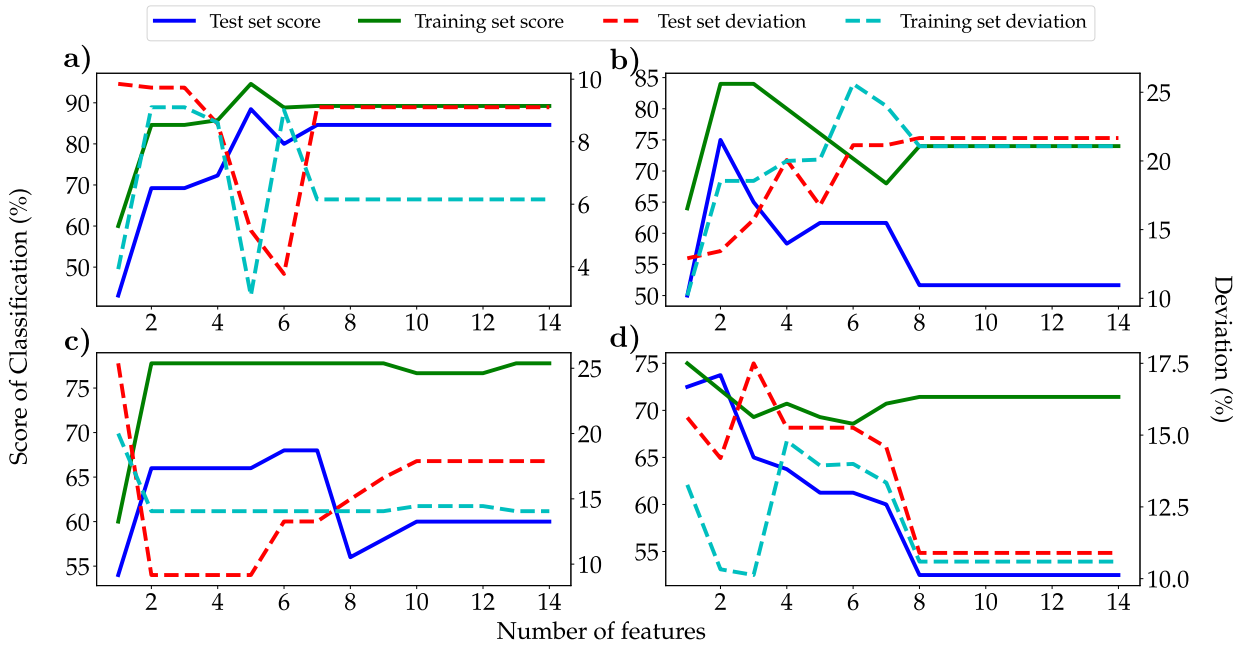


Figure 2.14: Level 4 Wavelet Packet Transform(WPT) feature extraction method results for all stickout cases. (a) 5.08 cm (2 inch), (b) 6.35 cm (2.5 inch), (c) 8.89 cm (3.5 inch), and (d) 11.43 cm (4.5 inch).

attributed to the fact that the lower level WPT contain information in a broader frequency range than the higher level WPT, and for chatter detection, only the detection of chatter frequencies in the spectrum is relevant but not their frequency value or the exact shape of the peaks. The full classification results for each level of the WPT up to level 4 are tabulated in Tables B.1–B.4 of the Appendix. Since the feature ranking is different for each realization of the splitting into training and test data, the ith ranked feature is only denoted by r_i. Below in Table 2.5, the WPT results are reported with the highest average accuracy out of all the different combinations of WPT levels and feature vectors, and they are compared to the results of the EEMD method. The performance of both methods, WPT and EEMD, are also tested with the classifiers explained in Section 2.2. Tables 2.6– 2.7 provide the accuracies obtained from Level 1 and Level 2 WPT and EEMD feature extraction methods with four different classifiers and compare the methods to each other.

Table 2.5: Comparison of the classification results obtained with SVM classifier and run times for each chatter detection method. Given run times include feature computation and classification for EEMD and level 4 WPT.

	Classification Results			Time Comparison (seconds)	
Stickout Length	WPT Level	WPT	EEMD	WPT	EEMD
5.08 cm (2 inch)	1	$93.9\% \pm 5.8\%$	$84.2\% \pm 0.8\%$	115.99	14540.06
6.35 cm (2.5 inch)	2	$100.0\% \pm 0.0\%$	$78.6\% \pm 1.2\%$	36.65	3371.58
8.89 cm (3.5 inch)	1	$84.0\% \pm 15.0\%$	$90.7\% \pm 1.4\%$	4.51	1583.38
11.43 cm (4.5 inch)	2	$87.5\% \pm 11.2\%$	$79.1\% \pm 1.2\%$	6.53	3096.07

Table 2.6: Results obtained by using Level 1 WPT and EEMD feature extraction methods with four different classifiers.

	WPT				EEMD			
Stickout Length	SVM	Logistic Regression	Random Forest	Gradient Boosting	SVM	Logistic Regression	Random Forest	Gradient Boosting
5.08 cm (2 inch)	93.9%	84.6%	93.1%	90.0%	84.2%	93.5%	94.8%	94.9%
6.35 cm (2.5 inch)	85.0%	71.7%	91.7%	91.7%	78.6%	79.4%	80.1%	82.2%
8.89 cm (3.5 inch)	84.0%	94.0%	100.0%	96.0%	90.7%	89.0%	93.5%	94.5%
11.43 cm (4.5 inch)	78.8%	81.3%	86.3%	87.5%	79.1%	78.7%	81.6%	81.4%

2.3.3.2 Ensemble Empirical Mode Decomposition with RFE

Similar to Section 2.3.3.1, EEMD is combined with RFE and utilizes four different classifiers in each realization of the splitting into test and train data sets. The classification accuracy is on average better than the results from the level 3 and level 4 WPT and comparable to the accuracy of the lower level WPT. The combination with the best accuracies in each cutting case is reported when comparing the different methods in Table 2.5. In this table, the results highlighted with dark blue represent the highest accuracy across a given row, while those highlighted in light blue have an average accuracy which is encapsulated by the error bars of the method with the highest average accuracy.

Table 2.5 shows that features based on the WPT algorithm give the highest accuracy for three stickout cases out of four cutting configurations. Specifically, feature extraction with WPT and RFE is the most accurate for the 5.08 and 6.35 cm (2 2.5 and 4.5) stickout cases

Table 2.7: Results obtained by using Level 2 WPT and EEMD feature extraction methods with four different classifiers.

	WPT				EEMD			
Stickout Length	SVM	Logistic Regression	Random Forest	Gradient Boosting	SVM	Logistic Regression	Random Forest	Gradient Boosting
5.08 cm (2 inch)	91.5%	87.7%	93.8%	90.0%	84.2%	93.5%	94.8%	94.9%
6.35 cm (2.5 inch)	100.0%	80.0%	95.0%	96.7%	78.6%	79.4%	80.1%	82.2%
8.89 cm (3.5 inch)	78.0%	58.0%	94.0%	78.0%	90.7%	89.0%	93.5%	94.5%
11.43 cm (4.5 inch)	87.5%	78.8%	88.8%	80.0	79.1%	78.7%	81.6%	81.4%

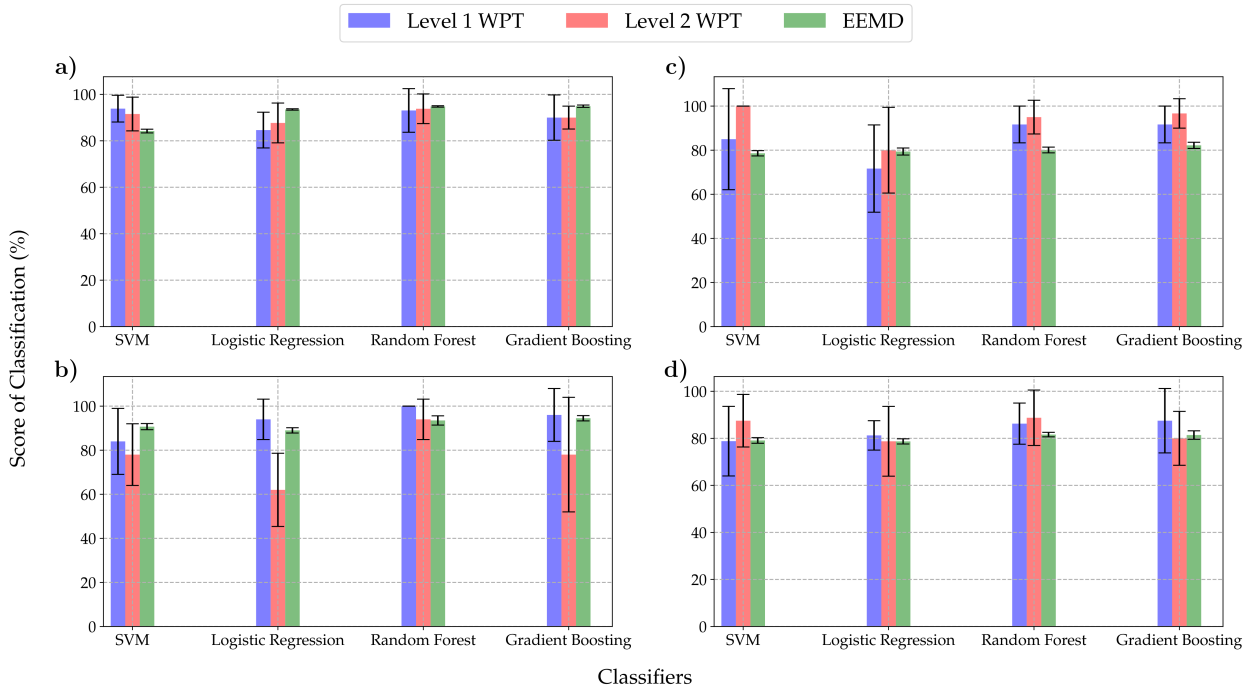


Figure 2.15: Bar plot including the error bars of the classification results for Level 1 WPT, Level 2 WPT and EEMD with four different classifier. a) 5.08 cm (2 inch) stickout size, b)6.35 cm (2.5 inch) stickout size, c)8.89 cm (3.5 inch) stickout size, d) 11.43 cm (4.5 inch) stickout size.

scoring 93.9%, 100.0% and 87.5% respectively. While the results from EEMD give the highest accuracies for 8.89 cm (3.5 inch) stickout cases, the WPT result for this case still lies within the error bars of EEMD results. The results with different classifiers other than SVM are also provided in Tables 2.6 and 2.7. In Table 2.6, the performance of Level 1 WPT is better than EEMD since WPT has the highest accuracies in three cutting configuration cases and EEMD results are in the error bars of WPT results. On the other hand, Table 2.7 indicates that both methods have the highest accuracy for two cutting configurations. These two tables also provide evidence that lower level (Level 1) WPT outperforms EEMD. Further, 100% accuracy is observed in Tables 2.6 and 2.7 for two different cutting configurations. These cutting configurations have the lowest number of time series as experimental data. Since time series are not split into smaller pieces for the WPT method, so the size of the test set is quite small, and it is possible to get such high results.

The standard deviation of the WPT results is quite high, as seen from Table 2.5 and Figure 2.15 since the computation time for this method does not require splitting a long time series into smaller pieces. Therefore, the total number of samples for identical stickout cases is smaller in comparison to the EEMD method, where long time series were split into shorter ones of approximately 1000 points, thus increasing the number of samples and resulting in tighter error bars. Therefore, the amount of deviation can be reduced, especially for the WPT-based approach, by increasing the size and the number of the training sets. In addition, Table 2.5 compares the run time in seconds for each of the different featurization methods for chatter detection. These comparisons were performed using a Dell Optiplex 7050 desktop with Intel Core i7-7700 CPU and 16.0 GB RAM. It can be seen that feature extraction with WPT and RFE is the fastest across all of the stickout cases. This study points out that the built-in WPT package that is used is highly optimized, whereas, in comparison, the EEMD does not enjoy the same level of code optimization. Moreover, for EEMD, the EMD is performed for an ensemble of time series with an ensemble size 200, which needs much higher computation effort and can be reduced by varying the ensemble

parameters of the EEMD.

2.4 Traditional Time and Frequency Domain Features

2.4.1 Feature Extraction

This section describes feature extraction using three traditional signal processing approaches. These approaches are Fast Fourier Transform (FFT), Power Spectral Density (PSD), and Autocorrelation Function(ACF). Features for FFT, PSD, and ACF are obtained by using their peaks' coordinates. The x and y components of the first five peaks in each of these three functions are used as features. Although there are some built-in commands for peak finding in the most common scientific software tools, reliable peak selection remains a challenging task. This is due to the large number of returned 'bumps' that correspond to local maxima that are artifacts of noise and are not true features of the signal. Therefore, some constraints are imposed to sift out the redundant peaks and capture the most useful ones. The FFT, PSD, and ACF peaks are selected by defining the minimum peak height (MPH) and the minimum distance between two consecutive peaks. The minimum peak distance (MPD) was kept constant for all sequences, while the minimum peak height was computed as a fraction of the difference between the 5th and the 95th percentile of the peaks according to

$$MPH = y_{\min} + \alpha(y_{\max} - y_{\min}), \text{ where } \alpha \in [0, 1], \tag{2.8}$$

and $y_{\min}$ is the 5th percentile of the amplitude of the FFT/PSD/ACF, while $y_{\max}$ is the 95th percentile. Figure 2.16 provides the original cutting signals along with its spectrum including the first five peaks. In Figure 2.16, two sets of the first five peaks are provided, and the peaks shown in the figure are selected with respect to the minimum peak distance (MPD) parameter. The figure shows that $MPD = 500$ chooses more points near the maximum amplitude in comparison to $MPD = 2500$. However, some of the spectra of the cutting signals do not contain five peaks if $MPD = 2500$ is used. Therefore, the MPD parameter

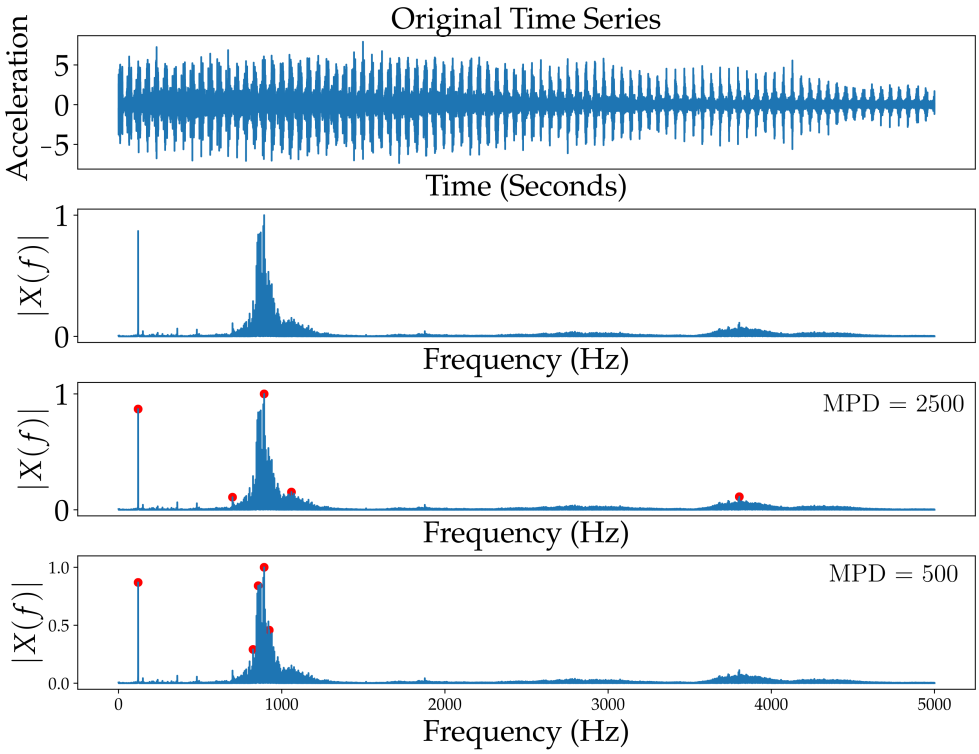


Figure 2.16: First five peaks in the Fast Fourier Transform (FFT) plot of the time series of acceleration signal collected from cutting configuration with 5.08 (2 in) cm overhang length, 320 rpm rotational speed of the spindle, and 0.0127 cm (0.005 inc) depth of cut. (Minimum Peak Distance (MPD): 500 and 2500).

for FFT is set to 500 to consistently extract five peaks from the FFT of all signals, but $MPD = 1000$ is used with the auto-correlation function. Because the power spectral density plots were smooth, the MPD parameter is not used as a constraint for them.

The feature matrices were given as input to four different supervised machine learning techniques: Support Vector Machine (SVM), Logistic Regression (LR), Random Forest Classification (RF), and Gradient Boosting (GB). Vibration signals for each overhang size were split into %67 training set and %33 test set. Recursive Feature Elimination (RFE) was then used to rank the features. Splitting the data and performing classification were performed ten times. The resulting mean accuracies and standard deviations are provided in Section 2.4.2.

2.4.2 Results

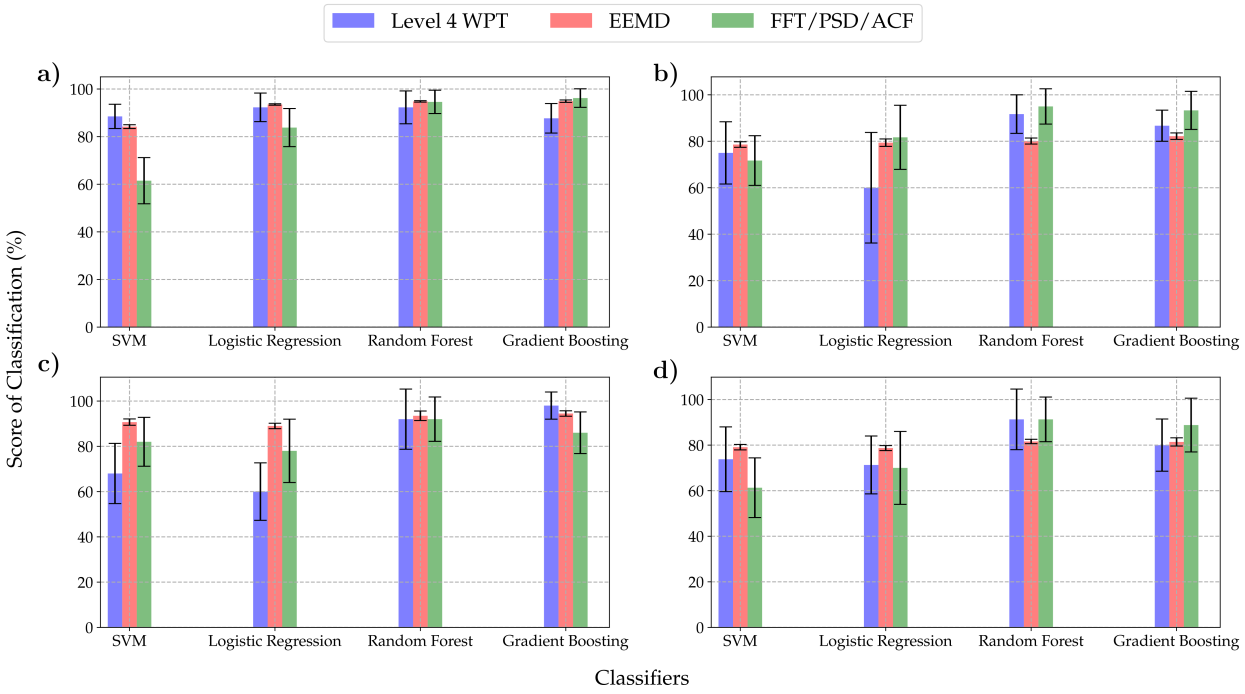


Figure 2.17: Comparison of classification results of Level 4 WPT, EEMD, and FFT/PSD/ACF methods based on four classifiers with Recursive Feature Elimination (RFE). a) 5.08 cm (2 inch) overhang length b) 6.35 cm (2.5 inch) overhang length c) 8.89 cm (3.5 inch) overhang length d) 11.43 cm (4.5 inch) overhang length.

Table 2.8: Results obtained with the signal processing feature extraction method for four overhang lengths with four different classifiers.

	Without RFE				With RFE			
	5.08 cm (2 inch)		6.35 cm (2.5 inch)		5.08 cm (2 inch)		6.35 cm (2.5 inch)	
Classifier	Test Set	Training Set	Test Set	Training Set	Test Set	Training Set	Test Set	Training Set
SVM	46.2% ± 6.0%	100.0% ± 0.0%	38.3% ± 13.0%	100.0% ± 0.0%	61.5% ± 9.7%	69.6% ± 8.3%	71.7% ± 10.7%	86.0% ± 6.6%
LR	78.5% ± 12.8%	100.0% ± 0.0%	76.7% ± 8.2%	100.0% ± 0.0%	83.8% ± 8.0%	82.3% ± 5.8%	81.7% ± 13.8%	89.0% ± 8.3%
RF	93.8% ± 3.1%	100.0% ± 0.0%	86.7% ± 10%	100.0% ± 0.0%	94.6% ± 4.9%	98.5% ± 1.9%	95.0% ± 7.6%	100.0% ± 0.0%
GB	84.6% ± 6.9%	100.0% ± 0.0%	76.7% ± 17.0%	100.0% ± 0.0%	96.2% ± 3.9%	100.0% ± 0.0%	93.3% ± 8.2%	100.0% ± 0.0%
	Without RFE				With RFE			
	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)		8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
SVM	62.0% ± 24.4%	100.0% ± 0.0%	43.8% ± 17.0%	100.0% ± 0.0%	82.0% ± 10.8%	95.6% ± 5.4%	61.3% ± 13.1%	68.6% ± 9.1%
LR	76.0% ± 12.0%	100.0% ± 0.0%	76.3% ± 10.4%	100.0% ± 0.0%	78.0% ± 14.0%	88.9% ± 8.6%	70.0% ± 16.0%	82.9% ± 6.5%
RF	90.0% ± 9.4%	100.0% ± 0.0%	90.0% ± 9.4%	100.0% ± 0.0%	92.0% ± 9.8%	100.0% ± 0.0%	91.3% ± 9.8%	99.3% ± 2.1%
GB	86% ± 23.7%	100.0% ± 0.0%	83.8% ± 11.3%	100.0% ± 0.0%	86.0% ± 9.2%	100.0% ± 0.0%	88.8% ± 11.8%	100.0% ± 0.0%

This section provides the results obtained for turning cutting experiments explained in Section A.1 and compares traditional feature extraction results to WPT and EEMD approaches. Four different classifiers explained in Section 2.2 were utilized to compare traditional signal processing based feature extraction methods to the WPT/EEMD results

Table 2.9: Cross validated results obtained with signal processing feature extraction method for four overhang lengths with four different classifiers.

	Without RFE				With RFE			
	5.08 cm (2 inch)		6.35 cm (2.5 inch)		5.08 cm (2 inch)		6.35 cm (2.5 inch)	
Classifier	Test Set	Training Set	Test Set	Training Set	Test Set	Training Set	Test Set	Training Set
SVM	74.2% ± 4.7%	94.8% ± 4.7%	58.3% ± 3.1%	98.5% ± 3.1%	90.0% ± 1.9%	89.8% ± 1.9%	73.3% ± 6.5%	75.2% ± 6.5%
LR	74.2% ± 0.0%	100.0% ± 0.0%	71.7% ± 0.0%	100.0 ± 0.0%	79.2% ± 1.3%	89.5% ± 1.3%	73.3% ± 6.7%	82.5% ± 6.7%
RF	89.2% ± 0.0%	100.0% ± 0.0%	93.3% ± 0.0%	100.0% ± 0.0%	91.7% ± 1.3%	95.2% ± 1.3%	93.3% ± 0.0%	100.0% ± 0.0%
GB	91.7% ± 0.0%	100.0% ± 0.0%	93.3% ± 0.0%	100.0% ± 0.0%	97.5% ± 0.0%	100.0% ± 0.0%	100.0% ± 0.0%	100.0% ± 0.0%
	Without RFE				With RFE			
	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)		8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
SVM	86.7% ± 0.0%	100.0% ± 0.0%	41.7% ± 7.0%	88.4% ± 7.0%	80.0% ± 5.8%	91.1% ± 5.8%	83.3% ± 2.3%	81.8% ± 2.3%
LR	93.3% ± 0.0%	100.0% ± 0.0%	46.7% ± 6.6%	94.0% ± 6.6%	86.7% ± 3.6%	92.9% ± 3.6%	78.3% ± 3.2%	78.3% ± 3.2%
RF	93.3% ± 0.0%	100.0% ± 0.0%	91.7% ± 0.0%	100.0% ± 0.0%	93.3% ± 0.0%	100.0% ± 0.0%	96.7% ± 1.5%	95.5% ± 1.5%
GB	73.3% ± 0.0%	100.0% ± 0.0%	85.0% ± 0.0%	100.0% ± 0.0%	80.0% ± 0.0%	100.0% ± 0.0%	91.7% ± 0.0%	100.0% ± 0.0%

obtained from Reference [9]. The results obtained using these classifiers for each cutting configuration are provided in Table 2.8. The table shows that classifiers trained with the traditional signal processing features have an overfitting problem (significantly higher accuracy rates when training versus testing) when RFE is not utilized. Recursive feature elimination was used to solve this problem (see Section 2.3.1.2). Table 2.8 only reports the best results obtained from the classifiers for each overhang length. These results are also compared to the ones obtained with WPT and EEMD in Figure 2.17. It is seen that the FFT/PSD/ACF method has higher accuracies in Figure 2.17.

The feature extraction is explained in Section 2.4.1. Classification for this method has been performed in the same way it is performed for WPT/EEMD. The left hand side of Table 2.8 shows the classification results without using feature ranking. Note how the test accuracy is significantly lower than the training accuracy. This is a typical symptom of overfitting, i.e., using too many (unnecessary) features for training. In contrast, the right hand side of the same table shows that using feature ranking mitigated the overfitting problem. In addition, if the results in Table 2.8 are compared, it is seen that the FFT/PSD/ACF based feature extraction methods have better accuracies than WPT and EEMD. Another approach to combat overfitting is to utilize Cross Validation (CV). Table 2.9 provides the results obtained with CV where 10-fold CV was used for the 5.08 cm (2 inch) and 11.43 cm (2.5 inch) cases, while 5-fold CV was used for the remaining cutting configurations. The results show

that while CV somewhat mitigates the overfitting problem, it does not completely solve it. For example, there is overfitting for the classification accuracy of SVM for the results obtained without using RFE in Table 2.8. Although CV decreased the difference between mean accuracies of test set and training set, there is still at least 20% accuracy difference between training and test sets for overhand lengths 5.08, 6.35, and 11.43 cm (2, 2.5, and 4.5 inch). In addition, a decrease in the standard deviation of the results is observed in Table 2.9 in comparison to the ones in Table 2.8. Figure 2.17 provides bar plots for the classification accuracy for each cutting configuration along with the associating error bars. Figure 2.18 shows how many times each feature was ranked for the traditional signal processing based methods. This indicates where the most highly ranked features come from. The figure shows that the most highly ranked features correspond to the FFT peaks, followed by PSD, then ACF features.

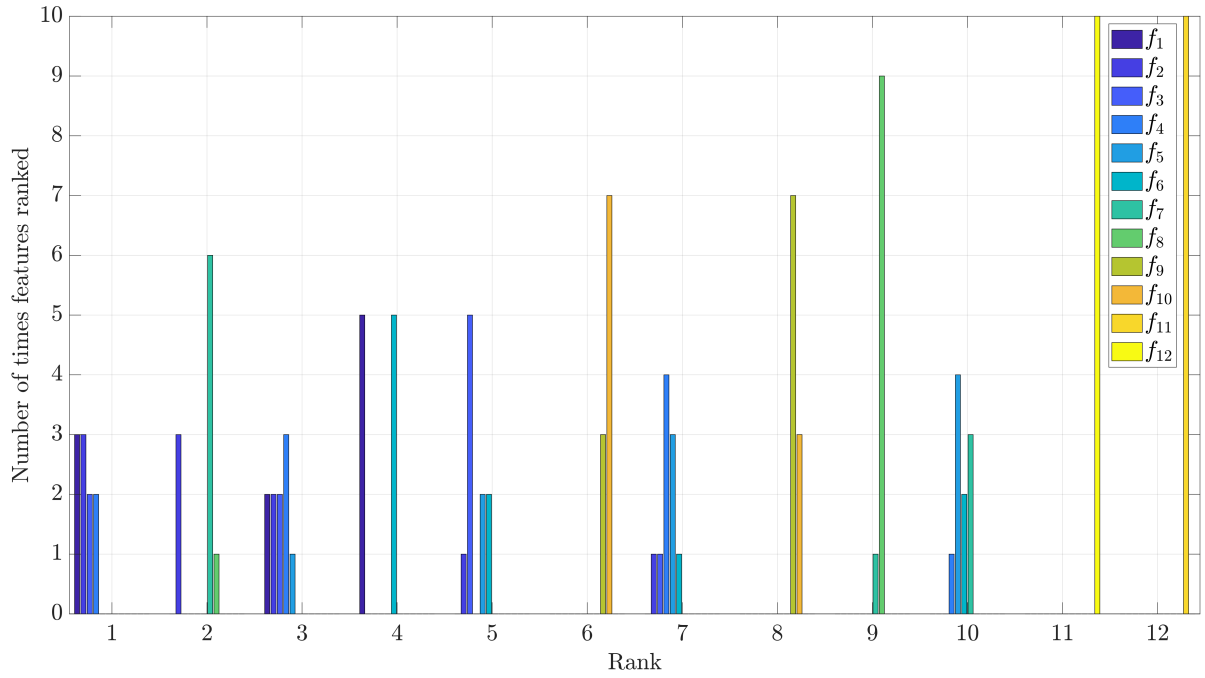


Figure 2.18: Bar plot for feature ranking obtained from SVM-RFE for 5.08 cm (2 inch) overhang case with FFT/PSD/ACF based method. $(f_1, \ldots, f_4)$, $(f_5, \ldots, f_8)$ and $(f_9, \ldots, f_{12})$ belong to features obtained from peaks of FFT, PSD and ACF respectively.

2.5 Similarity Measures of Time Series

2.5.1 Dynamic Time Warping (DTW)

Dynamic Time Warping is an algorithm that is capable of measuring distance or similarity between two time series even if they have dissimilar lengths. Let TS_1 and TS_2 be two time series with elements x_i and y_j whose lengths are m and n as follows:

$$TS_1 = x_1, x_2, \ldots, x_i, \ldots, x_m, \tag{2.9}$$

$$TS_2 = y_2, y_2, \ldots, y_j, \ldots, y_n. \tag{2.10}$$

Berndt and Clifford state that the warping path $w_k = (x_{i(k)}, y_{j(k)})$ between two time series can be represented by mapping the corresponding elements of the time series on $m \times n$ matrix (see Figure 2.19 for warping path example) [130]. The warping path is composed of the points w_k, which indicate alignment between the elements $x_{i(k)}$ and $y_{j(k)}$ of the time series. The length L of the warping path fulfills the constraints $m \leq L \leq n$, where it is assumed that $n \geq m$. For instance, w_3 in Figure 2.19b corresponds to the alignment of x_2 and y_3. In general, warping paths are not unique, and several warping paths can be generated for the same two time series. For two different time series, the DTW algorithm chooses the warping path that gives the minimum distance between the element pairs under certain constraints. While there are several options for computing the distance between a pair (x_i, y_j) of elements of the time series, in this implementation, the Manhattan distance $d(x_i, y_j) = \|x_i - y_j\|_1$ is used. The minimization of the distance between TS_1 and TS_2 in the DTW algorithm can then be written according to ([130]) such that

$$D_{\mathrm{TW}}(TS_1, TS_2) = min\left(\sum_{k=1}^{L} d(w_k)\right). \tag{2.11}$$

There are several restrictions to define the optimum warping path. These are monotonicity, continuity, adjustment window condition, slope constraint, and boundary conditions. These restrictions are applied to the alignment window to reduce the possible number of

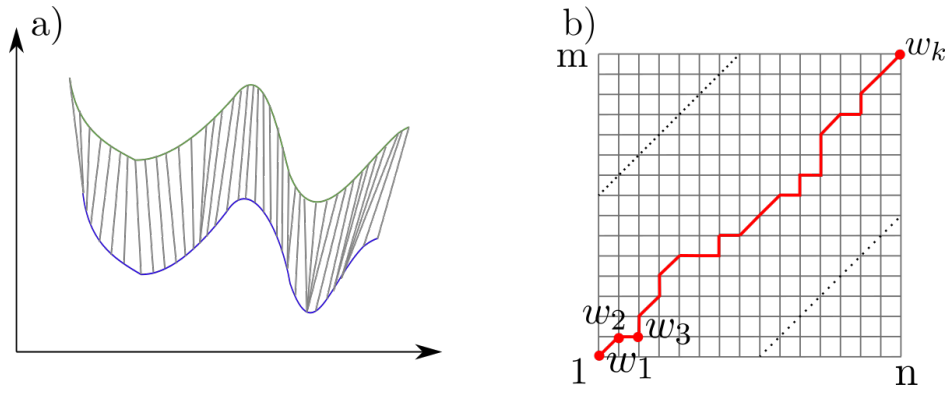


Figure 2.19: DTW alignment (a) and warping path(b) for two different time series.

warping paths since there is an excessive number of possibilities for warping paths without any constraint ([131]).

- Monotonicity : The indices i and j should always either increase or stay the same such that $i(k) \geq i(k-1)$ and $j(k) \geq j(k-1)$.

- Continuity: The indices i and j can only increase at most by one such that $i(k) - i(k-1) \leq 1$ and $j(k) - j(k-1) \leq 1$.

- Boundary condition: The warping paths should start where i and j are equal to 1 and should end where $i = n$ and $j = m$.

- Adjustment window condition: The warping path with minimum distance is searched on a restricted area on the alignment window to avoid significant timing difference between the two paths ([131]). The restricted area is given by $i - r \leq j \leq i + r$.

- Slope constraint: This condition avoids significant movement in one direction ([130]). After a steps in horizontal or vertical direction, it cannot move in the same direction without having b steps in the diagonal direction ([131]). The effective intensity of the slope constraint can be defined as $P = b/a$. P is chosen as 1, which was reported as an optimum value in an experiment on speech recognition ([131]).

In this work, the distances between the time series are computed using the cDTW package. There is another widely adopted algorithm named FastDTW [132]. The time complexity

of FastDTW algorithm is given as $N(8r + 14)$, while cDTW package has time complexity of rN [132, 133]. N is the number of points in the time series, and r is a parameter for the adjustment window condition explained above. Both packages have similar time complexity, which is $\mathcal{O}(N)$.

2.5.2 K-Nearest Neighbor (KNN)

In this approach, K-Nearest Neighbor (KNN) algorithm is used to train a classifier. KNN is a supervised machine learning algorithm based on classifying objects with respect to labels of nearest neighbors ([134]). The 'K' corresponds to the number of neighbors chosen to decide the label of newly introduced samples. Figure 2.20 shows an example that illustrates the classification process with KNN.

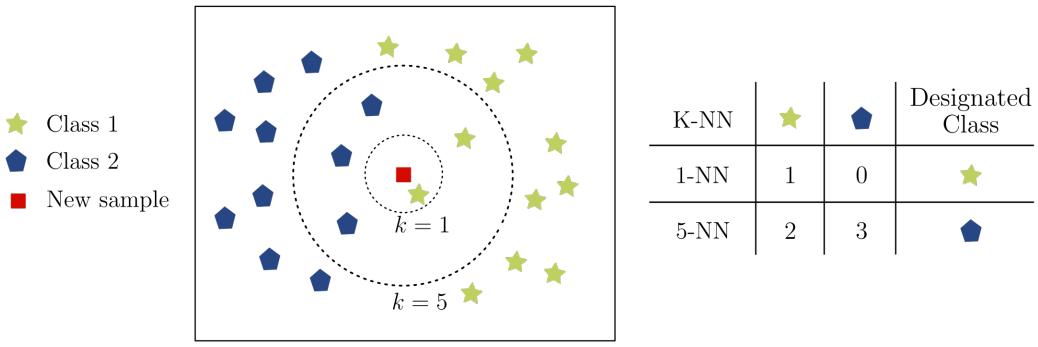


Figure 2.20: K-Nearest Neighbor classification example for two-class classification.

Specifically, Figure 2.20 assumes that there are two different classes for a classification problem denoted by pentagons and stars. Pentagons and stars belong to the training set, and the red square belongs to a new sample from the test set. When a new sample is encountered (the square in the figure), a tag is assigned based on the number of K nearest neighbors to each class. For instance, for the 1-NN case, the closest neighbor is from the star class; therefore, the test sample is tagged as a star class. On the other hand, for the 5-NN case, the test sample has two neighbors from the star class and three neighbors from the pentagon class. Consequently, the label for the test sample is set as pentagon since there are

more neighbors from this class than there are from the star class. If the number of nearest neighbors is the same for multiple classes, the class label is assigned randomly with equal probability ([135]).

2.5.3 Similarity matrices and classification

DTW provides a measure of how different/similar any pair of time series TS_1 and TS_2 is. By comparing N time series $TS_1, \ldots, TS_N$ with each other, similarity matrices whose entries are the distance between the two corresponding time series can be generated. Since DTW is commutative, the resulting matrices are symmetric. Consequently, the resulting similarity matrix for DTW requires $N(N-1)/2$ computations. The similarity matrices can then be combined with a K-Nearest Neighbor classifier to inform us whether the time series corresponds to chatter or chatter-free cutting. When chatter occurs in metal cutting, the dominant frequencies of the vibrations at the tool and workpiece change from harmonics of the spindle rotation period to chatter frequencies, which are close to some eigenfrequecies of the mechanical structure. Moreover, the amplitude of the vibrations increases significantly. This characteristic behavior can be used to distinguish between stable cutting and chatter by comparing current time-domain signals (e.g. acceleration) with existing labeled signal segments from a training phase. During classification, the data set is split into training and test sets. Indices of the training set and test set samples are found, and then the distance matrix for the training set and test set is generated by using the square distance matrix that was computed in advance for all the cases. After obtaining these distances, the nearest training set samples to the test sample are found based on the selected number of nearest neighbors. Labels of each nearest neighbor are counted as shown in the illustration in Figure 2.20. The label with the highest count is assigned to the test sample as the predicted label. In this application, either chatter or chatter-free labels are assigned to the test sample. After repeating these processes for each of the test samples, the predicted labels are compared with the ground truth and define the accuracy of the DTW method. Splitting

data into training and test sets is repeated 10 times. The distances between new test sets and training sets do not have to be computed since the pairwise distances between all samples are already computed in the beginning. Finding the indices of the samples in each iteration will be enough to generate new similarity matrices for training and test sets. The standard deviation of the classification is also provided since the classification is repeated 10 times.

When a new test sample is introduced to a classifier, distance computations between all training samples and the test sample is required, which can be computationally expensive. Therefore, the Approximate and Eliminate Search Algorithm (AESA) (see Section 2.5.4) is implemented to reduce the number of DTW computations per new test sample.

2.5.4 Approximate and Eliminate Search Algorithm (AESA)

Approximate and Eliminate Search Algorithm (AESA) is a method designed for reducing the number of distance computations during the test phase of classification. Derivation of the AESA starts with the question of whether DTW is a metric or not. There are four requirements for a function to be a metric [136]:

1. $D(x,y) \geq 0$,
2. $D(x,y)=0 \iff x=y$,
3. $D(x,y) = D(y,x)$,
4. $D(x,z) \leq D(x,y)+D(y,z)$.

Although DTW always satisfies the first two properties, the commutativity condition is only satisfied when the DTW algorithm does not approximate the distance measure. Therefore, if the exact DTW is computed, then the first three conditions will be satisfied. However, the fourth condition may not be satisfied, and a combination of three time series that violate the triangular inequality can be found in a data set. Consequently, DTW is not accepted

as a metric. Depending on the data set, the fourth condition may or may not be satisfied; therefore, DTW can still be used by relaxing the strict triangular inequality condition. Specifically, Ruiz et al. introduced the triangle inequality looseness in [105] as follows

$$H(x, y, z) = D(x, y) + D(y, z) - D(x, z), \tag{2.12}$$

where x, y and z represents the time series in a data set. Loose triangular inequality is also defined as

$$D(x, y) + D(y, z) \geq D(x, z) + H_L. \tag{2.13}$$

Algorithm 2.1: AESA

Input: P(set of training samples), D_{train}(pairwise distance matrix between training set samples), $c \in P$(pseudocenter of P), $x \in P$(test sample), $H \in \mathbb{R}$(looseness constant), D_{TW}(cDTW function)
Output: Nearest sample (n) and assigned label count=0

```
while E≠ P do
    if count=0 then
        s = c
        U = {c}
        n = c
        E = {c} ∪ {q | D_TW(q,c)>2D_TW(x,c)-H or D_TW(q,c)< H, where q∈ {P-{c}}}
    else
        s = q, such that min( Σ_{∀u∈U} |D_TW(q,u) − D_TW(x,u)|), where q ∈ {P − E}
        D_TW(x,s) = cDTW(x,s)
        U = U ∪ {s}
        E = E ∪ {s}
        if D_TW(x,s)< D_TW(x,n) then
            Q=U
            n=s
        else
            Q={s}
        end if
        for every q ∈ Q do
            E = E ∪ {p| D_TW(p,q)> D_TW(x,q)+D_TW(x,n)-H or D_TW(p,q)< D_TW(x,q)-
D_TW(x,n)+H, where p ∈ {P-E}}
        end for
        count=count+1
    end if
end while
```

Ruiz et al. call DTW distance a loose metric space when a data set does not violate the triangular inequality [105]. All training samples are considered as potential candidates for the nearest sample to a new test set. The main purpose of the algorithm is to eliminate these training samples and approximate the nearest one correctly. In addition, the algorithm performs the classification part and assigns the label of the nearest training sample to the test sample. In other words, it applies 1-NN classification, where the classification step can be part of the AESA algorithm. Results based on 1-NN classification will be presented in this implementation.

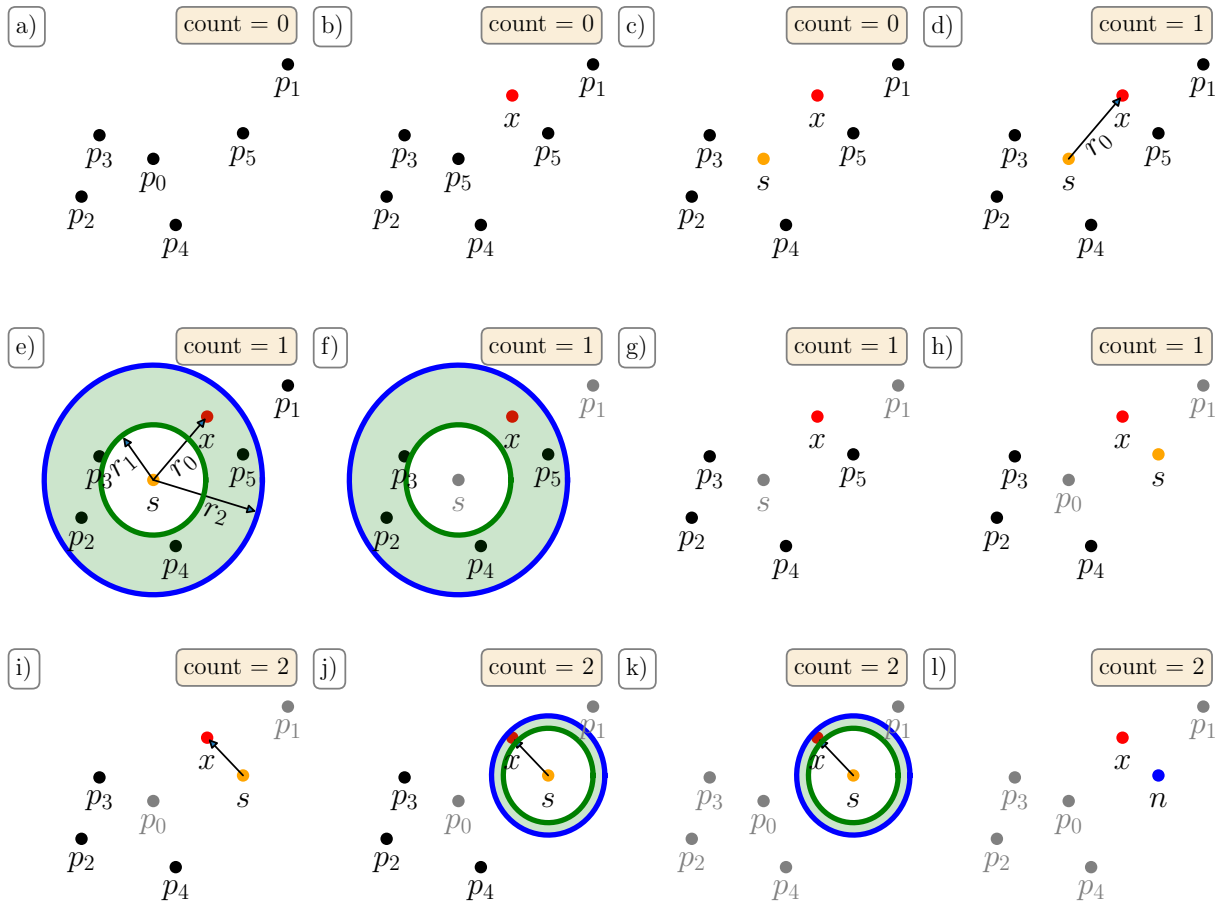


Figure 2.21: Illustration of elimination and approximation steps of AESA. *Count* refers to the number of distance computations made in the algorithm.

The pseudo code for the algorithm is given in Algorithm 2.1 ([105]). Illustrations of

some steps are provided in Figure 2.21. Assume that training set is denoted as P and its samples p_i are shown in Figure 2.21a. A new test sample x is introduced to the classifier (see Figure 2.21b). The aim of AESA is to classify x as accurately as possible with fewer DTW distance computations. The first step of the algorithm is to select a first candidate for the nearest training sample s to x. Reference [105] points out that using pseudo center c of P as the first candidate improves the results; alternatively, s can be randomly selected. The definition of the pseudo center is

$$PC(P) = \{c | \sum_{\forall p \in P} D_{TW}(c,p) = \min_{\forall q \in P}(\sum_{\forall p \in P} D_{TW}(q,p))\}. \tag{2.14}$$

After this choice of s in Figure 2.21c, the distance between s and x is computed. That distance is shown in Figure 2.21d) with r_0, and the count number increases by one. The nearest sample n to x is defined by comparing the distances made inside of the algorithm. Since only one distance computation is performed so far, s is assigned as the nearest sample n. The approximation step is completed in the first iteration, now the elimination should be performed. Reference [105] defined the elimination criteria such that the training set samples p_i which do not satisfy $D_{TW}(x,p_i) < D_{TW}(x,n)$ are eliminated. Applying Equation (2.13)

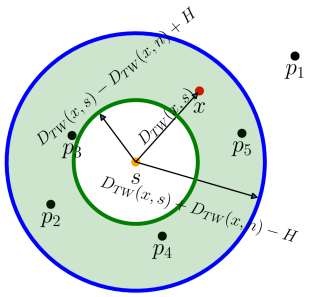


Figure 2.22: Illustration for elimination criteria.

yields two elimination criteria such that

$$\begin{aligned} D_{TW}(p_i,s) &< D_{TW}(x,s) + D_{TW}(x,n) - H \\ D_{TW}(p_i,s) &> D_{TW}(x,s) - D_{TW}(x,n) + H. \end{aligned} \tag{2.15}$$

These two conditions correspond to the two circles shown in Figure 2.21e with r_1 and r_2. In Figure 2.21e, the region where we look for possible candidates for the nearest samples (n) is

defined based on the elimination criteria. The training samples outside of the shaded, green region are eliminated as shown in gray in Figure 2.21f thus completing the first iteration. The set that includes the eliminated samples is called E. Iterations continue until $P = E$. Only two samples are eliminated (see Figure 2.21g) so far in this example. Therefore, we continue searching for new s in the second iteration. The new $s = q$ is selected for iterations except the first one such that

$$\min(\sum_{\forall u \in U} |D_{TW}(q,u) - D_{TW}(x,u)|), \tag{2.16}$$

where $q \in \{P - E\}$ [105]. This choice makes p_5 the new s as shown in orange in Figure 2.21h. Now, the distance between s and x is computed again, and the count is increased by one (see Figure 2.21i). Then, circles that define the elimination criteria are drawn (see Figure 2.21j), and it is seen that all remaining samples are outside of the defined region. Therefore, all of them are eliminated (see Figure 2.21k). Since all training set samples are eliminated, there will be no further iterations in the algorithm. In the last step, s is assigned as nearest sample n to the test sample x if $D_{TW}(x,s) < D_{TW}(x,n)$ (see Figure 2.21l). This completes the algorithm, and the label of n can be assigned to x.

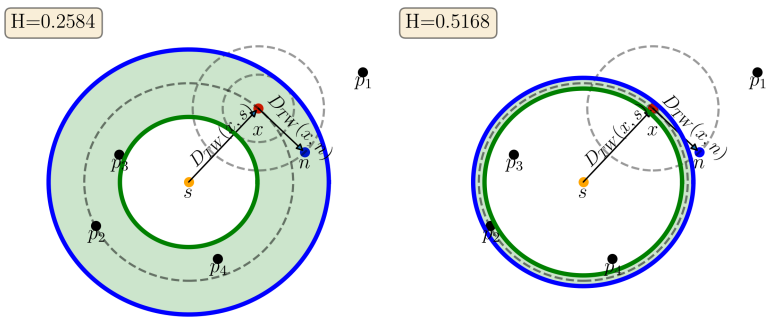


Figure 2.23: Effect of H on the elimination criteria and accuracy of the classification.

In traditional classification, it would be required to compute the distance between P and x. This would be equal to six DTW computations for the example in Figure 2.21. However, only two DTW computations were made with the AESA algorithm. This demonstrates how AESA is capable of significantly reducing the number of DTW computations.

The choice of the parameter H influences computation reduction and classification accuracy. To show that, two illustrations that correspond to two H values are provided in Figure 2.23. Figure 2.23 shows how the defined area is reduced when H is increased. The decrease in the area increases the number of eliminated samples from the training set, and P and E will be equal to each other in a small number of iterations. Thus, the algorithm performs fewer DTW computations as H increases. In addition, increasing H can lead to misclassification. Larger shaded area with low H can cover the nearest sample n in the region (see Figure 2.23 (left)), while smaller one can exclude n (see Figure 2.23 (right)). This exclusion can lead to misclassification since the true n is eliminated. Therefore, a decrease is expected in classification accuracy when H becomes too large. The process for choosing H is described in more detail in Section 2.5.5.3.

2.5.5 Results

This section presents the results for the classification accuracy using the approach proposed in Section 2.5.1 and 2.5.4 as well as current state-of-the-art methods in the literature. The results presented in this section are obtained with turning cutting signals explained in Section A.1. Specifically, Section 2.5.5.1 compares the classification accuracy using the same data set for the similarity-based method to the WPT/EEMD methods [9], and the TDA-based results [10]. Section 2.5.5.2 describes how parallel computing is employed with DTW approach. Further, Section 2.5.5.3 provides the results obtained using Approximate and Eliminate Search (AESA) explained in Section 2.5.4.

2.5.5.1 Classification Results for Dynamic Time Warping (DTW)

Table 2.10 provides classification accuracies and the time needed to compute the distance between two time series. Although the same time series is used to find the time needed to compute the distance between them, cDTW is faster compared to the FastDTW, as seen from Table 2.10. As the larger r parameter is selected for FastDTW, its computational time

increases. However, lower r values do not approximate the distance between two time series well. In addition, the cDTW algorithm matched the accuracy obtained from FastDTW. Therefore, cDTW algorithm is used to obtain results in this study.

Table 2.10: Comparison of classification accuracy and the time required to compute the distance between two time series for cDTW and FastDTW packages.

Overhang Length cm (inch)	cDTW		$\text{FastDTW}_{r=21}$	
	Accuracy	Time (seconds)	Accuracy	Time (seconds)
5.08 (2)	$98.34\% \pm 1.08\%$	1.5	$99.24\% \pm 0.73\%$	51.1

Table 2.11 compares the best classification scores obtained from WPT, EEMD, and the TDA-based methods to the results from DTW. Results for $K = 1, 2, \ldots, 5$ for the KNN classifier are obtained, and the best accuracies are obtained for DTW in Table 2.11. The cells highlighted in green are the ones with the highest overall classification score, while those highlighted in blue represent results with error bands that overlap with the best overall accuracy in the same row. A full list of the average classification scores and the corresponding standard deviations can be found in Tables B.6 and B.7.

Table 2.11: Comparison of results for similarity-based methods with their counterparts available in the literature.

Overhang Length cm (inch)	Similarity Measure		Topological Data Analysis (TDA)			Signal Processing	
	DTW*	DTW	Template Functions	Carlsson Coordinates	Persistence Images	WPT	EEMD
5.08 (2)	94.5%	98.3%	91.5%	93.6%	96.4%	93.9%	84.2%
6.35 (2.5)	86.9%	72.3%	89.3%	86.3%	85.8%	100.0%	78.6%
8.89 (3.5)	92.9%	93.0%	83.9%	95.7%	93.0%	84.0%	90.7%
11.43 (4.5)	70.9%	75.7%	65.1%	72.2%	72.5%	87.5%	79.1%

*Intermediate chatter cases are excluded.

Table 2.11 shows that features based on WPT, Carlsson Coordinates, which is a TDA-based method, and DTW give the highest accuracy for the different overhang cases. For the 5.08 and 8.89 cm (2 and 3.5 inch) overhang cases, DTW and Carlsson Coordinates have the highest classification accuracies of 98.3% and 95.7%, respectively. However, the DTW is in the error band of the highest accuracy for the 8.89 cm (3.5 inch) case. On the other hand, feature extraction with WPT and RFE is the most accurate for the 6.35 and 11.43 cm (2.5 and 4.5 inch) overhang cases scoring 100% and 87.5%. While the results from other methods are not the highest for any of the considered cases, some of them still lie within the error bars for the 11.43 cm (4.5 inch) overhang cases. Specifically, EEMD is within one standard deviation of the best results for the 11.43 cm (4.5 inch) case.

Note that the results provided in Table 2.11 are for two-class classification: chatter and chatter-free. In the column named DTW*, the results excluding the intermediate chatter cases are presented. In contrast, the DTW column shows the results when intermediate chatter cases are assumed as chatter in classification. The reason why the intermediate chatter cases are excluded in DTW is revealed in the heat maps of the similarity matrices. The heatmaps of the average DTW distances between the three classes (chatter, intermediate chatter, and no-chatter/stable) are given in Figures 2.24, 2.25, B.1, B.2. Each of the nine regions in these figures shows the average DTW distance between all the cases marked according to the row and the column labels in that region, e.g., the top right region reports the average DTW distance between the time series tagged as no-chatter versus those tagged as chatter. These heatmaps show us how the time series belonging to different classes are similar to each other. Ideally, the average distance between time series with the same label is expected to be small compared to the average distance between time series with different labels. This can also be observed in Figure 2.24. The average distance between stable cases is the lowest, while the one between stable and unstable (chatter) cases is the highest. Therefore, the DTW algorithm can distinguish the signals with different labels. However, this may not hold all the time. For instance, the heat map of the 6.35 cm (2.5 inch) case in

Figure 2.25 indicates that the average distance between intermediate chatter and no-chatter time series is almost identical to the average distance among intermediate chatter time series. This explains the low accuracy when the intermediate chatter is included as a separate class in Table 2.11. In this case, the classification algorithm can classify intermediate cases as stable ones, although these cases are taken into account as chatter cases, thus reducing the resulting accuracy. When the intermediate cases are excluded completely, the classification score increases from 72.3% to 86.9% since there is a large difference between the average distances of stable and chatter cases, as seen in Figure 2.25.

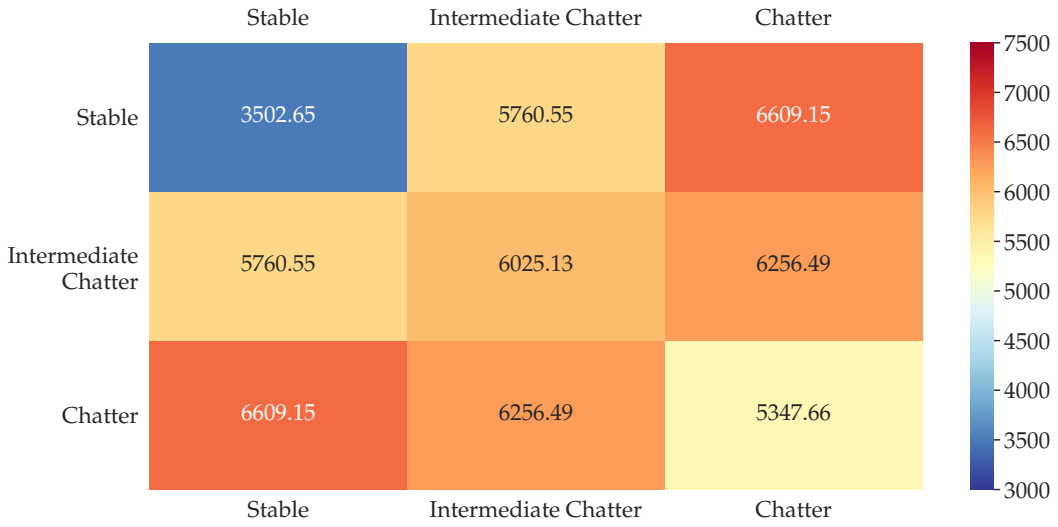


Figure 2.24: The heat map of averages DTW distances of time series belongs to three classes for the 5.08 cm (2 inch) case.

In the 11.43 cm (4.5 inch) case, Figure B.2 indicates that there is no significant difference in the average DTW distances. As a consequence, there might be errors in the classification of the chatter cases, and this leads to low overall classification accuracy, as shown in Table 2.11. When the intermediate chatter cases are not included, the classification score is 70.9%. Since the difference between the average distance of intermediate-stable and intermediate-chatter is small, the classification algorithm can still classify intermediate cases as chatter. This can increase the classification accuracy when the intermediate cases are included, as shown in Table 2.11. The score increases from 70.9% to 75.7%. For 5.08 and 8.89 cm (2 and 3.5 inch)

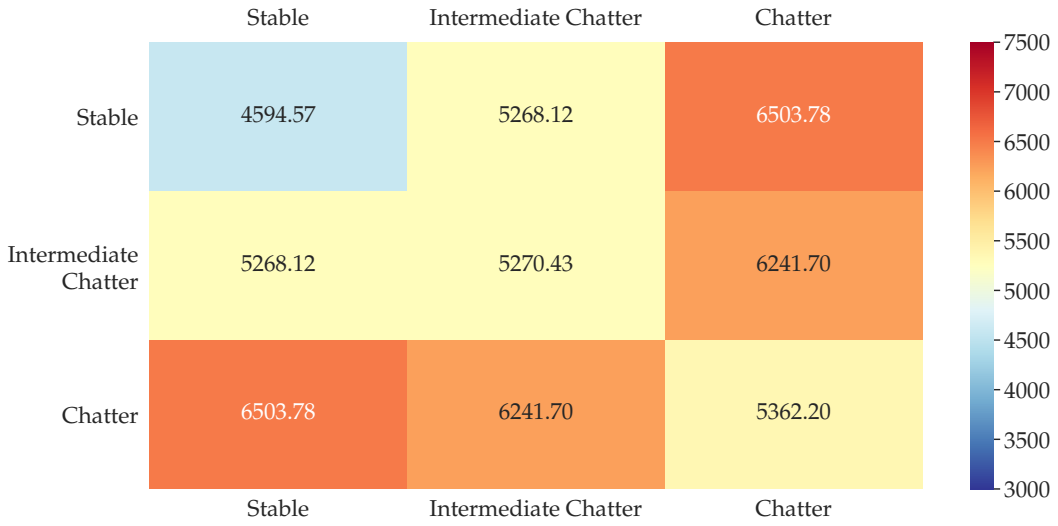


Figure 2.25: The heat map of averages DTW distances of time series belongs to three classes for the 6.35 cm (2.5 inch) case.

cases, the heatmaps (see Figure 2.24 and B.1) show clear differences between the three cases. This explains the high classification accuracies for both cases.

However, there might be interest in identifying intermediate chatter as part of a prediction algorithm that intervenes before the process develops into full chatter. Alternatively, inducing or sustaining intermediate chatter might be desirable for surface texturing applications. Figures A.4b–d show that the power spectrum for chatter and intermediate chatter is very similar, making the featurization in the frequency domain extremely challenging. However, Figure A.4a shows a clear difference between the two chatter regimes in the time domain. Therefore, it is more advantageous to extract features in the time domain for a three-class classification (chatter, intermediate chatter, and no chatter).

For 5.04 and 8.89 cm (2 and 3.5 inch) overhang cases, Figure 2.24 and B.1 show that DTW can differentiate between chatter and intermediate chatter as evidenced by the high average distance between time series tagged as chatter and intermediate chatter. The regions that list the distances between intermediate chatter and chatter cases show that the distances between chatter-chatter and chatter-intermediate chatter cases are quite different. This confirms the ability of the proposed approach to distinguish the differences between these two

cases. However, the KNN algorithm may not differentiate these cases due to the similarities between average distances in the case of 6.35 and 11.43 cm (2.5 and 4.5 inch) overhang distances. As a concrete example for the three-class classification, the distance matrices are computed for all overhang distances and fed to the KNN classification algorithm to obtain the best 3-class classification accuracy. Table 2.12 provides the best results of corresponding cases (the full classification results can be found in Table B.8). Table 2.12 shows that the DTW approach successfully distinguishes the three different classes, and the success rates are only slightly below the success rates of the two-class classification (cf. Table 2.11) for 5.04 and 8.89 cm (2 and 3.5 inch), while the low classification accuracy is observed for 6.35 and 11.43 cm (2.5 and 4.5 inch) cases as expected.

Table 2.12: The best accuracy results for three class classification with the DTW approach and the corresponding number of nearest neighbors used in the KNN algorithm.

Overhang Length cm (inch)	DTW	K-NN algorithm
5.08 (2)	$97.7\% \pm 1.1\%$	1-NN
6.35 (2.5)	$71.4\% \pm 7.0\%$	4-NN
8.89 (3.5)	$95.5\% \pm 5.4\%$	3-NN
11.43 (4.5)	$73.9\% \pm 4.6\%$	4-NN

2.5.5.2 Parallel Computing

In this section, parallel computing is utilized to expedite calculating the distance matrices. In a parallelized way, multiple distances can be computed simultaneously, which reduces the total run time. The High Performance Computing Center (HPCC) of Michigan State University (MSU) is used, and all the distance matrices are computed to produce the results shown in this section. HPCC is composed of several supercomputers, each with hundreds of nodes. Each node can be thought of as a computer with a certain number of CPUs and cores.

In parallel computing, 175, 82, 22 and 154 jobs are submitted at the same time for 5.08, 6.35,8.89, and 11.43 cm (2, 2.5, 3.5 and 4.5 inch) cases, respectively. For the 5.08 cm (2

inch) case, the number of distance computations made in each job is 1000, while it is 100 for other cases. One to five nodes, 5 CPU per job, and 2GB of RAM per CPU are requested. Therefore, each job is run with 10 GB of RAM in total. It is also worth mentioning that HPCC-MSU has a job submission policy, and this policy determines the queue time for a user depending on the requested resources from HPCC-MSU. Each time a user requests a large amount of resources, the queue time for the job submitted by that user gets higher. In addition, the queue time depends on the current usage of HPCC-MSU since it is open to all university members. Although all jobs are submitted simultaneously to HPCC-MSU, their computation is not started at the same time due to queue time, which causes deviations from the ideal time, which is equal to the runtime of a single job.

Table 2.13 provides the times required to obtain classification results with DTW and its counterparts. For DTW, two different run times are reported: traditional and parallel. In traditional DTW, only one distance computed is performed at a time. Therefore, a significant difference is observed between parallel and traditional computations in spite of the fact that the times reported in Table 2.13 for DTW (Parallel) include the queue time. The times reported for traditional computation are estimated based on the time required to complete one distance computation on Dell Optilex 7050 desktop with Intel Core i7-7700 CPU and 16.0 GB RAM. In addition, the times reported for TDA-based feature extraction methods are obtained by applying parallel computing only on persistence diagram computations, while parallel computing is not involved in WPT and EEMD. Even though WPT and EEMD are the fastest methods, parallel computing with DTW significantly reduces the total run time, making the latter the third-fastest method.

It is pointed out that the classification based on WPT is optimized, whereas the classification based on DTW is far from optimal. Therefore, one of the aims of this chapter is the introduction of DTW for chatter detection and the presentation of its general performance. In fact, this includes the necessity for an optimization of the algorithm, as seen from Table 2.13 even if parallel computing is used. However, the values reported in Table 2.13

Table 2.13: Time (seconds) comparison between similarity measure method and its counterparts.

Overhang Length cm (inch)	Similarity Measure		Topological Data Analysis (TDA)			Signal Processing	
	DTW (Traditional)	DTW (Parallel)	Template Functions	Carlsson Coordinates	Persistence Images	WPT	EEMD
5.08 (2)	227500*	7263	9495	9424	9454	116	309
6.35 (2.5)	10984*	3192	3523	3452	3482	37	83
8.89 (3.5)	2896*	1152	2148	2077	2107	5	47
11.43 (4.5)	20020*	1932	4894	4823	4853	7	68

*These run times are rough estimations.

for WPT and EEMD do not include the time required for choosing the informative wavelet packets or IMFs using manual preprocessing since the speed of manual preprocessing depends on the skill of the user and is more difficult to track. It is expected that the overall time will be much higher for WPT and EEMD after including the manual processing time. First, the computing time can be decreased by decreasing the length and/or the number of time series. At the moment, it is not clear how such a reduction affects the performance of the method. For example, the success rates for the 8.89 cm (3.5 inch) case and the 5.08 cm (2 inch) case are comparable even though the available training data is much smaller for the former. Second, the upper diagonal of the distance matrix, including pairwise distances between the training data, is computed. Although Table 2.13 shows that DTW clocks the second-fastest runtime, it is noted that this slowdown is mostly related to the training phase because of the large number of necessary pairwise distance computations during the training/testing phase. However, once the classifier is obtained, the necessary runtime for DTW will be significantly reduced because any new data is classified upon computing its pairwise distance with the training set, i.e., the only needed computation is equivalent to the evaluation of one row of the training/testing similarity matrix (see Table2.14). Finally, it is possible that the code

for calculating the DTW distance matrix can be further optimized. Many researchers have published on DTW optimization, especially for data mining ([137]), including a speedup of runtime for distance matrix computations between time series and its query. The resulting algorithm allows performing fast queries on a single core machine in a very short time, thus allowing using small consumer electronics to handle the data and possibly extract features from it in real-time. However, the AESA algorithm explained in Section 2.5.4 and given Algorithm 2.1 is applied to reduce the number of distance computations and the time required for testing.

2.5.5.3 Approximate and Eliminate Search Algorithm Results

The run times for DTW methods with parallel computing given in Table 2.13 are for training a classifier. When a new test sample is introduced, distances between the test sample and all the training set samples need to be computed to identify the nearest neighbors. However, computing these distances the traditional way can be too lengthy, especially when considering DTW for online chatter detection applications. Therefore, the AESA algorithm is employed to reduce the number of DTW computations for a test sample during classification.

The first step is to check if there is any violation of the loose triangular inequality given in Equation (2.13). The looseness constants for all combinations of three different time series are computed for turning cutting data set, and plots are provided in Fig. 2.26.

This figure shows that all combinations of three time series comply with the triangular inequality since the frequencies are accumulated on positive looseness values. Then, a range of looseness constant(H) is chosen between 0 and 10^5 with an increment of 100. Therefore, 101 different H values are used as input to the AESA algorithm. The data set of each overhang distance is split into training (67%), and test (33%) sets. For every H, the number of DTW distance computations made per test sample, and the predicted labels of test samples are obtained as output. Then, these predicted labels are compared with the true labels of the time series to determine the accuracy level, and the average number of distance computations

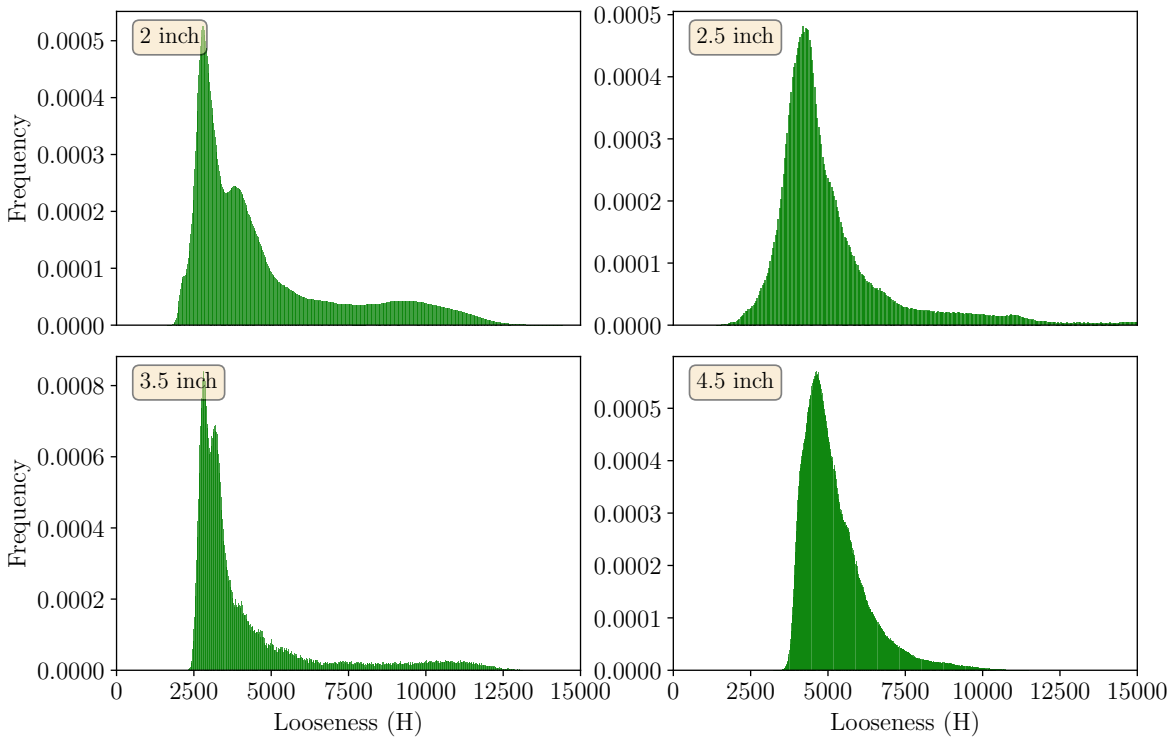


Figure 2.26: Histograms for looseness constant of all combinations of three different time series for all overhang distance.

made for the samples in the test set is computed. A plot is provided to show how the average number of distance computations per test sample and classification accuracy change with varying H in Figure 2.27. All of the results presented in Figure 2.27 are obtained with HPCC-MSU and using 1NN implementation in the AESA. However, AESA can be modified to perform classification with a larger number of nearest neighbors.

Figure 2.27 indicates that there is a decrease in the average number of distance computations and accuracy as the looseness constant increases. However, the average number of distance computations decreases dramatically. Therefore, AESA algorithms can reduce the number of DTW distance computations while keeping the accuracy as large as possible. One can find a value of H with low distance computation and high accuracy value. Furthermore, an increase in accuracy with increasing H is observed for 6.35, 8.89, and 11.43 cm (2.5, 3.5, and 4.5 inch) overhang distances. However, 6.35 and 8.89 cm (2.5 and 3.5 inch) have fewer samples in comparison to the other cases (see Table A.1). The fewer data samples in an

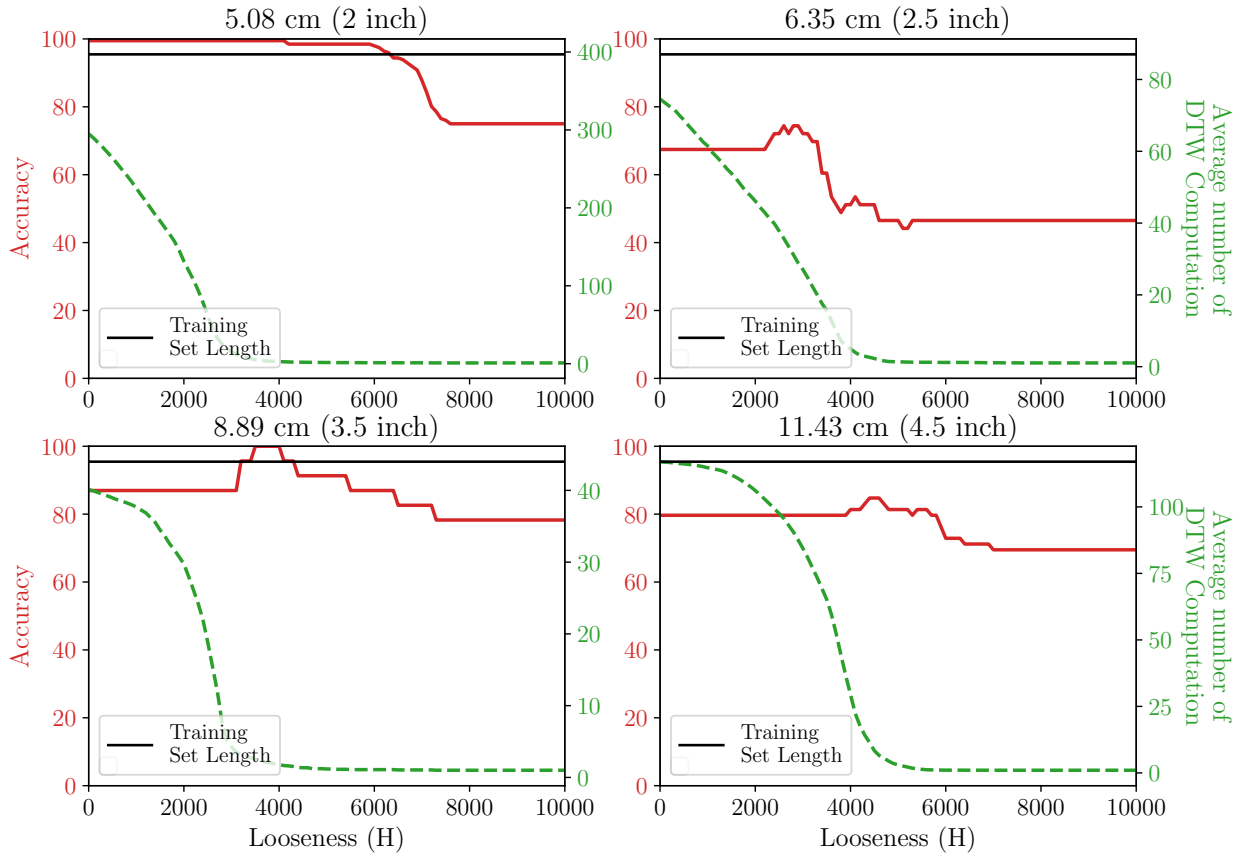


Figure 2.27: Classification accuracy (%) (solid red line) and the average number of DTW computations (green dashed line) for varying looseness constant (H).

overhang distance case cause bigger jumps in accuracy, as shown in Figure 2.27. In addition, the plots shown in Figure 2.27 are only for one train-test split. One may obtain a smoother decrease in accuracy plots by applying train-test split several times and taking the average. Then, the accuracy values for small H will converge to the values which are obtained with 10 times train-test split and provided in Table 2.11.

Some H values are chosen, and their corresponding classification score and the average number of distance computations are found for all overhang distance cases from Figure 2.27. The time required to classify one test sample is estimated for the traditional and parallelized way, and a comparison between traditional computing, parallel computing, and AESA algorithm is provided in Table 2.14. Accuracy reported in Table 2.14 is the corresponding accuracies shown in Figure 2.27 and Table 2.11.

Table 2.14: Classification time of one test sample for traditional way, parallel computing, and AESA algorithm and corresponding average accuracy obtained from Table 2.11 and Figure 2.27.

	Traditional		AESA				Parallel	
Overhang Distances cm (inch)	Acc.	Time(s)	H	Acc.	Avg. count	Time(s)	Acc.	Time(s)
5.08 (2)	98.3	595.5*	6900	90.81	1.13	1.70	98.3	≈ 1.5
6.35 (2.5)	72.3	130.5*	3300	74.42	20.04	30.06	72.3	≈ 1.5
8.89 (3.5)	92.9	66*	6300	86.96	1.09	1.64	92.9	≈ 1.5
11.43 (4.5)	75.7	175.5*	5600	81.36	1.19	1.79	75.7	≈ 1.5

*These run times are rough estimates.

The H values chosen in Table 2.14 is an example. One can choose different values for H as well. There is a trade-off between accuracy and the average number of distance computations. Higher H values can be selected to have a small number of distance computations, but this comes with a lower classification score. In the traditional way, the reported times are estimated based on the time required to complete one distance computation. On the other hand, parallel computing could be the fastest method among them. Ideally, if all distance computations between the test set sample and the training set are sent to HPCC-MSU in separate jobs, each job will take nearly 1.5 seconds to complete. However, there might be some queue time which leads to a delay in time to obtain results. One can also use the available workstations to perform parallel computing without needing the extravagant supercomputers that HPCC-MSU has. Using parallel computing is optional and a viable option if a high-performance computing cluster is available. Alternatively, the user can speed up the computations by implementing the AESA algorithm on a workstation. Moreover, AESA algorithm provide promising classification times as seen from Table 2.14. Classification can be completed in less than two seconds with the chosen H values for all overhang distances except the 6.35 cm (2.5 inch). One can choose another value of H for the 6.35 cm (2.5 inch) case to obtain results faster, but this corresponds to a lower classification score. Table 2.14 shows us that AESA and parallel computing can enable in-process chatter detection on the

cutting centers with the similarity measure approach using a classifier that is trained first offline and then loaded to a controller attached to the manufacturing center. Moreover, the implementation of AESA in an online manufacturing application is cheaper than buying supercomputers or workstations to perform parallel computing.

The DTW-based approach operates directly on the time series, thus bypassing the preprocessing step involved in the WPT and EEMD methods. Further, in contrast to deep learning techniques, such as neural networks, using DTW does not necessitate a large number of datasets for training. The feasibility of implementing DTW in a real cutting center is examined by investigating its transfer learning capabilities. Providing these results with smaller deviation and eliminating the manual preprocessing are significant advantages for the DTW approach that still enables it to achieve high classification accuracies even if the system parameters (in this case, the eigenfrequencies) shift during the process.

2.6 Topological Data Analysis (TDA) Based Approach

2.6.1 Topological Data Analysis

Topological Data Analysis (TDA) extracts information by investigating the shape of the data. In this study, persistent homology, which is a powerful tool of Topological Data Analysis (TDA), is proposed to extract features from the persistence diagrams and use them in supervised machine learning algorithms. Experimental data is embedded using Takens embedding theorem [138], and 1-D persistent homology is investigated for feature matrix generation. In this section, persistent homology is briefly explained, and one can refer to [62, 63, 64, 65, 139, 140] for detailed information about TDA and persistent homology.

Persistent homology provides a compact tool for studying the topology of data embedded in a Euclidean space which is often called a point cloud. The resulting shape information is represented by a two dimensional plot called the persistence diagram. A persistence diagram can be obtained for different shape characteristics of interest. For instance, if the main interest is in the connectivity of the points in the point cloud, then that information

can be represented in a 0-dimensional (0-D) persistence diagram. Alternatively, the 1-D persistence diagram is computed to investigate the loops in the point cloud. For voids, 2-D persistence diagrams are computed, and so on. This study only focuses on extracting features from the 0-D and 1-D persistence, so in the following, the basic idea for obtaining the 1-D persistence will be explained, i.e., for representing loops that emerge and disappear as the point cloud is thickened. The process for obtaining the 0-D persistence is similar, only instead of considering loops, the connectivity of the points would need to be tracked as the point cloud is uniformly thickened.

Consider the point cloud shown in Figure 2.29a. Then, the point cloud starts to thicken, i.e., expand disks with radius ϵ around each data point. As ϵ is increased, disks can start to intersect. The intersection of two disks forms an *edge* as shown by the two edges in Figure 2.29b. Increasing ϵ further can lead to three disks intersecting, thus forming a triangle which is filled in as shown by the nine triangles in Figure 2.29c. At some values of ϵ, some disk intersections will lead to *cycles*. The time (here, the ϵ value) at which a cycle appears is called the birth time of the cycle. Figure 2.29d shows three example cycles numbered 1,2 and 3 with birth time $b_1 = b_2 = b_3$.

As the disks continue thickening in the point cloud, more disks will intersect, leading to more triangles filling in, and at some point, some cycles may fill in. The time at which a cycle disappears is called its death time. For example, Figs. 2.29e–g show the death times of cycles 1–3, respectively. The information about the birth and death of cycles is succinctly summarized in a persistence diagram. In this diagram, each point corresponds to the paired birth and death times of a cycle. For example, Figure 2.29h shows the tuples (b_1, d_1), (b_2, d_2), and (b_3, d_3) corresponding to the birth and death times of the cycles 1–3 shown in Figure 2.29d. The cycle that persists the longest is characterized by the highest point above the diagonal and is called maximum persistence. In this example, cycle 3 leads to the maximum persistence in the persistence diagram.

2.6.1.1 Simplicial complexes

Let $\{u_0, \ldots, u_k\} \in \mathbb{R}^d$ be a set of data points, and the vectors defined between these data points (u_1-u_0,u_2-u_0,...,u_k-u_0) are linearly independent. A geometric k-simplex, σ is a set of all points in $\mathbb{R}^d$ such that $\sum_{j=0}^{k} \lambda_j u_j$ where $\sum_{j=0}^{n} \lambda_j = 1$ and $\lambda_j \geq 0$ for all j. Figure 2.28 provides illustrations for $0, 1$, and $2-$simplex. Each data point on a point cloud is represented as 0-dimensional simplex and they are called vertices. When two vertices are connected, an edge is formed and it is 1-dimensional simplex. Connection of three vertices will form $2-$simplex which is a triangle. Simplicies spanned by any subset of $u_0, \ldots, u_k$ are the faces of σ. In general, $n-$simplex contains $n+1$ vertices, and the set of these simplices, are called geometric simplicial complexes, K, if the following two conditions are satisfied [139]: 1) If $\sigma \in K$, then faces of σ are also in K, 2) If two $n-$simplex, σ_1 and σ_2 are in K, then the intersection of them is either common face or empty. The dimension of the simplicial complex, K, is equal to the largest dimension of its simplices.

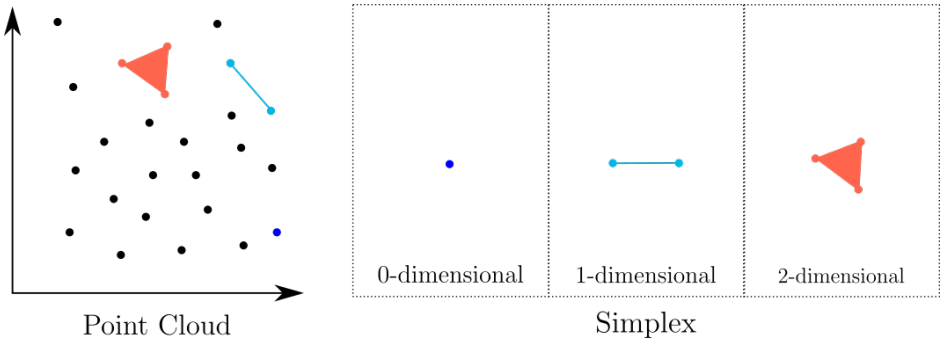


Figure 2.28: Formation of simplicial complexes from a point cloud.

2.6.1.2 Persistent Homology

The simplicial complex K is used to compute homology $H_n(K)$ in a different dimension to identify the shape of the data. 0 dimensional homology, $H_0(K)$ represents connected components and one dimensional homology, $H_1(K)$ represents loops, while two dimensional homology $H_2(K)$ represents voids. The simplicial complex K is not fixed in persistent homology, and it varies over time.

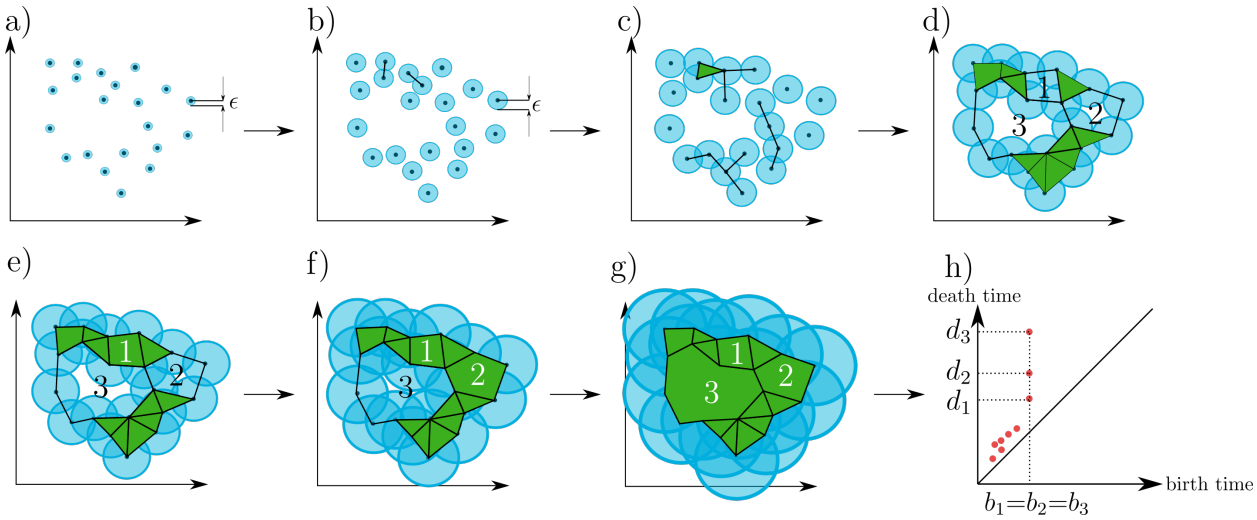


Figure 2.29: Generation of persistence diagrams using The Rips Complex.

Disks centered at data points of the point cloud start to expand and let ϵ be the radius of these disks. As ϵ increases, n-dimensional simplicies are formed, as shown in Figure 2.29. The intersection of two disks forms an edge (1-simplex), while a triangle (2-simplex) is formed when three disks intersect with each other (see Figure 2.28). Each ϵ will result in different simplicial complexes, and they can be approximated using filtration functions. This study uses a Python package that employs the Rips complex. The definition of Rips complex is given as

$$R_\epsilon(K, d) = \{\sigma \subset K | \max_{(x,y)\in\sigma} d(x, y) < \epsilon\}, \tag{2.17}$$

where d is the distance between vertices of simplicial complex σ. Let $\{\epsilon_1 < \epsilon_2 < \ldots < \epsilon_m\}$ be the set of varying radius of the disk. These radii form Rips complexes such that

$$R_1 \subseteq R_2 \subseteq \ldots \subseteq R_m \tag{2.18}$$

where $R_j = R_{\epsilon_j}(K, d)$. Then, a specific dimension n can be chosen to identify the shape of the data along the simplicial complexes R_j. For instance, if a loop is seen first in R_i, this is called birth time ($b = \epsilon_i$). When it disappears in R_j, this is denoted as death time ($d = \epsilon_j$, where $d > b$). That allows one to generate persistence diagrams, D for a given point cloud, and selected persistent homology dimension. Figure 2.29h provides an example of a persistence diagram. The horizontal axis represents the birth time, while the vertical axis

is for the death time, and all the points in the diagram are above the diagonal. The points with larger lifetime $(d-b)$ are farther away from the diagonal, and these points often include important information about the overall shape and structure of the data. While it is possible to generate a persistence diagram for each homology dimension, one dimensional persistent homology is mostly focused on feature extraction since it can capture circular structure in the data, which is often a characteristic of chatter in turning.

2.6.2 Method

The method proposed for chatter detection using topological features can be summarized using Figure 2.30. The parts of the pipeline related to data collection, processing, and labeling were described in Sections A.1–A.1.1, respectively. In this section, the rest of the steps shown in Figure 2.30 is described.

Recall that the cutting tests are composed of four different overhang configurations. 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 ([141]).

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 2.6.3.1) and for obtaining the persistence diagrams (see Section 2.6.1.2). Upon finding the appropriate embedding parameters, the data is embedded using delay reconstruction, also known as Takens' delay embedding, see Section 2.6.3.1. The resulting point cloud is then used to compute the corresponding persistence diagrams using two different approaches: 1)traditional way where persistence diagrams are computed with Ripser package for Python, 2) approximating to point cloud with Bezier curves and computing persistence diagrams using line segments generated with Bézier curves ([77]). The second method is a different approach introduced in [77], and it uses Bézier curves to approximate to reduce the time to compute persistence

diagrams. Section 2.6.3 explains both approaches for persistence diagram computation, and sections 2.6.4.1–2.6.4.5 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. 1-dimensional persistent homology H_1 is mainly utilized for feature extraction except when the template function approach is used. For template functions, 0-dimensional persistence H_0 is also used ([72]).

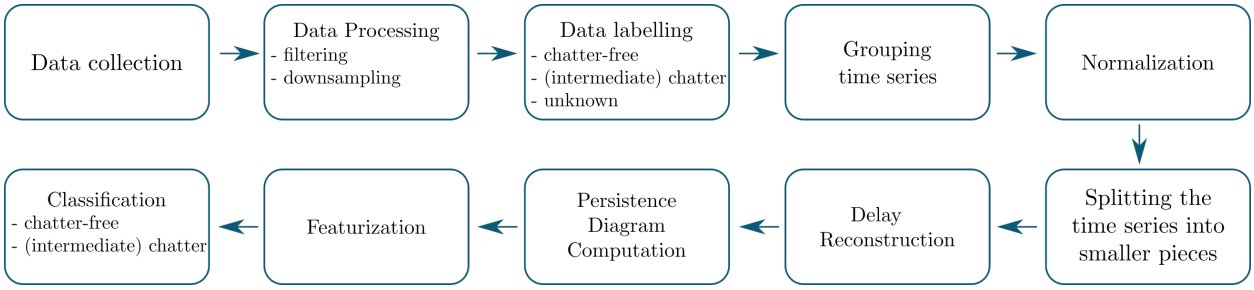


Figure 2.30: Pipeline for feature extraction using topological features of data.

2.6.3 Persistence Diagram Computation

In this section, the embedding of time series to a higher dimension and the persistence diagram computation using the Bézier curve approximation method are explained briefly. The steps for persistence diagram computation are summarized as shown in Figure 2.31.

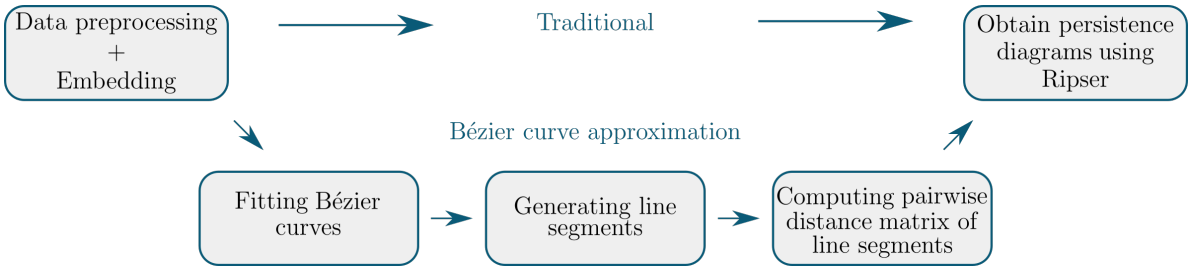


Figure 2.31: Persistence diagram computation steps.

2.6.3.1 Delay Reconstruction

Takens' theorem lays a theoretical framework for studying deterministic dynamical systems ([138]). It states that, in general, embedding of the attractor of a deterministic dynam-

ical system can be obtained 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 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 the flow of the system is not available for a 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

$$\begin{aligned} \Phi_{\phi,\beta} &= (\beta(\mathbf{x}), \beta(\phi(\mathbf{x})), \ldots, \beta(\phi^{d-1}(\mathbf{x}))) \\ &= (\beta(\mathbf{x}_t), \beta(\mathbf{x}_{t+\tau}, \beta(\mathbf{x}_{t+2\tau}, \ldots, \beta(\mathbf{x}_{t+(d-1)\tau}))), \end{aligned} \tag{2.19}$$

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 study, τ was found by using the method of Least Median of Squares (LMS) ([142]) combined with the magnitude of the Fast Fourier Transform (FFT) of the signal. Specifically, the FFT spectrum is obtained, and the maximum significant frequency is identified in the signal using LMS [143]. Then Nyquist's sampling criterion is used to choose the delay value according to the inequality described in [144]. This approach yielded reasonable delay values in comparison to the standard mutual information function approach ([145]), 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 ([146, 147]). 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. 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) makes it 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 $d \in \{1, 2, \ldots, 10\}$, 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 2.6.1. Five different methods are studied to extract features from the resulting persistence diagrams as shown in Sections 2.6.4.1–2.6.4.5.

2.6.3.2 Persistence Diagram Computation with Bézier Curve Approximation

Bézier curves were introduced by Pierre Bézier, and they have been widely used in computer aided design software and path planning applications for robots ([148, 149, 150, 151]). Recently, [77] utilized Bézier curves to speed up the persistence diagram computation. In this approach, the first step is to divide the point cloud into groups. The user defines the number of samples per group (*spg*), then a Bézier curve is individually fitted to each group. Line segments are then generated using these curves according to the number of line segments per curve r selected by the user. Pairwise distance matrix between the line segments is computed, and it is given to *Ripser* as input, and it will provide the persistence diagrams as output based on the selected maximum number of homology dimensions. This section explains two of three main steps, which are fitting a Bézier curve and computing the distance matrix between the line segments, and the effect of *spg* and r on the approximation of persistence diagrams.

Fitting a Bezier Curve: In this implementation, the cubic Bézier curve is used, and its expression is defined as

$$p(t) = \sum_{i=0}^{3} (1-t)^{3-i} t^i p_i \quad 0 < t < 1, \tag{2.20}$$

where t is the parametrization variable and p_i is called the control points of the curve. For cubic Bézier curves, there are four different control points. The first one and the last one should match with the first and last point of the group of samples where the Bézier curve is fit, respectively. Solution for these control points is obtained by using the least squares error method, and the expression is given as

$$L(p_0, p_1, p_2, p_3) = \sum_{i=1}^{l} ||p(t_i) - x_i||^2, \tag{2.21}$$

where x_i is the data points of the point cloud. Solution of $\nabla L = 0$ provides the control points. The expression for $\nabla L = 0$ can be rewritten such that

$$\sum_{k=1}^{l} 2 \left(\sum_{i=0}^{3} \binom{3}{i} (1-t_k)^{3-i} t_k^i p_i - x_k \right) \left(\frac{\partial p(t_k)}{\partial p_i} \right) = 0 \tag{2.22}$$

$$\frac{\partial p(t_k)}{\partial p_i} = (1-t_k)^3 \hat{e}_1 + 3(1-t_k)^2 t_k \hat{e}_2 + 3(1-t_k) t_k^2 \hat{e}_3 + t_k^3 \hat{e}_4 = 0, \tag{2.23}$$

where t_k represents varying parametrization variable along a Bézier curve. The above expression can also be written in matrix form as

$$\left(\sum_{k=1}^{l} A_k \right) p = \sum_{k=1}^{l} b_k \quad \text{or} \quad Ap = b. \tag{2.24}$$

The equations for A_k, p and b_k are defined as

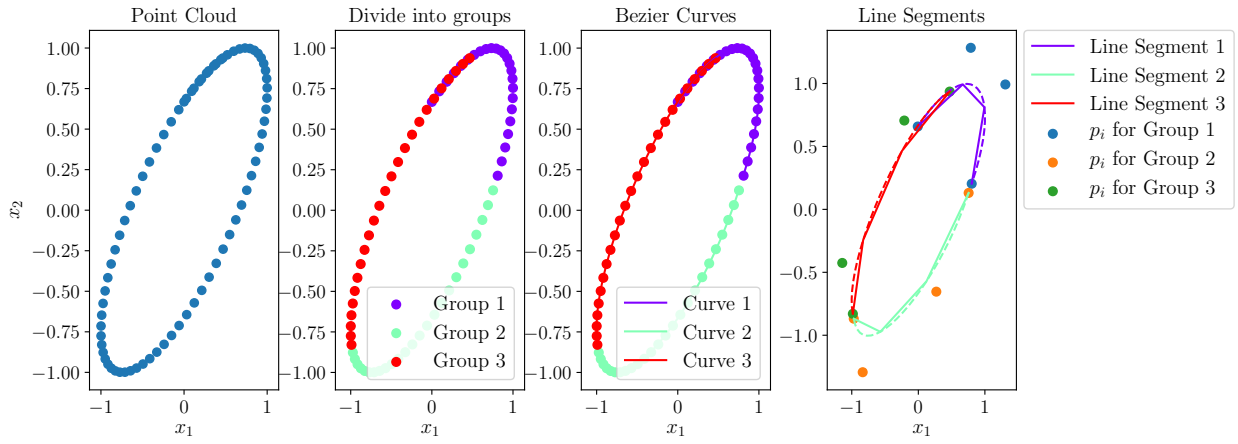


Figure 2.32: Illustration showing Bézier curve fit and generation of line segments.

$$A_k = \begin{bmatrix} (1-t_k)^6 & 3(1-t_k)^5 t_k & 3(1-t_k)^4 t_k^2 & (1-t_k)^3 t_k^3 \\ 3(1-t_k)^5 t_k & 9(1-t_k)^4 t_k^2 & 9(1-t_k)^3 t_k^3 & 3(1-t_k)^2 t_k^4 \\ 3(1-t_k)^4 t_k^2 & 9(1-t_k)^3 t_k^3 & 9(1-t_k)^2 t_k^4 & 3(1-t_k) t_k^5 \\ (1-t_k)^3 t_k^3 & 3(1-t_k)^2 t_k^4 & 3(1-t_k) t_k^5 & t_k^6 \end{bmatrix} \tag{2.25}$$

$$p = \begin{bmatrix} p_0^1 & p_0^2 & \dots & p_0^d \\ p_1^1 & p_1^2 & \dots & p_1^d \\ p_2^1 & p_2^2 & \dots & p_2^d \\ p_3^1 & p_3^2 & \dots & p_3^d \end{bmatrix}, \quad b_k = \begin{bmatrix} (1-t_k)^3 \\ 3(1-t_k)^2 t_k \\ 3(1-t_k) t_k^2 \\ t_k^3 \end{bmatrix} \begin{bmatrix} x_k^1 \\ x_k^2 \\ \vdots \\ x_k^d \end{bmatrix}^T, \tag{2.26}$$

where d is the dimension of the data set. To illustrate the Bézier curve fitting, a sinusoidal signal is embedded to dimension two. Then, the samples are divided into groups, and bezier curves are fit. Figure 2.32 represents the control points and fitted lines on the groups.

Computing pairwise distance matrix: Bézier curves are split into intervals, and the number of intervals (r) is selected by the user. The endpoints of the intervals are connected to each other, and line segments are generated. The next step is to compute the distance matrix between the line segments. Lets assume that $\overrightarrow{l_0 l_1}$ and $\overrightarrow{m_0 m_1}$ represent two lines. The distance between these two lines is defined as follows

$$d(\overrightarrow{l_0 l_1}, \overrightarrow{m_0 m_1}) = \min_{l \in \overrightarrow{l_0 l_1}, m \in \overrightarrow{m_0 m_1}} d(l, m), \tag{2.27}$$

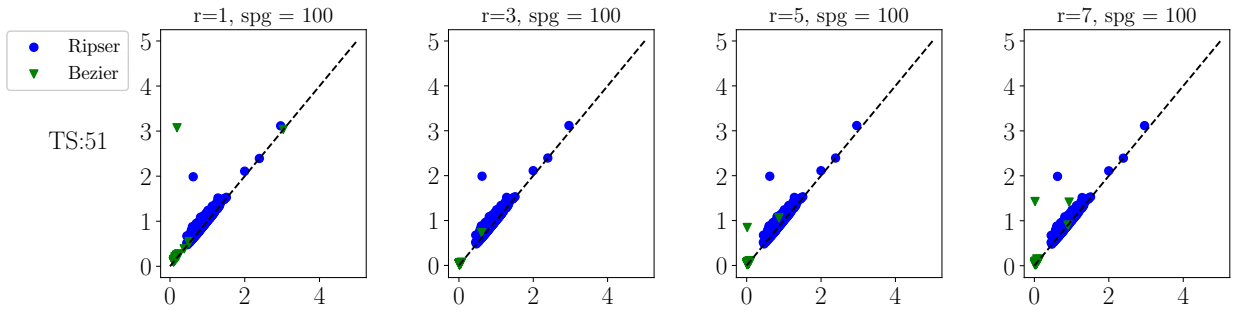


Figure 2.33: Comparison of persistence diagrams computed with the approximation for varying r to true diagrams obtained with *Ripser*. All diagrams belong to time series number 51 of the 8.89 cm (3.5 inch) case, and they are computed in H_1.

where l_0, l_1, m_0 and m_1 are the end points of the two lines. The distance is computed by minimizing the function given as

$$f(l, m) = d(l(s), m(t))^2 = ||l(s) - m(t)||^2, \tag{2.28}$$

where s and t are parametrization variables. [77] solved this problem using a gradient descent algorithm. However, simplicial homology global optimization (SHGO) is employed in this study ([152]). After finding the parameters making function f minimum, their values are used to compute the distance between two lines. This is repeated for all combinations between line segments. Only the upper diagonal of the distance matrix is computed since it is symmetric. Then, it is given to the *Ripser* package as input to obtain persistence diagrams.

Effect of parameter selection: The parameters sample per group (spg) and the number of line segments (r) define how well persistence diagrams of a time series are approximated. A time series belonging to the 8.89 cm (3.5 inch) overhang distance is chosen, and persistence diagrams of that time series are computed with $r = 1, 3, 5, 7$ and $spg = 100$. In Figure 2.33, both diagrams computed with *Ripser* in blue and approximated diagrams in green color are provided. All diagrams represent the first-dimensional persistence (H_1), and they belong 51^{th} time series in the 8.89 cm (3.5 inch) case. Figure 2.33 shows that approximated diagrams converge to the diagram obtained from *Ripser* as the number of line segments (r) increases. Each line segments have only two points, so this means that

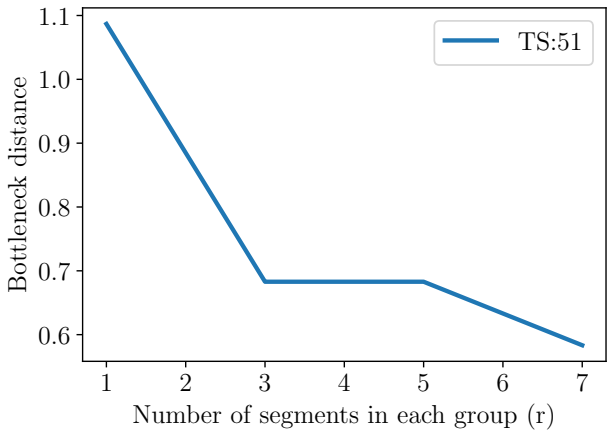


Figure 2.34: The Bottleneck distance between the diagram with blue color in Figure 2.33 and the approximated diagrams with green color in Figure 2.33.

the Bézier curve approximation uses only $2r$ points instead of using all 100 points for the examples provided in Figure 2.33. To show the effect of selection r, Bottleneck distances are computed. The definition of Bottleneck Distance is given as ([153]),

$$W_\infty(X_1, X_2) = \inf_{\eta: X_1 \to X_2} \sup_{x_1 \in X_1} ||x_1 - \eta(x_1)||_\infty, \tag{2.29}$$

where X_1 and X_2 represent two different diagrams and η is the bijections between the points of the diagrams. Figure 2.34 shows the Bottleneck distances between the diagrams, and it is seen that increasing r results in smaller distances for the 8.89 cm (3.5 inch) case, which means that larger values of r approximate to blue one better. In addition, this type of behavior can be observed when the *spg* decreases since it will lead to a larger number of groups and line segments.

2.6.4 Feature Extraction Using Persistence Diagrams

In this section, feature extraction from persistence diagrams is explained. Five different featurization techniques are described in this section, and these are persistence landscapes, persistence images, Carlsson Coordinates, the kernel for persistence diagrams, and the signature path of persistence path's signatures. The source code for these featurization techniques are available in Teaspoon package of Python [154].

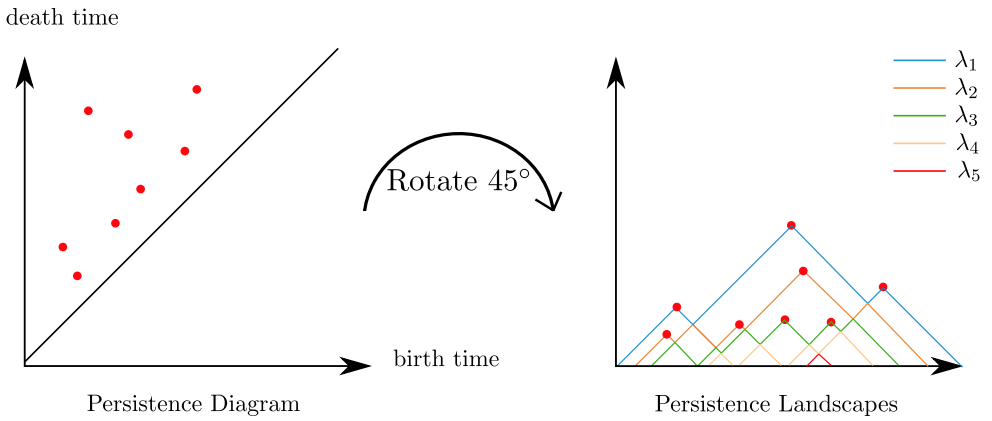


Figure 2.35: A schematic showing the process of obtaining the landscape functions from a persistence diagram.

2.6.4.1 Persistence Landscapes

Persistence landscapes are functional summaries of persistence diagrams [155]. They are obtained by rotating persistence diagrams by 45° clockwise and drawing isosceles right triangles for each point in the rotated diagram [156], see Figure 2.35 where the landscape functions are represented by λ_k. Given a persistence diagram, the piecewise linear functions are defined as [155]

$$g_{(b,d)}(x) = \begin{cases} 0 & \text{if } x \notin (b,d) \\ x-b & \text{if } x \in (b, \frac{b+d}{2}] \\ -x+d & \text{if } x \in (\frac{b+d}{2}, d) \end{cases} \tag{2.30}$$

where b and d correspond to birth and death times, respectively. Figure 2.35 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 $g_{(b_i,d_i)}(x)$ [155]. If the second topmost components of $g_{(b,d)}(x)$ are connected, the second landscape function λ_2 is obtained. The other landscape functions are obtained similarly. Note that the landscape functions are also piecewise linear functions.

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 work, a feature vector from the persistence landscapes is extracted 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. Then features from all $|\mathbb{K}|$ landscapes are combined 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, $\mathbb{K} = \{1, 2, \ldots, 5\}$ is used since it gives good results for the turning data. Similarly, the mesh may also be optimized in a similar way; however, this is a more difficult task due to the infinite domain of $\mathbf{b}$, so the mesh can be defined as follows and as shown in Figure 2.36.

Let $\lambda_{i,j}$ be the ith landscape corresponding to the jth persistence diagram from a training set in a supervised learning setting. Fix i and overlay the chosen landscape functions corresponding to all of the persistence diagrams in the training set. Figure 2.36 provides an example of this process, and it selects second landscapes to extract features. Now project all the points that define the linear pieces of each of the landscape functions onto the birth axis. The red dots in Figure 2.36 represents these projected points. Sort the projected points in ascending order and remove duplicates. The resulting set of points is a mesh $\mathbf{b}_i$ with length $|\mathbf{b}_i|$ for the ith landscape. The same process can be repeated to get the feature vector for all the $|\mathbb{K}|$ landscapes and construct the overall feature vector $\mathbf{b}$. It is emphasized 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, it is evaluated at the mesh points. Computationally, this is efficiently accomplished using piecewise linear interpolation functions.

Upon extracting the features from the persistence landscapes, a feature matrix is con-

structed, and it represents all the tagged feature vectors. For instance, Table 2.15 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 supervised machine learning algorithms, for example, to train a classifier.

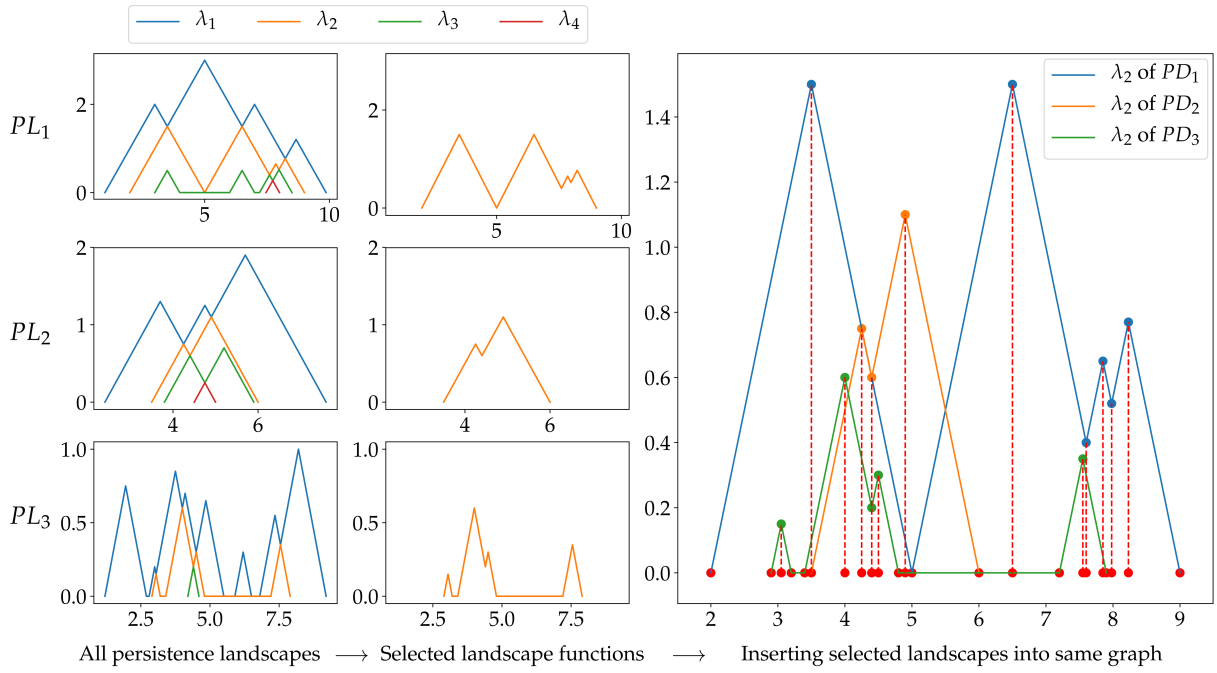


Figure 2.36: Persistence landscape feature extraction.

Table 2.15: 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.

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

For turning cutting data (see Section A.1), the persistence diagrams and the corresponding first five persistence landscapes for each overhang case are computed. Then the resulting landscapes are split into a training set (67%) and a test set (33%), and feature matrices are created for each of the first five landscapes separately. SVM with 'rbf' kernel, Logistic Regression and Random Forest and Gradient Boosting algorithms are used as a classifier. The split-train-test process is repeated 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 Tables B.9–B.11 in the appendix. In the results section, the results with the highest accuracy for each of the overhang cases are utilized from Tables B.9–B.11 when comparing the different TDA-based featurization methods.

2.6.4.2 Persistence Images

Persistence images are another functional summary of persistence diagrams [74, 156]. 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

$$T(b_i, d_i) = (b_i, d_i - b_i) = (b_i, p_i), \tag{2.31}$$

which transforms the persistence diagram from the birth-death coordinates to birth-lifetime coordinates (see Figure 2.37a-b). 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 (see Figure 2.37c)

$$D_k(x, y) = \frac{1}{2\pi\sigma^2} e^{-[(x-b_k)^2+(y-p_k)^2]/2\sigma^2}. \tag{2.32}$$

It was shown in [157] that the persistence images method is not very sensitive to σ, which is set to 0.1 in this study. A weighting function is also defined for the points in the

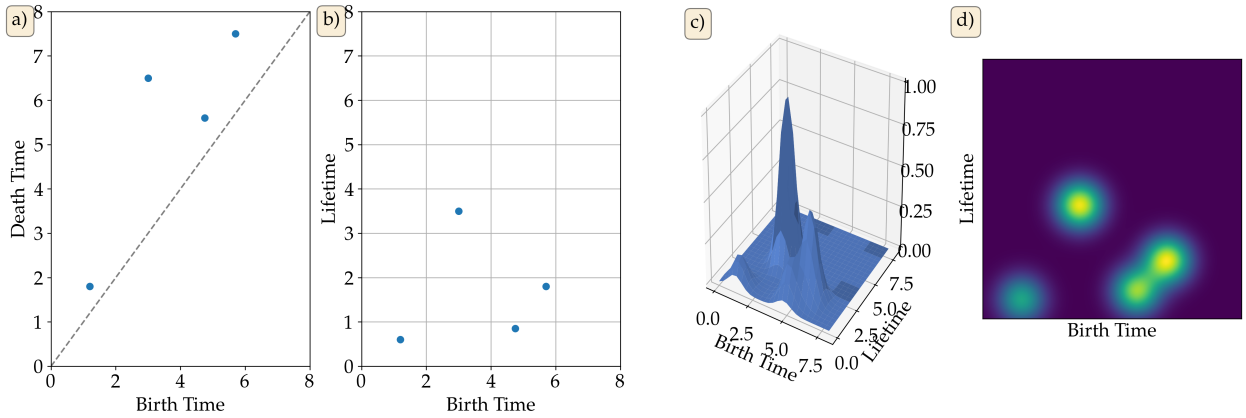


Figure 2.37: Steps for persistence image computation.

persistence diagram $W(k) = W(b_k, p_k) : (b_k, p_k) \in T(X) \to \mathbb{R}$ according to

$$W(k) = W(b_k, p_k) = \begin{cases} 0 & \text{if } p_k \leq 0; \\ \frac{p_k}{b} & \text{if } 0 < p_k < b; \\ 1 & \text{if } p_k \geq b. \end{cases} \tag{2.33}$$

Note that this is not the only possible weighting function, but it satisfies the requirements needed to guarantee the stability of persistence images [74]: it vanishes along the horizontal axis, is continuous, and is piecewise differentiable. Now define the integrable persistence surface

$$S(x, y) = \sum_{k \in T(X)} W(k)\, D_k(x, y). \tag{2.34}$$

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

$$I_{i,j}(S) = \iint S\, dxdy, \tag{2.35}$$

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 (see Figure 2.37d).

Featurization of persistence images: Persistence images can be used for support vector machine classification [158]. 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

$$\begin{bmatrix} I_{1,1} & I_{1,2} & \dots & I_{1,100} \\ \vdots & \ddots & & \\ I_{100,1} & & & I_{100,100} \end{bmatrix}, \tag{2.36}$$

and a typical feature vector is obtained by concatenating the entries of this matrix row-wise as shown by the rows in Table 2.16. 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 $I^k_{i,j}$, where $k \in \{1, 2, \dots, n\}$ is the persistence diagram index while i, j are row and column numbers, respectively, in the image.

Table 2.16: Feature matrix for persistence images.

Persistence Diagrams	Label	Persistence Image									
X_1	1	$I^1_{1,1}$	$\dots$	$I^1_{1,100}$	$I^1_{2,1}$	$\dots$	$I^1_{2,100}$	$\dots$	$I^1_{100,1}$	$\dots$	$I^1_{100,100}$
X_2	0	$I^2_{1,1}$	$\dots$	$I^2_{1,100}$	$I^2_{2,1}$	$\dots$	$I^2_{2,100}$	$\dots$	$I^2_{100,1}$	$\dots$	$I^2_{100,100}$
$\vdots$	$\vdots$	$\ddots$									
X_n	1	$I^p_{1,1}$	$\dots$	$I^p_{1,100}$	$I^p_{2,1}$	$\dots$	$I^p_{2,100}$	$\dots$	$I^p_{100,1}$	$\dots$	$I^p_{100,100}$

Python's PersistenceImages package is used to featurize the cutting signals, and then the resulting images are randomly split into 67%-33% train-test sets. The persistence images have boundaries depending on lifetime and birth time ranges. Therefore, the maximum lifetime and maximum birth time are found by checking all diagrams of a data set. These maximum values can correspond to a point with a significant lifetime, and significant features can be lost if the boundaries of the image are set to those values exactly. Accordingly, each value is summed with 1 to be able to capture all important features nearby them. A classifier

is trained using SVM and the 'rbf' kernel, Logistic Regression, Random Forest, and Gradient Boosting classifiers for two different pixel sizes: 0.05 and 0.1. The training and testing results for each different overhang case are available in Tables B.12–B.13 of the appendix. When the classification accuracy is compared for persistence images to the other featurization methods, the best results are chosen from these tables for each cutting configuration.

2.6.4.3 Carlsson Coordinates

Another method for featurizing persistence diagrams is Carlsson's four Coordinates [71] with the addition of the maximum persistence [69], 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

$$\begin{aligned} f_1(X) &= \sum b_i(d_i - b_i), \\ f_2(X) &= \sum (d_{\max} - d_i)(d_i - b_i), \\ f_3(X) &= \sum b_i^2(d_i - b_i)^4, \\ f_4(X) &= \sum (d_{\max} - d_i)^2(d_i - b_i)^4, \\ f_5(X) &= \max\{(d_i - b_i)\} \end{aligned} \tag{2.37}$$

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, the persistence diagrams are computed from the embedded accelerometer signals and randomly split the data into training (67%), and testing (33%) sets. Then all five coordinates are calculated for each diagram, and SVM, Logistic

Regression, Random Forest, and Gradient Boosting are utilized to train a classifier. The feature vectors tested in this study 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 Table B.14 in the appendix. However, in the results section, the feature vectors that yielded the highest accuracy are utilized when the classification results of Carlsson coordinates are compared to the other featurization methods.

2.6.4.4 Kernels for Persistence Diagrams

In addition to featurization methods, many kernel methods have also been developed for machine learning on persistence diagrams [75, 159, 160, 161, 162, 163, 164]. As an example, the kernel introduced by [75] is chosen, and it is defined for two persistence diagrams X and Y according to

$$\kappa_\sigma(X,Y) = \frac{1}{8\pi\sigma} \sum_{z_1 \epsilon X, z_2 \epsilon Y} \exp\left(-\frac{||z_1 - z_2||^2}{8\sigma}\right) - \exp\left(-\frac{||z_1 - \hat{z}_2||^2}{8\sigma}\right), \tag{2.38}$$

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, two values are investigated 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 Equation (2.38), the kernel matrix can be defined as

$$\kappa_\sigma = \begin{bmatrix} \kappa_\sigma(X_1, X_1) & \kappa_\sigma(X_1, X_2) & \dots & \kappa_\sigma(X_1, X_N) \\ \vdots & \ddots & & \\ \kappa_\sigma(X_N, X_1) & & & \kappa_\sigma(X_N, X_N) \end{bmatrix}. \tag{2.39}$$

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 $\mathcal{O}(|X| \cdot |Y|)$ time [75]. Therefore, the computation time for kernel methods is generally high, and this can complicate optimizing the tuning parameter σ. To emphasize the effect of the computational

complexity, in this study, the long runtime for the 5.08 cm (2 inch) overhang case caused by its large number of samples has led to reporting the corresponding classification results for a smaller number of iterations than the other overhang cases and the other featurization approaches.

For turning data (see Section A.1), a 67%/33% train/test split is performed for the labeled persistence diagrams. For each of the training and testing sets, the corresponding kernel matrices are precomputed, and Python's LibSVM [165] is used for classification. For almost all but the 5.08 cm (2 inch) overhang case where only 1 iteration was used, the split-train-test process is repeated 10 times. The average and the standard deviation of the resulting accuracies are recorded. The resulting classification accuracies are reported in Table 2.23. Note that [75] describes another approach for training a classifier based on measuring the distances between two kernels in combination with a k-Nearest Neighbor (k-NN) algorithm. However, this alternative method is not explored in this work, and only the computations using the kernel matrix and the LibSVM library are performed.

2.6.4.5 Persistence Paths' Signatures

Persistence paths' signatures are a recent addition to featurization tools for persistence diagrams [76]. Let $\gamma : [a, b] \to \mathbb{R}^d$ be the piecewise differentiable path given by

$$\gamma(t) = \gamma_t = [\gamma_t^1, \gamma_t^2, \ldots, \gamma_t^d], \tag{2.40}$$

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 [166]

$$S(\gamma)_{a,t}^i = \int_a^t d\gamma_s^i = \gamma_t^i - \gamma_a^i, \qquad (a < s < t); \tag{2.41a}$$

$$S(\gamma)_{a,t}^{i,j} = \int_a^t S(\gamma)_{a,s}^i d\gamma_s^k = \int_a^t \int_a^s d\gamma_r^i d\gamma_s^j, \qquad (a < r < s < t); \tag{2.41b}$$

$$S(\gamma)_{a,t}^{i,j,\ldots,k} = \int_a^t S(\gamma)_{a,s}^{i,j,\ldots,k-1} d\gamma_s^k = \int_a^t \ldots \int_a^{t_2} d\gamma_{t_1}^i \ldots d\gamma_{t_k}^k, \quad (a < t_1 < t_2 < \ldots < t). \tag{2.41c}$$

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 2.6.4.1, it is seen that the kth landscape function $\lambda_k(t)$ can be written as a two-dimensional path

$$\gamma_t(\lambda_k(t)) = [t, \lambda_k(t)]. \tag{2.42}$$

Therefore, signatures can be obtained from persistence landscapes and used as features in machine learning algorithms [76]. In this study, path signatures are used up to the second level. Specifically, let $\lambda_{r,i}$ be the rth persistence landscape corresponding to the ith persistence diagram. Then the signatures used from the rth landscape function are given by $\mathbf{S}(\gamma_t(\lambda_r(t))) = [S^1_{r,i}, S^2_{r,i}, S^{1,1}_{r,i}, S^{1,2}_{r,i}, S^{2,1}_{r,i}, S^{2,2}_{r}]$.

Table 2.17: 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	$S^1_{1,1}$	$S^2_{1,1}$	$S^{1,1}_{1,1}$	$S^{1,2}_{1,1}$	$S^{2,1}_{1,1}$	$S^{2,2}_{1,1}$	$S^1_{2,1}$	$S^2_{2,1}$	$S^{1,1}_{2,1}$	$S^{1,2}_{2,1}$	$S^{2,1}_{2,1}$	$S^{2,2}_{2,1}$
X_2	0	$S^1_{1,2}$	$S^2_{1,2}$	$S^{1,1}_{1,2}$	$S^{1,2}_{1,2}$	$S^{2,1}_{1,2}$	$S^{2,2}_{1,2}$	$S^1_{2,2}$	$S^2_{2,2}$	$S^{1,1}_{2,2}$	$S^{1,2}_{2,2}$	$S^{2,1}_{2,2}$	$S^{2,2}_{2,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
X_n	1	$S^1_{1,n}$	$S^2_{1,n}$	$S^{1,1}_{1,n}$	$S^{1,2}_{1,n}$	$S^{2,1}_{1,n}$	$S^{2,2}_{1,n}$	$S^1_{2,n}$	$S^2_{2,n}$	$S^{1,1}_{2,n}$	$S^{1,2}_{2,n}$	$S^{2,1}_{2,n}$	$S^{2,2}_{2,n}$

By incorporating higher order signatures or signatures from more landscape functions, a longer feature vector can be constructed for classification. For example, Table 2.17 shows the second level feature vectors computed using the first and second landscape functions for n persistence diagrams.

In the experiment, a classifier is trained using 75% of the data and tested using the remaining 25%. A feature vector is constructed for each of the first five landscape functions. Table B.16 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 2.23.

2.6.4.6 Template Functions

Template functions are introduced in Reference. [72]. Given a persistence diagram $\mathcal{D}$, its coordinate system is converted into birth-lifetime diagram.

A template function for a persistence diagram is defined as

$$v_f(\mathcal{D}) = \sum_{(b,p)\in\mathcal{D}} f(b,p), \tag{2.43}$$

where b and p represent the birth time and lifetime, respectively. The set of template functions forms a template system $\mathcal{T}$. For more details about template functions and template systems, one can refer to Reference [72]. Here a template system of Chebyshev polynomials is defined using $f(x,y) = \beta(x,y) \cdot |l_i^{\mathcal{A}}(x) l_i^{\mathcal{B}}(y)|$, where $l_i^{\mathcal{A}}$ and $l_i^{\mathcal{B}}$ are the Lagrangian functions [72] computed on mesh $\mathcal{A}$ and $\mathcal{B}$ which are defined to include all points in the persistence diagram.

2.6.5 Modeling of Milling Process

This section explains how to generate time series using an analytical model of the milling process. A milling operation is considered with straight edge cutters as shown in Figure 2.38. A single degree of freedom model in the x direction for the tool oscillations is used as shown in Figure 2.38a, and both upmilling and downmilling processes are considered in the analysis. The equation of motion that describes the tool oscillations is

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = \frac{1}{m}F(t), \tag{2.44}$$

where m, w_n, ζ and $F(t)$ represent the modal mass, natural frequency, damping ratio and the cutting force in the x direction, respectively. τ is the time delay given by $\tau = 2\pi/N\Omega$ where ω is the spindle's rotational speed in rad/s, while N is the number of cutting edges or teeth. The expression for the cutting force is given by [167, 168]

$$F = \sum_{n=1}^{z} \big[-bK_t g_n(t)(\cos\theta_n(t) + \tan\gamma \sin\theta_n(t)) \sin\theta_n(t) \big] \big[(f + x(t) - x(t-\tau)) \big], \tag{2.45}$$

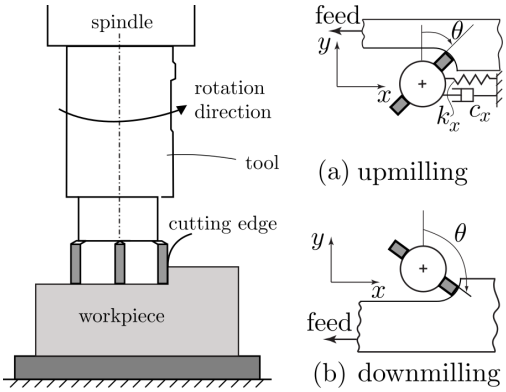


Figure 2.38: Milling process illustrations. a) Upmilling b) Downmilling

where θ_n is the angle between the vertical line and the leading tooth of the cutting tool as shown in Figure 2.38. The constant K_t is the linearized cutting coefficient in the tangential direction and $\tan\gamma = K_n/K_t$ where $K_n(t)$ is the cutting coefficient in the normal direction. The screening function $g_n(t)$ is either 0 or 1 depending on whether the nth tooth is engaged in the cut or not, respectively, and f represents the feed per tooth of the cutting tool. The expression for angular position of the nth tooth $\theta_n(t)$ is given by [168]

$$\theta_n(t) = (2\pi\Omega/60)t + 2\pi(n-1)/z, \tag{2.46}$$

where z is the total number of cutting teeth while Ω is the rotational speed given in revolutions per minute (rpm).

One of the important cutting parameters is the radial immersion ratio (RI) which is defined as the ratio of the radial depth of cut to the diameter of the cutting tool. Smaller radial immersions indicate shallower cuts and thus more intermittent contact between the tool and the workpiece, while higher radial immersions indicate deeper cuts with more continuous contact. In the simulations for both downmilling and upmilling, RI is set to 0.25.

Inserting Equation (2.45) into Equation (2.44) results in

$$\ddot{x}(t) + 2\zeta\omega_n(t)\dot{x}(t) + w_n^2 x(t) = -\frac{bh(t)}{m}[x(t) - x(t-\tau)] - \frac{bf_0(t)}{m}, \tag{2.47}$$

where b is the nominal depth of cut and $h(t)$ is the τ-periodic function

$$h(t) = \sum_{n=1}^{z} K_t g_n(t)\big[\cos\theta_n(t) + \tan\gamma\sin\theta_n(t)\big]\sin\theta_n(t), \tag{2.48}$$

and $f_0(t) = h(t)\, f$. The term $f_0(t)$ does not affect the stability analysis, so it is dropped in the subsequent equations; however, it is kept in the simulation.

After dropping $f_0(t)$, the equations of motion can be written in state space form according to

$$\frac{d\boldsymbol{\xi}(t)}{dt} = \mathbf{A}(t)\boldsymbol{\xi}(t) + \mathbf{B}(t)\boldsymbol{\xi}(t-\tau), \tag{2.49}$$

where $\mathbf{A}$ and $\mathbf{B}$ are T-periodic with $T = \tau$. Then, using the spectral element method [2], the state space is discretized. Then a dynamic map is obtained such that

$$\boldsymbol{\xi}_{n+1} = U\boldsymbol{\xi}_n, \tag{2.50}$$

where $\mathbf{U}$ is the finite dimensional monodromy operator. The eigenvalues of U approximate the eigenvalues of the infinite dimensional monodromy operator of the equation of motion. If the modulus of the largest eigenvalue is smaller than 1, then the corresponding spindle speed and depth of cut pair lead to a chatter-free process; otherwise, chatter occurs. Therefore, the stability of the milling model and the bifurcation associated with the loss of stability (chatter) can be obtained by examining these eigenvalues, see Figure 2.39.

In this study, 10000 time series were generated corresponding to a 100×100 grid in the plane of the spindle speeds and depths of cut. Each time series is tagged using the largest eigenvalue of the monodromy matrix corresponding to the same grid point.

2.6.6 Simulation Results

In this section, classification accuracies for each featurization method are provided for noisy and non-noisy time series of up milling and down milling processes with 4 teeth ($N = 4$). The details of the simulation can be found in Section 2.6.5. Ranges of rotational speed and depth of cut parameters for the simulations are chosen with respect to the stability diagrams given for both processes in [168]. The 1- and 2-dimensional persistence diagrams were used with the methods described in Section 2.6.4. Feature matrices were computed for 1D and 2D persistence diagrams individually, and the features were concatenated when using both

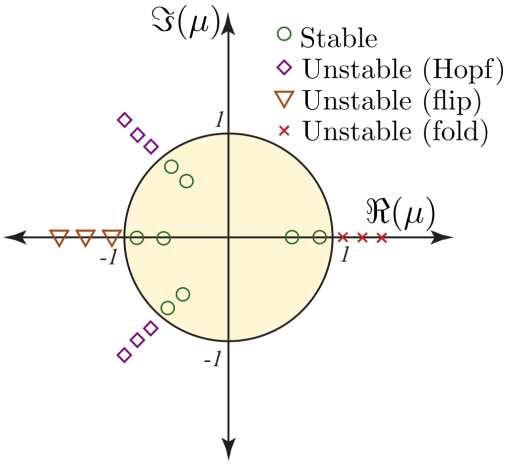


Figure 2.39: Stability criteria used in this study based on the eigenvalues of the monodromy matrix **U**.

dimensions. 0D persistence has been omitted in this study due to its poor performance on noisy data sets, as evidenced by a reduction in the classification accuracy by 10% in some cases. Data sets are randomly split, using 67% for training and 33% for testing. The split-train-test is performed 10 times, and the mean accuracies with the corresponding standard deviations are reported in this section.

This data can be used for both a two-class and three-class classification problem. The first is classifying either chatter-free or chatter, while the second further divides chatter into two types: Hopf-unstable and period2-unstable. Classification for both two and three class problems is done using four different algorithms: support vector machines, logistic regression, random forests, and gradient boosting. Default parameters have been used for all classification algorithms except random forest classification ($n\_estimator = 100$ and $max\_depth = 2$). These two types of chatter are based on Hopf and period doubling bifurcation behaviors as described in Fig. 2.39.

Two class classification results for downmilling and upmilling simulations with $N = 4$ are provided in Tables 2.18 and 2.19, respectively. For each data set, the highest accuracy is highlighted in blue. For instance, 95.5% accuracy is obtained as the best classification accuracy for non-noisy data sets when gradient boosting classifiers are trained with combined 1D, and 2D persistence features based on Template Functions method in Table 2.18. In most

Table 2.18: 2 Class classification results for noisy (SNR:20,25,30 dB) and non-noisy data sets which belong to downmilling process with $N = 4$ (N: Teeth Number, CC: Carlsson Coordinates, TF: Template Functions, SVM: Support Vector Machine, LR: Logistic Regression, H_1: 1D persistence, H_2: 2D persistence).

Down Milling	Without Noise						SNR: 20 dB					
$N = 4$	CC			TF			CC			TF		
Classifier	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2
SVM	94.3%	85.1%	94.6%	92.9%	94.4%	93.7%	94.2%	92.5%	94.6%	94.3%	94.8%	94.7%
LR	92.4%	84.3%	92.8%	93.9%	91.6%	94.5%	78.5%	78.3%	91.1%	93.8%	93.5%	94.5%
RF	93.6%	90.9%	93.8%	95.0%	94.1%	95.7%	92.4%	92.3%	93.0%	94.3%	94.4%	94.6%
GB	95.0%	93.5%	95.2%	94.7%	94.2%	95.5%	94.2%	93.7%	94.9%	94.7%	94.4%	94.7%
Down Milling	SNR: 25 dB						SNR: 30 dB					
$N = 4$	CC			TF			CC			TF		
Classifier	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2
SVM	80.8%	77.2%	83.1%	89.4%	83.2%	90.8%	81.0%	77.4%	82.8%	89.2%	83.4%	90.9%
LR	76.0%	72.2%	75.4%	83.7%	77.5%	85.7%	76.2%	72.3%	75.1%	83.8%	77.2%	85.6%
RF	76.9%	75.0%	77.5%	88.9%	82.4%	89.6%	76.6%	75.1%	77.1%	88.7%	82.2%	89.9%
GB	88.3%	79.1%	89.5%	89.0%	82.7%	90.2%	88.5%	79.0%	89.5%	88.7%	83.0%	90.5%

of the cases for downmilling and upmilling, it is seen that the highest accuracies are obtained when 1D and 2D persistence diagrams features are combined.

Some of the time series embeddings, especially in the chatter-free regime, do not have any 2 dimensional topological features, thus giving an empty H_2 diagram. If a specific cutting configuration has a lot of time series with empty H_2, feature matrices for these time series have a lot of zeros when either featurization method is used. Because of the lack of 2 dimensional information for many of the time series, classifications using only H_2 have lower accuracies than only using H_1 as is shown in Tables 2.18 and 2.19.

When comparing persistence diagram featurizations, the template function method has the best results for all data sets, with the exception of two: the noisy data set with an SNR value of 20 dB for downmilling and the one without noise for upmilling. However, for those two data sets, template functions' results are very close to those provided by Carlsson coordinates. When comparing classification algorithms, SVM yields the highest accuracy for five of the eight data sets, while gradient boosting yields the highest accuracy for the remaining three data sets.

To compare the results of different levels of noise and different dimensions of persistence

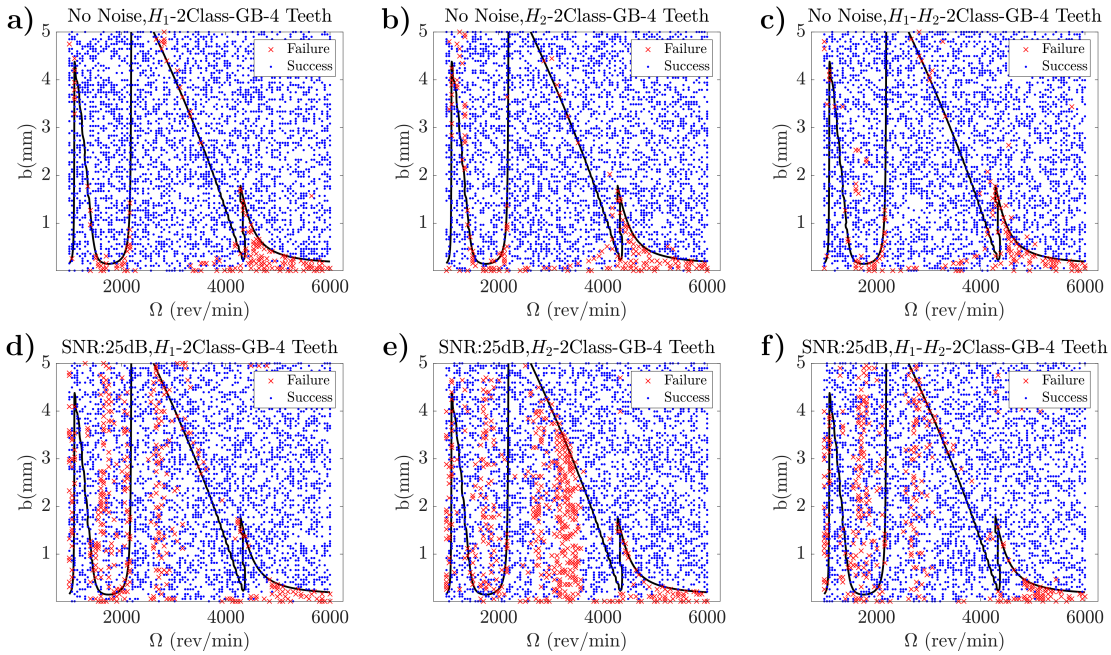


Figure 2.40: Success and failure of two class classifications performed with Template Function feature matrices and Gradient Boosting algorithm for test set of data set without noise and with SNR value of 25 dB. a) Classification with 1D persistence features for non-noisy data set, b) Classification with 2D persistence features for non-noisy data set, c)Classification with 1D-2D persistence combined features for non-noisy data set, d) Classification with 1D persistence features for noisy data set with SNR:25 dB, e) Classification with 2D persistence features for noisy data set with SNR:25 dB, f) Classification with 1D-2D persistence combined features for noisy data set with SNR:25 dB.

diagrams, classification results are plotted on the 100×100 grid of the stability diagram for the milling process. Figure 2.40 presents the stability diagrams belonging to teeth number $N = 4$ of down milling process for noisy data with SNR value of 25 dB and non-noisy data sets. Figures on the first and second columns belong to the classifications performed with only H_1 and H_2 features, respectively, while the ones in the third column represent the results of combinations of H_1 and H_2 features. Red crosses on the stability diagrams denote the case that the prediction of the classifier does not match with the true label of the corresponding time series while blue dots show matching between predictions and true labels. From the figures, it is clear that the number of misclassifications increases slightly when the noise is introduced into the simulation data. This is also reflected in Table 2.18 in the decrease in accuracies for different levels of noise, especially the noisy data sets with SNR values of 25

Table 2.19: 2 Class classification results for noisy (SNR:20,25,30 dB) and non-noisy data sets which belong to upmilling process with $N = 4$ (N: Teeth Number, CC: Carlsson Coordinates, TF: Template Functions, SVM: Support Vector Machine, LR: Logistic Regression, H_1: 1D persistence, H_2: 2D persistence).

UpMilling	Without Noise						SNR: 20 dB					
$N = 4$	CC			TF			CC			TF		
Classifier	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2
SVM	86.0%	78.5%	85.8%	86.1%	80.4%	86.0%	76.8%	80.7%	82.1%	84.6%	84.0%	85.1%
LR	85.3%	77.8%	85.3%	86.2%	81.3%	85.8%	69.4%	80.4%	80.9%	82.7%	81.3%	84.1%
RF	84.9%	80.3%	84.9%	85.9%	81.2%	85.7%	75.5%	80.7%	81.1%	82.8%	81.8%	83.2%
GB	86.1%	80.9%	86.0%	85.6%	81.3%	86.0%	80.6%	82.2%	82.5%	84.1%	83.4%	84.6%
Upmilling	SNR: 25 dB						SNR: 30 dB					
$N = 4$	CC			TF			CC			TF		
Classifier	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2	H_1	H_2	H_1-H_2
SVM	85.3%	83.4%	84.8%	85.5%	84.4%	85.5%	83.2%	71.6%	83.1%	85.9%	75.0%	86.2%
LR	79.2%	84.0%	84.1%	84.2%	84.5%	84.5%	79.4%	72.3%	79.5%	84.1%	75.3%	85.0%
RF	83.8%	84.1%	84.5%	83.5%	82.6%	83.0%	84.3%	75.0%	84.4%	83.9%	75.3%	83.8%
GB	85.2%	84.3%	84.8%	85.1%	84.5%	84.9%	85.1%	74.6%	85.1%	85.7%	75.7%	85.2%

and 30 db.

In addition, there is a small accuracy difference which is at most 5% between noisy (SNR:25, 30 dB) and non-noisy data set for downmilling cases, while this difference is less for upmilling results presented in Table 2.19 This suggests that the featurization methods used yield promising results even with noisy data. Persistent homology is known to be very robust against noise, as noise only adds points close to the diagonal, which have short lifetimes. Thus, these points do not contribute significantly to the Carlsson coordinate or template function methods, making both featurizations robust against noise as well.

Figure 2.41 shows a comparison of the results obtained for up and downmilling with respect to different noise levels. Since the deviations of accuracies for both featurization methods are relatively low, the classification accuracies can be considered reliable. This trend is noticeable for both up and downmilling and for all levels of noise. However, the classification results for upmilling are noticeably lower than those for downmilling. Additionally, it is clear that H_2 features do not perform as well due to the lack of higher dimensional topological structure, as was explained earlier. Figure 2.41 also presents the classification results on the stability diagrams for upmilling and downmilling for noisy data sets. It is

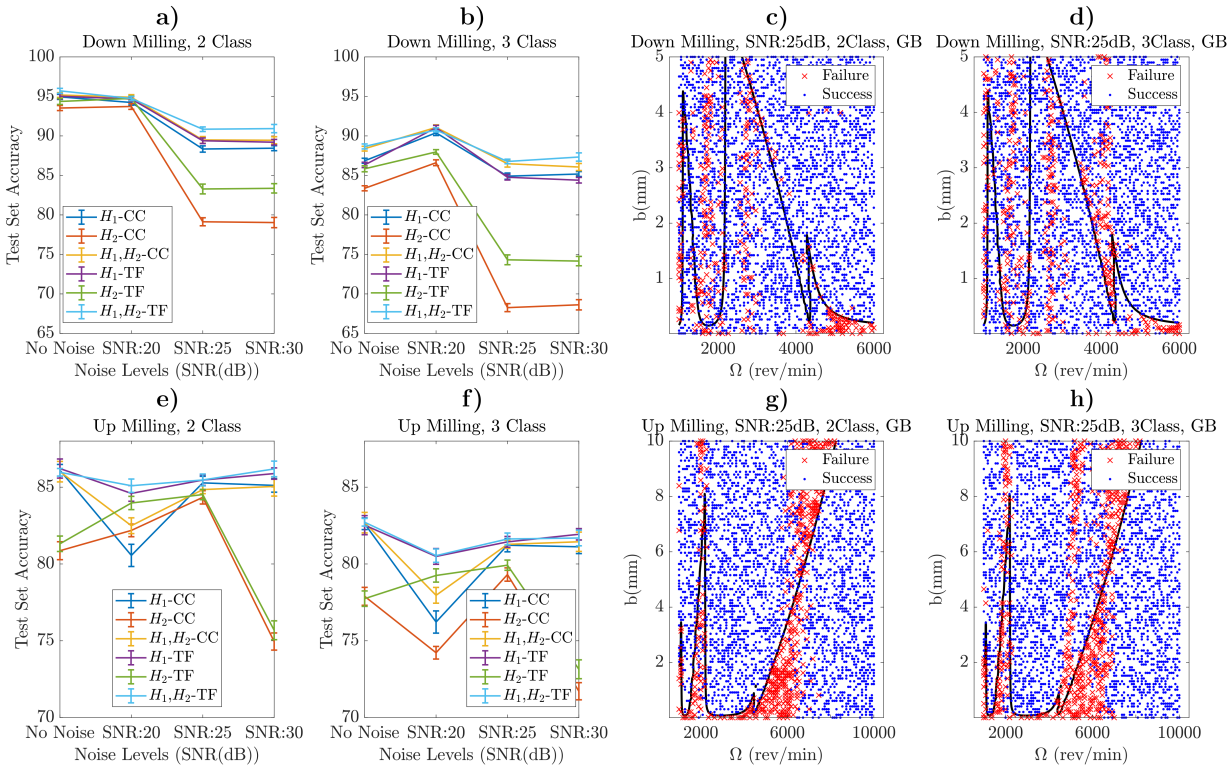


Figure 2.41: Mean accuracies of downmilling process (a,b) and upmilling process (f,g) obtained for two class and three class classification performed with Carlsson Coordinates and Template Functions for non-noisy and noisy data sets where teeth number is 4. Two class(c) and three class (d) classification results obtained with Gradient Boosting algorithm is shown on the stability diagram for downmilling simulation data set whose SNR is 25 dB. Two class(e) and three class (h) classification results obtained with Gradient Boosting algorithm are shown on the stability diagram for upmilling simulation data set whose SNR is 25 dB.

seen that many misclassifications occur nearby the boundary of the stability diagram, especially for the upmilling process. This boundary separates the unstable (above the boundary curve) and stable (under the boundary curve) cases, so misclassifications on this boundary are likely. It is also clear that when increasing to the three-class problem, there are an increased number of misclassification. However, the overall accuracy difference between both implementations, two and three-class classification, does not exceed 5% at their maximum accuracies.

The findings of this study indicate that topological features of data are appropriate descriptors for chatter recognition in milling. One advantage of the described approach is its ability to provide promising results without the need for manual preprocessing not only

for non-noisy data sets but also for time series with noise.

2.6.7 Experimental Data Results

2.6.7.1 Runtime Comparison

Runtime is a criterion for comparing the different feature extraction methods. For the TDA-based methods, the total runtime required for classification is split among three main computations: (1) obtaining the persistence diagrams, (2) obtaining features or computing kernels, and (3) training and testing the corresponding classifier. However, obtaining results with serial computing takes significantly larger runtime. Therefore, parallel computing is implemented to improve the runtime. High Performance Computing Center (HPCC) of Michigan State University is utilized for parallel computing. It includes several supercomputers, which are composed of hundreds of nodes. Each node represents a computer with a certain number of processors and RAM capacity. Users are allowed to submit multiple jobs at the same time to HPCC, and they can define the number of CPUs per job and memory per CPU. 10 CPUs per job and 2GB of memory per CPU are requested to compute the embedding parameters and persistence diagrams in parallel. The number of jobs submitted to HPCC changes depending on the number of time series of the overhang distance.

Embedded time series are subsampled such that every 10^{th} point is taken into account to compute persistence diagrams. The times to complete persistence diagram computation of all overhang cases are recorded, and they are reported in Table 2.20. It also includes times for serial computing, where one persistence diagram is computed at a time. It is seen that parallel computing reduces the computation time significantly, although most part of the runtimes for parallel computing is the queue time. Parallel computing can also be performed with some workstations available in the market without having the need for expensive supercomputers that HPCC has. Entry-level workstations with a CPU having 64 cores and 512 GB of RAM can be afforded by small workshops.

Table 2.20: Comparison of runtimes (seconds) for embedding parameters and persistence diagram computation of all overhang cases with parallel and serial computing.

	5.08 cm (2 inch)		6.35 cm (2.5 inch)		8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
	Parallel	Serial	Parallel	Serial	Parallel	Serial	Parallel	Serial
Persistence Diagram	9420	84346	3448	23570	2073	11319	4819	37617

Despite the long computation time for persistence-based methods, it is noted that after obtaining the persistence diagrams, they can be saved and used in multiple TDA-based classification methods. In addition, it was observed that delay and embedding dimension parameters do not change significantly for changing time series of the same cutting configuration. Parameters for embedding can be computed in the training phase of a classifier, and they will be used in the test phase. Therefore, once these diagrams are computed, the time required for featurization and classification would be a fraction of the ones reported in Table 2.20. It is worth mentioning 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. Therefore, the runtimes needed for a single time series are compared with different methods. Table 2.21 provide runtimes for embedding parameters computations and persistence diagram computation with different methods.

First column in Table 2.21 represent the runtimes of computing embedding dimension and delay parameter. When these parameters are computed, they can be saved and used in embedding time series In the second column of Table 2.21, the runtime of persistence diagram computation of subsampled point cloud is given. In this way, nearly 1000 points are used from the embedded time series to compute persistence diagrams. However, the computation time for persistence diagrams of a point cloud with that size is still high, as seen from Table 2.21. Therefore, greedy permutation subsampling and the Bézier curve approximation technique are also employed. The greedy permutation option of the *Ripser* package is utilized, and it

Table 2.21: Runtime (seconds) for embedding parameter computation and persistence diagram computation of a single time series with different methods.

	Embedding Parameters	Persistence Diagram						
		Number of points≈ 1000	Number of points≈ 100			Number of points≈ 300		
Overhang Distances	Delay and Dimension	Subsampled Point Cloud	Greedy Permutation $n_{perm} = 100$	Bézier $r = 1$ $spg = 100$ (Serial)	Bézier $r = 1$ $spg = 100$ (Parallel)	Greedy Permutation $n_{perm} = 300$	Bézier $r = 3$ $spg = 100$ (Serial)	Bézier $r = 3$ $spg = 100$ (Parallel)
5.08 cm (2 inch)	242.63	266.95	0.2	106.00	0.93	3.62	569.78	6.21
6.35 cm (2.5 inch)	191.10	208.95	0.29	106.15	0.85	3.79	538.98	6.19
8.89 cm (3.5 inch)	166.79	296.21	0.18	96.00	0.72	3.87	541.62	6.48
11.43 cm (4.5 inch)	200.51	276.38	0.17	113.86	0.77	3.53	600.29	6.99

is a method that subsamples the point cloud and computes the persistence diagrams with less number of points. n_{perm} is a parameter that defines the number of points selected by the greedy permutation algorithm. 100 and 300 points are chosen for this option, and runtimes are reported for the resulting persistence diagrams. In Table 2.21, the runtimes are grouped with respect to the number of points used in the corresponding method. It is seen that the Bézier curve approximation with $r = 1$ and $r = 3$ uses approximately 100 and 300 points as in the case of greedy permutation. Runtime for both serial and parallel computing are provided in Table 2.21. Parallel computing can only be applied to Bézier curve approximation among the methods of persistence diagram computation given in Table 2.21. The reason is that persistence diagrams are obtained directly from the *Ripser* package for other methods. However, the steps of the Bézier curve approximation method, computation of coefficients for the line segments, and the distance matrix between these line segments can be performed in parallel. In these two steps, a job can only compute coefficients of the lines in a single group or a distance between two lines. The number of the jobs will be equal to the group number and number of combinations between lines for coefficient computation and distance matrix computation, respectively. Ideally, all jobs for a step can be computed simultaneously if there is no queue time. Therefore, runtimes are recorded for computation of coefficients

of the line segments in a single group, computation of a distance between two line segments, and persistence diagram from a distance matrix individually. Then, they are summed up and reported in Table 2.21. Combining parallel computing with the Bézier curve approximation reduces the runtime significantly. Moreover, it is seen that the fastest method is the greedy permutation with $n_{perm} = 100$ and Bézier curve approximation computed in parallel places second. Both methods are able to complete the diagram computation in less than a second, while runtime gets larger with increasing n_{perm} and r parameters.

Table 2.22 provides the times required to complete classification of a single time series. To be fair in comparison between the runtime of WPT/EEMD and TDA-based methods, it is assumed that the classifier is already trained and required parameters for all methods are selected. It is seen that WPT is the fastest method, and EEMD places second. The runtime for the TDA-based method is comparable to the ones for EEMD. Further, the WPT and EEMD methods use codes that have been highly optimized, whereas the TDA-based methods are still under active research with huge future potential for improved optimization. It is believed that the runtimes for the TDA-based method can be further decreased with optimization.

Table 2.22: Runtime (seconds) for performing classification with a single time series for TDA-based methods and signal decomposition-based ones.

	Topological Data Analysis				Signal Decomposition	
Overhang Distances	Persistence Landscapes	Template Functions	Carlsson Coordinates	Persistence Images	WPT	EEMD
5.08 cm (2 inch)	1.01	0.97	0.97	0.97	0.03	0.52
6.35 cm (2.5 inch)	0.92	0.90	0.90	0.87	0.08	0.65
8.89 cm (3.5 inch)	0.81	0.81	0.76	0.76	0.09	0.70
11.43 cm (4.5 inch)	0.87	0.81	0.81	0.81	0.06	0.52

2.6.7.2 Classification Scores

This section presents the classification accuracies for all the methods that are introduced in Section 4.2.2 and compares them to the results in Reference [9], 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. For persistence images, Template Functions, and Carlsson Coordinates, four different classifiers are applied, and these are Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), and Gradient Boosting (GB) algorithms. All of these classifiers except Gradient Boosting are used in the Persistence Landscape method, while the Kernel method and Persistence Paths results are obtained with LibSVM and SVM classifiers, respectively. The classification results are summarized in Table 2.23 where for each cutting configuration, the best results of the classification algorithms for each method are included. Table 2.23 also includes the classification results obtained using a new TDA approach, which is not included in Section 4.2.2, based on template functions [72]. In this table, the best accuracy for each dataset is highlighted in green. Further, methods whose accuracy is within one standard deviation of the best result in the same category are highlighted in blue.

Table 2.23: Comparison of results for each method where WPT is the Wavelet Packet Transform, and EEMD stand for Ensemble Empirical Mode Decomposition.

Overhang Length cm (inch)	Persistence Landscapes	Persistence Images	Template Functions	Carlsson Coordinates	Kernel Method	Persistence Paths	WPT	EEMD
5.08 (2)	96.8%	96.4%	91.5%	93.6%	74.5%*	83.0%	93.9%	84.2%
6.35 (2.5)	88.6%	85.8%	89.3%	86.3%	58.9%	84.2%	100.0%	78.6%
8.89 (3.5)	92.2%	93.0%	83.9%	95.7%	87.0%	85.9%	84.0%	90.7%
11.43 (4.5)	68.6%	72.5%	65.1%	72.2%	59.3%	70.0%	87.5%	79.1%

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

Table 2.23 shows that the WPT approach yields the highest classification accuracy for

the 6.35 cm (2.5 inch) and the 11.43 cm (4.5 inch) overhang cases. However, it is also seen that for the 5.08 cm (2 inch) and the 8.89 cm (3.5 inch) case, persistence landscape, and Carlsson Coordinates yield the highest accuracy, respectively. For the 6.35 cm (2.5 inch) overhang case, it is worth noting that the number of time series is small. Specifically, for this case, less than 10 time series were divided into small pieces and used as the test set, see Table A.1. Therefore, the 100% classification accuracy using WPT for this case does not represent a robust result. Nevertheless, for the same case Table 2.23 shows that the TDA methods based on persistence landscapes, persistence images, 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 95.7%, placing ahead of both WPT and EEMD. Further, the other TDA-based method for this cutting configuration score classification accuracies of at least 83.9%. For the last case, the TDA-based approaches underperform in comparison to WPT. To investigate the reason of this result, persistence diagram plots belonging to some of the time series from each overhang distance are provided in Figure 2.42. For the first three cases (5.08, 6.35, and 8.89 cm), it is seen that persistence diagrams of stable time series show a single significant feature with a high persistence value, which indicates the existence of a loop i.e., periodic behavior, in reconstructed state space. However, this single high persistence point is not observed in the persistence diagrams of unstable (chatter) time series. This significant difference between persistence diagrams enables classification algorithms to distinguish chatter and chatter-free time series. For the last case (11.43 cm), it is seen that all persistence diagrams look similar, thus making blurring the signature of chatter in a time series. Persistence Images and Carlsson Coordinates depend on the persistence value of the features obtained from the persistence diagrams. Since all persistence diagrams have features with similar persistence values in the 11.43 cm (4.5 inch) case, classifiers can not distinguish the two classes. This leads to reduced accuracy, as shown in Table 2.23. Nevertheless, WPT and EEMD methods for chatter detection necessitate manual preprocessing by well-trained

expert users, thus requiring significant extra time and advanced expertise [9]. Therefore, the automation of these processes is not straightforward with WPT and EEMD, while all the steps in TDA-based feature extraction can be fully automatized.

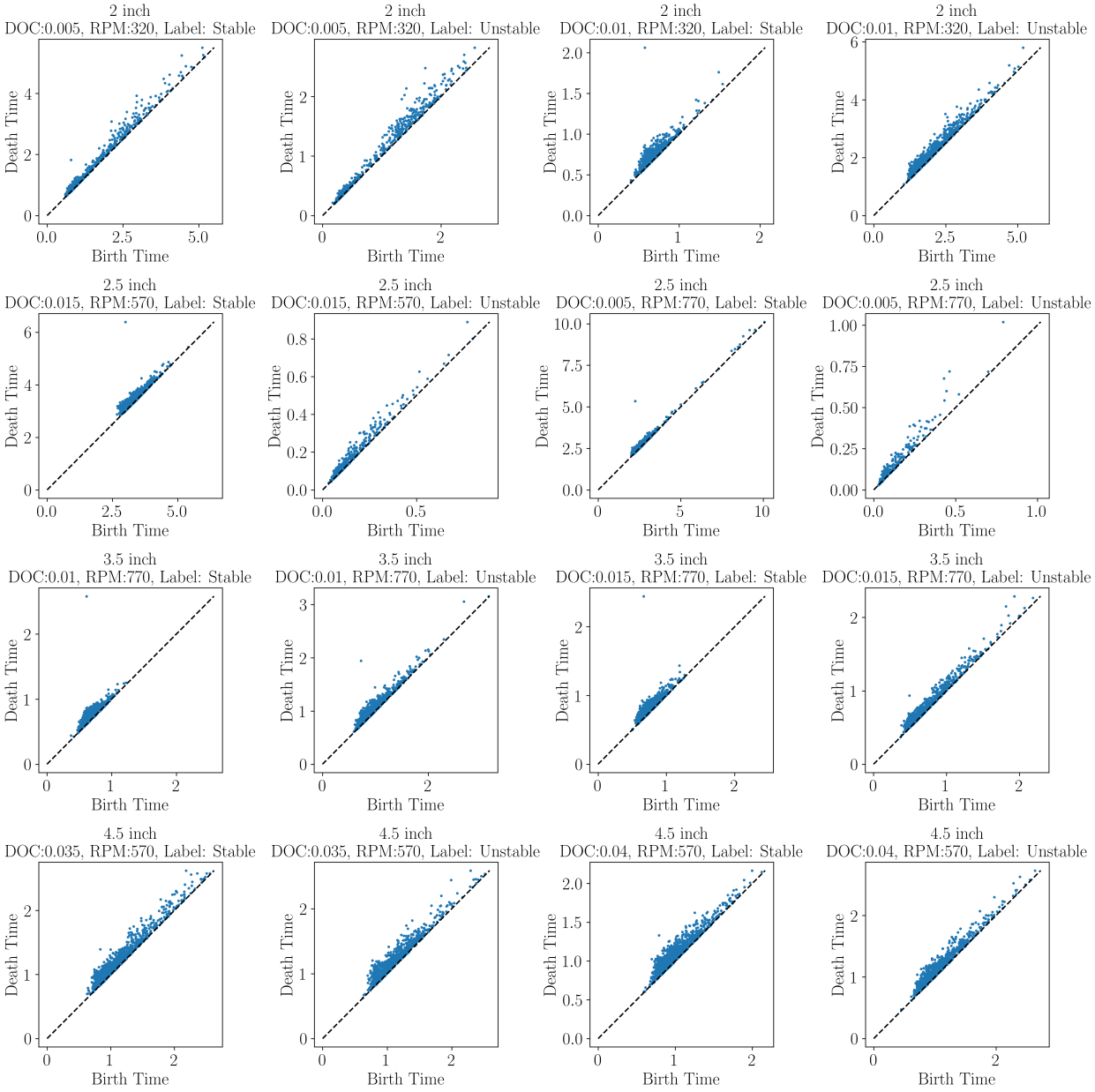


Figure 2.42: Sample persistence diagrams for each overhang distance.

The performance of different persistence diagram computation methods is compared, and their runtime is compared in Section2.6.7.1. Figure 2.43 shows you the mean classification accuracies and error for the persistence diagrams obtained with the ways shown in

Table 2.21. Subsampling the point cloud at every 10^{th} point is the first method to compute persistence diagrams. Table 2.21 shows that the Bézier curve approximation has a slightly larger computation time compared to the greedy permutation subsampling method when it is computed in parallel. However, it is seen that for all overhang cases Bézier curve approximation method results in higher accuracy compared to greedy permutation. Also, its results are the closest results to the ones obtained from subsampled point cloud.

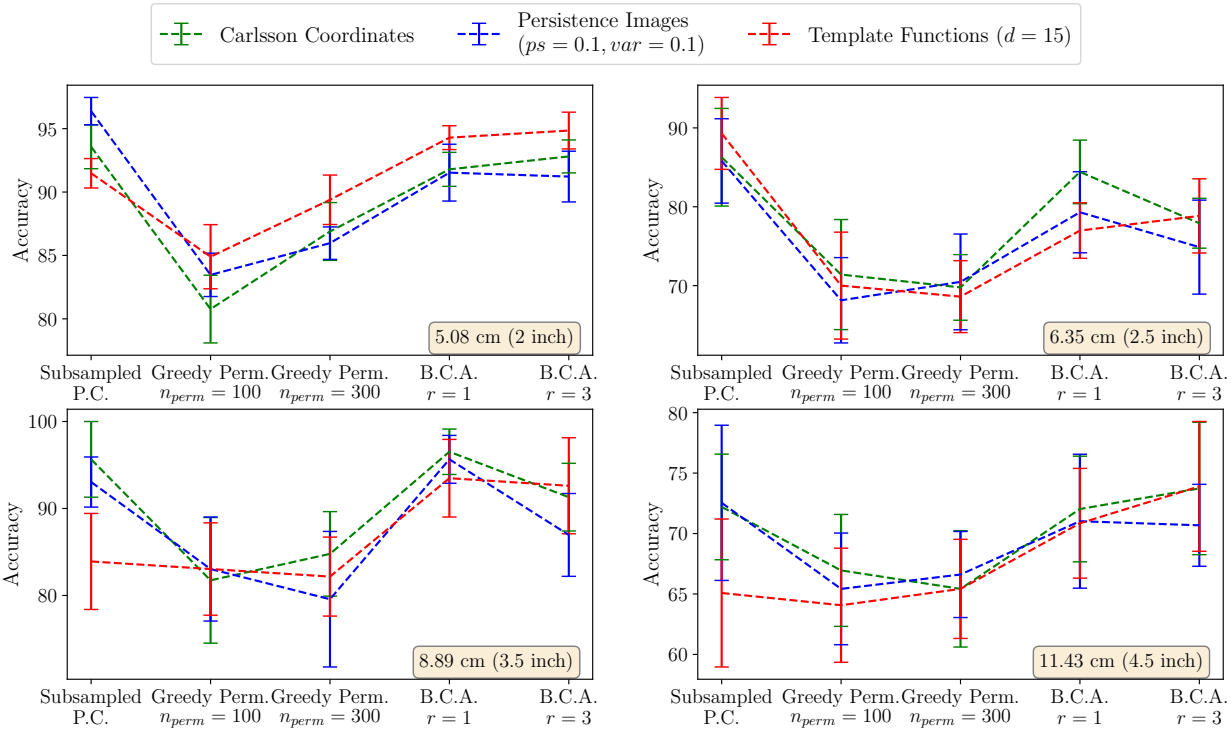


Figure 2.43: Classification performance of persistence diagrams obtained with different methods for all overhang cases.

Increasing the number of points in greedy permutation or increasing the number of line segments (r) generated for a group does not always yield higher accuracy, as seen from Figure 2.43. The reason could be that increasing the number of points or the number of line segments can cause more topological noise on persistence diagrams. For example, Figure 2.33 shows that the new point appears closed to a significant feature, which is the point with the highest lifetime, as r increases from 5 to 7. Therefore, this could cause small drops in accuracy for some overhang cases, as seen from Figure 2.43.

Greedy permutation and Bézier curve approximation method can provide persistence diagrams in less than 1 second without optimization. For the greedy permutation method, optimization can not be applied. However, coefficient computation for the line segments in the Bézier curve approximation method can be optimized. This could further decrease the runtimes and opens the possibility for exploring in-situ chatter detection using TDA-based methods, especially with properly optimized algorithms.

2.7 Transfer Learning

In traditional machine learning, a classifier is trained and tested on a data set originating from the same source. However, real-life applications, such as chatter or fault detection in machining, can experience a shift in the parameters between the time the classifier was trained and the time the system is put into operation. This means that the data collected from these applications may no longer have the same feature space as the training set. Therefore, traditional machine learning can require data collection for each parameter combination, thus leading to increased cost and low automation potential. As another motivating example, some experiments are expensive to set up and perform. This includes chatter studies which result in long downtime for production machines and personnel during the data collection phase. Besides the cost, some sensor data may be collected during machining one-off products and, therefore, may be considered of limited use in traditional machine learning settings. Therefore, it is extremely beneficial to leverage extracted features related to similar phenomena across different settings and operations. In this case, *Transfer Learning* presents a useful machine learning framework that allows training and testing on data sets from different sources. As an example, Figure 2.44 shows a transfer learning application where a chatter classifier was trained using a turning process, and the gained information is then transferred for detecting chatter in a milling operation.

Transfer learning is categorized according to the similarity between tasks and the domain of each source and target. The source is the system used to train a classifier, while the

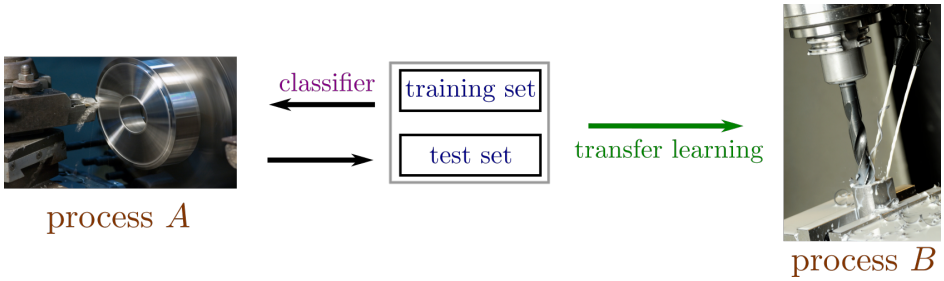


Figure 2.44: An example of transfer learning where training for chatter detection is performed using a turning process (the source) and the gained knowledge is imported via transfer learning to a milling operation (the target).

target is the system where the classifier is tested. There are two main terms in the definition of transfer learning, and these are domain and task. A domain can be described as the combination of a feature space $\mathcal{F}$ and the marginal probability of the feature space $P(F)$, while the task contains a label space $\mathcal{L}$ and the conditional probability $(P(l|f))$ [169]. $\mathcal{F}$ represents the space of feature vectors, $\mathbf{x_i}$, and F is the an instance set such that $F = \{\mathbf{f} \mid \mathbf{f_i} \in \mathcal{F}, i = 1, \ldots, n\}$ [170]. For a given domain, $\mathcal{D} = (\mathcal{F}, P(F))$, a task is defined as $\mathcal{T} = (\mathcal{L}, P(l|f))$. $P(l|f)$ is also considered as a predictive function f which estimates the label for a given feature space.

Based on the differences between domains and tasks of the source and the target, several transfer learning settings can be obtained (see Figure 2.45). The interested reader is referred to [169, 171, 172] for more details on transfer learning. In this study, the machine learning framework is included under *inductive* transfer learning category because the same sets of features are used for the source and the target. The main purpose of inductive transfer learning is to improve the performance of the target prediction function f_T using the information in the domain and task of the source $\mathcal{D}_S$ and $\mathcal{T}_S$, respectively [169]. There are several approaches to transfer learning. These include instance-transfer, feature representation transfer, parameter-transfer, and relational knowledge transfer [170]. In this study, the knowledge of parameters is transferred by using the same trained classifier in the testing phase. The same set of features is used for training and testing. However, the distribution

of the features is different in each domain since the source, and the target is represented by two different machining processes: turning and milling. More details about the application of *inductive* transfer learning are available in Section 2.7.2.

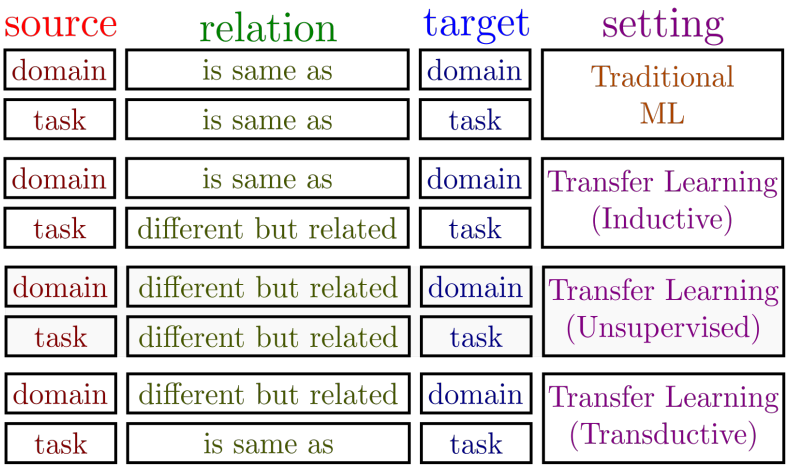


Figure 2.45: Categorization of transfer learning.

2.7.1 Methods

This section provides a brief description of feature extraction for the milling data set explained in Section A.2. The preprocessing for commonly adopted approaches, WPT and EEMD, and the traditional feature extraction approaches are explained in this section. In addition, Figure 2.46 provides a block diagram that explains the procedure followed in this study. Specifically, the leftmost block shows the experimental setup and the data collection process. This is followed by the middle block, which lists the featurization methods used and the similarity-based approach using DTW. The rightmost block shows the pairwise distance matrices and feature matrices obtained from the similarity-based approach and the feature extraction approaches, respectively. Figure 2.47 provides a cartoon of the transfer learning framework whereby classifiers trained on the turning data are used to detect chatter in milling and vice versa.

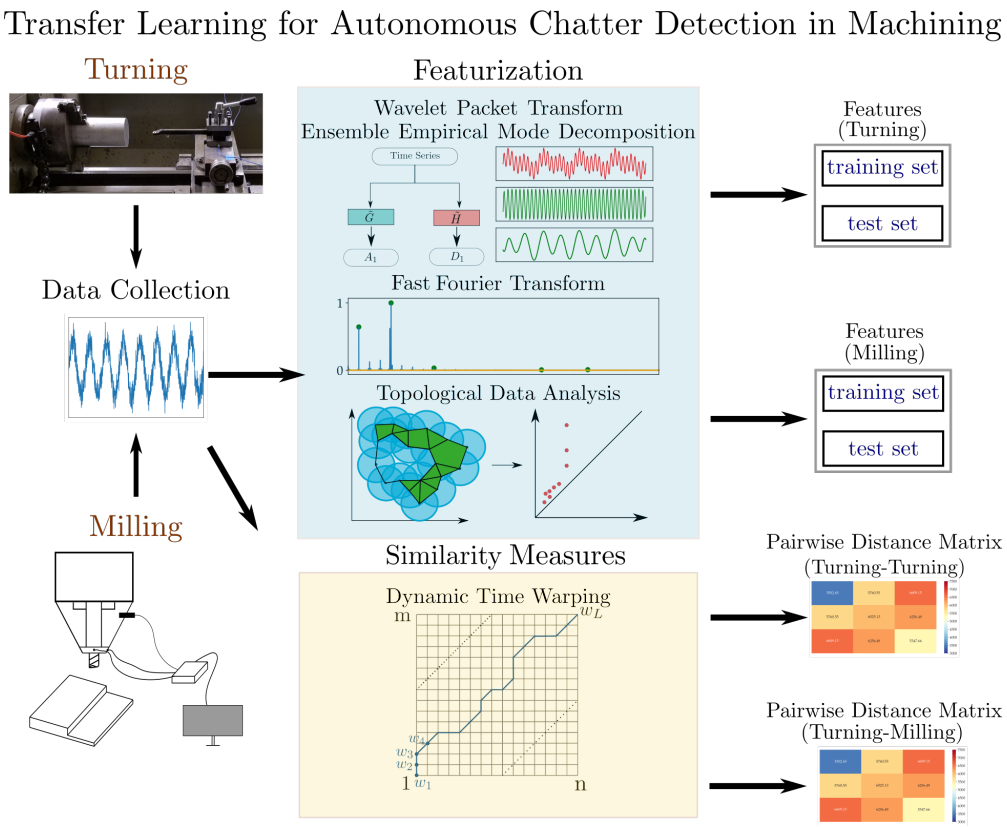


Figure 2.46: Outline of the general procedure and the featurization methods used in this study.

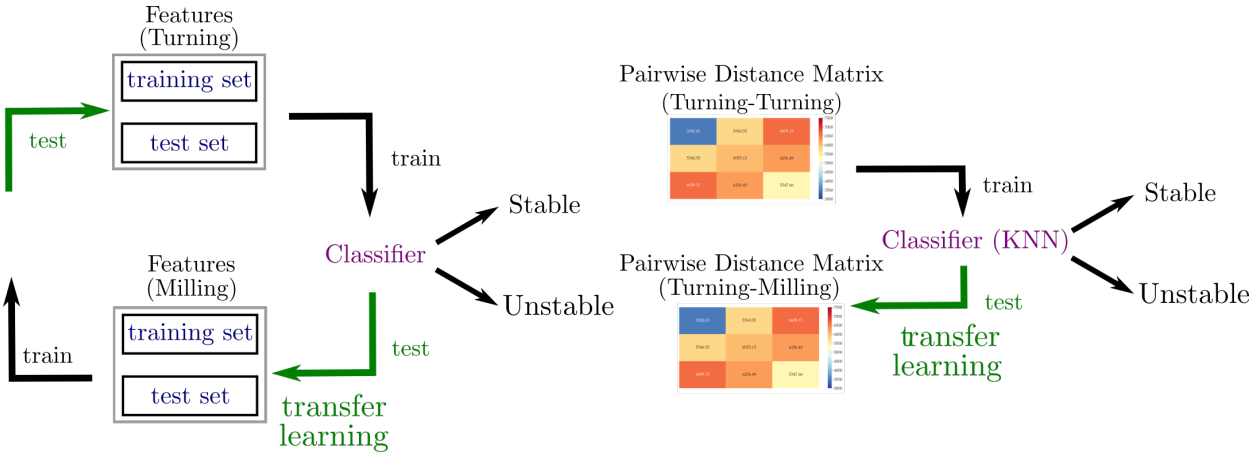


Figure 2.47: Transfer learning approach used in this chapter for feature extraction and similarity measure-based approaches.

2.7.1.1 Wavelet Packet Transform (WPT)

This section describes the salient details of the Wavelet Packet Transform (WPT) method. One can refer to Section 2.3.1 for more information. The WPT method decomposes a signal into approximation and detail coefficients at each level of the transform. Figure 2.7 provides the decomposition of a time series into three levels of WPT and shows the corresponding frequency content for each wavelet packet. Detail and approximation coefficients are obtained after applying the high-pass and low-pass filters, respectively. They are denoted as D_i and A_i as shown in Figure 2.7. At each level of the transform, additional letters A or D are added to the left side of the previous notation, and the indices change with respect to the level of the transform. For example, in the second level of transform, the approximation coefficient A_1 passes through the high pass filter and becomes DA_2 (see Figure 2.7). In addition, the number of wavelet packets at level k of the transform is 2^k.

Milling data set: The procedure followed in this study is the same as the one described in Reference [9]. Level 4 WPT is applied to downsampled time series for both the milling and turning data sets. The first step was to define the chatter frequency by checking the spectrum of the downsampled data. Figure 2.48 provides FFT plots of three different time series from the milling data set. It can be seen that the chatter frequency is around 850 Hz which is close to the resonant frequency of 728.7 Hz; this leads us to look for wavelet packets that also have a frequency content near this frequency. Time series were decomposed into wavelet packets, and the energy ratio of each wavelet packet was computed. The energy ratio plots and the Fast Fourier Transform (FFT) of the signals, reconstructed from the packets, have been provided in Figure 2.49 and 2.50. Figure 2.49 shows that most of the energy belongs to the third wavelet packet for the unstable time series. It is also seen that the spectrum of the third wavelet packet, the unstable time series, has a frequency content of around 1000 Hz. Thus the third wavelet packet can be selected as the informative packet for feature extraction.

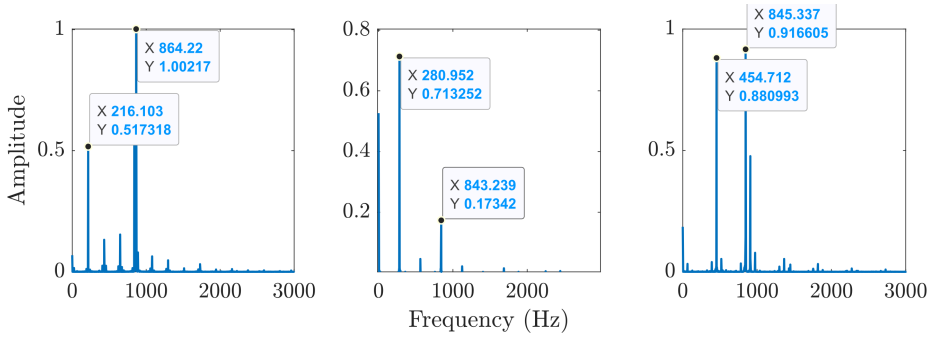


Figure 2.48: The spectrum of three different time series from milling experiment: (left) 13227 rpm, 2.54 mm depth of cut (doc), unstable,(mid) 16861 rpm, 1.905 mm doc, stable, (right) 27285 rpm, 1.905 doc, unstable.

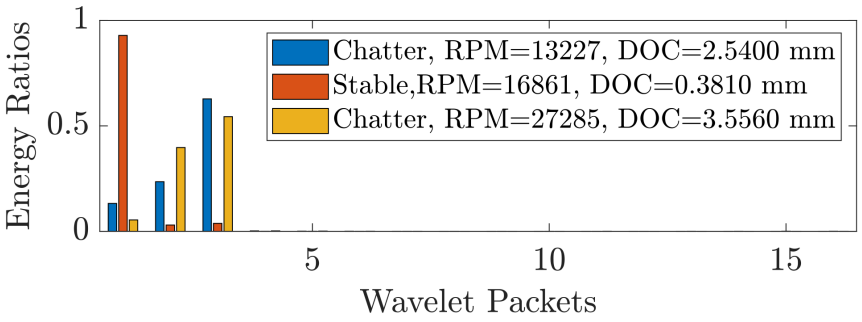


Figure 2.49: Energy ratio of the wavelet packets obtained from decomposition of the three time series whose spectrum is provided in Figure 2.48. (Blue bars) Milling - Unstable - RPM=13227 - DOC = 2.54mm. (Red bars) Milling - Stable - RPM=16861 - DOC = 0.38 mm. (Orange bars) Milling - Unstable - RPM=27285 - DOC = 3.556 mm.

2.7.1.2 Ensemble Empirical Mode Decomposition (EEMD)

This section provides the preprocessing of experimental milling signals using the EEMD approach explained in Section 2.3.2. The experimental signal was decomposed into IMFs, and the informative IMFs were selected to generate a feature matrix. The spectrum from the original signal and the IMFs was then compared to determine the overlap between them. The IMF with the largest overlap was selected as the informative IMF. Figure 2.51 provides an example for the selection of the informative IMF. The original time series has frequency content of around 1000 Hz, and the first two IMF are the candidates to be informative IMF. Since the spectrum of the first IMF overlaps with the original signal's spectrum, it is selected

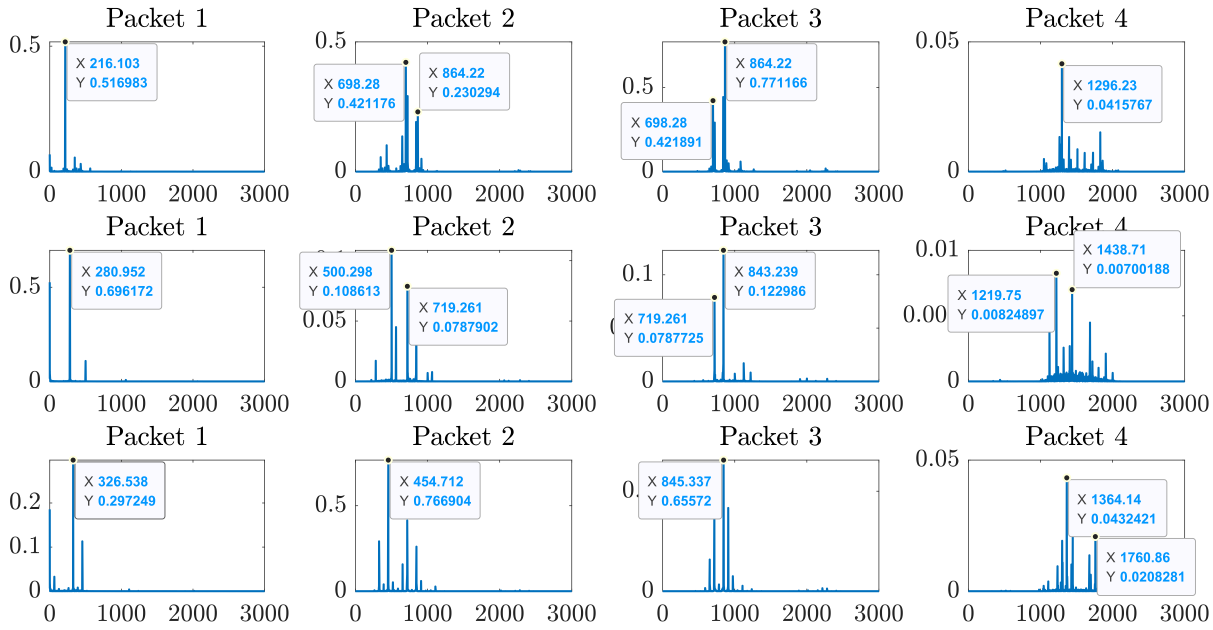


Figure 2.50: The spectrum of reconstructed signals from the first four wavelet packets of three different time series whose spectrum is shown in Figure 2.48. (First row) Milling, 13300 rpm, 2.54 mm depth of cut (doc), unstable, (second row) milling, 17300 rpm, 0.3810 mm doc, stable, and (last row) milling, 28000 rpm, 3.5560 mm doc, unstable.

as an informative IMF. Ideally, the spectrum of all signals and their decomposition should be checked to determine the informative IMF. However, this is a manually intensive and time-consuming process. Therefore, this process is repeated only for a couple of time series, and chose an informative IMF to use for all time series. Then, the selected informative IMF was used to compute the features given in Table. 2.4, and a feature matrix was generated as an input for a supervised classification algorithm.

2.7.1.3 Fast Fourier Transform (FFT), Power Spectral Density (PSD) and Auto-correlation Function (ACF)

This method computes the Fast Fourier Transform (FFT), Power Spectral Density (PSD), and Auto-correlation (ACF) for each downsampled data set. The next step was to find the significant peaks in these plots and use their x and y coordinates as a feature in the classification algorithm. Since built-in functions in computing software for peak finding can

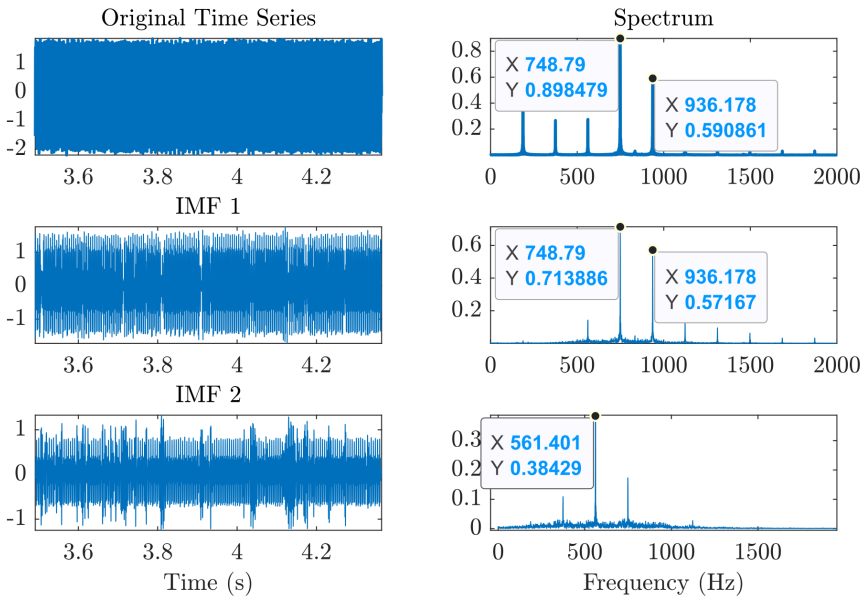


Figure 2.51: Intrinsic mode functions and their spectrum for the time series with 11210 rpm and 3.556 mm depth of cut from milling experiments.

result in incorrect peaks, two restriction parameters are used for peak selection that enabled us to find the true peaks. These parameters are minimum peak height (MPH) and minimum peak distance (MPD). The definition for minimum peak height is provided in Reference [55] as

$$MPH = y_{min} + \alpha(y_{max} - y_{min}), \tag{2.51}$$

where $\alpha \in [0,1]$, y_{min} and y_{max} correspond to 5^{th} and 95^{th} percentile of the amplitude of FFT/PSD/ACF plots. The α parameter is defined with respect to the peak amplitudes. Since auto-correlation function has negative amplitudes, the choice for α is chosen separately, while the same α value is used for FFT and PSD plots. In this implementation, respectively, α was 0.1 and 0.5 for FFT/PSD and ACF plots.

The second parameter, MPD, was defined by visual inspection on FFT/PSD/ACF plots of several time series. An example is provided in Figure 2.52. This figure shows the effect of the chosen MPD value on detecting the peaks in the FFT and ACF plots. The first two plots provide the spectrum of a time series and the peaks found by a peak detection algorithm

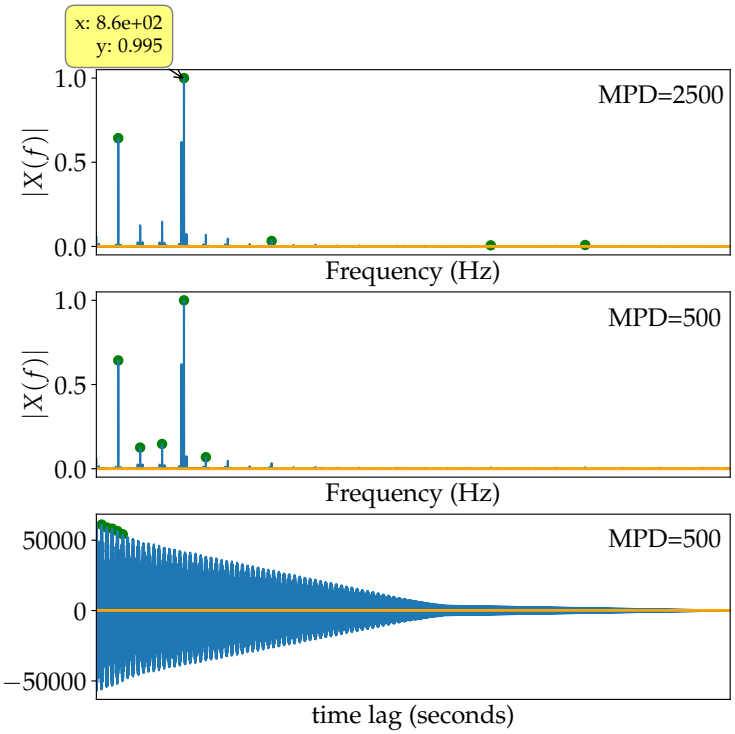


Figure 2.52: Effect of different MPD values on selected peaks in FFT and ACF plots of time series with RPM=13300 and DOC=2.54 mm from milling experiments. (Top) FFT plot and selected peaks with MPD=2500 (top) and MPD=500 (middle). Auto-correlation function with MPD=500 (bottom). Orange lines represent the MPH.

with two MPD values. It is seen that a smaller MPD value brings the selected peaks closer to each other and results in the detection of the true peaks. Therefore, MPD was chosen for FFT and PSD plots as 500. The same value was also used in the ACF function. After defining the two constraints for peak detection, MPD and MPH, the number of peaks to use to generate feature matrices is selected. In this implementation, coordinates of the first two peaks are used as features, and they were given to supervised classification algorithms.

2.7.2 Transfer Learning Results

This section describes the classification approach and the transfer learning details. As mentioned in Section A.1 and A.2, the turning data set contains four different cases, and the milling data does not have categorization. Therefore, the total number of combinations

between the cases of turning data and the milling data is 20. Classification is performed for all 20 combinations. Section 2.7.2.1 provides the results for the combination between cases of the turning data set, while Section 2.7.2.2 discusses the results for the combinations between turning and milling data set. In addition, the mild chatter cases are taken into account in turning data set as unstable while performing the classification. This is performed since the turning data is labeled in three classes (see Section A.1). All results provided in this section belong to two-class classification for both milling and turning data sets.

2.7.2.1 Results of Transfer Learning Applications Between the Overhang Distance Cases of Turning Data Set

The classification was performed with 10 realizations of training and test sets for each method. 67% of the training set and 70% of the test set were used to train and test the classifier, respectively. To be fair in comparing methods, the same training and test sets were used; they were generated with a set of predefined random state parameters. Support vector machine (SVM) with *rbf* kernel, logistic regression, random forest classifier with 100 estimators and a maximum depth of two for the trees, and gradient boosting algorithm were used to train and test a classifier. In addition to these classifiers, similarity measure based approach, DTW can only be used with the K-Nearest Neighbor (KNN) classifier since it uses a pairwise distance matrix between the time series. KNN is only used with the similarity-based approach. Predicted labels were used to compute the average and standard deviation of the accuracy and F1-score for training and test set separately. In addition to this, the methods are categorized into three groups: 1) Time-Frequency based approaches (WPT, EEMD, and FFT/PSD/ACF (FPA)), 2) TDA-based approach, and 3) Similarity measure (DTW). The results of these groups are compared. Features and classifiers used for each approach are given in Table 2.24. For more details about the features, one can refer to Refs. [3, 11, 9].

For each combination of the cases between turning data sets, figures which show the

Table 2.24: Features and classifiers used for three main category of approaches. SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting.

Category	Features	Classifiers
Time-Frequency-based	**WPT**: Mean, Standard Deviation, Root Mean Square (RMS), Peak, Skewness, Kurtosis, Crest Factor, Clearance Factor, Shape Factor, Impulse Factor, Mean Square Frequency, Standard Frequency, One Step Auto-Correlation Function, Frequency Center **EEMD:** Energy Ratio, Peak to Peak, Standard Deviation, RMS, Crest Factor, Skewness, Kurtosis **FF/PSD/ACF (FPA):** The coordinates of the peaks	SVM, LR, RF, GB
TDA-Based	Carlsson Coordinates, Persistence Landscapes, Persistence Images Template Functions	SVM, LR, RF, GB
Similarity measure (DTW)	Pairwise Distance Matrix	K-Nearest Neighbor

accuracy of each feature extraction method have been provided for the classifiers mentioned above. These plots are provided in Figs. B.3-B.7 in the Appendix. However, these plots can only compare the methods within the same application of transfer learning. Therefore, a summary plot is provided in Figure 2.53. It provides the highest accuracies obtained out of four classifiers for all methods except DTW, while it shows the highest score out of all KNN, where K=(1,...,5), for DTW. It can be seen that the time-frequency-based methods, such as WPT, EEMD, and FPA, are the methods that give the highest score when training and testing are performed between the overhang cases of the turning data set. However, WPT outperforms other approaches in most of the applications in the group of time-frequency-based approaches. On the other hand, the TDA-based approach and DTW have the highest accuracy in a few applications. For TDA-based approaches, Carlsson Coordinates (TDA-CC) performs better than other featurization techniques within the group. It is not easy to distinguish each result in Figure 2.53. Therefore, WPT, TDA-CC, and DTW are selected to summarize their results for different applications of transfer learning between turning data set cases and presented the results in Figure 2.54.

Figure 2.54 contains only the highest scores of the selected approaches and the ones that are in the error band of the highest score. Each color represents a method; the best results and the ones in the error band are represented with two different hatches on the bar plots. The bars with the '○' hatch are the methods with the highest accuracy, and the '/' hatch shows the methods that are in the error band. Using Figure 2.54, we can define how many

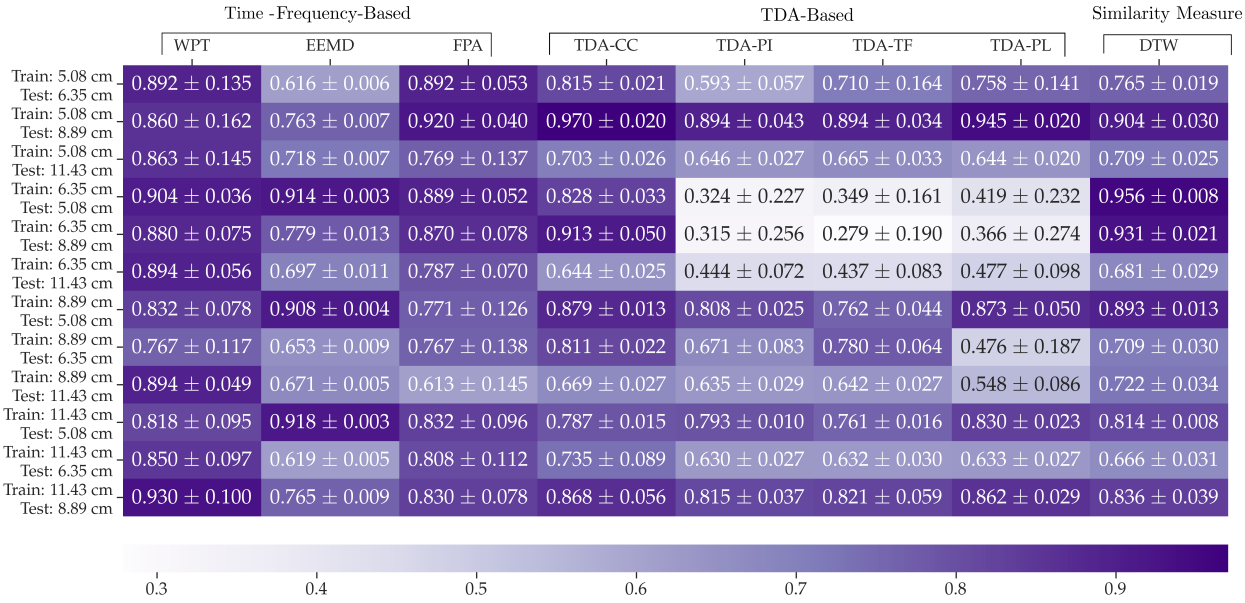


Figure 2.53: The highest accuracy out of four different classifiers (or out of selected numbers of nearest neighbor for DTW) for each approach used in transfer learning applications between overhang distance cases of turning experiments.

times a group of methods is the best method (BM) or the method in the error band (MIEB), and these numbers are given in Table 2.25. It is seen that the frequency-based approach (WPT) has the highest score in 7 out of 12 transfer learning applications between the cases of turning data set, and it is not in the error band when the TDA-based approach and DTW provide the highest score. On the other hand, the TDA based-approach and DTW provide the highest score two times and three times, respectively. The TDA-based method is in the error band of the highest accuracy in 4 out 12 applications, while this number is three for DTW. It is also worth noting that WPT results provided in Figure2.54 have a larger deviation compared to DTW and TDA-based approaches, even though WPT provides the highest accuracies in most of the applications.

Another criterion to compare the performance of the methods is to check the F1 score. The F1-score is computed for all transfer learning applications and each method. Then, the highest F1 scores out of all classifiers were chosen, and the summary plots are given in Figure 2.55 and 2.56. In addition, the counting is performed for the number of best methods

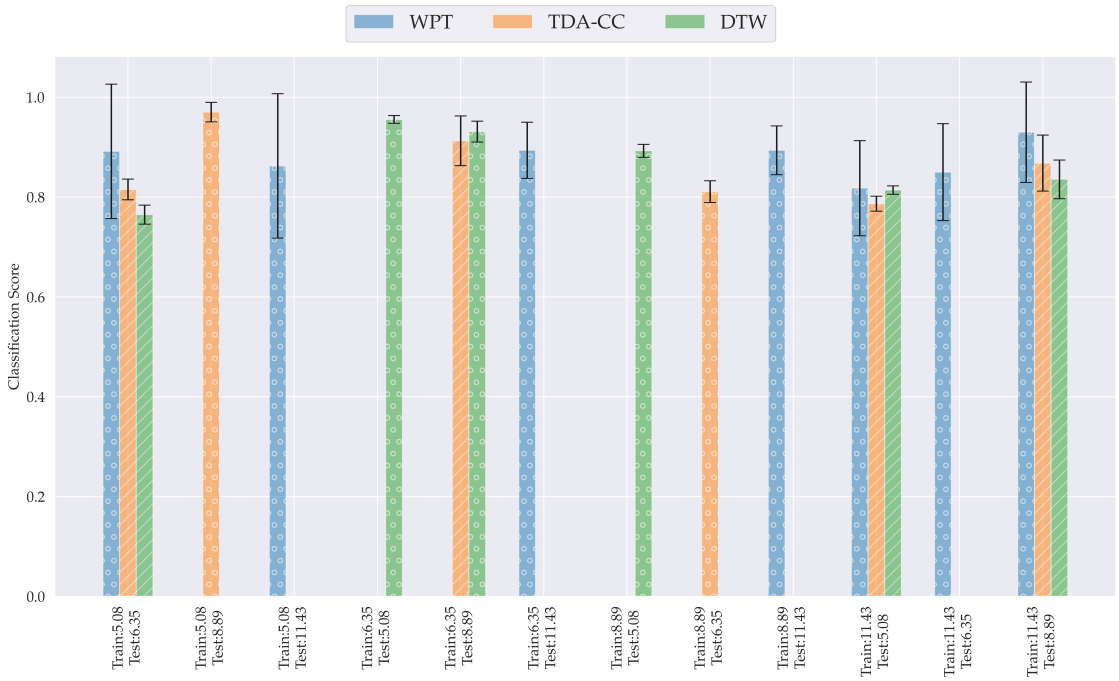


Figure 2.54: The classification results are obtained from the selected methods when training and testing are performed between the overhang distance cases of the turning data set. The selected methods that give the highest accuracy are represented with the '○' bar hatch, and the ones that are in the error band of the highest accuracy are shown with the '/' bar hatch. One approach is selected from each category of the methods, and these are Wavelet Packet Transform (WPT), Carlsson Coordinates (TDA-CC), and Dynamic Time Warping (DTW).

Table 2.25: The number of times when a selected method gives the highest accuracy out of 12 different applications between the cases of turning data set is denoted with BM. The number of times when a method is in the error band of the highest accuracy is denoted with MIEB. These two numbers are provided for accuracy and F1-score.

	Accuracy		F1-Score	
Method	BM	MIEB	BM	MIEB
Time - Frequency-based (WPT)	7	0	9	1
TDA-based (TDA-CC)	2	4	3	0
Similarity Measure (DTW)	3	3	0	0

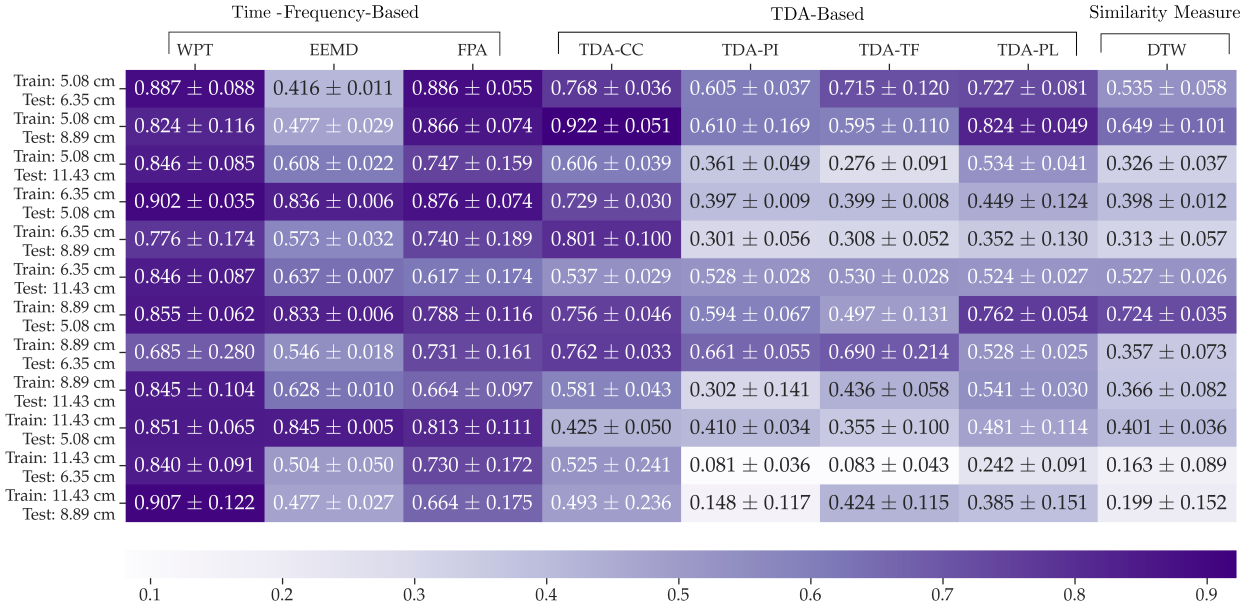


Figure 2.55: The highest F1-Score out of four different classifiers (or out of selected numbers of nearest neighbor for DTW) for each approach used in transfer learning applications between overhang distance cases of turning experiments.

and the methods that are in the error band as in the case of accuracy, and these numbers are reported in Table 2.25. It is seen that the performance of the time-frequency-based approach is better since it has the highest F1 score in 9 out of 12 cases of the applications. The TDA-based approach provides the best F1 score three times. WPT again provides the best results with larger deviations compared to the TDA-based approach (see Figure 2.56).

2.7.2.2 Results of Transfer Learning Applications Between Turning and Milling Data Sets

There are eight different permutations when transfer learning is applied between the turning and milling operations. The classification scores for four different classifiers are provided in Figure B.8- B.10. However, these plots include detailed results within the same application of transfer learning. Therefore, summary plots are provided for these permutations in Figure 2.57. It is seen that FFT/PSD/ACF (FPA) gives the highest accuracies between time-frequency-based approaches in most transfer learning applications. The results

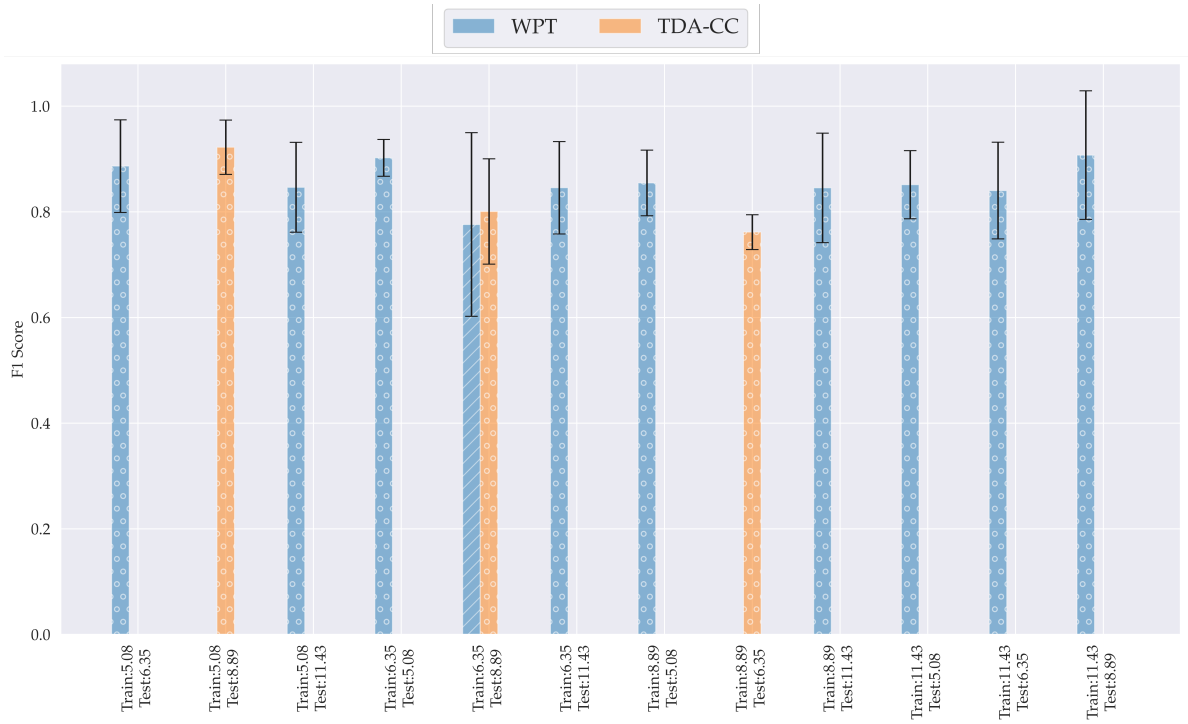


Figure 2.56: F1 scores obtained from the selected methods when training and testing are performed between the overhang distance cases of the turning data set. The selected methods that give the highest accuracy are represented with the '○' bar hatch, and the ones that are in the error band of the highest accuracy are shown with the '/' bar hatch. One approach is selected from each category of the methods, and these are Wavelet Packet Transform (WPT), Carlsson Coordinates (TDA-CC), and Dynamic Time Warping (DTW).

obtained with TDA-based approaches are similar to each other, especially for Carlsson Coordinates (TDA-CC) and Template Functions (TDA-TF). Since the TDA-CC provides the highest accuracy when training is performed on a 6.35 cm overhang distance of the turning data set and testing is performed on the milling data set, TDA-CC is chosen to represent the TDA-based approaches. Accordingly, FPA, TDA-CC, and DTW are compared with respect to classification scores in Figure 2.58. Similar figures for F1-Score can also be found in Figure B.11 and B.12.

In Reference [9, 55], the authors mentioned several drawbacks of the time-frequency-based approaches. One of the main drawbacks of these methods is that they require checking the frequency spectrum of each time series manually to decide informative decomposition

for WPT and EEMD or the restriction parameters for FPA. In this study, this manual preprocessing is performed only for a couple of time series. This process gets cumbersome as the size of the data set increases. In a real-time application, when a new time series is introduced to a classifier, the frequency spectrum of it and its reconstructed time series obtained from the wavelet packets need to be investigated to find the decomposition whose spectrum has the largest overlap with the signal's spectrum. On the other hand, the processes for the TDA-based approach and the DTW do not require any parameter selection, and all steps can be completed autonomously.

Based on Figure 2.58 and B.12, Table 2.26 is generated and it shows the number of times when a selected method gives the highest accuracy (BM) or it is in the error band of the highest accuracy (MIEB). If the accuracy is considered as the main criterion, the DTW method provides the highest accuracy in three out of eight applications, and it is in the error band of the highest accuracy in two applications. In addition, the results of the TDA-based approach are in the error band in two out of eight applications. Considering the drawbacks of the frequency-based approach and deviations of the results of the frequency-based approach, DTW and TDA-based approaches can be preferred when transfer learning is applied between different machining operations.

Table 2.26: The number of times when a selected method gives the highest accuracy out of 8 different applications between the cases of turning data set and the milling data set is denoted with BM. The number of times when a method is in the error band of the highest accuracy is denoted with MIEB. These two numbers are provided for accuracy and F1-score.

	Accuracy		F1-Score	
Method	BM	MIEB	BM	MIEB
Time - Frequency-based (FPA)	4	1	6	0
TDA-based (TDA-CC)	1	2	0	4
Similarity Measure (DTW)	3	2	2	1

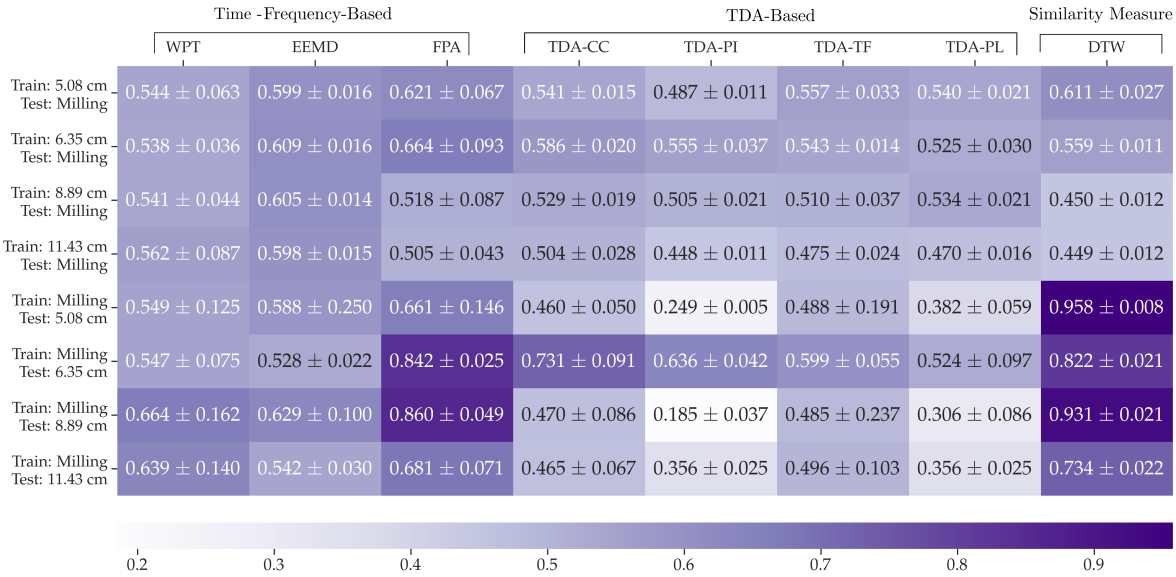


Figure 2.57: The highest accuracy out of four different classifiers (or out of selected numbers of nearest neighbor for DTW) for each approach used in transfer learning applications between overhang distance cases of turning and milling experiments.

2.7.2.3 Transfer Learning Using Deep Learning

In addition to traditional machine learning algorithms, Artificial Neural Networks(ANNs) are also utilized to test the performance of several approaches. Deep learning frameworks can learn from raw data sets without the need for feature extraction. However, Zhoa et. al state that inadequate size of data set, noisy raw signal, and complex machining operations make it necessary to preprocess data before feeding it into deep learning algoritms [160]. Therefore, some of the features extracted from TDA-based approaches are used to apply deep learning in transfer learning. Some of the studies in the literature (see Refs. [92, 91]) trained the deep learning algorithms using the simulation data set to eliminate the need for an extensive amount of experimental data set to train the classifier. However, in this work, only the existing experimental data and the features extracted from them are only used to train deep learning algorithms to compare the results to traditional machine learning algorithms. One should be aware of the fact that more observation is needed to have a fair comparison between deep learning-based transfer learning and traditional machine learning-

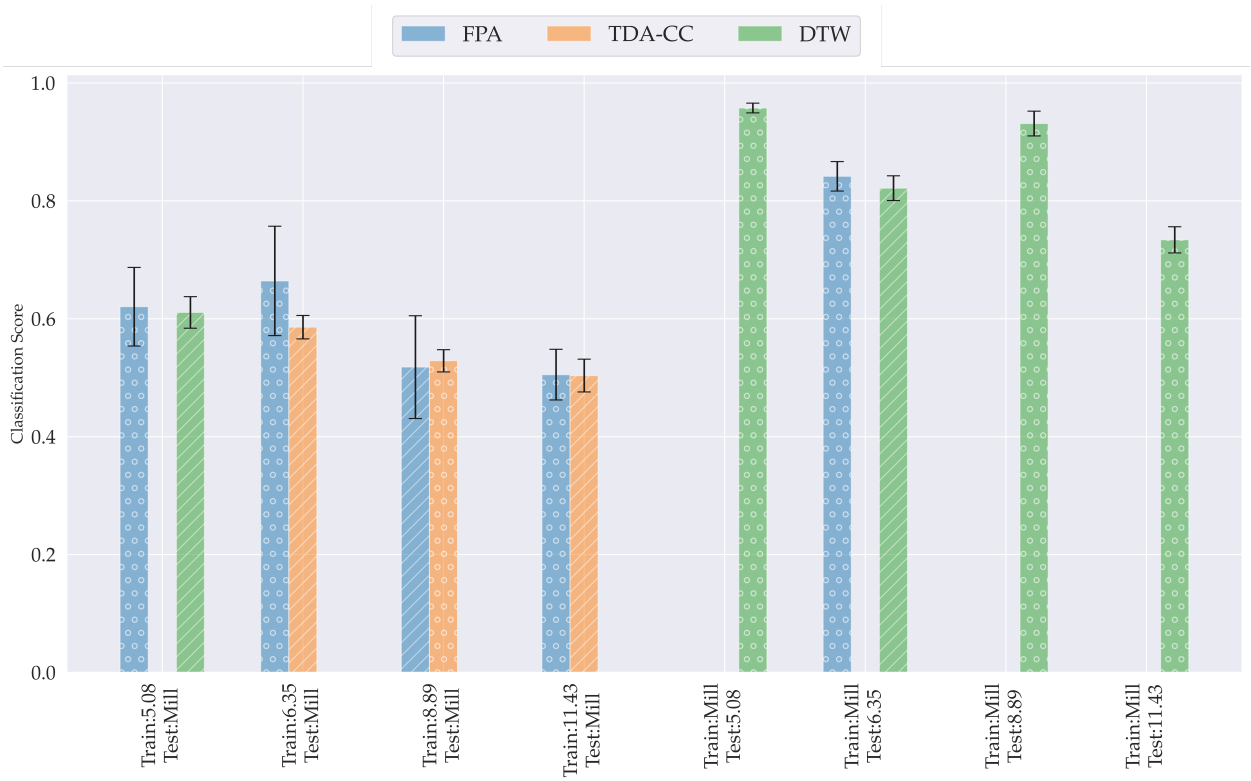


Figure 2.58: The classification results obtained from the selected methods when training and testing are performed between the overhang distance cases of the turning data set and the milling data set. The selected methods that give the highest accuracy are represented with the 'o' bar hatch, and the ones that are in the error band of the highest accuracy are shown with the '/' bar hatch. One approach is selected from each category of the methods, and these are FPA, TDA-CC, and DTW.

based transfer learning. Since the raw experimental signals are not split into small pieces for Time-Frequency based approaches, there are fewer observations for these approaches. Hence, the features extracted using Time-Frequency based approaches are not utilized for training deep learning algorithms.

The ANN structure used in this work has one input, three hidden, and one output layer. The number of inputs fed into the input layer is based on the number of features extracted from TDA-based approaches. For instance, Carlsson Coordinates can provide five features for each persistence diagram, so the number of inputs will be five for this approach. The first and last hidden layers have 25 neurons, while the second hidden layer has 12 neurons.

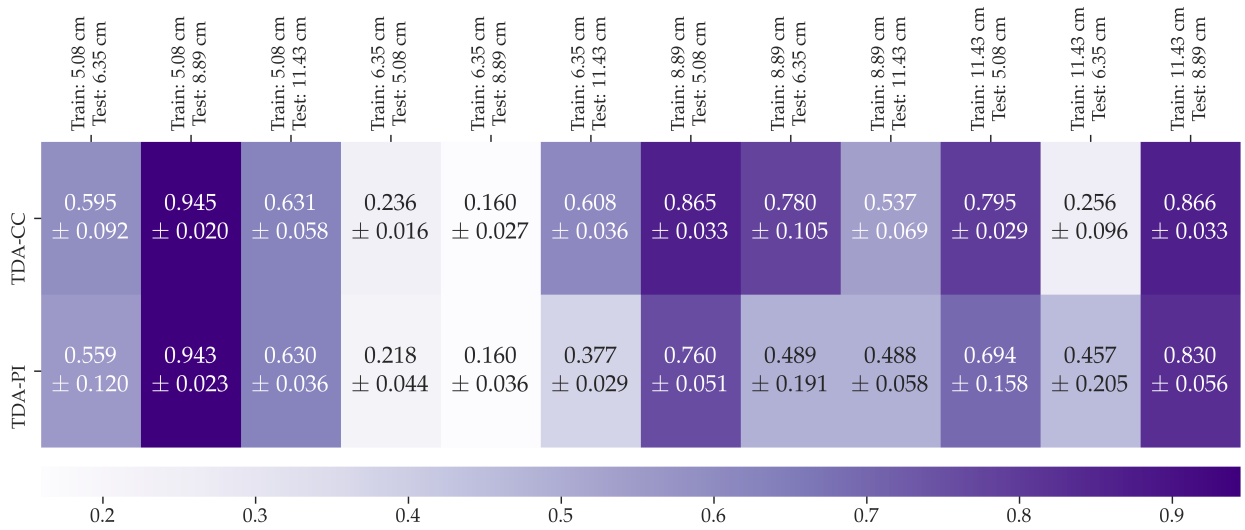


Figure 2.59: The classification accuracies obtained using Carlsson Coordinates and Persistence Images features with ANN algorithms for the transfer learning between the cases of turning experiments.

The hyperbolic tangent function is used as an activation function in all layers except the output layer. Since the classification output is binary, the sigmoid function is chosen as the activation function in the output layer. Adam optimization algorithm and binary cross-entropy loss functions are used to compile the ANN. Epoch number and the batch size to update the weights of the fully connected layers are selected as 100 and 5, respectively. There are 12 permutations between the overhang distance cases of the turning data set and eight permutations between the turning and milling data set. 67% of the training data set is used as the training set, and 70% of the test set data set is used to test the ANNs. Train-test split is repeated for 10 different pre-defined random state numbers, and the mean accuracy and standard deviation are computed out of these 10 realizations. The results for transfer learning applications between the overhang distance cases of the turning data set are provided in Figure 2.59, while Figure 2.60 provides the accuracies with error bands for the transfer learning between the milling and turning data set.

From Figure 2.59 and 2.53, it is seen that traditional machine learning algorithms provide better accuracies compared to deep learning in 11 out of 12 different transfer learning

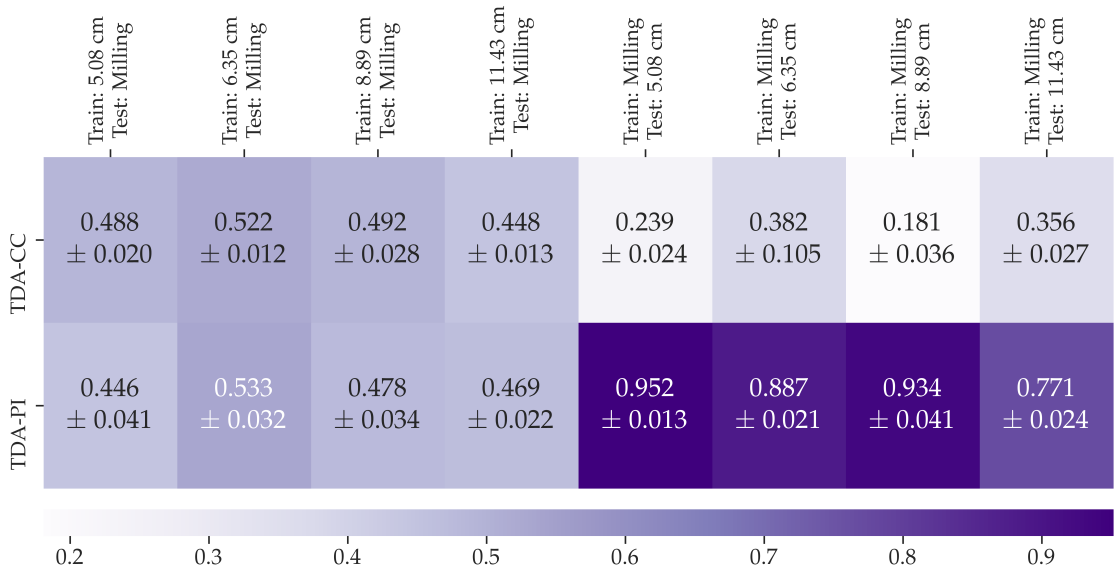


Figure 2.60: The classification accuracies obtained using Carlsson Coordinates and Persistence Images features with ANN algorithms for the transfer learning between the milling and turning experiments.

applications for Carlsson Coordinates, while deep learning is outperformed in 9 out of 12 applications of transfer learning between the Looking into Figure 2.60 and 2.57, traditional machine learning algorithms outperform deep Learning in all applications of transfer learning for Carlsson Coordinates. However, deep learning outperforms traditional machine learning in 4 out of 8 applications of transfer learning between the milling and turning data set using persistence images. Overall, it is obvious that the amount of experimental data set fed to deep learning is insufficient, and this leads to poor performance against the traditional machine learning algorithms. In addition, hyperparameter tuning is not performed for ANNs in this study. This could also be another reason for the poor performance of deep learning.

2.8 Conclusion

This chapter studied two advanced chatter detection methods, i.e., the Wavelet Packet Transform (WPT) and the Ensemble Empirical Mode Decomposition (EEMD) with Recursive Feature Elimination (RFE) and traditional feature extraction using FFT/PSA/ACF,

and presented the DTW and TDA-based approaches for chatter diagnosis. These approaches have been used for the classification of recorded acceleration signals from a turning process into chatter-free cutting or chattering motion. They are not used only to classify measured test data with the same cutting conditions as used in the training phase but also for transfer learning, which means that the test data originates from a cutting process with different cutting conditions. In particular, the chatter frequencies between the training data and the test data differ significantly.

For WPT and EEMD approaches, the results in Table 2.5 show that WPT has the highest accuracies for three cutting configurations when the classifier is trained and tested on the same data set, while EEMD provides the best results for one cutting configuration. In addition, training different classifiers other than SVM leads to increase in mean accuracies for both method as shown in Table 2.6 and 2.7. These two tables are evidence that WPT performance decreases as the level of transform increases. In addition to accuracy comparisons, the overall runtime of each method is also recorded. Table 2.5 shows that WPT has the fastest runtime, while the EEMD method clocks the longest runtime. This slowdown is mostly related to the computation of the ensemble of IMFs and can be reduced by changing the ensemble parameters and optimizing the code.

There are two main drawbacks of these two methods. 1) the WPT featurization process is cumbersome since it requires taking the WPT of the signal, investigating the packets that contain the chatter frequencies, and then choosing the packet that has a considerably high energy ratio, and that includes chatter frequency. Once these packets are found, they are fixed for the investigated process and are used for chatter classification. However, inherent to this process is the a priori identification of chatter frequency and the assumption that the chosen packets (referred to as the informative packets) will always contain it. This is a limitation since (a) it requires highly skilled users for analyzing the signal and extracting the informative packets, and (b) the chatter frequency band can move during the cutting process, which will yield the informative packets ineffective for chatter classification. Further,

Section 2.3.1 points out that the informative packets are not necessarily the ones with the higher energy ratio. This makes automating the feature selection process more difficult in the WPT approach. The EEMD also suffers some of these drawbacks since the process for choosing the informative IMFs and the informative packets in WPT is quite similar. The second drawback is that 2) it is not always possible to differentiate between intermediate and full chatter. Specifically, although the intermediate chatter time series (Figure 2.8c) and the chatter time series (Figure 2.8e) are visually very different in the time domain, their energy content shown in the top graph of Figure 2.9 can be too close to distinguish between the two cases.

In this study, traditional signal processing tools (FFT, PSD, and ACF) are also applied for feature extraction from cutting signals obtained from a turning experiment. The results are compared to the ones obtained from two widely used chatter detection methods: WPT and EEMD. The results show that the classification results for FFT/PSD/ACF with feature ranking can provide higher test set accuracies for some of the classifiers, namely Random Forest Classification and Gradient Boosting. If the overall best score for each overhang length distance of the turning data set is compared, it is seen that the FFT/PSD/ACF based methods provide the best results. Despite their shortcomings, traditional signal processing methods can yield highly-tuned classifiers with superior accuracy. This makes these methods suitable for manufacturing processes whose parameters are not expected to drift too much in comparison to the training data. In contrast, if the parameters of the underlying process are expected to shift significantly, then features based on FFT/PSD/ACF should be used with caution.

The first novel method presented in this chapter for chatter detection combines similarity measures of time series via Dynamic Time Warping (DTW) with machine learning. In this approach, the similarity of different time series is measured using their DTW distance, and any incoming data stream is then classified using the KNN algorithm. The classification accuracy of the DTW approach is tested using a set of turning experiments with four

different tool overhang lengths, and the resulting accuracy is compared to two widely used methods: the Wavelet Packet Transform (WPT) and the Empirical Mode Decomposition, as well as to the second novel method based on Topological Data Analysis (TDA). The results in Table 2.11 show that the DTW's classification accuracy matches or exceeds those of existing methods for two out of four different overhang cases. This indicates that temporal features extracted using DTW are effective markers for detecting chatter in cutting processes. Topological Data Analysis (TDA) methods results are also close to the ones for similarity measures; however, one advantage of the DTW approach in comparison to TDA-based tools is that it does not require embedding the data into a point cloud, hence avoiding the complications associated with choosing appropriate embedding parameters.

The DTW approach can distinguish stable and unstable cases from each as evidenced by the heat map of the average distances between time series with different labels (see Figure 2.24), 2.25 and B.1. The DTW approach also successfully distinguishes between chatter and intermediate chatter, as shown in Table 2.12. These comparisons are difficult or impossible using only frequency domain features because the frequency content in these two cases is too similar while the time domain signals are different, as evidenced by Figure A.4 and the heat maps shown in Figure 2.24 and B.1.

In addition, the combination of the DTW approach with the AESA algorithm provides a significant decrease in the number of DTW computations, thus reducing the classification time of a single test sample to less than two seconds for three out of four overhang distances. Depending on the user choice of H, one can obtain the results even faster with the cost of loss in classification accuracy as seen from Figure2.27. Therefore, there is a trade-off between reduction in the number of distance computation and classification performance. In contrast to the AESA algorithm, parallel computing has the capability of completing the classification of single time series in about 1.5 seconds if ideal conditions such as zero queue time and enough resources–the number of processors/cores and RAM capacity–are obtained to be able to run all jobs simultaneously. Therefore, it is observed that both AESA and

parallel computing can be combined with similarity measures for real-time online chatter detection applications. It is also noted that Table 2.13 does not include the time required for the manual preprocessing in the WPT and EEMD methods for choosing informative packets or decompositions. The actual time for these two methods is larger than the ones provided in the table, depending on the number of the investigated time series and the skill of the person performing the preprocessing. This is because WPT and EEMD require a process for checking the frequency spectrum of the times series and examining the energy ratio of the wavelet packets of the time series. Furthermore, whereas the WPT algorithms are highly optimized, the Python scripts that are used in this study for computing the DTW have little to no optimization. This study hypothesizes that further optimization using, for example, the ideas in [137] and combining the DTW approach with AESA and parallel computing will speed up the runtime for the similarity measures making them a viable option for on-machine chatter detection.

In this study, another novel approach is presented for chatter identification and classification based on featurizing the time series of the cutting process using its topological features. In contrast to the WPT and EEMD methods, this study uses persistent 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, the leading tools are investigated for feature extraction from persistence diagrams. The featurization methods are based on persistence landscapes, persistence images, Carlsson Coordinates, a kernel method, template functions, and persistence paths' signatures. The resulting features are then combined with several classification algorithms 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 featurization

method and for each cutting configuration of the turning data set.

Tables 2.23 summarizes the classification accuracy of all the TDA-based tools as well as their WPT and EEMD counterparts. In terms of the classification accuracy across the different cutting configurations, the Carlsson coordinates and persistence landscapes approaches yield the best accuracies for two cases, and the former has the smallest runtime in comparison to the other TDA tools. On the other hand, in the remaining cases, WPT yielded the best accuracies. Table 2.23 shows that WPT yields an accuracy of 100% (with a standard deviation of zero) for the 6.35 cm (2.5 inch) overhang case; however, as pointed out in Section 2.6.7.2 and Table A.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. Specifically, Table 2.23 shows that for the 6.35 cm (2.5 inch) case, persistence landscapes and Carlsson coordinates yield accuracies that are off by 13.7%, and 10.7%, respectively, from the WPT result. Similarly, for the 11.43 cm (4.5 inch) case, Carlsson Coordinates and persistence images are within 13.3% and 13% of the EEMD result. As mentioned in Section 2.6.7.2, persistence diagrams of stable and unstable time series of 11.43 cm (4.5 inch) case are similar in contrast to other overhang distance cases, and this is why there is a large accuracy difference between the results of TDA-based approaches and the WPT approach. For the 5.08 cm (2 inch) and 8.89 cm (3.5 inch) overhang case, persistence landscapes and Carlsson Coordinates yield the highest accuracy scoring 96.8% and 95.7%, respectively, with tight error bounds that do not enclose the WPT accuracies.

Runtime comparisons in Table 2.20 show a dramatic decrease in persistence diagrams computation by at least 82% when they are computed in parallel. Table 2.22 shows that WPT is the fastest followed by EEMD; however, the reported time for the TDA-based approach is comparable to the ones of EEMD. The runtime can be reduced for computing persistence diagrams for a single time series to less than 1 seconds using Greedy permutation and the Bézier curve approximation method in combination with parallel computing. Figure2.43 also

shows that the Bézier curve approximation method results in larger accuracies compared to Greedy permutation. Therefore, the combination of parallel computing and the Bézier curve approximation makes the TDA-based approaches a feasible option for online chatter detection.

This study shows 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 preprocessing is needed before extracting and using the features in the persistence diagram. It is also noted 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.

Another main contribution of this chapter is to analyze the performance of the traditional and novel chatter diagnosis approach between two different cutting operations. The features extracted from turning and milling cutting signals are used to transfer knowledge from turning experiments to milling experiments or vice versa. The highest scores obtained from the transfer learning applications, between the cases of the turning data, were between 80% and 100%, while that drops to 60% when the classifier is trained on turning data and tested on the milling data. A period-doubling bifurcation was observed in 19 out of 318 time series of milling data, and a Hopf bifurcation was observed in the rest of the unstable cases of milling data. On the other hand, the turning data only contains the Hopf bifurcation. When the classifier is trained on the turning data set, the model is not trained to recognize the descriptors of period-doubling bifurcation, so it performs poorly when it is tested on milling data. On the other hand, a classifier is trained with both features of Hopf and period-doubling when the milling data is used as the training set. This explains why training on milling data and testing on turning data sets perform better. In addition, the mathematical model for the milling process has time-varying coefficients, while the turning process has an autonomous system. Since the coefficients are constant in turning processes, this can

lead to misclassification when the classifier is tested on milling processes with time-varying coefficients.

This study compared the performance of feature extraction methods from established methods alongside those recently proposed in the literature. Turning and milling data sets were used to evaluate the performance of each method. The size of the training sets and the test sets were kept the same for each method. Since the training set data and test data are different from each other, 67% and 70% of the training set and test data are used to train and test a classifier, respectively. Ten random state numbers were used to generate training and test splits, and these were used to train and test a classifier for each method. The average and standard deviation of the 10 realizations were computed, and the final results were reported. This has been repeated for all 20 combinations between the milling data and overhang cases of turning data.

Two types of figures are provided for each comparison criterion to compare the results. Figure 2.53, 2.54, 2.57, and 2.58 were obtained when the criterion was accuracy, while Figure 2.55, 2.56, B.11, and B.12 were given for the F1-score. It can be seen that the time-frequency-based approaches give the highest accuracy in most of the applications of transfer learning with larger deviations in comparison to the TDA-based approach and DTW. When only the transfer learning between the milling and turning data sets is considered, it is seen that the accuracies obtained from DTW can be as high as 96% while the time-frequency-based approaches can be up to 86% (see Figure 2.57). For the same cases of transfer learning, the highest score obtained from TDA is 73% (see Figure 2.57). For the transfer learning applications where the classifier is trained and tested between the cases of turning, the time-frequency-based approach has the highest accuracy of 93%; the best score for the TDA approach and DTW are 97% and 96%, respectively (see Figure 2.53). The results of traditional machine learning algorithms are also compared to the ones obtained from ANNs. It is seen that insufficient experimental data set leads to poor results against traditional machine learning approaches. The small size of the experimental data set also

prevents comparing different techniques to each other using deep learning algorithms. In this work, only several TDA-based approaches are compared to each other. Using synthetic data sets generated using the analytical model of milling and turning operations can allow one to further extend the comparison of more approaches using deep learning frameworks in the future.

In summary, the TDA-based and DTW approaches can provide accuracies and F1 scores as high as the time-frequency-based methods. DTW outperforms all other methods when training on the milling data set and testing on the turning data set. In addition, the TDA-based approach and DTW can be applied without needing manual preprocessing. All of the steps in their pipeline can be completed automatically. Therefore, these approaches may be preferred over the time-frequency-based approaches in either real-time or in fully automated chatter detection schemes. It is worth noting that this study does not perform any optimization on hyperparameters of the traditional machine learning and deep learning algorithms. Thus future studies should also consider the effect of hyperparameter tuning.

CHAPTER 3

DATA DRIVEN PARAMETER IDENTIFICATION FOR A CHAOTIC PENDULUM WITH VARIABLE INTERACTION POTENTIAL

3.1 Introduction

Machine learning has led to many advances in the estimation of governing equations of physical systems from sensor signals. This is useful in the engineering processes where in the design phase, an abstraction of the physical system is used to write the governing differential equations. However, the validity of the assumptions used in these models is not tested until the part is manufactured and utilized in applications. This necessitates a data-driven approach for studying the true underlying model of the system versus the idealized model utilized in the design phase.

There are many studies based on data-driven model identification including eigensystem realization [173], equation-free modeling [174], empirical dynamic modeling [175], modeling emergent behavior [176], automated inference of dynamics [177, 178, 179], nonlinear Laplacian spectral analysis [180], symbolic regression [181, 182, 183], Kalman Filter [184], neural networks [185, 186, 187] and time series [188] for model identification. These methods have some drawbacks. For example, the symbolic regression is computationally expensive, and it suffers from overfitting problems as the system complexity increases [189]. Furthermore, neural network-based methods act as a black box and require large amount of data. It is also difficult to assign a physical meaning to these black boxes.

Another widely used method is sparse regression. The most commonly used sparse regression algorithms are LASSO [190, 191] and Ridge regression [191]. Schaffer et al. employed sparse regression to approximate coefficients of nonlinear differential equations [192]. Tran and Ward used sparse regression, splitting optimization, and compressed sensing to identify governing equations [193]. Other recent studies on the sparsity of nonlinear systems can be

found in [194, 195, 196]. In addition, Rey et al. implemented time-delay embedding for model identification when measurements for some state variables are not available [197]. Wang et al. provided an overview of the methods for data-driven identification of complex nonlinear systems [198].

Brunton et al. introduced SINDy for the identification of model parameters of nonlinear systems [189]. SINDy is composed of three parts: 1) a feature library that includes a complete basis of possible terms in the system equation. Since the feature library is composed of all combinations of the functions, its size can become significantly large even if the number of the chosen candidate functions is small. 2) An estimate of the derivatives from the experimental measurements, and 3) application of sparse regression to determine the coefficients of the governing equations. To reduce the size of the feature library, Rudy et al. [199] and Schmidt et al. [182] used Pareto Front analysis [200] to reduce the number of candidate models.

Other extensions of SINDy include using it for feedback control [201], and Model Predictive Control (MPC) [202]. The latter work showed that SINDy-MPC outperforms neural-network-based methods in terms of robustness and performance. Recently, SINDy was further extended to stochastic dynamical systems [203].

Most of the nonlinear models to which SINDy applies are composed of polynomial, trigonometric, and rational expressions. Generally, a combination of various polynomial functions and sinusoidal functions is used in the feature library. However, one of the limitations of SINDy is that it performs poorly when using polynomial and sinusoidal terms to approximate rational expressions in the governing equations [189]. Despite the large number of publications on SINDy, its applicability to experimental data from nonlinear mechanical systems has not been widely studied (see Refs. [204, 205] for some examples). Therefore, more testing is needed before this method can be reliably used as part of the engineering design cycle. This study takes a step in that direction and highlights some admonitions that need to be considered when using SINDy on actual systems with unknown models. For example, this study reveals the sensitivity of SINDy to the derivative approximation and

shows that there is a limit on the time horizon over which the model yields a good fit. It is also shown that the fitted model matches for the whole time range if the exact derivative is known, further illuminating the important role of an accurate derivative estimation in SINDy.

There are many methods for derivative estimation from noisy signals [206], and SINDy utilizes the Total Variation Regularization (TVR) method [207, 208]. However, TVR is highly dependent on the selection of two parameters, α and ϵ, whose values are positive real numbers, thus making TVR difficult to tune. This chapter shows that the resulting coefficients obtained with SINDy for two example nonlinear systems are quite sensitive to the selection of these two parameters. TVR is replaced by several methods which are comparatively easier to tune, such as cubic spline approximation, Savitzky-Golay smoothing, Gaussian moving average approximation, and convolution smoothing. Other sparse regression algorithms such as LASSO and ridge regression are also used. SINDy is applied to two simulated systems: the simple nonlinear pendulum and a model of a more complicated chaotic pendulum with varying interaction potential. The latter model is based on an experiment that Mork et al. [209] designed and built. In addition to numerical simulation, the rotational angle from the chaotic physical pendulum is used as input to SINDy to see if the identified model is similar to the theoretical one used during the design stage.

The results show that the estimated nonzero coefficients grow rapidly as the TVR parameter α varies while keeping its other parameter ϵ constant. In fact, due to the large support of α and ϵ, it is very difficult to find a parameter combination that yields the most accurate derivative estimate. Therefore, this chapter examines and suggests several other derivative estimation methods whose parameters can be more easily tuned. This study shows that some of these methods can outperform TVR in terms of coefficient estimation and fit the response data. However, even if the estimated response matches the actual response, the results show that the estimated coefficients can significantly differ from the true model. The robustness of all methods to noise is also compared, and an open repository for replicating

the experiment [209] is provided.

This chapter is organized as follows. Section 3.2 explains the modeling and the experimental setup for the chaotic pendulum. Section 3.3 explains the SINDy algorithm with detailed information on sparse regression. The results are provided in Section 3.4, while the concluding remarks are included in Section 3.5.

3.2 Modeling and Experimental Setup

This section describes the model and the experimental setup for the chaotic pendulum. This experiment is for a complicated nonlinear pendulum whose interaction potential and initial conditions could be accurately controlled. While the idea for the experimental apparatus was inspired by the work in [210], the details in that work were not sufficient to reproduce the original design. Therefore, the apparatus is re-designed and built, keeping the main outline of the original design—see [209] for the design documentation and CAD files. Specifically, the setup is composed of two towers: right and left. The right tower is placed on a mini labjack (Table 3.1-No 1). A DC motor (Table 3.1-No 3) is connected on top of a non-magnetic linear ball slider (Table 3.1-No 2). A disk with a radius of 4.445 cm (1.75 in.) is connected to the DC motor via a coupler and a shaft. Magnets (Table 3.1-No 4) with alternating polarity are attached to the circumference of the disk. A stepper motor (Table 3.1-No 5) drives the DC motor and the disk with magnets on the slider with a scotch yoke mechanism. The DC motor rotates at high speed, thus causing the rotating and translating magnets to apply an eddy force to the aluminum disk on the left tower. The aluminum disk is connected to an aluminum shaft of 6.096 cm (2.4 in) diameter. This shaft is fixed to the left tower with two bearings (Table 3.1-No 7) and is connected to a simple pendulum with an end magnet (Table 3.1-No 4). A photo-interrupter (Table 3.1-No 6) checks the speed of the stepper motor. Another magnet (3.1-No 4) can be placed under the single pendulum as an option. The rotational speed of the aluminum disk is measured with an optical encoder (Table 3.1-No 8) and a rotary disk (Table 3.1-No 9). Figure 3.1 shows each part with labels

that match Table 3.1.

Table 3.1: Part list for the experimental setup.

Item Number	Part Name
1	THORLABS L200 - 10.16 cm x 7.62 cm (4 in. x 3 in.) Lab jack, 2.62 cm (1.03 in.) vertical
2	Deltron Non-Magnetic Linear Ball Slides- S2-3
3	TSINY TRS-775W 12000 rpm DC motor
4	0.635 cm (0.25 in.) dia. x 0.635 cm thick K& J cylinder magnets
5	SparkFun ROB-09238 Bipolar stepper motor with 200 steps
6	Panasonic PM-L45 Photointerrupter
7	Bone Swiss Bearings 8mm (0.315 in.)
8	US Digital EM2-2-10000-I Optical encoder module
9	HUBDISK-2-10000-315-IE Transmissive rotary disk
10	NI USB-6356 Data acquisition box

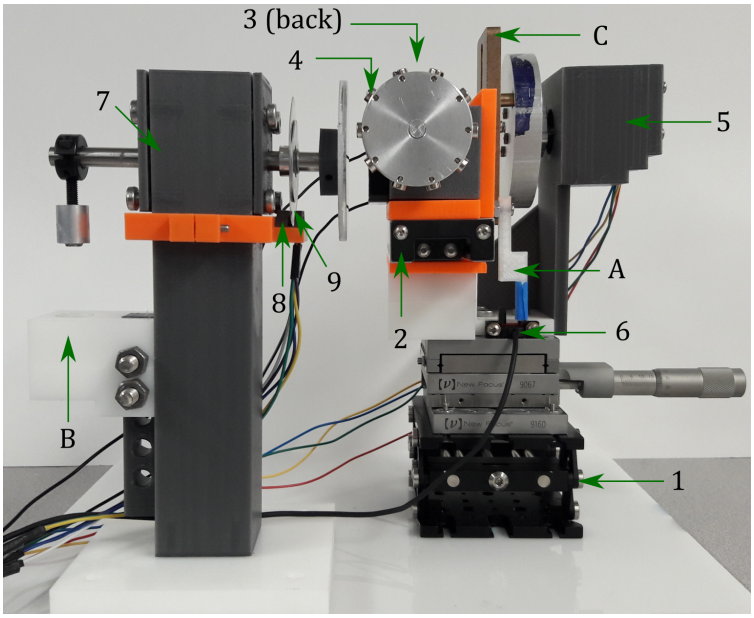


Figure 3.1: Experimental Setup. See Table3.1 for parts 1-9. A: Photo-interrupter blocker, B:Optional Magnet hole, C:Scotch Yoke.

A safety polycarbonate cage was built with a thickness of 0.64 cm (0.25 in.) to protect against the possibility of dislodged magnets. The speed of the DC motor and the stepper motor is kept constant during the experiments, and they are controlled with an L298N motor driver. An NI USB 6356 data acquisition box was used to collect data. In this study, both the simulation from the experiment's model as well as the resulting experimental data are

used. For the simulation, the equation of motion is

$$\ddot{\theta} = -\frac{mgr_{cm}sin\theta}{I} - \frac{b_v\dot{\theta}}{I} - \frac{b_c sgn(\dot{\theta})}{I} + \frac{\tau_F cos(\phi)}{I} + \frac{\tau_{dipole}}{I}. \tag{3.1}$$

where mg is the weight of the pendulum and r_{cm} is the distance between rotation axis and the center of mass of the pendulum. b_v is the magnetic damping and viscous drag coefficient. b_c is the Coulomb damping coefficient, I is the moment of inertia for the pendulum, τ_F is the driving torque provided by the rotating magnets at high speed, and τ_{dipole} is included in the equation of motion when the optional magnet (Figure 3.1-B) is used in the experimental setup. The derivation of the expression for the dipole moment can be found in Reference [210]. In addition, there is a phase difference ϕ between the disk with magnets and the aluminum disk given by $\phi = w_F t + \delta$, where $w_F = 2\pi f_F$, f_F is the driving frequency of the stepper motor, and δ is the initial position of the disk with magnets.

3.3 Sparse Identification of Nonlinear Dynamics (SINDy)

There are two main components in SINDy: sparse representation and sparse regression. Each of these components is described below.

Sparse representation: A nonlinear system can be represented according to $\dot{\boldsymbol{x}}(t) = A(\boldsymbol{x})\beta$, where $\boldsymbol{x}_{n\times m}$ contains the state variables of the system and $\dot{\boldsymbol{x}}_{n\times m}$ is the time derivative of the state variables while n and m are the numbers of samples and state variables, respectively. State space representation of the nonlinear system can be composed of a combination of many nonlinear functions of the state variables. Possible combinations of these nonlinear functions are stacked into a matrix $A(\boldsymbol{x})_{n\times p}$ called a feature library where each column represents one possible nonlinear function. There is a coefficient corresponding to each candidate function in the $\beta_{p\times m}$ matrix where p is the number of these nonlinear functions.

$\dot{\boldsymbol{x}}(t) = A(\boldsymbol{x})\beta$ can be expanded into the matrix form

$$\begin{bmatrix} \dot{x}_1(t_1) & \dots & \dots & \dot{x}_m(t_1) \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \dot{x}_1(t_n) & \dots & \dots & \dot{x}_m(t_n) \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{x} & \boldsymbol{x}^{P_2} & \dots & \boldsymbol{x}^{P_k} & sin(\boldsymbol{x}) & cos(\boldsymbol{x}) & \dots \end{bmatrix} \beta, \tag{3.2}$$

where $\boldsymbol{x}^{P_k}$ represents the kth order nonlinearities in the system such that

$$\boldsymbol{x}^{P_k} = \begin{bmatrix} x_1^k(t_1) & x_1^{k-1}(t_1)x_2(t_1) & \dots & x_2^k(t_1) & \dots & x_m^k(t_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^k(t_n) & x_1^{k-1}(t_n)x_2(t_n) & \dots & x_2^k(t_n) & \dots & x_m^k(t_n) \end{bmatrix}. \tag{3.3}$$

The $\sin(\boldsymbol{x})$ and $\cos(\boldsymbol{x})$ terms represent the sinusoidal nonlinear candidate functions, and this provides the flexibility to choose the type of nonlinear functions to be used in this feature library $A(\boldsymbol{x})$ depending on the system whose parameters are investigated [189]. While generating the feature matrix, the different combinations of state variables and their kth order polynomial versions are used. The number of the functions p can increase dramatically when the maximum order k for the polynomial functions increases slightly. Since the equation of motion of the system contains several terms among these candidate functions, the coefficients of most of them will be zero. Therefore, β is a *sparse* matrix. The nonzero coefficients in β can be found using sparse regression, and they describe the governing equations of the system.

Sparse regression: Let us assume there is a model $y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_{ij}$, where p represents the number of predictors for the model, n is the number of samples with $1 \leq i \leq n$, and β contains the coefficients for each predictor. Generally, this model can be solved using the linear regression method when the number of samples is larger than the number of predictors ($n > p$). The intercept term β_0 can be removed by normalizing the columns of the X_{ij} matrix to zero mean and unit variance. Therefore, the optimization problem becomes

$\min_{\beta \in \mathbb{R}^p} ||y - \sum_{j=1}^{p} \beta_j X_{ij}||^2$. However, for the case where $p \gg n$, overfitting can occur [211, 212]. Therefore, the coefficients for each predictor are more accurately obtained by training data samples. Nevertheless, when these coefficients are validated on the test set, the test set and the model obtained with the coefficients found in the training set will not match. It has been discovered that the algorithm finds non-zero coefficients for some predictors which actually do not exist in the actual system. Therefore, sparse regression is used to decrease the number of predictors with non-zero coefficients during regression. In sparse regression, most of the predictors will have zero coefficients, and the weight matrix β will be sparse. There are several ways to decrease the number of predictors, and these are called best-subset selection, forward, and backward-step wise selection, forward-stage wise regression, and shrinkage methods such as ridge classifier and LASSO [190, 191].

An alternative method for sparse regression is used in Reference [189]: an initial guess for the coefficients in the β matrix is found by linear regression. Then, a threshold value for all the coefficients is selected. The coefficients under this threshold value are set to zero, and the indices of non-zero coefficients are defined. A new feature library($A(\boldsymbol{x})$) is generated by eliminating the columns with zero coefficients. Finally, least squares is again applied to this feature library, and the coefficients with values below the threshold are eliminated. This procedure is iterated 10 times resulting in a sparse coefficients' matrix β.

3.4 Results and Discussion

In this section, the results for simulated and experimental data are explained. This study also shows the effect of the regularization parameter of TVR on the estimated coefficients and compares TVR to other derivative estimation methods.

3.4.1 Total Variation Regularization (TVR)

SINDy requires the derivatives of the state variables as input, and when these derivatives are not available, an estimate is obtained using TVR. TVR has two different parameters

that affect the approximation quality: the regularization parameter α and a constant ϵ. Despite the large influence of these parameters on the derivative estimates, they are difficult to optimize. Some guidance on choosing α was provided in Reference [208] when the noise amount in the measurement is known. However, this can be challenging in practice due to unknown levels of noise. Therefore, trial and error approach is used for selecting the TVR parameters. For simulated data, the ground truth for the derivatives is known, so TVR's accuracy can be checked. However, for experimental data, it is extremely difficult to judge how well TVR approximates the true derivatives.

3.4.1.1 Noise-free Case

One of the nonlinear systems to which SINDy is applied is the simple pendulum governed by the equation of motion

$$\ddot{\theta} = \frac{-g}{L} sin(\theta) - \frac{c}{m}\dot{\theta}, \tag{3.4}$$

where m is the mass of the pendulum, L is the length of the rod of the pendulum and c is the damping coefficient. Figure 3.2 shows the computed TVR derivatives using $\alpha = 10$ and $\epsilon = 10^{12}$. Polynomial order up to 4 and the *usesine* option of the code provided in Reference [189] are used for sparse regression. This code applies sparse regression by iteratively thresholding the least square coefficients. For all applications presented in this section, the threshold value $\lambda = 0.1$ was chosen for sparse regression. Figure 3.2 shows that the estimated and the simulated derivatives match well for both state variables. However, the aim of SINDy is to predict the underlying model, so the coefficients found by sparse regression need to be checked. The number of nonzero coefficients in the coefficient matrix β is higher than the number of terms in the equation of motion (see Table 3.2). Further, most of the nonlinear terms whose coefficients are not supposed to appear in the matrix have large coefficients. To pinpoint the reason for this mismatch, the simulation derivatives were injected directly into the sparse regression algorithm, and an exact model match was

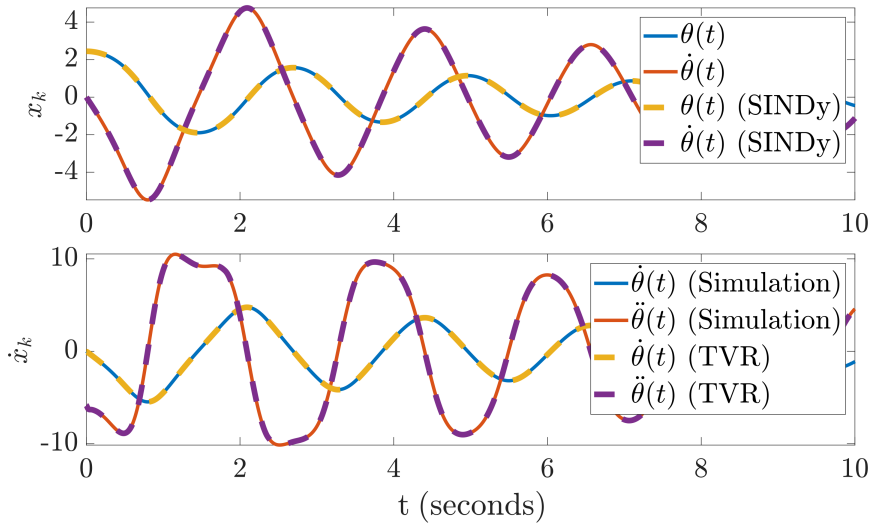


Figure 3.2: Simple pendulum simulation vs SINDy approximation based on TVR derivatives. ($c = 0.5$ Ns/m, $m = 2$ kg and L=1).

obtained. This highlights the importance of the derivative estimate and cautions that even though the estimated derivative may look good, the model identification can still fail.

Table 3.2: Coefficients found by sparse regression for the simple pendulum based on TVR derivative estimations.

Term	1	x_1	x_2	...
x_1	0	0	1	...
x_2	8.2693×10^6	1.8066×10^6	-0.2506	...
Term	$\sin(x_1)$	$\cos(x_1)$	$\sin(x_2)$	...
x_1	0	0	0	...
x_2	-2.1477×10^6	0	-8.6466×10^6	...

3.4.1.2 Regression Overfitting Check

In addition, since the time series of the estimated model matches that of the exact model—despite estimating significant superfluous coefficients, this suggests the possibility of overfitting in the regression. In order to check this possibility, the simulation data is split into a training set (70%) and a test set (30%). Specifically, the training set is used to predict the model of the system over the first part of the signal, which was 7 seconds of the pendulum simulation. The last part of the simulation—which is 3 seconds for the pendulum

example—was assigned as the test set, and the predicted model was solved for the test set time span. Figure 3.3 shows that the estimated model matches the simulation; therefore, although the possibility of overfitting is eliminated for this data, a similar check is needed for other data sets.

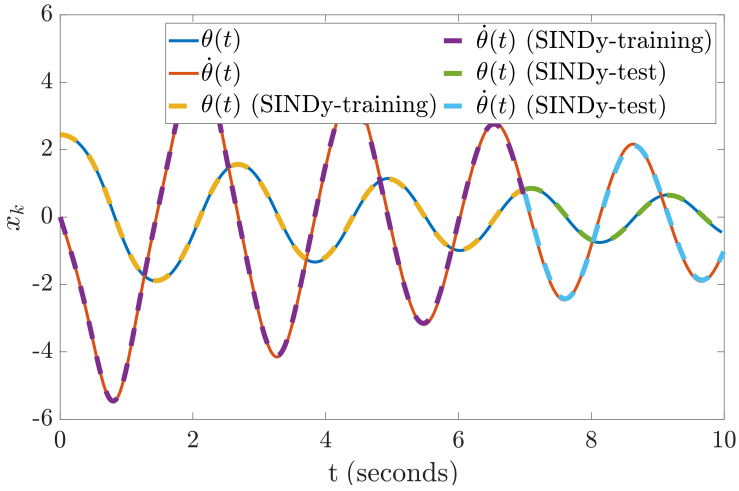


Figure 3.3: Simple pendulum training and test set predictions.

To further test the performance of SINDy, it is applied for the estimation of the model of a chaotic pendulum with varying interaction potential, see Section 3.2. This study forgoes the optional stationary magnet that repels the rotating pendulum magnet, see part B in Figure 3.1. Using this magnet in the experimental setup results in a dipole torque term in the equation of motion. Since this nonlinear torque term includes rational expressions and SINDy does not perform well in the presence of rational terms [189], the optional magnet was omitted. The system was simulated with zero initial conditions, and its derivatives were predicted with TVR. The coefficients for each term in Eq. (3.1) are obtained from Reference [210], and the state space form of the nonlinear system reads

$$\begin{aligned} \dot{x}_1 &= \dot{\theta} \\ \dot{x}_2 &= -73.698\sin(\theta) - 0.3734\dot{\theta} - 0.423\mathrm{sgn}(\dot{\theta}) + 46.596\cos(\phi), \\ \dot{x}_3 &= 6.4717 \end{aligned} \tag{3.5}$$

where $x_1 = \theta$, $x_2 = \dot{\theta}$, $x_3 = \phi$, and $\alpha = 2 \times 10^{-5}$ and $\epsilon = 10^{12}$ was set. Note that since the

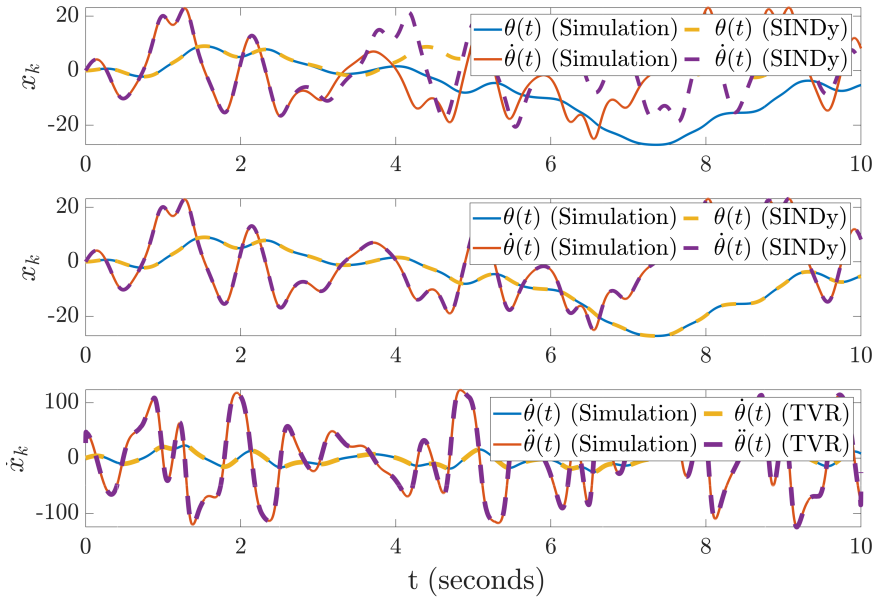


Figure 3.4: Estimation of the chaotic pendulum response with SINDy when TVR derivatives (top) and simulation derivatives (middle) are used. The bottom panel compares the derivatives of the simulation to their TVR counterpart. (TVR parameters: $\alpha = 2 \times 10^{-5}$ and $\epsilon = 10^{12}$).

signum function appears in Eq. (3.1), it is added to the feature library. For the polynomial function candidates, a maximum degree $k = 3$ was selected, and trigonometric function terms were also included in the feature library. Figure 3.4 shows the resulting estimated model and derivatives. The top panel shows that SINDy was able to estimate nearly the first four seconds of the simulation even though the bottom panel shows that the TVR-estimated and the simulated derivatives match for the full 10-second time horizon. The middle panel shows that when using the simulated derivatives in SINDy, the estimated system response matches the simulation throughout the time horizon.

Table 3.3 compares the estimated model coefficients and the ones used for the simulation. The left part shows that using the 'exact' derivatives correctly identifies the model coefficients. On the other hand, the right part shows that estimating the derivatives using TVR leads to correctly estimating four different coefficients; however, some of the candidate functions which should vanish have nonzero coefficients leading to a mismatch between the estimated and the simulated models.

Table 3.3: Coefficients used in the simulation versus those estimated by SINDy. Shaded cells highlight the matching coefficients.

	Simulation Coefficients							Estimated Coefficients						
Term	1	x_1	x_2	$\sin(x_1)$	$\cos(x_3)$	$\text{sgn}(x_2)$	$\text{sgn}(x_3)$	1	x_1	x_2	$\sin(x_1)$	$\cos(x_3)$	$\text{sgn}(x_2)$	$\text{sgn}(x_3)$
$\dot{x}_1$	0	0	1	0	0	0	0	0	0	1	0	0	0	0
$\dot{x}_2$	0	0	-0.3734	-73.698	46.596	-0.423	0	-19.386	0	-0.3706	-73.8088	46.5961	-0.4216	19.3841
$\dot{x}_3$	6.4717	0	0	0	0	0	0	3.6638	0	0	0	0	0	2.8073

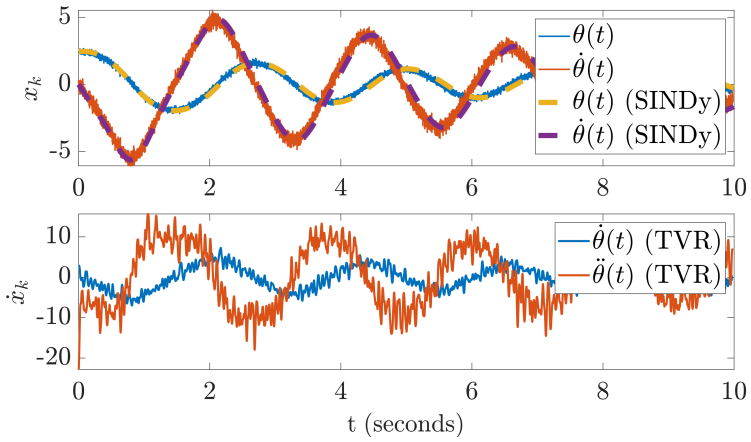


Figure 3.5: Estimation of the simple pendulum response based on SINDy and simulation data with SNR = 20 dB.

3.4.1.3 Noise Effects

To investigate the effect of noise on model estimation, SINDy was applied to the simple pendulum simulation data with additive Gaussian noise with SNR = 20 dB. Figure 3.5 shows the predicted model response and the derivatives estimated with TVR. It is seen that SINDy is able to match the simulation response despite a large amount of noise in the simple pendulum simulation. However, Table 3.2 shows that there is a mismatch between the coefficients of the estimated and simulated models. Similar to the noise-free case, most of the nonlinear terms in the feature library of the noisy system have large coefficients. While SINDy correctly estimated the system response for this numerical example, this may not be the case for other systems.

In addition, Gaussian white noise with SNR = 35 dB has been added to the chaotic pendulum simulation (an SNR of 20 dB did not produce meaningful results). $\alpha = 100$ is used for TVR to have a smooth derivative. Figure 3.6 provides the estimated model response

where it is seen that SINDy can estimate only the first three seconds. This further shows that α and ϵ need to be carefully tuned; however, these two parameters can be any positive real number which makes them hard to optimize. The number of the estimated nonzero coefficients is larger than the ones in Eq. (3.5). Some of the terms in the feature library which are supposed to have zero coefficients have nonzero coefficients due to estimating the derivative. Therefore, the sensitivity of the model estimation to TVR parameters is once again confirmed.

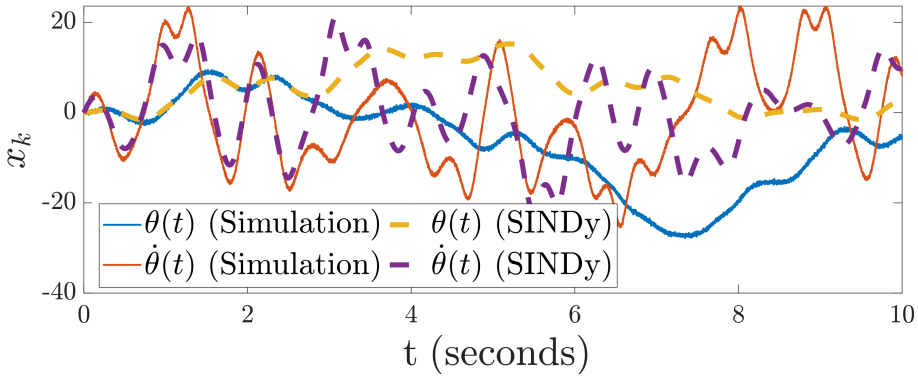


Figure 3.6: Estimated response of the chaotic pendulum using TVR derivatives with $\alpha = 100$, $\epsilon = 10^{12}$, and SNR = 35 dB.

3.4.1.4 Parameter Sensitivity in TVR via an Example

Next this study focuses on the search for suitable α and ϵ parameters using a Lorenz system with the parameters listed in the caption of Figure 3.7, whose model contains $N = 7$ nonzero terms. Lorenz system example given in the original SINDy paper [189] was used since it was an example where the actual model was correctly predicted. The Lorenz system simulation contained zero mean Gaussian noise with a variance of 0.01. Figure 3.7 shows the influence of varying α and ϵ on the number of significant nonzero terms. Two different ϵ values are used, and for each value of ϵ, α is varied, and the number of nonzero terms is plotted.

Figure 3.7 shows the sensitivity of the model prediction to the TVR parameter values. Specifically, focusing on the left plot in Figure 3.7 for $\epsilon = 10^{12}$, large jumps are observed

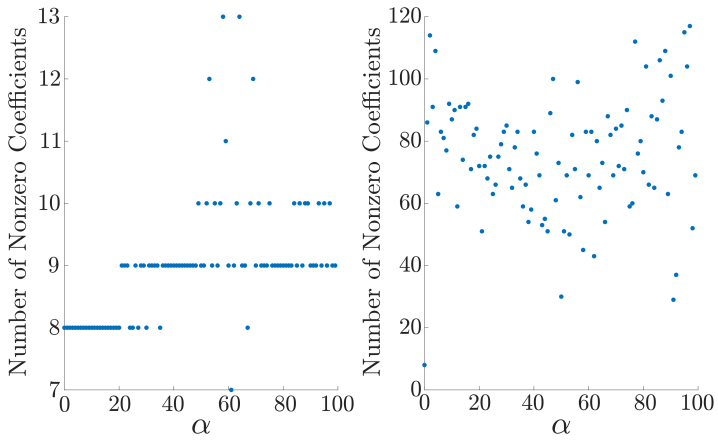


Figure 3.7: Number of nonzero coefficients estimated by SINDy for Lorenz system with parameters: $\sigma = 10$, $\beta = 2.667$, $\rho = 28$. Initial conditions: $x_0 = -8$, $y_0 = 8$ and $z_0 = 27$. $\epsilon = 10^{12}$ (left), $\epsilon = 10^{-3}$ (right). The correct number of nonzero terms is $N = 7$.

in the number of nonzero coefficients for small variations in the α value with the number of coefficients ranging in 7–13 for $\epsilon = 10^{12}$. The right plot shows even more radical values for N when $\epsilon = 10^{-3}$: varying α, in this case, results in large and scattered values of N. Although the correct total number of nonzero coefficients $N = 7$ can be achieved when the regularization parameter is too small—(as seen in Figure 3.7(right) for $\epsilon = 10^{-3}$)—N jumps to 80 when α is slightly increased. These two plots show that SINDy is unstable with regard to the TVR parameters α and ϵ.

3.4.2 Alternative Derivative Estimation Methods

The limitations of using TVR led us to seek other methods for estimating derivatives of noisy signals [213]. Among the methods in Reference [213], Savitzky-Golay approximation was investigated, cubic splines, Gaussian moving average, and convolution smoothing to estimate the derivative. The following paragraphs briefly describe the tuning parameters for each method.

Savitzky-Golay: Savitzky-Golay filter is used to smooth noisy data and based on linear least squares [214]. It has two parameters defined on the natural numbers: the window length and the polynomial order. There are certain limitations on the window length: it

must be an odd natural number and cannot be less than the chosen polynomial order or greater than the signal length. The Savitzky-Golay Python built-in function has a derivative option.

Cubic smoothing splines: These splines depend only on the smoothing parameter varying between 0 and 1.

Gaussian weighted moving averages: This method has only one positive integer parameter: the window length.

Convolution smoothing: This method smooths noisy signals, and the derivative can then be computed using centered difference or the function derivative option of MATLAB. Its parameters are the window length and the window type, where for the latter, a Hanning window was chosen. The window length must be an odd, natural number similar to Savitzky-Golay. Arguably, these methods can be more easily tuned than the TVR approach.

This study uses all four, in addition to TVR, for the chaotic pendulum simulation and compares the resulting performance of SINDy in terms of the response match and the estimated coefficients. Figure 3.8 compares the ground-truth response to the estimated model responses corresponding to the different derivative estimates as well as the simulated derivative. It is seen that the estimated response matches the simulation when the 'true' derivatives are used in SINDy. In this case, the estimated coefficients also match the true model. Moreover, it is seen that Savitzky-Golay performs the best for both state variables. The deviations of the different methods from the true response can be explained by either their extraneous estimated terms or by errors in the nonzero coefficients. With the exception of the Gaussian moving average, all the methods resulted in more nonzero coefficients than the true model. The deviation of the Gaussian moving average method can be attributed to the error in the estimated coefficients.

The effect of noise on the performance of the derivative estimation methods was tested by adding white noise with SNR= 35 dB to the chaotic pendulum simulation. Figure 3.9 shows that with noise at best the methods can estimate roughly the first 3 seconds of $\theta(t)$,

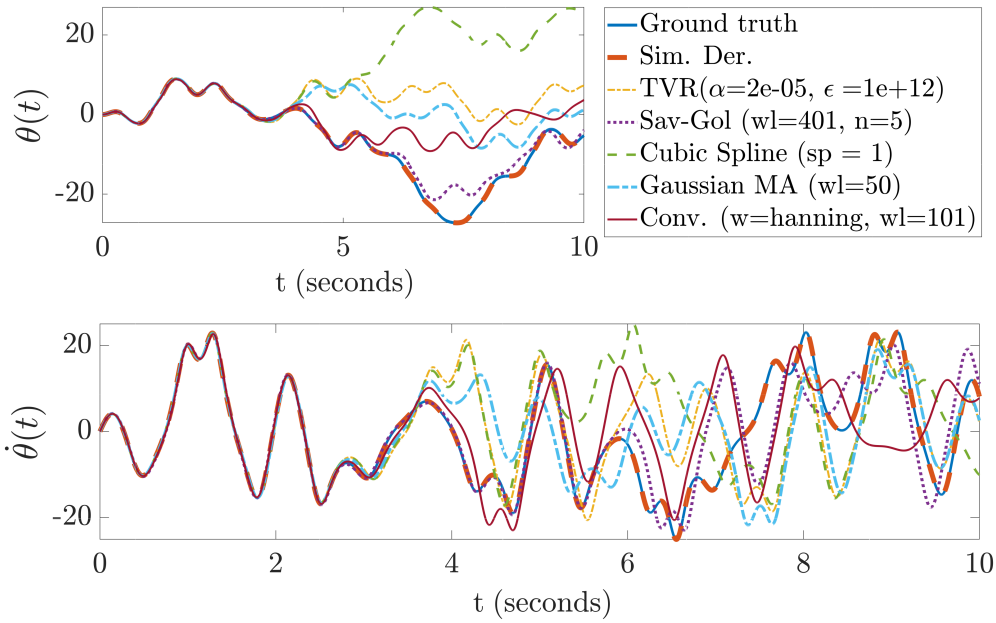


Figure 3.8: Estimated responses for the simulated chaotic pendulum.

but they all perform poorly for $\dot{\theta}(t)$. Responses obtained with Savitzky-Golay, convolution smoothing, and TVR-based derivatives deviate from the simulation around $t = 3$ for $\theta(t)$. However, in contrast to TVR, Savitzky-Golay and convolution smoothing are easier to tune than TVR, which gives them an advantage. Further, the estimated coefficients for Savitzky-Golay and the convolution smoothing are closer than TVR to the true model. Nevertheless, all derivative estimation methods resulted in spurious nonzero terms in comparison to the true model.

3.4.3 Experimental Results

Experimental data is collected from the setup explained in Section 3.2. Similar to the numerical simulation, the optional magnet B was not included. Position data of the pendulum was collected using an incremental encoder. The data contains A and B quadrature signals, and the raw data is collected on analog channels of a data acquisition box with a sampling frequency of 10^6 Hz per channel. The analog signals were digitized by hard-thresholding at 4V, and by tracking the A and B signals at each time step, the pendulum's angular position

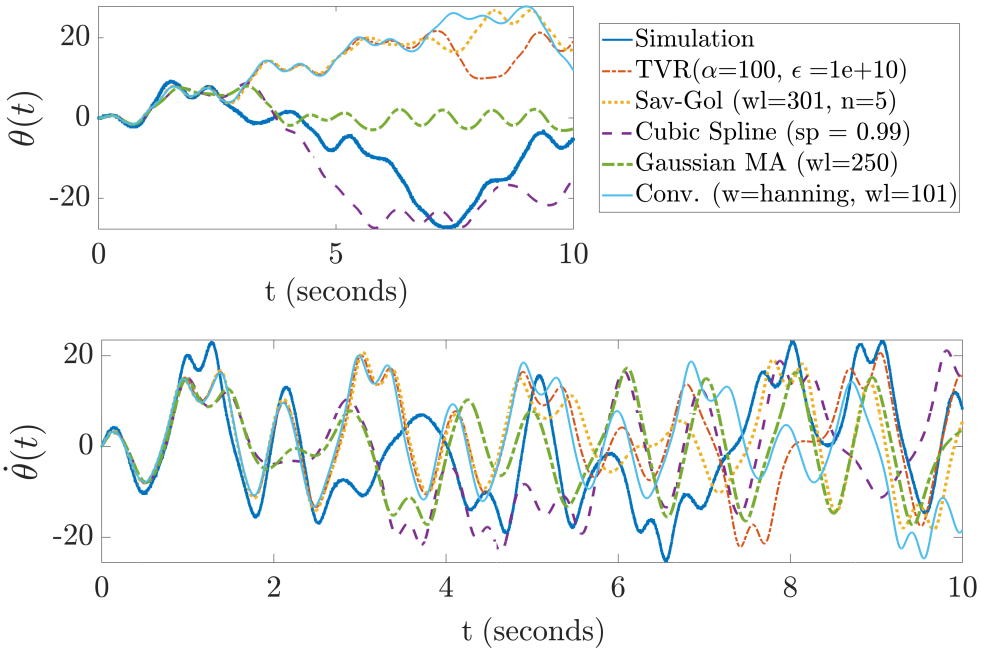


Figure 3.9: Estimated responses for the simulated chaotic pendulum with noise (SNR=35 dB).

in radians is obtained.

In contrast to numerical data where both $\theta(t)$ and its derivative are known, only $\theta(t)$ can be observed in the experimental setting. In this case, [189] suggests using delay reconstruction of the time series combined with Singular Value decomposition (SVD) to obtain the remaining state variables. Therefore, this study used Taken's embedding with embedding dimension 10 and delay parameter 1 and the first three columns of the right singular matrix V to compute the derivatives. However, this approach led to nearly zero response, so we proceeded by estimating the derivative of $\theta(t)$ using TVR as well as the methods of Section 3.4.2.

The results for estimating the response using the experimental data are plotted in Figure 3.10. The figure shows that none of the derivative estimation methods fits the experimental data, and while they all have large errors, the error in TVR, in particular, grows rapidly beyond $t = 20$. In contrast, Savitzky-Golay derivatives result in the closest fit to the experimental data.

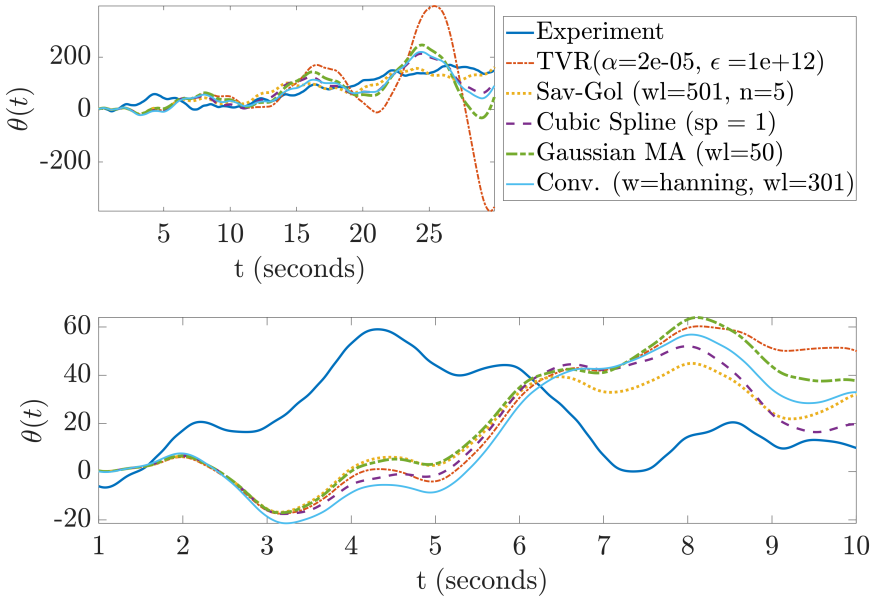


Figure 3.10: Estimated model responses for experimental data of chaotic pendulum (Top) and a zoomed-in version (bottom).

3.5 Conclusion

This study investigated the performance of SINDy, a data-driven tool for model identification from numerical and experimental data. Specifically, SINDy was applied to two nonlinear systems: a simple pendulum and a chaotic pendulum with variable interaction potential. The time series for the former was obtained only via simulation, while the time series for the latter originated from both simulation and experiments. The performance was assessed based on the estimated model's fit to the data and the estimated coefficients' correctness– when the true model is known. It is found that when the derivatives of the states of the system are precisely known, SINDy yielded the correct response and the correct model. However, sparse representation often requires estimating derivatives from the time series in practice. So TVR, the standard tool in SINDy, was compared to other available tools for derivative estimation with noisy and noise-free data.

The results show that TVR is hard to tune due to the unbounded support of its parameters and that even if it is well-tuned, it can result in a good fit for a limited time duration that seems to depend on the system complexity. Further, even when TVR fits the data, the

resulting coefficients can be incorrect. Among the other derivative estimation methods, the Savitzky-Golay filter showed promising results for both simulation and experimental data, despite choosing its parameters using trial and error. Convolution smoothing is another method that results in a good fit for simulation data.

One reason for adopting TVR in SINDy is its noise robustness. However, it is shown that for the examples shown in this study, Savitzky-Golay provided nearly the same performance as TVR, see Figure 3.6. It may be possible to obtain better results for both nonlinear systems by judiciously tuning the parameters. Nevertheless, even if the fit with any given data is perfect, it is still possible that the identified model will be incorrect. This is a crucial drawback when dealing with experimental data where the ground truth is unknown. Therefore, more work is needed to generate error bounds on the model identified by SINDy, and guide the parameter choices in derivative estimation.

CHAPTER 4

ANALYSIS OF ENGINEERING SURFACES USING TOPOLOGICAL DATA ANALYSIS

4.1 Introduction

Surface texture analysis is a prominent field of research with many applications, including tribology [215], metrology, remote sensing [216], medical imaging [217], and the marine industry [218]. One specific active area of research is the fast and automatic feature extraction from image data that reduces the need for the input of expert users. In addition to the need for reliable, automatic feature extraction, other challenges in surface texture analysis include the size of the data, which significantly increases with increasing the resolution. Therefore, there is a need for adaptive and automatic tools for feature extraction from surface images.

The majority of the tools proposed for roughness analysis of engineering surfaces are based on decomposing the image data using a set of basis functions that can be grouped into three main components: form, waviness, and roughness. Form contains the lowest frequencies, while waviness is composed of sinusoidal waves in the middle frequency range. Larger frequencies are included in the roughness component. Generally, most surface analysis tools focus on finding the reference surface or profile. Depending on the feature extraction tool used, the reference surface (profile) is composed of form, or it is the combination of both form and waviness. The surface roughness can then be obtained by subtracting the form and the waviness from the original surface.

For surface profile analysis, the Gaussian filter is one of the most commonly used filters in the literature [219, 220, 221, 222, 223]. It is used as a low-pass filter to obtain a smoother surface, and the roughness profile is then obtained by subtracting the filtered profile from the original one. Raja et al. used a Gaussian filter to obtain an approximation to surface profiles, and they compared this approximation with the ones obtained from the 2RC filter, one of

the earliest filters used for surface metrology [220]. Hendarto et al. focus on the roughness analysis of wood surface using Gaussian filter [222]. However, the main drawback of the Gaussian filtering approach is the boundary distortion, where the mean of the end parts of a surface profile cannot be used [220]. Raja et al. suggested that the end parts of the mean line should be ignored for evaluation [220], while this is not feasible for profiles with shorter lengths. Therefore, Janecki proposed a solution that extrapolates both ends of profiles with polynomial functions to eliminate the edge effect [223]. Fast Fourier Transform (FFT) is also another widely adopted filtering approach for feature extraction from 1D signals [55] including surface profiles. For example, Raja and Radhakrishnan used FFT to denoise 1D surface data and obtain the corresponding roughness profiles [224]. For surface areal data, two dimensional implementations of FFT and Gaussian filter can be utilized [225, 226]. Dong et al. provide an extensive understanding of two-dimensional FFT (2D-FFT) analysis on engineering surfaces [227]. Peng and Kirk applied 2D-FFT to surface images of three different wear particles and used spectral intensity values in angular and radial spectra to identify the type of wear particle [226]. Empirical Mode Decomposition (EMD), one of the most commonly adopted signal decomposition tools, is another approach used for the analysis of engineering surfaces. Several versions of EMD are proposed to analyze surfaces such as Bidimensional EMD (BEMD) [228], Image EMD (IEMD)[229], Bidimensional Multivariate EMD (BMEMD) [230]. However, the computation of EMD in 2D is slow compared to other approaches.

Discrete Cosine Transform (DCT) is another widely used approach for decomposing a surface scan into its form, waviness, and roughness components [231, 232, 233, 234]. Lecompte et al. developed an approach to identify the form and the contribution of classical defects such as positioning error and tool deflection [232]. They used only a certain percentage of the DCT coefficients to obtain a filtered surface. However, when there are a large number of images, each image may require the usage of a different percentage of the DCT coefficients to generate the form. In general, DCT requires selecting two threshold values for delineating

the three different components of the surface.

Discrete Wavelet Transform (DWT) is another approach used extensively for surface texture analysis [235, 236, 220, 234, 237, 238, 239, 240, 241]. Chen et al. introduced DWT for surface profiles [235]. Liu et al. obtained a threshold that isolates the form of the surface by computing all possible approximations that can be obtained using the coefficients at each level. Another example of this approach is seen in Reference [220, 237] where the separation of the three components of a profile is performed using multi-resolution analysis approximations. The common procedure is to apply the DWT at a certain level and obtain the approximation and detail coefficients, and then use the approximation coefficients for the reconstruction of the form component [242]. The detail coefficients are then used to reconstruct waviness and roughness. Nevertheless, there is a need for a guideline on how to automatically choose the threshold that separates the mid-frequency content from the higher ones in the DWT approach. In addition, the selection of the mother wavelet function can also affect the resulting components. Stkepien et al. used autocorrelation, cross-correlation, and entropy-based test to evaluate the performance of different wavelet functions used in surface texture analysis [243].

It is believed that there is no approach for automatically separating the form, waviness, and roughness components for DCT and DWT, and the current practice is to manually select them using the user's experience and judgment call [244]. Therefore, the first contribution of this study is to propose an automatic, data-driven approach for identifying the needed thresholds for DCT and DWT. For DWT, this study utilizes the energy of the reconstructed signals to separate the waviness and roughness from each other, while for DCT, this study leverage the surface entropy to define the form and waviness components. Roughness is then found by subtracting the filtered surface from the original one.

In addition to this study's contributions to the automatic threshold selection in DCT and DWT, the machining processes through which surface samples are generated are identified. Most studies in the literature are focused on small patch processes with few samples where

human interpretation is heavily used to identify and compute profile or surface roughness. In contrast, the proposed approach is validated for a large data set obtained from simulated surfaces and experimental surface scans. The roughness components of surfaces and profiles are obtained using proposed automatic threshold algorithms, and the 1D and 2D features introduced in the ISO standards [245, 246] are extracted. Machine learning is then utilized to assess the accuracy of the automatic thresholds. Specifically, the obtained features are used in supervised classification algorithms to classify surfaces labeled with respect to the generating surface parameter for the simulated surfaces and the generating machining process for the experimental data.

The second contribution of this study is to use persistent homology, which is a tool from TDA for quantifying the roughness of surfaces. Specifically, 0D and 1D sublevel set persistence are used on surface profiles and surface images to compute the sublevel set persistence diagrams [247]. Then, Carlsson Coordinates [71, 69], persistence images [74], and template functions [72] are utilized to extract features from these diagrams. The TDA-based approach is used on synthetic data sets to identify the level of roughness, and its performance is compared to features extracted using traditional image analysis.

The final contribution of this study is to develop an approach for real-time surface texture analysis. Investigation of surface scans can be performed using the traditional signal processing approaches mentioned above. A combination of automatic threshold selection algorithms and High Performance Computing tools can make these approaches viable for real-time surface texture analysis. Since TDA-based tools show promising results for surface roughness analysis, this study proposes an alternative approach for real-time surface texture analysis using topological simplification tools from TDA. Specifically, the proposed framework takes the physical surface images as input and utilizes topological saliency [1] from TDA. The proposed pipeline provides clusters of the surface images to identify the regions where additional machining is required. It is hypothesized that this framework can significantly reduce material waste and time for obtaining a smooth surface finish.

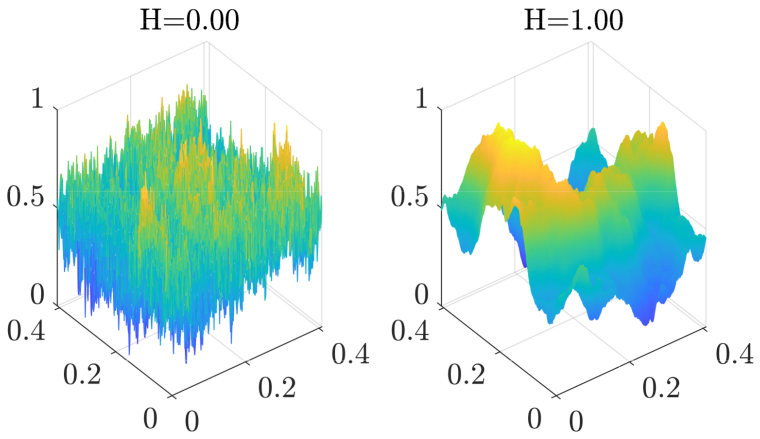


Figure 4.1: The roughest and smoothest surface in the synthetic data set obtained with $H = 0$ and $H = 1$, respectively.

This chapter is organized as follows. Section 4.2 explains how the synthetic data set was obtained and analyzed using traditional tools and the TDA-based approach. It also compares traditional feature extraction approaches to the TDA-based method. In Section 4.3, preprocessing of experimental data and the automatic threshold selection algorithms are described. In addition, the results of heuristic threshold selection and proposed automatic threshold selection algorithms are compared. Section 4.4 explains the topological simplification approach based on topological saliency and provides the resulting surface clustering.

4.2 Data-driven and Automatic Surface Texture Analysis Using Persistent Homology

4.2.1 Simulation

Synthetic surfaces are used to test the proposed approach, and they are generated using the model provided in Reference [248]. The surface roughness of the resulting surfaces is controlled by Hurst roughness parameter $H \in [0, 1]$. As the value of H varies from 0 to 1, the generated surface gets smoother and smoother, see the example surfaces in Figure 4.1.

The $[0, 1]$ range is divided into 200 intervals, and 201 roughness parameters are obtained. Each roughness parameter was then used to generate synthetic surfaces. Then, the resulting surfaces are categorized according to their roughness parameter value into three classes. The

first and last 67 surfaces are categorized as rough and smooth surfaces, respectively. The surfaces in between these two cases were tagged as somewhat rough.

In addition to the generated surface data, surface profiles are also investigated in this study. Six surface profiles are extracted in two perpendicular directions of surfaces. Therefore, there are totally 1206 surface profiles whose labels match the underlying generated surfaces.

4.2.2 Methodology

This section briefly explains the feature extraction methods from surfaces and surface profiles. The methods used in this study are categorized into two groups: 1) traditional image/signal processing methods and 2) TDA-based approach. For the first one, the general idea is to find a reference surface or a profile and subtract it from the original measurement to obtain the roughness surface or profile. Then, height parameters, spatial parameters, and hybrid parameters provided in Sections 4.1-4.3 of Reference [246] are computed for roughness profiles. While working with roughness surfaces, height and hybrid parameters are used as features, and they are provided in Secs. 4.2 and 4.4 of Reference [245]. For the 1D peak selection method of FFT, the coordinates of the peaks of FFT and PSD plots are used. The angular spectral densities are used as features in the case of the two-dimensional FFT.

For the TDA based approach, three featurization techniques are used to generate feature vectors for persistence diagrams: Carlsson Coordinates [71, 69], persistence images[74], and template functions[72].

4.2.2.1 Gaussian Filtering

1D-Implementation Gaussian filtering is one of the most commonly used tools for profile filtering [220]. Gaussian filtering is implemented in 1D and 2D to analyze surface profiles

and areas, respectively. The 1D kernel definition is given as [219],

$$G(x) = \frac{1}{\alpha\lambda_c}\exp\left(- \pi\Big(\frac{x}{\alpha\lambda_c}\Big)^2\right), \tag{4.1}$$

where $\alpha = \sqrt{ln2/\pi}$, and λ_c is the roughness long wavelet cutoff [220]. Cutoff selection is performed with respect to the iterative procedure provided in Reference [249]. First, the surface roughness parameter, R_a is estimated for surface profiles using the expression [246],

$$R_a = \frac{1}{L}\int_L \mid z(x) \mid dx, \tag{4.2}$$

where L represents the measurement length of the profile. A cutoff value is chosen from Table 3-3.20.2-1 provided in Reference [249]. Then, R_a is measured for the roughness profile after applying the filter using the chosen cutoff value. If the new R_a is outside of the range of the old R_a, a new cutoff is selected with respect to new R_a. However, if it is larger than the measurement length, the algorithm automatically picks the first chosen value as cutoff. This procedure is for nonperiodic profiles, and one can refer to [249] for more details.

After setting the cuttoff value and applying the Gaussian filter, a filtered profile is obtained. This profile is also called the roughness mean line. The roughness profile is obtained by subtracting the mean line from the original surface profile. Then, the profile features provided in Reference [246] are computed to generate the feature matrix for supervised classification.

2D-Implementation The 2D Gaussian kernel expression is given as

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-x^2-y^2}{2\sigma^2}\right), \tag{4.3}$$

where σ is the standard deviation. After the kernel in 2D is computed, the surface measurement is convoluted with the kernel to obtain the filtered surface. The convolution is performed using

$$I[i,j] = \sum_{u=-W}^{W}\sum_{v=-W}^{W} G[u,v]f[i-u,j-v], \tag{4.4}$$

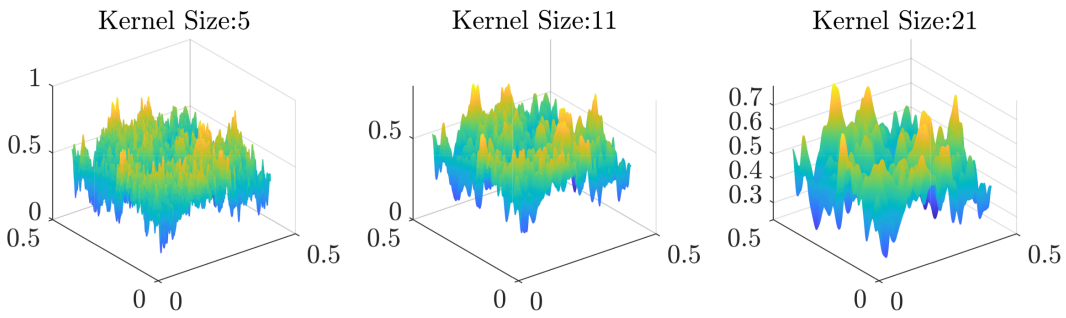


Figure 4.2: Filtered surfaces obtained using kernel sizes 5, 11 and 21.

where $2 \times W + 1$ equals the kernel size K, f is the surface measurement, and I is the filtered surface. The standard deviation σ is defined using the expression, $\sigma = K/6$.

Gaussian filtering is applied in 2D to the roughest surface in the synthetic data set with three different kernel sizes. The resulting surfaces are provided in Figure 4.2. It is seen that larger kernel sizes provide smoother filtered surfaces. The roughness surface is obtained by subtracting the filtered surface from the original surface. Smoother filtered surfaces allow having higher frequency components in the roughness surface. Therefore, a kernel size of 21 is selected and kept constant in all filtering operations. Then, areal parameters obtained from Reference [245] are computed on roughness surfaces obtained after filtering. These parameters constitute the features for supervised classification.

4.2.2.2 Fast Fourier Transform (FFT)

1D - Denoising Fast Fourier Transform is one of the most adopted signal and image processing tools. It was employed to analyze surface profiles in Reference [224]. The main idea is to manipulate the spectrum and then apply inverse FFT to obtain a filtered profile. FFT was applied on the surface profiles, and their normalized spectra were obtained. A cutoff value is selected between zero and one. The amplitudes below that cutoff are set to zero, thus eliminating the corresponding frequencies from the data. Inverse FFT is then applied to the modified spectrum to yield a mean line profile. Subtracting the filtered profile from the original one gives the roughness profile.

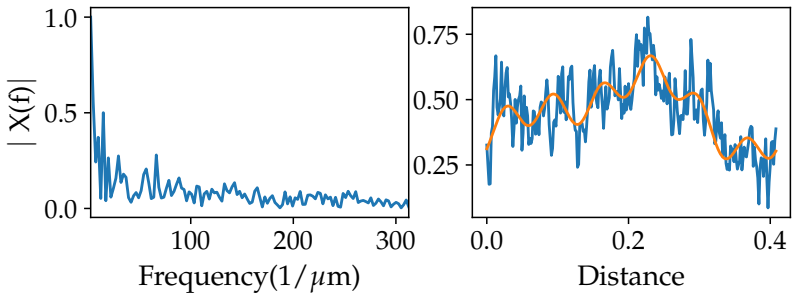


Figure 4.3: (Left) The spectrum of the plots. Filtered profiles obtained from two cutoff values, 0.2 (middle) and 0.4 (right).

An example of a filtered profile is provided in Figure 4.3. The figure shows that larger cutoff values provide smoother profiles. Therefore, a cutoff value of 0.4 is chosen to eliminate high frequencies in the filtered profile. Profile parameters are computed for each roughness profile, and a feature matrix is generated.

1D - Peak Selection The peaks' coordinates in Fast Fourier Transform, Power Spectral Density (PSD), and Autocorrelation (ACF) plots can be used as features in 1D signals [55], and that is the approach implemented here for identifying the level of roughness in the synthetic data set. However, ACF plots were excluded since no peaks were detected in the ACF plots (see Figure 4.4).

First, the FFT and the PSD spectra are computed from the surface profiles. Then, peak selection is performed with respect to two restriction parameters to locate the true peaks of the spectrum. These parameters are minimum peak height (MPH) and minimum peak distance (MPD). MPD is the minimum sample number between two consecutive peaks. MPD is selected as 7 and 10 for PSD and FFT plots, respectively. The expression for MPH is

$$MPH = y_{\min} + \alpha(y_{\max} - y_{\min}), \tag{4.5}$$

where $y_{\min}$ and $y_{\max}$ are 40th and 50th percentile of the amplitudes in the spectrum, respectively, and α is set to 0.5.

MPD and the parameters in the MPH expression can be adjusted depending on the

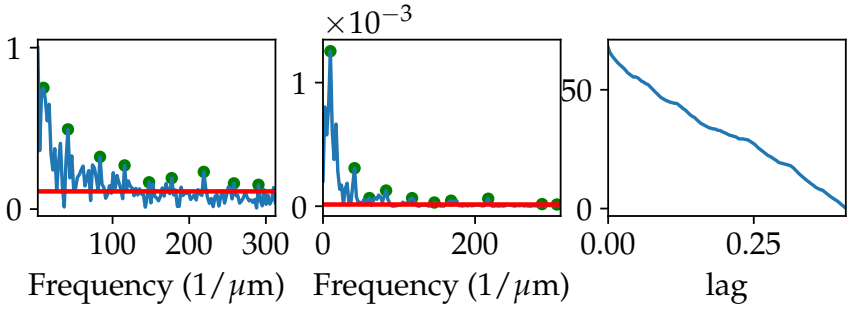


Figure 4.4: Selected peaks for FFT and PSD plots with respect to MPH and chosen MPD values. Red horizontal lines represent the MPH.

data set by *visually* inspecting the selected peaks. This parameter tuning was performed for three different surface profiles obtained from the roughest surface in the data set. After several adjustments, some meaningful peaks are obtained for both spectra, as shown with an example in Figure 4.4. Since manually inspecting all the spectra is time-consuming, this parameter tuning for MPD and MPH is performed for only three profiles, and the tuned parameters are fixed for all the other profiles. After selecting the peaks, their coordinates are used as features for classification. The user can control the size of the feature matrix by specifying the number of peaks.

2D - Implementation FFT can also be applied to images. Two-dimensional FFT is applied to gray scale synthetic surfaces. Areal power spectral density is computed with respect to the formula [226]

$$G(n/NT_x, m/MT_y) = \frac{1}{MNT_xT_y} \mid H(n/NT_x, m/MT_y) \mid^2, \tag{4.6}$$

where $n = 0, 1, \ldots, N-1$ and $m = 0, 1, \ldots, M-1$. M and N are the size of the image, while T_x and T_y are the sampling intervals in x and y directions. $H(n/NT_x, m/MT_y)$ is the 2D Discrete Fourier Transform obtained by using

$$H(n/NT_x, m/MT_y) = \sum_{q=0}^{M-1} \sum_{p=0}^{N-1} h(pT_x, qT_y) e^{-j2\pi np/N} e^{-j2\pi mq/M} \tag{4.7}$$

where $h(pT_x, qT_y)$ represents the surface measurement, $p = 0, 1, \ldots, N-1$ and $q = 0, 1, \ldots, M-1$.

Areal power spectral density (APSD) plots are analyzed using polar coordinates. In this study, Polar FFT [250] is computed to obtain angular and radial spectrums, similar to [226]. In addition, Dong and Stout applied 2D FFT directly to the roughness surface obtained after subtracting the reference surface from the original measurement. In this study, this approach is also employed, and the Gaussian filtering explained in Section 4.2.2.1 is combined with it. APSD plots, as well as radial and angular spectra for two surfaces, are provided in Figure 4.5. Since there are fewer peaks in the radial spectra, only the angular spectra

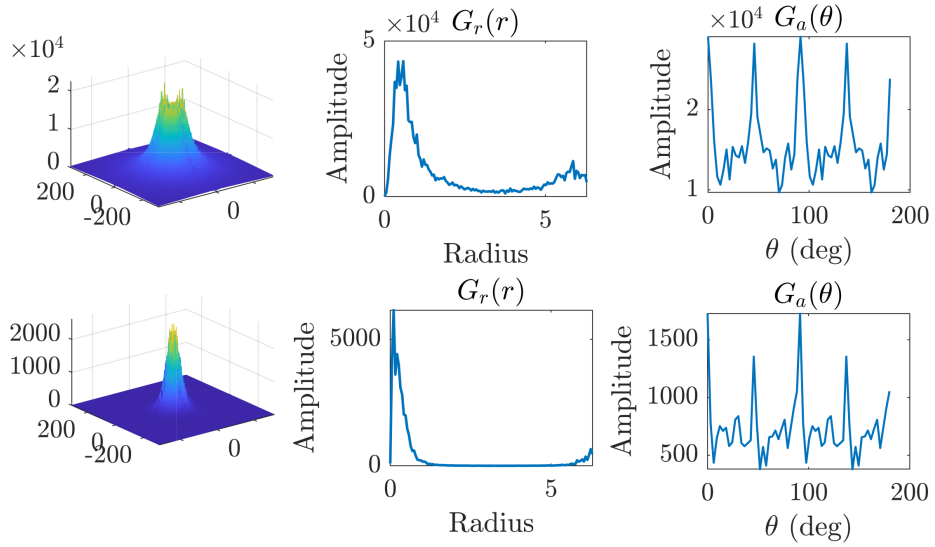


Figure 4.5: APSD plots, radial and angular spectrum of roughest (first row, $H = 0$) and smoothest (second row, $H = 1$) surfaces. APSDs are obtained after applying the 2D FFT on roughness surfaces.

is taken into account. Density values of the five peaks in the angular spectrum are used as features in addition to $\zeta^c_{\max}$ and $\zeta^d_{\max}$, given in Reference [226].

4.2.2.3 Topological Data Analysis (TDA)

In addition to standard signal processing tools used in Secs. 4.2.2.1 and 4.2.2.2, this study uses persistent homology from TDA to extract features from synthetic surfaces. Persistent homology is the flagship tool from TDA, and it analyzes the shape of the data. This section briefly explains persistent homology, and one can refer to Refs. [140, 139, 62] for more details.

Background The sublevel sets of the images (see Figure 4.6a and 4.6b) and of surface profiles are used. Let f be a function that represents data set such that $f : \mathcal{X} \to \mathbb{R}$. The domain of surface profiles or surfaces is denoted as $\mathcal{X}$. Then, the sublevel sets of f are defined as

$$L_\lambda = \{x : f(x) \leq \lambda\} = f^{-1}([-\infty, \lambda)), \tag{4.8}$$

where λ is a threshold [251]. The sorted set of threshold values, $\lambda_1 < \lambda_2 < \ldots < \lambda_l$ forms an ordered collection of subsets such that

$$L_{\lambda_1} \subseteq L_{\lambda_2} \subseteq \ldots \subseteq L_{\lambda_l}. \tag{4.9}$$

The collection of these ordered sets $\mathcal{L} = \cup_\lambda L_\lambda$ is called filtration with respect to f.

Persistent homology tracks the changes in a given filtration. For instance, persistent homology in 0D is concerned with connected components, while the homology in 1D tracks loops. In this study, both 0D and 1D persistent homology are utilized. The threshold value where a topological feature is observed for the first time is called the birth time of that

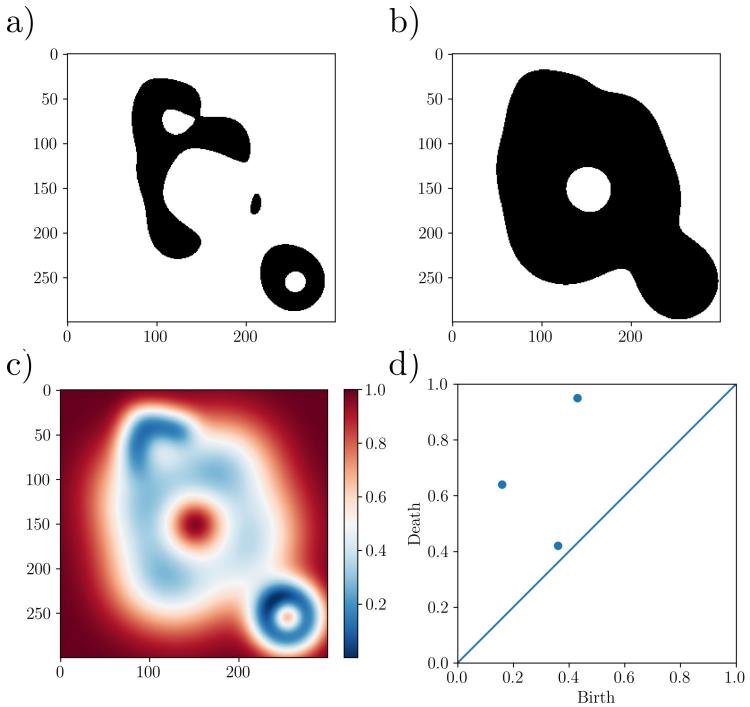


Figure 4.6: a,b) Sublevel sets of the image given in c. d) Persistence diagram of the image shown in c.

feature. When the feature disappears, the corresponding threshold is denoted as the death time of the feature. For instance, a loop can first appear (is born) at threshold λ_i, and it can fill in (die) at λ_j. Pairs of birth and death times for each topological feature are plotted in a persistence diagram (see Figure 4.6d).

Working directly with persistence diagrams is not easy due to their complex structure, and algebraic operations can not be performed simply with persistence diagrams since the number of topological features can be different for each diagram [251]. Therefore, features are extracted from persistence diagrams using their functional summaries. Three methods are employed to featurize the diagrams, namely Carlsson coordinates [71, 69], persistence images [74] and template functions [72]. One can refer to Sections 2.6.4.2,2.6.4.3 and 2.6.4.6 for more details about these approaches.

4.2.3 Results

This section compares the results obtained from feature extraction methods explained in Section 4.2.2. 10-fold cross validation is applied while training and testing the performance of four supervised classification algorithms: support vector machine (SVM), logistic regression (LR), random forest (RF), and gradient boosting (GB). In order to be consistent, the same random state number for cross validation is used to generate the same sets for training and testing for all feature extraction methods.

The performance of each feature extraction method is compared with respect to the accuracy of classification using default parameters for all classification algorithms. The resulting accuracies of each classifier are represented with box plots. Figure 4.7 shows surface profile classification results obtained from traditional signal processing approaches. It is seen that Gaussian Smoothing and peak selection from FFT and PSD plots outperform the FFT method, where the signal is denoised by setting a threshold in its spectrum. There is no significant difference between the results of the four classifiers, as seen in the figure.

The second approach used to extract features from surface profiles is the TDA-based ap-

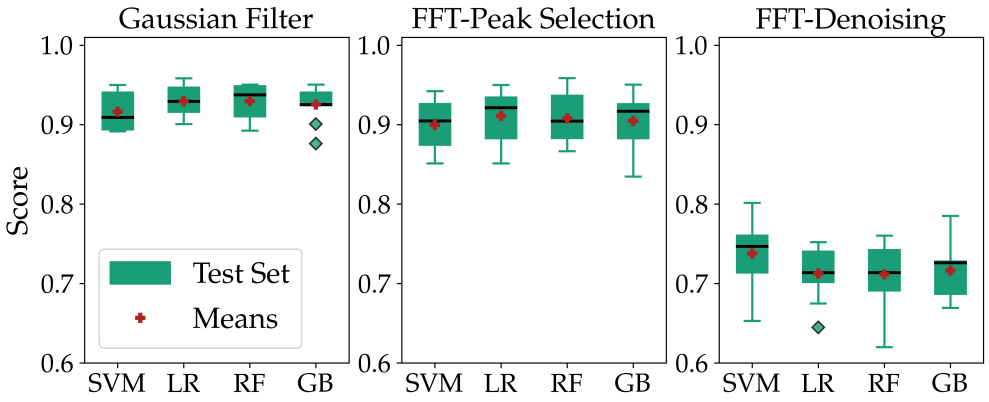


Figure 4.7: Surface profile test set results obtained with the 1D implementation of traditional signal processing tools.

proach. 0D sublevel set persistence is utilized to compute persistence diagrams, and feature extraction is performed using the three methods explained in Section 4.2.2.3. Figure 4.8 provides the resulting test set accuracies for four classifiers. Principal Component Analysis (PCA) is also applied to the features obtained from persistence images. The first 10 components with the highest variance ratios are used to project the feature space onto a 10 dimensional space. The resulting feature matrix is used for classification, and the corresponding accuracies are provided in Figure 4.8. It is seen that all three feature extraction methods give mean accuracies greater than or around 90%. One can notice that this dimensionality reduction does not increase the accuracy of the classifiers for persistence images. The chosen 10 components may not correspond to regions where a Gaussian is placed, so an important descriptor may be eliminated accidentally while reducing the dimension of the feature space. For surface profile data, the feature space dimension is reduced from 320 to 10. This provides an advantage in terms of the time required for classification, and the resulting mean accuracies are still around 90%. Thus, it is worthwhile to apply PCA on persistence image features.

Surface classification results for traditional signal processing tools are provided in Figure 4.9. It is seen that Gaussian smoothing combined with FFT provides the highest scores. However, directly applying FFT in 2D on surface measurements yields poor results for all

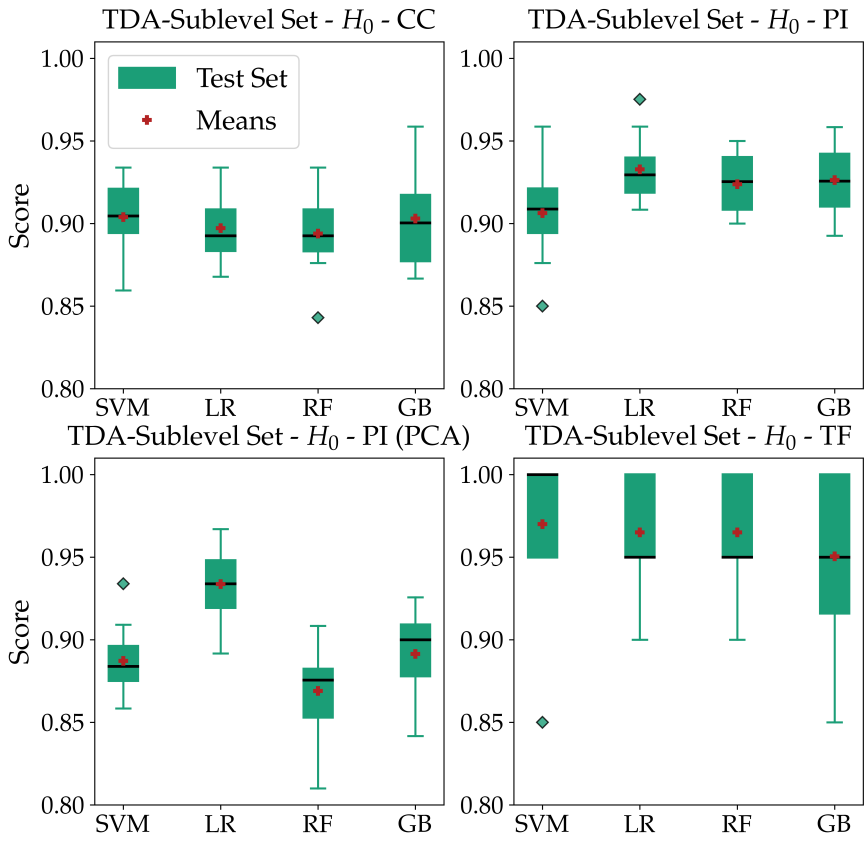


Figure 4.8: Surface profile classification results obtained with OD (H_0) sublevel set persistence. Carlsson Coordinates (CC), persistence images (PI), and template functions (TF) are used to extract features. The first plot in the second row represents the results after applying dimensionality reduction to features obtained from persistence images.

classifiers. This shows the importance of obtaining the roughness component of a given surface. In addition, the results obtained with TDA based approach are provided in Figure 4.10. All feature extraction methods from persistence diagrams yield mean accuracies above 90%. PCA is applied to the feature space of persistence images. Again, the application of PCA does not increase the classification accuracy, but it still provides mean accuracies around 95%. 0D (H_0) and 1D (H_1) persistence provide similar results, so only 1D persistence (H_1) results are provided in Figure 4.10. The highest mean accuracies are obtained by using the template function methods for both of them.

Results of traditional and TDA-based approaches for profile/surface classification are comparable. However, the TDA-based approach provides three main advantages: 1) it

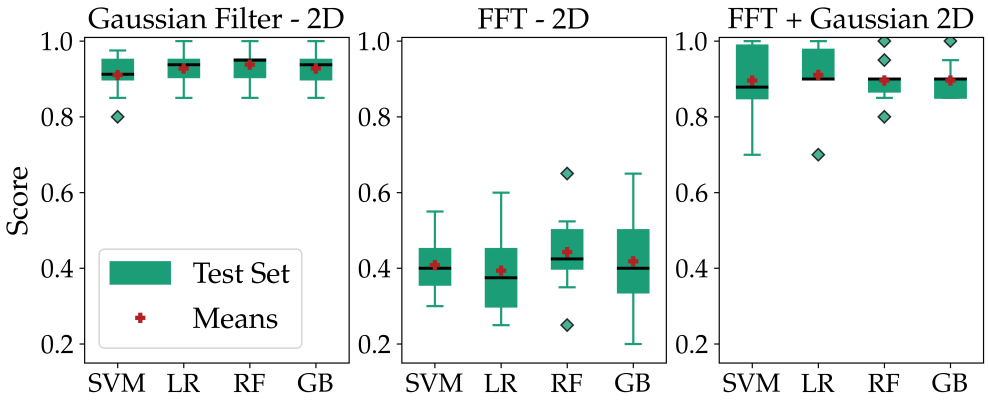


Figure 4.9: Results of surface classification obtained with the 2D implementation of signal processing tools.

requires no parameter selection, 2) it provides an automatic and systematic way for feature extraction, and 3) it allows adaptive feature extraction. Traditional methods do not share all these advantages. For instance, two restriction parameters (MPD and MPH) need to be selected, and a kernel size is needed for FFT/PSD and Gaussian 2D implementations. In addition, the selection of the MPD and MPH or kernel size for Gaussian smoothing requires visually inspecting the spectra. Thus they have low automation potential. These parameters and thresholds are typically selected using only a small portion of the data set. The selected parameters may not be suitable for every surface profile or surface, so traditional signal processing approaches are non-adaptive.

4.3 Automated Surface Texture Analysis via Discrete Cosine Transform and Discrete Wavelet Transform

4.3.1 Data Preprocessing

This study uses both synthetic surfaces (Section 4.2.1), and digital scans of machined surfaces (see Section C). This section only provides information on data preprocessing. For the data collection procedure of the experimental data set and the detailed information of simulation, one can refer to Sections C and 4.2.1.

The raw surface scans include different numbers of pixels with lower gray-level intensity

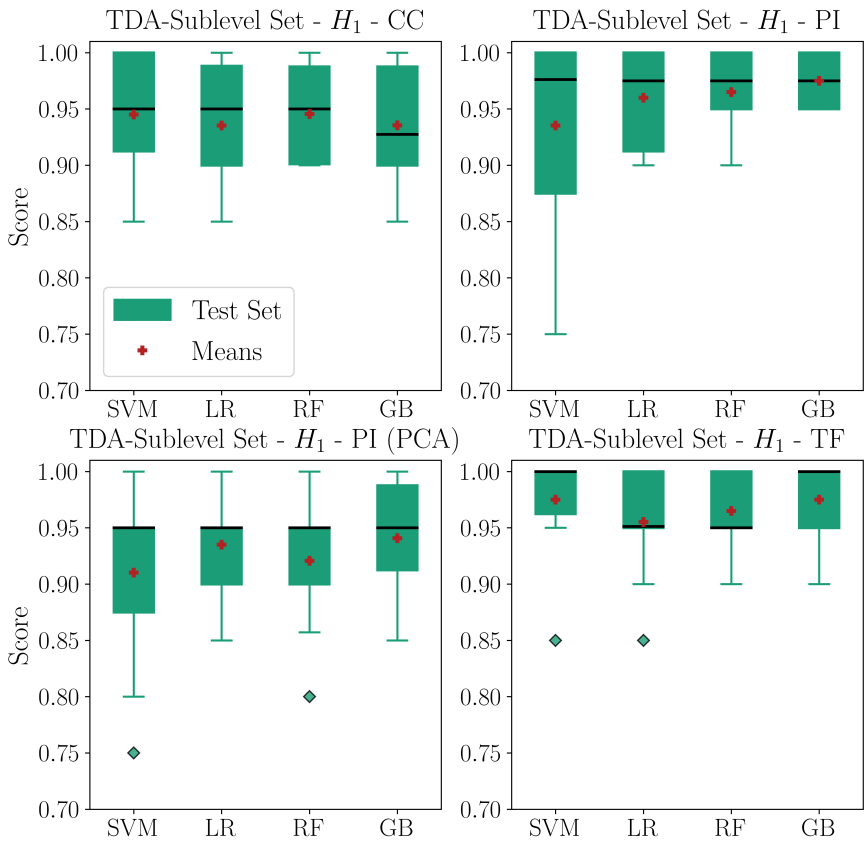


Figure 4.10: Results of surface classification obtained with 1D persistence H_1 using Carlsson coordinates (CC), persistence images (PI), and template functions (TF).

values on the edges of the image, as shown in Figure C.1. These pixels are tedious to isolate manually, so the images are cropped and adaptively removed these pixels using the following algorithm. First, the remainder of image pixel values is found in each direction when they are divided by 1000. The halves of the remainders are used as the number of pixels to remove from each edge. This procedure successfully reduced the number of pixels with lower gray-level intensity values at the boundaries, and the resulting images had similar sizes.

The other challenge was the large dimension of the microscope surface scans, which can exceed 10000 pixels in each direction of the image, thus elevating computational expenses. Consequently, each surface scan is split into 25 sub-images, each with a dimension of 2400×2400 pixels.

Image Subsampling The resulting subimages still presented computational challenges for some signal processing tools such as DCT, where the maximum number of modes is equal to the total number of pixels. Therefore, the images are subsampled to further reduce the number of samples in the subimages when using 2D signal processing tools for surface classification. Several approaches are available for image scaling/resampling in the literature. These also include some signal processing approaches to upscale or downscale an image. One of the simple and widely used subsampling methods is to replace a block of pixels with their average values, and that is the approach used in this study. After testing different sampling factors such as 0.1, 0.2, and 0.5, this study adopted a sampling factor of 0.1 for the experimental data set. This means the size of each block is 10×10 pixels.

4.3.2 Methodology

4.3.2.1 Discrete Wavelet Transform

Discrete Wavelet Transform (DWT) is one of the widely adopted signal processing tools [252, 9, 253, 254, 255]. While a signal's frequency spectrum can only be represented over the entire time domain with Fourier Transform, Wavelet Transform can decompose the signal into components with different time and frequency resolutions [256]. In DWT, the time series is passed through low pass and high pass filters to obtain approximation and detail coefficients. These filters are obtained from a filter bank derived from scaling and translating a wavelet function [252]. The diagram for Discrete Wavelet Transform is provided in Figure 4.11 where the level of the transform is denoted as k. As the level increases, only approximation coefficients are passed through the filters to obtain new approximation and detail coefficients. For example, the approximation coefficient of the first level transform A_1 is passed through the filters, and new approximation and detail coefficients are obtained as AA_2 and DA_2. As the level of the transform increases, the frequency resolution of decomposition increases as well. As seen from Figure 4.11, each component corresponds to a distinct frequency range.

In this implementation, DWT is utilized to analyze simulated and experimental surface profiles. 1D DWT is applied to surface profiles with biorthogonal wavelet functions (BIOR4.4) recommended by the ISO standards [257]. The specified wavelet functions are symmetric, and surface profiles can be reconstructed without any loss [257]. The level of the transform is chosen based on the maximum allowable limit defined by the number of samples in the profile and the type of the wavelet function. Approximation coefficients at the maximum level of the transform are used to obtain the form of the profile [242]. The waviness and roughness profiles of the surface are reconstructed using the detail coefficients of the transform. For instance, DAA_3, DA_2 and D_1 can be used to reconstruct profiles for waviness and roughness when 3 levels of DWT are employed (see Figure 4.11). Profile reconstructed from AAA_3 represents the form of the profile. The separation between waviness and roughness is done heuristically in the literature, and it is believed that there is not any guide on how to select a threshold for detail coefficients. Therefore, this study describes an approach that leverages signal energy to automatically select that threshold.

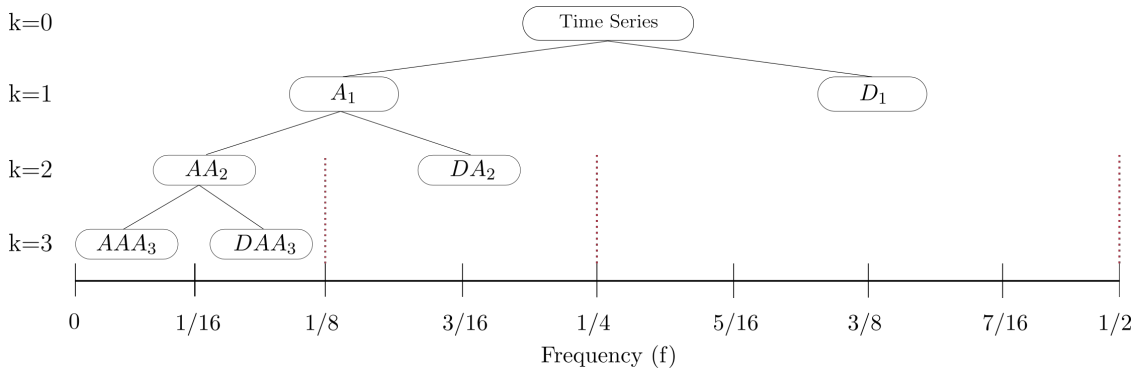


Figure 4.11: DWT tree for the first three level of the transform.

The energy of a discrete signal is defined as $E = \sum_{n=-\infty}^{\infty} \mid x[n] \mid^2$. After applying DWT, profiles are reconstructed using the detail coefficients at all levels and computed their signal energy. Figure 4.12 provides the energy ratio of each detail coefficient and the cumulative energy ratios in addition to the decomposition obtained with automatic threshold selection. The first row of the plots is obtained with the roughest surface, and the ones in the second row belong to the smoothest surface. Both surfaces have a size of 4096×4096 pixels. Three

cross-sections are taken in each direction of the surfaces represented with Profile-x (or y) i in the figure.

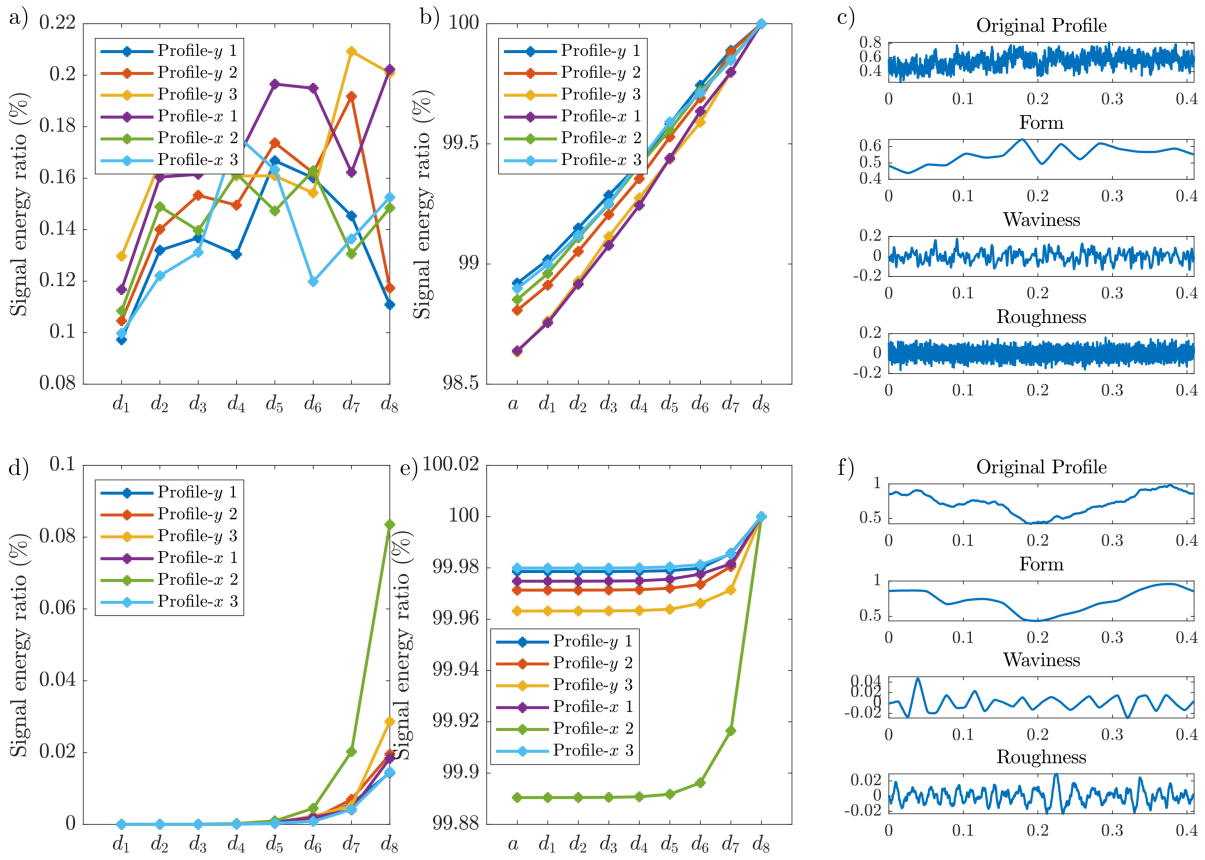


Figure 4.12: The plots on the first row belong to the roughest surface ($H = 0$) simulation, while the plots in the second row belong to the smoothest surface ($H = 1$) a) Energy ratios of detail coefficients (d_i: detail coefficients of level i). b) Cumulative energy ratios including approximation coefficients (a: approximation coefficients of maximum level). c) The resulting three main components after applying the automatic threshold selection for profile $x - 1$.

The cumulative energy ratio plot is given in Figs. 4.12b and 4.12e for the smoothest and the roughest surface of the simulation. It is seen that most of the energy is accumulated in approximation coefficients. Since the main aim is to separate waviness and roughness, this study only takes into account the energy ratios of detail coefficients. In Figure 4.12a, it is hard to notice the significant increase in signal energy as the level of the transform increases. Therefore, the differences between consecutive energy ratios are used. For example, the biggest difference between consecutive energy ratios for Profile-y 1 in Figure 4.12a is between

levels 4 and 5. The proposed algorithm uses detail coefficients of the first four levels to reconstruct roughness. The rest of the detail coefficients are reconstructed and summed up to obtain a waviness profile. The resulting surface components are shown in Figure 4.12c. When the roughness of the surface decreases, the threshold is easier to capture since a dramatic increase is seen in signal energy between levels 7 and 8 (see Figure 4.12d) for most of the profiles. In this case, roughness is equal to the summation of reconstructed signals from detail coefficients of the first seven levels, and the 8th level coefficients are used to generate the waviness profile. In Figure 4.13, energy plots of the reconstructed signals are provided. Cumulative energy plots show that the signals obtained using approximation coefficients have the highest energy. It is evident that the form profile, which is obtained from approximation coefficients, has the largest amplitudes in the decomposition. The significant increase in energy occurs between levels 7 and 8. Therefore, for this example, the threshold is selected at level 7. This approach is applied to all profiles extracted from the sub-images in the experimental data, and the features are computed accordingly.

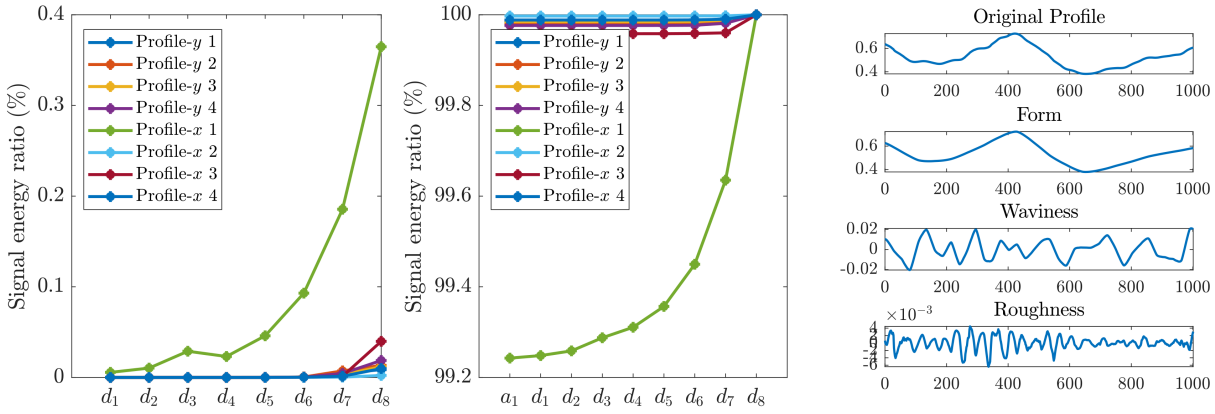


Figure 4.13: (left) Energy ratios of detail coefficients (d_i: detail coefficients of level i). (middle) Cumulative energy ratios including approximation coefficients (a: approximation coefficients of maximum level). (right) The resulting three main components after applying the automatic threshold selection for profile $x-1$.

4.3.2.2 Discrete Cosine Transform

Discrete Cosine Transform (DCT) decomposes a signal into cosine functions with different frequencies. It is similar to Fourier Transform, except that it uses only real coefficients. There are several types of DCT, and in this implementation, Type II DCT is used. The definition of 2D transform is given as [258]

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_{mn} cos\left(\frac{\pi(2m+1)p}{2M}\right) cos\left(\frac{\pi(2n+1)q}{2N}\right), \tag{4.10}$$

where

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}} & p = 0 \\ \sqrt{\frac{2}{M}} & 1 \leq p \leq M-1 \end{cases} \quad \text{and} \quad \alpha_q = \begin{cases} \frac{1}{\sqrt{N}} & q = 0 \\ \sqrt{\frac{2}{N}} & 1 \leq q \leq N-1 \end{cases}.$$

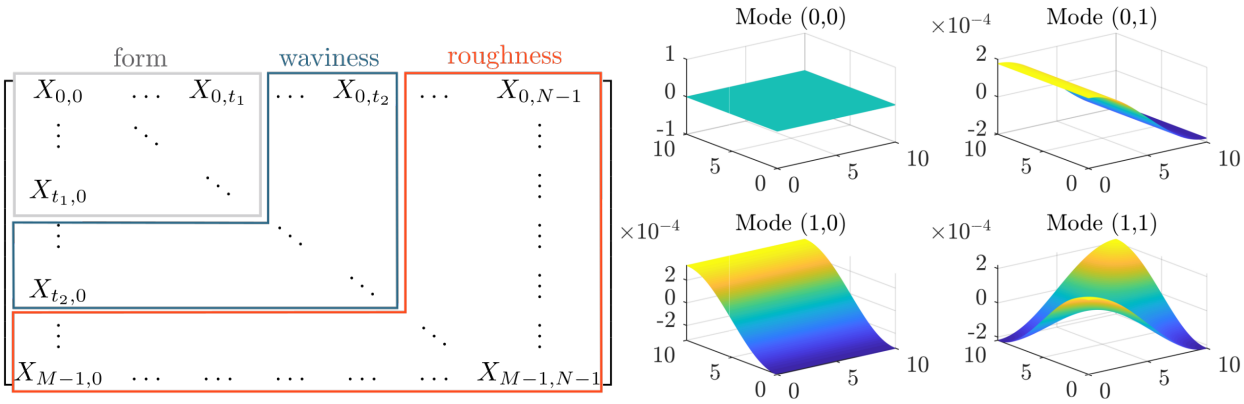
M and N are the numbers of elements in each direction of the image, and I_{mn} represents the image. The inverse transform is provided as

$$I_{ij} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} cos\left(\frac{\pi(2i+1)p}{2M}\right) cos\left(\frac{\pi(2j+1)q}{2N}\right). \tag{4.11}$$

If Eq. (4.10) is inserted into Eq. (4.11), the resulting expression is

$$A_{ij} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q \left(\alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} cos\left(\frac{\pi(2m+1)p}{2M}\right) cos\left(\frac{\pi(2n+1)q}{2N}\right)\right) cos\left(\frac{\pi(2i+1)p}{2M}\right) cos\left(\frac{\pi(2j+1)q}{2N}\right) \tag{4.12}$$

Basis functions (modes of a given image) are defined with indices p and q. Summing all the modes recovers the original image as shown in Eq. (4.12). In Figure 4.14a, the modes are represented by $X_{i,j}$. For instance, $X_{0,0}$ represents the first mode of a given surface. The first four modes of a synthetic surface are provided in Figure 4.14b. Form, waviness, and roughness components are obtained by summing the modes inside the boxes shown with grey, blue, and red colors, respectively. Figure 4.14a shows that two thresholds t_1 and t_2 need to be selected to separate the three components. Since the goal is to find the surface roughness component, t_1 is neglected, and form and waviness are treated as one combined



(a) DCT modes in matrix format (b) The first four modes of a synthetic surface.

Figure 4.14: Matrix format of the DCT modes and example modes.

component. This study then introduces a new approach based on image entropy to generate an automatic threshold selection for DCT, as described in the next section.

Threshold Selection Using Image Entropy: Information entropy is known as Shannon's entropy [259] and it is defined as

$$H(X) = \sum_{i=1}^{n} p_i log(1/p_i), \tag{4.13}$$

where X is a discrete random variable with probability distribution p. An analogous expression is obtained for computing the image entropy [260] according to

$$H(I) = \sum_{i} (h_I(i)/N) log(N/h_I(i)), \tag{4.14}$$

where $h_I(i)$ is the counts in histogram of the grayscale image I, and N is the number of bins the histogram.

As mentioned earlier, two components are computed for a surface: the roughness component and the surface that includes waviness and form components. Initially, t_2 is set to 0. The first mode $X_{0,0}$ combines the form and waviness components of the surface, while the rest of the modes represent the roughness component. Combining form and waviness also gives the advantage of avoiding additional mode computation for the surface. The roughness component can be obtained by simply subtracting the first component (form+waviness) from

the original image. After obtaining both components, the next step is to compute their image entropies. Then, the threshold index t_2 is increased by one. Now, the first four modes shown in Figure 4.14b represent the first component (form+waviness). The roughness component is obtained by subtracting the first component from the original image, and image entropy for both components is computed. This procedure is iterated by increasing the threshold index t_2. This process is repeated for all threshold indices from 0 to 256 for the roughest surface in the data set, and entropy curves are obtained for both components as shown in Figure 4.15.

Figure 4.15 shows that the entropy of the first component (form+waviness) is increasing dramatically for a small change of t_2. After a certain value of the threshold, its rate of increase slows down, as seen from its derivative curve. Therefore, a threshold value is selected for the slope of the entropy curve of the first form+waviness component. Two threshold values

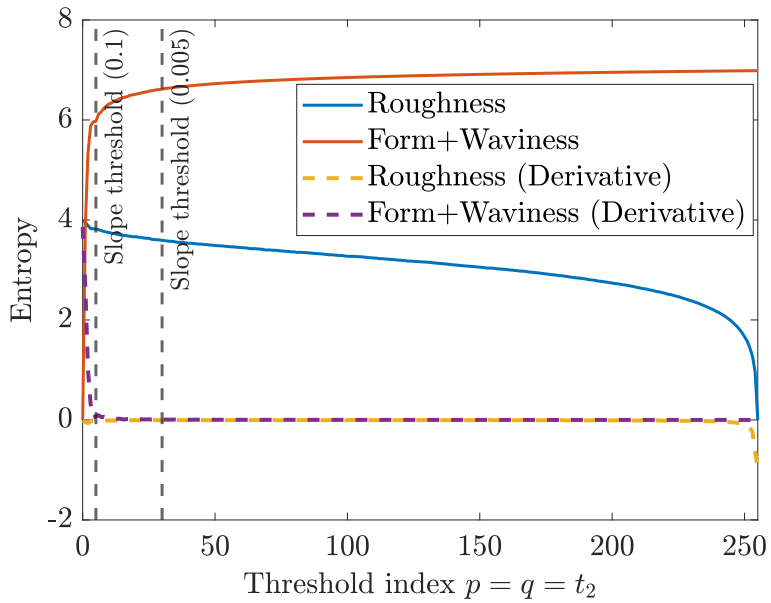


Figure 4.15: The entropy of the roughness and waviness+form surface for varying threshold index.

are tested, and these are 0.1 and 0.005 (see Figure 4.15, grey vertical lines). When the slope of the curve is under these slope thresholds, their corresponding t_2 values are taken as thresholds to separate the roughness component. For these two slope thresholds, 0.1 and

0.005, t_2 is found as 5 and 30, respectively. Then, two components of the surface are obtained using these threshold values as shown in Figure 4.16. The figure shows only surface profiles to clearly illustrate the difference between the components. It is seen that threshold 30 is more reasonable to use since the form+waviness component obtained with $t_2 = 5$ does not provide a good approximation to the surface. Therefore, it is decided to use a slope threshold of 0.005 for all surfaces in the data set.

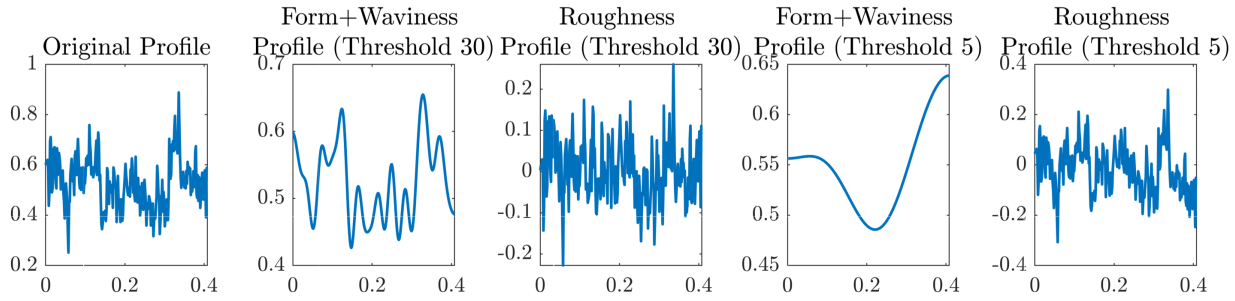


Figure 4.16: Reconstructed surface profiles obtained using thresholds from the entropy analysis shown in Figure 4.15.

An automatic threshold selection algorithm is defined such that it can terminate the loop when the slope of the entropy curve of the form+waviness component is below 0.005. For instance, only the first 30^2 modes of the surface are computed instead of computing all 256^2 modes, and profiles of the reconstructed surfaces are given in Figure 4.16. Therefore, the proposed algorithm avoids unnecessary and expensive mode computations. This algorithm is applied to both synthetic and experimental data sets to obtain roughness components that can be used to compute the 2D features needed for applying machine learning.

4.3.2.3 Feature Extraction from Surfaces

This section describes the feature extraction procedure from surfaces. The roughness component of each surface or profile is used to extract features. Depending on the type of data used in the analysis, the profile or areal parameters given in Reference [246, 245] are used. While working with surface area measurements, the height and hybrid parameters provided in Sections 4.2 and 4.4 of Reference [245] are computed for roughness components

of the surfaces. For surface profiles, height, spatial and hybrid parameters given in Secs. 4.1-4.2 of [246] are computed. The list of features and their definitions are provided in Table 4.1. In addition, definitions for all the features are given in the continuous form. The discrete form of these equations can be found in References [245] and [246]. Then, these feature matrices are fed to supervised classification algorithms, namely, Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), and Gradient Boosting (GB). This study uses the default parameters of the classifiers.

Table 4.1: The features used in the classification of surfaces and surface profiles. $f(x)$ and $f(x,y)$ represent surface profile and surface, respectively.

1D Features	2D Features
$Rq = \sqrt{\frac{1}{l_m}\int_{l_m} f(x)2dx}, \quad Rsk = \frac{1}{R_q^3}\frac{1}{l_m}\int_{l_m} f(x)^3 dx$	$Sq = \sqrt{\frac{1}{A}\iint_{\tilde{A}} f^2(x,y)dxdy}$
$Rku = \frac{1}{R_q^4}\frac{1}{l_m}\int_{l_m} f(x)^4 dx, \quad Rt = \max(f(x)) + \lvert\min(f(x))\rvert$	$Ssk = \sqrt{\frac{1}{AS_q^3}\iint_{\tilde{A}} f^3(x,y)dxdy}$
$Ra = \frac{1}{l_m}\int_{l_m} \lvert f(x)\rvert dx$	$Sku = \frac{1}{AS_q^4}\iint_{\tilde{A}} f^4(x,y)dxdy$
$Ral = \min_{t_x \in R} t_x$, where $R = \{t_x : \mathrm{ACF}(t_x) < s\}$	$Sp = \max(f(x,y))$
$Rsw = \frac{2\pi}{arg\max_p \lvert F(p)\rvert}, \quad Rdt = \max_{x\in R} \lvert\frac{dz(x)}{dx}\rvert$	$Sv = \lvert\min(f(x,y))\rvert$
$Rdq = \sqrt{\frac{1}{l_m}\int_{l_m}\left(\frac{df(x)}{dx}\right)^2 dx}$	$Sz = Sp + Sv$
$Rda = \frac{1}{l_m}\int_{l_m} \lvert\frac{df(x)}{dx}\rvert dx$	$Sa = \frac{1}{A}\iint_{\tilde{A}} \lvert f(x,y)\rvert dxdy$
$Rdl = \int_{l_m}\left(\sqrt{1+\left(\frac{df(x)}{dx}\right)^2}\right)dx$	$Sdq = \sqrt{\frac{1}{A}\iint_{\tilde{A}}\left[\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2\right]dxdy}$
$Rdr = \frac{1}{l_m}\int_{l_m}\left(\sqrt{1+\left(\frac{df(x)}{dx}\right)^2} - 1\right)dx$	$Sdr = \frac{1}{A}\iint_{\tilde{A}}\sqrt{\left[1+\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2\right]} - 1\Big)dxdy$

4.3.3 Results

This section presents the results obtained with automatic threshold selection algorithms introduced in Section 4.3.2, and compares the results to the ones obtained from the Gaussian filter. Both algorithms are applied to the synthetic and experimental data sets described in Section 4.3.1, and the surfaces and their profiles are classified. There are three labels for synthetic surfaces representing the surface's roughness level. For experimental data, samples come from three main machining operations, each of which corresponds to three different surface roughness ranges. Therefore, there are nine labels for the experimental

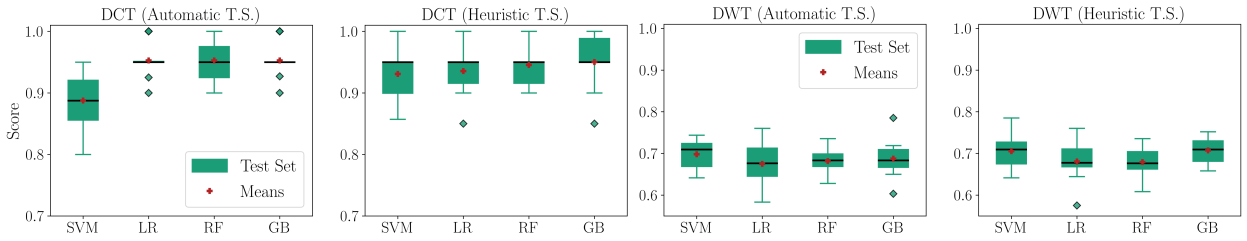


Figure 4.17: Classification accuracies obtained with automatic and heuristic threshold selection for DCT and DWT using a synthetic data set. Three-class classification is performed in this case.

data set. Three-class classification is also performed by only taking into account the type of the machining operation. In addition to using the automatic threshold selection algorithms, surfaces are decomposed into form, waviness, and roughness heuristically. Specifically, the decomposition of a few surfaces or profiles is inspected, and a decision is made on the threshold values mentioned in Section 4.3.2.1 and 4.3.2.1. These thresholds are then fixed and used for the whole data set. The resulting roughness surfaces or profiles are used to extract features as explained in Section 4.3.2.3, and supervised classification is performed.

Figure 4.17 provides the figures for the classification results obtained from DCT and DWT using automatic and heuristic threshold selection. It is seen that there is no significant difference between the classification accuracies of automatic threshold selection and heuristic threshold selection. It is seen that the automatic threshold algorithm helps to decrease the deviation in the results obtained with LR and GB classifiers for DCT. One should note that heuristic threshold selection requires manual inspection of several surfaces to decide the threshold value. Depending on the number of surfaces inspected, the manual process can add a significant amount of time for identifying the roughness level. In contrast, automatic threshold selection algorithms for both DWT and DCT do not require manual inspection since only a slope threshold needs to be selected, as explained in Section 4.3.2.2.

The first row of the figure provides plots for 3-class classification where the labels are milling (M), profiling (P), and shaped or turned (ST). The second row contains the results obtained with the nine-class classification, where each machining operation has three dif-

ferent roughness values. It is seen that DCT automatic threshold selection provides similar accuracies compared to the heuristic approach. Nine-class classification is also performed by assigning distinct labels to the main surfaces shown in Figure C.2b. Figure 4.18 shows that nine-class classification provides better classification scores. DCT results obtained using the automatic threshold selection approach are slightly lower when they are compared to the heuristic approach. However, nine-class classification with automatic threshold selection for DWT outperforms the heuristic threshold selection (see Figure 4.18). Nevertheless, three-class classification with the heuristic approach of DWT provides better accuracies. This can be explained by the selected threshold being suitable for a large portion of the data set. Since the threshold selection is highly dependent on the person who performs the manual inspection, the heuristic approach results may not be consistent.

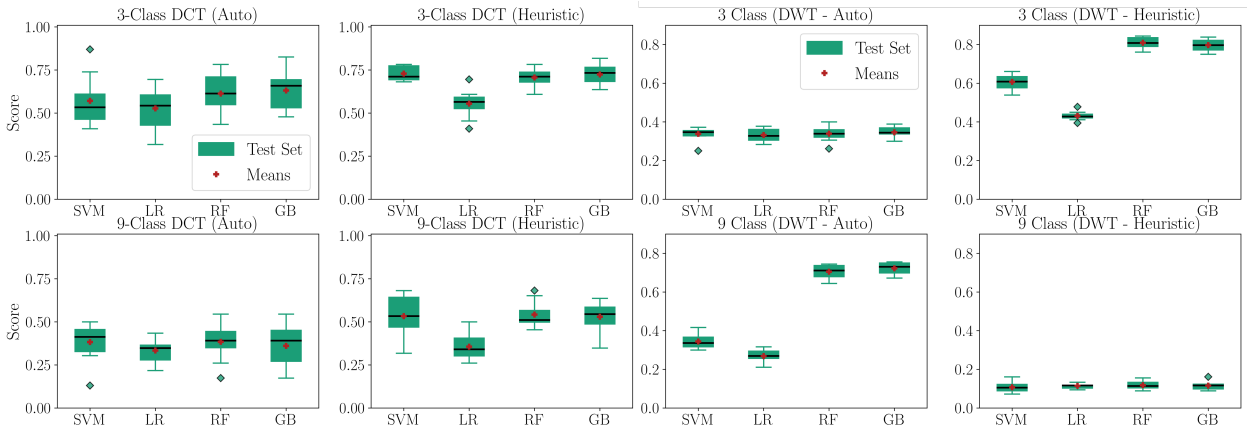


Figure 4.18: Experimental data results for DWT (profile classification) and DCT (Surface classification).

Manual threshold selection for DWT and DCT is performed using visual inspection. One may choose a large value for the threshold depending on the size of the given surfaces. In this case, the thresholds of DCT are selected as 50 for both synthetic and experimental data sets, while thresholds of DWT were chosen as 2 and 4 for synthetic and experimental data sets, respectively. Figure 4.20 provides the roughness components of three surfaces obtained from the heuristic approach and the proposed automatic threshold selection algorithms. It is seen that roughness profiles obtained from the heuristic approach have smaller amplitudes

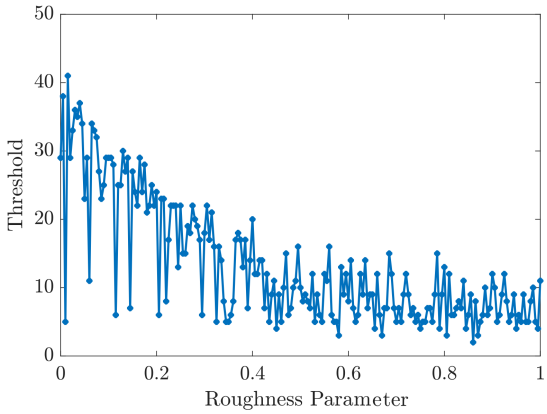


Figure 4.19: The thresholds selected by proposed algorithm for the synthetic data set. The roughness parameter 0 represents the roughest surface, while the smoothest surface has a roughness parameter of 1.

compared to the ones obtained from the automatic threshold selection approach. This is because of the fact higher thresholds remove more modes from the main surface, and this leads to less number of modes for the roughness component. In addition, the selected thresholds are less than the heuristic thresholds value, which is kept constant for all surfaces in the experimental data set. Figure 4.21 provides the selected threshold values from the automatic selection algorithm and the constant heuristic threshold.

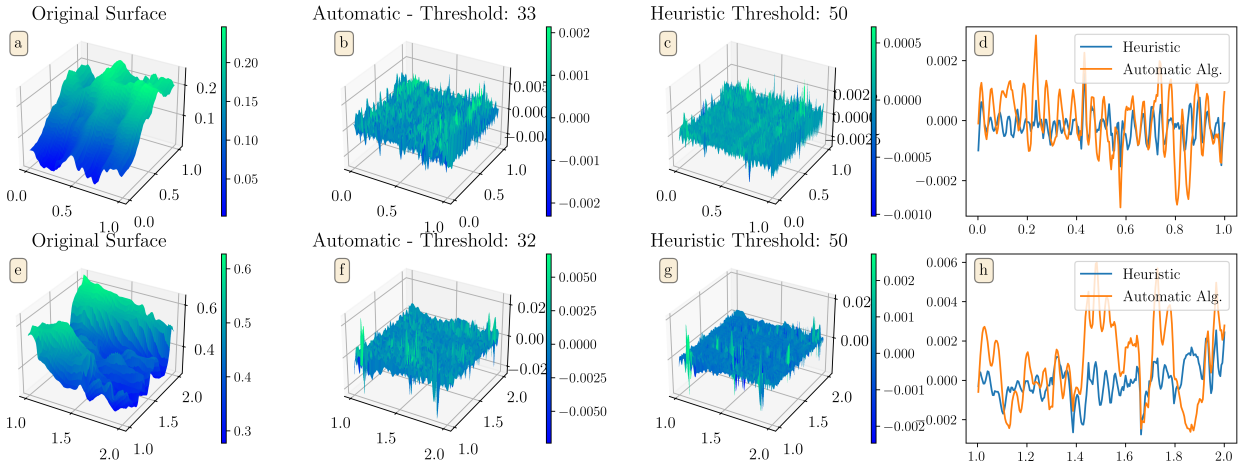


Figure 4.20: Roughness surfaces and roughness profiles obtained from three surfaces in the experimental data set. (a-d) Milling, (e-h) Profiled.

Proposed threshold selection algorithms remove the manual inspection and make the

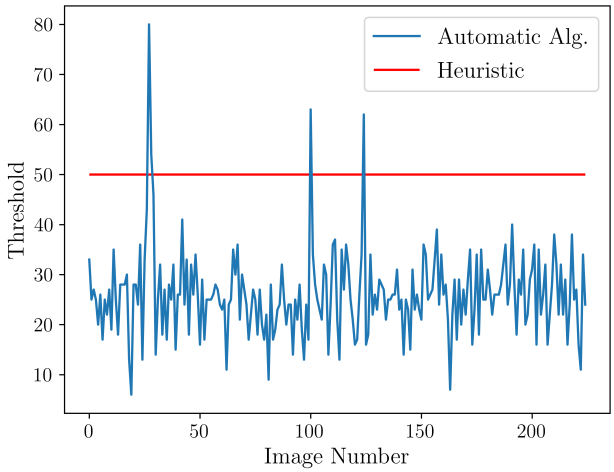


Figure 4.21: Threshold values selected from the automatic threshold selection algorithm and the constant heuristic threshold.

decomposition fully automatic. For DCT, the proposed approach provides a significant reduction in computational time. Since the threshold is selected as 50 for the heuristic DCT approach, 50^2 surface modes need to be computed for each surface in the data set. However, the automatic threshold selection algorithm picks different values of the threshold depending on the surface. Figure 4.19 provides the selected threshold values for the synthetic surfaces for varying roughness parameters. It is seen that the maximum selected threshold is nearly 40. That means that the algorithm computes 40^2 modes at maximum for the corresponding surface. Compared to the number of mode computations performed with the heuristic threshold, the proposed algorithm saves the time needed to compute 900 modes. The time reduction is even more significant with smoother surfaces with a higher value of roughness parameter H. In this case, 10^2 modes are computed instead of 50^2. Thus this shows that automatic threshold selection avoids redundant mode computations and dramatically decreases the computational time in comparison to the heuristic threshold selection.

The experimental results for DWT and DCT are compared to those obtained from Gaussian Filtering, another tool widely adopted in digital image and signal processing. Figure 4.22 provides classification scores for the Gaussian Filter applied to surface profiles (1D) and surfaces (2D) from the synthetic data set. Figure 4.22 shows that the highest classification

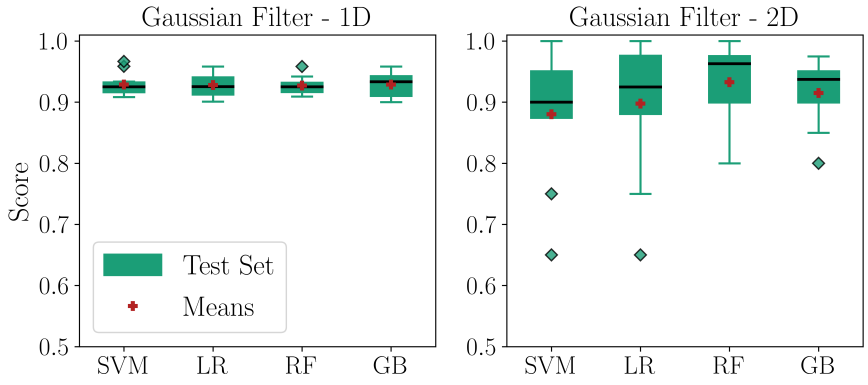


Figure 4.22: Gaussian filtering results obtained from profile and surface classification for synthetic data set.

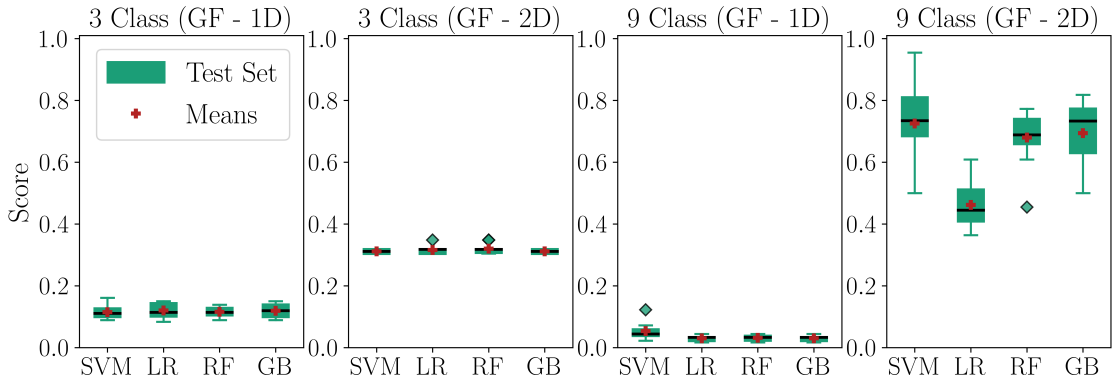


Figure 4.23: Three and nine class classification results for surface profiles (1D) and the surfaces (2D) in experimental data using Gaussian Filter.

accuracy is obtained with Gaussian Filtering. However, the results for experimental data do not show the same trend. Figure 4.23 shows that three-class classification does not perform well. In addition, the nine-class classification for profiles does not perform well either. This can be explained by the fact that the selected number of profiles from the surfaces is not enough to extract texture information. Subsampling is not applied for profiles, and 8 profiles are extracted from the images whose sizes are 2400×2400. Therefore, while more profiles may increase the score, exploring that direction is out of the scope of this work. Proposed approaches for DWT provide better accuracies for both profile classification in three and nine-class classification than the ones obtained from Gaussian Filtering (see Figure 4.18 and Figure 4.23). Nine-class classification for surface classification provides mean accuracies

around 70%, which is higher than the results of DCT shown in Figure 4.18. However, the Gaussian filter, when applied in 2D, is parameter-dependent, and the same parameter (kernel size) is used for the whole data set in this study. The same wavelet function is used for all surface scans, so the algorithm for DWT is parameter independent, while the algorithm for DCT only takes the slope threshold from the user. This threshold value can be set to a value near zero and can be used for the whole data set. However, the threshold selection is still data-driven, and it depends on the entropy curve of the given surface (see Figure 4.15).

4.4 Surface Finish Monitoring Using Persistent Homology

4.4.1 Topological Saliency and Simplification

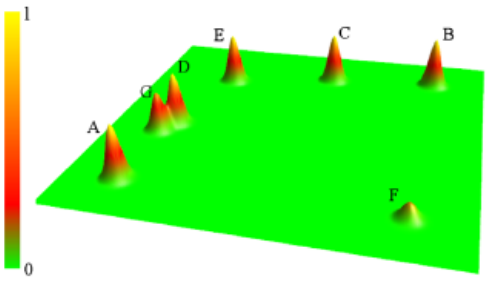


Figure 4.24: Motivation example given in Reference [1].

The presence of noise can make it challenging to distinguish intrinsic surface features from noise. Therefore, an elimination needs to be performed on the given surface to delineate between surface features and noise. Topological approaches have been successfully utilized to extract and identify the importance of features in images, as shown in Section 4.2.3. One of these tools is topological saliency [1] which was introduced to capture the relative importance of a topological feature relative to other features within its neighborhood. The original applications of topological saliency included key feature extraction, scalar field simplification, and feature clustering. However, this study leverages, for the first time, topological saliency to propose a novel digital-twin for surface treatment.

Topological saliency is proposed to capture the features which are eliminated due to

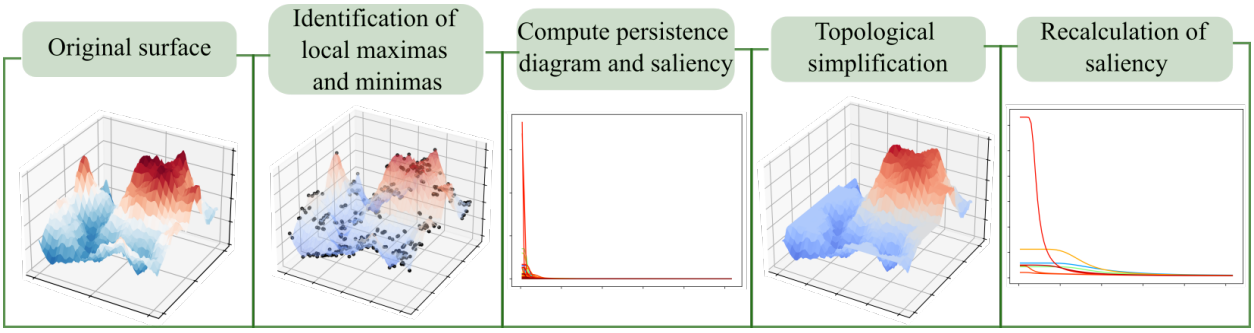


Figure 4.25: Steps to perform topological saliency-based simplification

their low persistence [1]. Doraiswamy et al. provided an example to explain this in more detail (see Figure 4.24). In this example, it is seen that there are seven significant features for the given scalar data, and each feature represents a peak. If topological persistence-based simplification is used to rank the features, the peak with the label F will have the lowest rank, or it will be eliminated since its persistence is the smallest. However, there are no significant peaks around feature F, and it dominates its neighborhood. Therefore, topological saliency has been suggested to evaluate the significance of the features instead of using their topological persistence [1]. Figure 4.25 provides block diagrams that show the steps for topological saliency-based simplification. The first step is to define the critical points of the surface, which are local minima and maxima. Users can either select local minimums if the main focus is on the valleys or local maximums if the user wants to work on peaks. Let assume that $C = c_1, c_2, \ldots, c_t$ is the set of local minimas of the given surface $f : \mathbb{M} \to \mathbb{R}$, where $\mathbb{M}$ is the $d - manifold$. Sublevel set persistence is utilized to compute persistence diagrams of the given function, then the difference between birth and death times of the features located at critical points is called the persistence $P(i)$ of that feature. The definition of topological saliency is given as [1].

$$T_r(i) = \frac{w_i^i P(i)}{\sum_{cj \in C} w_j^i P(j)} \tag{4.15}$$

where r is the radius of neighborhood chosen around each feature c_i, and w_j^i represents the weight of the feature j with respect to feature i. There are two main weighting functions: 1)

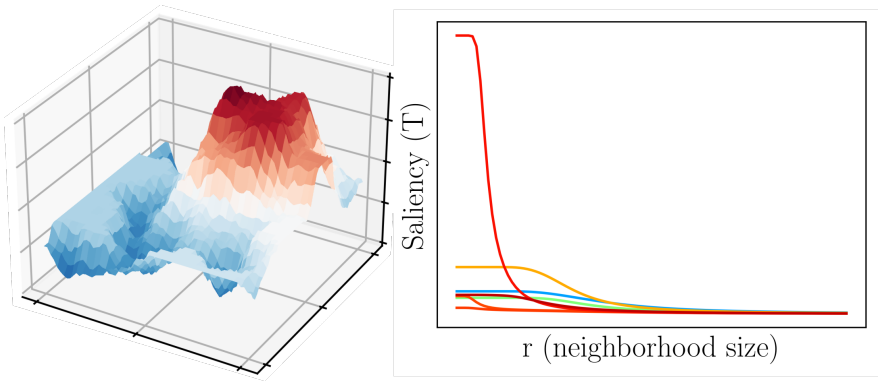


Figure 4.26: Topological saliency plot for a simplified synhetic surface.

Uniform weighting, 2) Gaussian weighting, and their definitions are given as

$$w_j^i = \begin{cases} 1, & \text{if} \quad c_j \in N_r(i) \\ 0, & \text{otherwise} \end{cases} \tag{4.16}$$

$$w_j^i = e^{-d_g(c_i,c_j)^2/r^2}, \tag{4.17}$$

where $d_g(p,q)$ represents the geodesic distance between two critical points. Doraiswamy et al. suggested using the Gaussian weighting function [1]. For a given surface f, the saliency of a feature is computed for varying neighborhood radius r. Then, topological saliency plots are obtained, as shown in Figure 4.26. It is seen that the topological saliency of features decreases as the neighborhood size increases. This is because of the fact more features appear as the radius increases.

Elimination based on persistence, which is the difference between birth and death times of the features, can eliminate the points with larger saliency, as shown in the motivation example in Figure 4.24. To circumvent this issue, Doraiswamy et al. proposed a saliency-based simplification. They fixed the neighborhood size, r, and then a saliency threshold was selected to eliminate the features under the threshold. This threshold is generally fixed to a value which is found by multiplying the maximum saliency for the corresponding r value by a ratio. After the elimination of features (peaks/valleys), the saliency of the remaining features is recalculated since simplification provides a new surface. This process is repeated until the desired number of features remains.

4.4.2 Results

This study introduces a pipeline that can simplify a given surface based on topological saliency. Users can define the number of features (peaks/valleys) they want after the simplification. Figure 4.27 presents the simplified surfaces and their corresponding saliency plot. The plots in the first column represent the results obtained from the original synthetic surface. The number of features decreases during the simplification process. The neighborhood size is fixed in simplification. Then, the algorithm automatically finds the saliency threshold for the corresponding neighborhood size r. The saliency threshold is calculated by multiplying the maximum saliency by 0.15. The algorithm automatically removes the features with saliency value under that threshold, and a simple surface is obtained. Compared to traditional signal decomposition approaches such as DCT and DWT, the user does not select any thresholds that have a significant effect on the resulting decomposition. They only need to identify the number of features of the simplified surface. In addition, topological simplification for the example provided in Figure 4.27 is computed in ≈ 0.5 seconds. The image in Figure 4.27 has a dimension of 64×64. It is expected to have longer runtime if the pixel size increases. However, it is believed that all simplification processes can still be computed in less than a second by implementing parallel computing into the proposed algorithm.

Surface patches can be clustered based on topological saliency. In this case, the feature similarity between peaks/valleys is used. The similarity of the two features is defined with respect to the area between their saliency plots. The smaller area between the plots represents the likelihood of similar features. The areas between saliency plots of each feature are computed. Then, these areas are used as distances to generate a similarity matrix. Precomputed similarity matrices are given to the AgglomerativeClustering algorithm to cluster the features. This clustering of critical points/features needs to be generalized to all surface points to obtain surface patches. In this step, the critical points are treated as cluster centers, and the pairwise geodesic distances between all critical points and the surface points are computed. Each surface point is assigned a cluster number based on its closest critical

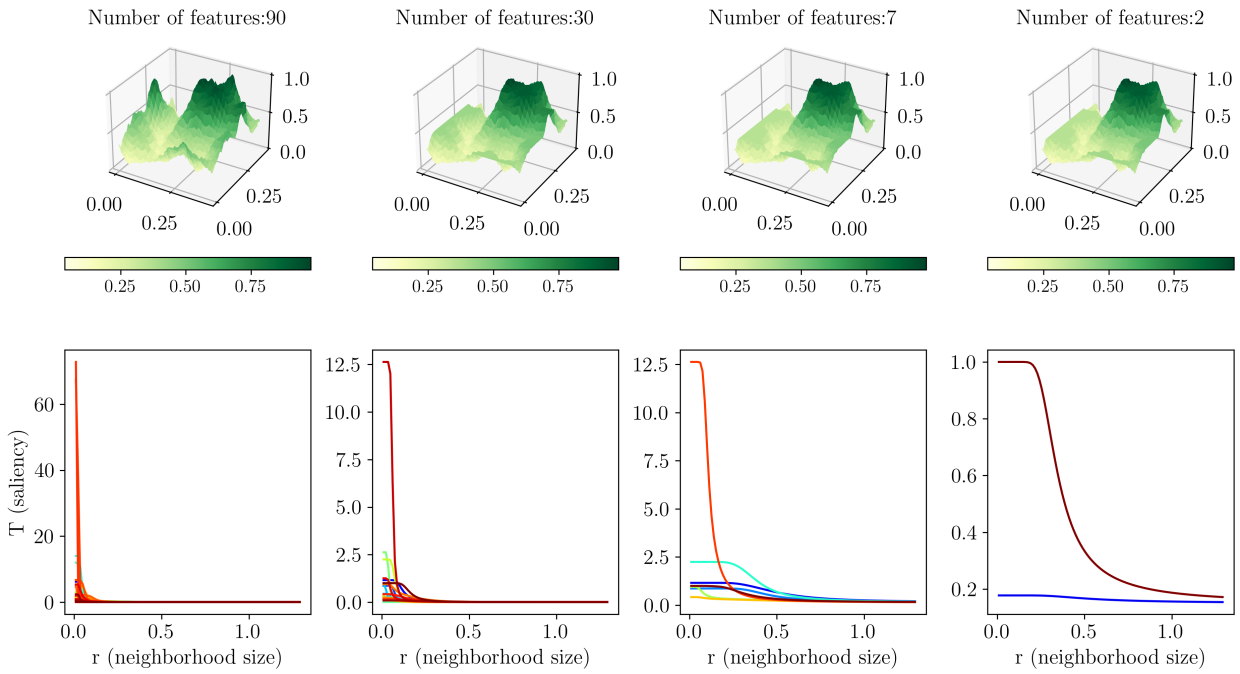


Figure 4.27: Saliency based simplification on an example synthetic surface using approximated geodesic distances between the critical points.

points. This step can be considered as KNN classification where $K = 1$. Labeling of the surface points provides surface patches. This clustering is applied on the example surface and its simplified versions provided in Figure 4.27. Figure 4.28 provides clustered surfaces based on the saliency of their critical points. It is seen that large peaks can be clustered with different surface patches for varying heights. In this case, users can also select the number of clusters so that they can identify what part of the surface will be machined.

Topological saliency provides automatic surface simplification, and it eliminates the surface decomposition as in the case of traditional approaches. In addition, the clustering with topological saliency can provide regions where more machining is required to obtain desired surface finish.

4.5 Conclusion

This chapter introduces three main approaches for surface texture analysis: 1) feature extraction from surfaces using TDA, 2) proposing automatic threshold selection algorithms

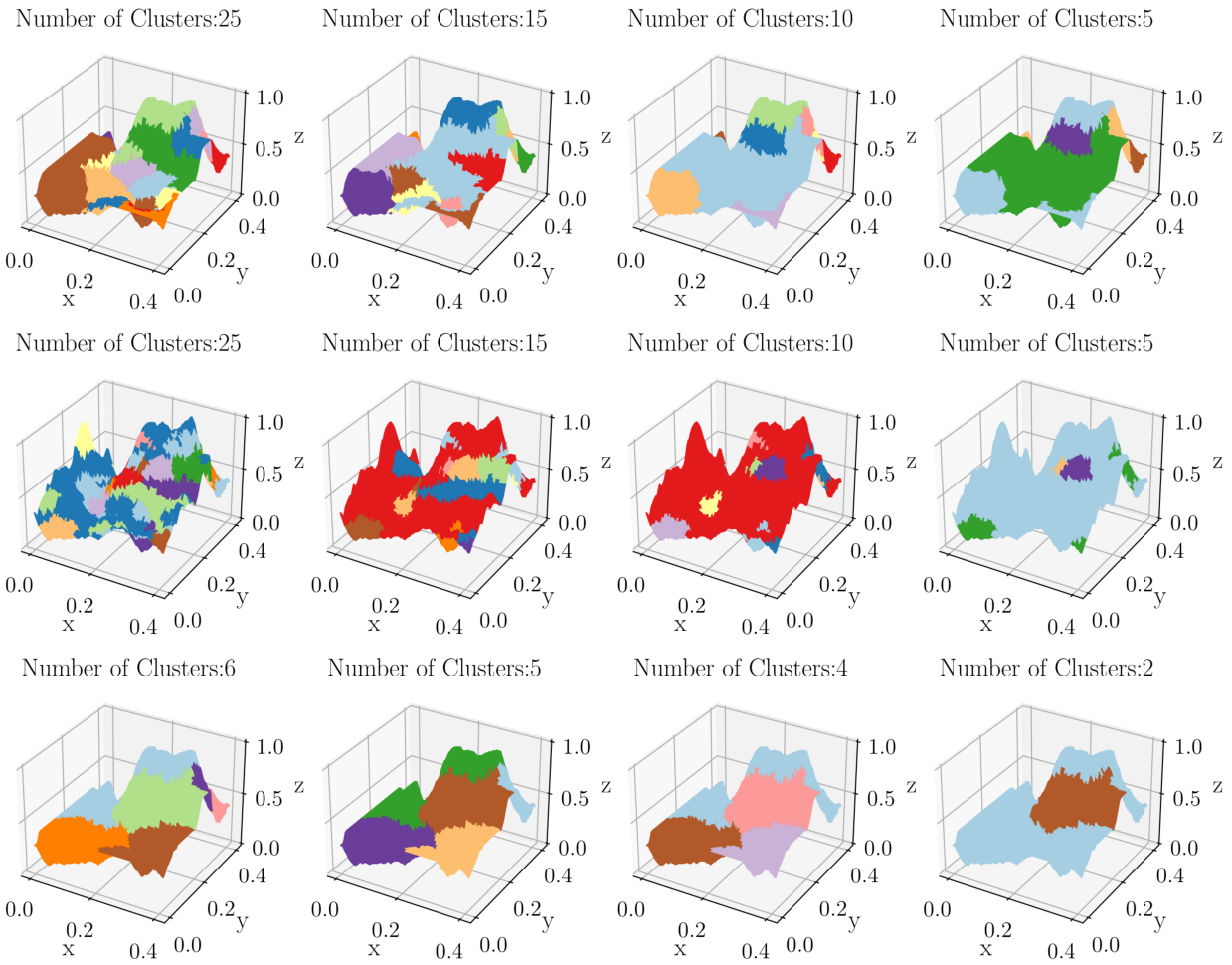


Figure 4.28: Saliency-based clustering of surfaces. The first row represents the clustering results of the original surface with a feature number of 90 for different cluster numbers. The second and third rows represent the results for the simplified surface at the second iteration (feature number: 30) and third iteration (feature number: 7) given in Figure 4.27.

for DCT and DWT, and 3) clustering surface patches using topological saliency and simplification.

In Section 4.2, this study proposed an automatic and adaptive feature extraction approach to determine the level of roughness in profile and areal measurements of surfaces. The proposed approach yields similar classification accuracies for surface and profile classification, and it eliminates the manual preprocessing which is required by traditional signal processing tools. All of the results in this study are obtained using the default parameters of the classification algorithms. Further parameter tuning for each classifier can result in better

classification accuracies with smaller deviations. In addition, TDA-based approaches can be computationally expensive as the number of pixels in images or the number of samples in surface profiles increases. Therefore, one direction of the future work of this study can include parameter tuning and optimizing TDA-based approaches to speed up the computation, as well as applying the proposed approach to experimental data.

Section 4.3 proposed two automatic threshold selection algorithms for two widely used methods, namely DCT and DWT, to identify the appropriate roughness components in synthetic and experimental data sets. In contrast to the traditional way of decomposing engineering surfaces, proposed algorithms are data-driven, and they automatically adapt to a given surface to provide the needed threshold values. The algorithm for DWT does not require any input parameter selection from the user, while the one for DCT only requires the user to give a slope threshold for the derivative of the image entropy. However, this threshold value is generally close to zero and is easy to set, thus not requiring a high level of expertise.

The classification accuracies obtained from both automatic selection algorithms for DCT and DWT either exceed or match the accuracies obtained from the heuristic threshold selection. This shows that proposed algorithms are capable of autonomously decomposing the surfaces to extract the appropriate descriptor of the surface roughness. This study also eliminates human error, which may happen during the manual inspection process in heuristic threshold selection. When the results are compared to the ones obtained from Gaussian smoothing, it is seen that proposed algorithms provide better classification scores in profile classification, and they are comparable to their Gaussian smoothing counterparts. In addition, the proposed DCT algorithm is capable of eliminating the redundant mode computations for a given surface. This considerably reduces the computational time, as evidenced by an order of magnitude improvement in DCT mode computations for a single surface. The mode computation for DCT can be parallelized using High-Performance Computing (HPC) tools, and the computational time needed for DCT can further be significantly reduced. This

study claims that the combination of the proposed algorithms with HPC tools will enable this approach to be used in real-time surface characterization applications.

Section 4.4 proposed to use topological saliency and simplification for surface finish monitoring. The proposed approach is applied to synthetic surfaces explained in Section 4.2.1. It is seen that peaks and valleys can be clustered separately, and this can provide guidance on the regions where additional machining is required to obtain the desired surface finish. This section only introduces a pipeline to cluster given surface scans. The future direction of this study can involve the application of this pipeline into experimental data set and the integration of the resulting framework into a machining center to test the viability of the approach in real-time cutting experiments.

CHAPTER 5

TOOL WEAR IDENTIFICATION USING MACHINE LEARNING

5.1 Literature Review

Machining accuracy and final surface finish during cutting operations are dependent on several factors. One of them is the tool wear during the cutting. Tool wear can affect the tool life, and excessive tool wear can cause breakage as well as some damage to the cutting lathe. Therefore, its detection and prediction have been prominent research fields during the last decades. Tool wear analysis is essential for reducing material waste and prolonging the cutting tool life [261].

Tool wear analysis can be divided into two main kinds, and these are indirect and direct tool wear analysis [262]. Indirect approaches focus on the analysis of experimental data such as force, vibration, and acoustic emission sensor signals and finding the correlation between experimental signals and the tool wear [262]. Some studies in the literature focused on model-based approaches to detect tool wear. They compared the simulation results to the experimental ones using acquired signals from cutting experiments. For instance, Huang et al. designed an observer based on the derived linear model and estimated the cutting forces during the operation [263]. The estimated cutting forces are compared with the measured one, and threshold values for tool wear detection are proposed based on the model. Stavropoulos et al. extracted tool wear curves using numerical simulations based on Reference [264] and compared them to the experimental curves [265].

A wide range of studies focused on data-driven indirect approaches in the literature. Experimental data such as force, vibration, and acoustic emission signals are collected and analyzed using digital signal processing and time series analysis methods. Then, resulting signals are used to extract features, and they are combined with machine learning algorithms to identify the tool wear. One of the commonly used approaches is Singular Spectrum Anal-

ysis (SSA)[266, 267, 268]. For instance, Alonso et al. used SSA to decompose acceleration signals and grouped them into three clusters [267]. Statistical features extracted from these three main clusters are fed into neural networks to estimate the wear. Kundu et al. combined SSA and band-pass filtering to remove the noise from vibration signals and used time and frequency domain features in four different classification algorithm [268]. Wavelet Transform (WT) and Wavelet Packet Transform (WPT) are also commonly used approaches for indirect tool wear analysis [269, 270, 271, 272, 273, 274, 275]. Li et al. provide a review for analysis of experimental data obtained from acoustic emission sensors using different signal processing tools, including WT [269]. Pechin et al. studied Continuous Wavelet Transform (CWT) to identify the informative component in acoustic emission signals and proposed a tool wear identification system [270]. Leng et al. study the correlation of the energy of the wavelet packets and the AE signals with tool wear [271].

Statistical features such as kurtosis, root mean squared, and skewness are the most commonly adopted features for indirect methods of tool wear analysis in the literature[276, 277, 278, 279, 280]. For instance, Hassan et al. focused on signal segmentation and normalization process to reduce the noise in current signals obtained from the spindle of milling experiments and extracted statistical features to perform classification [276]. Simon et al. used several subsets of statistical features extracted from vibration signals in drilling experiments to detect tool wear status [277]. Another indicator used for determining the transition between tool wear status is mean power analysis. Rmili et al. employed mean power analysis and developed an algorithm that finds the transition between the tool wear status of vibration signals in real time [281].

Deep learning algorithms have been widely adopted in tool wear analysis during the last decades with the enhancement in data collection. Convolutional Neural Networks (CNNs) [282, 283, 284], artificial neural netweorks (ANNs) [267, 285, 286, 287, 280, 288] Deep learning algorithms may require a large number of cutting experiments to have enough data set to train the algorithms. For instance, Gouarir et al. used existing data set obtained

from the 2010 PHM Data Challenge to test the performance of a CNN model for tool condition monitoring [282]. However, the data used in this study has 315 cutting tests, and this number can not be easily obtained due to the required cost of conducting experiments. Another study was also focused on the analysis of images using CNNs and Fully Convolutional Networks (FCNs) to identify the tool wear status [283]. Their image data set includes 400 images obtained from cutting inserts, milling tool, and the drill at different magnification levels with a microscope.

On the other hand, direct approaches are based on the analysis of images taken from the cutting to define the amount of wear [289]. For example, Hou et al. developed an online tool wear inspection system based on machine vision [290]. The developed system is able to measure the bottom and flank wear. They designed a triple prism to overcome reflection on the bottom and flank faces. In another study, Wu et al. developed a framework called Toolwearnet that can identify the tool wear type and measure the tool wear using Convolutional Neural Networks (CNN) [291]. Real-time object detection algorithms are also employed in tool wear analysis. For instance, Lin et al. combined the You Only Look Once (YOLO) [292] algorithm with segmentation models such as U-net [293] and Segnet [294] to identify the tool wear areas in cutting tool images [295]. García-Ordás et al. developed an online approach that can determine the tool wear status based on computer vision and machine learning [296]. They first obtained the cutting edge images of the inserts, then identified the wear patches. These patches are classified as worn and serviceable, and finally, the wear status of the insert is determined based on the wear status in the wear patches.

This chapter only focuses on the indirect approaches used by the sensor measurements. During data collection, force sensors, acoustic emission sensors, accelerometers, and microphones are among the commonly used sensors. Since multiple sensors are used during experiments, it is essential to apply data fusion to combine the information gained from each sensor. According to Reference [261, 297], there are three different data fusion techniques. These are the data level, feature level, and decision level. The studies reviewed in this study

are categorized into these three data fusion techniques. Out of 44 different studies, it was seen that only 10 of them applied sensor fusion to improve the tool wear identification/prediction performance. In addition, it is seen that mostly feature level and decision level data fusion are used in these studies. Decision level study is only used once among the reviewed studies, while the rest uses the feature level fusion. Feature level fusion generally extracts features from the signals, and then the relationship between them is found by using different techniques. One of the most commonly used feature-level-based data fusion is the fuzzy inference system [298, 299, 300]. For instance, Yao et al. used Fuzzy Neural Network to identify tool wear states in turning operation [298]. Signals obtained from the acoustic emission sensor, feed motor current, and spindle motor current are used to extract features. The relationship between the extracted features and the tool wear status is found using fuzzy neural networks. Wu et al. utilized the Ensemble Empirical Mode Decomposition to eliminate the noise effects in vibration, acoustic emission, and motor current signals [300]. Then they extracted statistical features from these signals in both the time and frequency domains. Optimum feature selection is performed using correlation analysis, monotonicity analysis, and residual analysis. Finally, they used an adaptive network-based fuzzy inference system (ANFIS) to find the relationship between tool wear level and the extracted features [300].

Another technique used for feature level data fusion in tool wear analysis is Mahalanobis-Taguchi System (MTS) [301]. It is a pattern recognition algorithm based on the combination of Mahalanobis Distance [302] and Taguchi methods. Rizal et al. employed MTS to perform tool wear classification [303]. Time and frequency domain features are extracted from the six-channel data set. Then, Mahalanobis Space is identified and validated. The Taguchi method is used to eliminate redundant features, and Mahalanobis Distance (MD) is used to identify tool wear status. Another study applied the correlation-based sensor fusion technique to find the optimal set of features obtained from various sensors, and the optimal set is combined with ensemble machine learning to classify tool wear, breakage, and chipping [304]. It utilized both feature and decision-level data fusion techniques. Chen et al. studied the effect of five

different data fusion techniques in feature level for tool condition monitoring in milling, and neural networks are used to detect tool wear condition [242]. Wang et al. proposed an in-process tool condition monitoring approach based on Self-Organizing Map (SOM) to estimate the flank tool wear [305]. Flank wear is estimated using captured images, and the features extracted from force signals are used to train the SOM network. In addition to these techniques, Liu et al. combined all the features extracted from the accelerometer and dynamometer using Discrete Wavelet Transform (DWT) [261]. Then the condition of the machine is classified using the Support Vector Machine classifier.

Some studies apply data and decision level data fusion. For instance, Xu et al. introduced a deep learning-based sensor fusion at the data level [284]. Data obtained from an accelerometer, acoustic emission sensor, and dynamometer are fed into parallel convolutional networks to obtain a feature map that defines the relationship between the features extracted from each sensor with the tool wear. Kannatey-Asibu et al. utilized class-weighted voting to decide on tool condition [306]. They have combined the decision made by Hidden Markov Model (HMM), Gaussian Mixture Model (GMM), Bayesian rule, and K-means model using majority voting.

This study focuses on indirect tool wear analysis using feature level data fusion technique. Experimental signals are collected using an acoustic emission sensor, a microphone, and a force dynamometer. Then, tool wear for each cutting insert is measured using digital microscopy. Crater and flank wear are used as a benchmark to label each cutting signal. The signals are labeled low, medium, and severe wear. Two of the most commonly adopted signal decomposition techniques, Discrete Wavelet Transform (DWT) and the Ensemble Empirical Mode Decomposition, are implemented in this study. Feature extraction using DWT and EEMD is based on the selection of several parameters, such as the level of the transform for DWT and the informative coefficients or Intrinsic Mode Functions (IMFs). This study presents an automatic and adaptive pipeline for these two approaches and tests it on experimental data. The experimental signals are cleaned using the low pass and bandpass

filters, and then downsampling is applied to reduce the sample size. For DWT and EEMD approaches, statistical features are extracted from the sensors and combined into a single vector in the feature matrix. Then, Principal Component Analysis is used to reduce the dimension of the feature matrix. The resulting feature matrix and the labeling obtained from wear measurements are fed into supervised classification algorithms. In addition to traditional approaches for feature extraction, Topological Data Analysis (TDA) based feature extraction is also utilized in this study. Feature level data fusion is also applied to the TDA-based approach. This study compares the DWT and EEMD results to the ones obtained from the TDA-based approach.

This chapter is organized as follows. Section 5.2 describes the experimental procedure for titanium cutting experiments and data cleaning. In Section 5.3, the feature extraction procedure is described for TDA based approach in addition to two widely adopted approaches, DWT and EEMD. Section 5.4 compares the results of the TDA-based approach to the DWT and EEMD. In Section 5.5 provides the concluding remarks.

5.2 Experimental Procedure and Data Cleaning

This section explains data collection and cleaning for titanium cutting experiments. Cutting experiments are conducted in dry conditions on a Haas TL1 CNC lathe shown in the top image of Figure 5.1. A titanium bar Ti64-STA with an initial diameter of 12.7 cm is used during the experiments. Three sensors are utilized to collect data from the experimental setup. Force dynamometer is placed into CNC lathe as shown in the Figure 5.1. It measures the forces in three orthogonal directions shown in the right bottom corner of the top image in Figure 5.1. A Kistler 8152C acoustic emission sensor with a measuring range of 100-900 kHz is used to measure the elastic waves in the tool holder. Procedure in Reference [307] is followed to check the consistency of acoustic emission sensor output. The sensor is mounted on the backside of the tool holder. Audio signals are also collected during the experiments. A PCB 130F20 with a measuring frequency range of 10 Hz-20 kHz is placed approximately

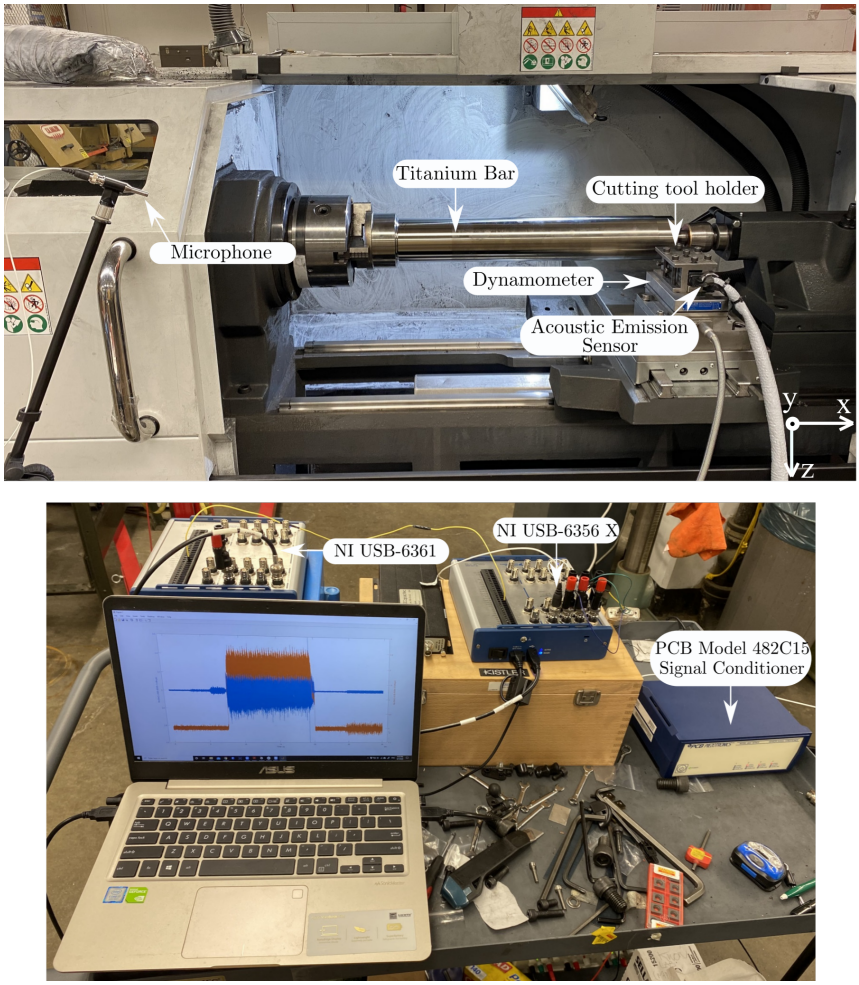


Figure 5.1: Experimental setup for titanium cutting experiments. (top) Titanium bar and the sensors used in data collection, (bottom) Data acquisition boxes, and signals conditioner used during the experiments.

a meter away from the cutting lathe to reduce the effect of reflected sound waves.

The bottom image in Figure 5.1 shows the data acquisition boxes and the signal conditioner used during cutting experiments. Since 900 kHz is the maximum frequency measured by the acoustic emission sensor, 1.8 MHz is the sampling rate needed with respect to the Nyquist sampling theorem. The acoustic emission sensor is connected to the data acquisition box NI6361 since its sampling rate can reach up to 2 MHz. However, when multiple channels are used for data acquisition with the same box, 2 MHz is divided into the number of channels used. Therefore, the second data acquisition box NI6356 is used to collect data

from the microphone and force dynamometer, and it can reach up to 1.25 MHz per channel. Since the measuring range for the microphone and the dynamometer is low compared to the acoustic emission sensor, the sampling rate for these two sensors is selected as 180 kHz. The microphone is first connected to a signal conditioner PCB 482C15 and then to the data acquisition box. MATLAB's data acquisition toolbox is utilized to collect data and synchronization of two data acquisition boxes.

Sandvik carbide inserts with and without chip breaker are used in the cutting experiments, and their model numbers are SCMW 12 04 08 H13A and SCMT 12 04 08-KR H13A. During the cutting operations, the cutting speed is kept constant at 122 m/min, while the rotational speed of the bar changes as the radius of the titanium bar decreases. Rotational speed varies between 407 and 654 rpm during the experiments. Depth of cut and feed rate are also kept constant at 1.2 mm and 0.127, respectively. The corners of each cutting tool are numbered from 1 to 4. For each cutting experiment, a cutting length required to break the tool is estimated based on the cutting conditions and prior experience. The time required to complete this length of cutting is called T_F. T_F is divided into four different time intervals, whose length is $\Delta t = T_F/4$. For the first corner of the cutting tool, a cutting operation is performed for $\Delta t = T_F/4$ seconds, and the resulting data set is saved. Then, the second corner of the cutting tool is used to cut titanium for $2\Delta t = T_F/2$ seconds. Every time when a new corner of the cutting tool is used, the cutting time is increased by Δt. This results in having a tool breakage in the fourth corner where cutting time is equal to T_F.

5.2.1 Data Processing

High sampling rates increase the number of samples dramatically, even if the sampling time is not large. Therefore, for each cutting operation, the data acquisition is started in the last 30 seconds of the corresponding cutting time. During experiments, six cutting tools without chip breaker and ten tools with chip breaker are used. After data collection, the next step is to clean the data set. Previously, it has been noted that the acoustic emission sensor

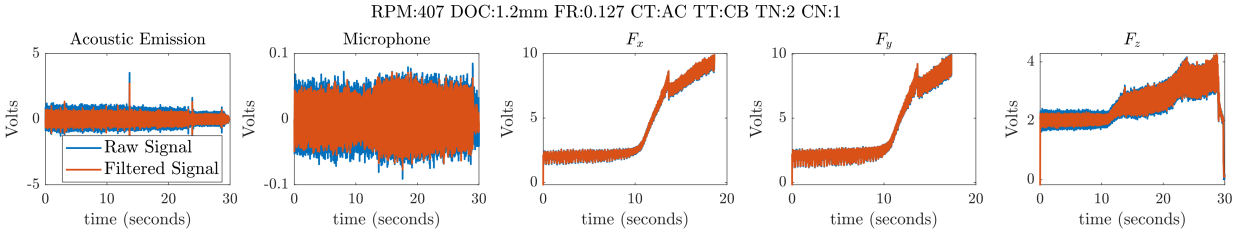


Figure 5.2: Filtered signals and raw measurements obtained from cutting configuration with RPM:407, DOC (depth of cut):1.2 mm, CT (cutting time): AC (All cut, where cutting time is T), TT (tool type): chip breaker, TN (tool number): 2 and CN (corner number): 1.

also captures the AC frequency (60 Hz) during data collection. This component needs to be removed from the signals as the first step. This study uses a baseline correction approach based on Stationary Wavelet Transform to remove the AC frequency component. One can refer to Reference [308] for more details about the approach.

Then, acoustic emission signals are filtered using a bandpass filter since the typical range for acoustic events in machining is between 100-300 kHz [309]. The maximum voltage output is 10V for force signals. During experiments, the acquired signals in several axes may reach up to 10V due to a dramatic increase in the force along the corresponding axis. Therefore, force signals are thresholded before applying filters to them. When the amplitude of signals reaches 9.95 V, the algorithm automatically neglects the rest of the time series. For audio and force signals, least square method with a filter order of 100 is used to design a FIR filter to reduce the effect of noise in raw measurements. The Force dynamometer used in these experiments can measure frequencies up to 45 kHz, while the frequency range for the microphone is between 10 Hz and 20 kHz. Therefore, force signals and audio signals are passed through low pass filters designed using the desired frequency of 45 and 20 kHz, respectively. Figure 5.2 shows the raw measurements and the filtered signals obtained from a cutting configuration.

After completing the filtering of all signals, the next step is to downsample the signals so that we can reduce the number of samples. Downsampling is performed by choosing

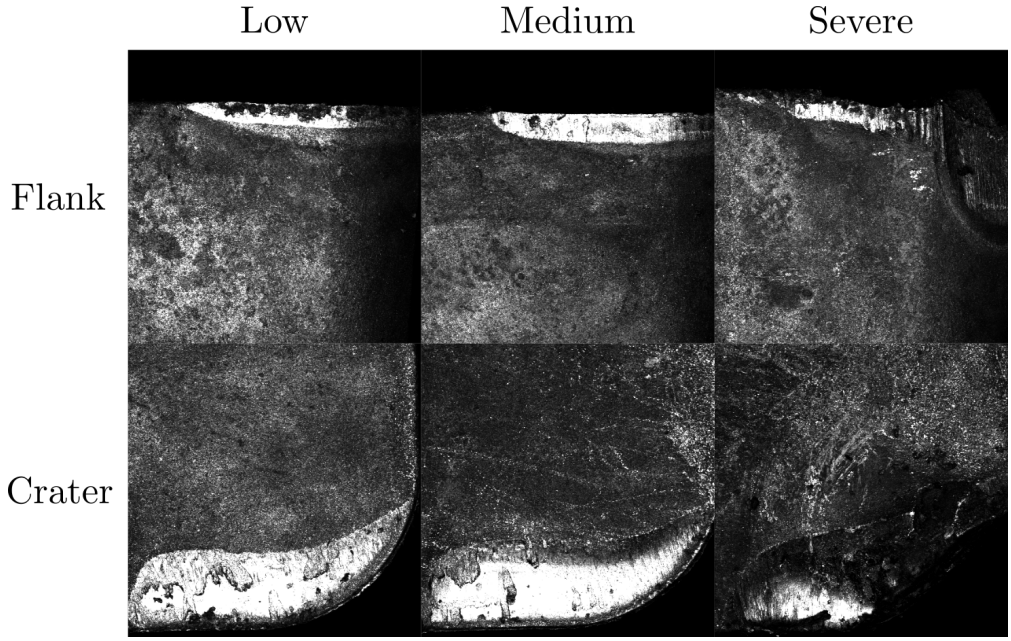


Figure 5.3: Surface scans for cutting inserts with low, medium and severe wear levels. All scans are obtained from cutting inserts without chip breaker used in different cutting tests.

a downsampling factor for each sensor data and choosing every n^{th} sample in the signals based on these factors. These factors are chosen such that the resulting sampling frequency is two times the desired frequency. For acoustic emission signals, the downsampling factor of 3 is chosen, while 4 and 10 are chosen for microphone and force signals, respectively. Figure 5.2 shows that force signals experience significant jumps and baseline shifts during cutting. Therefore, it is decided to crop these signals such that only the initial part of the signals before the change points are used in the analysis. This procedure is completed by manually selecting the two endpoints for the corresponding time series since the data set has only 55 different cutting configurations. This also removes the high amplitude peaks at lower frequencies in the frequency spectrum of the force signals.

5.2.2 Data Labeling

Preprocessed time series are labeled based on the wear amount measured for each cutting experiment. Crater and flank wear for each corner of the cutting inserts are measured. Then,

time series are labeled into three categories: 1) low wear, 2) medium wear, and 3)severe wear. The surface scans of the cutting inserts are also taken under a microscope, and Figure 5.3 presents the surface scans of three different wear levels for both flank and crater wear. Table 5.1 also provides the range of wear amounts used in the labeling. After labeling the time series, the next step is to extract features from them.

Table 5.1: The ranges for wear amounts used in labeling of time series data.

	Inserts without chip breakers (WOCB)		Inserts with chip breakers (CB)	
Wear Level	Crater ($d(\mu\text{m})$)	Flank ($vb(\mu\text{m})$)	Crater ($d(\mu\text{m})$)	Flank ($vb(\mu\text{m})$)
Low	$d < 25$	$vb < 110$	$d < 26$	$vb < 127$
Medium	$25 \leq d < 72$	$110 \leq vb < 127.5$	$26 \leq d < 62$	$127 \leq vb < 187$
Severe	$d \geq 72$	$vb \geq 127.5$	$d \geq 62$	$vb \geq 187$

5.3 Methodology

In this section, feature extraction approaches from experimental signals are explained. This study utilizes two widely adopted signal decomposition approaches, DWT and EEMD, in addition to novel TDA-based approach. The background information for these approaches can be found in Sections 4.3.2.1, 2.3.2 and 2.6.1. Therefore, this section only describes the feature extraction procedure.

5.3.1 Discrete Wavelet Transform

Discrete Wavelet Transform based feature extraction approach for tool wear analysis is introduced in Reference [57]. The first step is to standardize the processed signals such that they have zero mean and standard deviation of one. In Reference [57], frequency spectrum of the selected signals are investigated, and the frequencies whose amplitudes are significant in the spectrum are defined. These frequencies are used to identify the level of DWT. In this study, the approach is modified such that sensitive frequency is selected in an automated and adaptive way. Algorithm 5.1 provides the pseudo code for the adaptive algorithm.

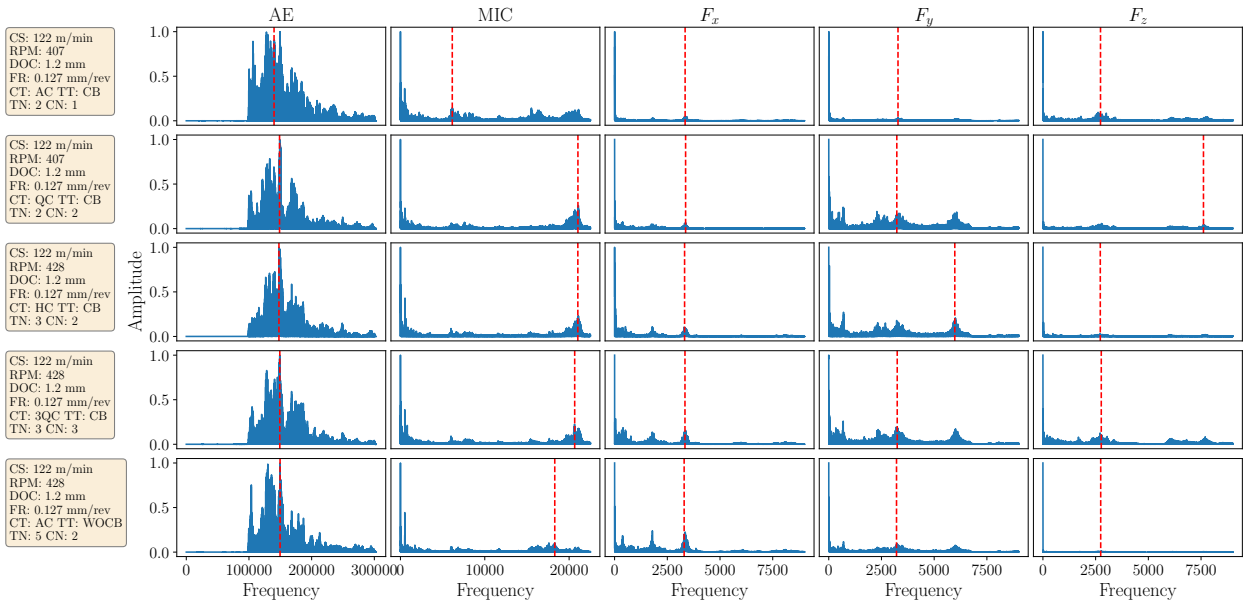


Figure 5.4: FFT spectrum of the processed signals obtained from the first five cutting configurations. CS: Cutting speed, DOC: depth of cut, FR: feed rate, CT: cutting time, QC: quarter cut, HC: half-cut, 3QC: three-quarter cut, AC: all cut, TT: tool type, CB: chip breaker, WOCB: without chip breaker, TN: tool number, CN: corner number.

FFT is first applied to processed signals, and then the peaks in the spectrum are found using the built-in peak finding algorithm in Python. The frequency spectrum is normalized by dividing all amplitudes by the maximum amplitude. The number of peaks found by this algorithm is large and contains redundant peaks as well. Therefore, a restriction parameter minimum peak height is set. Figure 5.4 presents the spectrum of the signals of five cutting configurations. It is seen that the spectrums of force and microphones signals have a large amplitude in lower frequencies. To eliminate these frequencies, the proposed algorithm neglects the first 200 samples in the frequency domain. Then, the threshold for the minimum peak height is decided by multiplying the maximum amplitude of the remaining samples in the spectrums by 0.2. However, this threshold value can still lead to capturing low frequencies in the spectrum in some cases. Therefore, a minimum desired frequency for each sensor is decided. 100 kHz, and 5 kHz are chosen as the minimum desired frequency for acoustic emission sensor and microphone, respectively. For force signals, 1000 Hz is chosen.

Algorithm 5.1: Automatic and adaptive algorithm to find the sensitive frequency.

Input: desMinFreq=100000, p=0.2, freq, amp, thrshldFreq=5000
Output: senFreq
  peaks = []
  while max(peakFreq)<desMinFreq **do**
    peaks = findpeaks(amp, height=max(amp[200:]*p))
    PeakAmp = amp[peaks]
    PeakFreq = freq[peaks]
    p -= 0.03
  end while
  peakFreq=peakFreq[peakFreq>thrshldFreq]
  counts, freqHist = histogram(peakFreq)
  Set SecondMaxCount = sorted(counts)[-1]
  Find the index of SecondMaxCount in counts,
  Set SecFreqHist = freqHist[index]
  Find the closest frequency to SecFreqHist in freq and set it to senFreq

Algorithm 5.1 shows that iterations are performed in a while until the amplitude of the desired minimum frequency is in the selected peaks. In addition to setting desired minimum frequency, the peaks found for microphone and force signals are thresholded by predefined frequencies. The frequencies under 5 kHz and 1 kHz are neglected for microphone and force signals, respectively. These frequencies generally either coincide with structural modes of the system or do not contain useful information in the spectrum. Then, the histogram of the resulting peaks is found with 100 bins. This means that the frequency range in the spectrum is divided into 100 equal intervals, and the number of frequencies that fall into each interval is counted. The highest count generally corresponds to the lower frequency region, so the proposed algorithm selects the second highest count in the histogram and finds its corresponding frequency provided by the histogram. However, this frequency can not be directly used as a sensitive frequency since the histogram only provides 100 frequency values. Therefore, the proposed algorithm finds the closest frequency to it, and this is the sensitive frequency chosen by the algorithm.

The proposed adaptive algorithm is tested on the time series data of the first five cutting configurations, and the resulting sensitive frequencies are marked in the spectrum with red

dashed lines as shown in Figure 5.4. It is seen that selected frequencies are meaningful and can vary from case to case. This shows the effectiveness of the adaptive approach. These resulting sensitive frequencies are used to determine the level of the wavelet transform. Figure 4.11 shows 3 level of DWT. In the first level, approximation and detail coefficients correspond to $[0, f/4]$ and $[f/4, f]$ ranges, respectively. An iterative approach is utilized to find the level of DWT L, and it is set to 1 initially. Then, if the sensitive frequency is smaller than $f/(2L)$, the algorithm applies the DWT in the first level $L = 1$. Otherwise, L is increased by one, and $f/(2L)$ is computed again. This can be iterated until L reaches the maximum level of the transform that is found with respect to the selected mother wavelet function and the length of the corresponding signal. After determining the level of DWT, approximation and detail coefficients are obtained, and two signals are reconstructed based on these coefficients. For each reconstructed signals, 10 statistical features given in Table 2.3 are computed. Since features from all signals are combined for a cutting configuration, feature level data fusion is applied in this approach. The resulting feature matrix is given as input to supervised classification algorithms.

5.3.2 Ensemble Empirical Mode Decomposition

This study follows the feature extraction approach used in Reference [300]. IMFs are obtained using the PyEMD package of Python for each time series. Then, autocorrelation functions for the original time series and its IMFs are computed. The Pearson correlation coefficients between the autocorrelation functions of the original time series and the IMFs are computed. An initial threshold value of 0.5 and 0.15 are selected for the acoustic emission sensor and the rest of the sensors, respectively. If the correlation coefficient between the autocorrelation functions is greater than this threshold value, corresponding IMF is used in the reconstruction of the signal. If the algorithm can not find IMFs whose correlation coefficient is greater than the threshold value, the threshold is lowered by 0.05. This process is iterated until the algorithm finds at least an IMF. The resulting IMFs are summed up to

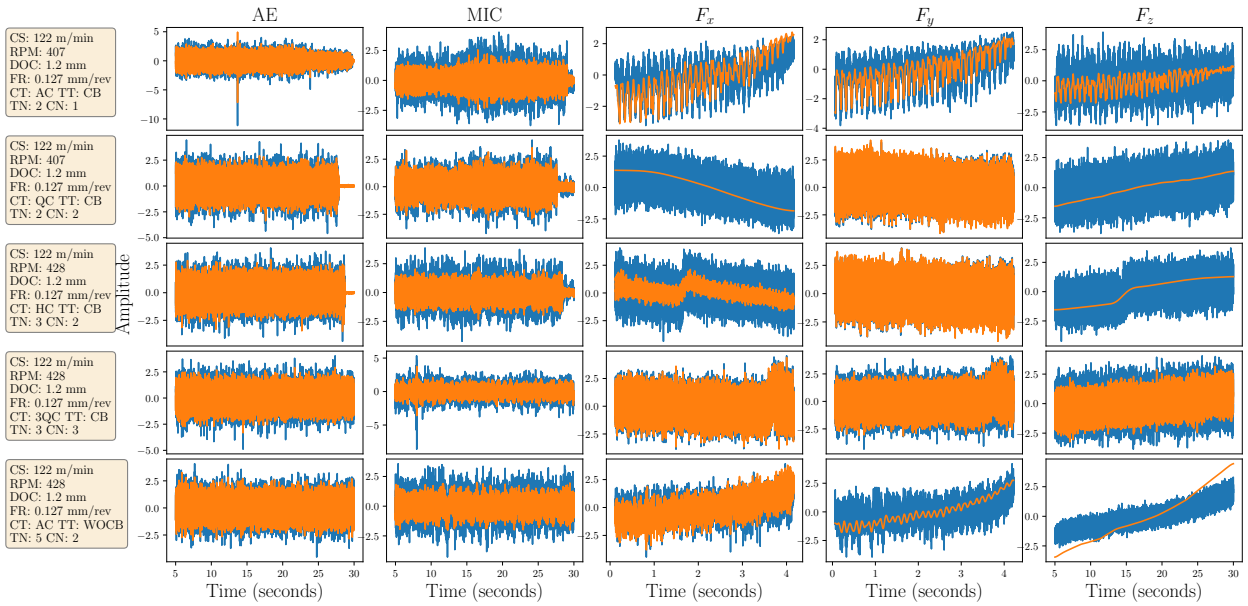


Figure 5.5: Processed time series (blue) and the reconstructed signals using EEMD (orange) for five cutting configurations when the last 25 seconds of the time series used in the analysis. CS: Cutting speed, DOC: depth of cut, FR: feed rate, CT: cutting time, QC: quarter cut, HC: half-cut, 3QC: three-quarter cut, AC: all cut, TT: tool type, CB: chip breaker, WOCB: without chip breaker, TN: tool number, CN: corner number.

reconstruct the signal. Figure 5.5 and 5.6 show the reconstructed signals and the processed signals when the last 25 and 5 seconds of the time series that is used in the analysis. It is seen that only IMFs which correspond to lower frequency content are selected to reconstruct the signals for some of the force signals (see reconstructed signals of F_x and F_z of the second cutting configuration in Figure 5.5). However, better reconstructions are obtained when only the last five seconds of the time series that is used in the analysis. These reconstructed signals are used to extract statistical features as listed in Table 2.3.

5.3.3 Topological Data Analysis

The first step in the TDA-based approach is to embed the time series to higher dimensional space so that a point cloud representation of the data can be obtained. There are two parameters needed for this step: embedding dimension and delay parameters. This study

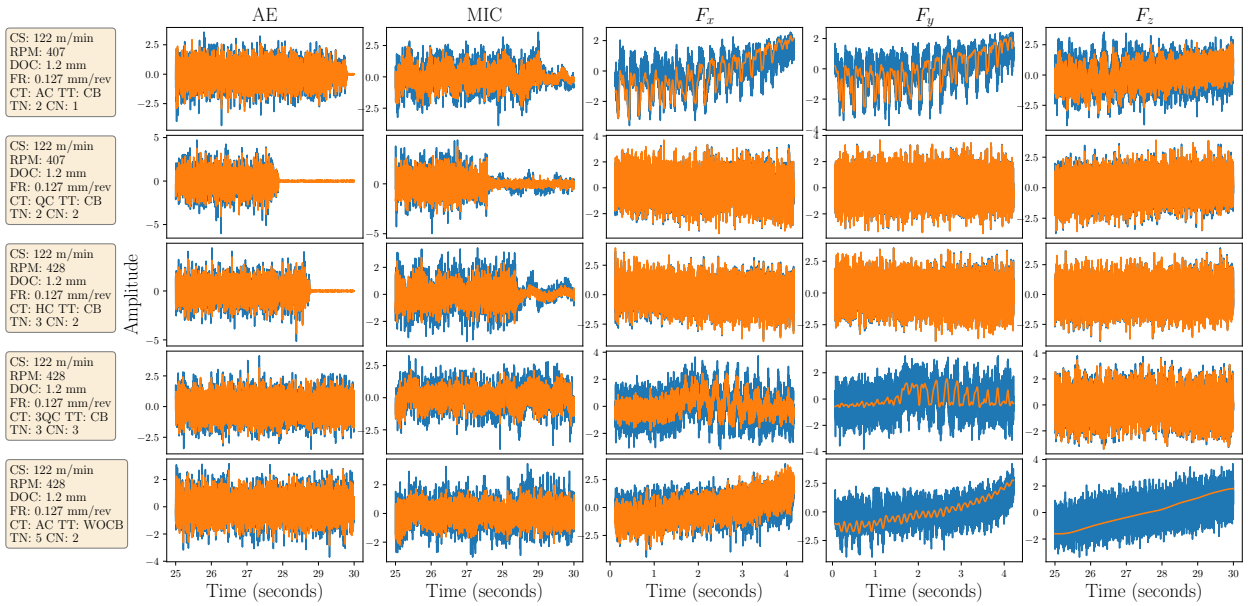


Figure 5.6: Processed time series (blue) and the reconstructed signals using EEMD (orange) for five cutting configurations when the last 5 seconds of the time series that is used in the analysis. CS: Cutting speed, DOC: depth of cut, FR: feed rate, CT: cutting time, QC: quarter cut, HC: half-cut, 3QC: three-quarter cut, AC: all cut, TT: tool type, CB: chip breaker, WOCB: without chip breaker, TN: tool number, CN: corner number.

utilizes the Least Means Square (LMS) based approach introduced in Reference [143]. This approach uses LMS to identify the noise floor in the Fourier spectrum of the given signal. Then, the noise floor is used to identify the maximum significant frequency in the spectrum. The delay parameter is defined as $\tau = f_s/(2f_{max})$ where f_s and f_{max} represents the sampling frequency and the maximum significant frequency, respectively [143]. Then, the embedding dimension is computed using the delay parameter and the Multi-scale permutation entropy [310, 143]. After finding these two parameters, the time series are embedded using Taken's embedding (see Section 2.6.3.1).

The embedded time series represents the point clouds that are used to compute the persistence diagrams. In this chapter, only considered the first-dimensional persistent homology is considered (H_1), where the main interest is loops in the point cloud. The Ripser package in Python is utilized to compute the persistence diagrams. Since the time series contains a large

number of samples, embedding results in giant point clouds. Prior experience shows that having more than 1000 points in the point cloud can increase the computation time of the diagrams. Therefore, previously, this study used the greedy permutation [78] option of the Ripser and Bezier curve approximation approach to reduce the runtime for computation of persistence diagrams (see Section 2.6.3.2 for more details.) This chapter uses only the greedy permutation option in the Ripser to obtain persistence diagrams. The algorithm subsamples point clouds to the final size of 1000 samples, and the resulting persistence diagrams are used to extract features. In Section 2.6.4, five different featurization option for persistence diagrams are explained. This section utilizes only Carlsson Coordinates, persistence images, and template functions. The resulting features obtained from persistence diagrams are fed into supervised classification algorithms to determine the amount of wear in cutting signals.

5.4 Results

Features obtained from cutting signals using the approaches explained in Section 5.3 are scaled such that each column of the feature matrix has zero mean and unit standard deviation. Then the next step is to apply Principal Component Analysis (PCA) [311] to the feature matrix to reduce the dimension of the feature matrix. The maximum number of components that can be selected is equal to min(r,c), where r and c represent the row and column number of the feature matrix, respectively. For instance, there are 100 features for each time series in the DWT approach, but the number of samples is 55. Therefore, 55 is the maximum number of components to select in PCA. Then the variance contributions of each feature are calculated and sorted. When the cumulative sum of variance contributions exceeds 90%, the algorithm automatically selects the corresponding number of components and applies PCA again to obtain the final feature matrix. For DWT and EEMD approaches, the variance plots are provided in Figure 5.7.

After applying PCA, the next step is to generate the training and test set. This study uses stratified 5-fold cross-validation to generate the training and test sets. For each feature

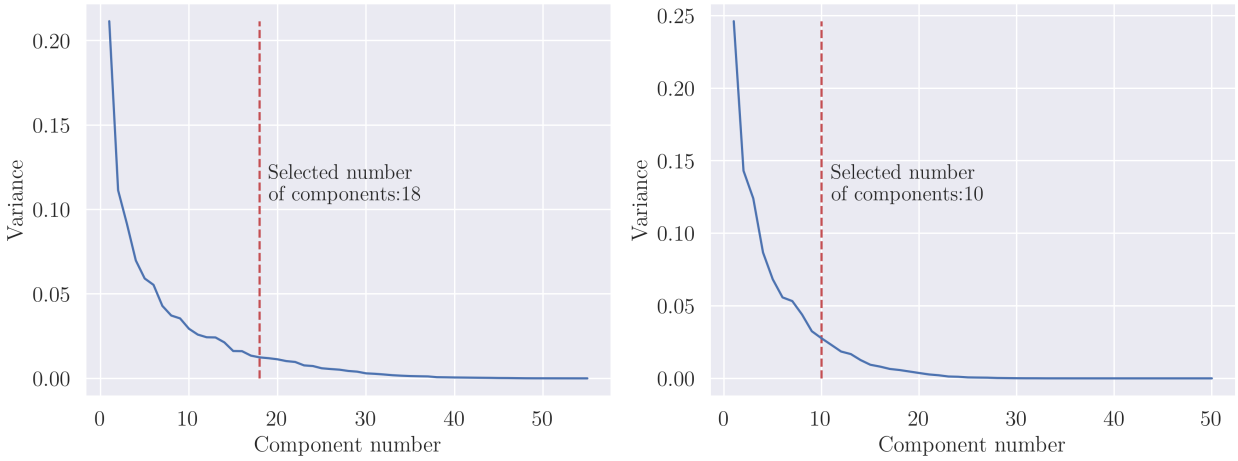


Figure 5.7: The number of components obtained after applying PCA for features of DWT (left) and EEMD (right) and computing the cumulative variance ratio. These variance plots are obtained when the last 25 seconds of the time series are used in feature extraction.

extraction approach, the same random state number is used for the k-fold cross validator to generate the same sets of training and test sets for a fair comparison. In this study, four supervised classification algorithms explained in Section 2.2 are used to test the performance of each feature extraction approach. Parameter tuning for each classifier is performed using a grid search-based approach. A range of values or a list of options is predefined for the parameter to be tuned. Then, the grid search algorithm finds the set of parameters that results in the best estimator, providing the highest accuracy. The classifier is trained and tested based on these parameters, and several metrics such as accuracy and F1 scores are saved. Time series are labeled based on three wear levels (low, medium, and high) measured after running the experiments. Therefore, in this study, a three-class classification is applied.

This study also investigates how the time duration of the signals used in the analysis affects the classification results. Therefore, the last ΔT seconds of the time series are only taken into account during feature extraction where $\Delta T = 5, 10, 15, 20$ and 25. The machine learning framework is run for each ΔT value, and the results are obtained. This allows us to see how much data is needed to identify different types of wear with the explained approaches.

5.4.1 Discrete Wavelet Transform Results

After obtaining the feature matrix using the DWT approach explained in Section 5.3.1, classification results are obtained using 5-fold cross-validation. Two types of labeling are performed based on the wear type. Figures 5.8 and 5.9 provide the results obtained with the labeling based on flank and crater wear, respectively. Figure 5.8 shows that DWT can detect the flank wear level with 50% accuracy at most when the last 5 seconds of the time series are used, and logistic regression is chosen as a classifier. However, this result has a large deviation amount which is 15.9%. The same accuracy with a smaller deviation can only be obtained when the last 20 seconds of the time series are used. This will also increase the number of samples analyzed in the framework, potentially increasing the runtime. On the other hand, Figure 5.9 provides better accuracies when the labeling of time series is done with respect to crater wear measurements. The highest accuracy reaches 58% when $\Delta T = 20$ seconds with the SVM classifier. If only the last 5 seconds of the time series are used, the resulting accuracy is 54% with a similar deviation compared to the one for the highest accuracy. This shows that DWT can detect the wear level without needing many samples. However, it is worth noting that several assumptions are required to identify sensitive frequencies in the Fourier spectrum.

In addition to test set results, training set results for the same approach are provided in Figures D.1 and D.2 for flank and crater wear based labeling, respectively. It is seen that there is a large difference between the test set and training set accuracies. This could be the indication of overfitting in the classifier. However, it is worth noting that the total number of samples (55) and the 5-fold cross-validation results in samples 11 samples in the test set. Since this is a three-class classification, approximately every test set has four samples from each group. This could also lead to lower classification results in the test set and higher deviations in the accuracies.

Most of the forces extracted from time series belong to the force sensor since three time series are obtained in each cutting experiment. To see the effect of features from the force

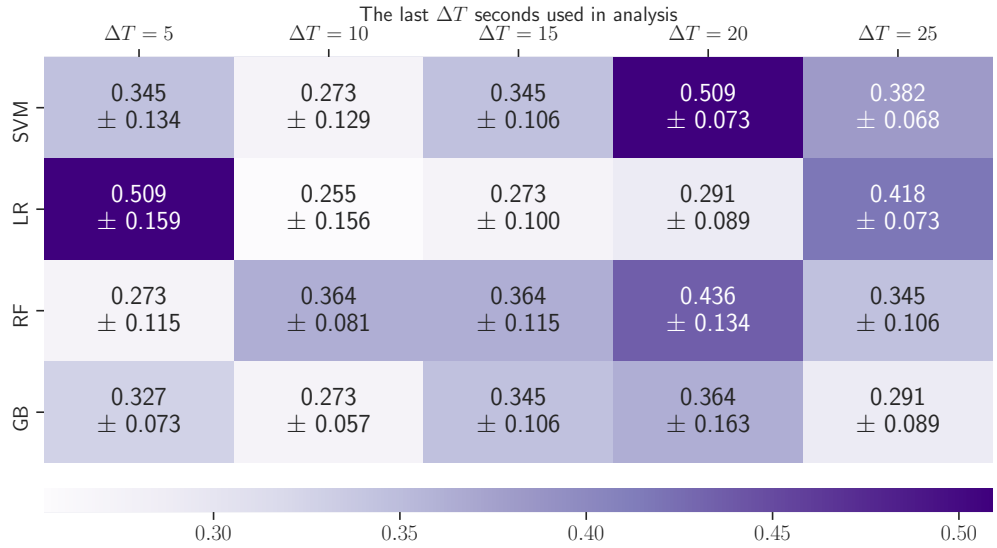


Figure 5.8: The test set classification scores obtained with the DWT approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on flank wear measurements.

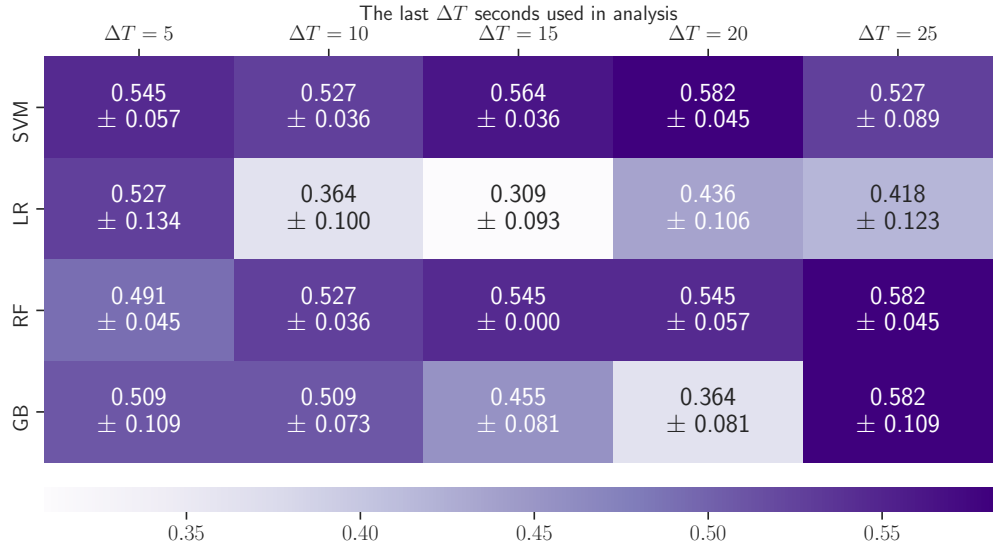


Figure 5.9: The test set classification scores obtained with the DWT approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

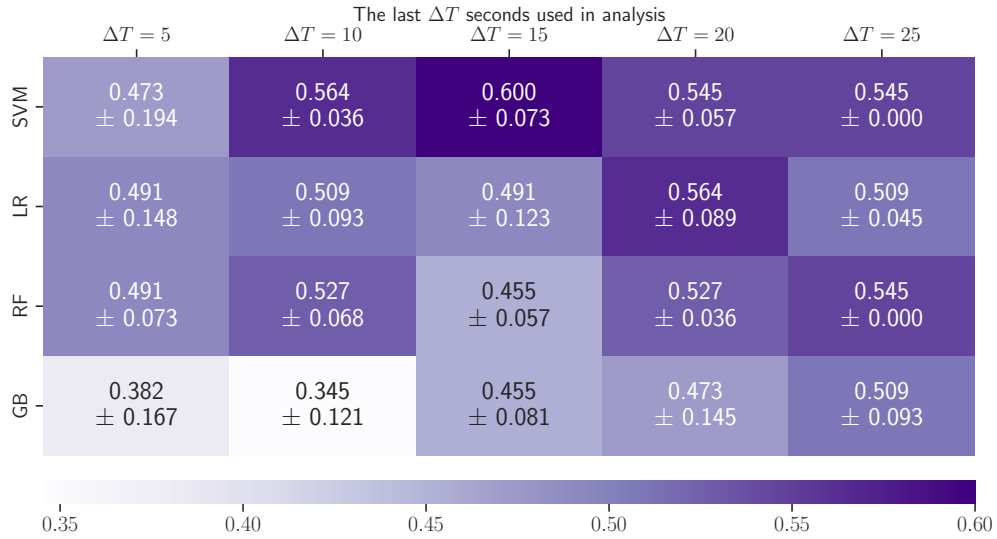


Figure 5.10: The test set classification scores obtained with the DWT approach by excluding features from force signals. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

sensor, classification is performed by excluding them. The resulting classification results are provided in Figure 5.10. This figure only contains the results obtained with labeling based on crater wear since flank wear results provide poor accuracies. Figure 5.10 shows that classification scores are relatively lower compared to the ones provided in Figure 5.9, especially for small ΔT. When features from force signals are included in the feature matrix, DWT can detect the wear level with 54.5% accuracy when $\Delta T = 5$. However, a similar classification performance can be obtained with $\Delta T = 10$ when we exclude the features from force signals (see Figure 5.10). In addition, the user needs to increase ΔT even further to obtain the highest accuracy, which is 60%. This indicates that the DWT approach requires users to include force signal features for better performance if the user wants to identify the wear level in a signal in a short time, in this case, smaller ΔT. Selected sensitive frequency in the Fourier spectrum of shorter signals can provide better descriptors for identifying wear levels in the time series.

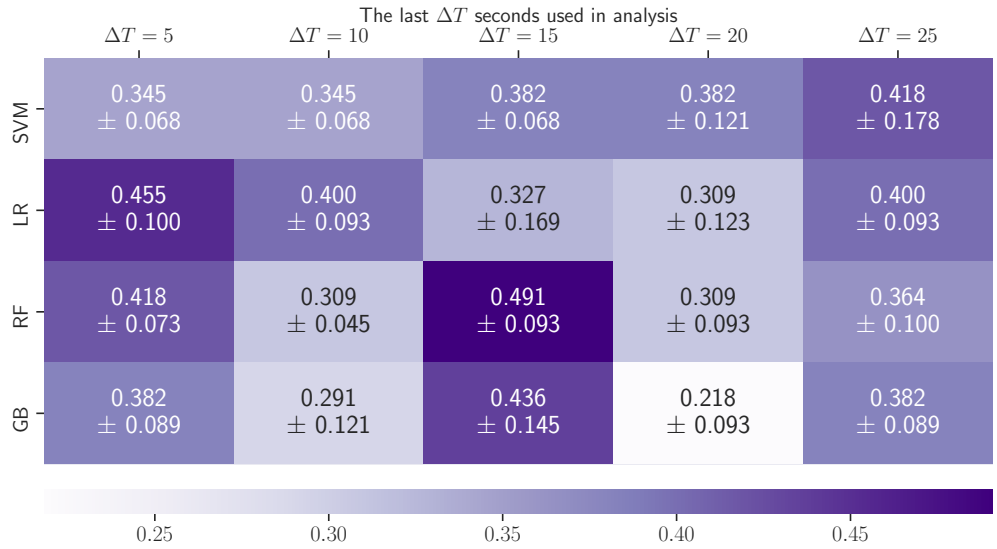


Figure 5.11: The test set classification scores obtained with the EEMD approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on flank wear measurements.

5.4.2 Ensemble Empirical Mode Decomposition Results

The same time domain feature set presented in Table 2.3 is used to extract features from reconstructed signals with the EEMD approach. The resulting classification scores of the test set for four different classifiers and varying ΔT parameters are presented in Figures 5.11 and Figures 5.12, while Figures D.1 and D.2 It is seen that labeling based on crater wear measurements performs better than labeling with flank wear measurements. The highest accuracies are obtained when $\Delta T = 15$ for flank and crater wear measurements. When ΔT is chosen as 5 seconds, crater wear-based labeling can provide 54.5% accuracy with a 17.2% deviation. However, this deviation is quite larger than the one for the highest accuracy (see Figure 5.12). In addition, the DWT approach can provide the same accuracy as EEMD with lower deviation when the last five seconds of the time series are used in feature extraction.

The results without including the features from force signals are obtained for EEMD. Figure 5.13 shows that excluding features of force signals can lead to an increase in accuracies when $\Delta T = 10$. However, accuracies decreases for $\Delta T = 5$ if Figures 5.12 and 5.13 are

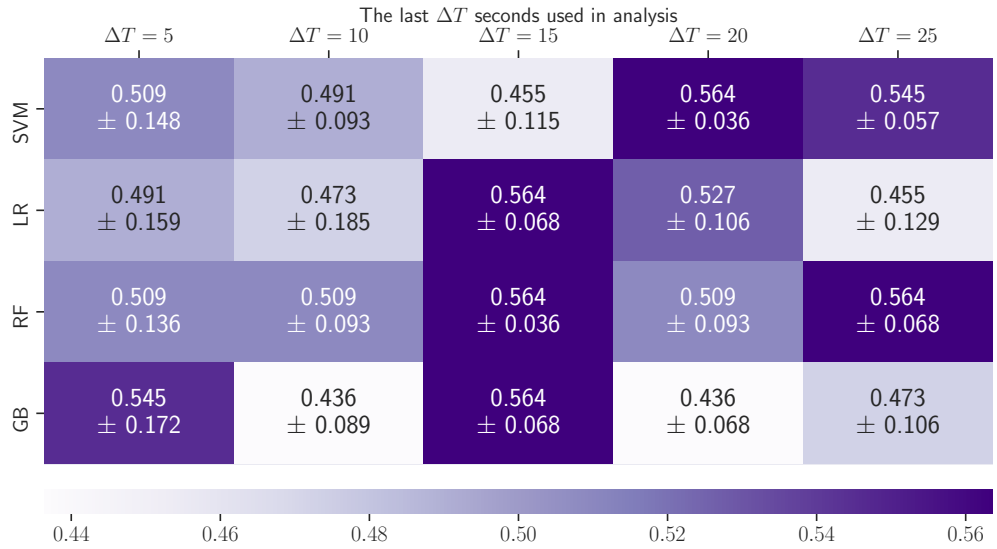


Figure 5.12: The test set classification scores obtained with the EEMD approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

compared. This shows us the effect of selected IMFs in feature extraction. Reconstructed signals obtained from force sensor using EEMD may not have all the information needed to identify the wear level since the reconstructed signals in some cutting configurations seems to be an approximation to the original signal (see Figure 5.6). If most of the reconstructed signals look like a curve fit to the original signal, the features obtained from them can not be a good descriptor for identifying the wear level. Removing these features can lead to better accuracies, as seen from the case when $\Delta T = 10$.

5.4.3 Topological Data Analysis Based Approach Results

This study also proposes a TDA-based approach for tool wear analysis. Three types of featurization techniques are utilized in this study to summarize the persistence diagrams obtained from time series data. These are Carlsson Coordinates, persistence images, and template functions. Figures 5.14 and 5.15 provides the classification results obtained with Carlsoon Coordinates using flank and crater wear based labeling, respectively. As seen previ-

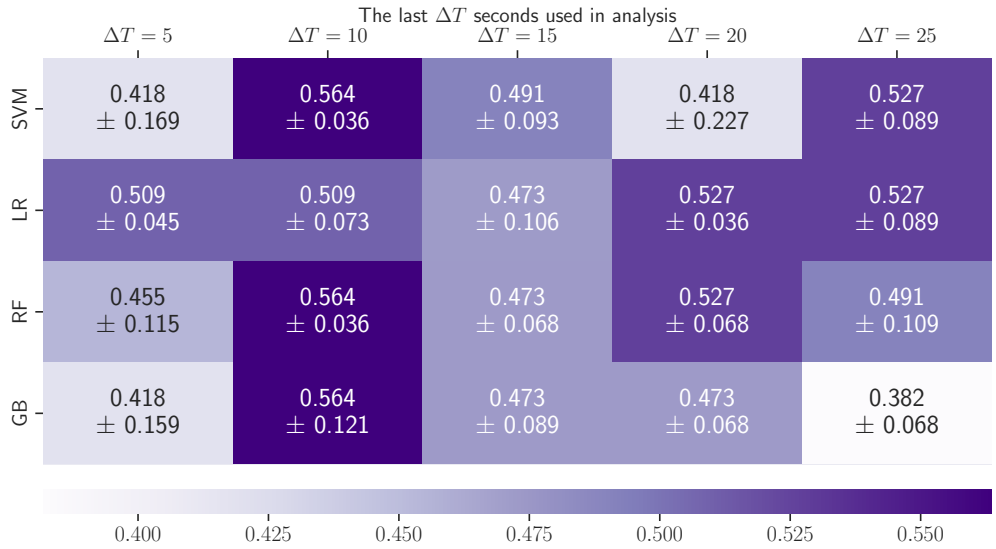


Figure 5.13: The test set classification scores obtained with the EEMD approach by excluding features from force signals. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

ously, crater wear-based labeling performs better than flank wear-based labeling. Therefore, this section only explains the results obtained with crater wear-based labeling. Carlsson Coordinates can detect the tool wear with 56.4% accuracy when $\Delta T = 5$ seconds. For the same time window, Carlsson Coordinates provides slightly higher accuracy compared to EEMD and DWT-based approaches (see Figures 5.9 5.12 and 5.15). However, DWT requires selecting thresholds for the Fourier spectrum to avoid selecting small sensitive frequencies, and EEMD based approach is computationally more expensive than the Carlsson Coordinates.

The classification scores obtained without including the features from force signals are presented in Figure 5.16. It is seen that Carlsson Coordinates can detect wear levels with 56.4% and 58.2% when $\Delta T = 5$ and $\Delta T = 10$, respectively. This indicates that features from the acoustic emission sensor and the microphone can only be used in data acquisition. Carlsson coordinates eliminate the need for features from the force dynamometer, whose setup is cumbersome and time-consuming. For the same time period ($\Delta T = 5$), Carlsson

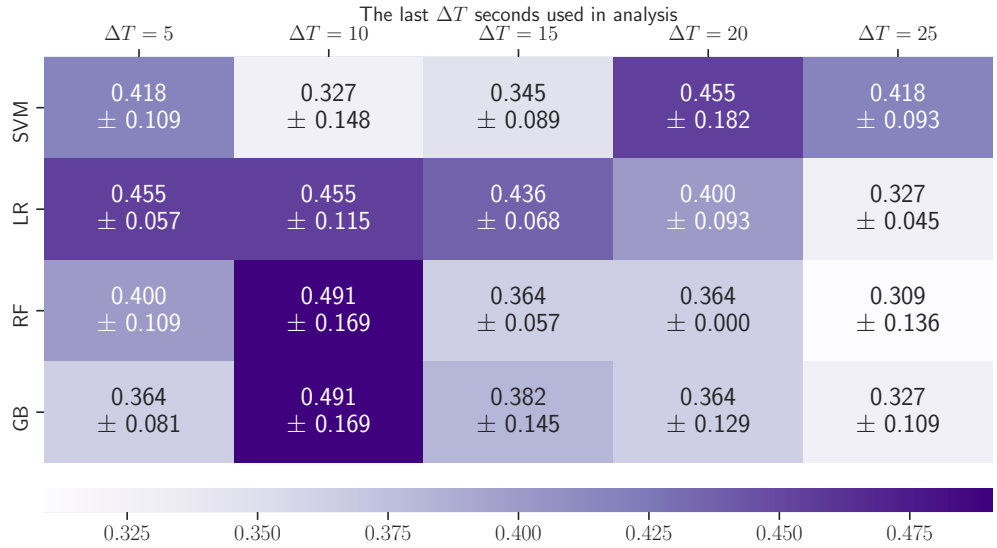


Figure 5.14: The test set classification scores obtained with the Carlsson Coordinates. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on flank wear measurements.

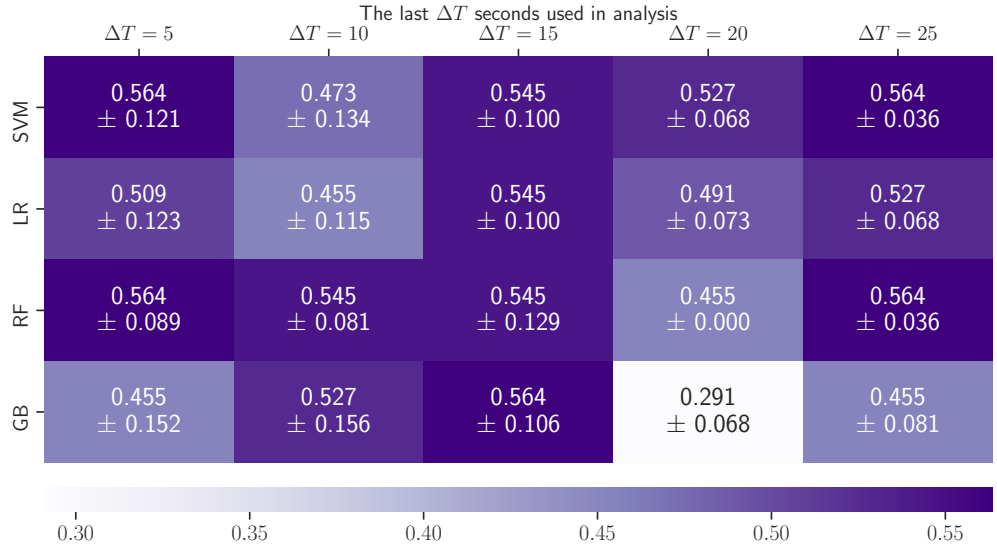


Figure 5.15: The test set classification scores obtained with the Carlsson Coordinates. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

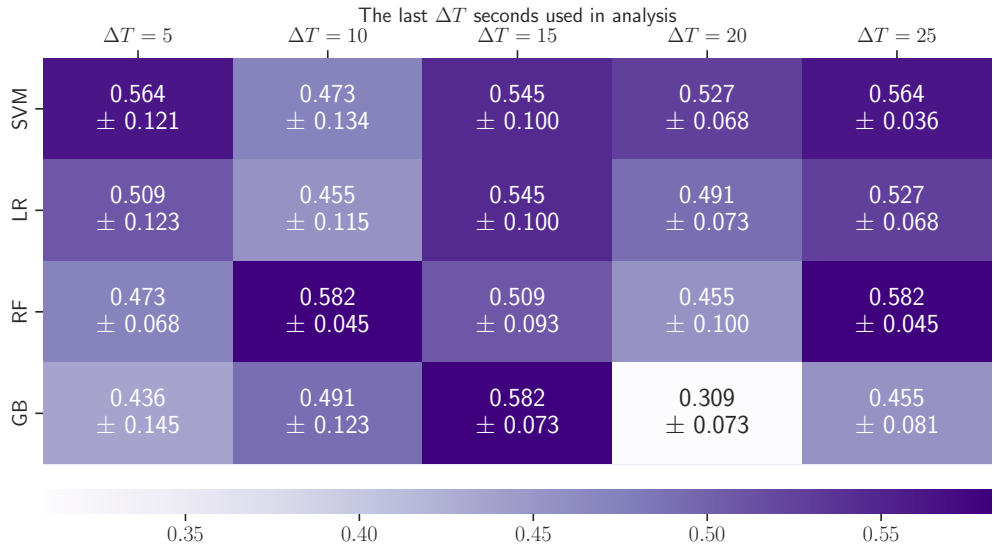


Figure 5.16: The test set classification scores obtained with the Carlsson Coordinates by excluding features from force signals. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

coordinates provides the highest score compared to EEMD and DWT (see Figures 5.10, 5.13, and 5.16).

The second featurization technique utilized for persistence diagrams is the persistence images. In this approach, persistence diagrams are converted to images, and their pixel values are used as features in the classification. Figures 5.17 and D.7 provide the results obtained with two types of labeling. Crater wear-based labeling provides higher accuracies than flank wear-based labeling. Similar to Carlsson Coordinates, persistence images capture the information related to tool wear in a five-second time window with 56.4% accuracy and lower deviation compared to persistence images. When the features of the force signals are excluded, the resulting classification scores in Figure 5.18 decrease 2%. Highest classification scores are obtained when $\Delta T = 10$ and $\Delta = 15$ seconds.

The last featurization approach is the template functions. Since crater wear-based labeling performs better than flank wear-based labeling, the results for flank wear can be found

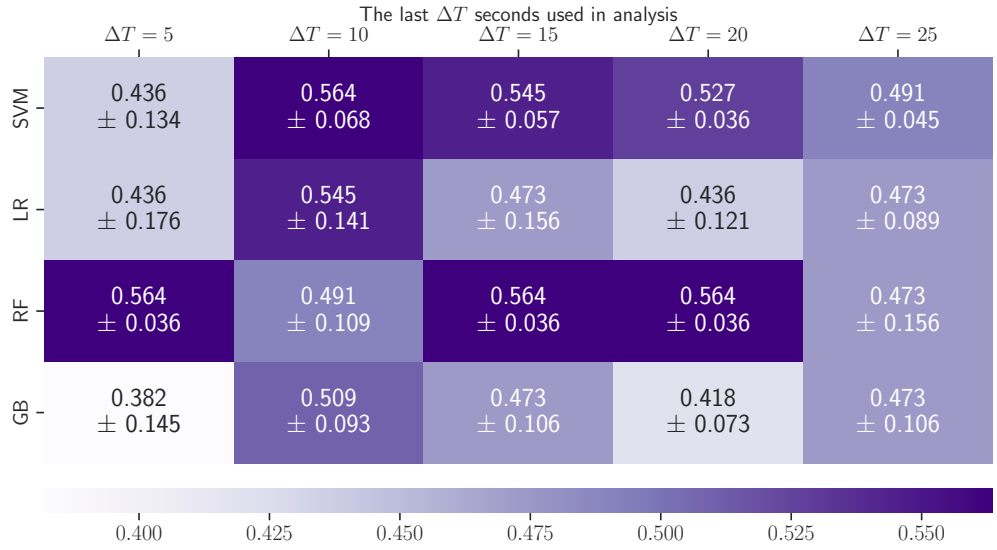


Figure 5.17: The test set classification scores obtained with the persistence images. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

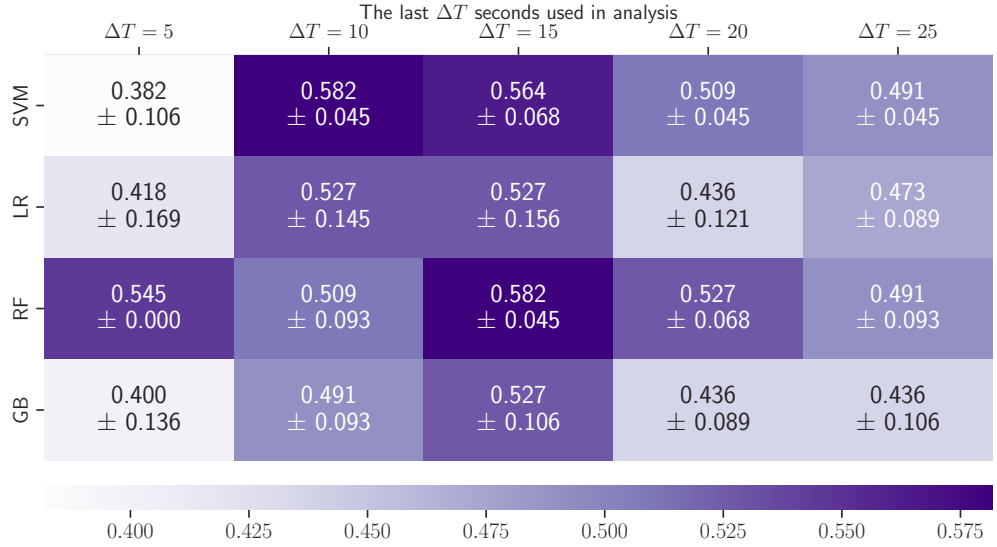


Figure 5.18: The test set classification scores obtained with the persistence images by excluding features from force signals. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

in Appendix Figure D.10. Figures 5.19 and 5.20 show that ignoring features of force signals increased the highest accuracy obtained out of combinations of four different classifiers and five different ΔT parameters. Template functions, including the features from force signals, are able to identify the wear level with 56.4%, while the accuracy increases up to 60% without including force signals. 60% accuracy is the highest accuracy obtained out of five approaches. While persistence images can provide this accuracy with five seconds time window, DWT needs at least 15 seconds of the time series to reach the same accuracy (see Figures 5.10 and 5.16).

These results show that TDA-based approaches can capture the information related to tool wear from the time series in a shorter time window. As a result, the user does not need to collect a large number of samples to identify the wear level. This also brings an advantage in terms of the runtime of the algorithms. Providing larger classification scores for smaller time series and eliminating the need to use a force dynamometer in data acquisition can make the integration of TDA-based approaches into real-time tool wear analysis easier than DWT and EEMD based approaches.

5.5 Conclusion

This study contributes to the DWT-based approach widely adopted in the literature by introducing an automatic and adaptive algorithm to find the sensitive frequency in the Fourier spectrum. EEMD based approach is also used as a benchmark case to compare results. In addition, it is believed that this is the first study to implement persistence homology from TDA for tool wear analysis in machining.

It has been shown that DWT and EEMD based approaches are dependent on the features from force signals when a shorter time is chosen for analysis. However, TDA-based approaches can still provide similar accuracies when the features of force signals are excluded. Setting up a force dynamometer is time-consuming, and it requires additional fixtures to be designed for the cutting lathe. TDA-based approaches can eliminate the usage of force dy-

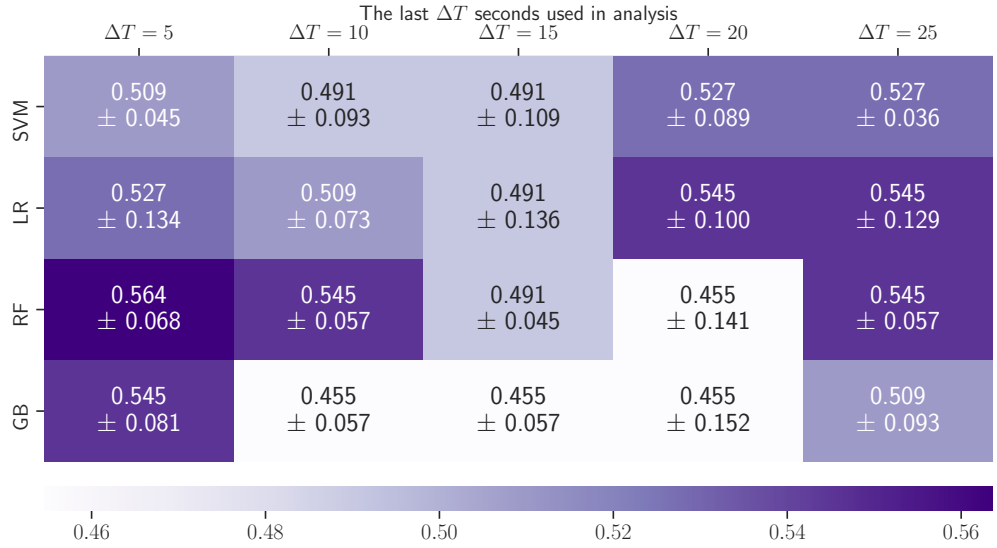


Figure 5.19: The test set classification scores obtained with the template functions. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

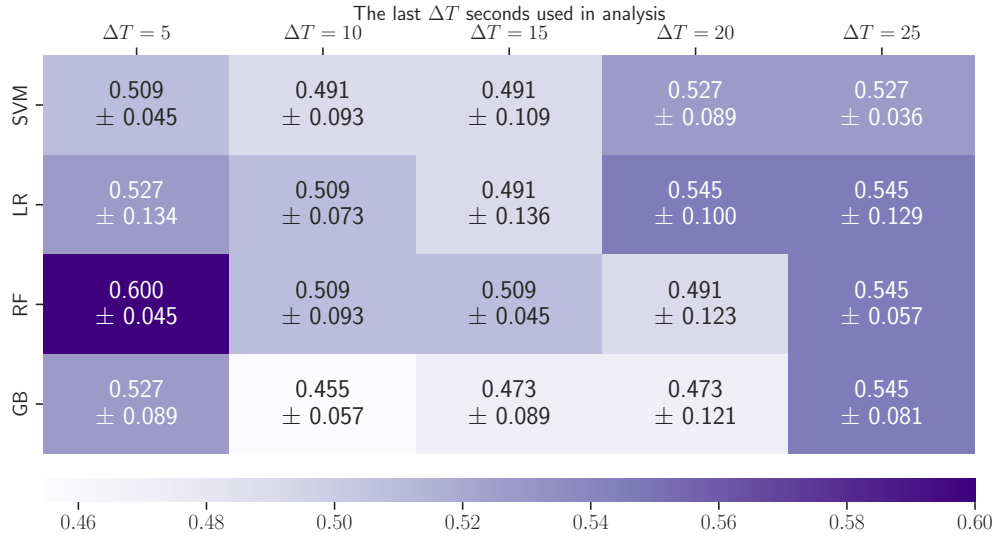


Figure 5.20: The test set classification scores obtained with the template functions by excluding features from force signals. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on crater wear measurements.

namometer in data acquisition. Therefore, they can provide more efficient data acquisition in terms of time and money.

DWT can provide similar accuracies with TDA-based approaches when smaller ΔT is chosen for the analysis and features from force signals are included. Although the proposed algorithm for DWT to identify the sensitive frequencies is automatic and adaptive, it requires users to enter the minimum desired frequency. This parameter needs to be chosen to avoid the selection of small frequencies as sensitive frequencies. The user needs to know the structural modes of the experimental setup or review the spectrum to identify frequencies whose amplitudes are not significant. Therefore, this challenges DWT to be implemented in a real-time tool wear analysis pipeline.

The EEMD approach can provide similar accuracies with large deviation compared to the TDA-based approach when features of the force signals are included, and a small ΔT is used. To obtain similar scores with smaller deviations, the user needs to include the last 15 seconds of the time series in the analysis. Including more data points will also increase the time required to compute the IMFs, while the computation of IMFs is computationally expensive in Python, even for smaller ΔT. Therefore, implementing EEMD into real-time tool wear analysis requires additional optimization to decrease the runtime for IMF computations.

TDA-based approaches provide promising results with or without features from force signals. Their performance either exceeds or matches the results obtained from EEMD based approaches. The machine learning framework with TDA-based approaches does not require the user to make assumptions. The pipeline is automatic and adaptive. While the greedy permutation option of Ripser package is utilized to compute persistence diagrams in a reasonable time in this chapter, this work previously showed that the combination of parallel computing and the Bézier curve approximation technique can significantly reduce the runtime for persistence diagram computation. Therefore, users have multiple choices to utilize for a faster pipeline. It is worth noting that point clouds obtained after embedding are subsampled to 1000 points with the greedy permutation option of Ripser. Although this is

a significantly smaller number compared to the original size of the point clouds, TDA-based approaches can provide similar classification scores to DWT and EEMD based approaches. This study hypothesizes that approximated persistence diagrams obtained from the Bézier curve approach can lead to higher classification scores. Therefore, future work of this study can include obtaining the results with persistence diagrams obtained from the combination of Bézier curve approximation and parallel computing. Another direction of future studies can focus on feature ranking to identify the pixels that provide the most informative features in persistence images. This will show us whether focusing on the areas that provide the most informative features can lead to better accuracy. In addition to improving classification accuracy, real-time tool wear analysis can be built using the TDA-based approaches, which show promising results for offline tool wear analysis in this study.

CHAPTER 6

CONCLUDING REMARKS

This research mainly focuses on machine learning applications in dynamical systems. Specifically, manufacturing applications and the parameter identification of complex systems are the two main areas investigated in this study. It has been observed that the signal decomposition approaches are widely utilized in chatter diagnosis, tool wear analysis, and surface texture analysis. While these approaches can provide high classification accuracy in some cases, this study shows that they are user-dependent and require manual preprocessing of the data set. The main goal of this study is to improve existing approaches to reduce human intervention and introduce novel approaches based on Topological Data Analysis.

For chatter detection, a novel approach based on similarities between the time series is developed in this study for chatter diagnosis. It implements DTW and combines this algorithm with KNN to classify cutting signals. In addition, persistence homology is utilized in chatter diagnosis with six different featurization techniques of persistence diagrams. It is shown that the results of DTW and TDA-based approaches can either match or exceed the classification scores obtained with existing approaches. This study also improves the runtime of the proposed approaches by implementing High Performance Computing tools. For small batch operations, this study proposes the combination of DTW with AESA to reduce the number of distance computations in the test phase. For the TDA-based approach, persistence diagrams are approximated using Bézier curves. These improvements in the pipeline can allow users to complete the classification of a single time series in less than two seconds. Therefore it is shown that TDA is viable for real-time chatter diagnosis approaches.

For parameter identification of dynamical systems, this study investigates one of the most popular approaches called SINDy. This approach is generally used in literature with data obtained from analytical models, while the experimental studies are limited. Therefore, this study tests the performance of SINDy with a complex single pendulum apparatus for

the first time, and it reports the caveats of SINDy while making modifications in derivative estimation steps of the approach.

Surface texture is another application that is taken into account in this study. Two widely adopted approaches, DCT and DWT, are investigated in this study. Two automatic and adaptive threshold selection algorithms are proposed for these approaches. It is seen that the proposed algorithms can match the results obtained from heuristic threshold selection while removing manual preprocessing of the surface scans and surface profiles. In addition to these two approaches, sublevel set persistence from TDA is utilized to summarize the information in given surfaces. The resulting classification accuracies are either higher than the results of traditional feature extraction approaches or match them. Therefore, this study also provides another TDA-based approach called topological saliency to cluster surface patches to identify the regions where additional machining is required. It is believed that further research is needed to implement topological saliency into real-time cutting operations to reduce material waste and provide an efficient surface finish monitoring system.

Tool wear analysis is studied in the last chapter of this study. DWT and EEMD based approaches are applied to experimental titanium cutting signals. A new automatic and adaptive algorithm is proposed to select sensitive frequencies in the Fourier spectrum for the DWT approach. In addition, featurization techniques from TDA tools are utilized to classify time series based on their wear level. TDA-based approaches show that they can detect the wear level in an experiment with a similar amount of samples needed by the DWT approach, while the EEMD approach needs a larger number of samples compared to the other two. However, DWT requires the user to select minimum desired frequencies to locate sensitive frequencies correctly. Moreover, the TDA-based approach can still show the same classification performance when force signal data is excluded from the experiment. On the other hand, DWT and EEMD show a decrease in accuracy or need more data samples to obtain the same performance with the analysis, including force sensor data. Therefore, the TDA-based approach does not need an expensive force sensor, and it can save the

time needed to integrate a force dynamometer into an experimental setup. The TDA-based approach provides a fully automated pipeline without needing user input in addition to several options to speed up the computation, such as Bézier curve approximation, parallel computing, and greedy permutation. Therefore, TDA is shown to be a viable option for real-time tool wear analysis. One future direction of this study can include the prediction of tool wear in real-time using standard and novel approaches.

APPENDICES

APPENDIX A

CUTTING EXPERIMENTS

A.1 Aluminum Cutting Experiment

Figure A.1 shows the turning experiment that was used to collect the measurement data for training and testing the chatter detection algorithms. It consists of a 6061 aluminum cylindrical workpiece mounted into the chuck of the spindle of a Clausing-Gamet 33 cm (13 inch) engine lathe. An S10R-SCLCR3A boring bar from the Grizzly T10439 carbide insert boring bar set with an attached 0.04 cm (0.015 inch) radius Titanium nitride coated cutting insert is secured to the tool holder.

The rod's stiffness, and therefore, the eigenfrequencies of the tool vibration, are varied by changing the overhang or stickout length of the rod. Four stickout lengths are used in the experiment: 5.08 cm (2 inch), 6.35 cm (2.5 inch), 8.89 cm (3.5 inch), and 11.43 cm (4.5 inch). In order to obtain more accurate measurements, the stickout length is measured as the distance between the flat back surface of the tool holder and the heel of the boring rod. The visual representation of stickout distance is given on the right hand side of FigureA.1. Increasing the stickout length leads to a stiffer cutting tool and higher eigenfrequencies for lateral vibrations. Since the lateral direction is the most flexible and chatter frequencies appear in the neighborhood of dominant eigenfrequencies of the structure, the dominant chatter frequencies increase with increasing stickout length.

The boring rod is instrumented with two PCB 352B10 miniature, lightweight, uni-axial ceramic shear accelerometers that are ninety degrees apart to measure lateral vibrations of the rod. The two accelerometers are superglued onto the rod at about 3.81 cm (1.5 inch) away from the cutting tool to protect them from moving parts and cutting debris. A PCB 356B11 triaxial, miniature ceramic shear accelerometer is also attached to the bottom clamp of the tool holder, as shown in Figure A.1. The data from all the accelerometers are collected on

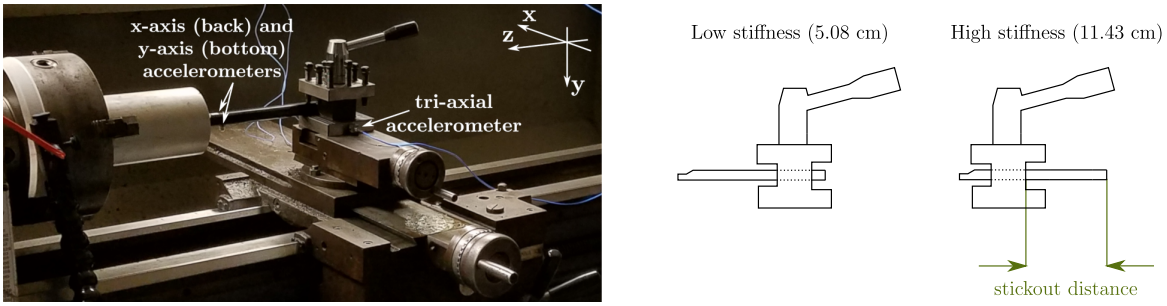


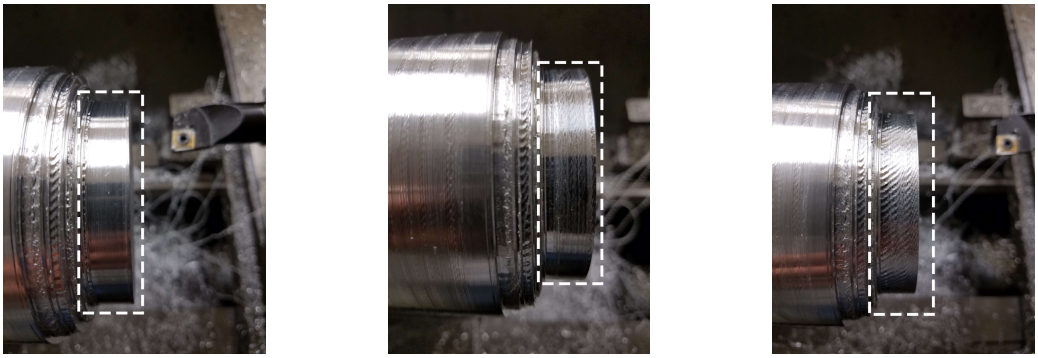
Figure A.1: The experimental setup showing the workpiece, the cutting tool, and the attached accelerometers (left). The visual representation of the stickout distance (right).

the analog channels of an NI USB-6366 data acquisition box using Matlab. No in-line analog filter is used; however, the signals are oversampled at 160kHz. Digital filtering is used before subsampling, thus eliminating noise while avoiding the undesirable effects of antialiasing. In particular, a Butterworth low-pass filter with order 100 and a cutoff frequency of 10 kHz is used. The data is then downsampled to 10 kHz without the risk of causing aliasing effects. The resulting conditioned data is what is considered in Section A.1.1. In addition, both the raw and filtered data are provided in a Mendeley repository [312]. Also, one can find the codes for WPT and EEMD approaches in a GitHub repository.

A.1.1 Data Labeling

Before tagging the signals, the time series of the two uni-axial accelerometers on the boring rod are analyzed as well as the signals from the tri-axial accelerometer on the tool post, see Figure A.1. It is found that although the data of the accelerometers is mostly redundant, the x-vibration at the tool post, which is measured by the x-axis signal of the tri-axial accelerometer, had the best signal-to-noise ratio. Therefore, the data tagging is performed exclusively using the data from this channel. Another sanity check was the comparison of the tagged signals with a few photographs of the resulting machined surface taken during the experiment, as shown in Figure A.2.

Each time series from every cutting test was examined, and the different parts of the signals were labeled as either no chatter, mild/intermediate chatter, chatter, or unknown.



(a) No chatter. (b) Mild chatter. (c) Chatter.

Figure A.2: The surface finish corresponding to (a) no-chatter case with 6.35 cm (2.5 inch) stickout length, 320 rpm, and 0.127 mm (0.005 inch) depth of cut, (b) Mild chatter case with 6.35 cm (2.5 inch) stickout length, 770 rpm, and 0.127 mm (0.005 inch) depth of cut, and (c) chatter case with 6.35 cm (2.5 inch) stickout length, 570 rpm, and 0.127 mm (0.005 inch) depth of cut.

Figure A.3a shows an example of how one time series is labeled using these categories. The separation into different parts has been done based on the characteristics of the amplitude in the time domain. In particular, parts with a low amplitude were separated from parts with a large amplitude. In addition, parts with an impact-like structure with an abrupt, very strong increase of the accelerations and a relatively fast decay were also separated. Then the frequency domain characteristics were studied for final classification of the signal. In the frequency domain, only the frequency components lower than 5 kHz were considered. Specifically, the criteria used for classifying the signals are:

1. No chatter (stable):
    a) Low amplitude in the time domain
    b) Low amplitude in the frequency domain (highest peaks at spindle rotation frequencies [313])
2. Mild or intermediate chatter:
    a) Low amplitude in the time domain
    b) Large amplitude in the frequency domain (highest peaks at chatter frequencies)

3. Chatter:

    a) Large amplitude in time domain

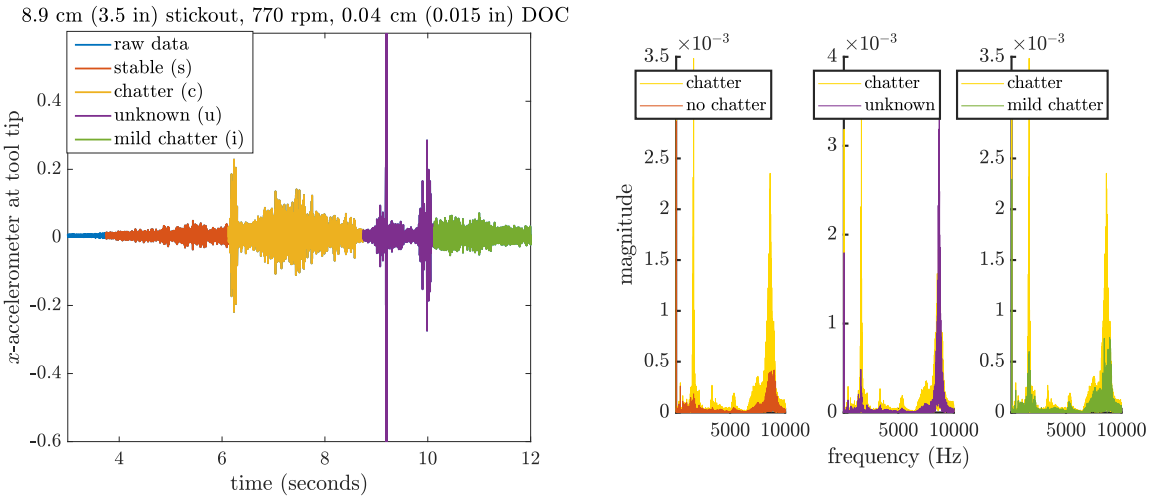
    b) Large amplitude in the frequency domain (very high peaks at chatter frequencies which are not equal to the spindle rotation frequencies)

4. Unknown:

    a) All other cases

The unknown data are parts of the time series with large amplitude in the time domain but no large peaks in the frequency domain at chatter frequencies (lower than 5 kHz). Typically, this corresponds to the parts with impact-like structure, which might occur due to chip breakage or other inhomogeneities during the process. Also, there can be another eigenmode at 10kHz, which is vibrating (chattering) for the unknown portion of the time series in FigureA.3. Figure A.4 also provides time and frequency domain of the cutting configurations provided in Figure A.2. However, typical chatter frequencies in this process lie between 0-150 Hz for structural modes, 200-800 Hz for workpiece vibrations, and 1000-3000 Hz for tool vibrations. Therefore, it is not clear if this is chatter or something else, and therefore it was excluded from the analysis. The time domain and the frequency domain characteristics for an example time series that includes all four classes are shown in Figure A.3a and Figure A.3b, respectively. The first part of the time series was not classified because, in this case, the tool is still not engaged in the workpiece.

From Figure A.3, it also becomes clear that a process with fixed cutting conditions is not necessarily clearly stable or unstable, which is another reason why chatter detection algorithms might be helpful for practical applications. For example, the process in Figure A.3 is stable at the beginning between 4s and 6s. Then, a strong perturbation drives the system away from the stable state to some chattering motion, which is a reasonable scenario because there can be bistability between stable cutting and chatter [314, 315, 316]. Table A.1 shows the breakdown and the total number of the tagged time series for each stickout length.



(a) time series (b) spectrum comparison

Figure A.3: Tagging example in the (a) time domain and (b) the frequency domain for the case with 8.9 cm (3.5 in) stickout, 770 rpm, and 0.04 cm (0.015 in) depth of cut.

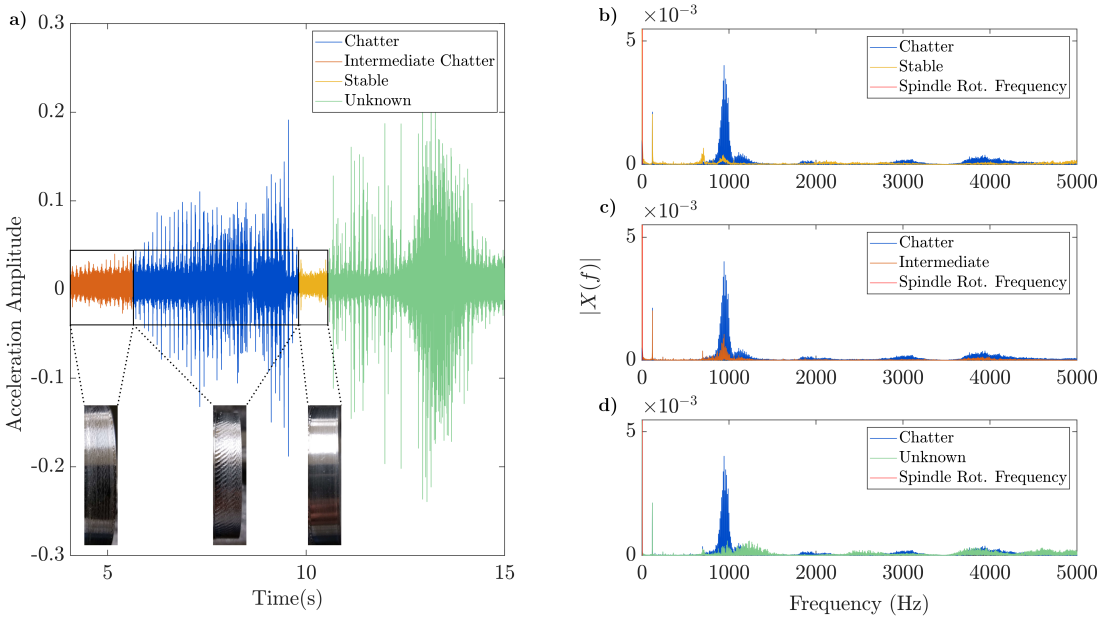


Figure A.4: (a) Sample tagged time series and some samples of the resulting surface finish. The right panel shows a comparison of the frequency spectra between a signal labeled as chatter versus (b) no chatter, (c) intermediate chatter, and (d) unknown.

Processed time series are split into smaller pieces whose lengths are around 10000 to reduce the computational time in several feature extraction methods. The numbers provided in parenthesis in Table A.1 represent the number of time series after splitting.

Table A.1: Case numbers before and after splitting (in parenthesis) the time series and overall amount of tagged data for each stickout case.

Stickout length (cm (inch))	Stable	Mild chatter	Chatter	Total
5.08 (2)	17 (443)	8 (31)	11 (118)	36 (592)
6.35 (2.5)	7 (82)	4 (33)	3 (13)	14 (128)
8.89 (3.5)	7 (55)	2 (5)	2 (6)	11 (66)
11.43 (4.5)	13 (114)	4 (14)	5 (48)	22 (176)

A.2 Milling Experiment

This section describes the experimental setups for milling. The experimental data is received from Reference [317]. An illustration showing the experimental milling system is shown in Fig. A.5. An Ingersol machining center with a Fischer 40000 rpm and 40kW spindle was used to conduct experiments on an aluminum workpiece (7050-T7451). The type of milling conducted in these experiments is down milling. The depth of cut is 2.03 mm, and radial immersion is kept constant at 5%. Lion precision capacitive probes were used to collect the tool displacements along the x, and y axes [317]. The data were sampled at 25 kHz. As in the turning experiments, a low pass filter was used, and the data was downsampled to 12.5 kHz. In addition, a laser tachometer was used to independently verify the spindle rotational speed from the machine setting. The cutting tool was a 19.05 mm end mill with two teeth and a 106 mm overhang distance. Data tagging was performed using power spectral density (PSD) plots and Poincaré sections. Tool displacement plots in the x direction, along with the corresponding Poincaré sections, are shown in Figure A.6.

Table A.2: Cutting parameters for the turning and milling experiments.

Cutting operation	Rotational Speeds (rpm)	Depth of Cut (mm)	Feed Rate
Milling	11206-32161	0.381 - 3.556	0.191 mm/tooth/rev

The first milling example shown in Figure A.6 is a stable cut ($\Omega = 19488$ rpm, 1.524 mm cutting depth), whereas the second example shows a Hopf bifurcation example ($\Omega = 27285$ rpm, 3.556 mm cutting depth). The first column of Figure A.6 presents the tool displacements

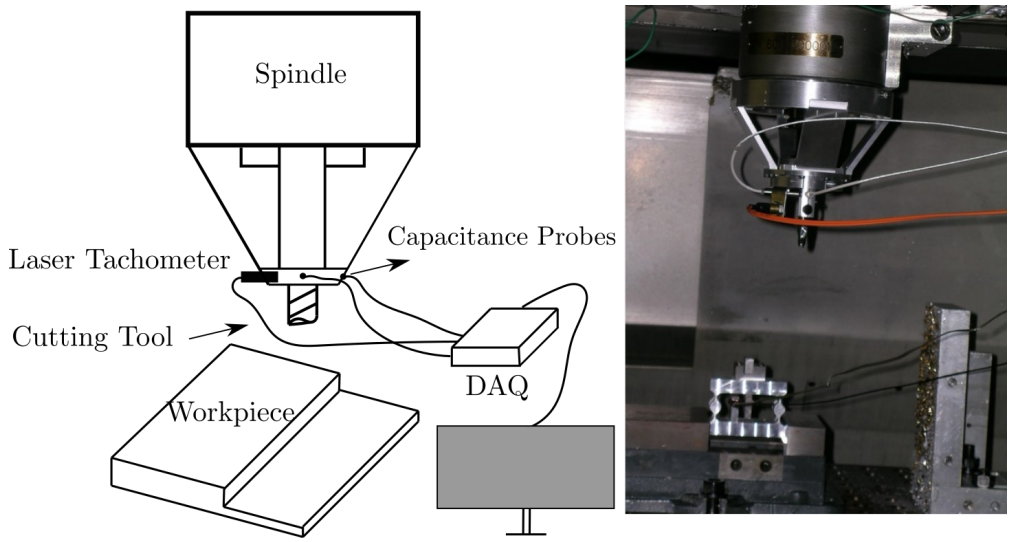


Figure A.5: Experimental setup of milling cutting experiments.(left) Illustration of the setup (right) picture of the cutting tool and the workpiece.

for two teeth and the second column provides the Poincare sections of these time series. x_n and x_{n2} represent the time-delayed coordinates. In this study, the constant time delay parameter is used, and it is chosen as 6. The third column of Figure A.6 shows the power spectrum or a PSD plot that helped us better see the frequency content of the time series [317]. If the spectral peaks were synchronous with the tooth passage frequency, that meant the corresponding time series was stable. However, if the spectral peaks were not aligned with the tooth passage frequency, the cutting test was unstable, as shown in the second row and third column of Figure A.6.

Stability predictions were made for the milling system using the measured modal parameters and the spectral element approach [2]; the resulting stability diagram is shown in Figure A.8. The spectral element method uses eigenvalues of a dynamic map to determine the stability of the process [317, 3]. If the magnitude of the eigenvalue is smaller than 1, the process is stable. If the eigenvalue has only a positive real part that is larger than 1, then the process is unstable. Eigenvalues with only negative real part less than -1 represent the flip bifurcations, and Hopf bifurcation occurs when an eigenvalue has an amplitude larger than 1. The illustration for the stability analysis using the real and imaginary parts of the

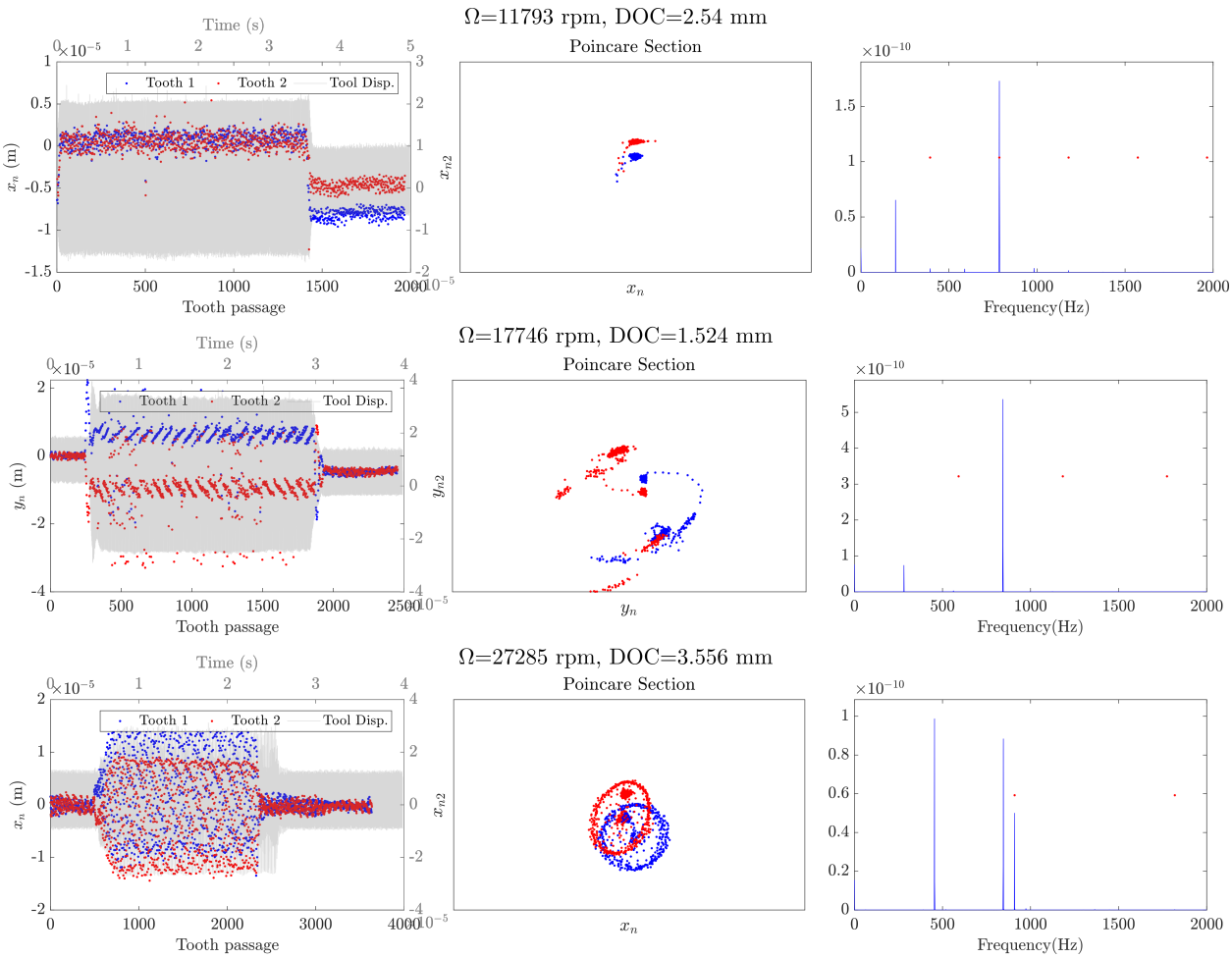


Figure A.6: The first column represents tool displacements for two teeth, and the second column provides Poincare sections for two time series obtained with three different rotational speeds and depth of cuts in milling experiments. The third column shows the PSD plots of the three-time series whose Poincare sections are shown in the first column. Red dots represent tooth passage frequency. The time series shown in the first row is stable with cutting conditions $\Omega = 11793$ rpm and depth of cut of 2.54 mm, while the one in the second row is unstable with cutting conditions $\Omega = 17746$ rpm and depth of cut of 1.524 mm. The third row represents an unstable milling signal with cutting conditions $\Omega = 27285$ rpm and depth of cut of 3.556 mm.

eigenvalues is also provided in Figure A.7. Based on this stability criteria, 10000 time series for a 100×100 grid of points in the rpm vs. cutting depth parameter space were used to produce the stability diagram shown in Figure A.8. The stability of the experimental data set is decided based on the Poincare section and the PSD plots. Then experimental data

set is included in the stability diagram shown with black diamonds (unstable) and triangle (stable) markers in Figure A.8. It is seen that the labels obtained using the eigenvalues and the ones obtained using frequency domain analysis and the Poincare section may not match. The labels obtained from the analysis shown in Figure A.6 are used to perform classification.

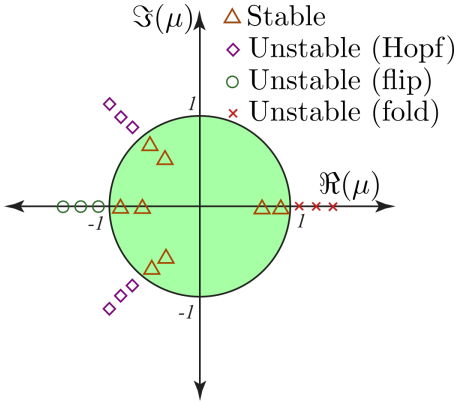


Figure A.7: Illustration for the stability analysis using eigenvalues of the dynamic map obtained using spectral element approach [2].

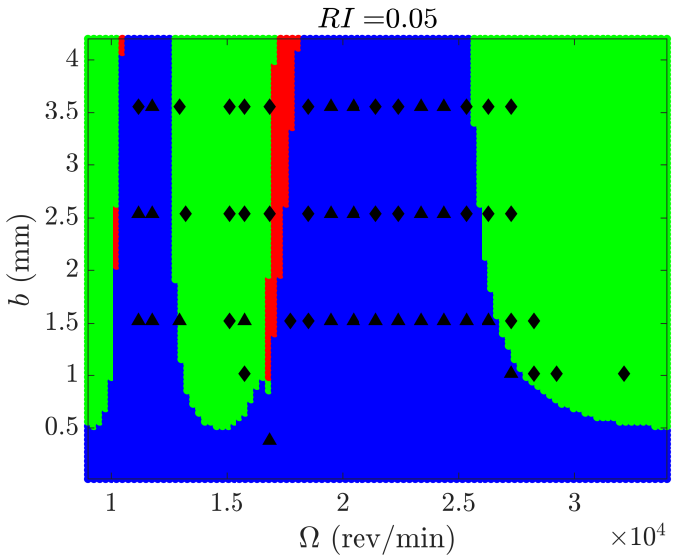


Figure A.8: The stability of time series obtained using the analytical model provided in Section 2.6.5 [3] with different depth of cut (b) and spindle speed (Ω) on 100×100 grid. The green color corresponds to the time series with Hopf bifurcation (unstable), while the blue color represents the stable time series. The red color shows the time series with flip bifurcation. Experimental data, whose stability is defined based on the Poincare section and PSD plots, is shown with diamond (unstable) and triangle (stable) symbols.

APPENDIX B

CLASSIFICATION RESULTS FOR CHATTER DETECTION

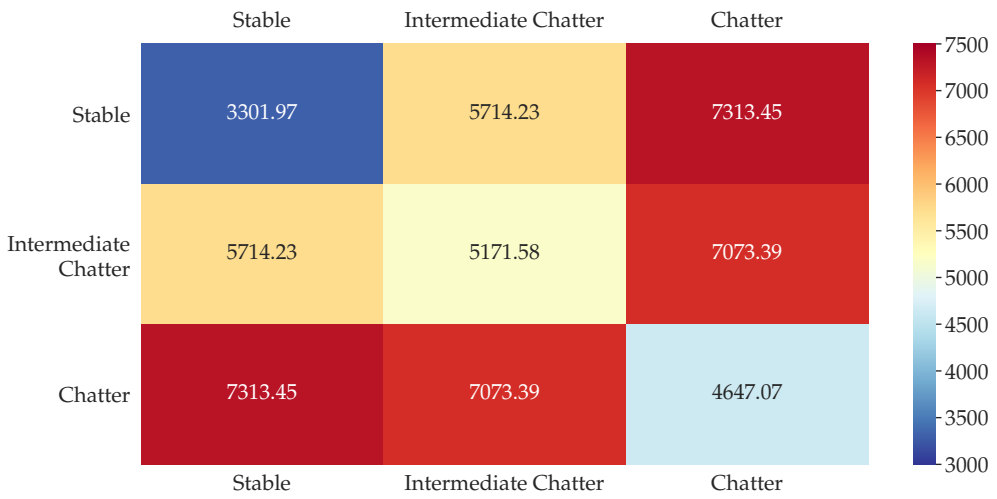


Figure B.1: The heat map of averages DTW distances of time series belongs to three classes for the 8.89 cm (3.5 inch) case.

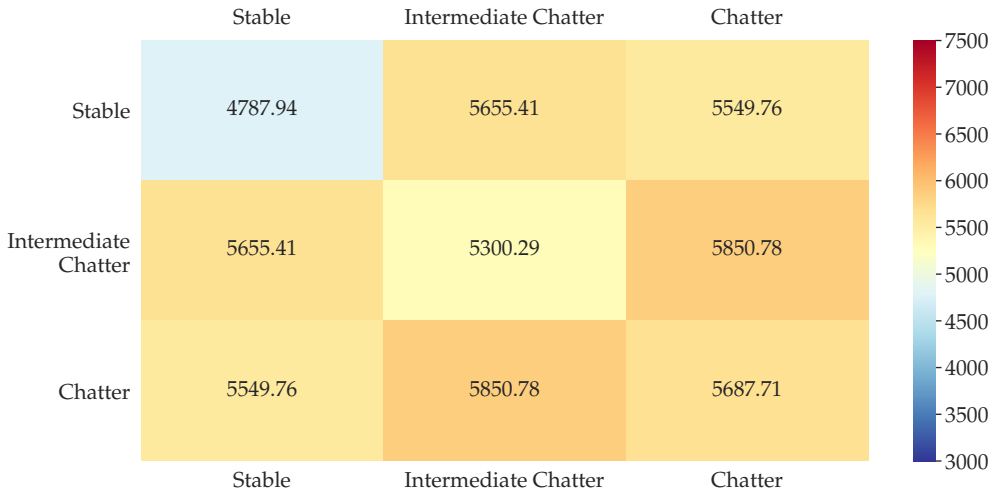


Figure B.2: The heat map of averages DTW distances of time series belongs to three classes for the 11.43 cm (4.5 inch) case.

Table B.1: Results obtained with Level 1 Wavelet Packet Transform feature extraction for 2, 2.5, 3.5, and 4.5 inch stickout cases.

Classifier: SVM	5.08 cm (2 inch)		6.35 cm (2.5 inch)	
Features	Test Set	Training Set	Test Set	Training Set
r_1	93.1%6.4%	85.0% ± 3.6%	73.3% ± 21.3%	83.0% ± 13.5%
r_1,r_2	93.9% ± 5.8%	85.8% ± 3.0%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,r_2,r_3	92.3% ± 11.4%	89.6% ± 1.8%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_4	90.8% ± 13.7%	90.0% ± 3.1%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_5	89.2% ± 13.9%	89.2% ± 3.4%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_6	81.5% ± 15.1%	85.0% ± 4.7%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_7	76.9% ± 14.6%	85.0% ± 4.7%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_8	78.5% ± 13.7%	85.8% ± 6.0%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_9	78.5% ± 13.7%	86.2% ± 5.5%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_{10}	78.5% ± 13.7%	86.2% ± 5.5%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_{11}	78.5% ± 13.7%	86.2% ± 5.5%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_{12}	78.5% ± 13.7%	86.2% ± 5.5%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_{13}	78.5% ± 13.7%	86.2% ± 5.5%	85.0% ± 22.9%	100.0% ± 0.0%
r_1,...,r_{14}	78.5% ± 13.7%	86.2% ± 5.5%	85.0% ± 22.9%	100.0% ± 0.0%
Classifier: SVM	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
r_1	56.0% ± 17.4%	83.3% ± 11.4%	78.8% ± 14.8%	83.6% ± 8.5%
r_1,r_2	84.0% ± 15.0%	100.0% ± 0.0%	68.8% ± 17.0%	82.9% ± 9.7%
r_1,r_2,r_3	84.0% ± 15.0%	100.0% ± 0.0%	67.5% ± 13.9%	80.7% ± 12.4%
r_1,...,r_4	84.0% ± 15.0%	100.0% ± 0.0%	68.8% ± 15.1%	82.9% ± 9.7%
r_1,...,r_5	84.0% ± 15.0%	100.0% ± 0.0%	68.8% ± 15.1%	82.9% ± 9.7%
r_1,...,r_6	84.0% ± 15.0%	100.0% ± 0.0%	73.8% ± 10.4%	83.6% ± 10.1%
r_1,...,r_7	84.0% ± 15.0%	100.0% ± 0.0%	76.3% ± 10.4%	84.3% ± 7.7%
r_1,...,r_8	84.0% ± 15.0%	100.0% ± 0.0%	76.3% ± 10.4%	84.3% ± 7.7%
r_1,...,r_9	84.0% ± 15.0%	100.0% ± 0.0%	76.3% ± 10.4%	84.3% ± 7.7%
r_1,...,r_{10}	84.0% ± 15.0%	100.0% ± 0.0%	76.3% ± 10.4%	84.3% ± 7.7%
r_1,...,r_{11}	84.0% ± 15.0%	100.0% ± 0.0%	76.3% ± 10.4%	84.3% ± 7.7%
r_1,...,r_{12}	84.0% ± 15.0%	100.0% ± 0.0%	76.3% ± 10.4%	84.3% ± 7.7%
r_1,...,r_{13}	84.0% ± 15.0%	100.0% ± 0.0%	76.3% ± 10.4%	84.3% ± 7.7%
r_1,...,r_{14}	84.0% ± 15.0%	100.0% ± 0.0%	76.3% ± 10.4%	84.3% ± 7.7%

Table B.2: Results obtained with Level 2 Wavelet Packet Transform feature extraction for 2, 2.5, 3.5 and 4.5 inch stickout cases.

Classifier: SVM	5.08 cm (2 inch)		6.35 cm (2.5 inch)	
Features	Test Set	Training Set	Test Set	Training Set
r_1	92.3% ± 6.0%	93.8% ± 1.9%	100.0% ± 0.0%	100.0% ± 0.0%
r_1,r_2	90.8% ± 8.3%	93.5% ± 1.8%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,r_2,r_3	90.0% ± 6.9%	91.5% ± 4.1%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_4	90.0% ± 6.9%	91.5% ± 4.1%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_5	91.5% ± 7.3%	92.7% ± 3.6%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_6	76.2% ± 11.1%	78.5% ± 8.8%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_7	83.8% ± 7.3%	78.5% ± 7.5%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_8	84.6% ± 8.4%	77.3% ± 6.8%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_9	81.5% ± 7.8%	76.5% ± 6.1%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_{10}	81.5% ± 7.8%	76.5% ± 6.1%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_{11}	81.5% ± 7.8%	76.5% ± 6.1%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_{12}	81.5% ± 7.8%	76.5% ± 6.1%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_{13}	81.5% ± 7.8%	76.5% ± 6.1%	95.0% ± 7.6%	100.0% ± 0.0%
r_1,...,r_{14}	81.5% ± 7.8%	76.5% ± 6.1%	95.0% ± 7.6%	100.0% ± 0.0%
Classifier: SVM	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
r_1	70.0% ± 22.4%	78.9% ± 11.6%	66.3% ± 16.8%	69.3% ± 15.7%
r_1,r_2	72.0% ± 16.0%	92.2% ± 10.0%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,r_2,r_3	72.0% ± 16.0%	93.3% ± 8.9%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_4	68.0% ± 25.6%	88.9% ± 14.1%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_5	74.0% ± 12.8%	87.8% ± 15.3%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_6	78.0% ± 14.0%	90.0% ± 10.5%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_7	76.0% ± 12.0%	86.7% ± 15.6%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_8	76.0% ± 12.0%	86.7% ± 15.6%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_9	76.0% ± 12.0%	86.7% ± 15.6%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_{10}	76.0% ± 12.0%	86.7% ± 15.6%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_{11}	76.0% ± 12.0%	86.7% ± 15.6%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_{12}	76.0% ± 12.0%	86.7% ± 15.6%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_{13}	76.0% ± 12.0%	86.7% ± 15.6%	87.5% ± 11.2%	82.1% ± 8.6%
r_1,...,r_{14}	76.0% ± 12.0%	86.7% ± 15.6%	87.5% ± 11.2%	82.1% ± 8.6%

Table B.3: Results obtained with Level 3 Wavelet Packet Transform feature extraction for 2, 2.5, 3.5 and 4.5 inch stickout cases.

Classifier: SVM	5.08 cm (2 inch)		6.35 cm (2.5 inch)	
Features	Test Set	Training Set	Test Set	Training Set
r_1	70.8% ± 13.2%	71.5% ± 4.3%	63.3% ± 20.8%	73.0% ± 19.0%
r_1,r_2	77.7% ± 8.7%	85.4% ± 3.4%	78.3% ± 7.6%	94.0% ± 12.0%
r_1,r_2,r_3	78.5% ± 6.7%	85.8% ± 2.5%	76.7% ± 8.2%	95.0% ± 10.2%
r_1,...,r_4	77.7% ± 7.3%	84.6% ± 3.8%	78.3% ± 10.7%	97.0% ± 6.4%
r_1,...,r_5	76.9% ± 4.9%	84.6% ± 6.9%	78.3% ± 10.7%	97.0% ± 6.4%
r_1,...,r_6	73.8% ± 13.0%	78.1% ± 7.5%	78.3% ± 10.7%	97.0% ± 6.4%
r_1,...,r_7	62.3% ± 11.6%	74.2% ± 8.9%	78.3% ± 10.7%	97.0% ± 6.4%
r_1,...,r_8	60.8% ± 11.1%	72.3% ± 7.7%	81.7% ± 11.7%	98.0% ± 4.0%
r_1,...,r_9	60.8% ± 11.1%	72.3% ± 7.7%	81.7% ± 11.7%	98.0% ± 4.0%
r_1,...,r_{10}	60.8% ± 11.1%	72.3% ± 7.7%	81.7% ± 11.7%	98.0% ± 4.0%
r_1,...,r_{11}	60.8% ± 11.1%	72.3% ± 7.7%	81.7% ± 11.7%	98.0% ± 4.0%
r_1,...,r_{12}	60.8% ± 11.1%	72.3% ± 7.7%	81.7% ± 11.7%	98.0% ± 4.0%
r_1,...,r_{13}	60.8% ± 11.1%	72.3% ± 7.7%	81.7% ± 11.7%	98.0% ± 4.0%
r_1,...,r_{14}	60.8% ± 11.1%	72.3% ± 7.7%	81.7% ± 11.7%	98.0% ± 4.0%
Classifier: SVM	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
r_1	62.0% ± 14.0%	73.3% ± 16.6%	85.0% ± 9.4%	83.6% ± 10.6%
r_1,r_2	60.0% ± 15.5%	80.0% ± 18.5%	72.5% ± 16.6%	81.4% ± 14.7%
r_1,r_2,r_3	66.0% ± 9.2%	83.3% ± 17.4%	72.5% ± 16.6%	81.4% ± 14.7%
r_1,...,r_4	66.0% ± 9.2%	83.3% ± 17.4%	75.0% ± 18.5%	82.1% ± 14.7%
r_1,...,r_5	64.0% ± 12.0%	82.2% ± 18.7%	75.0% ± 18.5%	82.1% ± 14.7%
r_1,...,r_6	58.0% ± 6.0%	81.1% ± 20.5%	75.0% ± 18.5%	82.1% ± 14.7%
r_1,...,r_7	68.0% ± 13.3%	83.3% ± 15.9%	77.5% ± 14.6%	84.3% ± 13.5%
r_1,...,r_8	64.0% ± 15.0%	78.9% ± 19.5%	76.3% ± 15.3%	87.1% ± 7.7%
r_1,...,r_9	64.0% ± 15.0%	78.9% ± 19.5%	76.3% ± 15.3%	87.1% ± 7.7%
r_1,...,r_{10}	64.0% ± 15.0%	78.9% ± 19.5%	76.3% ± 15.3%	87.1% ± 7.7%
r_1,...,r_{11}	64.0% ± 15.0%	78.9% ± 19.5%	76.3% ± 15.3%	87.1% ± 7.7%
r_1,...,r_{12}	64.0% ± 15.0%	78.9% ± 19.5%	76.3% ± 15.3%	87.1% ± 7.7%
r_1,...,r_{13}	64.0% ± 15.0%	78.9% ± 19.5%	76.3% ± 15.3%	87.1% ± 7.7%
r_1,...,r_{14}	64.0% ± 15.0%	78.9% ± 19.5%	76.3% ± 15.3%	87.1% ± 7.7%

Table B.4: Results obtained with Level 4 Wavelet Packet Transform feature extraction for 2, 2.5, 3.5 and 4.5 inch stickout cases.

Classifier: SVM	5.08 cm (2 inch)		6.35 cm (2.5 inch)	
Features	Test Set	Training Set	Test Set	Training Set
r_1	$43.1\% \pm 9.9\%$	$60.0\% \pm 3.9\%$	$50.0\% \pm 12.9\%$	$64.0\% \pm 10.2\%$
r_1,r_2	$69.2\% \pm 9.7\%$	$84.6\% \pm 9.1\%$	$75.0\% \pm 13.4\%$	$84.0\% \pm 18.5\%$
r_1,r_2,r_3	$69.2\% \pm 9.7\%$	$84.6\% \pm 9.1\%$	$65.0\% \pm 15.7\%$	$84.0\% \pm 18.5\%$
r_1,...,r_4	$72.3\% \pm 8.6\%$	$85.7\% \pm 8.6\%$	$58.3\% \pm 20.1\%$	$80.0\% \pm 20.0\%$
r_1,...,r_5	$88.5\% \pm 5.1\%$	$94.6\% \pm 3.1\%$	$61.7\% \pm 16.8\%$	$76.0\% \pm 20.1\%$
r_1,...,r_6	$80.0\% \pm 3.7\%$	$88.8\% \pm 9.0\%$	$61.7\% \pm 21.1\%$	$72.0\% \pm 25.6\%$
r_1,...,r_7	$84.6\% \pm 9.1\%$	$89.2\% \pm 6.2\%$	$61.7\% \pm 21.1\%$	$68.0\% \pm 24.0\%$
r_1,...,r_8	$84.6\% \pm 9.1\%$	$89.2\% \pm 6.2\%$	$51.7\% \pm 21.7\%$	$74.0\% \pm 21.1\%$
r_1,...,r_9	$84.6\% \pm 9.1\%$	$89.2\% \pm 6.2\%$	$51.7\% \pm 21.7\%$	$74.0\% \pm 21.1\%$
r_1,...,r_{10}	$84.6\% \pm 9.1\%$	$89.2\% \pm 6.2\%$	$51.7\% \pm 21.7\%$	$74.0\% \pm 21.1\%$
r_1,...,r_{11}	$84.6\% \pm 9.1\%$	$89.2\% \pm 6.2\%$	$51.7\% \pm 21.7\%$	$74.0\% \pm 21.1\%$
r_1,...,r_{12}	$84.6\% \pm 9.1\%$	$89.2\% \pm 6.2\%$	$51.7\% \pm 21.7\%$	$74.0\% \pm 21.1\%$
r_1,...,r_{13}	$84.6\% \pm 9.1\%$	$89.2\% \pm 6.2\%$	$51.7\% \pm 21.7\%$	$74.0\% \pm 21.1\%$
r_1,...,r_{14}	$84.6\% \pm 9.1\%$	$89.2\% \pm 6.2\%$	$51.7\% \pm 21.7\%$	$74.0\% \pm 21.1\%$
Classifier: SVM	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
r_1	$54.0\% \pm 25.4\%$	$60.0\% \pm 20.0\%$	$72.5\% \pm 15.6\%$	$75.0\% \pm 13.3\%$
r_1,r_2	$66.0\% \pm 9.2\%$	$77.8\% \pm 14.1\%$	$73.8\% \pm 14.2\%$	$72.1\% \pm 10.3\%$
r_1,r_2,r_3	$66.0\% \pm 9.2\%$	$77.8\% \pm 14.1\%$	$65.0\% \pm 17.5\%$	$69.3\% \pm 10.1\%$
r_1,...,r_4	$66.0\% \pm 9.2\%$	$77.8\% \pm 14.1\%$	$63.8\% \pm 15.3\%$	$70.7\% \pm 14.8\%$
r_1,...,r_5	$66.0\% \pm 9.2\%$	$77.8\% \pm 14.1\%$	$61.3\% \pm 15.3\%$	$69.3\% \pm 13.9\%$
r_1,...,r_6	$68.0\% \pm 13.3\%$	$77.8\% \pm 14.1\%$	$61.3\% \pm 15.3\%$	$68.6\% \pm 14.0\%$
r_1,...,r_7	$68.0\% \pm 13.3\%$	$77.8\% \pm 14.1\%$	$60.0\% \pm 14.6\%$	$70.7\% \pm 13.3\%$
r_1,...,r_8	$56.0\% \pm 15.0\%$	$77.8\% \pm 14.1\%$	$52.5\% \pm 10.9\%$	$71.4\% \pm 10.6\%$
r_1,...,r_9	$58.0\% \pm 16.6\%$	$77.8\% \pm 14.1\%$	$52.5\% \pm 10.9\%$	$71.4\% \pm 10.6\%$
r_1,...,r_{10}	$60.0\% \pm 17.9\%$	$76.7\% \pm 14.4\%$	$52.5\% \pm 10.9\%$	$71.4\% \pm 10.6\%$
r_1,...,r_{11}	$60.0\% \pm 17.9\%$	$76.7\% \pm 14.4\%$	$52.5\% \pm 10.9\%$	$71.4\% \pm 10.6\%$
r_1,...,r_{12}	$60.0\% \pm 17.9\%$	$76.7\% \pm 14.4\%$	$52.5\% \pm 10.9\%$	$71.4\% \pm 10.6\%$
r_1,...,r_{13}	$60.0\% \pm 17.9\%$	$77.8\% \pm 14.1\%$	$52.5\% \pm 10.9\%$	$71.4\% \pm 10.6\%$
r_1,...,r_{14}	$60.0\% \pm 17.9\%$	$77.8\% \pm 14.1\%$	$52.5\% \pm 10.9\%$	$71.4\% \pm 10.6\%$

Table B.5: Results obtained with the EEMD feature extraction method for 2, 2.5, 3.5, and 4.5 inch stickout cases.

Classifier: SVM	5.08 cm (2 inch)		6.35 cm (2.5 inch)	
Features	Test Set	Training Set	Test Set	Training Set
r_1	$74.4\% \pm 0.9\%$	$74.4\% \pm 0.5\%$	$78.3\% \pm 1.2\%$	$78.6\% \pm 0.5\%$
r_1,r_2	$84.2\% \pm 0.8\%$	$84.2\% \pm 0.4\%$	$78.3\% \pm 1.2\%$	$78.6\% \pm 0.5\%$
r_1,r_2,r_3	$84.2\% \pm 0.8\%$	$84.2\% \pm 0.4\%$	$78.3\% \pm 1.4\%$	$78.6\% \pm 0.5\%$
r_1,. . .,r_4	$84.2\% \pm 0.8\%$	$84.2\% \pm 0.4\%$	$78.5\% \pm 1.4\%$	$78.7\% \pm 0.4\%$
r_1,. . .,r_5	$84.2\% \pm 0.8\%$	$84.2\% \pm 0.4\%$	$78.5\% \pm 1.4\%$	$78.7\% \pm 0.4\%$
r_1,. . .,r_6	$84.2\% \pm 0.8\%$	$84.2\% \pm 0.4\%$	$78.5\% \pm 1.2\%$	$78.7\% \pm 0.5\%$
r_1,. . .,r_7	$84.2\% \pm 0.8\%$	$84.2\% \pm 0.4\%$	$78.6\% \pm 1.2\%$	$78.8\% \pm 0.5\%$
Classifier: SVM	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
r_1	$90.7\% \pm 1.4\%$	$91.1\% \pm 0.6\%$	$77.1\% \pm 1.2\%$	$77.1\% \pm 0.5\%$
r_1,r_2	$90.7\% \pm 1.4\%$	$91.1\% \pm 0.7\%$	$77.3\% \pm 1.1\%$	$77.4\% \pm 0.6\%$
r_1,r_2,r_3	$90.6\% \pm 1.4\%$	$91.0\% \pm 0.6\%$	$77.4\% \pm 1.0\%$	$77.5\% \pm 0.6\%$
r_1,. . .,r_4	$90.6\% \pm 1.4\%$	$91.0\% \pm 0.6\%$	$78.9\% \pm 1.1\%$	$78.9\% \pm 0.7\%$
r_1,. . .,r_5	$90.5\% \pm 1.4\%$	$90.9\% \pm 0.6\%$	$78.9\% \pm 1.2\%$	$79.0\% \pm 0.7\%$
r_1,. . .,r_6	$90.5\% \pm 1.4\%$	$91.0\% \pm 0.6\%$	$78.9\% \pm 1.2\%$	$79.0\% \pm 0.7\%$
r_1,. . .,r_7	$90.5\% \pm 1.4\%$	$90.8\% \pm 0.6\%$	$79.1\% \pm 1.2\%$	$79.0\% \pm 0.6\%$

Table B.6: Two class classification (chatter and stable) results obtained with the DTW similarity measure method for 2, 2.5, 3.5, and 4.5 inch overhang cases.

	5.08 cm (2 inch)		6.35 cm (2.5 inch)	
K-NN	Test Set	Training Set	Test Set	Training Set
1-NN	$94.52\% \pm 1.70\%$	$93.92\% \pm 0.88\%$	$55.00\% \pm 10.93\%$	$54.13\% \pm 6.04\%$
2-NN	$79.52\% \pm 2.24\%$	$78.69\% \pm 1.11\%$	$85.94\% \pm 3.76\%$	$86.51\% \pm 1.91\%$
3-NN	$78.98\% \pm 2.69\%$	$78.96\% \pm 1.34\%$	$84.69\% \pm 6.47\%$	$87.14\% \pm 3.29\%$
4-NN	$79.09\% \pm 1.88\%$	$78.91\% \pm 0.93\%$	$85.63\% \pm 5.96\%$	$86.67\% \pm 3.03\%$
5-NN	$79.46\% \pm 4.11\%$	$78.72\% \pm 2.04\%$	$86.88\% \pm 3.90\%$	$86.03\% \pm 1.98\%$
	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
1-NN	$89.52\% \pm 2.86\%$	$89.75\% \pm 2.36\%$	$67.78\% \pm 6.48\%$	$68.89\% \pm 4.37\%$
2-NN	$92.86\% \pm 4.39\%$	$88.75\% \pm 2.30\%$	$70.93\% \pm 5.11\%$	$70.09\% \pm 2.55\%$
3-NN	$87.62\% \pm 6.10\%$	$91.50\% \pm 3.20\%$	$65.00\% \pm 6.28\%$	$73.06\% \pm 3.14\%$
4-NN	$90.95\% \pm 4.49\%$	$89.75\% \pm 2.36\%$	$70.56\% \pm 2.55\%$	$70.28\% \pm 1.27\%$
5-NN	$89.52\% \pm 4.15\%$	$90.50\% \pm 2.18\%$	$70.93\% \pm 5.68\%$	$70.09\% \pm 2.84\%$

Table B.7: Two class classification (chatter including intermediate chatter cases as chatter and stable) results obtained with the DTW similarity measure method for 2, 2.5, 3.5, and 4.5 inch overhang cases.

	5.08 cm (2 inch)		6.35 cm (2.5 inch)	
K-NN	Test Set	Training Set	Test Set	Training Set
1-NN	98.04% ± 0.83%	97.98% ± 0.39%	68.84% ± 4.79%	66.32% ± 3.41%
2-NN	98.34% ± 1.08%	100.00% ± 0.00%	59.30% ± 6.76%	66.90% ± 3.28%
3-NN	97.27% ± 0.69%	98.38% ± 0.27%	72.33% ± 5.14%	72.99% ± 3.82%
4-NN	97.09% ± 1.13%	98.64% ± 0.27%	60.70% ± 5.74%	64.14% ± 3.87%
5-NN	97.42% ± 0.97%	97.37% ± 0.60%	67.67% ± 6.28%	70.11% ± 5.04%
	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
1-NN	91.96% ± 5.37%	91.25% ± 2.81%	75.34% ± 4.84%	75.30% ± 2.63%
2-NN	91.52% ± 6.22%	100.00% ± 0.00%	74.49% ± 3.88%	98.16% ± 0.95%
3-NN	93.04% ± 4.84%	92.27% ± 1.96%	74.92% ± 4.69%	77.39% ± 2.17%
4-NN	92.17% ± 4.68%	93.18% ± 1.76%	74.75% ± 4.61%	78.16% ± 2.07%
5-NN	93.04% ± 4.22%	91.14% ± 2.26%	75.68% ± 4.30%	75.94% ± 2.05%

Table B.8: Three-class classification is applied with the DTW approach.

	5.08 cm (2 inch)		6.35 cm (2.5 inch)	
K-NN	Test Set	Training Set	Test Set	Training Set
1-NN	97.65% ± 1.12%	97.07% ± 0.47%	60.47% ± 6.90%	62.47% ± 3.81%
2-NN	97.19% ± 0.89%	97.58% ± 0.54%	65.81% ± 6.24%	97.76% ± 1.23%
3-NN	97.14% ± 1.12%	97.85% ± 0.26%	59.30% ± 7.22%	70.59% ± 4.04%
4-NN	95.77% ± 0.89%	97.45% ± 0.56%	71.40% ± 6.98%	78.47% ± 7.21%
5-NN	95.20% ± 1.29%	97.35% ± 0.38%	61.16% ± 8.45%	66.00% ± 3.95%
	8.89 cm (3.5 inch)		11.43 cm (4.5 inch)	
1-NN	93.64% ± 3.64%	95.00% ± 1.9%	69.66% ± 4.64%	69.57% ± 3.86%
2-NN	93.64% ± 2.23%	94.55% ± 1.51%	75.93% ± 4.41%	83.59% ± 2.38%
3-NN	95.45% ± 5.38%	94.55% ± 1.51%	72.20% ± 5.26%	77.69% ± 1.60%
4-NN	89.55% ± 7.62%	93.18% ± 2.27%	73.90% ± 4.62%	76.58% ± 2.99%
5-NN	89.55% ± 7.06%	93.64% ± 1.98%	71.36% ± 4.70%	73.42% ± 3.51%

Table B.9: Persistence Landscape Results with Support Vector Machine (SVM) classifier. The landscape number column represents which landscapes were used to extract features.

Landscape Number	2 inch		2.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	$96.8\% \pm 1.5\%$	$96.7\% \pm 0.5\%$	$87.0\% \pm 4.8\%$	$86.9\% \pm 2.4\%$
2	$86.6\% \pm 1.7\%$	$91.5\% \pm 0.8\%$	$87.0\% \pm 3.8\%$	$86.0\% \pm 2.0\%$
3	$89.1\% \pm 2.7\%$	$93.4\% \pm 0.5\%$	$84.7\% \pm 4.7\%$	$86.6\% \pm 1.9\%$
4	$90.5\% \pm 1.9\%$	$92.9\% \pm 0.4\%$	$85.8\% \pm 2.6\%$	$86.1\% \pm 1.4\%$
5	$90.5\% \pm 2.3\%$	$92.2\% \pm 0.7\%$	$88.4\% \pm 1.5\%$	$85.2\% \pm 0.6\%$
Landscape Number	3.5 inch		4.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	$92.2\% \pm 5.4\%$	$93.2\% \pm 1.8\%$	$66.8\% \pm 5.6\%$	$78.5\% \pm 3.0\%$
2	$78.3\% \pm 4.3\%$	$85.2\% \pm 2.1\%$	$65.3\% \pm 4.3\%$	$69.3\% \pm 2.3\%$
3	$82.6\% \pm 4.3\%$	$83.6\% \pm 2.0\%$	$68.6\% \pm 4.8\%$	$72.6\% \pm 1.5\%$
4	$84.3\% \pm 6.8\%$	$84.3\% \pm 2.8\%$	$67.6\% \pm 5.2\%$	$71.8\% \pm 2.6\%$
5	$85.7\% \pm 4.8\%$	$84.3\% \pm 3.1\%$	$66.8\% \pm 5.5\%$	$73.0\% \pm 3.4\%$

Table B.10: Persistence Landscape Results with Logistic Regression (LR) classifier. The landscape number column represents which landscapes were used to extract features.

Landscape Number	2 inch		2.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	$95.7\% \pm 1.6\%$	$98.3\% \pm 0.4\%$	$82.6\% \pm 4.2\%$	$91.3\% \pm 2.5\%$
2	$85.5\% \pm 1.9\%$	$93.7\% \pm 1.0\%$	$86.3\% \pm 4.7\%$	$91.5\% \pm 2.1\%$
3	$88.5\% \pm 1.1\%$	$93.8\% \pm 0.6\%$	$84.7\% \pm 4.3\%$	$92.0\% \pm 2.6\%$
4	$89.4\% \pm 1.6\%$	$94.8\% \pm 0.8\%$	$86.5\% \pm 5.0\%$	$90.8\% \pm 2.4\%$
5	$88.3\% \pm 1.7\%$	$94.7\% \pm 0.6\%$	$86.0\% \pm 3.4\%$	$91.1\% \pm 2.7\%$
Landscape Number	3.5 inch		4.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	$91.3\% \pm 3.4\%$	$96.4\% \pm 1.8\%$	$63.1\% \pm 3.5\%$	$82.0\% \pm 3.4\%$
2	$82.2\% \pm 6.6\%$	$90.7\% \pm 2.6\%$	$63.1\% \pm 3.5\%$	$82.0\% \pm 3.4\%$
3	$87.8\% \pm 6.4\%$	$91.4\% \pm 2.8\%$	$64.2\% \pm 4.5\%$	$84.1\% \pm 1.1\%$
4	$90.4\% \pm 4.3\%$	$91.8\% \pm 1.5\%$	$63.1\% \pm 5.5\%$	$82.9\% \pm 2.4\%$
5	$85.7\% \pm 4.8\%$	$93.4\% \pm 1.9\%$	$65.9\% \pm 4.6\%$	$83.0\% \pm 2.2\%$

Table B.11: Persistence Landscape Results with Random Forest (RF) classifier. The landscape number column represents which landscapes were used to extract features.

Landscape Number	2 inch		2.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	$96.1\% \pm 1.7\%$	$100.0\% \pm 0.0\%$	$86.7\% \pm 5.6\%$	$100.0\% \pm 0.0\%$
2	$87.7\% \pm 2.2\%$	$100.0\% \pm 0.0\%$	$88.6\% \pm 4.0\%$	$100.0\% \pm 0.0\%$
3	$88.0\% \pm 1.9\%$	$100.0\% \pm 0.0\%$	$87.4\% \pm 2.4\%$	$100.0\% \pm 0.0\%$
4	$88.8\% \pm 1.2\%$	$100.0\% \pm 0.0\%$	$86.7\% \pm 5.8\%$	$100.0\% \pm 0.0\%$
5	$89.6\% \pm 2.2\%$	$100.0\% \pm 0.0\%$	$86.7\% \pm 5.8\%$	$100.0\% \pm 0.0\%$
Landscape Number	3.5 inch		4.5 inch	
	Test Set	Training Set	Test Set	Training Set
1	$91.3\% \pm 3.9\%$	$100.0\% \pm 0.0\%$	$65.1\% \pm 7.7\%$	$100.0\% \pm 0.0\%$
2	$80.9\% \pm 7.1\%$	$100.0\% \pm 0.0\%$	$66.9\% \pm 2.9\%$	$100.0\% \pm 0.0\%$
3	$83.0\% \pm 6.3\%$	$100.0\% \pm 0.0\%$	$66.8\% \pm 4.9\%$	$100.0\% \pm 0.0\%$
4	$80.4\% \pm 5.6\%$	$100.0\% \pm 0.0\%$	$63.6\% \pm 4.2\%$	$100.0\% \pm 0.0\%$
5	$85.2\% \pm 5.9\%$	$100.0\% \pm 0.0\%$	$64.7\% \pm 7.1\%$	$100.0\% \pm 0.0\%$

Table B.12: Persistence Images results with pixel size = 0.1.

Classifier	2 inch		2.5 inch	
	Test Set	Training Set	Test Set	Training Set
SVM	$82.5\% \pm 1.4\%$	$82.7\% \pm 0.6\%$	$79.3\% \pm 4.2\%$	$81.4\% \pm 2.9\%$
LR	$80.2\% \pm 2.0\%$	$82.4\% \pm 0.9\%$	$77.0\% \pm 7.0\%$	$82.4\% \pm 4.5\%$
RF	$96.4\% \pm 1.1\%$	$100.0\% \pm 0.0\%$	$85.8\% \pm 5.3\%$	$100.0\% \pm 0.0\%$
GB	$95.9\% \pm 1.5\%$	$100.0\% \pm 0.0\%$	$84.4\% \pm 4.4\%$	$100.0\% \pm 0.0\%$
Classifier	3.5 inch		4.5 inch	
	Test Set	Training Set	Test Set	Training Set
SVM	$82.6\% \pm 5.5\%$	$83.2\% \pm 2.5\%$	$66.9\% \pm 5.8\%$	$63.7\% \pm 2.9\%$
LR	$82.6\% \pm 5.1\%$	$85.7\% \pm 2.0\%$	$62.7\% \pm 5.9\%$	$65.5\% \pm 3.3\%$
RF	$93.0\% \pm 2.9\%$	$100.0\% \pm 0.0\%$	$72.5\% \pm 6.4\%$	$100.0\% \pm 0.0\%$
GB	$91.3\% \pm 4.3\%$	$100.0\% \pm 0.0\%$	$68.5\% \pm 2.8\%$	$100.0\% \pm 0.0\%$

Table B.13: Persistence Images results with pixel size = 0.05.

	2 inch		2.5 inch	
Classifier	Test Set	Training Set	Test Set	Training Set
SVM	$82.0\% \pm 2.5\%$	$83.4\% \pm 1.2\%$	$81.2\% \pm 5.1\%$	$80.2\% \pm 2.8\%$
LR	$80.3\% \pm 2.3\%$	$79.5\% \pm 1.0\%$	$68.6\% \pm 7.3\%$	$75.2\% \pm 3.4\%$
RF	$96.0\% \pm 1.4\%$	$100.0\% \pm 0.0\%$	$85.8\% \pm 4.1\%$	$100.0\% \pm 0.0\%$
GB	$95.5\% \pm 1.6\%$	$100.0\% \pm 0.0\%$	$85.6\% \pm 3.4\%$	$100.0\% \pm 0.0\%$
	3.5 inch		4.5 inch	
Classifier	Test Set	Training Set	Test Set	Training Set
SVM	$85.2\% \pm 4.0\%$	$81.8\% \pm 2.5\%$	$63.4\% \pm 5.2\%$	$65.6\% \pm 2.4\%$
LR	$80.9\% \pm 5.9\%$	$85.9\% \pm 3.0\%$	$63.6\% \pm 4.5\%$	$64.8\% \pm 2.1\%$
RF	$90.9\% \pm 3.0\%$	$100.0\% \pm 0.0\%$	$70.7\% \pm 3.1\%$	$100.0\% \pm 0.0\%$
GB	$90.4\% \pm 4.7\%$	$100.0\% \pm 0.0\%$	$70.0\% \pm 5.4\%$	$100.0\% \pm 0.0\%$

Table B.14: Carlsson Coordinates results.

	2 inch		2.5 inch	
Classifier	Test Set	Training Set	Test Set	Training Set
SVM	$87.8\% \pm 2.0\%$	$87.1\% \pm 1.2\%$	$72.1\% \pm 7.1\%$	$79.7\% \pm 3.9\%$
LR	$93.1\% \pm 1.8\%$	$92.9\% \pm 0.9\%$	$86.3\% \pm 6.2\%$	$100.0\% \pm 0.0\%$
RF	$93.0\% \pm 2.0\%$	$100.0\% \pm 0.0\%$	$69.5\% \pm 4.8\%$	$70.9\% \pm 2.6\%$
GB	$93.6\% \pm 1.7\%$	$100.0\% \pm 0.0\%$	$84.7\% \pm 5.0\%$	$100.0\% \pm 0.0\%$
	3.5 inch		4.5 inch	
Classifier	Test Set	Training Set	Test Set	Training Set
SVM	$95.2\% \pm 4.5\%$	$95.2\% \pm 1.9\%$	$68.1\% \pm 6.5\%$	$68.6\% \pm 2.4\%$
LR	$90.9\% \pm 9.2\%$	$93.6\% \pm 1.4\%$	$70.8\% \pm 4.0\%$	$72.6\% \pm 3.0\%$
RF	$95.7\% \pm 4.3\%$	$100.0\% \pm 0.0\%$	$72.2\% \pm 4.4\%$	$100.0\% \pm 0.0\%$
GB	$92.2\% \pm 4.3\%$	$100.0\% \pm 0.0\%$	$71.9\% \pm 3.3\%$	$100.0\% \pm 0.0\%$

Table B.15: Kernel Method results with LibSVM package.

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

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

Table B.16: Path signature method results obtained with SVM classifier. Landscape numbers represent which landscapes were used to compute signatures.

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

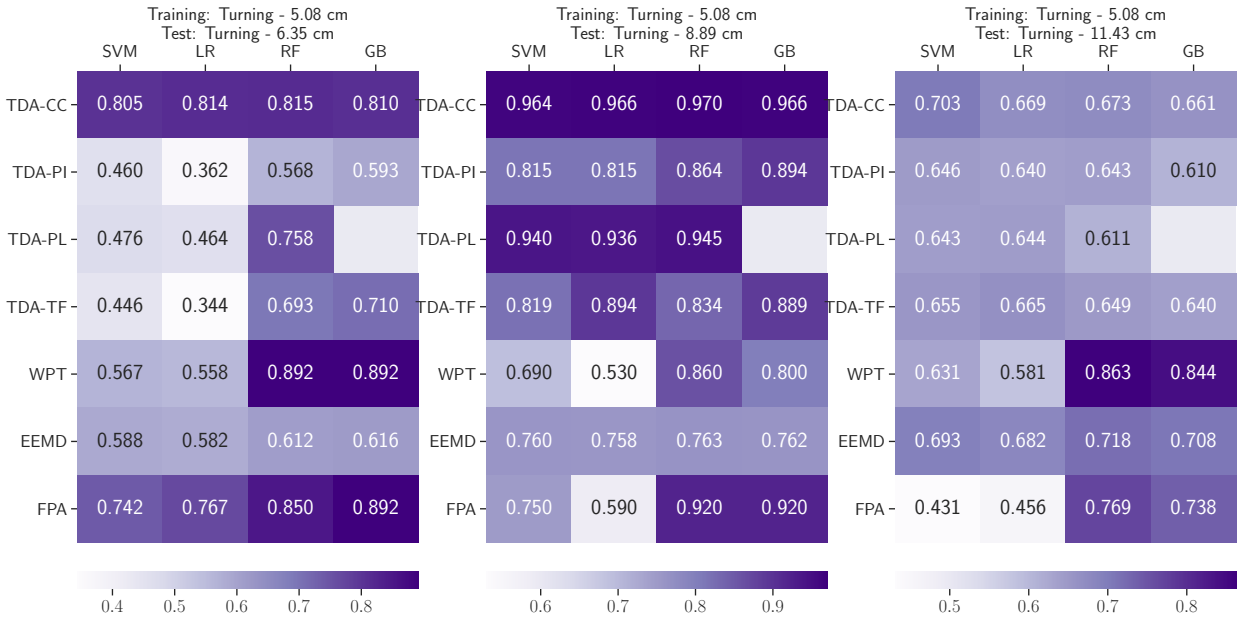


Figure B.3: Classification accuracies obtained from transfer learning applications for turning experiment case with 5.08 cm overhang distance. (left) Training: 5.08 cm Test: 6.35 cm (middle) Training: 5.08 cm Test: 8.89 cm, (right) Training: 5.08 cm Test: 11.43 cm. CC: Carlsson Coordinates, PI: Persistence Images, PL: Persistence Landscapes, TF: Template Functions, WPT: Wavelet Packet Transform, EEMD: Ensemble Empirical Mode Decomposition, FPA: FFT/PSD/ACF, SVM: Support Vector Machine, RF: Random Forest, GB: Gradient Boosting. The results for TDA-PL implementation with Gradient Boosting classifier (GB) are not available due to the large amount of time required in training and testing. Therefore, it represents an empty box in the figure.

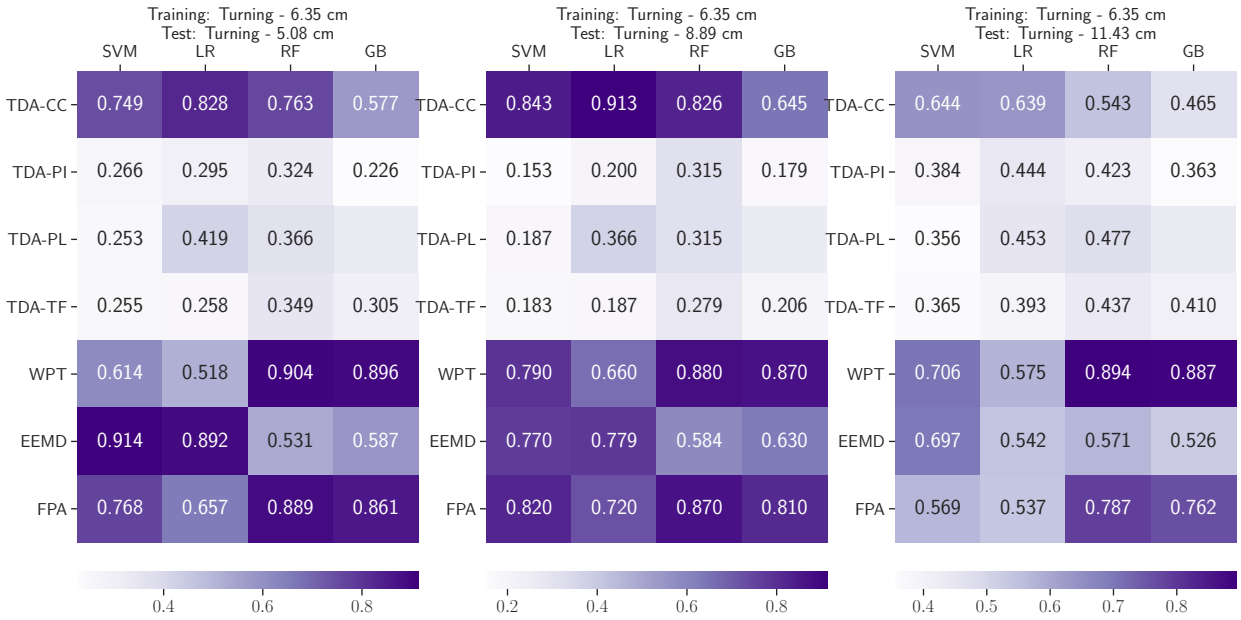


Figure B.4: Classification accuracies obtained from transfer learning applications for turning experiment case with 6.35 cm overhang distance. (left) Training: 6.35 cm Test: 5.08 cm (middle) Training: 6.35 cm Test: 8.89 cm, (right) Training: 6.35 cm Test: 11.43 cm. CC: Carlsson Coordinates, PI: Persistence Images, PL: Persistence Landscapes, TF: Template Functions, WPT: Wavelet Packet Transform, EEMD: Ensemble Empirical Mode Decomposition, FPA: FFT/PSD/ACF, SVM: Support Vector Machine, RF: Random Forest, GB: Gradient Boosting. The results for TDA-PL implementation with Gradient Boosting classifier (GB) are not available due to the large amount of time required in training and testing. Therefore, it represents an empty box in the figure.

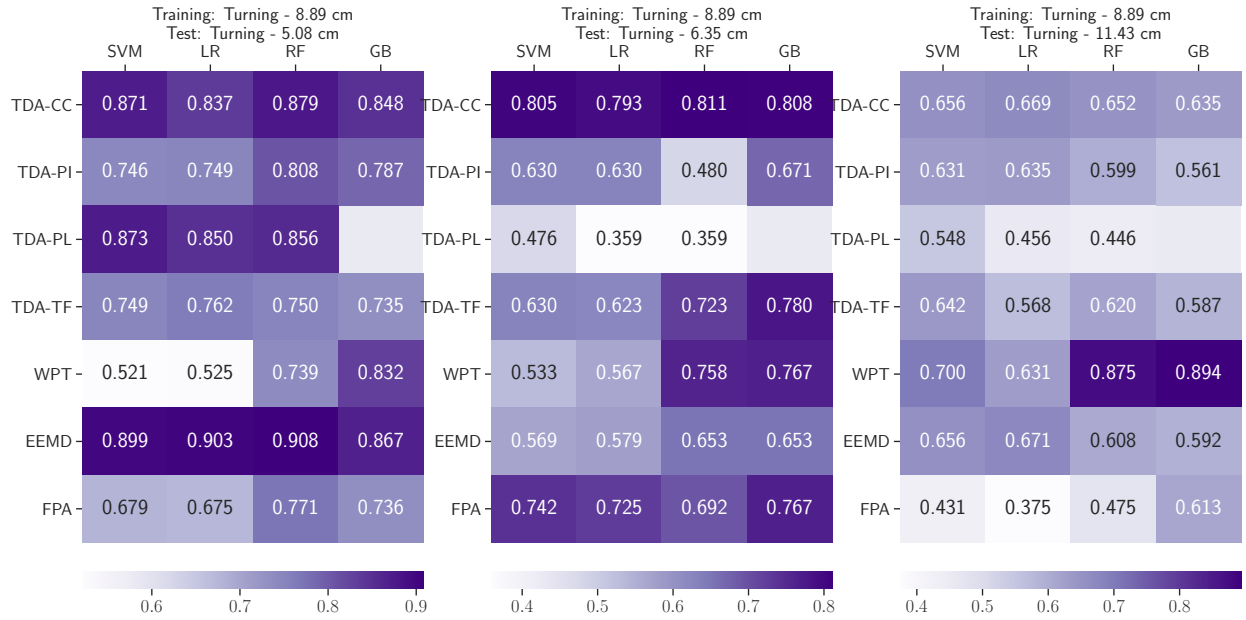


Figure B.5: Classification accuracies obtained from transfer learning applications for turning experiment case with 8.89 cm overhang distance . (left) Training: 8.89 cm Test: 5.08 cm (middle) Training: 8.89 cm Test: 6.35 cm, (right) Training: 8.89 cm Test: 11.43 cm. CC: Carlsson Coordinates, PI: Persistence Images, PL: Persistence Landscapes, TF: Template Functions, WPT: Wavelet Packet Transform, EEMD: Ensemble Empirical Mode Decomposition, FPA: FFT/PSD/ACF, SVM: Support Vector Machine, RF: Random Forest, GB: Gradient Boosting. The results for TDA-PL implementation with Gradient Boosting classifier (GB) are unavailable due to a large amount of time required in training and testing. Therefore, it represents an empty box in the figure.

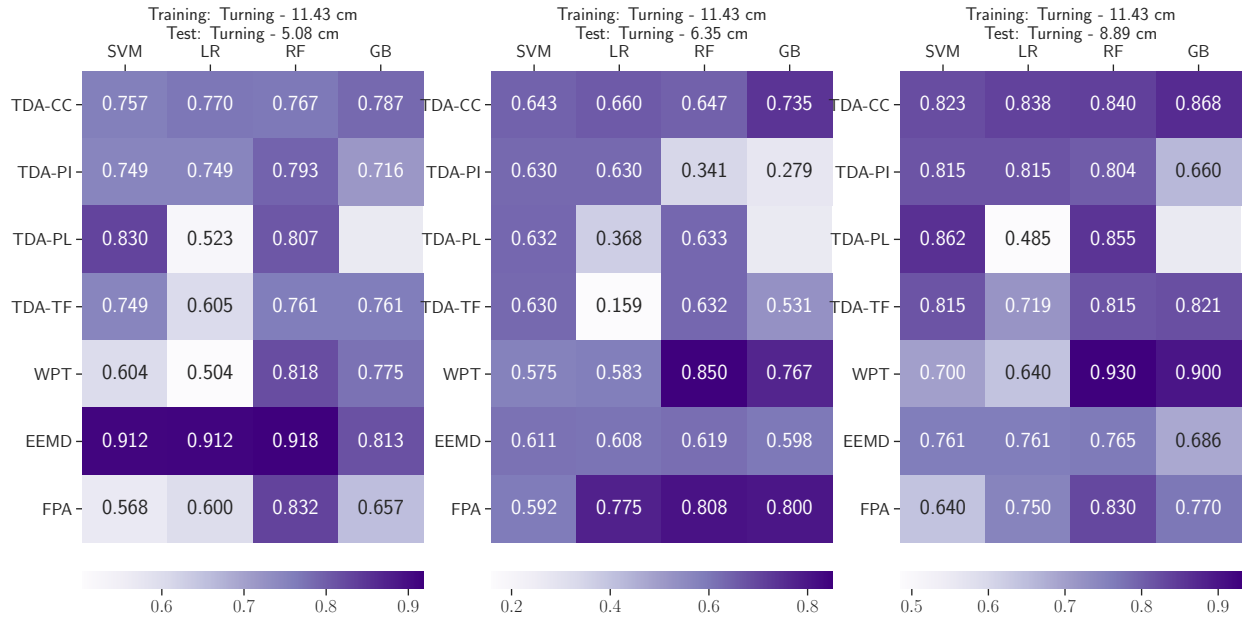


Figure B.6: Classification accuracies obtained from transfer learning applications for turning experiment case with 11.43 cm overhang distance . (left) Training: 11.43 cm Test: 5.08 cm (middle) Training: 11.43 cm Test: 6.35 cm, (right) Training: 11.43 cm Test: 8.89 cm. CC: Carlsson Coordinates, PI: Persistence Images, PL: Persistence Landscapes, TF: Template Functions, WPT: Wavelet Packet Transform, EEMD: Ensemble Empirical Mode Decomposition, FPA: FFT/PSD/ACF, SVM: Support Vector Machine, RF: Random Forest, GB: Gradient Boosting. The results for TDA-PL implementation with Gradient Boosting classifier (GB) are not available due to a large amount of time required in training and testing. Therefore, it represents an empty box in the figure.

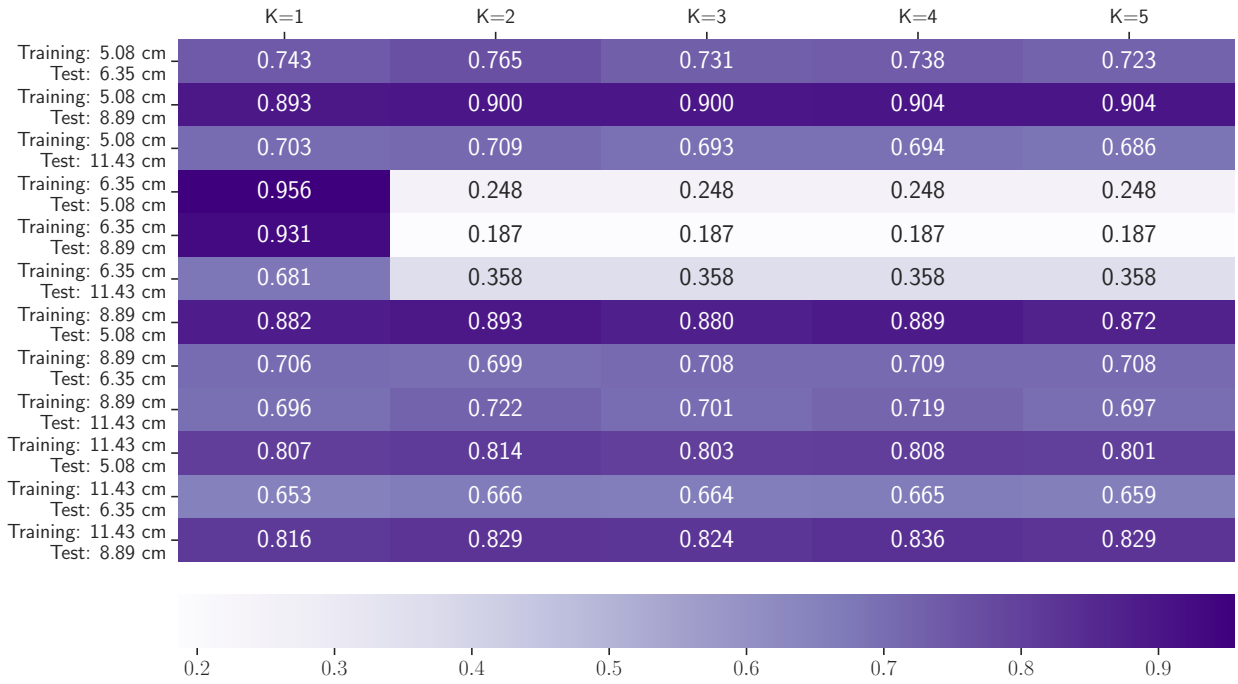


Figure B.7: Classification accuracies obtained from transfer learning applications for turning data set using DTW approach with $K = 1, 2, 3, 4, 5$, where K represents the nearest neighbor number. Overhang distances used as training and testing data sets are shown on y-axis.

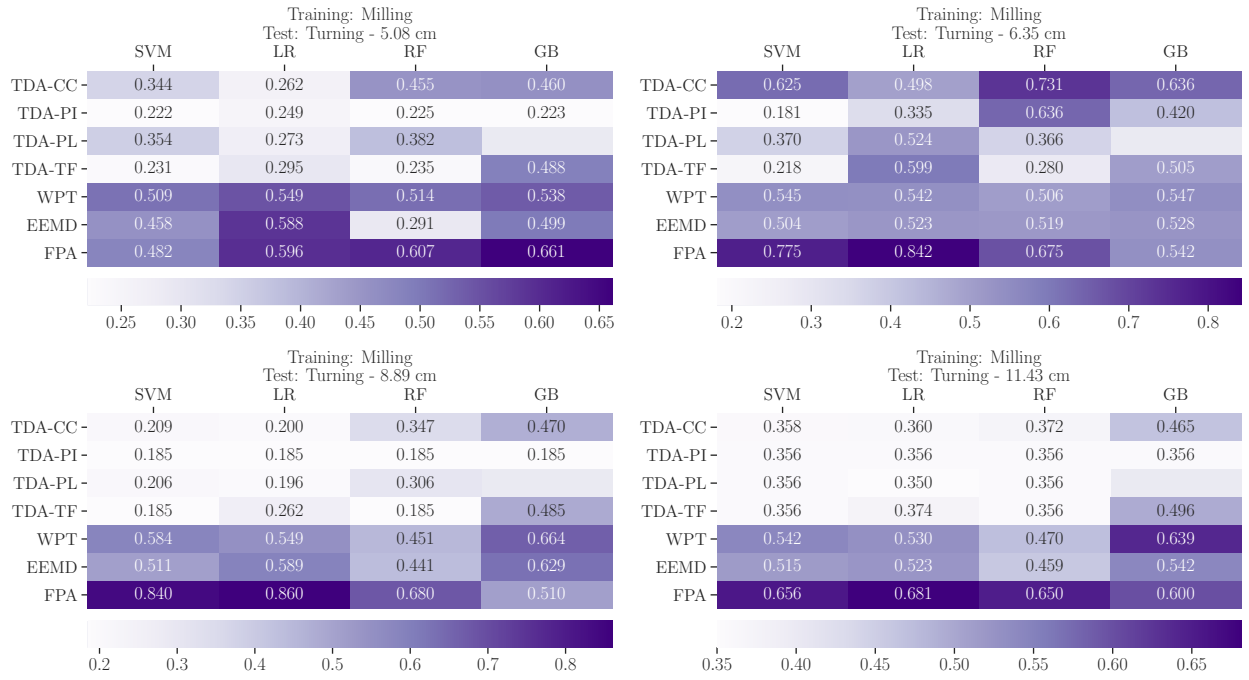


Figure B.8: Classification accuracies obtained from transfer learning applications when the milling data set is used as the training set and the turning data set is used as the test set. CC: Carlsson Coordinates, PI: Persistence Images, PL: Persistence Landscapes, TF: Template Functions, WPT: Wavelet Packet Transform, EEMD: Ensemble Empirical Mode Decomposition, FPA: FFT/PSD/ACF, SVM: Support Vector Machine, RF: Random Forest, GB: Gradient Boosting. The results for TDA-PL implementation with Gradient Boosting classifier (GB) are not available due to the large amount of time required in training and testing. Therefore, it represents an empty box in the figure.

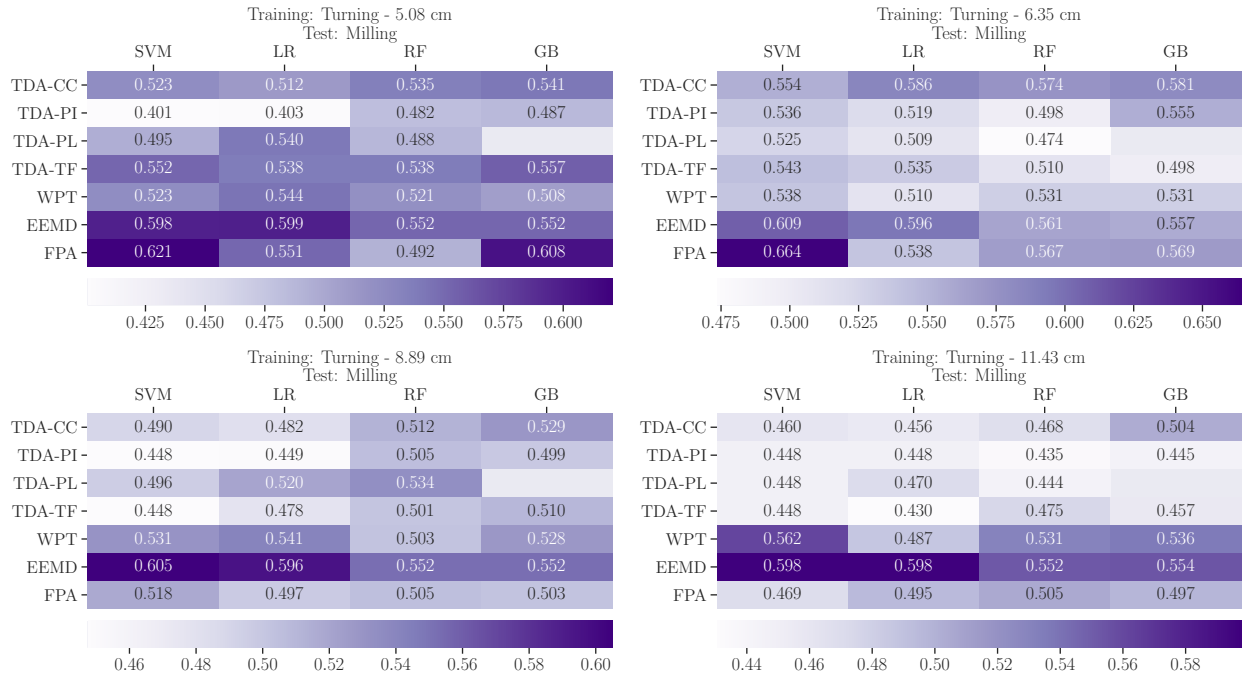


Figure B.9: Classification accuracies obtained from transfer learning applications when the milling data set is used as the test set, and the turning data set is used as the training set. CC: Carlsson Coordinates, PI: Persistence Images, PL: Persistence Landscapes, TF: Template Functions, WPT: Wavelet Packet Transform, EEMD: Ensemble Empirical Mode Decomposition, FPA: FFT/PSD/ACF, SVM: Support Vector Machine, RF: Random Forest, GB: Gradient Boosting. The results for TDA-PL implementation with Gradient Boosting classifier (GB) are not available due to the large amount of time required in training and testing. Therefore, it represents an empty box in the figure.

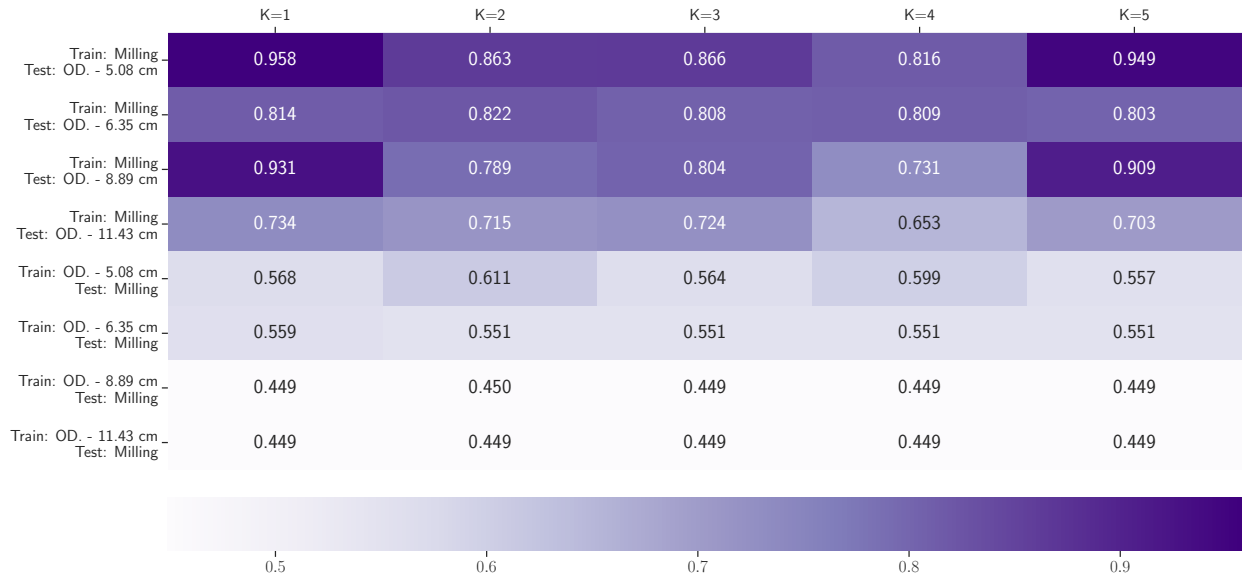


Figure B.10: Classification accuracies obtained from transfer learning applications between turning and milling data sets using the DTW approach with $K = 1, 2, 3, 4, 5$, where K represents the nearest neighbor number. Overhang distances (OD.) used as training or testing data sets are shown on y-axis.

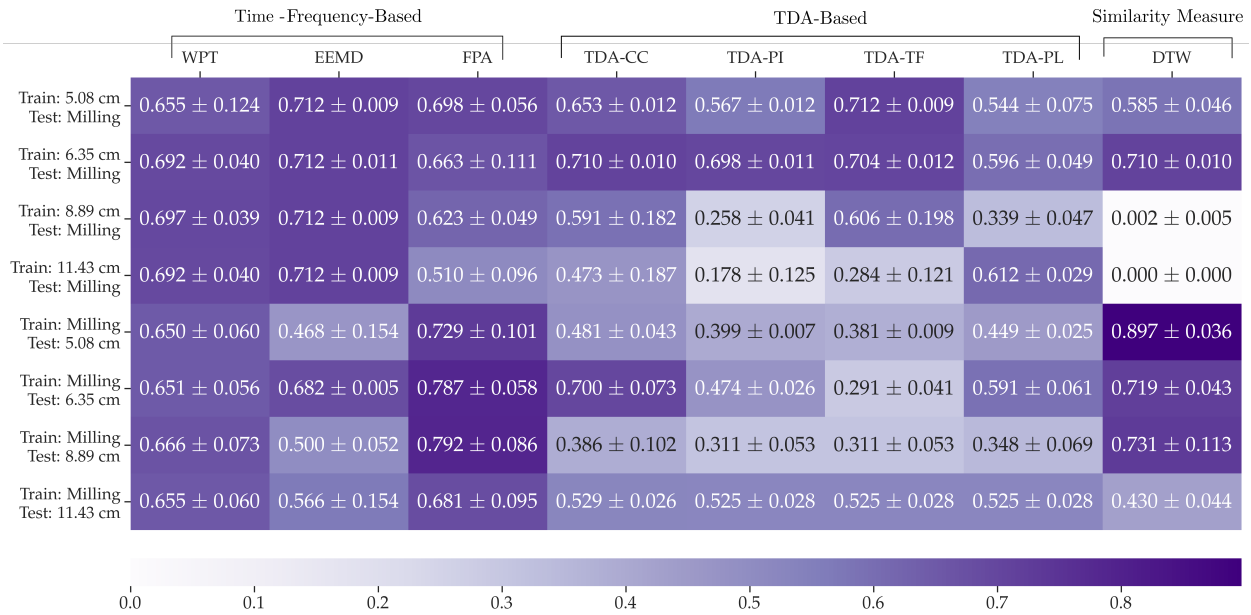


Figure B.11: The highest F1-score out of four different classifiers (or out of selected numbers of nearest neighbor for DTW) for each approach used in transfer learning applications between overhang distance cases of the turning and milling experiments.

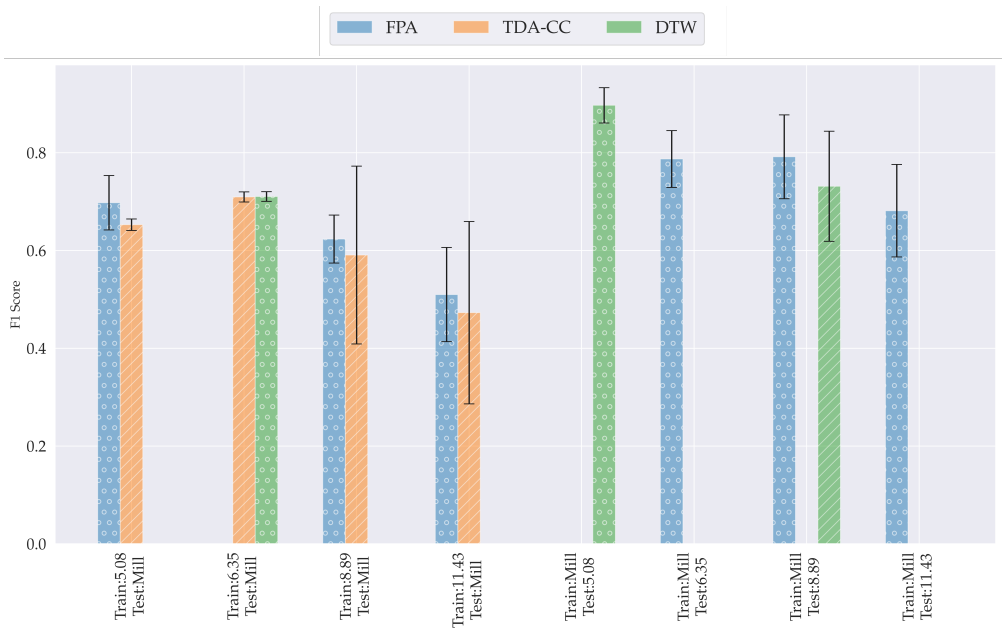


Figure B.12: F1 scores obtained from the selected methods when we train and test between the overhang distance cases of the turning data set and the milling data set. The selected methods that give the highest accuracy are represented with '○' bar hatch, and the ones that are in the error band of the highest accuracy are shown with '/' bar hatch. One approach is selected from each category of the methods, and these are FPA, TDA-CC, and DTW.

APPENDIX C

EXPERIMENTAL PROCEDURE FOR ENGINEERING SURFACE SCANS

Experimental surfaces scans are collected from the standard S-22 Microfinish Comparator. The roughness heights and the machining type of 9 selected surfaces are provided in Table C.1.

Table C.1: Selected surfaces under various machining conditions.

Machining Type	Roughness height (micrometers(microinches))		
M - milling	3.175 (125)	6.35(250)	12.7(500)
P - profiled	3.175 (125)	6.35(250)	12.7(500)
ST - shaped or turned	3.175 (125)	6.35(250)	12.7(500)

For each selected sample, the upper left corner area was selected for consistent scanning. The selected area is 5 mm $\times$ 5 m. The surface texture was measured using Keyence digital microscope (VHX6000) after placing the comparator on a free-angle XYZ motorized observation system (VHX-S650E). The setup for scanning is also provided in Figure C.2a. x500 magnification is selected to obtain enough spatial resolution (0.42 μm), and surface texture is obtained using the zoom lens Keyence VH-Z500R, RZ x500-x5000. The stitching technique was used to scan the designated area, and the procedure was completed by taking 11 $\times$ 11

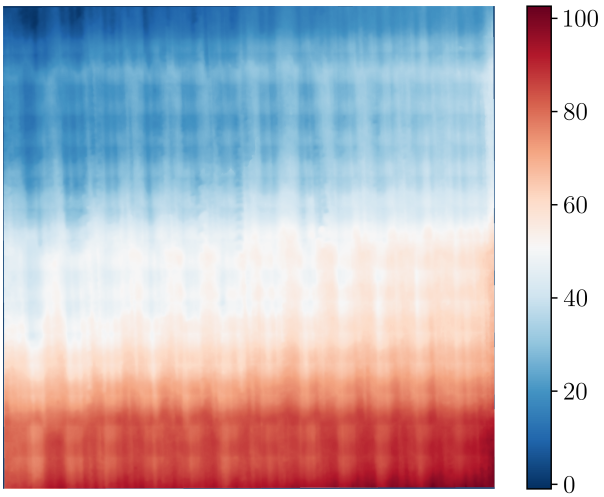
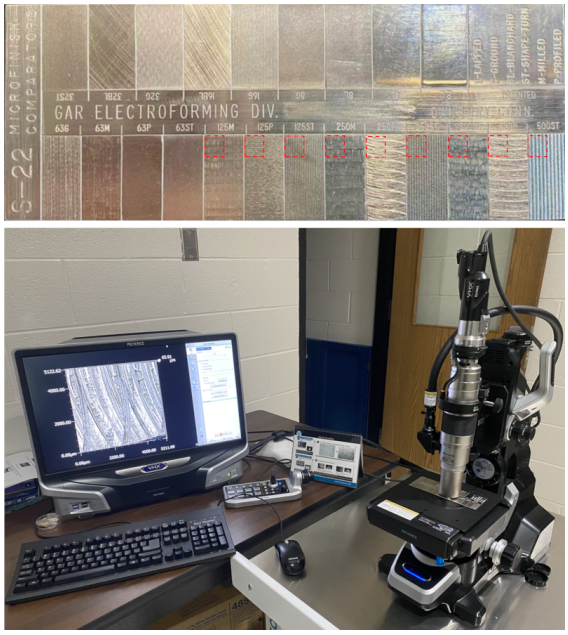
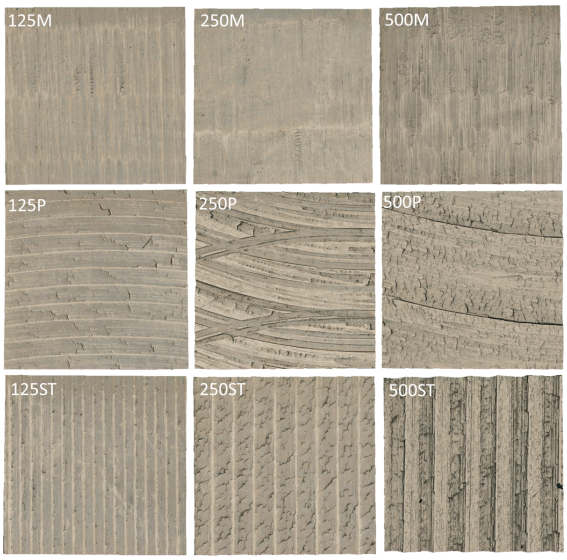


Figure C.1: Scan sample of 125M.



(a) Scanned portions of the sample and the digital microscope.



(b) The scanned surface textures.

Figure C.2: The microscope used for experimental data collection and the sample surfaces.

scans in horizontal and vertical directions. The spatial sampling rate for the scans is around 2.4 samples per μm. The resulting surface scans are shown in Figure C.2b.

APPENDIX D

CLASSIFICATION RESULTS FOR TOOL WEAR ANALYSIS

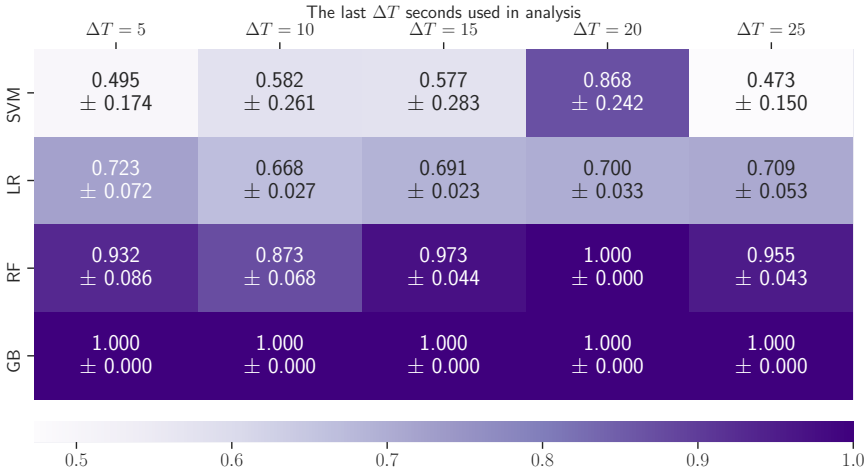


Figure D.1: Training set classification scores obtained from the DWT approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the data was performed with respect to flank wear measurements.

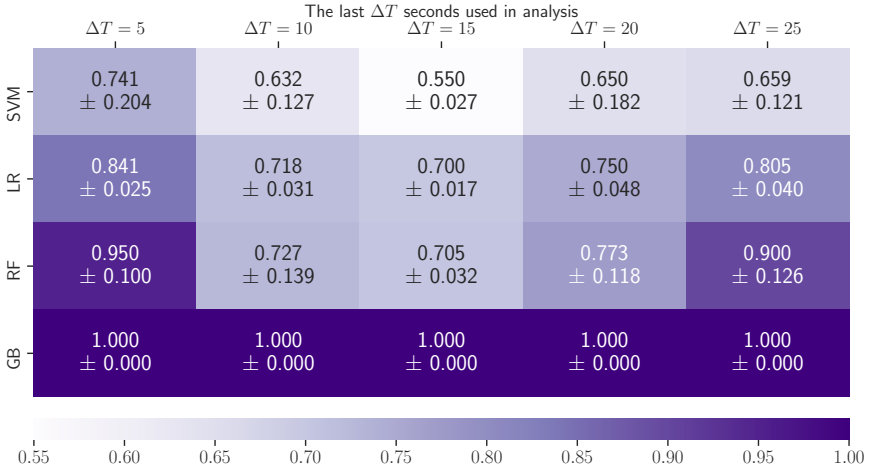


Figure D.2: Training set classification scores obtained from the DWT approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the data was performed with respect to crater wear measurements.

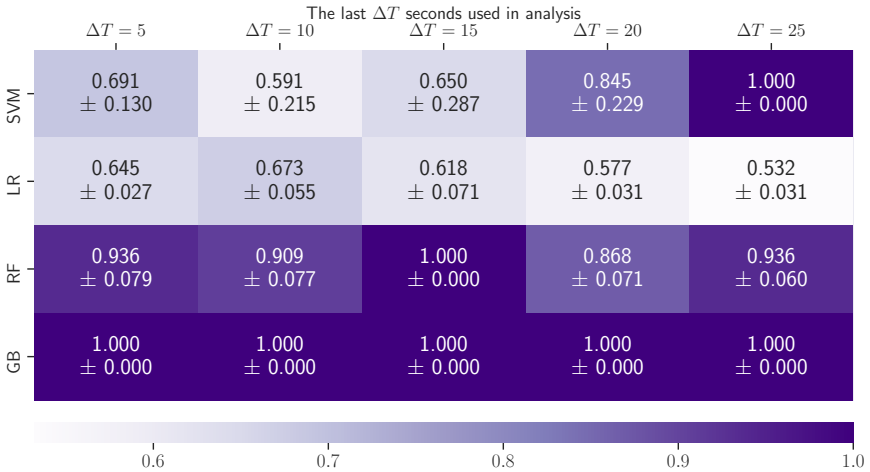


Figure D.3: Training set classification scores obtained from the EEMD approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling the of data was performed with respect to flank wear measurements.

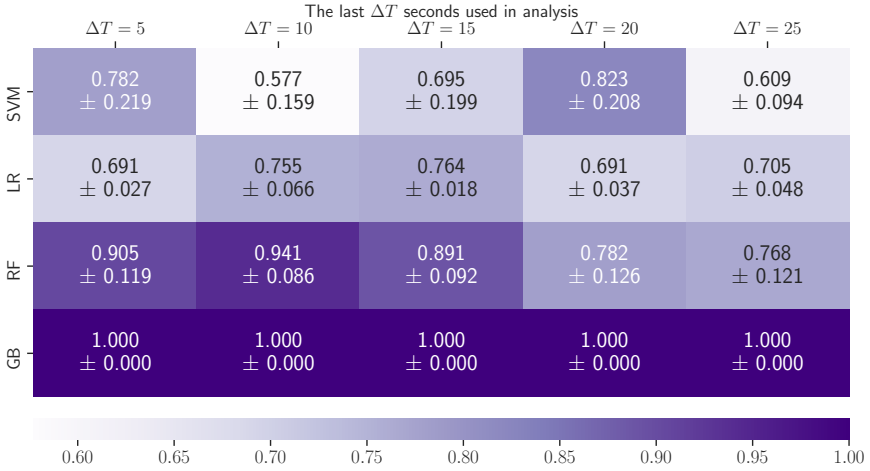


Figure D.4: Training set classification scores obtained from the EEMD approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the data was performed with respect to crater wear measurements.

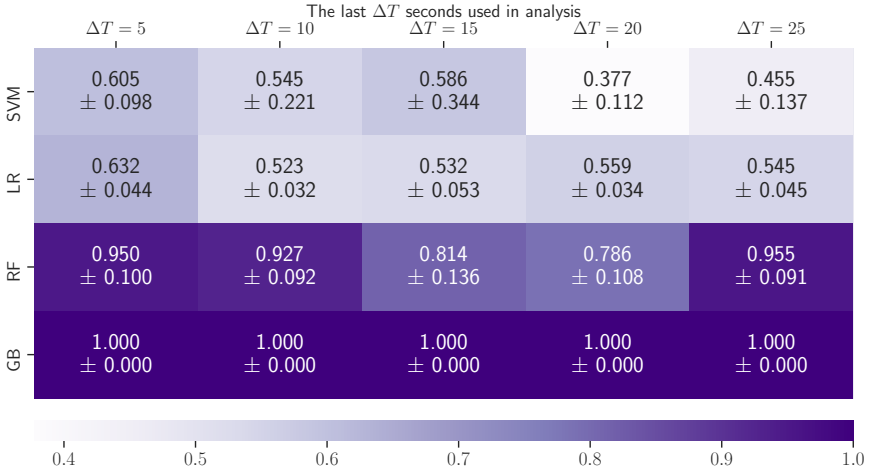


Figure D.5: Training set classification scores obtained from the Carlsson Coordinates approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the data was performed with respect to flank wear measurements.

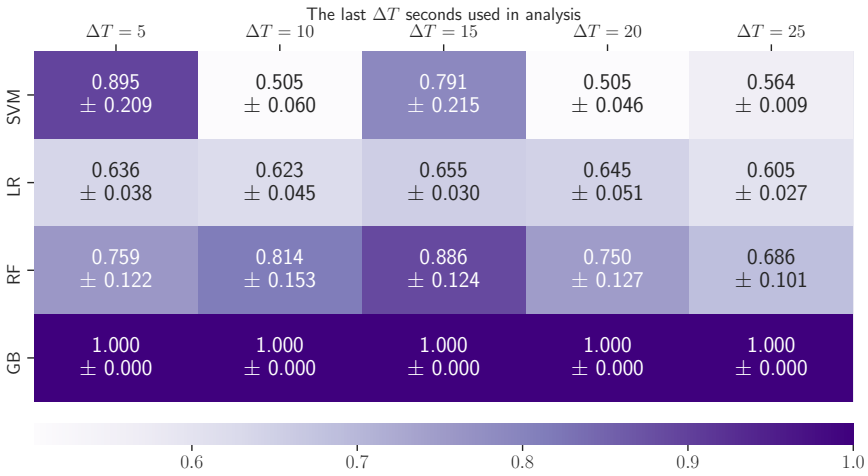


Figure D.6: Training set classification scores obtained from the Carlsson Coordinates approach. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the data was performed with respect to crater wear measurements.

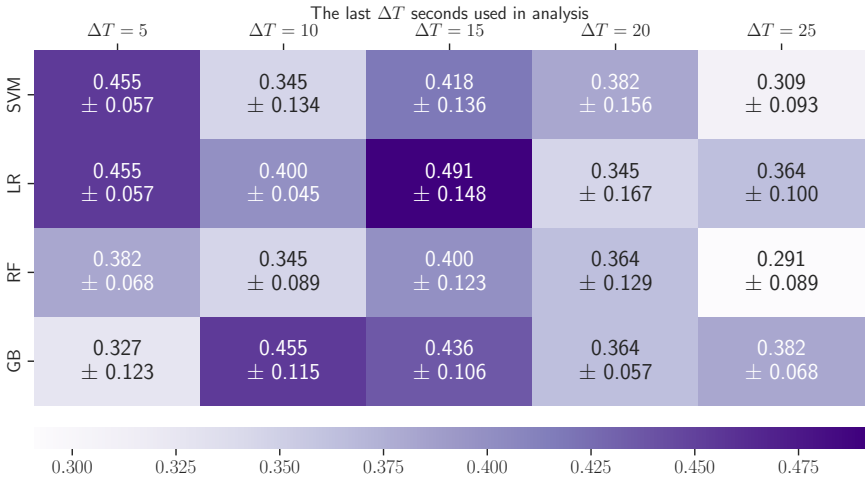


Figure D.7: The test set classification scores obtained with the persistence images. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on flank wear measurements.

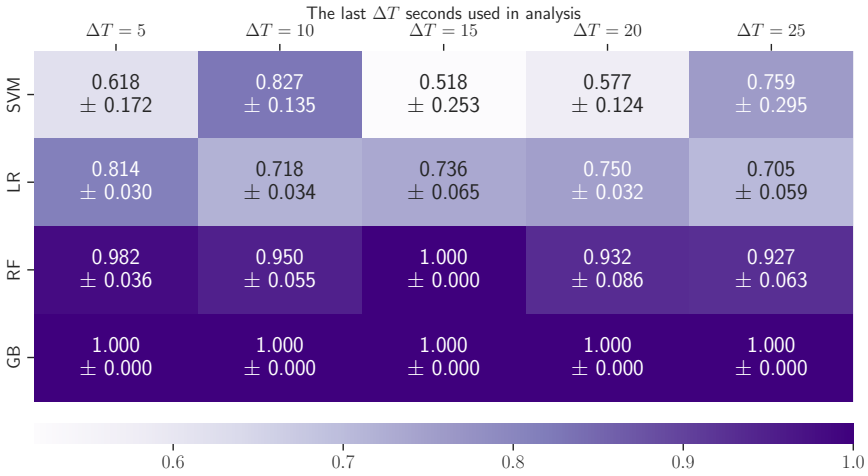


Figure D.8: Training set classification scores obtained from the persistence images. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the data was performed with respect to flank wear measurements.

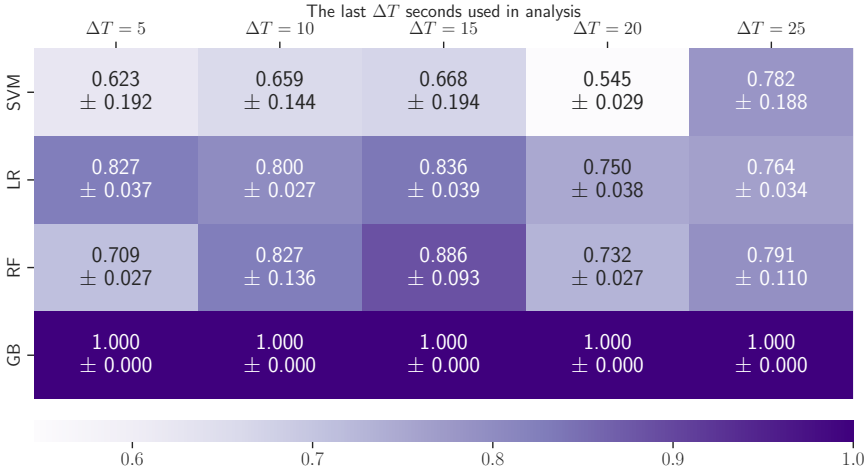


Figure D.9: Training set classification scores obtained from the persistence images. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the data was performed with respect to crater wear measurements.

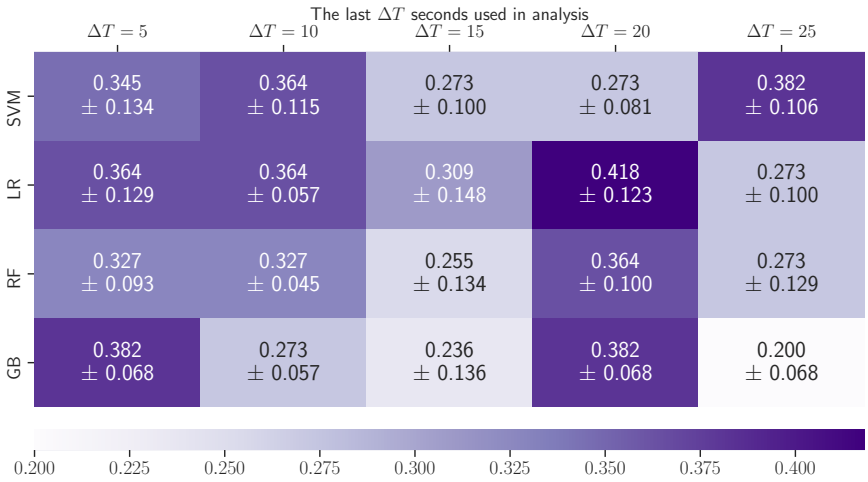


Figure D.10: The test set classification scores obtained with the template functions. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the time series used in the analysis is done based on flank wear measurements.

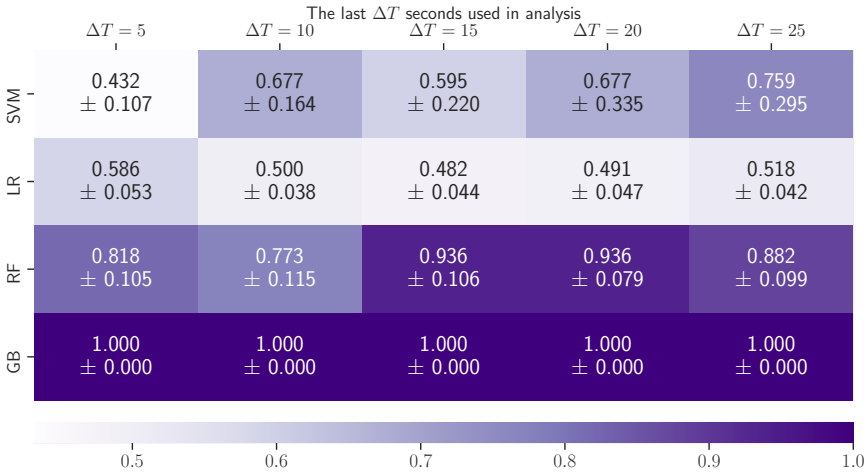


Figure D.11: Training set classification scores obtained from the template functions. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the data was performed with respect to flank wear measurements.

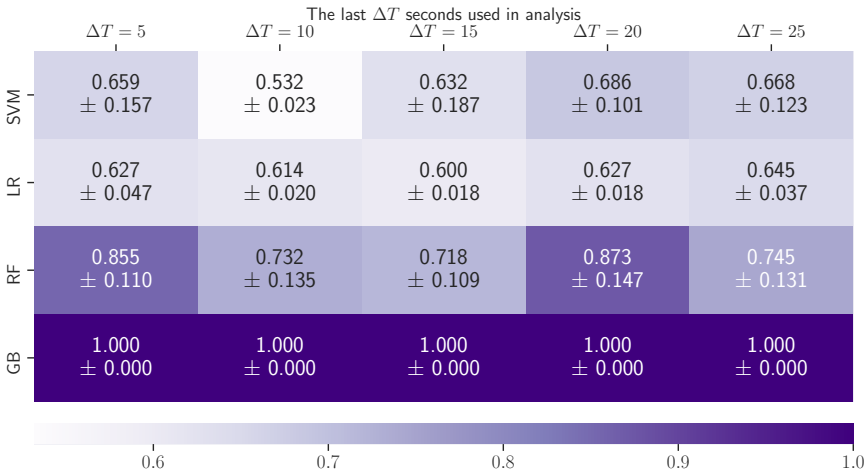


Figure D.12: Training set classification scores obtained from the template functions. Results are provided for four different classification algorithms: SVM: Support Vector Machine, LR: Logistic Regression, RF: Random Forest, GB: Gradient Boosting. The labeling of the data was performed with respect to crater wear measurements.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] H. Doraiswamy, N. Shivashankar, V. Natarajan, and Y. Wang, "Topological saliency," *Computers & Graphics*, vol. 37, no. 7, pp. 787–799, 2013.

[2] F. A. Khasawneh and B. P. Mann, "A spectral element approach for the stability of delay systems," *International Journal for Numerical Methods in Engineering*, vol. 87, no. 6, pp. 566–592, 2011.

[3] M. C. Yesilli, S. Tymochko, F. A. Khasawneh, and E. Munch, "Chatter diagnosis in milling using supervised learning and topological features vector," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1211–1218, IEEE, 2019.

[4] 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, 2014.

[5] 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, no. 1-4, pp. 775–784, 2017.

[6] 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, 2018.

[7] X. Li, G. Ouyang, and Z. Liang, "Complexity measure of motor current signals for tool flute breakage detection in end milling," *International Journal of Machine Tools and Manufacture*, vol. 48, no. 3-4, pp. 371–379, 2008.

[8] 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, no. 12, pp. 3119–3129, 2011.

[9] M. C. Yesilli, F. A. Khasawneh, and A. Otto, "On transfer learning for chatter detection in turning using wavelet packet transform and ensemble empirical mode decomposition," *CIRP Journal of Manufacturing Science and Technology*, vol. 28, pp. 118–135, 2020.

[10] M. C. Yesilli, F. A. Khasawneh, and A. Otto, "Topological feature vectors for chatter detection in turning processes," *The International Journal of Advanced Manufacturing Technology*, vol. 119, no. 9-10, pp. 5687–5713, 2022.

[11] M. C. Yesilli, F. A. Khasawneh, and A. Otto, "Chatter detection in turning using machine learning and similarity measures of time series via dynamic time warping," *Journal of Manufacturing Processes*, vol. 77, pp. 190–206, 2022.

[12] M. C. Yesilli and F. A. Khasawneh, "Data driven model identification for a chaotic pendulum with variable interaction potential," in *Volume 2: 16th International Conference on Multibody Systems, Nonlinear Dynamics, and Control*, American Society of Mechanical Engineers, aug 2020.

[13] A. Bruzzone, H. Costa, P. Lonardo, and D. Lucca, "Advances in engineered surfaces for functional performance," *CIRP Annals*, vol. 57, no. 2, pp. 750–769, 2008.

[14] T. D. B. Jacobs, T. Junge, and L. Pastewka, "Quantitative characterization of surface topography using spectral analysis," *Surface Topography: Metrology and Properties*, vol. 5, no. 1, p. 013001, 2017.

[15] A. Majumdar and C. Tien, "Fractal characterization and simulation of rough surfaces," *Wear*, vol. 136, no. 2, pp. 313–327, 1990.

[16] M. Hasegawa, J. Liu, K. Okuda, and M. Nunobiki, "Calculation of the fractal dimensions of machined surface profiles," *Wear*, vol. 192, no. 1-2, pp. 40–45, 1996.

[17] A. Wang, C. Yang, and X. Yuan, "Evaluation of the wavelet transform method for machined surface topography i: methodology validation," *Tribology International*, vol. 36, no. 7, pp. 517–526, 2003.

[18] A. Wang, C. Yang, and X. Yuan, "Evaluation of the wavelet transform method for machined surface topography 2: fractal characteristic analysis," *Tribology International*, vol. 36, no. 7, pp. 527–535, 2003.

[19] G. Petropoulos, W. Bouzid, C. Pandazaras, and D. Dramalist, "Fractal geometry of metal surfaces obtained by turning," *Materials Technology*, vol. 21, no. 3, pp. 163–169, 2006.

[20] X. Zuo, H. Zhu, Y. Zhou, and J. Yang, "Estimation of fractal dimension and surface roughness based on material characteristics and cutting conditions in the end milling of carbon steels," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 231, no. 8, pp. 1423–1437, 2015.

[21] Z. Chen, Y. Liu, and P. Zhou, "A comparative study of fractal dimension calculation methods for rough surface profiles," *Chaos, Solitons & Fractals*, vol. 112, pp. 24–30, 2018.

[22] F. W. Taylor, "On the art of cutting metals," *Transactions of ASME*, vol. 43, pp. 31–350, 1907.

[23] Y. Altintas and M. Weck, "Chatter stability of metal cutting and grinding," *CIRP Annals*, vol. 53, pp. 619 – 642, 2004.

[24] G. Quintana and J. Ciurana, "Chatter in machining processes: A review," *International Journal of Machine Tools and Manufacture*, vol. 51, no. 5, pp. 363–376, 2011.

[25] J. Munoa, X. Beudaert, Z. Dombovari, Y. Altintas, E. Budak, C. Brecher, and G. Stepan, "Chatter suppression techniques in metal cutting," *CIRP Annals*, vol. 65, no. 2, pp. 785–808, 2016.

[26] Y. Altintas, *Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations, and CNC Design*. New York: Cambridge University Press, 2012.

[27] A. Otto, S. Rauh, M. Kolouch, and G. Radons, "Extension of tlusty's law for the identification of chatter stability lobes in multi-dimensional cutting processes," *Int. J. Mach. Tools Manuf.*, vol. 82–83, pp. 50 – 58, 2014.

[28] S. Smith and J. Tlusty, "Stabilizing chatter by automatic spindle speed regulation," *{CIRP} Annals - Manufacturing Technology*, vol. 41, no. 1, pp. 433 – 436, 1992.

[29] Y. Altintas and P. K. Chan, "In-process detection and suppression of chatter in milling," *International Journal of Machine Tools and Manufacture*, vol. 32, no. 3, pp. 329 – 347, 1992.

[30] T. Choi and Y. C. Shin, "On-line chatter detection using wavelet-based parameter estimation," *Journal of Manufacturing Science and Engineering*, vol. 125, no. 1, pp. 21–28, 2003.

[31] E. Kuljanic, G. Totis, and M. Sortino, "Development of an intelligent multisensor chatter detection system in milling," *Mechanical Systems and Signal Processing*, vol. 23, no. 5, pp. 1704 – 1718, 2009.

[32] Z. Yao, D. Mei, and Z. Chen, "On-line chatter detection and identification based on wavelet and support vector machine," *Journal of Materials Processing Technology*, vol. 210, no. 5, pp. 713 – 719, 2010.

[33] J. Elias and V. Narayanan Namboothiri, "Cross-recurrence plot quantification analysis of input and output signals for the detection of chatter in turning," *Nonlinear Dynamics*, vol. 76, no. 1, pp. 255–261, 2014.

[34] J. Tlusty and G. Andrews, "A critical review of sensors for unmanned machining," *{CIRP} Annals - Manufacturing Technology*, vol. 32, no. 2, pp. 563 – 572, 1983.

[35] T. Delio, J. Tlusty, and S. Smith, "Use of audio signals for chatter detection and control," *Journal of Manufacturing Science and Engineering*, vol. 114, no. 2, pp. 146–157, 1992.

[36] J. Gradisek, E. Govekar, and I. Grabec, "Using coarse-grained entropy rate to detect chatter in cutting," *Journal of Sound and Vibration*, vol. 214, no. 5, pp. 941–952, 1998.

[37] T. L. Schmitz, K. Medicus, and B. Dutterer, "Exploring once-per-revolution audio signal variance as a chatter indicator," *Machining Science and Technology*, vol. 6, no. 2, pp. 215–233, 2002.

[38] I. Bediaga, J. Muñoa, J. Hernández, and L. L. de Lacalle, "An automatic spindle speed selection strategy to obtain stability in high-speed milling," *Int. J. Mach. Tools Manuf.*, vol. 49, no. 5, pp. 384 – 394, 2009.

[39] N. D. Sims, "Dynamics diagnostics: Methods, equipment and analysis tools," in *Machining Dynamics* (K. Cheng, ed.), Springer Series in Advanced Manufacturing, pp. 85–115, Springer London, 2009.

[40] U. Nair, B. Krishna, V. Namboothiri, and V. Nampoori, "Permutation entropy based real-time chatter detection using audio signal in turning process," *The International Journal of Advanced Manufacturing Technology*, vol. 46, no. 1-4, pp. 61–68, 2010.

[41] N. J. M. van Dijk, E. J. J. Doppenberg, R. P. H. Faassen, N. van de Wouw, J. A. J. Oosterling, and H. Nijmeijer, "Automatic in-process chatter avoidance in the high-speed milling process," *Journal of Dynamic Systems, Measurement, and Control*, vol. 132, no. 3, p. 031006, 2010.

[42] N.-C. Tsai, D.-C. Chen, and R.-M. Lee, "Chatter prevention for milling process by acoustic signal feedback," *The International Journal of Advanced Manufacturing Technology*, vol. 47, no. 9-12, pp. 1013–1021, 2010.

[43] Y. Kakinuma, Y. Sudo, and T. Aoyama, "Detection of chatter vibration in end milling applying disturbance observer," *{CIRP} Annals - Manufacturing Technology*, vol. 60, no. 1, pp. 109 – 112, 2011.

[44] L. Ma, S. N. Melkote, and J. B. Castle, "A model-based computationally efficient method for on-line detection of chatter in milling," *Journal of Manufacturing Science and Engineering*, vol. 135, no. 3, p. 031007, 2013.

[45] Y. Chen, H. Li, L. Hou, J. Wang, and X. Bu, "An intelligent chatter detection method based on EEMD and feature selection with multi-channel vibration signals," *Measurement*, vol. 127, pp. 356–365, 2018.

[46] Y. Wang, Q. Bo, H. Liu, L. Hu, and H. Zhang, "Mirror milling chatter identification using q-factor and SVM," *The International Journal of Advanced Manufacturing Technology*, vol. 98, no. 5-8, pp. 1163–1177, 2018.

[47] S. Saravanamurugan, S. Thiyagu, N. Sakthivel, and B. B. Nair, "Chatter prediction in boring process using machine learning technique," *International Journal of Manufacturing Research*, vol. 12, no. 4, p. 405, 2017.

[48] 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, no. 9-12, pp. 2731–2746, 2016.

[49] 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, no. 11, pp. 1471–1482, 2016.

[50] 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)*, pp. 1052–1057, IEEE, 2017.

[51] S. Qian, Y. Sun, and Z. Xiong, "Intelligent chatter detection based on wavelet packet node energy and LSSVM-RFE," in *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE, 2015.

[52] X. Li, Z. H. Yao, and Z. C. Chen, "An effective EMD-based feature extraction method for boring chatter recognition," *Applied Mechanics and Materials*, vol. 34-35, pp. 1058–1063, 2010.

[53] H. Cao, Y. Lei, and Z. He, "Chatter identification in end milling process using wavelet packets and Hilbert – Huang transform," *International Journal of Machine Tools and Manufacture*, vol. 69, pp. 11–19, 2013.

[54] 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, no. 7, pp. 1251–1266, 2013.

[55] M. C. Yesilli and F. A. Khasawneh, "On transfer learning of traditional frequency and time domain features in turning," in *Volume 2: Manufacturing Processes; Manufacturing Systems; Nano/Micro/Meso Manufacturing; Quality and Reliability*, American Society of Mechanical Engineers, 2020.

[56] M.-K. Liu, M.-Q. Tran, C. Chung, and Y.-W. Qui, "Hybrid model- and signal-based chatter detection in the milling process," *Journal of Mechanical Science and Technology*, vol. 34, no. 1, pp. 1–10, 2020.

[57] C. Liu, L. Zhu, and C. Ni, "Chatter detection in milling process based on VMD and energy entropy," *Mechanical Systems and Signal Processing*, vol. 105, pp. 169–182, 2018.

[58] Y. S. Tarng and M. C. Chen, "An intelligent sensor for detection of milling chatter," *Journal of Intelligent Manufacturing*, vol. 5, no. 3, pp. 193–200, 1994.

[59] S. Tangjitsitcharoen, "Advance in detection system to improve the stability and capability of CNC turning process," *Journal of Intelligent Manufacturing*, vol. 22, no. 6, pp. 843–852, 2009.

[60] Y. Fu, Y. Zhang, H. Gao, T. Mao, H. Zhou, R. Sun, and D. Li, "Automatic feature constructing from vibration signals for machining state monitoring," *Journal of Intelligent Manufacturing*, vol. 30, no. 3, pp. 995–1008, 2017.

[61] Cherukuri, Perez-Bernabeu, Selles, and Schmitz, "Machining chatter prediction using a data learning model," *Journal of Manufacturing and Materials Processing*, vol. 3, no. 2, p. 45, 2019.

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

[63] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society*, vol. 46, no. 2, pp. 255–308, 2009. Survey.

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

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

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

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

[68] 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.

[69] 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.

[70] 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*, ASME, 2014. Paper no. IMECE2014-40221.

[71] 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.

[72] J. A. Perea, E. Munch, and F. A. Khasawneh, "Approximating Continuous Functions on Persistence Diagrams Using Template Functions," *arXiv preprint: 1902.07190*, 2019.

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

[74] 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," *Journal of Machine Learning Research*, vol. 18, 2017.

[75] 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)*, 2015.

[76] 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.

[77] S. Tsuji and K. Aihara, "A fast method of computing persistent homology of time series data," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019.

[78] N. J. Cavanna, M. Jahanseir, and D. R. Sheehy, "A geometric perspective on sparse filtrations," 2015.

[79] C. Myers, L. Rabiner, and A. Rosenberg, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 6, pp. 623–635, 1980.

[80] C. S. Myers and L. R. Rabiner, "A comparative study of several dynamic time-warping algorithms for connected-word recognition," *Bell System Technical Journal*, vol. 60, no. 7, pp. 1389–1409, 1981.

[81] H. Sakoe, S. Chiba, A. Waibel, and K. Lee, "Dynamic programming algorithm optimization for spoken word recognition," *Readings in speech recognition*, vol. 159, p. 224, 1990.

[82] B.-H. Juang, "On the hidden markov model and dynamic time warping for speech recognition-a unified view," *AT&T Bell Laboratories Technical Journal*, vol. 63, no. 7, pp. 1213–1243, 1984.

[83] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 67–72, 1975.

[84] V. Niennattrakul and C. A. Ratanamahatana, "On clustering multimedia time series data using k-means and dynamic time warping," in *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, IEEE, 2007.

[85] F. Yu, K. Dong, F. Chen, Y. Jiang, and W. Zeng, "Clustering time series with granular dynamic time warping method," in *2007 IEEE International Conference on Granular Computing (GRC 2007)*, pp. 393–398, IEEE, 2007.

[86] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh, "Dynamic time warping averaging of time series allows faster and more accurate classification," in *2014 IEEE International Conference on Data Mining*, pp. 470–479, IEEE, 2014.

[87] A. P. Shanker and A. Rajagopalan, "Off-line signature verification using DTW," *Pattern Recognition Letters*, vol. 28, no. 12, pp. 1407–1414, 2007.

[88] M. Munich and P. Perona, "Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, IEEE, 1999.

[89] M. Parizeau and R. Plamondon, "A comparative analysis of regional correlation, dynamic time warping, and skeletal tree matching for signature verification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 710–717, 1990.

[90] R. Martens and L. Claesen, "On-line signature verification by dynamic time-warping," in *Proceedings of 13th International Conference on Pattern Recognition*, pp. 38–42, IEEE, 1996.

[91] M. Postel, B. Bugdayci, and K. Wegener, "Ensemble transfer learning for refining stability predictions in milling using experimental stability states," *The International Journal of Advanced Manufacturing Technology*, vol. 107, no. 9, pp. 4123–4139, 2020.

[92] H. O. Unver and B. Sener, "A novel transfer learning framework for chatter detection using convolutional neural networks," *Journal of Intelligent Manufacturing*, pp. 1–20, 2021.

[93] W. Li and Y. Liang, "Deep transfer learning based diagnosis for machining process lifecycle," *Procedia CIRP*, vol. 90, pp. 642–647, 2020.

[94] J. Wu, Z. Zhao, C. Sun, R. Yan, and X. Chen, "Few-shot transfer learning for intelligent fault diagnosis of machine," *Measurement*, vol. 166, p. 108202, 2020.

[95] Y.-M. Kim, S.-J. Shin, and H.-W. Cho, "Predictive modeling for machining power based on multi-source transfer learning in metal cutting," *International Journal of Precision Engineering and Manufacturing-Green Technology*, 2021.

[96] H. Mamledesai, M. A. Soriano, and R. Ahmad, "A qualitative tool condition monitoring framework using convolution neural network and transfer learning," *Applied Sciences*, vol. 10, no. 20, p. 7298, 2020.

[97] M. Marei, S. El Zaatari, and W. Li, "Transfer learning enabled convolutional neural networks for estimating health state of cutting tools," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102145, 2021.

[98] J. Wang, B. Zou, M. Liu, Y. Li, H. Ding, and K. Xue, "Milling force prediction model based on transfer learning and neural network," *Journal of Intelligent Manufacturing*, vol. 32, pp. 947–956, 2021.

[99] P. Wang and R. X. Gao, "Transfer learning for enhanced machine fault diagnosis in manufacturing," *CIRP Annals*, vol. 69, no. 1, pp. 413–416, 2020.

[100] Y. Kim, T. Kim, B. D. Youn, and S.-H. Ahn, "Machining quality monitoring (mqm) in laser-assisted micro-milling of glass using cutting force signals: An image-based deep transfer learning," *Journal of Intelligent Manufacturing*, pp. 1–16, 2021.

[101] Z. Gao, Q. Hu, and X. Xu, "Condition monitoring and life prediction of the turning tool based on extreme learning machine and transfer learning," *Neural Computing and Applications*, pp. 1–12, 2021.

[102] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, (New York, NY, USA), p. 193–200, Association for Computing Machinery, 2007.

[103] F. Shen, C. Chen, R. Yan, and R. X. Gao, "Bearing fault diagnosis based on svd feature extraction and transfer learning classification," in *2015 Prognostics and System Health Management Conference (PHM)*, pp. 1–6, IEEE, 2015.

[104] G. Chen, Y. Li, and X. Liu, "Pose-dependent tool tip dynamics prediction using transfer learning," *International Journal of Machine Tools and Manufacture*, vol. 137, pp. 30–41, 2019.

[105] E. V. Ruiz, F. C. Nolla, and H. R. Segovia, "Is the DTW "distance" really a metric? an algorithm reducing the number of DTW comparisons in isolated word recognition," *Speech Communication*, vol. 4, no. 4, pp. 333–344, 1985.

[106] E. Vidal and M.-J. Lloret, "Fast speaker-independent DTW recognition of isolated words using a metric-space search algorithm (AESA)," *Speech Communication*, vol. 7, no. 4, pp. 417–422, 1988.

[107] E. V. Ruiz, "An algorithm for finding nearest neighbours in (approximately) constant average time," *Pattern Recognition Letters*, vol. 4, no. 3, pp. 145–157, 1986.

[108] M. L. Micó, J. Oncina, and E. Vidal, "A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements," *Pattern Recognition Letters*, vol. 15, no. 1, pp. 9–17, 1994.

[109] J. R. Rico-Juan and L. Micó, "Comparison of AESA and LAESA search algorithms using string and tree-edit-distances," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1417–1426, 2003.

[110] M. C. Yesilli, F. A. Khasawneh, and B. Mann, "Transfer learning for autonomous chatter detection in machining," *arXiv preprint: 2204.05400*, Apr. 2022.

[111] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[112] J. Weston and C. Watkins, "Multi-class support vector machines," tech. rep., 1998.

[113] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2008.

[114] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of Biomedical Informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[115] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, vol. 398. John Wiley & Sons, 2013.

[116] C.-Y. J. Peng, K. L. Lee, and G. M. Ingersoll, "An introduction to logistic regression analysis and reporting," *The Journal of Educational Research*, vol. 96, no. 1, pp. 3–14, 2002.

[117] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[118] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[119] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24–31, 2016.

[120] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[121] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in Neurorobotics*, vol. 7, 2013.

[122] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[123] K. Yan and D. Zhang, "Feature selection and analysis on correlated gas sensor data with recursive feature elimination," *Sensors and Actuators B: Chemical*, vol. 212, pp. 353–363, 2015.

[124] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1971, pp. 903–995, 1998.

[125] Z. WU and N. E. HUANG, "Ensemble Empirical Mode Decomposition: A Noise Assisted Data Analysis Method," *Advances in Adaptive Data Analysis*, vol. 01, no. 01, pp. 1–41, 2009.

[126] T. Wang, M. Zhang, Q. Yu, and H. Zhang, "Comparing the applications of EMD and EEMD on time–frequency analysis of seismic signal," *Journal of Applied Geophysics*, vol. 83, pp. 29–34, 2012.

[127] O. Pele and M. Werman, "A linear time histogram metric for improved sift matching," in *Computer Vision–ECCV 2008*, pp. 495–508, Springer, 2008.

[128] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 460–467, IEEE, 2009.

[129] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1/3, pp. 389–422, 2002.

[130] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," in *KDD workshop*, vol. 10, pp. 359–370, Seattle, WA, 1994.

[131] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[132] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[133] R. Han, Y. Li, X. Gao, and S. Wang, "An accurate and rapid continuous wavelet dynamic time warping algorithm for end-to-end mapping in ultra-long nanopore sequencing," *Bioinformatics*, vol. 34, no. 17, pp. i722–i731, 2018.

[134] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 4, pp. 325–327, 1976.

[135] T. Horváth, S. Wrobel, and U. Bohnebeck, "Relational instance-based learning with lists and terms," *Machine Learning*, vol. 43, no. 1, pp. 53–80, 2001.

[136] B. Choudhary, *The elements of complex analysis*. New York: J. Wiley, 1993.

[137] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 262–270, ACM, 2012.

[138] 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.

[139] J. R. Munkres, *Elements of Algebraic Topology*. CRC Press, 2018.

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

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

[142] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American Statistical Association*, vol. 79, no. 388, pp. 871–880, 1984.

[143] A. Myers and F. A. Khasawneh, "On the automatic parameter selection for permutation entropy," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 3, p. 033130, 2020.

[144] 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, no. 3, pp. 521–528, 2016.

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

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

[147] 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, no. 3, pp. 1840–1853, 1994.

[148] A. Tharwat, M. Elhoseny, A. E. Hassanien, T. Gabel, and A. Kumar, "Intelligent bézier curve-based path planning model using chaotic particle swarm optimization algorithm," *Cluster Computing*, vol. 22, no. S2, pp. 4745–4766, 2018.

[149] M. Elhoseny, A. Tharwat, and A. E. Hassanien, "Bézier curve based path planning in a dynamic field using modified genetic algorithm," *Journal of Computational Science*, vol. 25, pp. 339–350, 2018.

[150] J. wung Choi, R. Curry, and G. Elkaim, "Path planning based on bézier curve for autonomous ground vehicles," in *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, IEEE, 2008.

[151] J.-H. Hwang, R. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bézier curve on-line trajectory generation for supervisory control," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, IEEE, 2003.

[152] S. C. Endres, C. Sandrock, and W. W. Focke, "A simplicial homology algorithm for lipschitz optimisation," *Journal of Global Optimization*, vol. 72, no. 2, pp. 181–217, 2018.

[153] M. Kerber, D. Morozov, and A. Nigmetov, "Geometry helps to compare persistence diagrams," *arXiv preprint, arXiv:1606.03357v1*, vol. cs.CG, pp. 1–20, 2016.

[154] A. D. Myers, M. Yesilli, S. Tymochko, F. Khasawneh, and E. Munch, "Teaspoon: A comprehensive python package for topological signal processing," in *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

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

[156] E. Berry, Y.-C. Chen, J. Cisewski-Kehe, and B. T. Fasy, "Functional summaries of persistence diagrams," *Journal of Applied and Computational Topology*, vol. 4, pp. 211–262, 2018.

[157] 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, 2018.

[158] 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.

[159] 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.

[160] Q. Zhao and Y. Wang, "Learning metrics for persistence-based summaries and applications for graph classification," *arXiv preprint, arXiv:1904.12189v1*, vol. cs.CG, pp. 1–21, 2019.

[161] 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.

[162] G. Kusano, K. Fukumizu, and Y. Hiraoka, "Kernel method for persistence diagrams via kernel embedding and weight factor," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6947–6987, 2017.

[163] M. Carrière, M. Cuturi, and S. Oudot, "Sliced wasserstein kernel for persistence diagrams," *arXiv preprint, arXiv:1706.03358*, vol. cs.CG, 2017.

[164] G. Kusano, "Persistence weighted gaussian kernel for probability distributions on the space of persistence diagrams," *arXiv preprint, arXiv:1803.08269v1*, vol. math.AT, 2018.

[165] 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`.

[166] I. Chevyrev and A. Kormilitzin, "A primer on the signature method in machine learning," *arXiv preprint, arXiv:1603.03788v1*, vol. stat.ML, pp. 1–45, 2016.

[167] Y. Altintas, *Manufacturing Automation*. Cambridge University Press, 2009.

[168] O. Bobrenkov, F. Khasawneh, E. Butcher, and B. Mann, "Analysis of milling dynamics for simultaneously engaged cutting teeth," *Journal of Sound and Vibration*, vol. 329, no. 5, pp. 585–606, 2010.

[169] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[170] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[171] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, 2016.

[172] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: A survey," *Knowledge-Based Systems*, vol. 80, pp. 14–23, 2015.

[173] J.-N. Juang and R. S. Pappa, "An eigensystem realization algorithm for modal parameter identification and model reduction," *Journal of Guidance, Control, and Dynamics*, vol. 8, no. 5, pp. 620–627, 1985.

[174] H. Ye, R. J. Beamish, S. M. Glaser, S. C. H. Grant, C.-H. Hsieh, L. J. Richards, J. T. Schnute, and G. Sugihara, "Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling," *Proceedings of the National Academy of Sciences*, vol. 112, no. 13, pp. E1569–E1576, 2015.

[175] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidid, O. Runborg, C. Theodoropoulos, *et al.*, "Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis," *Communications in Mathematical Sciences*, vol. 1, no. 4, pp. 715–762, 2003.

[176] A. J. Roberts, *Model Emergent Dynamics in Complex Systems*. CAMBRIDGE, 2015.

[177] M. D. Schmidt, R. R. Vallabhajosyula, J. W. Jenkins, J. E. Hood, A. S. Soni, J. P. Wikswo, and H. Lipson, "Automated refinement and inference of analytical models for metabolic networks," *Physical Biology*, vol. 8, no. 5, p. 055011, 2011.

[178] B. C. Daniels, W. S. Ryu, and I. Nemenman, "Automated, predictive, and interpretable inference of caenorhabditis elegans escape dynamics," *Proceedings of the National Academy of Sciences*, vol. 116, no. 15, pp. 7226–7231, 2019.

[179] B. C. Daniels and I. Nemenman, "Automated adaptive inference of phenomenological dynamical models," *Nature Communications*, vol. 6, no. 1, 2015.

[180] D. Giannakis and A. J. Majda, "Nonlinear laplacian spectral analysis for time series with intermittency and low-frequency variability," *Proceedings of the National Academy of Sciences*, vol. 109, no. 7, pp. 2222–2227, 2012.

[181] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 104, no. 24, pp. 9943–9948, 2007.

[182] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[183] M. Quade, M. Abel, K. Shafi, R. K. Niven, and B. R. Noack, "Prediction of dynamical systems by symbolic regression," *Physical Review E*, vol. 94, no. 1, 2016.

[184] X. Sun, L. Jin, and M. Xiong, "Extended Kalman filter for estimation of parameters in nonlinear state-space models of biochemical networks," *PLoS ONE*, vol. 3, no. 11, p. e3758, 2008.

[185] T. Qin, K. Wu, and D. Xiu, "Data driven governing equations approximation using deep neural networks," *Journal of Computational Physics*, vol. 395, pp. 620–635, 2019.

[186] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[187] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, "Discovering physical concepts with neural networks," *Physical Review Letters*, vol. 124, no. 1, 2020.

[188] J. P. Crutchfield and B. S. McNamara, "Equations of motion from a data series," *Complex systems*, vol. 1, no. 417-452, p. 121, 1987.

[189] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[190] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[191] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer New York, 2009.

[192] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher, "Sparse dynamics for partial differential equations," *Proceedings of the National Academy of Sciences*, vol. 110, no. 17, pp. 6634–6639, 2013.

[193] G. Tran and R. Ward, "Exact recovery of chaotic systems from highly corrupted data," *Multiscale Modeling & Simulation*, vol. 15, no. 3, pp. 1108–1129, 2017.

[194] H. Schaeffer, "Learning partial differential equations via data discovery and sparse optimization," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2197, p. 20160446, 2017.

[195] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Inferring biological networks by sparse identification of nonlinear dynamics," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 2, no. 1, pp. 52–63, 2016.

[196] J. L. Proctor, S. L. Brunton, B. W. Brunton, and J. N. Kutz, "Exploiting sparsity and equation-free architectures in complex systems," *The European Physical Journal Special Topics*, vol. 223, no. 13, pp. 2665–2684, 2014.

[197] D. Rey, M. Eldridge, M. Kostuk, H. D. Abarbanel, J. Schumann-Bischoff, and U. Parlitz, "Accurate state and parameter estimation in nonlinear systems with sparse observations," *Physics Letters A*, vol. 378, no. 11-12, pp. 869–873, 2014.

[198] W.-X. Wang, Y.-C. Lai, and C. Grebogi, "Data based identification and prediction of nonlinear and complex dynamical systems," *Physics Reports*, vol. 644, pp. 1–76, 2016.

[199] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science Advances*, vol. 3, no. 4, p. e1602614, 2017.

[200] G. F. Smits and M. Kotanchek, *Pareto-Front Exploitation in Symbolic Regression*, pp. 283–299. Boston, MA: Springer US, 2005.

[201] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics with control (SINDYc)," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 710–715, 2016.

[202] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2219, p. 20180335, 2018.

[203] L. Boninsegna, F. Nüske, and C. Clementi, "Sparse learning of stochastic dynamical equations," *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241723, 2018.

[204] S. K. Goharoodi, K. Dekemele, L. Dupre, M. Loccufier, and G. Crevecoeur, "Sparse identification of nonlinear duffing oscillator from measurement data," *IFAC-PapersOnLine*, vol. 51, no. 33, pp. 162–167, 2018.

[205] S. Li, E. Kaiser, S. Laima, H. Li, S. L. Brunton, and J. N. Kutz, "Discovering time-varying aerodynamics of a prototype bridge by sparse identification of nonlinear dynamical systems," *Physical Review E*, vol. 100, no. 2, 2019.

[206] I. Knowles and R. Wallace, "A variational method for numerical differentiation," *Numerische Mathematik*, vol. 70, no. 1, pp. 91–110, 1995.

[207] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[208] R. Chartrand, "Numerical differentiation of noisy, nonsmooth data," *ISRN Applied Mathematics*, vol. 2011, pp. 1–11, 2011.

[209] N. Mork, M. C. Yesilli, and F. A. Khasawneh, "Design of chaotic pendulum with a variable interaction potential, Zenodo, DOI: 10.5281/zenodo.3784897," 2020.

[210] V. Tran, E. Brost, M. Johnston, and J. Jalkio, "Predicting the behavior of a chaotic pendulum with a variable interaction potential," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 23, no. 3, p. 033103, 2013.

[211] C. M. Hurvich and C.-L. Tsai, "Regression and time series model selection in small samples," *Biometrika*, vol. 76, no. 2, pp. 297–307, 1989.

[212] M. A. Babyak, "What you see may not be what you get: A brief, nontechnical introduction to overfitting in regression-type models," *Psychosomatic Medicine*, vol. 66, no. 3, pp. 411–421, 2004.

[213] I. Knowles and R. J. Renka, "Methods for numerical differentiation of noisy data," *Electron. J. Differ. Equ*, vol. 21, pp. 235–246, 2014.

[214] R. Schafer, "What is a Savitzky-Golay Filter? [lecture notes]," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 111–117, 2011.

[215] A. Y. Suh, A. A. Polycarpou, and T. F. Conry, "Detailed surface roughness characterization of engineering surfaces undergoing tribological testing leading to scuffing," *Wear*, vol. 255, no. 1-6, pp. 556–568, 2003.

[216] N. Makarenko, M. Kalimoldayev, I. Pak, and A. Yessenaliyeva, "Texture recognition by the methods of topological data analysis," *Open Engineering*, vol. 6, no. 1, 2016.

[217] A. A. Taha and A. Hanbury, "Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool," *BMC Medical Imaging*, vol. 15, no. 1, 2015.

[218] S. Wu, B. Zhang, Y. Liu, X. Suo, and H. Li, "Influence of surface topography on bacterial adhesion: A review (review)," *Biointerphases*, vol. 13, no. 6, p. 060801, 2018.

[219] ISO, "ISO 16610-21 Geometrical product specifications (GPS) — Filtration — Part 21: Linear profile filters: Gaussian filters," 2011.

[220] J. Raja, B. Muralikrishnan, and S. Fu, "Recent advances in separation of roughness, waviness and form," *Precision Engineering*, vol. 26, no. 2, pp. 222–235, 2002.

[221] L. Gurau, H. Mansfield-Williams, and M. Irle, "Processing roughness of sanded wood surfaces," *Holz als Roh- und Werkstoff*, vol. 63, no. 1, pp. 43–52, 2004.

[222] B. Hendarto, E. Shayan, B. Ozarska, and R. Carr, "Analysis of roughness of a sanded wood surface," *The International Journal of Advanced Manufacturing Technology*, vol. 28, no. 7, pp. 775–780, 2006.

[223] D. Janecki, "Gaussian filters with profile extrapolation," *Precision Engineering*, vol. 35, no. 4, pp. 602–606, 2011.

[224] J. Raja and V. Radhakrishnan, "Filtering of surface profiles using fast fourier transform," *International Journal of Machine Tool Design and Research*, vol. 19, no. 3, pp. 133–141, 1979.

[225] U. G. Indahl and T. Naes, "Evaluation of alternative spectral feature extraction methods of textural images for multivariate modelling," *Journal of Chemometrics*, vol. 12, no. 4, pp. 261–278, 1998.

[226] Z. Peng and T. Kirk, "Two-dimensional fast fourier transform and power spectrum for wear particle analysis," *Tribology International*, vol. 30, no. 8, pp. 583–590, 1997.

[227] W. P. Dong and K. J. Stout, "Two-dimensional fast fourier transform and power spectrum for surface roughness in three dimensions," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 209, no. 5, pp. 381–391, 1995.

[228] J. Nunes, Y. Bouaoune, E. Delechelle, O. Niang, and P. Bunel, "Image analysis by bidimensional empirical mode decomposition," *Image and Vision Computing*, vol. 21, no. 12, pp. 1019–1026, 2003.

[229] A. LINDERHED, "IMAGE EMPIRICAL MODE DECOMPOSITION: A NEW TOOL FOR IMAGE PROCESSING," *Advances in Adaptive Data Analysis*, vol. 01, no. 02, pp. 265–294, 2009.

[230] Y. Xia, B. Zhang, W. Pei, and D. P. Mandic, "Bidimensional multivariate empirical mode decomposition with applications in multi-scale image fusion," *IEEE Access*, vol. 7, pp. 114261–114270, 2019.

[231] H.-D. Lin, "Tiny surface defect inspection of electronic passive components using discrete cosine transform decomposition and cumulative sum techniques," *Image and Vision Computing*, vol. 26, no. 5, pp. 603–621, 2008.

[232] J. Lecompte, O. Legoff, and J.-Y. Hascoet, "Technological form defects identification using discrete cosine transform method," *The International Journal of Advanced Manufacturing Technology*, vol. 51, no. 9-12, pp. 1033 –1044, 2010.

[233] P. H. Chandankhede, P. V. Puranik, and P. R. Bajaj, "Soft computing tool approach for texture classification using discrete cosine transform," in *2011 3rd International Conference on Electronics Computer Technology*, IEEE, 2011.

[234] G. L. Goïc, M. Bigerelle, S. Samper, H. Favrelière, and M. Pillet, "Multiscale roughness analysis of engineering surfaces: A comparison of methods for the investigation of functional correlations," *Mechanical Systems and Signal Processing*, vol. 66-67, pp. 437–457, 2016.

[235] X. Chen, J. Raja, and S. Simanapalli, "Multi-scale analysis of engineering surfaces," *International Journal of Machine Tools and Manufacture*, vol. 35, no. 2, pp. 231–238, 1995.

[236] X. Liu and J. Raja, "Analyzing engineering surface texture using wavelet filter," in *Wavelet Applications in Signal and Image Processing IV* (M. A. Unser, A. Aldroubi, and A. F. Laine, eds.), SPIE, 1996.

[237] S. Fu, B. Muralikrishnan, and J. Raja, "Engineering surface analysis with different wavelet bases," *Journal of Manufacturing Science and Engineering*, vol. 125, no. 4, pp. 844–852, 2003.

[238] B. Josso, D. R. Burton, and M. J. Lalor, "Frequency normalised wavelet transform for surface roughness analysis and characterisation," *Wear*, vol. 252, no. 5-6, pp. 491–500, 2002.

[239] P. Morala-Argüello, J. Barreiro, and E. Alegre, "A evaluation of surface roughness classes by computer vision using wavelet transform in the frequency domain," *The International Journal of Advanced Manufacturing Technology*, vol. 59, no. 1, pp. 213–220, 2012.

[240] S. I. Chang and J. S. Ravathur, "Computer vision based non-contact surface roughness assessment using wavelet transform and response surface methodology," *Quality Engineering*, vol. 17, no. 3, pp. 435–451, 2005.

[241] X. Wang, T. Shi, G. Liao, Y. Zhang, Y. Hong, and K. Chen, "Using wavelet packet transform for surface roughness evaluation and texture extraction," *Sensors*, vol. 17, no. 4, p. 933, 2017.

[242] Q. Chen, S. Yang, and Z. Li, "Surface roughness evaluation by using wavelets analysis," *Precision Engineering*, vol. 23, no. 3, pp. 209–212, 1999.

[243] K. Stępień, W. Makiela, A. Stoić, and I. Samardžić, "Defining the criteria to select the wavelet type for the assessment of surface quality," *Tehnički vjesnik–Technical Gazette*, vol. 22, no. 3, pp. 781–784, 2015.

[244] M. C. Yesilli, J. Chen, F. A. Khasawneh, and Y. Guo, "Automated surface texture analysis via discrete cosine transform and discrete wavelet transform," *arXiv preprint: 2204.05968*, Apr. 2022.

[245] ISO, "ISO 25178 Geometrical product specifications (GPS) — Surface texture: Areal."

[246] ISO, "ISO 21920 Geometrical product specifications (GPS) —Surface texture: Profile."

[247] M. C. Yesilli and F. A. Khasawneh, "Data-driven and automatic surface texture analysis using persistent homology," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, dec 2021.

[248] M. H. Müser, W. B. Dapp, R. Bugnicourt, P. Sainsot, N. Lesaffre, T. A. Lubrecht, B. N. J. Persson, K. Harris, A. Bennett, K. Schulze, S. Rohde, P. Ifju, W. G. Sawyer, T. Angelini, H. A. Esfahani, M. Kadkhodaei, S. Akbarzadeh, J.-J. Wu, G. Vorlaufer, A. Vernes, S. Solhjoo, A. I. Vakis, R. L. Jackson, Y. Xu, J. Streator, A. Rostami, D. Dini, S. Medina, G. Carbone, F. Bottiglione, L. Afferrante, J. Monti, L. Pastewka, M. O. Robbins, and J. A. Greenwood, "Meeting the contact-mechanics challenge," *Tribology Letters*, vol. 65, no. 4, 2017.

[249] ASME, "ASME B46.1 Surface Texture (Surface Roughness, Waviness and Lay)."

[250] A. Averbuch, R. Coifman, D. Donoho, M. Elad, and M. Israeli, "Fast and accurate polar fourier transform," *Applied and Computational Harmonic Analysis*, vol. 21, no. 2, pp. 145–167, 2006.

[251] E. Berry, Y.-C. Chen, J. Cisewski-Kehe, and B. T. Fasy, "Functional summaries of persistence diagrams," *Journal of Applied and Computational Topology*, vol. 4, no. 2, pp. 211–262, 2020.

[252] R. X. Gao and R. Yan, *Wavelets: Theory and applications for manufacturing*. Springer Science & Business Media, 2010.

[253] H. Olkkonen, *Discrete Wavelet Transforms: Biomedical Applications*. BoD–Books on Demand, 2011.

[254] S. Prabhakar, A. Mohanty, and A. Sekhar, "Application of discrete wavelet transform for detection of ball bearing race faults," *Tribology International*, vol. 35, no. 12, pp. 793–800, 2002.

[255] K. Gurley and A. Kareem, "Applications of wavelet transforms in earthquake, wind and ocean engineering," *Engineering structures*, vol. 21, no. 2, pp. 149–167, 1999.

[256] D. E. Newland, *An Introduction to Random Vibrations, Spectral & Wavelet Analysis*. Guilford Publications, 2012.

[257] ISO, "ISO16610-29 Geometrical product specifications (GPS) - Filtration - Part 29: Linear profile filters: wavelets," 2020.

[258] G. Strang, "The discrete cosine transform," *SIAM review*, vol. 41, no. 1, pp. 135–147, 1999.

[259] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[260] S. M.R., "Entropy-based image registration," *PhD Thesis, Princeton University*, 2006.

[261] C. Liu, Y. Li, G. Zhou, and W. Shen, "A sensor fusion and support vector machine based approach for recognition of complex machining conditions," *Journal of Intelligent Manufacturing*, vol. 29, no. 8, pp. 1739–1752, 2016.

[262] R. Peng, J. Liu, X. Fu, C. Liu, and L. Zhao, "Application of machine vision method in tool wear monitoring," *The International Journal of Advanced Manufacturing Technology*, vol. 116, no. 3-4, pp. 1357–1372, 2021.

[263] S. Huang, K. Tan, Y. Wong, C. de Silva, H. Goh, and W. Tan, "Tool wear detection and fault diagnosis based on cutting force monitoring," *International Journal of Machine Tools and Manufacture*, vol. 47, no. 3-4, pp. 444–451, 2007.

[264] J. Wassdahl, "Modeling of wear mechanisms in mechanical cutting," 2008.

[265] P. Stavropoulos, A. Papacharalampopoulos, and T. Souflas, "Indirect online tool wear monitoring and model-based identification of process-related signal," *Advances in Mechanical Engineering*, vol. 12, no. 5, p. 168781402091920, 2020.

[266] D. Salgado and F. Alonso, "Tool wear detection in turning operations using singular spectrum analysis," *Journal of Materials Processing Technology*, vol. 171, no. 3, pp. 451–458, 2006.

[267] F. Alonso and D. Salgado, "Analysis of the structure of vibration signals for tool wear detection," *Mechanical Systems and Signal Processing*, vol. 22, no. 3, pp. 735–748, 2008.

[268] B. Kilundu, P. Dehombreux, and X. Chiementin, "Tool wear monitoring by machine learning techniques and singular spectrum analysis," *Mechanical Systems and Signal Processing*, vol. 25, no. 1, pp. 400–415, 2011.

[269] X. Li, "A brief review: acoustic emission method for tool wear monitoring during turning," *International Journal of Machine Tools and Manufacture*, vol. 42, no. 2, pp. 157–165, 2002.

[270] V. A. Pechenin, A. I. Khaimovich, A. I. Kondratiev, and M. A. Bolotov, "Method of controlling cutting tool wear based on signal analysis of acoustic emission for milling," *Procedia Engineering*, vol. 176, pp. 246–252, 2017.

[271] S. Leng, Z. Wang, T. Min, Z. Dai, and G. Chen, "Detection of tool wear in drilling CFRP/TC4 stacks by acoustic emission," *Journal of Vibration Engineering & Technologies*, vol. 8, no. 3, pp. 463–470, 2019.

[272] W. Li, W. Gong, T. Obikawa, and T. Shirakashi, "A method of recognizing tool-wear states based on a fast algorithm of wavelet transform," *Journal of Materials Processing Technology*, vol. 170, no. 1-2, pp. 374–380, 2005.

[273] T. Benkedjouh, N. Zerhouni, and S. Rechak, "Tool wear condition monitoring based on continuous wavelet transform and blind source separation," *The International Journal of Advanced Manufacturing Technology*, vol. 97, no. 9-12, pp. 3311–3323, 2018.

[274] Y. Choi, R. Narayanaswami, and A. Chandra, "Tool wear monitoring in ramp cuts in end milling using the wavelet transform," *The International Journal of Advanced Manufacturing Technology*, vol. 23, no. 5-6, pp. 419–428, 2004.

[275] L. Xiaoli and Y. Zhejun, "Tool wear monitoring with wavelet packet transform—fuzzy clustering method," *Wear*, vol. 219, no. 2, pp. 145–154, 1998.

[276] M. Hassan, A. Damir, H. Attia, and V. Thomson, "Benchmarking of pattern recognition techniques for online tool wear detection," *Procedia CIRP*, vol. 72, pp. 1451–1456, 2018.

[277] G. D. Simon and R. Deivanathan, "Early detection of drilling tool wear by vibration data acquisition and classification," *Manufacturing Letters*, vol. 21, pp. 60–65, 2019.

[278] A. Rivero, L. L. de Lacalle, and M. L. Penalva, "Tool wear detection in dry high-speed milling based upon the analysis of machine internal signals," *Mechatronics*, vol. 18, no. 10, pp. 627–633, 2008.

[279] M. Kious, A. Ouahabi, M. Boudraa, R. Serra, and A. Cheknane, "Detection process approach of tool wear in high speed milling," *Measurement*, vol. 43, no. 10, pp. 1439–1446, 2010.

[280] D. Wu, C. Jennings, J. Terpenny, R. X. Gao, and S. Kumara, "A comparative study on machine learning algorithms for smart manufacturing: Tool wear prediction using random forests," *Journal of Manufacturing Science and Engineering*, vol. 139, no. 7, 2017.

[281] W. Rmili, A. Ouahabi, R. Serra, and R. Leroy, "An automatic system based on vibratory analysis for cutting tool wear monitoring," *Measurement*, vol. 77, pp. 117–123, 2016.

[282] A. Gouarir, G. Martínez-Arellano, G. Terrazas, P. Benardos, and S. Ratchev, "In-process tool wear prediction system based on machine learning techniques and force analysis," *Procedia CIRP*, vol. 77, pp. 501–504, 2018.

[283] T. Bergs, C. Holst, P. Gupta, and T. Augspurger, "Digital image processing with deep learning for automated cutting tool wear detection," *Procedia Manufacturing*, vol. 48, pp. 947–958, 2020.

[284] X. Xu, J. Wang, B. Zhong, W. Ming, and M. Chen, "Deep learning-based tool wear prediction and its application for machining process using multi-scale feature fusion and channel attention mechanism," *Measurement*, vol. 177, p. 109254, 2021.

[285] S.-L. Chen and Y. Jen, "Data fusion neural network for tool condition monitoring in CNC milling machining," *International Journal of Machine Tools and Manufacture*, vol. 40, no. 3, pp. 381–400, 2000.

[286] Y. Shaban, S. Yacout, M. Balazinski, and K. Jemielniak, "Cutting tool wear detection using multiclass logical analysis of data," *Machining Science and Technology*, vol. 21, no. 4, pp. 526–541, 2017.

[287] D. M. D'Addona, A. M. M. S. Ullah, and D. Matarazzo, "Tool-wear prediction and pattern-recognition using artificial neural network and DNA-based computing," *Journal of Intelligent Manufacturing*, vol. 28, no. 6, pp. 1285–1301, 2015.

[288] F. J. Alonso and D. R. Salgado, "Application of singular spectrum analysis to tool wear detection using sound signals," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 219, no. 9, pp. 703–710, 2005.

[289] R. Peng, H. Pang, H. Jiang, and Y. Hu, "Study of tool wear monitoring using machine vision," *Automatic Control and Computer Sciences*, vol. 54, no. 3, pp. 259–270, 2020.

[290] Q. Hou, J. Sun, Z. Lv, P. Huang, G. Song, and C. Sun, "An online tool wear detection system in dry milling based on machine vision," *The International Journal of Advanced Manufacturing Technology*, vol. 105, no. 1-4, pp. 1801–1810, 2019.

[291] X. Wu, Y. Liu, X. Zhou, and A. Mou, "Automatic identification of tool wear based on convolutional neural network in face milling process," *Sensors*, vol. 19, no. 18, p. 3817, 2019.

[292] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[293] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Lecture Notes in Computer Science*, pp. 234–241, Springer International Publishing, 2015.

[294] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[295] W.-J. Lin, J.-W. Chen, J.-P. Jhuang, M.-S. Tsai, C.-L. Hung, K.-M. Li, and H.-T. Young, "Integrating object detection and image segmentation for detecting the tool wear area on stitched image," *Scientific Reports*, vol. 11, no. 1, 2021.

[296] M. T. García-Ordás, E. Alegre-Gutiérrez, R. Alaiz-Rodríguez, and V. González-Castro, "Tool wear monitoring using an online, automatic and low cost system based on local texture," *Mechanical Systems and Signal Processing*, vol. 112, pp. 98–112, 2018.

[297] M. Schmitt and X. X. Zhu, "Data fusion and remote sensing: An ever-growing relationship," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 4, pp. 6–23, 2016.

[298] Y. Yao, X. Li, and Z. Yuan, "Tool wear detection with fuzzy classification and wavelet fuzzy neural network," *International Journal of Machine Tools and Manufacture*, vol. 39, no. 10, pp. 1525–1538, 1999.

[299] C. Aliustaoglu, H. M. Ertunc, and H. Ocak, "Tool wear condition monitoring using a sensor fusion model based on fuzzy inference system," *Mechanical Systems and Signal Processing*, vol. 23, no. 2, pp. 539–546, 2009.

[300] J. Wu, Y. Su, Y. Cheng, X. Shao, C. Deng, and C. Liu, "Multi-sensor information fusion for remaining useful life prediction of machining tools by adaptive network based fuzzy inference system," *Applied Soft Computing*, vol. 68, pp. 13–23, 2018.

[301] W. H. Woodall, R. Koudelik, K.-L. Tsui, S. B. Kim, Z. G. Stoumbos, and C. P. Carvounis, "A review and analysis of the mahalanobis—taguchi system," *Technometrics*, vol. 45, no. 1, pp. 1–15, 2003.

[302] R. D. Maesschalck, D. Jouan-Rimbaud, and D. Massart, "The mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1–18, 2000.

[303] M. Rizal, J. Ghani, M. Nuawi, and C. Haron, "Cutting tool wear classification and detection using multi-sensor signals and mahalanobis-taguchi system," *Wear*, vol. 376-377, pp. 1759–1765, 2017.

[304] S. Binsaeid, S. Asfour, S. Cho, and A. Onar, "Machine ensemble approach for simultaneous detection of transient and gradual abnormalities in end milling using multisensor fusion," *Journal of Materials Processing Technology*, vol. 209, no. 10, pp. 4728–4738, 2009.

[305] W. H. Wang, G. S. Hong, Y. S. Wong, and K. P. Zhu, "Sensor fusion for online tool condition monitoring in milling," *International Journal of Production Research*, vol. 45, no. 21, pp. 5095–5116, 2007.

[306] E. Kannatey-Asibu, J. Yum, and T. Kim, "Monitoring tool wear using classifier fusion," *Mechanical Systems and Signal Processing*, vol. 85, pp. 651–661, 2017.

[307] A. E2075/E2075M-15, "Standard practice for verifying the consistency of ae-sensor response using an acrylic rod," American Society for Testing and Materials West Conshohocken, PA, 2010.

[308] B. Arvinti-Costache, M. Costache, C. Nafornita, A. Isar, R. Stolz, and H. Toepfer, "A wavelet based baseline drift correction method for fetal magnetocardiograms," in *2011 IEEE 9th International New Circuits and systems conference*, IEEE, 2011.

[309] T. Childs, K. Maekawa, T. Obikawa, and Y. Yamane, *Metal Machining*. Elsevier, 2000.

[310] M. Riedl, A. Müller, and N. Wessel, "Practical considerations of permutation entropy," *The European Physical Journal Special Topics*, vol. 222, no. 2, pp. 249–262, 2013.

[311] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[312] F. Khasawneh, A. Otto, and M. Yesilli, "Turning dataset for chatter diagnosis using machine learning." Mendeley Data, v1. http://dx.doi.org/10.17632/hvm4wh3jzx.1, 2019.

[313] T. Insperger, B. P. Mann, T. Surmann, and G. Stepan, "On the chatter frequencies of milling processes with runout," *International Journal of Machine Tools and Manufacture*, vol. 48, no. 10, pp. 1081 – 1089, 2008.

[314] R. E. W. Dombovari, Zoltan and G. Stepan, "Estimates of the bistable region in metal cutting," *Proceddings of the Royal Society A*, vol. 464, no. 2100, pp. 3255–3271, 2008.

[315] Z. Dombovari and G. Stepan, "On the bistable zone of milling processes," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 373, no. 2051, p. 20140409, 2015.

[316] Y. Yan, J. Xu, and M. Wiercigroch, "Basins of attraction of the bistable region of time-delayed cutting dynamics," *Phys. Rev. E*, vol. 96, p. 032205, 2017.

[317] B. P. Mann, K. A. Young, T. L. Schmitz, and D. N. Dilley, "Simultaneous stability and surface location error predictions in milling," *Journal of Manufacturing Science and Engineering*, vol. 127, pp. 446–453, 2005.