國立台灣大學工學院機械工程研究所

碩士論文

Graduate Institute of Mechanical Engineering

College of Engineering

National Taiwan University

Master Thesis

應用於顫振辨識之特徵選擇與分類方法之研究

A study of feature selection and classification methods for chatter identification models

陳宇軒

Yu-Hsuan Chen

指導教授:劉建豪 博士

Advisor: Chien-Hao Liu, Ph.D.

中華民國 108 年 6 月

June, 2019

doi:10.6342/NTU201903636

國立臺灣大學碩士學位論文
口試委員會審定書

應用於顫振辨識之特徵選擇與分類方法之研究

A study of feature selection and classification methods for chatter identification models

本論文係陳宇軒君（學號 R06522519）在國立臺灣大學機械工程學系完成之碩士學位論文，於民國 108 年 6 月 28 日承下列考試委員審查通過及口試及格，特此證明

口試委員：　劉建豪　（簽名）

（指導教授）

施文彬　　蔡孟勳

蔡曜陽

系主任　黃美嬌　（簽名）

誌謝

首先要感謝指導教授劉建豪老師，在碩士班這兩年給我的指導與支持，並在研究上遇到困難時幫忙找資源協助我，讓我在實驗室這兩年學習到許多做研究的方法。也感謝施文彬老師，在我碩一時提供了許多研究方向的建議與指教，使我受益良多。口試委員蔡孟勳教授及蔡曜陽教授也提供了許多專業的見解，對這篇論文的研究提供了寶貴的意見與回饋，感謝你們的參與。

這篇研究可以順利完成，一部分要感謝泳辰學長的建議以及當初建立的研究方向。傑程與胤瑄平時提出的問題與想法也幫助我更仔細的思考研究上遇到的一些問題。感謝實驗室的夥伴們，星宇，善謙，建章，尚軒，家倫，炫奇，健宏，柏文，碩志，晉祥，永珊，均鴻，謝謝你們的幫忙與陪伴。也感謝家人一路上的支持，讓我可以順利完成這個學位。

doi:10.6342/NTU201903636

中文摘要

維持高產率對於銑削加工的效率而言十分重要。顫振是加工時發生的一種自激式振動，在實務中限制了產率。過去的研究提出了許多顫振偵測的方法，利用各種訊號處理的方法如快速傅立葉轉換(FFT)，小波包轉換(WPT)，及希爾伯特-黃轉換(HHT)。許多資料分類演算法也被應用於顫振偵測。雖然顫振偵測的領域已有許多文獻，我們仍不清楚何種方法可以達到較佳的正確率與偵測速率。

在本研究中，我們將測試多種訊號處理方法以及資料分類的演算法，使用的資料集中包含各種主軸轉速及切深。我們結合多種訊號處理方法及分類演算法，開發了一個顫振辨識平台以建立分類模型並評估其性能。資料分類方法包含了固定的閾值，最近鄰居法(k-NN)，單純貝氏分類器，支持向量機(SVM)，局部密度因子(LOF)，以及類神經網路。以分類精準度而言，結果顯示最近鄰居法搭配小波包轉換及希爾伯特-黃轉換是最佳的方法，誤判率僅 2.2%。

關鍵字：顫振，小波包轉換，希爾伯特-黃轉換，最近鄰居法，支持向量機

doi:10.6342/NTU201903636

Abstract

Maintaining high production yield is important for efficiency in the milling process. Chatter is a type of self-excited vibration that can occur during machining, and limits the production yield in practice. In the past, many chatter detection methods were proposed using different signal processing methods such as Fast Fourier transform (FFT), wavelet packet transform (WPT), and Hilbert-Huang transform (HHT). Several classification methods were also applied in chatter detection. Despite the large amount of researches regarding chatter detection, it is unclear which of these proposed methods are better in terms of accuracy and detection speed.

In this research, we test the signal processing methods and classification algorithms against the entire dataset, with a wide range of spindle speeds and depth of cuts. A chatter identification platform is developed to train models and evaluate their performance, using combinations of signal processing methods and classification algorithms. The classification methods include numerical threshold, k-nearest neighbors (K-NN), Naïve Bayes, support vector machine (SVM), local outlier factor (LOF), and artificial neural network. K-NN proves to be the optimal method when using WPT and HHT for signal processing, with an error rate of 2.2%.

Keywords: chatter, wavelet packet transform, Hilbert-Huang transform, k-nearest neighbor, support vector machine

doi:10.6342/NTU201903636

Table of Contents

doi:10.6342/NTU201903636

doi:10.6342/NTU201903636

List of figures

doi:10.6342/NTU201903636

doi:10.6342/NTU201903636

doi:10.6342/NTU201903636

List of tables

doi:10.6342/NTU201903636

List of abbreviations

MA	Missing alarm rate
FA	False alarm rate
ER	Error rate

doi:10.6342/NTU201903636

1. Introduction

1.1 High-speed milling and chatter

Production yield is important for CNC machines. One way to increase production yield is by increasing the spindle speed in the milling process. By increasing the spindle speed, the material removal rate is increased proportionally, so is the production yield [1].

However, by increasing the cutting speed, new problem arises. Chatter is a phenomenon caused by the mechanical interactions between the cutting tool and the workpiece. This self-excited vibration [2] causes significant issues in machining, causing large vibrations, which limits productivity and may produce poor surface finish on the workpiece [3] [4]. Fig. 1 shows a comparison of surface finishes between chatter and chatter-free cutting. Fig. 2 illustrates the effect of chip thickness in chatter development. Vibrations in the cutting process causes variations in chip thickness, and certain combination of spindle speed, feed rate, material, and cutting tool causes the chip thickness to drastically fluctuate like Fig. 2 (c). This is the origin of chatter.

The cause of chatter and its mechanical models are well-studied. The cutting process can be described with as a non-linear system [5], and a stability lobe diagram (SLD) was used to indicate which cutting conditions cause chatter [6]. Fourier series was used to obtain an analytical description of SLDs [7], and was later verified experimentally [8].

doi:10.6342/NTU201903636

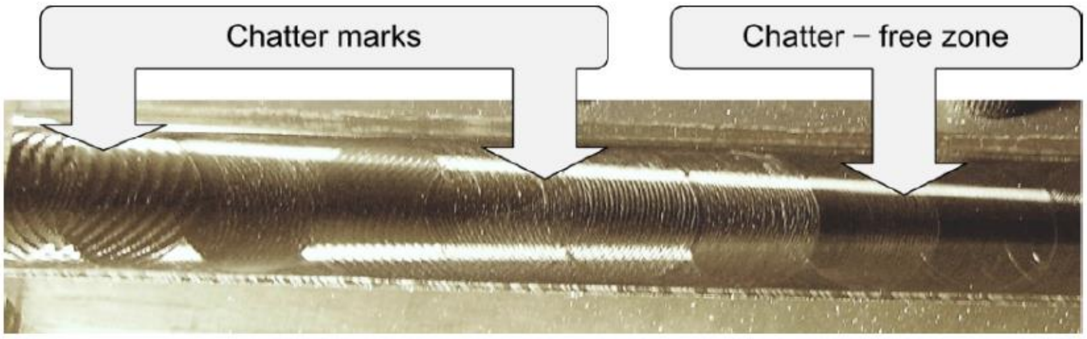


Fig. 1. Chatter leaving undesired marks on the surface of the workpiece [9]

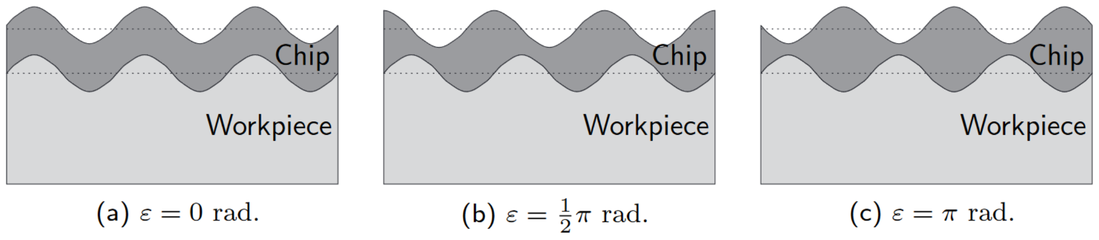

Chip
Workpiece
(a) $\varepsilon = 0$ rad.
Chip
Workpiece
(b) $\varepsilon = \frac{1}{2}\pi$ rad.
Chip
Workpiece
(c) $\varepsilon = \pi$ rad.

Fig. 2. Illustration of how chatter develops due to abrupt change in chip thickness [2]

Many methods were proposed to calculate the SLD, including a previous research using transfer functions [10], and a method using multi-frequency solution [11]. Semi-discretization method was applied to solve the non-linear delayed differential equations describing chatter stability [12], and had been verified as accurate approach for SLD computations [13]. Fig. 3 shows a SLD with spindle speed on the *x*-axis and depth of cut on the *y*-axis. The region above the black curve is the chatter region, i.e. any cutting condition above the black curve is unstable.

doi:10.6342/NTU201903636

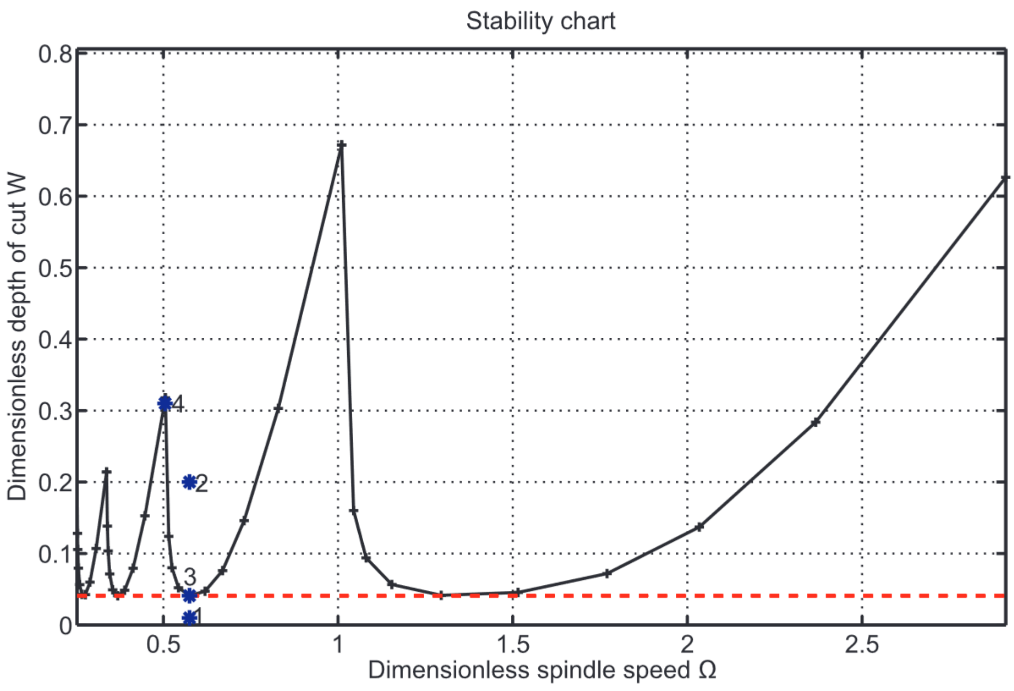


Fig. 3. An example of a stability lobe diagram [14]

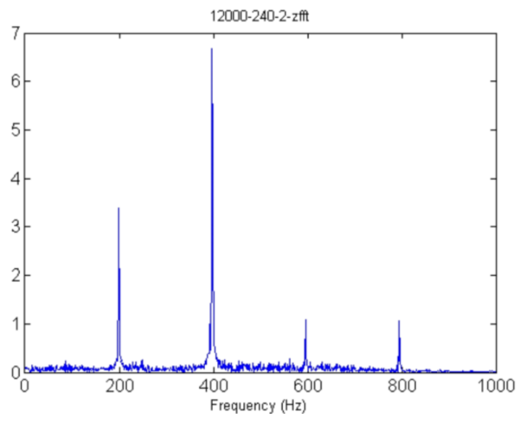
1.2 Chatter detection

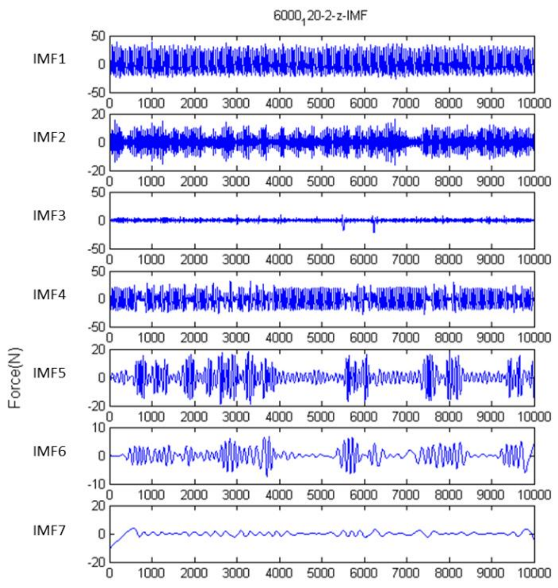
The most straightforward way to avoid chatter is to obtain the SLD [15] [16] [17], and avoid cutting in the unstable range. However, several parameters have effects on the SLD, including the vibration modes of the CNC machine, the cutting tool, the material of the workpiece, and the wear of the cutting tool. In addition, cutting force signal is required to calculate the SLD, which typically requires a dynamometer. In this research, we utilize chatter detection methods that do not require a dynamometer.

In the past, many chatter detection methods were proposed using different signal processing methods. FFT of vibration signals were used to calculate the optimal cutting path [18]. FFT can also be calculated every 16 samples for quick detection [19]. Wavelet transform is a signal processing method that was also applied to chatter recognition [20]. Wavelet packet transform (WPT) is an extension of wavelet transform to get better

doi:10.6342/NTU201903636

frequency resolution at certain frequency ranges, and was used in several previous works [21] [22] [23] [24] [25] [26] [27]. R-value was also used to monitor chatter by measuring the spindle drive current [28] [29]. Time domain cutting force signal [30], or its power spectrum density [31] can also be utilized. Hilbert-Huang transform has also been proven effective [32] [33] [34] [35] [36]. Fig. 4 (a) shows an example of FFT spectrum, and Fig. 4 (b) is the intrinsic mode functions (IMFs) decomposed from the time-domain vibration signals. Machine vision was also applied in combination with short-time Fourier transform (STFT) [37], or texture analysis using neural networks [38] [39].





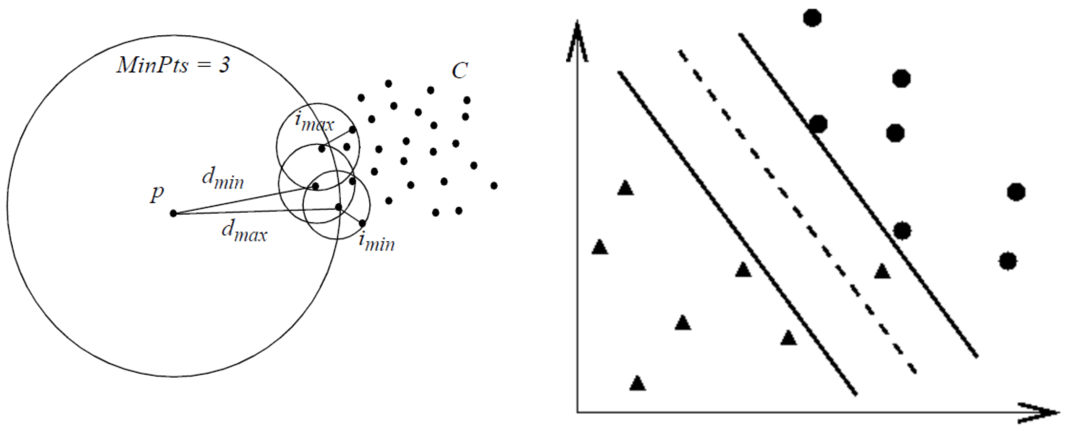
(a) (b)

Fig. 4. (a) Spectrum of the vibration signal after FFT, (b) Intrinsic mode functions (IMFs) obtained from Hilbert-Huang transform *[40]*

After features are extracted with one of the signal processing methods, some researches set a fixed threshold as the boundary of chatter (unstable) and non-chatter (stable) signals [21] [31] [41] [42] [43]. A classification algorithm may be used to train a model to classify unstable and stable data for better accuracy. Well-known classification

doi:10.6342/NTU201903636

algorithms such as support-vector machine (SVM) [21] [44] [45] [46], k-means [31], local outlier factor (LOF) [47], and artificial neural networks [41] [42] [43] were used in the past and were proven effective. Fig. 5 (a) and (b) shows examples of dataset classified with LOF and SVM, respectively.



(a) (b)

Fig. 5. Illustrations of LOF and SVM

(a) Each data point is assigned a local outlier factor (LOF) and outliers of the dataset can be identified *[48]*. (b) An illustration of data classified using support vector machine (SVM). The solid lines are support vectors separating the two classes – triangle and circle *[49]*.

1.3 Aim of this research

Despite the large variety of existing chatter identification methods, there are two main issues. The first is that in most researches, the dataset used to validate the proposed method is small, usually consisting of less than 10 cuts. Comprehensive validation was done only in rare cases, e.g. Zhehe Yao, et al. tested their detection method with a dataset

doi:10.6342/NTU201903636

consisting of 45 cuts [21]. Therefore, in most cases, the reader cannot obtain the actual accuracy of the given method, and it is near impossible to compare the effectiveness of different methods. For example, many researches claims that wavelet transform [50] [51], wavelet packet transform [52], or Hilbert-Huang transform [53] is a superior method compared to fast Fourier transform for chatter or machine fault identifications. However, the claims are usually based on a theoretical or empirical argument with little or no statistical evidence provided. We aim to resolve this issue by comparing different signal processing methods using the same dataset and common parameters such as window size. In fact, as will be shown in chapter 5, some of our findings are completely opposite to such popular claims.

The second issue is that, even if a chatter identification method is tested on a large dataset, and the accuracy is available for comparison, it is unfair to compare the accuracy of two methods from different research teams. This is because the datasets used for validation are different, and some datasets probably consists many data at the boundary of stable and unstable region, and is thus more difficult to classify correctly.

With the rise of industry 4.0, the availability of large amount of data from manufacturing processes should be utilized to help training models in order to improve chatter detection accuracy. In this research, we will take advantage of the cutting data to truly test the signal processing methods and classification algorithms against the entire dataset, with spindle speed ranging from 4500 to 7000 rpm, and depth of cut from 0.2 to 1.0 mm. We believe this approach can help developing a standard procedure to train a model, and evaluate the true accuracy of the model in a fair way.

doi:10.6342/NTU201903636

1.4 Structure of the thesis

This thesis consists of two main topics: feature extraction and classification algorithms. Chapter 2 briefly introduces the signal processing methods that will be compared in this research. Each signal processing method may generate one or more feature(s), and will be further processed by one of the classification algorithms discussed in chapter 3. Chapter 2 and 3 will mainly focus on the concepts, and the implementation details will be described in chapter 4, which is focused on the software implementation and optimizations of some of the algorithms.

Chapter 5 summarizes the results. Data collection procedure for the dataset used in this research is explained in detail. Then, the classification algorithms are compared when using the same feature extraction method. Since there are many parameters involved for each signal processing method, their parameters will be optimized. After optimization is completed within each signal processing method, all of the methods will be compared. The amount of combination is large, because, for our model training platform developed in this research, any extracted feature can be combined with any classification algorithm. Finally, since both the error rate and detection speed are critical for chatter detection, we will discuss how they are affected by window size, and the tradeoff involved.

Chapter 6 is the conclusion and we point out the potential direction for future researches. There is an appendix showing all the results from different models we trained which should make comparison easier.

doi:10.6342/NTU201903636

2. Signal processing methods and feature extraction

Four signal processing methods are used to in this research to extract features from the vibration signals in our dataset. They include Fast Fourier transform (FFT), wavelet packet transform (WPT), autocorrelation coefficient, and Hilbert-Huang transform (HHT).

Since chatter is a phenomenon that can be identified via the vibrations at chatter frequencies, it may be desirable to observe the vibration characteristics in the frequency domain. FFT and WPT are such algorithms, which are widely used in many chatter identification researches. Autocorrelation coefficient and HHT help us to look at the problem from time-domain. Roughly speaking, the former calculates the periodicity of the signal whereas the latter decomposes the signal into several intrinsic mode functions (IMFs) in a specific way.

2.1 Fast Fourier transform (FFT)

FFT is a popular method for chatter detection [54] [55]. FFT is an implementation of the well-understood Fourier transform, which transform discrete time domain data to frequency domain. The time complexity of N-point FFT is $O(NlogN)$, which makes it quicker than WPT, autocorrelation coefficients, and HHT with our implementation. The high performance, great theoretical foundations, and easy-to-interpret results make FFT a great candidate for chatter detection. Fig. 6 shows the spectrum after applying FFT on sound signal. The circles indicate the peaks at tooth pass frequencies and the asterisks indicate peaks at spindle speed frequencies. FFT is a power tool for chatter recognition due to its fast computation time,

doi:10.6342/NTU201903636

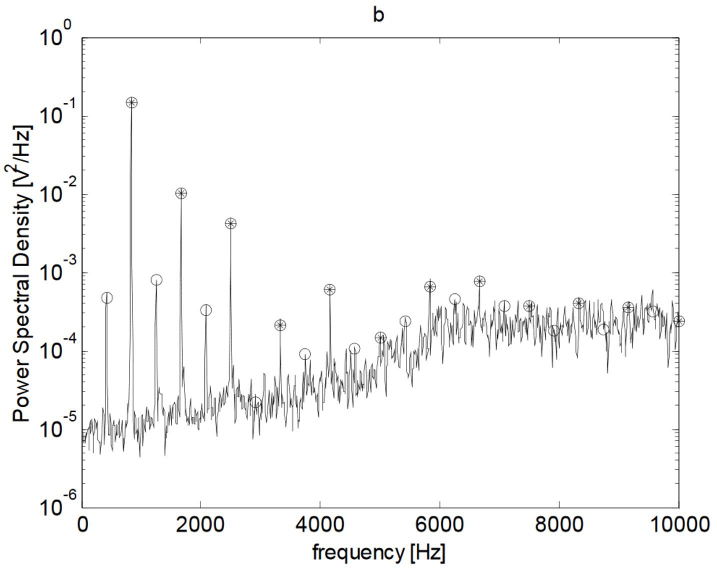


Fig. 6. Power spectrum density (PSD) obtained by applying FFT on the sound signal during cutting. *[56]*

Several features can be extracted from the spectrum obtained with FFT, the following ones will be investigated in this research:

(1) $E_{1,FFT}$: Relative energy between a frequency band and the entire spectrum. The energy in a certain frequency band is defined as

$$E_{f_{min},f_{max}} = \sum_{f_{min} \leq f \leq f_{max}} |s(f)|^p, \tag{1}$$

where $|s(f)|$ is the magnitude of spectrum at frequency *f*, and the exponent *p* is an adjustable parameter. The relative energy for an frequency band $[f_{min}, f_{max}]$ is defined as

$$E_{1,FFT} = \frac{E_{f_{min},f_{max}}}{E_{0,f_s/2}}, \tag{2}$$

doi:10.6342/NTU201903636

where f_s is the sampling rate. Therefore, $E_{0,f_s/2}$ is the energy of the entire frequency range after FFT.

In practice, f_{min} and f_{max} are chosen to include the dominant chatter frequency. Therefore, a higher $E_{1,FFT}$ indicates a higher chance of chatter occurring. Peaks at tooth pass frequencies should be filtered before calculating $E_{1,FFT}$ so that those normal peaks are not incorrectly considered as an indication of chatter. The DC component, i.e. amplitude at 0 Hz, is also set to zero.

In addition, it's also possible to calculate $E_{1,FFT}$ by ignoring all but the n highest peaks in the spectrum. The results will be discussed in chapter 5.

(2) $E_{2,FFT}$: Relative energy between the chatter frequency band and tooth pass frequencies. First, the spectrum is obtained from the vibration signal. Then, n peaks are chosen using the implementation described in section 4.2. Finally, the calculation of relative energy is performed similarly to $E_{1,FFT}$.

(3) $E_{3,FFT}$: Relative energy between non-tooth pass frequencies and the entire spectrum. The calculation steps are similar to $E_{2,FFT}$.

(4) Magnitude and phase of the spectrum: It is possible to recognize chatter from the spectrum of the vibration signal due to the rise of chatter peak. However, there are other spectral changes when chatter occurs, which are not easily identifiable visually. Nevertheless, we can facilitate artificial neural networks (ANNs) and use the magnitude or phase of the spectrum as input, to obtain a classification model.

FFT can be used with a sliding window to calculate the spectrum at fixed intervals as shown in Fig. 7. There are two parameters – window size and overlap. The window size describes how much data is taken to perform FFT, and the overlap is the amount of time

doi:10.6342/NTU201903636

that two consecutive windows intersects. We will use this concept throughout this article, and applying it to other signal processing methods, such as WPT and HHT.

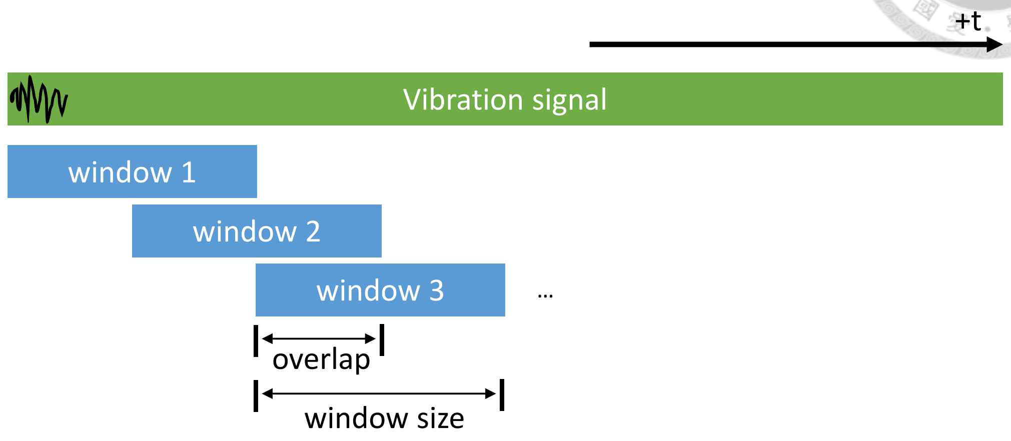


Fig. 7. Illustration of window and overlap

2.2 Wavelet packet transform (WPT)

Despite the high performance and popularity of FFT, it only decomposes a function into sines and cosines. Wavelet transform was developed to decompose a function into other sets of functions called mother wavelets. There are many families of mother wavelets, such as *Daubechies* (*db*), Haar, and *dmey*. Within each family, there can be one or more types of mother wavelets. For example, *Daubechies* (*db*) family includes *db1*, *db2*, *db3*, etc. Fig. 8 shows three families of mother wavelets.

A potential advantage of using wavelet transform comes from Gabor uncertainty principle, which shows that in a frequency analysis, the product of time resolution and frequency resolution is never smaller than a constant [57]. Wavelet transform sacrifices frequency resolution in favor of better time resolution compared to FFT. A better time resolution means chatter can be possibly detected earlier. However, it should be noted that

doi:10.6342/NTU201903636

similar effects can be acquired by using FFT with a sliding window, which is called short-time Fourier transform (STFT).

Wavelet transform splits the frequency domain into several bands as illustrated in Fig. 9. Since chatter occurs at certain frequencies, we can pick the band that contains the dominant chatter frequency, and calculate the vibration energy within that band. Intuitively, when the energy in the chatter frequency band is high, the probability of chatter occurring is high as well. This concept will be used in our chatter detection strategy.

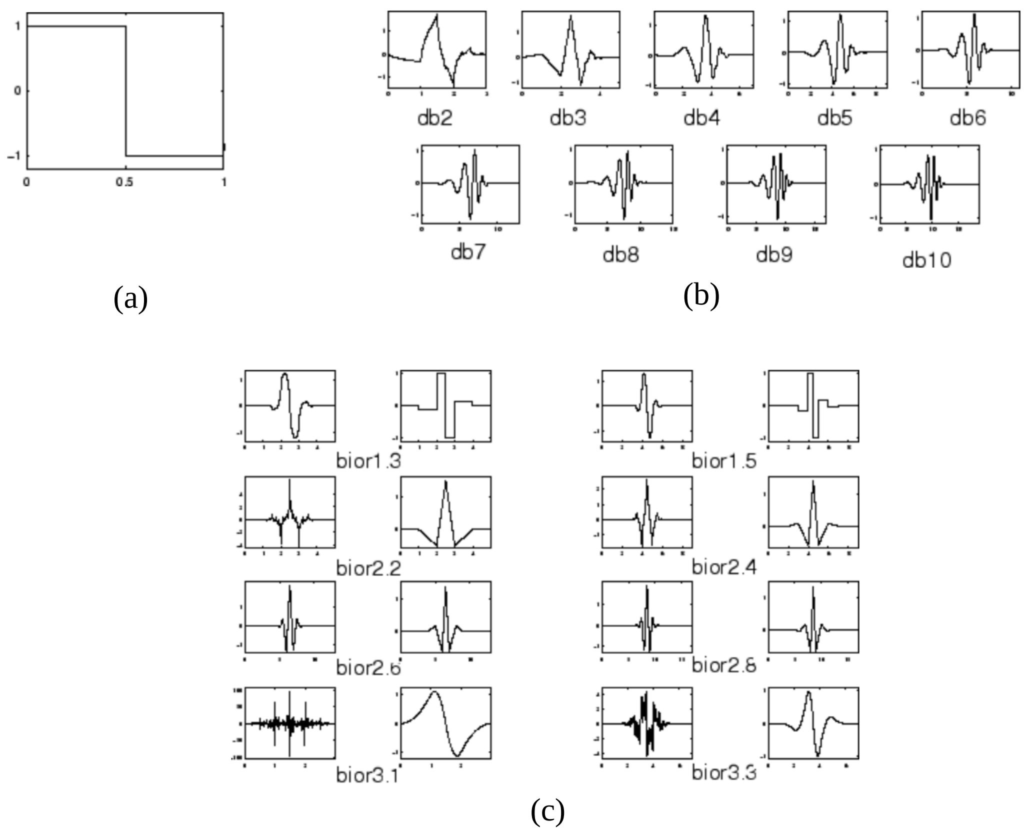


Fig. 8. Various families of mother wavelets, including (a) Haar, (b) Daubechies (*db*), and (c) biorthogonal (*bior*). *[58]*

doi:10.6342/NTU201903636

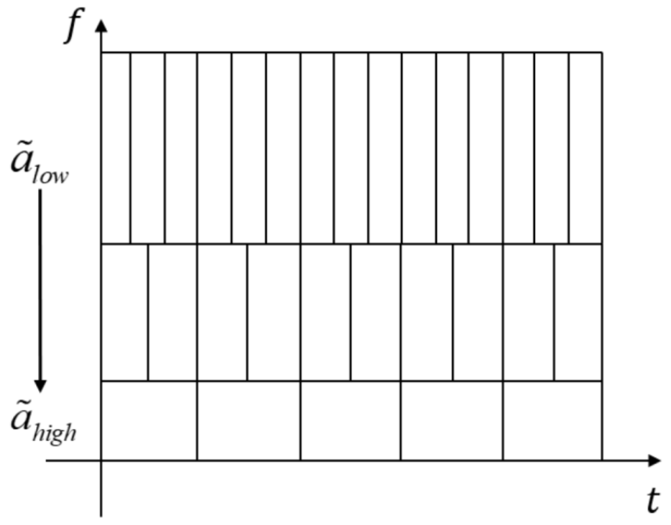


Fig. 9. Wavelet transform produces lower frequency resolution at high frequencies

In addition to wavelet transform (WT), there is a modified version called wavelet packet transform (WPT). The frequency resolution at high frequencies is lower than the low frequencies for WT as shown in Fig. 9. WPT uses a series of low-pass and high-pass filters to divide the signal into two parts as illustrated in Fig. 10. By doing this dividing process repeatedly for n times, we get a wavelet packet tree of n levels. Each level consists of nodes of coefficients, representing the vibration amplitude at its corresponding frequency band. Since the wavelet packet tree is symmetric, the frequency resolution at all frequencies are equal, which is more ideal than WT. Therefore, we use WPT in research. The feature we extract from the signal is the relative energy

$$E_{WPT} = \frac{E_{chatter}}{E_{all}}, \tag{3}$$

where $E_{chatter} = |c_{chatter}|^p$ and $E_{all} = \sum_i |c_i|^p$. Here $c_{chatter}$ is the wavelet coefficient of the chatter frequency band and c_i refers to the i-th wavelet coefficient at n-th level. p and n are both adjustable. In this research, we choose $n = 5$, which splits the frequency

doi:10.6342/NTU201903636

domain (0 to 5000 Hz) into $2^5 = 32$ bands. From experiment, the dominant chatter frequency is in the 6th band, which is between 781.25 and 937.25 Hz.

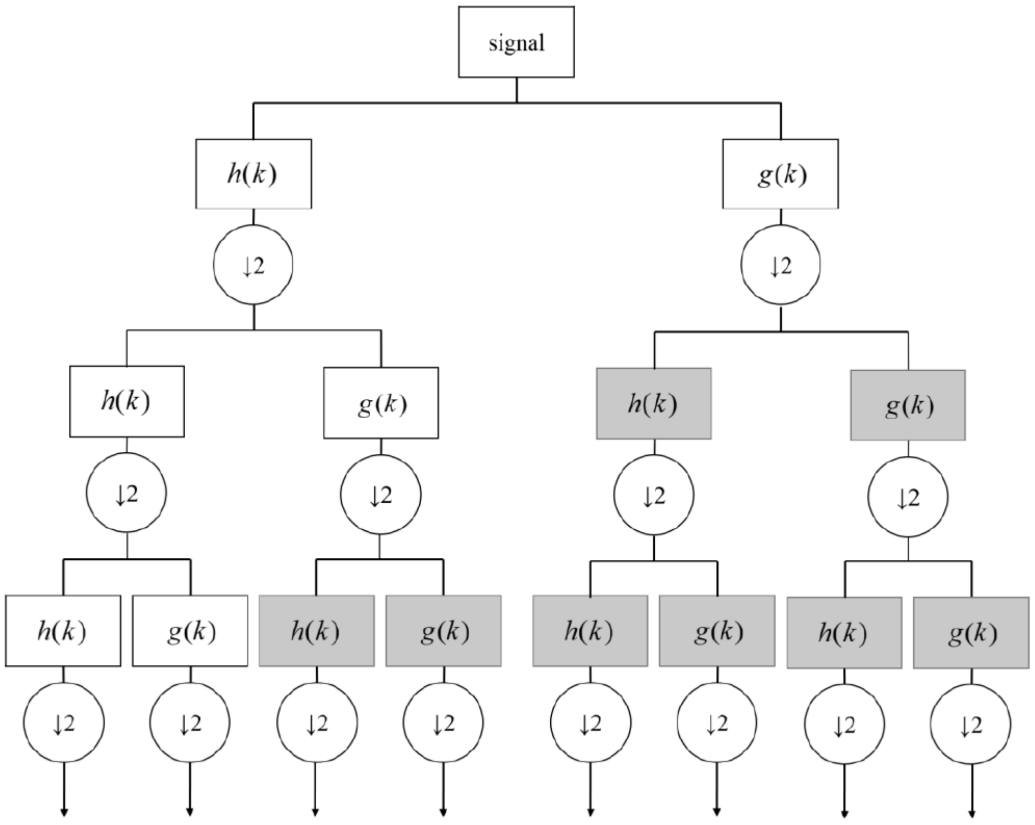


Fig. 10. Wavelet packet transform and its corresponding coefficient tree for each frequency bands [59]

doi:10.6342/NTU201903636

2.3 Autocorrelation coefficients

Autocorrelation coefficient is an indicator of the periodicity of the signal. Given a series of displacements *x*(*n*), define autocorrelation function [60]

$$Rxx(m)=\frac{1}{N}\sum_{n=0}^{N-1}x(n)x(n+m) \tag{4}$$

After standardization, we get the autocorrelation coefficient

$$R'_{xx}(m)=\frac{\frac{1}{N}\sum_{n=0}^{N-1}x'(n)x'(n+m)}{\sqrt{\frac{1}{N}\sum_{n=0}^{N-1}[x'(n)]^2}\sqrt{\frac{1}{N}\sum_{n=0}^{N-1}[x'(n+m)]^2}}, \tag{5}$$

where

$$x'(k)=x(k)-\frac{1}{N}\sum_{n=0}^{N-1}x(n).$$

For example, for a perfectly periodic signal, $x(n)=sin(2\pi\cdot 0.01n)$ for $1\le n\le N$, if we choose *m* so that it is identical to the period of the signal, i.e. *m* = 100, then $Rxx'(m)=1$ by (5). On the other hand, if *m* is equal to half of the period, i.e. *m* = 50, then $Rxx'(m)=-1$.

The motivation of using autocorrelation coefficient comes from the fact that stable cutting signal is close to periodic, with period being the inverse of tooth pass frequency [60]. In contrast, during unstable cutting, some chatter frequencies arise at the natural frequencies of the spindle, and some arises above and below the tooth pass frequencies. This can be seen from the vibration model of the milling process [5] [10] [61] [62], and this property allows us to identify chatter with autocorrelation coefficients.

The steps to calculate autocorrelation coefficient is as follows:

doi:10.6342/NTU201903636

1. Take a segment of vibration signal, which has to be at least as long as twice the tooth pass period *T_1*. This vibration signal can be of any form, e.g. acceleration, velocity, or displacement. Call this signal *x*, and its length *T*.
2. Shift the signal in time by τ. Let the sampling rate be f_s, then $m = f_s\tau$. Repeat this step for $0 \leq \tau \leq T$.
3. Since $m \propto \tau$, we can write autocorrelation coefficient $R'_{xx}(m)$ as $R'_{xx}(\tau)$. Calculate the $R'_{xx}(\tau)$ for each τ, where $0 \leq \tau \leq T$.
4. Find the time between peaks in $R'_{xx}(\tau)$, call this T_X. This should be identical to tooth pass period in stable cutting but not unstable cutting.
5. In stable cutting, T_1 should be an integer multiple of T_X because the vibration signal for every revolution of the spindle should be similar. Find the remainder of T_1 divided by T_X, call this ε. When ε is very close to 0, we say the cutting condition is stable. Otherwise, it is unstable.

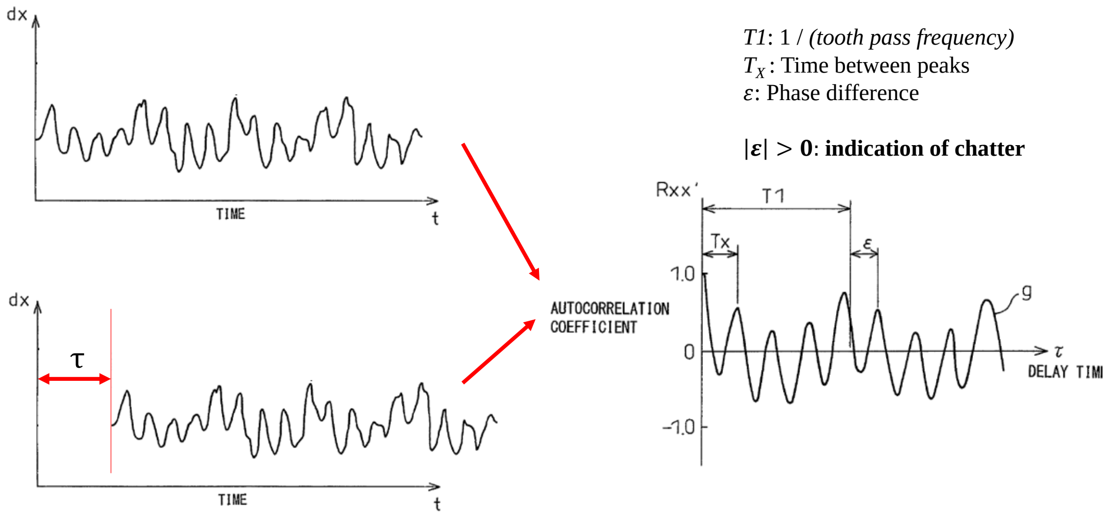

dx
TIME
t
dx
τ
TIME
t
AUTOCORRELATION COEFFICIENT
T1: 1 / (tooth pass frequency)
T_X: Time between peaks
ε: Phase difference
$|\varepsilon| > 0$: indication of chatter
Rxx'
T1
Tx
ε
1.0
0
-1.0
g
τ
DELAY TIME

Fig. 11. Illustration of the concept of autocorrelation coefficient [60]

doi:10.6342/NTU201903636

Step 4 above involves peak finding, which is non-trivial and the implementation is outlined in section 4.2. Calculation of ε is required in step 5, which requires the concepts of prominence of a peak discussed in section 4.2. The procedure is as follows:

1. Let the spindle rotation period be T_0. Find all peaks in $[0, 2T_0]$.
2. For $[0, T_0]$ and $[T_0, 2T_0]$, select peaks with top x% prominence in each interval, for some $x \in [0,100]$.
3. Among the peaks found in step 2, choose the peak just before $T_{0,}$ and the one just after T_0. Name them C and D, and let their corresponding times be T_C and T_D.
4. Let the minimum value between C and D be $y_{min} = \max_{T_C \le t \le T_D} y(t)$.
5. Let $y'(T_0) = y'(T_0) - y_{min},\ y'_C = y_C - y_{min}$, and $y'_D = y_D - y_{min}$.
6. Calculate

$$\phi = cos^{-1}\left(\frac{4y'(T_0)}{y'_C + y'_D} - 1\right), \tag{6}$$

For a cosine wave, $y = cos(\omega t)$, the above equation gives

$$\phi(T_0) = cos^{-1}\left(\frac{4[cos(\omega T_0) - 1]}{2 + 2} - 1\right) = cos^{-1}\big(cos(\omega T_0)\big).$$

Therefore, $cos\big(\phi(T_0)\big) = cos(\omega t)$ as expected.

In practice, the autocorrelation coefficient may differ significantly from one spindle rotation to another. If we use the result from one period to predict whether the condition is stable or unstable, incorrect prediction can occur frequently as the autocorrelation coefficient fluctuates. Therefore, we split the input signal into windows, which is much longer than spindle speed period, but small enough for chatter detection purposes, e.g. 0.2 seconds. This can reduce classification error rate. For each window, phase differences ε

doi:10.6342/NTU201903636

is calculated for every two spindle speed periods. The extracted feature is the average of ε in this window.

A theoretical advantage of using autocorrelation coefficients is that this is applied in time domain. A frequency-domain signal processing methods involves gathering sufficiently long signal before transforming into frequency domain to ensure good frequency resolution. For example, at a sampling rate of 10 kHz, FFT needs approximately 0.1 seconds of data to achieve a frequency resolution of 5 Hz, which is required because of the proximity of chatter frequency and tooth pass frequency on our CNC machine. On the other hand, autocorrelation coefficient method requires only a spindle rotation period to perform a calculation, which is only 0.02 seconds if the spindle speed is 6000 rpm and $T = 2T_1$. Since we use windows, the window size may be adjusted to exploit this theoretical advantage. This was tested and the results are shown in chapter 5.

2.4 Hilbert-Huang transform (HHT)

Hilbert-Huang transform is a signal processing method that is designed to analyze non-stationary and non-periodic signals [63], and has been applied various fields of study such as in medical, geophysics, and structure safety analysis [64]. HHT involves two steps:

1. Apply empirical mode decomposition (EMD) [65]. This decompose the input signal into the sum of several time-domain functions, called intrinsic mode functions (IMFs). A process called sifting is used to find the local maxima and minima for the signal and decompose it according to a set of rules. The number of IMFs a signal is decomposed into is determined by the signal itself. Contrary

doi:10.6342/NTU201903636

to sine and cosine functions used in Fourier transform, IMFs can vary in both time and frequency.

2. Apply Hilbert transform to each of the IMFs. Hilbert transform is defined by [66]

$$H(u)(t) = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{u(\tau)}{t-\tau} d\tau,$$

which has the effect of shifting the phase of the negative frequencies of $u(t)$ by $\frac{\pi}{2}$ and the phase of the positive frequencies by $-\frac{\pi}{2}$.

Fig. 12 shows the 11 IMFs that are decomposed from a cutting signal. Fig. 13 shows that Hilbert transform extracts the envelope of the signal, as shown in the red dotted curve. We can define $E_{HHT} = |x_n(t)/x(t)|^2$ as the energy for the HHT result, where $x_n(t)$ is the n-th IMF and $x(t)$ is the original vibration signal. Fig. 14 is the time-frequency visualization of the spectrum obtained using HHT. It's also possible to extract the chatter frequency band by applying WPT to the input signal, take the wavelet coefficients from the chatter frequency band, and apply inverse WPT. Then the signal can be processed with HHT to get the energy of each IMF. This approach has been shown to be more effective than using HHT alone [32]. In this research, we set $n = 1$, which implies E_{HHT} is high when the first IMF has large amplitudes.

doi:10.6342/NTU201903636

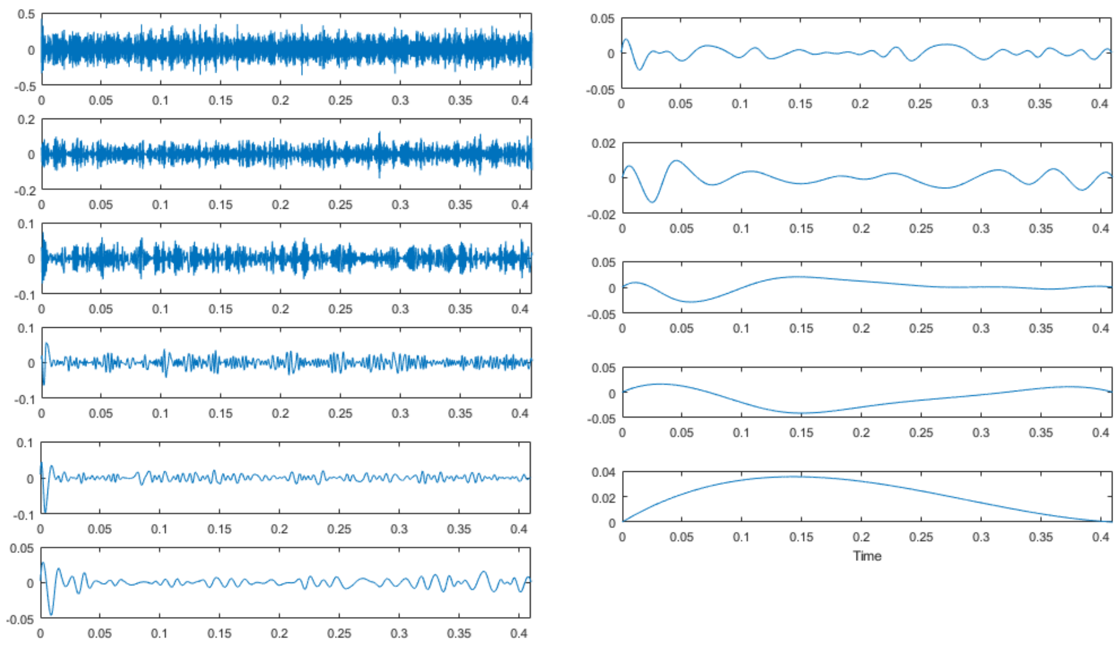


Fig. 12. IMFs of a cutting signal obtained using HHT

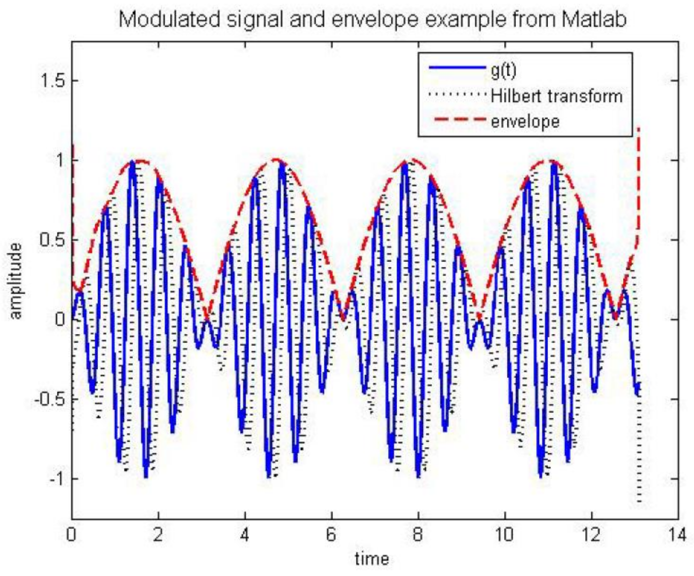


Fig. 13. Illustration of Hilbert transform on vibration signal *[67]*

doi:10.6342/NTU201903636

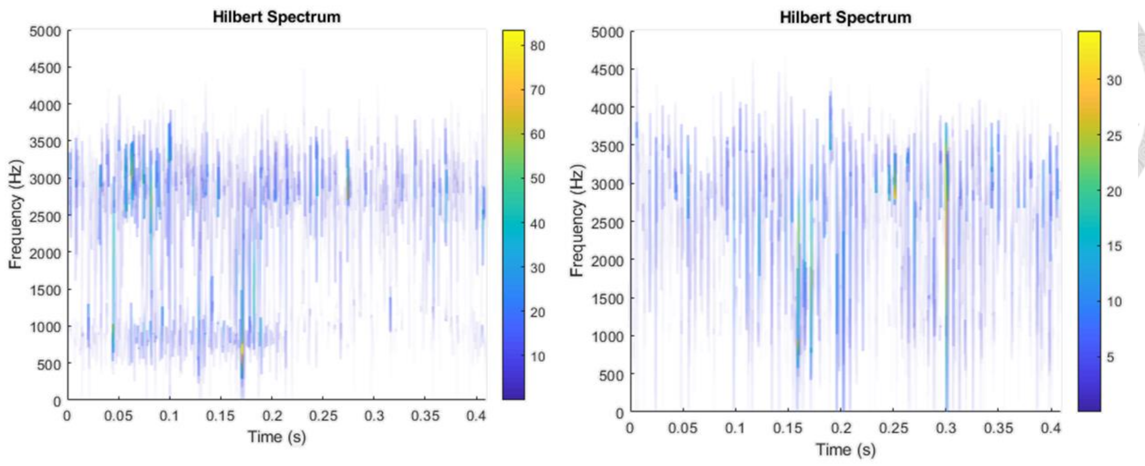


Fig. 14. Hilbert spectrum of unstable (left) and stable (right) cutting signals calculated and visualized with MATLAB

doi:10.6342/NTU201903636

3. Classification algorithms

After we obtain the feature vectors using the signal processing methods described in the previous chapter, they will be put into a classification algorithm to train a model. The model will be able to predict whether a signal belongs to stable or unstable. Since our dataset is labeled, we mainly focused on supervised learning. In this chapter, several methods and classification algorithms are used, and their effectiveness are compared.

The simplest method is to set a fixed threshold value to distinguish stable data from the unstable ones. A statistical method – Naïve Bayes was tested in 3.2. Some commonly used classification algorithms, such as local outlier factor (LOF), support vector machine (SVM), and k-nearest neighbor (k-NN), were applied to our dataset. Finally, we utilized artificial neural network (ANN) to obtain models for prediction.

3.1 Numerical threshold

Using a fixed threshold as the boundary of stable and unstable region is one of the simplest and most straightforward method. This method relies on the distribution of a feature to be separable by a single threshold. The calculation steps are as follows:

1. Choose a feature that is represented by a single numerical value, e.g. the relative energy after FFT of the *x*-axis signal.
2. Compute the values of this feature for the entire dataset to obtain the distribution of both stable and unstable data.
3. Let there be *m* stable data points and *n* unstable data points. Since relative energy is larger for unstable data points, choose the *m*-th smallest value in these $m + n$ data points. This is the selected threshold. A data point is considered stable if and only if its value is below this threshold.

doi:10.6342/NTU201903636

3.2 Naïve Bayes

Naïve Bayes is a probabilistic classifier based on Bayes' theorem, widely used in machine learning [68] [69] [70]. The general idea is as follows. Given a set of features, such as the energy ratios of *x*-, *y*-, and *z*-axis, we know the distribution of each feature for both unstable and stable data. Then, we fit the distribution curve with, e.g. Gaussian distribution. Now, given a new data point, we can estimate its probability of being unstable based on the fitted distribution and its energy ratio from *x*-axis. Same can be done with *y*- and *z*-axis.

Given a feature vector $\mathbf{x} = (x_1, \dots, x_n)$, and classes $\{C_k\}$, we want to calculate $p(C_k|x_1, \dots, x_n)$, the probability that $\boldsymbol{x}$ belongs to C_k , for each *k*. To determine which class $\mathbf{x}$ belongs to, we want to find *k* that maximizes $p(C_k|x_1, \dots, x_n)$. Assume all features $x_1, \dots, x_n$ are independent. By Bayes' theorem,

$$p(C_k|\mathbf{x}) = \frac{p(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})}.$$

By chain rule,

$$p(C_k|\mathbf{x}) \propto p(C_k, x_1, \dots, x_n) = p(\mathbf{x}|C_k) = p(C_k)p(x_1|C_k) \dots p(x_n|C_k),$$

where $p(C_k)$ is called prior probability, or simply prior. For our application, we set $p(C_k) = 1/k$ for all *k*. $p(x_1|C_k)$ depends on the model fitting methods, such as Gaussian or Bernoulli.

doi:10.6342/NTU201903636

3.3 Local outlier factor (LOF)

LOF is an algorithm to distinguish outliers from clusters of data points, and has been used in audio and image recognition [71]. Previous work used LOF to classify relative wavelet packet entropy [72]. To use this method in this research, it is required to limit the ratio of unstable data points so that they are significantly less than the stable once. This is because LOF finds the outliers, i.e. the points far away from other points.

Define k-distance $k(A)$ be the *k*-th nearest neighbor of point *A*. Denote the distance between two points *A* and *B* as $d(A,B)$. Define reachability distance as

$$r_k(A,B) = max\{k(B), d(A,B)\}.$$

The reason to use reachability distance instead of the distance between *A* and *B* is to get more stable results in computations. Furthermore, define local reachability density as

$$lrd_k(A) = \frac{|N_k(A)|}{\sum_{B\in N_k(A)} r_k(A,B)},$$

where $N_k(A)$ is the *k* nearest neighbors of *A*. Roughly speaking, local reachability density is the reciprocal of the average distance between *A* and its neighbors. The local outlier factor is

$$LOF_k(A) = \frac{\sum_{B\in N_k(A)} \frac{lrd(B)}{lrd(A)}}{|N_k(A)|} = \frac{\sum_{B\in N_k(A)} lrd(B)}{|N_k(A)| \cdot lrd(A)}.$$

doi:10.6342/NTU201903636

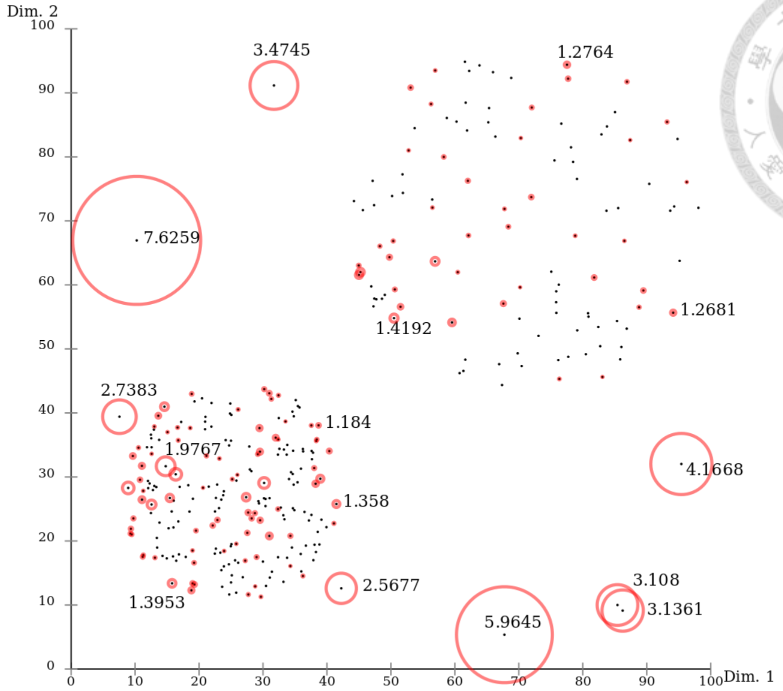


Fig. 15. An example of visualized local outlier factors

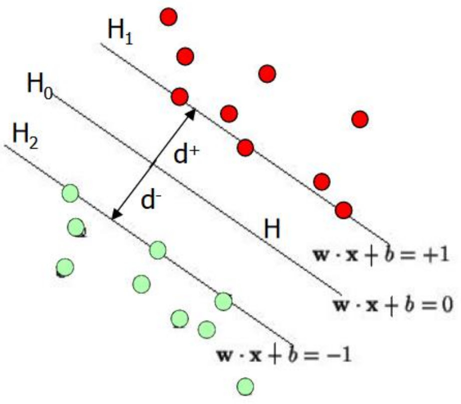
3.4 Support vector machine (SVM)

SVM is a well-known method to separate two classes of data points [73] [74], and has the advantage of good performance because the equations can be written in linear form. SVM has been used for chatter recognition when combined with information entropy [75], WPT [76], and Q-factor [77].

There are two types of SVMs, linear and non-linear. For linear SVM, suppose there are n points $(\mathbf{x_1}, y_1), \dots, (\mathbf{x_n}, y_n)$, where $y_i = \pm 1$, SVM attempts to separate the points with $y_i = 1$ from the ones with $y_i = -1$ using a hyperplane $\mathbf{w} \cdot \mathbf{x} - b = 0$ with some vector $\mathbf{w}$, called support vector. This is illustrated in Fig. 16 (a). In practice, there are cases when the points cannot be separated by a hyperplane as in Fig. 16 (b). In such cases, the points are mapped to higher dimensional space using a kernel function such as

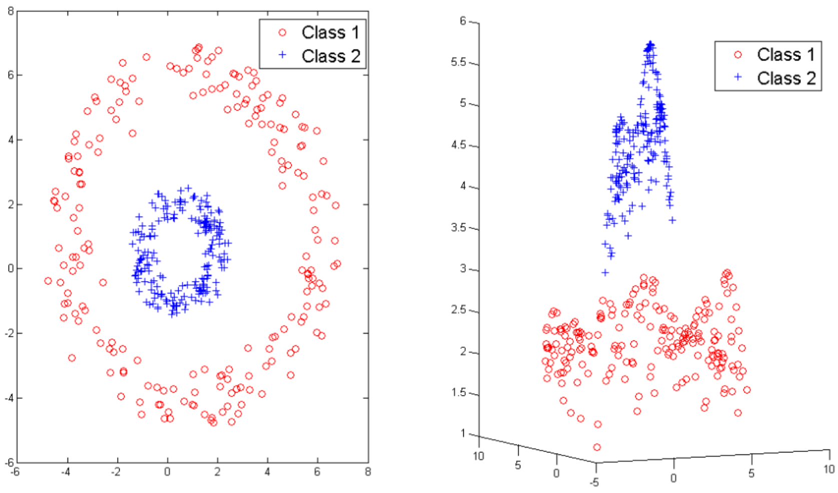
doi:10.6342/NTU201903636

polynomial, RBF, and sigmoid. The mapped points may be easily separable with a hyperplane.



H_1
H_0
H_2
d^+
d^-
H
$\mathbf{w}\cdot\mathbf{x}+b=+1$
$\mathbf{w}\cdot\mathbf{x}+b=0$
$\mathbf{w}\cdot\mathbf{x}+b=-1$

(a)



(b)

Fig. 16. (a) Illustration of linear SVM for a linearly-separable dataset *[78]* (b) Non-linear SVM with RBF kernel *[79]*

doi:10.6342/NTU201903636

3.5 K-nearest neighbor (k-NN)

K-NN is a simple classification algorithm based on votes from the neighbors [80]. Suppose we have a data point *A* without knowing whether it is stable or unstable. We find k-nearest neighbors of *A* for some k. Let's say $k = 5$, and there are 3 of them are stable, and 2 are unstable. Since $3 > 2$, k-NN classify *A* as stable.

It is easy to add a weight function $w(r)$ for each neighbor. For example, let the k neighbors of *A* be $p_1, \dots, p_k$, and the first *m* points are stable while the rest are unstable. Let the distance between A and p_i be r_i for all i. Then, define the scores

$$score(stable) = \sum_{i=1}^{m} w(r_i)$$

$$score(unstable) = \sum_{i=m+1}^{k} w(r_i)$$

If $score(stable) > score(unstable)$, then A should be classified as stable, and vice versa. Some common weight functions $w(r)$ are 1 (uniform), r^{-a}, and e^{-a}, for some $a > 0$.

doi:10.6342/NTU201903636

3.6 Artificial neural network (ANN)

Using ANN in chatter recognition has the advantage that a detailed model is not required and reasonable accuracy may still be achieved without manually select an optimal feature as input. For this reason, we will use the entire frequency spectrum as the input of the ANN. The vibration signal will be converted into frequency domain using FFT, and ANN should be able to distinguish the stable data from unstable data since chatter is more easily recognizable in frequency domain. The architecture and training parameters will be manually tuned to obtain a good accuracy.

doi:10.6342/NTU201903636

4. Implementation

4.1 Architecture of the data analysis and model training platform

Fig. 17 shows the architecture of the model training platform developed in this research. This platform makes it easy to change the feature or classification algorithm to use, adjust the corresponding parameters, and quickly obtain a validation report indicating the effectiveness of the trained model. Any combination of the mentioned features and classification algorithms can be used, along with several modes for tool entry/exit detection, data selection, etc.

The blocks with solid lines in Fig. 17 indicates a data processing or calculation step. A block with dashed lines represents an optional step. First, vibration data from the accelerometer, along with their labels, are loaded from files. Then, the tool entry and exit parts of the vibration signals are excluded to ensure the data being trained are valid cutting data. Certain tooth pass frequencies that are too close to chatter frequencies are filtered out to prevent misidentifications. Features are then computes according to the specified parameters, and saved into a cache file so that if the same feature is to be used again with another classification algorithm, we can simply load all features from cache. This can drastically reduce the computation time. Therefore, we have two data sources: from file (option 1) and from cache (option 2).

The dataset, consisting of feature vectors and labels, is then split into training and test datasets randomly. Models are trained and validated using stratified k-fold validation. Some hyperparameters are automatically tuned to minimize error rate. The validation report of the final model will be shown to summarize the results.

doi:10.6342/NTU201903636

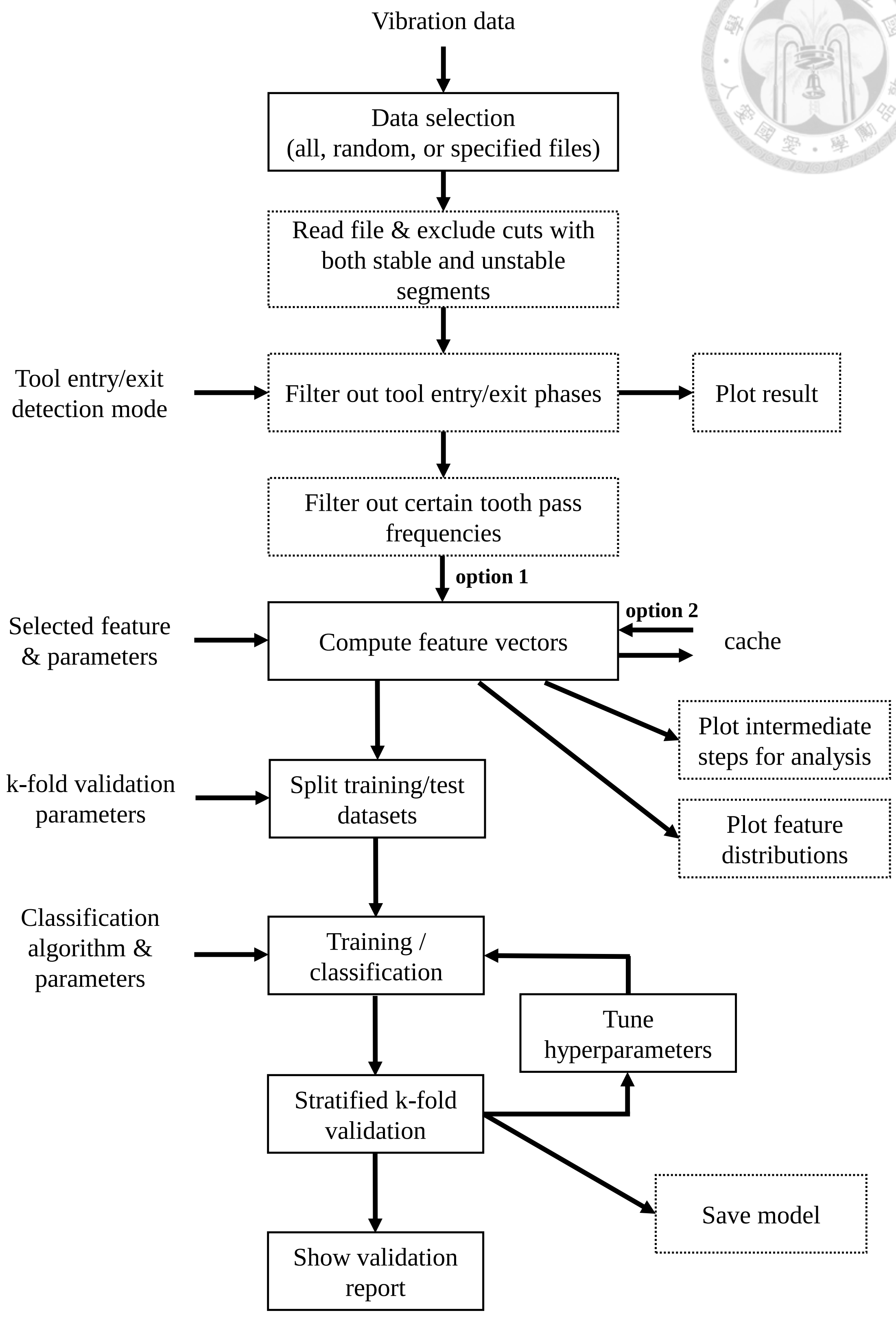

Vibration data
Data selection (all, random, or specified files)
Read file & exclude cuts with both stable and unstable segments
Tool entry/exit detection mode
Filter out tool entry/exit phases
Plot result
Filter out certain tooth pass frequencies
option 1
Selected feature & parameters
Compute feature vectors
option 2
cache
Plot intermediate steps for analysis
Plot feature distributions
k-fold validation parameters
Split training/test datasets
Classification algorithm & parameters
Training / classification
Tune hyperparameters
Stratified k-fold validation
Save model
Show validation report

Fig. 17 Architecture of model training platform for chatter identification

doi:10.6342/NTU201903636

4.2 Implementation details

This section will describe the techniques used to implement the model training platform shown in Fig. 17. The entire program is written in Python 3. The main libraries that was used is listed in *Table 1*.

Table 1. Major libraries used to implement the platform

library	**Main usage in our platform**
numpy	High performance numerical computations
scipy	FFT, filters
PyWavelets	WPT
pyhht	HHT
matplotlib	Data visualizations
scikit-learn	LOF, k-NN
keras	ANN
tensorflow	ANN

4.2.1 Zero-padding before FFT

Our program uses numpy for FFT computations. For some testing data, performance issues were observed. After some investigations, the conclusion is the implementation of numpy's *fft.rfft()* function is the cause of the issue. When the length of input array is not an integer power of 2, it tries to factor the length and split the computation into chunks. However, when the length is a prime number, the computation slows down drastically. As we can see in *Table 2,* the computation time when the array length is 16384 (= 2^{14}) is better than 12581 (= 23×547) and 12584 (= $2^3 \times 11^2 \times 13$), and more than 50 times better than 12583, which is a prime number.

A technique called zero-padding can be applied to resolve this issue. We zero-pad the original vibration signal to so that the total length is an integer power of 2. For example, if the vibration signal has 12583 samples, we add zeros to the array until the length is 16384. This can significantly improve performance while not negatively

doi:10.6342/NTU201903636

affecting the frequency spectrum of the FFT output. In fact, zero-padding improves the frequency resolution of the output spectrum. Note that when window function is used, the signal should be zero-padded before applying a window function.

Table 2. Performance test on numpy.fft.rfft()

Input array length	**Computation time (sec)**
12581	0.01698
12582	0.00898
12583	0.48424
12584	0.01039
16384	**0.00937**
32768	**0.01051**
65536	**0.01897**

4.2.2 Computing autocorrelation coefficients

The time complexity of direct computation of autocorrelation coefficients from (4) is *O*(N^2). However, this can be improved to *O*(*N log N*) using FFT because (4) is similar to the equation of discrete convolution

$$(f * g)[n] = \sum_{n=-N}^{N} f[m]g[n-m].$$

Fig. 18 shows how discrete convolution can be computed in *O*(*N log N*) instead of *O*(N^2) for the direct approach. We show the connection between (4) and the equation above below.

doi:10.6342/NTU201903636

$$\begin{tikzcd}[column sep=huge, row sep=huge]
f,g \arrow[r, "\boldsymbol{O(N^2)}", "{\begin{array}{c}\text{Direct computation}\\ (f*g)[n]=\sum_{n=-N}^{N} f[m]g[n-m]\end{array}}"'] \arrow[d, "\boldsymbol{O(N\log N)}"', "\text{FFT}"] & f*g \\
F,G \arrow[r, "\text{Multiplication}", "\boldsymbol{O(N)}"'] & F\cdot G \arrow[u, "\text{Inverse FFT}", "\boldsymbol{O(N\log N)}"']
\end{tikzcd}$$

Fig. 18. Illustration of the computation of discrete convolution

Proposition Given two 0-indexed arrays *x* and *y*, of length N_x and N_y respectively. Define discrete convolution as

$$(x*y)[m] = \sum_{n=0}^{N_y-1} x[n]y[m-n] \text{ for m} \in [0,\ N_x+N_y-2], \tag{7}$$

and assume $x[i] = y[i] = 0$ for $i < 0$. Let *z* be the reversed array of *x*, i.e. $z[i] = x[N_x - i - 1]$ for all i. Then,

$$(z*x)[m+N-1] = \sum_{n=0}^{N-1} x[n]x[m+n], \tag{8}$$

proof

$(z*x)[m]$

$$= \sum_{n=0}^{N_y-1} z[n]y[m-n] \qquad\qquad (by\ (7))$$

doi:10.6342/NTU201903636

$$= \sum_{p=0}^{N_y-1} z\left[N_y - 1 - p\right] y\left[m + p + 1 - N_y\right] \quad (p = N_y - 1 - n)$$

$$= \sum_{p=0}^{N-1} x[p]\, y[m + p + 1 - N] \qquad (by\ (2),\ and\ let\ N = N_x = N_y)$$

$$= \sum_{n=0}^{N-1} x[n]\, y[(m + 1 - N) + n]$$

Substituting *m* with *m*+*N*-1, we get (8).

The above proposition can be directly applied because (7) matches the implementation of *scipy*'s *fftconvolve*. Specifically, $(x * y)$ = *fftconvolve*(*x*, *y*, *mode='full'*).

4.2.3 Peak finding

Peak finding is used in parts of the program, e.g. finding n-highest peaks in the spectrums after FFT, and finding peaks in autocorrelation coefficient plot. We used $scipy.signal.find\_peaks$ to implement these parts of the program. The parameters used include the minimum horizontal distance between peaks, and prominence.

When trying to find peaks within a graph, e.g. a spectrum, the minimum horizontal distance between peaks is set to ensure that two peaks very close to each other won't be both included in the result. Suppose 20 peaks should be selected from a vibration signal spectrum, there may be 3 peaks near one tooth pass frequency. The unintended

doi:10.6342/NTU201903636

consequence is that probably only 7 tooth pass frequencies are included in the 20 peaks that the algorithm found, instead of 20.

Prominence is a property of a peak, describing the height of a peak relative to the neighboring values. The calculation of prominence is as follows:

(1) Draw a horizontal line, crossing the peak in discussion, until it intersects the signal or the window border. This is illustrated by the green lines in Fig. 19.
(2) For the left and right sides of the peak, find the minimum of each side, these are the bases of the peak as shown in red dots in Fig. 19.
(3) The prominence is the difference between the height of the peak and the higher value of the bases.

Therefore, it is possible to find *n* most prominent peaks from a graph by ordering the prominences of all peaks, and take the largest *n* results. This concept is used in the implementation of autocorrelation coefficients.

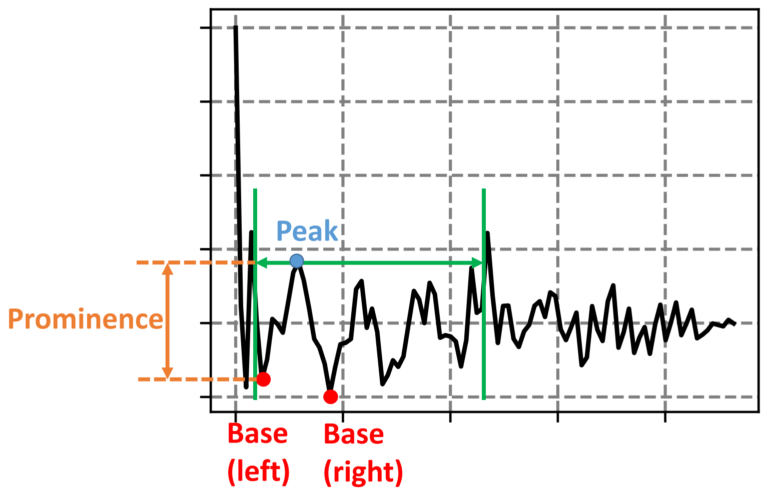


Fig. 19. Illustration of the peak finding procedure and the calculation of prominence of a peak

doi:10.6342/NTU201903636

4.3 Validation

In machine learning, training and test datasets must be separated. Otherwise, it is trivial to obtain a model that achieves 100% accuracy for supervised learning – simply memorize all data and their labels and build a lookup table. To evaluate the true accuracy of a model, we use one of the standard approaches – stratified k-fold validation. The concept is shown in Fig. 20 for the case $k = 3$, where the dataset is split into 3 parts. There are k rounds of validations. For the first round, the first $1/k$ data are used for training while the rest are used for testing. Note that each the ratios of A and B classes in each $1/k$ part should be identical.

In this research, the dataset consists vibration data from 143 cuts. Each cut may result in many data points due to the sliding window we use. However, during the learning and validation process, if a cut is used for training, all data points in that cut are used as training, and vice versa. This ensures the result is fair and minimizing overfitting.

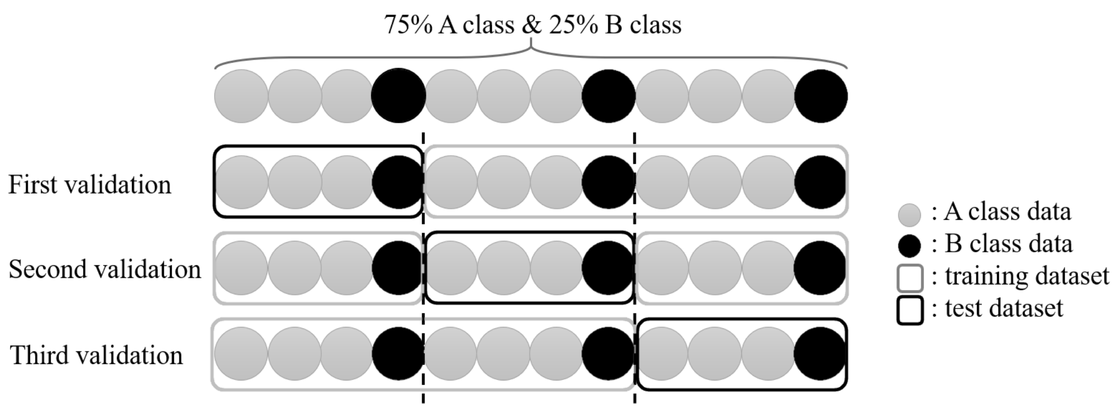


Fig. 20. Illustration of stratified k-fold validation *[47]*

doi:10.6342/NTU201903636

5. Results and discussion

5.1 Data collection and labeling

The experiment data of this research comes from previous work of our laboratory [47]. Since the goal of this research is chatter identification, a sufficient number of experiment data is required as inputs for classification algorithms. Our dataset contains vibration signals from a CNC milling machine, while cutting at different spindle speed, depth of cut, and feed rates. Stability lobe diagram (SLD) of the CNC machine was obtained by the traditional method - modal test and cutting force coefficients measurement. The cutting conditions, which includes spindle speed, depth of cut, and feed rate, were chosen so that it is near the stability boundary of the SLD. SLD is not necessary, but it makes it easier to search for cutting conditions that is in the stable or unstable region.

We used a CNC milling machine to perform the cutting experiment. The experiment setup is shown in Fig. 21. A tri-axial accelerometer was mounted on the spindle housing, and NI-9234 DAQ was used at sampling rate of 10240 Hz to acquire the vibration signals. 143 straight cuts were performed and each cut lasts around 5 seconds, excluding tool entry and exit phase. These 143 cuts contain at least 110 spindle speed and depth of cut combinations, so we have good variety in our dataset. While conducting the experiment, we labeled whether chatter occurs during this cut by listening to the sound. There are 3 possible values for this label – entirely stable, entirely unstable, and partially unstable. Therefore, the dataset consists of vibration signals from 143 cuts, along with a label.

doi:10.6342/NTU201903636

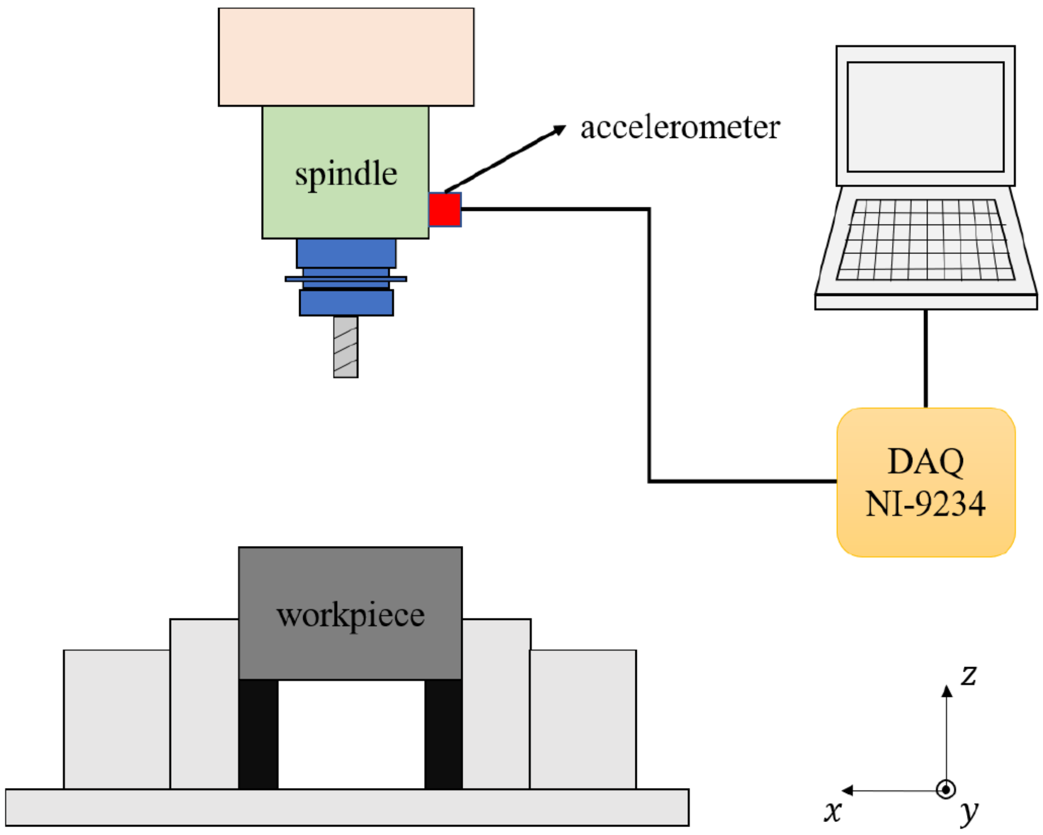


Fig. 21 Experiment setup when gathering the dataset

Accelerometer was installed on the spindle housing, and a NI DAQ was used to acquire the signal. [47]

5.2 Comparisons of classification algorithms

In this section, different classification algorithms are compared using the same feature. The feature used is FFT energy ratio, $E_{1,FFT}$, with window size of 1.2 seconds, no overlap, exponent of 2.75, and all three axes. Since the window and overlap are fixed, the number of data points used to train the models are identical, giving a fair comparison of different classification algorithms.

doi:10.6342/NTU201903636

The numeric threshold method yields an error rate of 6.579%, with false alarm and missing alarm rates both being 3.289%. Naïve Bayes produces error rates of 5.263% and 5.482% using Gaussian and Bernoulli Naïve Bayes, respectively.

Fig. 22 shows the testing cost when using LOF as the classifier. Using stratified k-fold validation with $k = 3$, the cost of 3 test datasets are shown, with the average at the bottom-right corner. In terms of error rate, the optimal amount of neighbors is approximately 150. The cost varies between 0 to 1, with 0 meaning all data points are correctly classified, and 1 meaning all are incorrectly classified. Also, since the unstable data point ratio is an adjustable parameter, the relation between it and error rate is listed in Table 3, where using 15% of unstable data is optimal with an error rate of 8.53%.

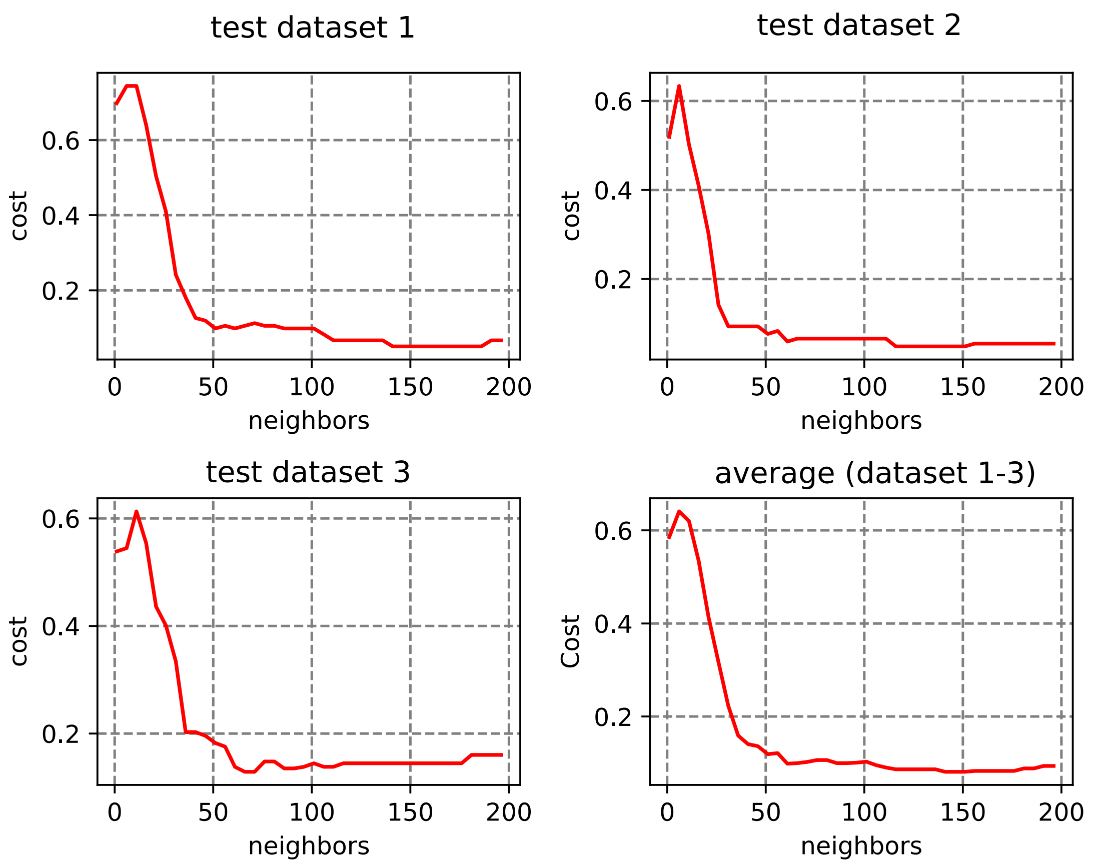


Fig. 22. Test costs of LOF for each of the three test datasets, and the average costs

doi:10.6342/NTU201903636

Table 3. Classification error rates using FFT ($E_{1,FFT}$) and different unstable data point ratio in LOF

Unstable data point ratio (%)	ER (%)
10	10.4 (14.8)
15	8.53 (14.4)
20	9.01 (18.1)

Feature: [window=1.2, overlap=0, exp=2.75, bw=10, xyz], classification: [LOF, auto[1-200], weight 1/r^3]

Fig. 33 shows the error rates on three test datasets and the average using k-NN as the classifier. Table 4 shows the error rates with different weights, including uniform, r^{-a}, and e^{-ar} with some a. The lowest error rate is 5.21% while the highest is 6.061%. The difference is small, which leads to the conclusion that the weight does not affect the error rate significantly.

Fig. 24 sums up this section with all optimal error rates from each classification method compared. K-NN is the best one at 5.21%, almost matched by Naïve Bayes at 5.263% and SVM at 5.647%. Numeric threshold and LOF are the worse ones. Incidentally, an error rate of 7.12% is achieved using ANN, placing it as the 5th best feature. However, since the methodology of it is quite different to other classification methods, it is best to not make direct comparison.

doi:10.6342/NTU201903636

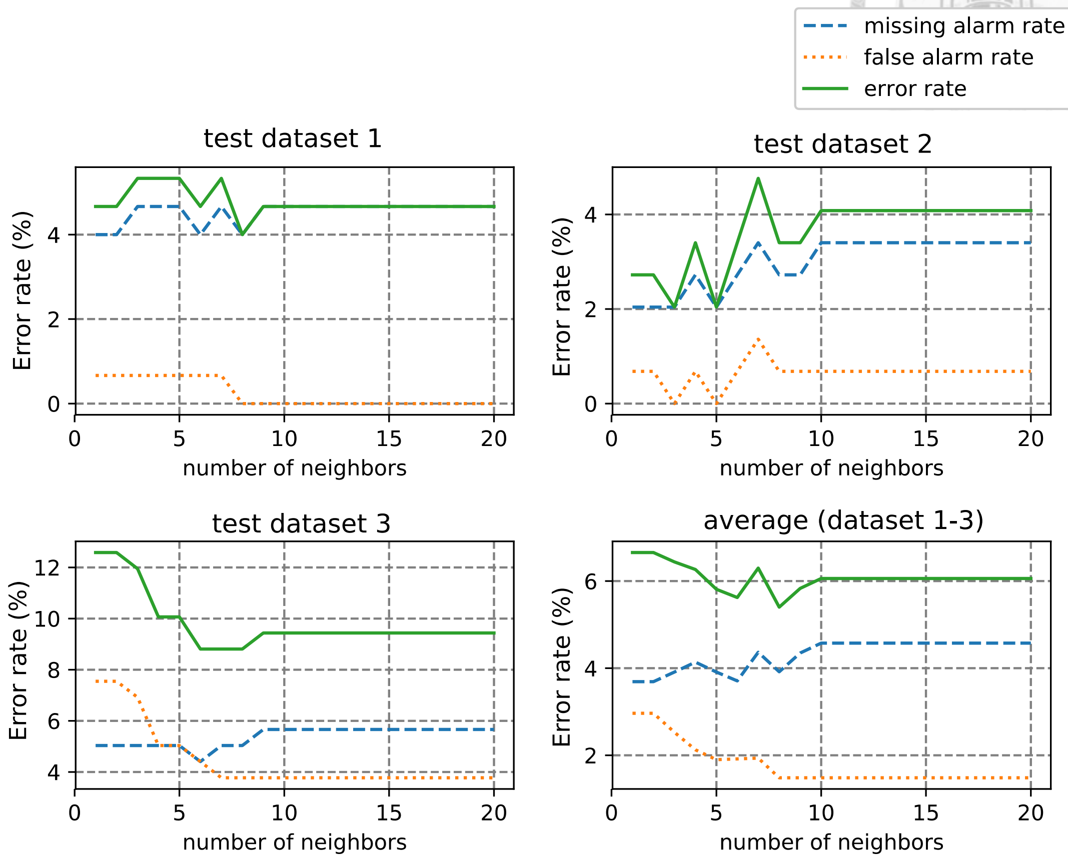


Fig. 23. Test error rates of K-NN for each of the three test datasets, and the average error rates.

doi:10.6342/NTU201903636

Table 4. Classification error rates using k-NN and different weights

Weight	FA (%)	MA (%)	ER (%)
uniform	0.646 (1.258)	5.222 (6.918)	5.868 (8.176)
r^{-1}	1.065 (2.516)	4.564 (6.289)	5.629 (8.805)
r^{-2}	1.275 (3.145)	4.354 (5.66)	5.629 (8.805)
r^{-3}	1.485 (3.774)	4.576 (5.66)	6.061 (9.434)
r^{-4}	1.48 (3.774)	4.349 (5.66)	5.829 (9.434)
e^{-r}	0.646 (1.258)	5.222 (6.918)	5.868 (8.176)
e^{-2r}	0.646 (1.258)	5.222 (6.918)	5.868 (8.176)
e^{-5r}	0.646 (1.258)	4.773 (6.918)	5.419 (8.176)
e^{-10r}	0.8557 (1.887)	4.354 (5.66)	5.21 (7.547)
e^{-100r}	2.096 (6.289)	3.691 (5.031)	5.787 (11.32)

Feature: [window=1.2, overlap=0, exp=2.75, bw=10, xyz], classification: [knn, auto[1-200], weight 1/r^3]

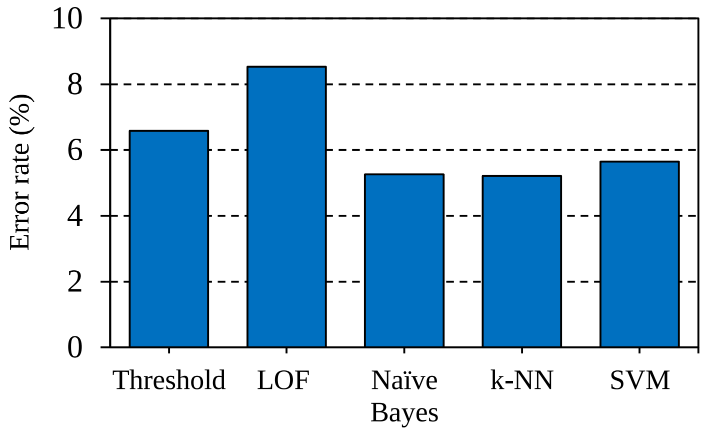


Fig. 24. Error rates when different classification methods are used with the same feature (FFT relative energy)

5.3 Parameters optimizations

This section is focus on parameter optimization for each feature extraction method. For example, the parameters involved in WPT includes mother wavelet, window and overlap sizes, and exponent used to compute the energy. We will find the optimal values for these parameters to obtain the best error rate for WPT, and in later sections this error rate can be compared to other feature types, such as autocorrelation coefficients, to give a fair comparison of the effectiveness of different features.

doi:10.6342/NTU201903636

5.3.1 Fast Fourier transform (FFT)

This section describes the classification results when using FFT-related features, and FFT parameter optimizations will be discussed. The first feature we investigate is the relative energy $E_{1,FFT}$. The exponent p in (1) is an adjustable parameter, and can take any value in $(0, \infty)$. Fig. 25 illustrates the spectrum amplitudes $|S(f)|^p$ for $p = 0.5, 1, 2.75$, and 8. Top right figure is for $p = 1$, i.e. the original FFT spectrum. The highest peak at around 900 Hz is the chatter frequency of the machine. There are several smaller peaks at tooth pass frequencies. In addition, the noise is can be clearly seen in the spectrum as well, especially in the frequency range of 2000 to 3500 Hz. These unwanted noise is sometimes unavoidable and may negatively impact the detection of chatter. We argue that the parameter p can reduce the effect of noise.

When we increase the value of p to 2.75, the smaller peaks become smaller relative to the dominant chatter peak. Due to their smaller amplitudes, the noise between 2000 to 3500 Hz become significantly lower relative to the chatter and tooth pass frequency peaks. If p is further increased to 8 as shown in the bottom right figure, only 2 largest peaks are visible. This is not ideal because most of the information in the spectrum is lost, and classification error rate will increase. On the other hand, using $p < 1$ increases the influence of smaller peaks and amplifies the noise. In summary, there are adverse effects when p is too large or too small. We aim to find an optimal value.

doi:10.6342/NTU201903636

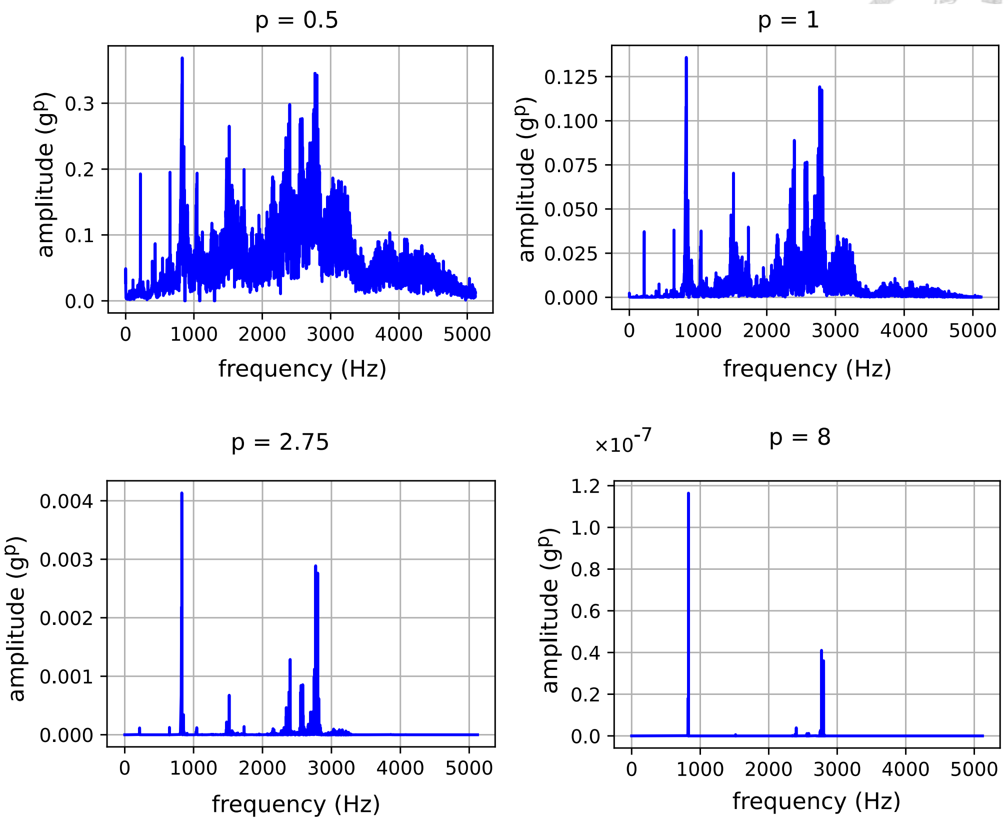


Fig. 25. FFT spectrum raised to exponent p for p = 0.5, 1, 2.75, and 8

Fig. 27 illustrates the effect of p on the distribution of $E_{1,FFT}$. Fig. 27 (a) is the scatter plot of $E_{1,FFT}(y)$ and $E_{1,FFT}(z)$ for $p = 0.5$. The top-right is a large area where the unstable and stable data points overlap. As p increases to 1 and 2.75, the overlapping significantly reduces, as can be seen from Fig. 27 (b) and (c), respectively. Fig. 27 (d) shows that when p is too large, some data points will be pushed to the edge of the graph.

Fig. 26 shows the error rates of trained models using different values of p from 0.5 to 10. K-NN is used with weight r^{-3} and the optimal k is chosen between 1 to 200. The dotted line is the false alarm (FA) rate, the slightly higher dashed line is the missing alarm (MA) rate, and their sum is the total error rate (ER). As we can see, the error rate is really high when $p < 1$. This may be due to the noise being amplified. Then, the error

doi:10.6342/NTU201903636

rates decrease as p increases. The lowest error rate of 6.875% is reached when $p = 2.75$. After this point, the higher peaks in the spectrum dominates, and the information of smaller peaks start getting overlooked. This causes an slight upward trend in error rates from $p = 2.75$ to 10.

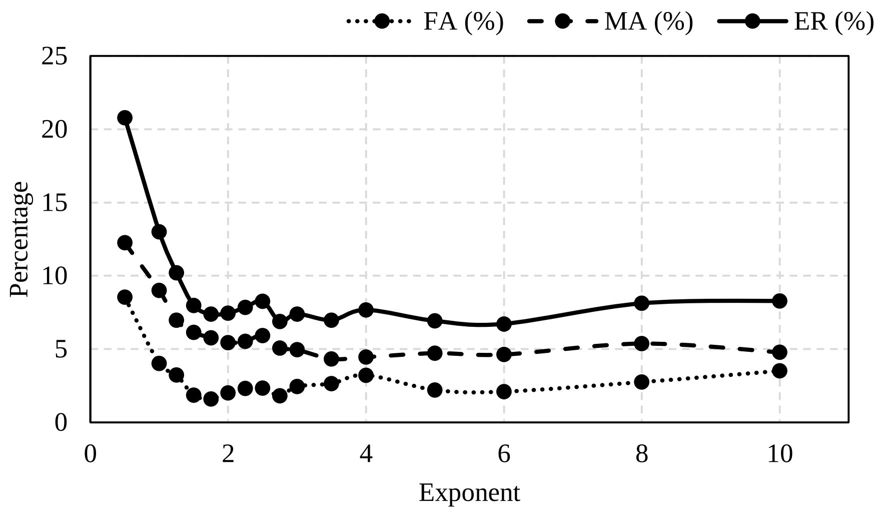
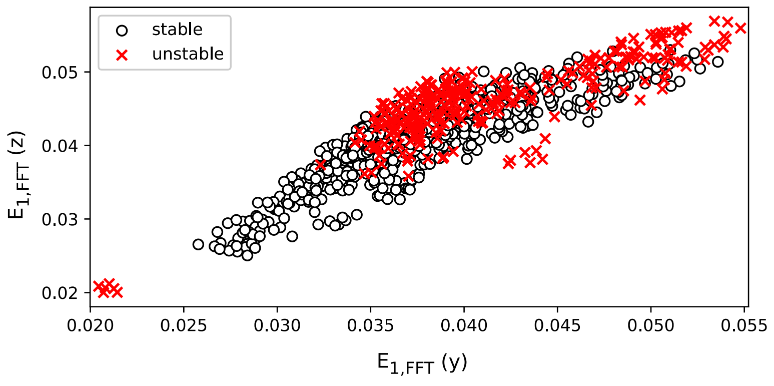


Fig. 26. Effect of exponent p on classification error rates

Feature: [window=1.2, overlap=0.8, bw=10, xyz]
classification: [kNN, auto[1-200], weight 1/r^3]



(a)

doi:10.6342/NTU201903636

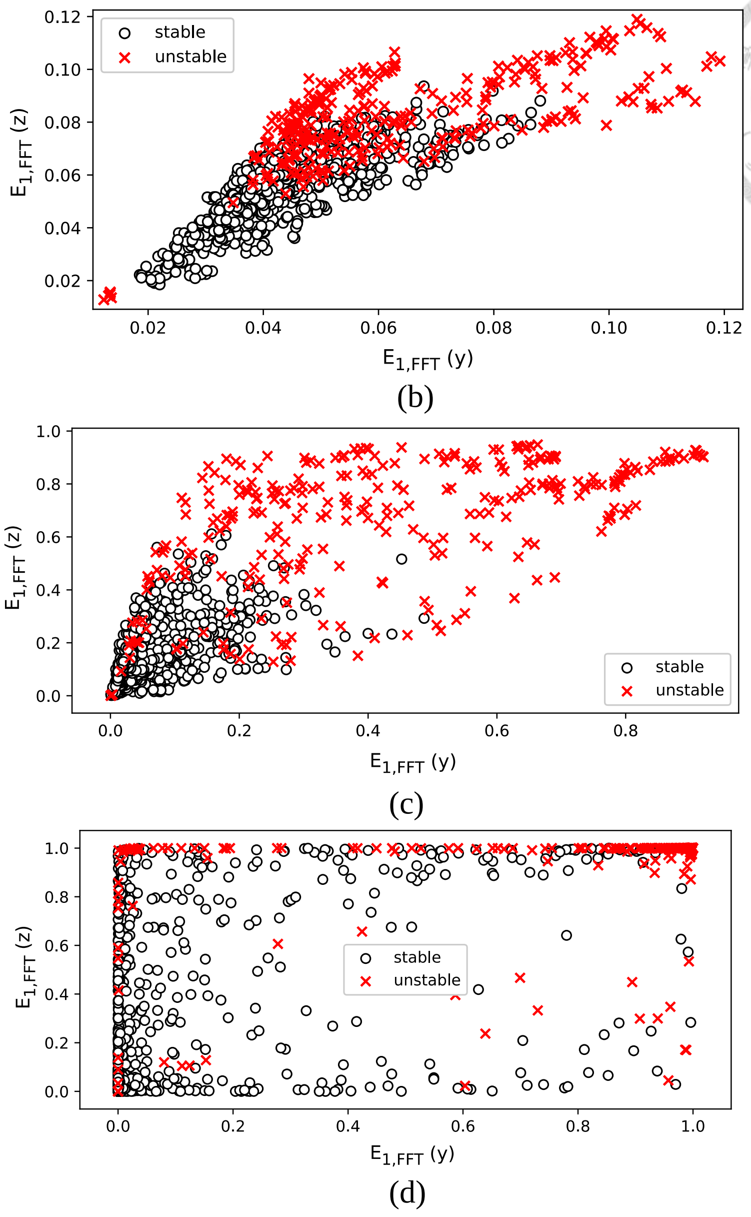


Fig. 27. FFT relative energy plots for (a) p = 0.5, (b) p = 1, (c) p = 2.75, and (d) p = 8

The effect of filter bandwidth is shown in Table 5. There is no clear relationship between the filter bandwidth and error rate. In theory, if the bandwidth is too large, the chatter peak may be also eliminated due to the proximity of tooth pass frequency and dominant frequency for out machine. However, the testing results do not show this trend. This might indicate there is another factor playing a role.

doi:10.6342/NTU201903636

The effect of window and overlap sizes is shown in Table 6. The error rate increases as the window size decreases. This is possibly due to the fluctuation of $E_{1,FFT}$ within a window. A high value of $E_{1,FFT}$ may appear during a stable cut, causing incorrect label and higher error rates when the window size is too small to average out the fluctuations in $E_{1,FFT}$. The overlap does not notably affect the error rate, possibly because the overlap does not increase the total amount of information for model training.

Table 5. Classification error rates using FFT ($E_{1,FFT}$) and different filter bandwidths

Filter bandwidth (Hz)	FA (%)	MA (%)	ER (%)
2.5	1.449 (3.58)	5.216 (7.635)	6.665 (7.635)
5	1.801 (2.837)	5.528 (7.565)	7.329 (10.4)
10	1.808 (2.778)	5.067 (6.921)	6.875 (9.069)
20	3.439 (5.314)	5.172 (7.488)	8.611 (12.8)
40	1.621 (3.095)	4.831 (6.112)	6.451 (7.857)

Feature: [window=1.2, overlap=0.8, exp=2.75, xyz], classification: [kNN, auto[1-200], weight 1/r^3]

doi:10.6342/NTU201903636

Table 6. Classification error rates using FFT ($E_{1,FFT}$) and different window sizes

Window size (sec)	Overlap (sec)	Data points	FA (%)	MA (%)	ER (%)
1.2	0.8	1216	1.808 (2.778)	5.067 (6.921)	6.875 (9.069)
0.9	0.6	1736	1.721 (2.152)	5.851 (7.719)	7.572 (8.772)
0.6	0.4	2780	2.455 (3.459)	5.818 (8.491)	8.273 (11.95)
0.3 *	0.2	5895	3.582 (5.143)	7.484 (11.05)	11.07 (12.51)
1.2	0	456	1.705 (4.459)	4.63 (6.579)	6.335 (7.237)
0.9	0	608	1.94 (3.81)	5.862 (8.867)	7.802 (10.48)
0.6	0	972	2.679 (3.481)	5.614 (8.358)	8.293 (11.04)
0.3	0	2010	2.917 (4.512)	7.107 (8.006)	10.02 (12.52)
0.1 *	0	6169	3.803 (5.082)	9.124 (9.545)	12.93 (14.16)
0.05 **	0	12338	3.827 (4.573)	9.895 (10.83)	13.72 (15.4)

Feature: [exp=2.75, bw=10, xyz], classification: [kNN, auto[1-200], weight 1/r^3]
* classification: [kNN, auto[1-500], weight 1/r^3]
** classification: [kNN, auto[1-2000], weight 1/r^3]

Table 7 shows the result using $E_{1,FFT}$, by ignoring all but n highest peaks in the spectrum. Here, d is the minimum frequency difference between two consecutive peaks. If there are two peaks with whose frequency difference is less than d, the latter one will be ignored. The optimal error rate of 6.441% is achieved at $d = 50\ Hz, n = 30$, closely followed by 6.511% is achieved at $d = 50\ Hz, n = 5$. It’s worth noting this is only slightly worse than 6.335% shown in Table 7 with windows size of 1.2 seconds and no overlap.

doi:10.6342/NTU201903636

Table 7. Error rates using $E_{1,FFT}$, by ignoring all but n highest peaks

n \ d	10 Hz	50 Hz	100 Hz
5	7.813 (11.14)	6.511 (8.92)	7.479 (10.9)
10	8.226 (10.55)	7.436 (10.36)	12.84 (18.1)
30	7.829 (8.458)	6.441 (12)	6.991 (7.214)

Table 8 shows the error rates using $E_{2,FFT}$ and different number of peaks (n). The error rate is slightly higher when n is small ($n = 5$), but there is no significant difference from $n = 10$ from $n = 30$. Table 9 shows the error rates for $E_{3,FFT}$. Similar to $E_{2,FFT}$, the parameter n has no significant effect for $n \geq 10$. Fig. 28 summarized the best results from $E_{1,FFT}$, $E_{2,FFT}$, and $E_{3,FFT}$. The data come from the minimum error rates from Table 6, Table 8, and Table 9, respectively. $E_{1,FFT}$ is the superior feature among the three with an error rate of 6.34 %, with $E_{2,FFT}$ close behind.

doi:10.6342/NTU201903636

Table 8. Classification error rates using FFT ($E_{2,FFT}$) and different number of peaks

Number of peaks (n)	FA (%)	MA (%)	ER (%)
5	3.53 (3.991)	6.466 (7.277)	9.996 (11.27)
10	2.251 (4.481)	6.107 (10.61)	8.358 (10.86)
20	3.081 (5.213)	5.515 (6.733)	8.596 (10.19)
30	2.295 (2.716)	6.154 (7.092)	8.449 (9.456)

Feature: [window=1.2, overlap=0.8, exp=2.75, range=5, xyz], classification: [kNN, auto[1-200], weight 1/r^3]

Table 9. Classification error rates using FFT ($E_{3,FFT}$) and different number of peaks

Number of peaks (n)	FA (%)	MA (%)	ER (%)
5	6.524 (9.39)	8.713 (11.06)	15.24 (17.84)
10	6.07 (11.37)	8.051 (8.294)	14.12 (19.67)
20	6.819 (7.769)	8.049 (8.772)	14.87 (16.54)
30	6.618 (8.983)	8.203 (9.456)	14.82 (18.44)

Feature: [window=1.2, overlap=0.8, exp=2.75, range=5, xyz], classification: [kNN, auto[1-200], weight 1/r^3]

doi:10.6342/NTU201903636

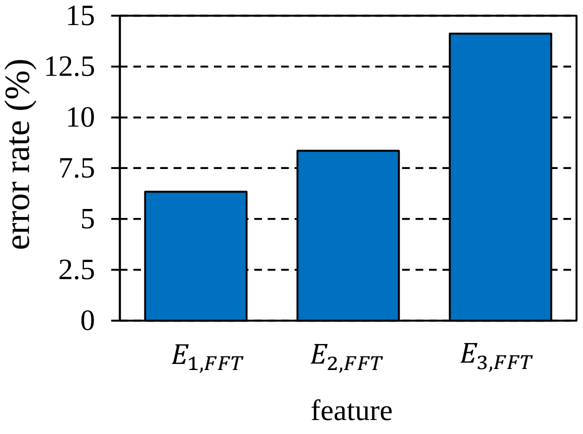


Fig. 28. Comparison of error rates with different FFT related features

5.3.2 Wavelet packet transform (WPT)

Fig. 29 shows the comparison of error rates within each mother wavelet family. Here, we tested Daubechies (*db*), Coiflet (*coif*), Symlet (*sym*), biorthogonal (*bior*), reverse biorthogonal. The best mother wavelets in each family are *db20, coif1, sym2, bior1.5,* and *rbio1.3*. Note that not all mother wavelets are tested. For instance, only *db1* to *db4, db10, db15, db20,* and *db30* are tested for Daubechies wavelets. The reason is that testing all wavelets is impractical given the large number of choices. More lower-numbered wavelets are tested compared to the higher-numbered ones because the difference between *db3* and *db4* is much more significant than *db19* and *db20* in terms of the shape of the wavelet function. Also, *db1*, *bior1.1,* and Haar refer to the same wavelet.

The error rates are typically within 6 to 10 % showing that the variation is not large regardless of which wavelet we choose. As shown in the figure, there is no clear way to pick a wavelet within a family. Fig. 30 is a comparison of error rates across different families with the lowest error rate in each family chosen as a representation.

doi:10.6342/NTU201903636

Daubechies (*db*) wavelets performs best with an error rate of 6.019%, followed by *bior* and *rbio* at around 6.6%, whereas *dmey* is the worst at 9.3%. However, since the difference in error rates is not too significant between the best and the worst wavelets, it is difficult to judge whether the same conclusion can be reached using another machine. Nevertheless, the main point is that among the wavelet families we tested, there should not be a large difference regardless which one is used.

Fig. 31 shows the error rates when different axes are used as feature. Since we use a tri-axial accelerometer to acquire data, there are 6 possible combinations. We can use only one axis: *x*, *y*, or *z*. We can also use a combination of them: *xy*, *xz*, *yz*, or *xyz*. The error rate when only one axis is used is substantially higher, between 13.52 to 18.11%. Using two axes produces error rates between 10.81 to 11.3%. The optimal result of 6.019% is obtained by using all three axes. It is interesting to know that even if the direction of cutting is *y*-axis, *x*- and *z*-axis both contribute heavily towards the final classification model.

doi:10.6342/NTU201903636

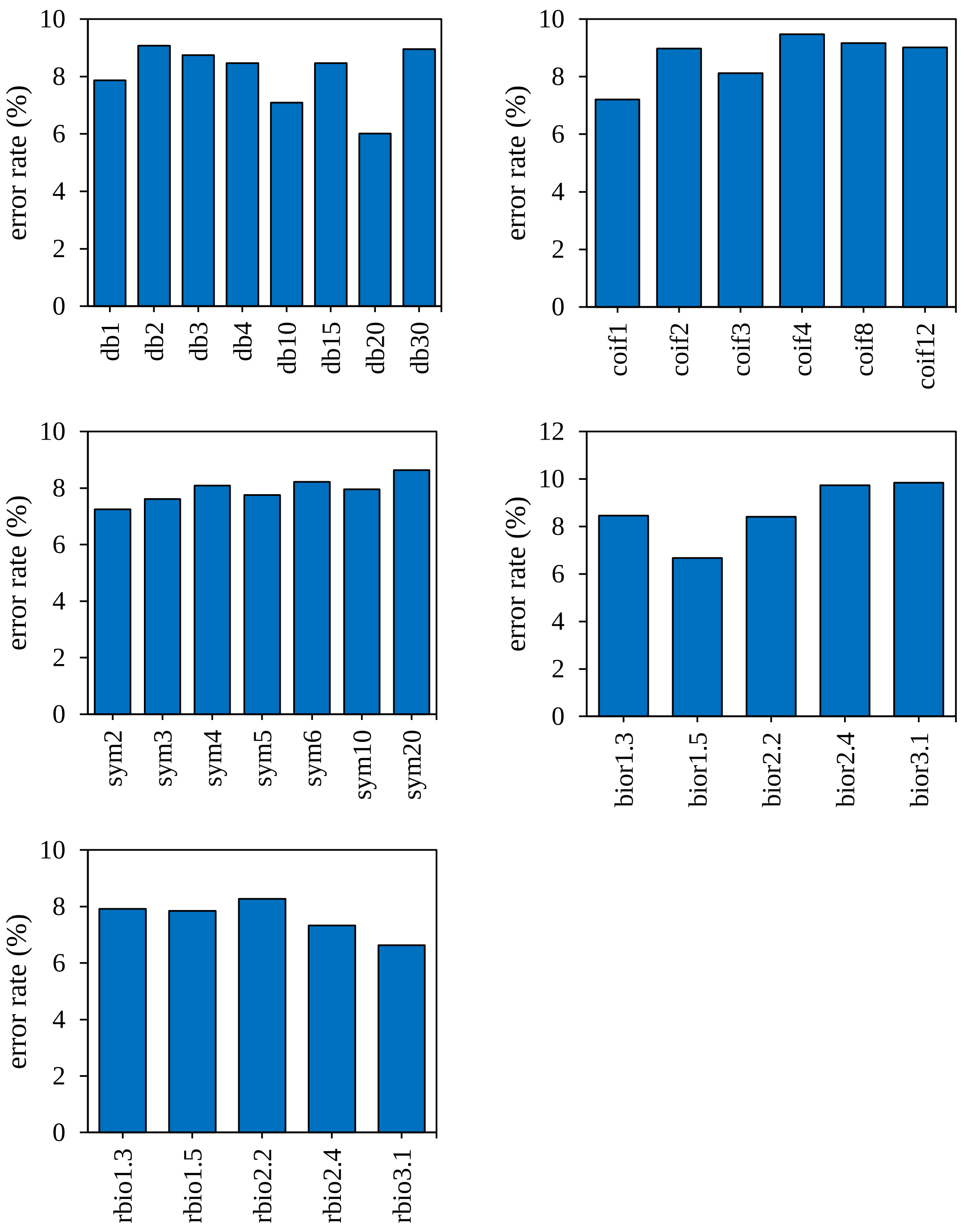


Fig. 29. Comparison of average error rates within each mother wavelets family

doi:10.6342/NTU201903636

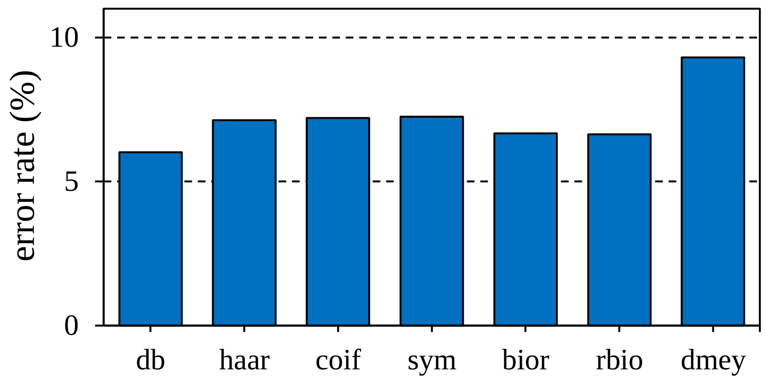


Fig. 30. Comparison of average error rates within each mother wavelets family

Feature: [window=1.2, overlap=0.8, exp=2.75, bw=10, xyz], classification: [kNN, auto[1-200], weight 1/r^3]

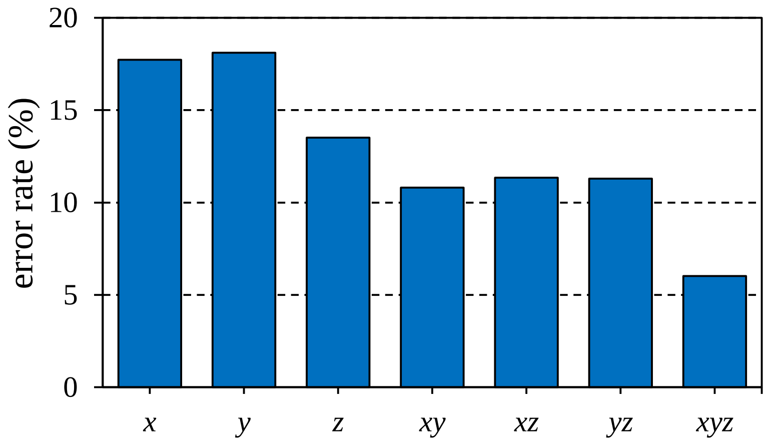


Fig. 31. Comparison of average error rates when different axes are used as feature in k-NN classifier

Feature: [window=1.2, overlap=0.8, exp=2.75, db20, bw=10], classification: [kNN, auto[1-200], weight 1/r^3]

doi:10.6342/NTU201903636

Table 10 shows the error rates when different window sizes are used. Two observations can be made from this table. The first is that the effect of overlap is marginal, improving the error rates by at most around 1%. The second is that larger window size decreases the error rates significantly, with a difference of more than 4% between 1.2 and 0.3 seconds. Fig. 32 shows the distribution of normalized relative energy. The stable data points lie in a relatively small region, i.e. the orange box.

Table 10. Classification error rates using WPT and different window sizes

Window size (sec)	Overlap (sec)	Data points	FA (%)	MA (%)	ER (%)
1.2	0.8	1216	1.13 (2.118)	4.89 (6.824)	6.019 (8.941)
0.9	0.6	1736	3.094 (6.811)	6.15 (9.683)	9.244 (10.63)
0.6	0.4	2780	3.266 (7.239)	6.391 (8.705)	9.657 (11.27)
0.3 *	0.2	5895	3.375 (5.891)	6.975 (8.277)	10.35 (14.17)
1.2	0	456	1.287 (2.5)	5.883 (8.054)	7.17 (9.375)
0.9	0	608	2.936 (5.742)	6.412 (7.656)	9.347 (13.4)
0.6	0	972	3.248 (6.928)	6.464 (7.53)	9.712 (14.46)
0.3	0	2010	3.825 (7.648)	7.688 (10.63)	11.51 (12.84)
0.1 *	0	6169	4.325 (5.477)	9.005 (11.35)	13.33 (14.69)
0.05 **	0	12338	4.531 (5.464)	10.41 (12.65)	14.94 (18.12)

Feature: [exp=2.75, db20, bw=10, xyz], classification: [kNN, auto[1-200], weight 1/r^3]
* classification: [kNN, auto[1-500], weight 1/r^3]
** classification: [kNN, auto[1-2000], weight 1/r^3]

doi:10.6342/NTU201903636

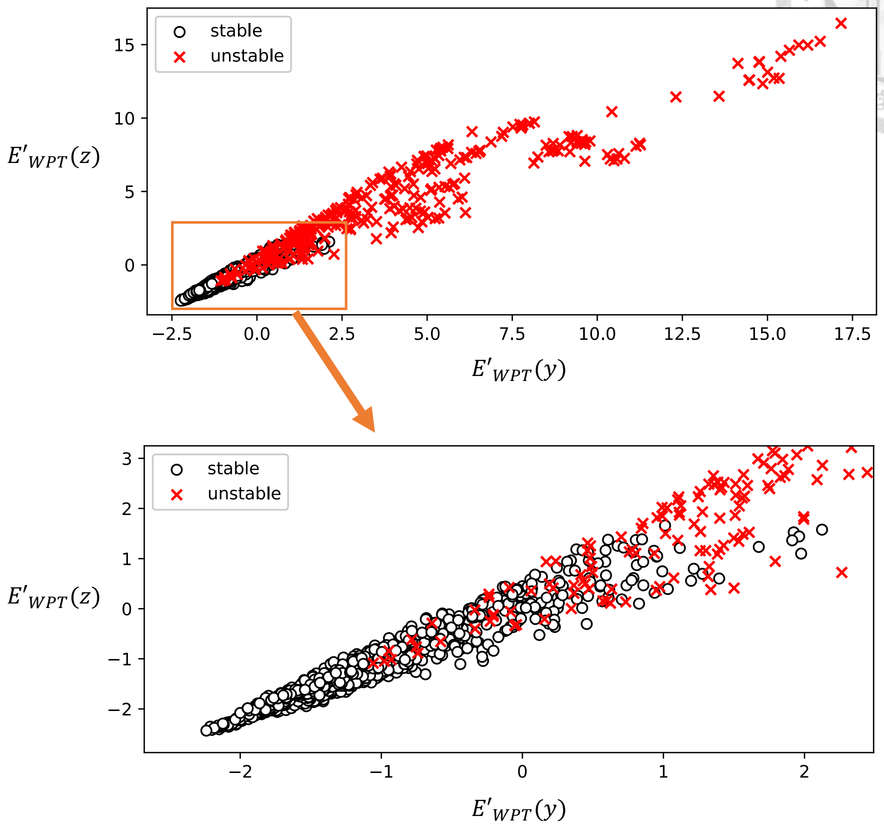


Fig. 32. Normalized relative energies $E'_{WPT}(y)$ and $E'_{WPT}(z)$ using WPT

Feature: [window=1.2, overlap=0.8, exp=2.75, haar, xyz], classification: [kNN, auto[1-200], weight 1/r^3]

doi:10.6342/NTU201903636

5.3.3 Autocorrelation coefficients

Fig. 33 shows the autocorrelation coefficient for acceleration signal under stable cutting conditions. The left side shows the signal from accelerometer. The length acceleration signal is approximately 9.4 ms, which is equal to two spindle rotation periods. In order to find T_X, during peak finding, the peaks with top 50% prominence is chosen. The right side is the autocorrelation coefficient calculated from (5), with time delay τ as the *x*-axis. The vertical blue line indicates the position of T_1 and it coincides with a peak in autocorrelation coefficient as expected.

Fig. 34 shows the result of an unstable cut. The blue line does not coincide with a peak in autocorrelation coefficient as expected. In fact, the blue line coincides with a local minimum, indicating there is significant vibration in frequencies other than tooth pass frequencies.

Fig. 35 is the autocorrelation coefficients of acceleration, velocity, and displacement. The DC component of the measured acceleration signal is subtracted, and then we integrate the signal with respect to time to obtain the velocity. Similarly, displacement can be calculated. Peaks in autocorrelation coefficients are clearly visible when τ is equal to spindle rotation period, which is the expected behavior. However, when we integrate the signal, the finer vibration details are lost. The vibrations caused by each tooth is visible in the acceleration graph, barely recognizable in the velocity graph, and completely disappeared in the displacement graph.

doi:10.6342/NTU201903636

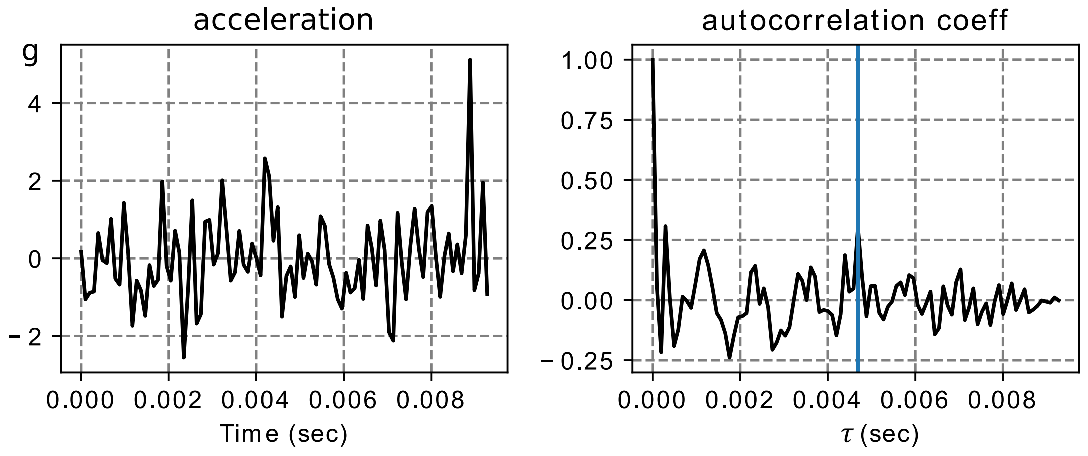


Fig. 33. Autocorrelation coefficient for a stable cut

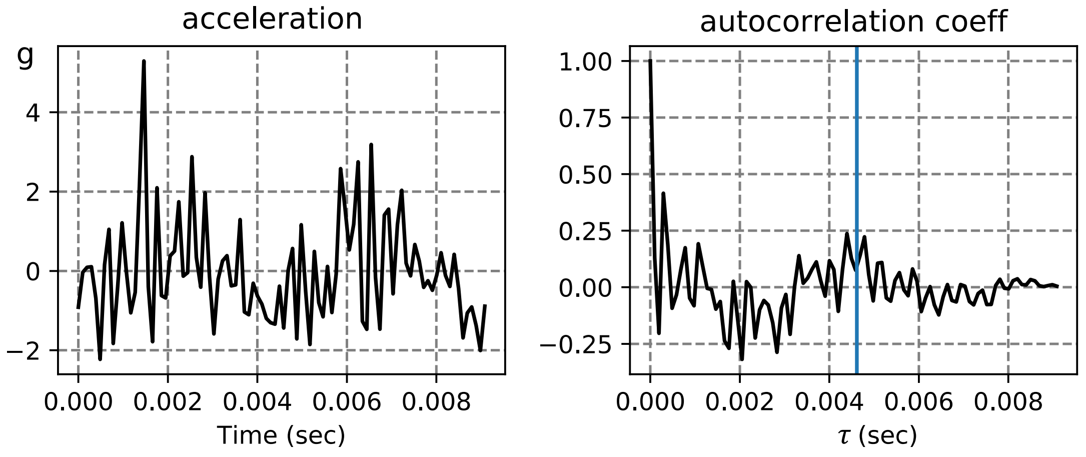
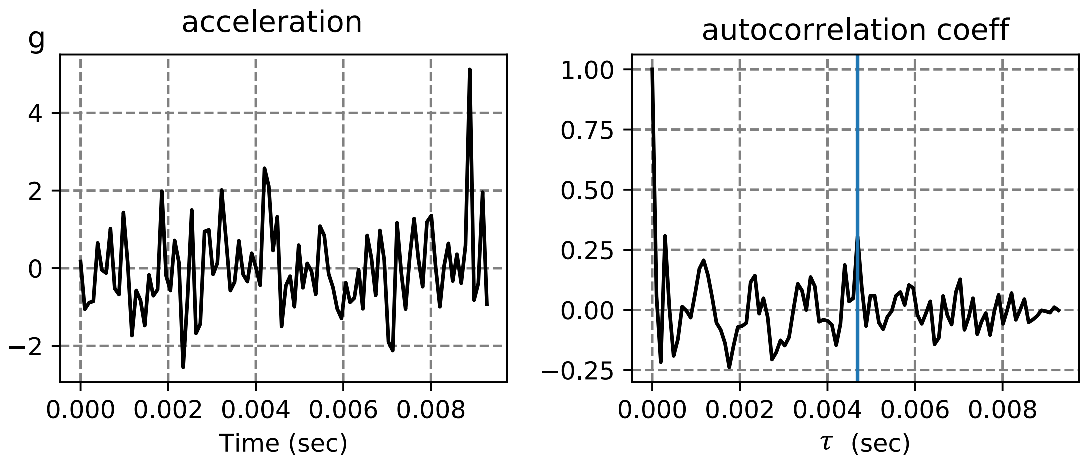
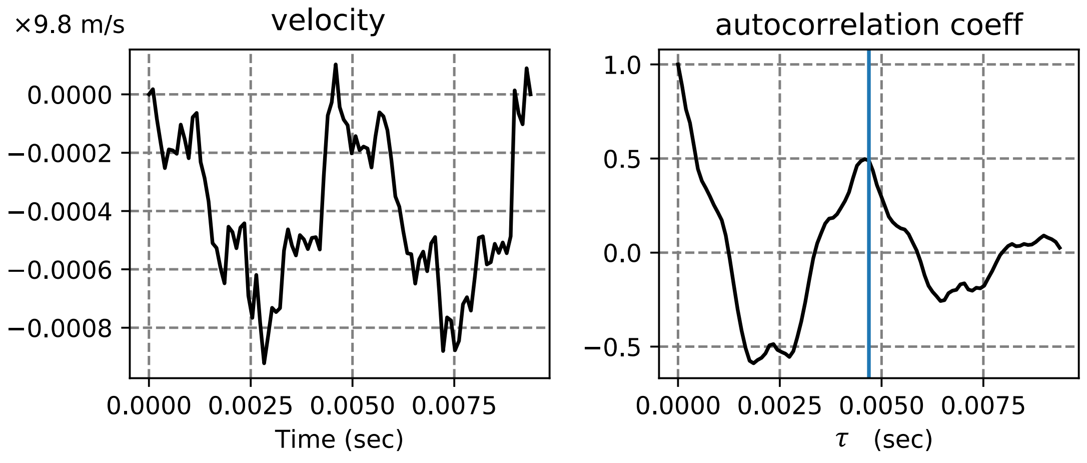


Fig. 34. Autocorrelation coefficient for an unstable cut

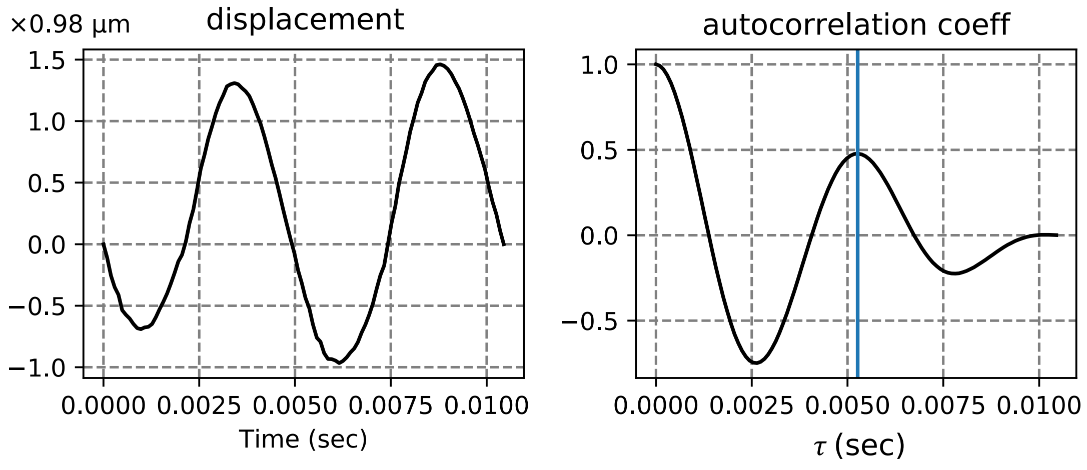
doi:10.6342/NTU201903636



(a)



(b)



(c)

Fig. 35 Autocorrelation coefficient of (a) original acceleration signal, (b) velocity signal, and (c) displacement signal.

doi:10.6342/NTU201903636

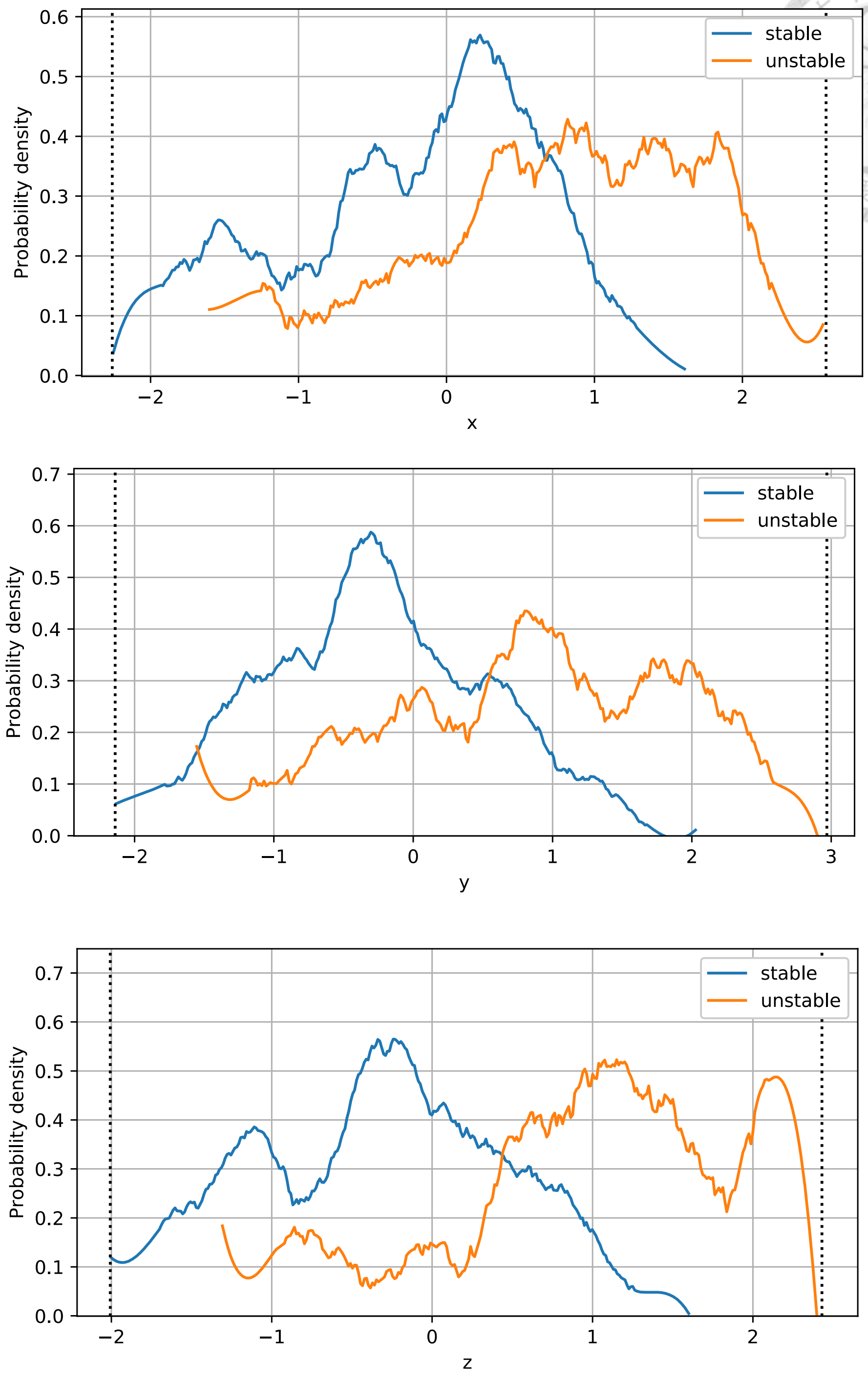


Fig. 36. Distribution of standardized phase differences ε for *x*-, *y*-, and *z*-axes.

Fig. 37 shows the phases differences ε calculated from autocorrelation coefficients. A large amount stable and unstable data points overlap, which can be also seen in the distribution charts in Fig. 36 and is not ideal. This pattern can be observed

doi:10.6342/NTU201903636

regardless of the window size, overlap, or prominence parameter. Phase differences is often smaller for stable cuts compared to the unstable ones, but sometimes the phase difference of unstable cuts are small as well.

Table 11 is a comparison of different window sizes and overlap lengths. As we can see, the error rate increases from 16.93% to 22.8% as the window size decreases from 1.2 seconds to 0.05 seconds. Because the characteristics of vibration signal fluctuates with time, we cannot expect the phase difference ε to remain the same even during a single cut. The fluctuations in ε averages out when a larger window size is used, and produces a more representative feature and overall better accuracy. The effect of overlap in error rates far less significant. Despite the increase in number of data points when the overlap is larger, the extra data points contributes little to the trained classification models.

Table 12 shows how the prominence parameter affect the error rates. A value between 70% to 90% is marginally better in terms of the results. Data from Table 13 indicates that the velocity is a marginally better feature, having a 16.42% error rate compared to 17.93% and 17.96% for acceleration and displacement respectively.

doi:10.6342/NTU201903636

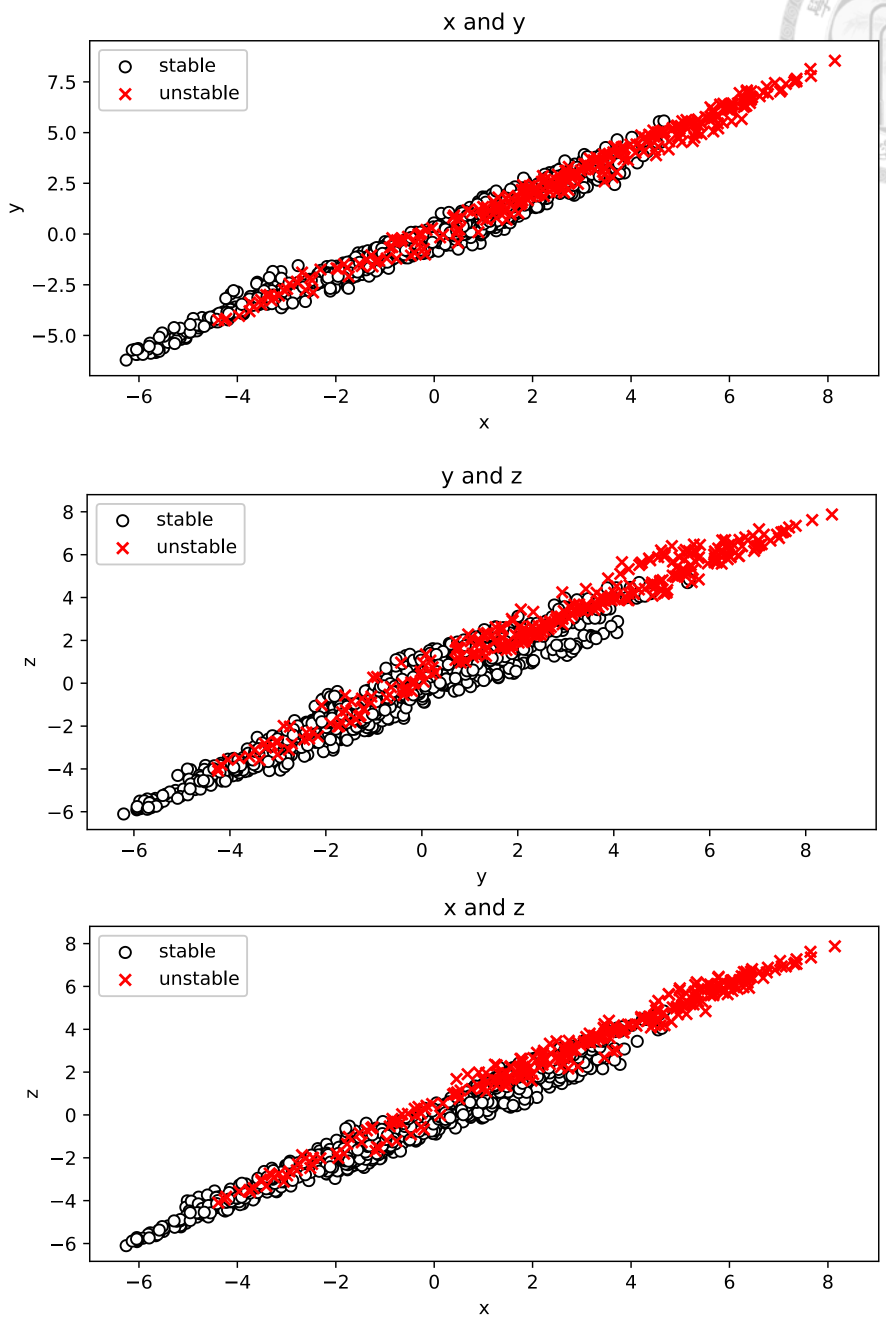


Fig. 37. Phase differences ε of the entire dataset, in *x*-, *y*-, and *z*-directions.

Window size is 1.2 seconds with overlap of 0.8 seconds. Data is acceleration signal. Peaks with prominence in top 50% are considered. The phase is in degree, with a period equaling to spindle speed period.

Table 11. Classification error rates using autocorrelation coefficients and different window sizes

Window size (sec)	Overlap (sec)	Data points	FA (%)	MA (%)	ER (%)
1.2	0.8	1216	6.961 (11.03)	11.01 (14.96)	17.97 (18.97)
0.9	0.6	1736	7.258 (10.44)	10.79 (14.54)	18.05 (21.37)
0.6	0.4	2780	6.1 (10.53)	11.24 (14.62)	17.34 (19.78)
0.3	0.2	5895	6.493 (9.336)	11.48 (16.49)	17.97 (23.54)
1.2	0	456	6.895 (13.51)	10.03 (13.42)	16.93 (18.24)
0.9	0	608	6.806 (11.68)	12.33 (13.43)	19.13 (22.84)
0.6	0	972	6.361 (9.873)	12.57 (16.41)	18.93 (23.53)
0.3	0	2010	6.357 (9.035)	10.96 (12.59)	17.31 (17.57)
0.1 *	0	6169	6.654 (8.674)	12.08 (14.57)	18.74 (19.79)
0.05 *	0	12338	8.793 (11.8)	14.01 (15.78)	22.8 (23.05)

Feature: [prominence=50%, a, xyz], classification: [kNN, auto[1-200], weight 1/r^3]
* classification: [kNN, auto[1-2000], weight 1/r^3]

doi:10.6342/NTU201903636

Table 12. Classification error rates using autocorrelation coefficients and different prominence percentages

Prominence	FA (%)	MA (%)	ER (%)
10%	7.137 (14.01)	11.45 (16.99)	18.58 (21.59)
30%	7.39 (9.591)	11.58 (14.12)	18.97 (19.79)
50%	7.258 (10.44)	10.79 (14.54)	18.05 (21.37)
70%	6.805 (10.05)	10.66 (13.38)	17.46 (17.99)
90%	6.16 (10.12)	11.77 (14.59)	17.93 (18.8)

Feature: [window=0.9, overlap=0.6, a, xyz], classification: [kNN, auto[1-200], weight 1/r^3]

Table 13. Comparison of classification error rates between acceleration, velocity, and displacement using autocorrelation coefficients.

Feature	Data points	FA (%)	MA (%)	ER (%)
acceleration	1736	6.16 (10.12)	11.77 (14.59)	17.93 (18.8)
velocity	1736	7.548 (10.55)	8.868 (11.05)	16.42 (20.18)
displacement	1736	8.846 (9.615)	9.11 (9.464)	17.96 (18.71)

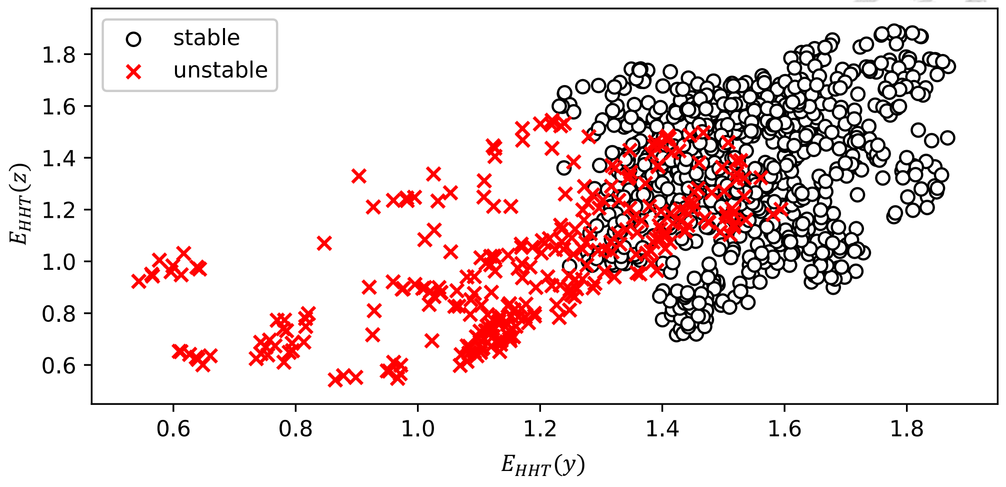
Feature: [window=0.9, overlap=0.6, prominence=50%, xyz], classification: [kNN, auto[1-200], weight 1/r^3]

doi:10.6342/NTU201903636

5.3.4 Hilbert-Huang transform (HHT)

Fig. 38 shows the distribution of relative energy E_{HHT} in the y- and z-directions. The top figure is the result with HHT only, and the bottom one is first processed by WPT. It is interesting that the two distributions are opposite. When using HHT only, E_{HHT} is higher for the stable data points. That implies in stable conditions, the amplitude of the first IMF is high. However, the pre-processing using WPT reverses the trend. Table 14 gives a comparison between the error rates between the two approaches. It is clear that WPT+HHT is better than using HHT only, as suggested by a previous research [53]. Fig. 39 shows the error rates when using different axes as feature. Surprisingly, using only *x*- and *z*-axis is slightly better than using all three axes. Nevertheless, using all three axes is still much more robust compared to using one axis.

doi:10.6342/NTU201903636



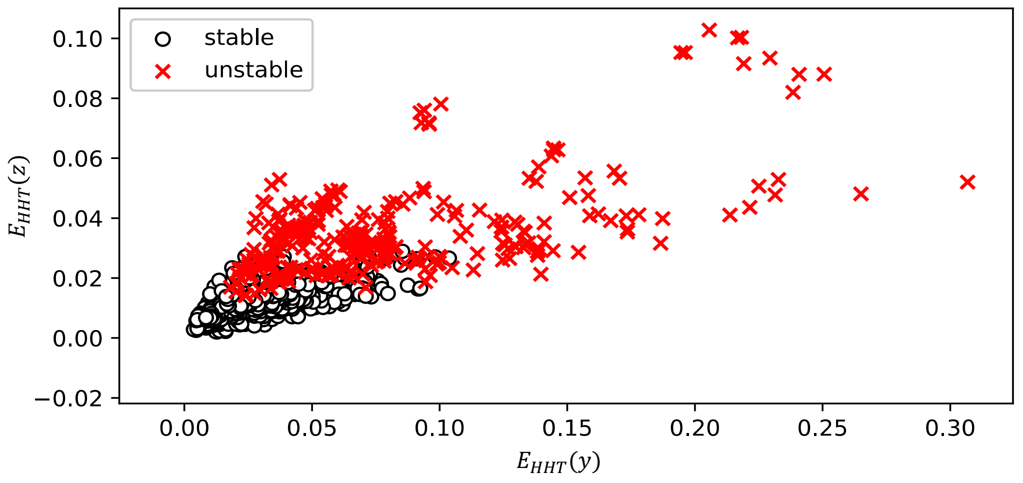


Fig. 38. Distribution of the *y*- and *z*-axis relative energy after only HHT (top), and WPT+HHT (bottom)

doi:10.6342/NTU201903636

Table 14. Classification error rates using only HHT, and WPT+HHT

Feature	FA (%)	MA (%)	ER (%)
HHT only	4.18 (5.952)	8.043 (9.383)	12.22 (14.05)
WPT + HHT	2.049 (3.465)	2.907 (5.437)	4.957 (7.092)

Feature: [window=1.2, overlap=0.8, xyz], classification: [kNN, auto[1-200], weight 1/r^3]

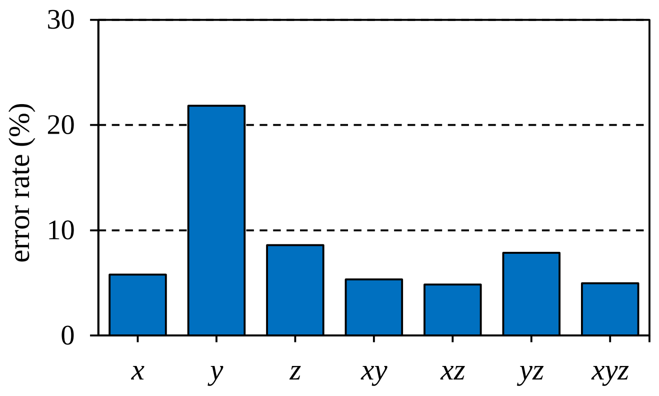


Fig. 39. Comparison of average error rates when different axes are used as feature

5.3.5 Frequency spectrum (with artificial neural network)

The ANN in our platform is implemented using *tensorflow*. Some simple neural networks architectures were attempted, including fully connected layers and dropout layers. The input is a vector consisting of the magnitude of the spectrum after FFT. Since the sampling rate is 10 kHz, the length of the input vector is $\frac{10000}{2} + 1 = 5001$. The input vector is normalized for optimal results. The output is a single number indicating whether it is stable or unstable. The number of layers, number of units in each layer, activation function, dropout rate, and batch size are varied to find the optimal test accuracy. The dataset is split into 2 parts, with 70% used for training, and 30% used for testing.

doi:10.6342/NTU201903636

92.88% test accuracy is achieved using two fully connected layers with 20 and 10 units, respectively. The activation functions for both layers are *relu*. Loss function is binary cross-entropy, optimizer is *adam*, and batch size is 200. 92.88% accuracy is achieved with 10 epochs. In general, the training accuracy is higher than test accuracy, and the parameters above is mainly selected to avoid overfitting.

5.4 Comparison of features

Fig. 40 is a collection of probability density functions from each feature. For each plot of the figure, the parameters used are the ones that process the lowest error rate. Only the distribution of y-axis is shown since it is the direction of feed. A feature is a good chatter indicator if the overlap between the stable and unstable distributions is small. For example, the overlap is large for the phase difference ε from autocorrelation coefficient, which indicates that it is not a good feature. This is evident from its classification error rate, which is the highest amongst the six. The first two FFT-related features, $E_{1,FFT}$ and $E_{2,FFT}$, both show a high peak for the stable curve, which implies a large amount of stable data points have low $E_{1,FFT}$ or $E_{2,FFT}$. However, the unstable data points distribute relatively evenly.

doi:10.6342/NTU201903636

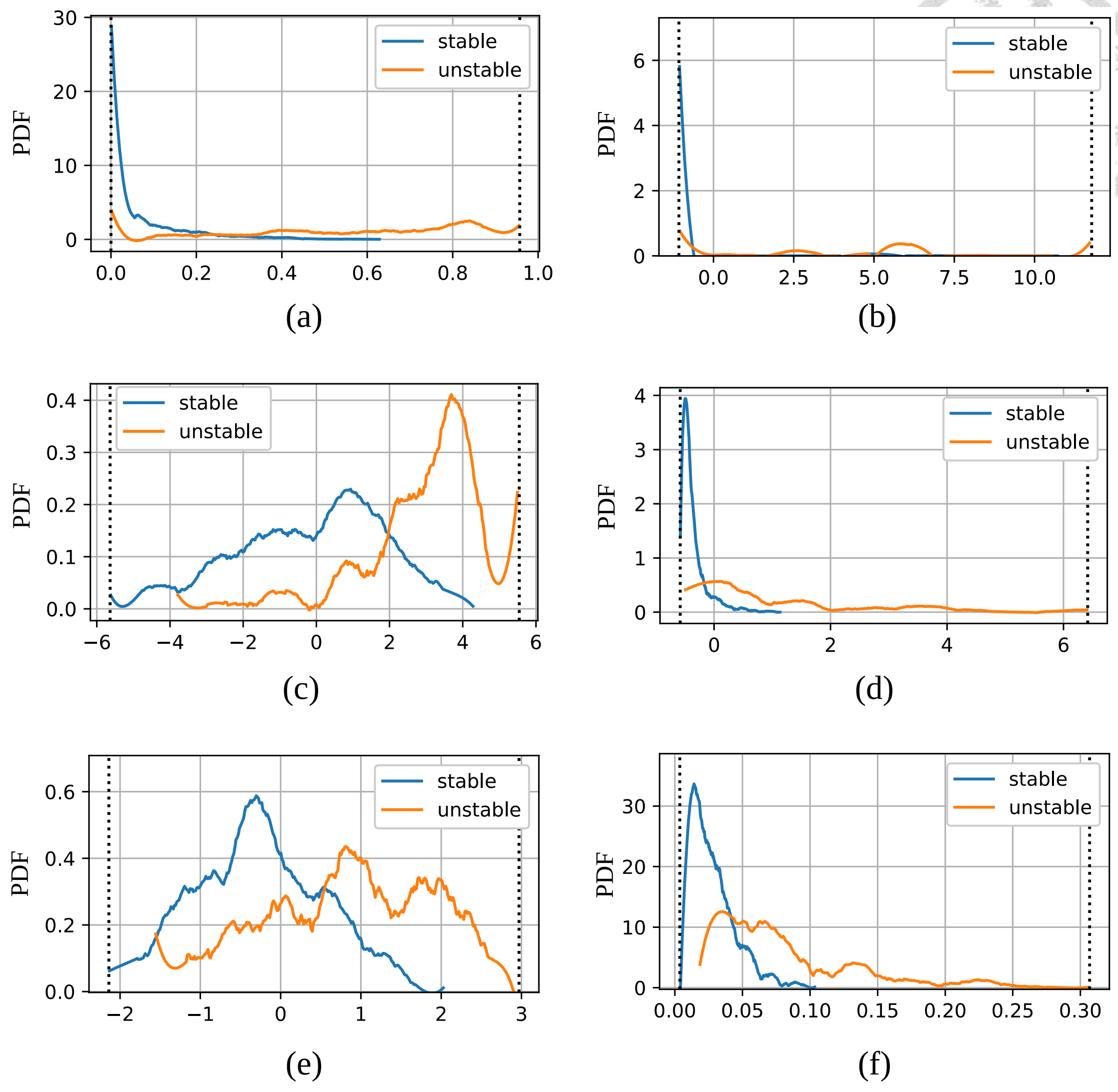


Fig. 40. Probability distribution function (PDF) of the *y*-axis features used in this research, for both stable and unstable categories, including (a) $E_{1,\,FFT}$, (b) $E_{2,\,FFT}$, (c) $E_{3,\,FFT}$, (d) E_{WPT}, (e) ϵ from autocorrelation coefficient, and (f) E_{HHT}

doi:10.6342/NTU201903636

Fig. 41 shows the lowest error rates for each type of feature. HHT, when used with WPT, has the best performance in terms of error rate, at 4.856%. WPT and $E_{1,FFT}$ are the second and third, at 6.019% and 6.335% respectively. Phase difference from autocorrelation coefficient is the worst, with an error rate of 16.42%, which is more than 3 times of the best error rate. For all six features, HHT is the only one that is based on empirical formulas and does not have a convincing theoretical background for chatter identification. It is surprising that it beats other methods with good theoretical foundations, although by a very small margin.

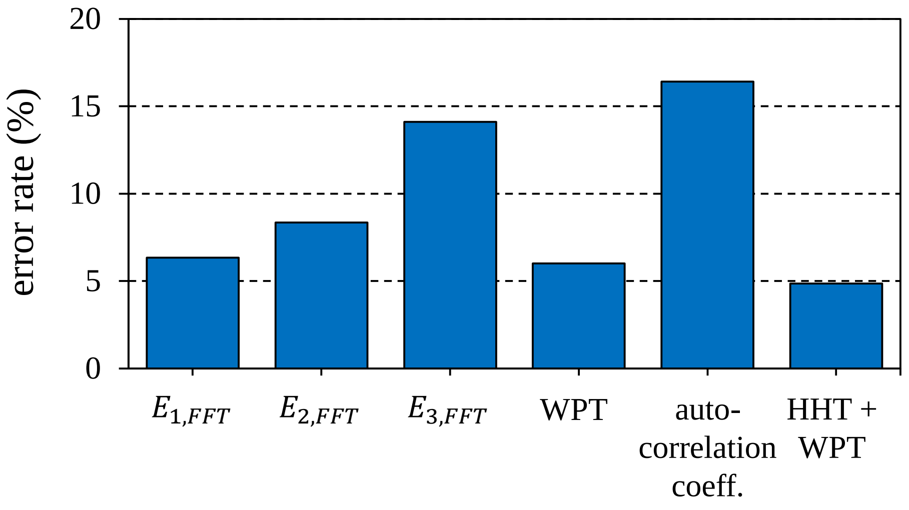


Fig. 41. Comparison of error rates of all features

doi:10.6342/NTU201903636

5.5 Effect of window size

As discussed previously in the sections regarding parameter optimization of FFT, WPT, and autocorrelation coefficient, the error rate generally decreases with increased windows size. This is likely due to the fluctuation of the feature within a window. E.g. a high value of energy ratio may appear promptly during a stable cut, causing it to be incorrectly classified as unstable. This can result in higher error rates when the window size is too small. However, a lower window size might have the benefit of quicker detection time in real-time chatter detection applications. This is a trade-off between error rate and detection time. In this section, the effect of window size on error rates and detection speeds is discussed, and the detection speeds of models trained with different features will be compared.

5.5.1 Error rates

Fig. 42 shows the relation between window size and error rate for 4 features, FFT, WPT, autocorrelation coefficient, and HHT. The window size varies from 0.05 seconds to 1 second and overlap is set to 50% of the window size for both training and test data. K-NN is used as the classification method with weight r^{-3} and the optimal k is selected between 1 and 200. The trends of the four figures are similar where the error rate is high when window size is below 0.2 seconds, and quickly decreasing as the window size increases to 0.3 to 0.5 seconds. It is worth noting that the error rate for HHT drops to 2.2% at window size of 0.37 seconds.

doi:10.6342/NTU201903636

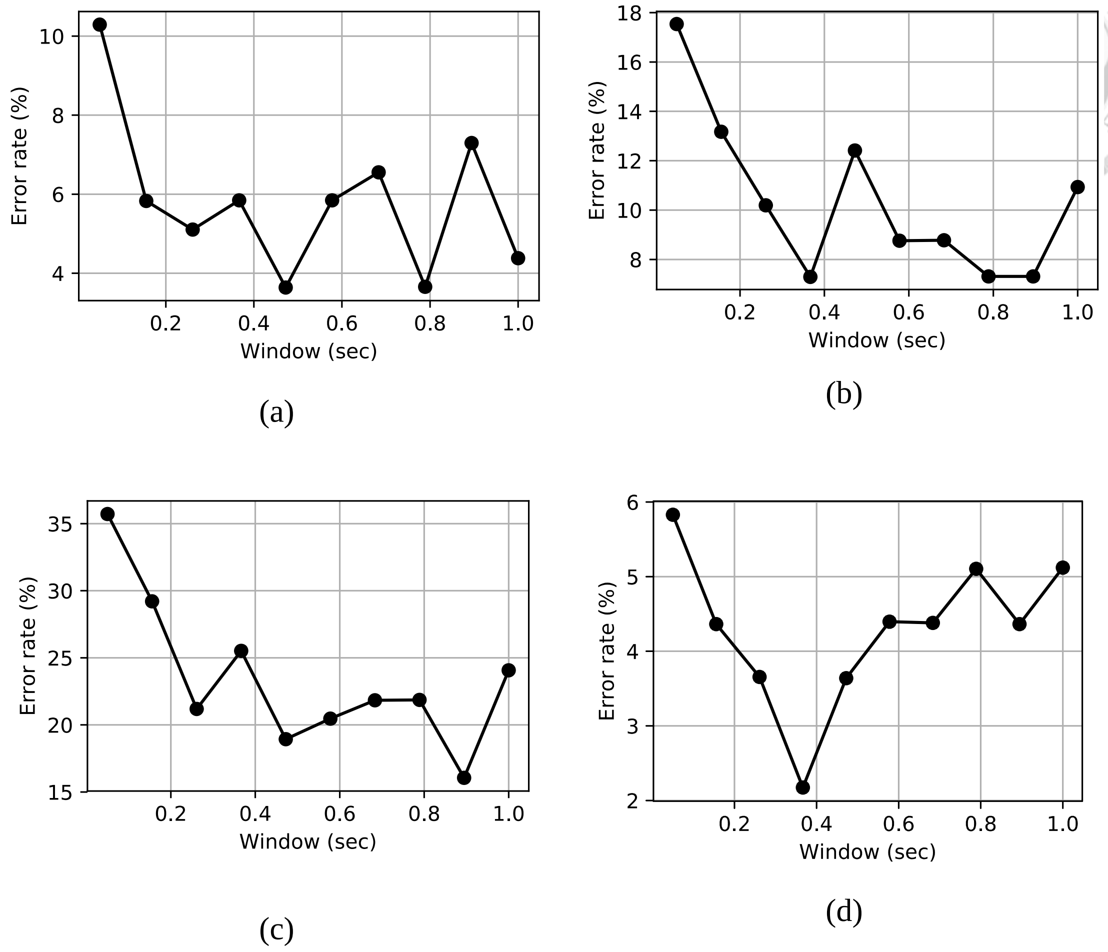


Fig. 42. Variation of error rates with respect to window size for (a) $E_{FFT,1}$, (b) WPT, (c) phase ε for autocorrelation coefficient, and (d) HHT

doi:10.6342/NTU201903636

5.5.2 Detection speed

Fig. 43 show the variation of detection speeds with different window sizes. The data points are the average detection speeds for all cuts, and the error bar indicates one standard deviation. The detected time is relative to the time where window size is 0.05 seconds, and lower value is better. Surprisingly, the detected time in general decreases as window size increases. We explain this using WPT as an example, with a small window size. During the training process, due to the inevitable fluctuations, the energy ratio feature needs to be high enough for the model to be considered as unstable. During testing, we must wait longer for the energy ratio to rise above the stable-unstable boundary so that the model identifies it as unstable. This may be the cause of slower detection for small window sizes.

Detected times between different features are also compared in Fig. 44. Autocorrelation coefficient produces the fastest detected time at window size of 0.47 seconds. HHT results in a slower detection, typically 0.2 to 0.3 seconds behind other three features. These detected times are the average of the entire dataset. For each unstable cut in the dataset, we simulate the time when the chatter would be detected using a trained model. These times are then averaged and compared with another model. Therefore, these detected times are relative, and is unrelated to the actual time of chatter occurrence.

doi:10.6342/NTU201903636

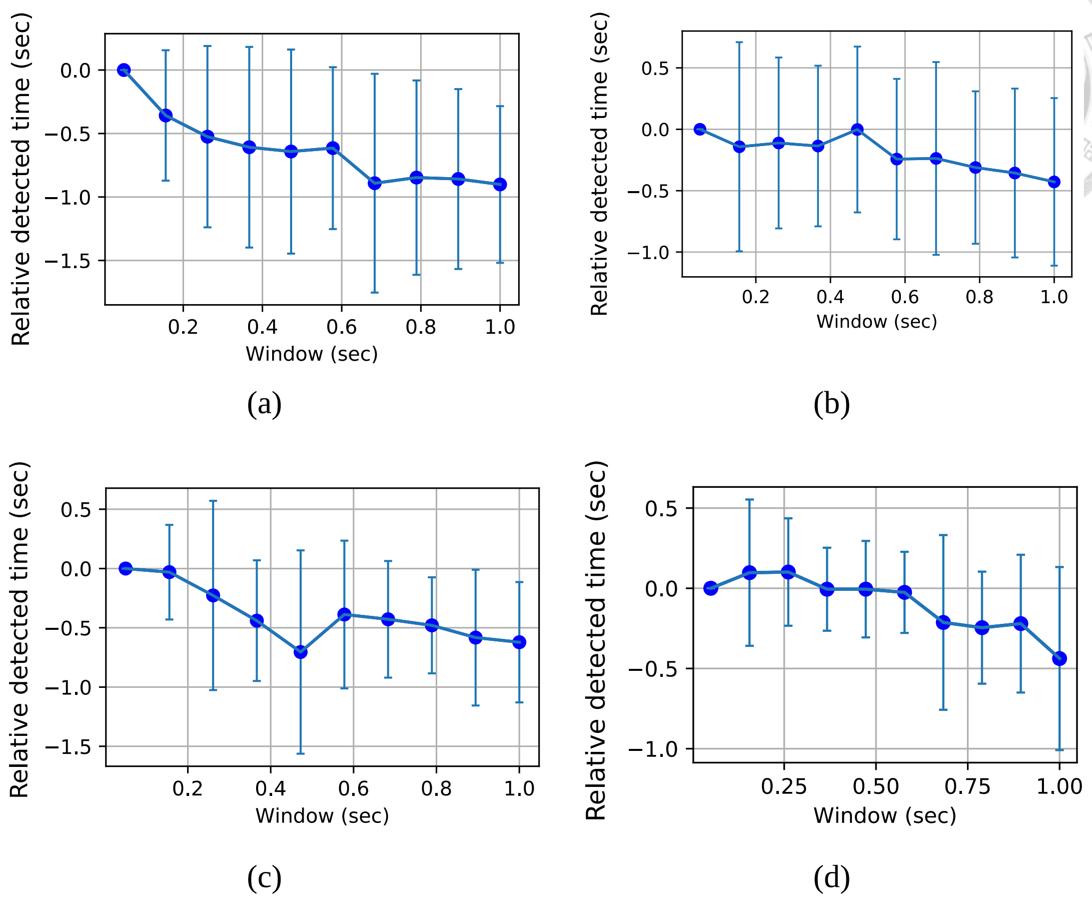


Fig. 43. Variation of detected time with respect to window size for (a) $E_{FFT,1}$, (b) WPT, (c) phase ε for autocorrelation coefficient, and (d) HHT.

doi:10.6342/NTU201903636

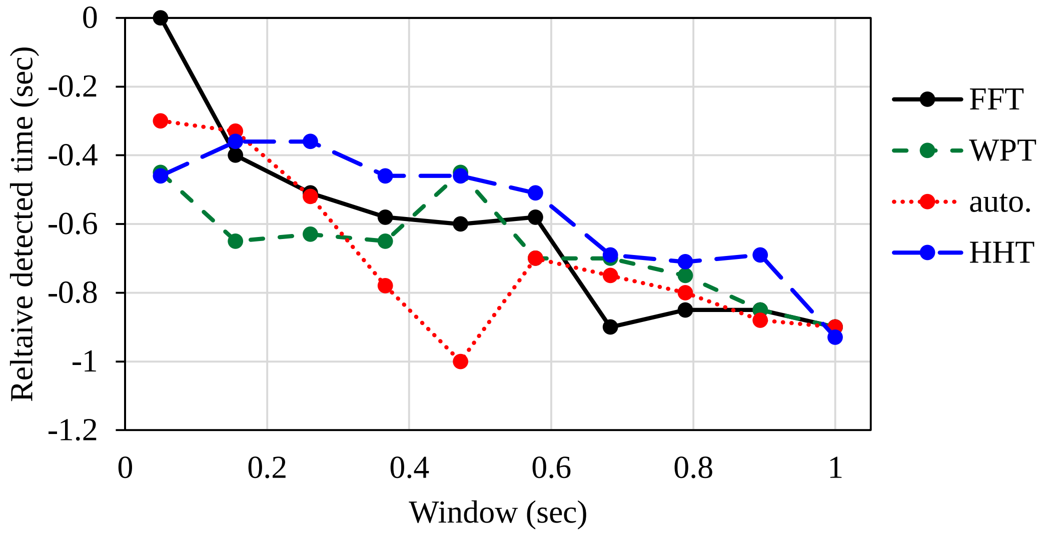


Fig. 44. Relative detected times for each different feature and window sizes for FFT ($E_{1,FFT}$), WPT, autocorrelation coefficient, and HHT (with WPT)

doi:10.6342/NTU201903636

6. Conclusions and future work

In this research, a chatter identification platform is developed to train models and evaluate their performance, using combinations of signal processing methods and classification algorithms and a dataset consisting of 143 cuts under various cutting condition. We compared several classification methods in terms of their performance on chatter identification after parameter optimization for each classifier. K-NN, Naïve Bayes, and SVM are the superior methods, with error rates from 5.21% to 5.647%. The effect on accuracy of feature selection is far more significant compared to classifier selection. Efforts are put into parameter optimizations for each of these features. Using the optimal classifier, k-NN, and window size of 1.2 seconds and no overlap, the optimal error rate is achieved by using HHT and WPT together, at 4.856%. The rest are E_{WPT}, $E_{1,FFT}$, $E_{2,FFT}$, $E_{3,FFT}$, and autocorrelation coefficient, ordered by error rate, from low to high. Autocorrelation coefficient proves to be the least effective, with an error rate of 16.42%. Incidentally, using all three axes as feature is shown to be much better than using only 1 or 2 axes in some circumstances. Finally, the effect of window size on error rates and detection speeds is also investigated using the platform we developed. A window size around 0.3 to 0.5 seconds is optimal in terms of error rate, and the best error rate of 2.2% was found using a window size of 0.37 seconds with HHT+WPT. However, HHT+WPT results in a marginally slower chatter detection compared to other features, and what comes as a surprise is that smaller window size does not lead to faster chatter detection.

There are some potential directions for future researches. Due to the large amount of possible variations in ANN architectures, it is not explored fully in this work and may be worth investigating. Another optimization opportunity is to use different window sizes for training and testing, and observe the trend of error rate and detection time.

doi:10.6342/NTU201903636

References

[1] Atanas Ivanov, Rebecca Leese, and Alexandre Spieser, Micromanufacturing Engineering and Technology (Second Edition), Micro and Nano Technologies, 2015.

[2] Ronald Faassen, "Chatter Prediction and Control for High-Speed Milling Modelling and Experiments," *Ph. D. thesis,* 2007.

[3] Guillem Quintana and Joaquim Ciurana, "Chatter in machining processes: A review," *International Journal of Machine Tools & Manufacture,* pp. 363-376, 2011.

[4] Erol Turkes, Sezan Orak, Suleyman Neseli, and Suleyman Yaldiz, "Linear analysis of chatter vibration and stability for orthogonal cutting in turning," *Int. Journal of Refractory Metals and Hard Materials,* pp. 163-169, 2011.

[5] J. Tlusty and F. Ismail, "Basic Non-Linearity in Machining Chatter," *CIRP Annals, vol. 30, no.1,* pp. 299-304, 1981.

[6] J. Tlusty, W. Zaton, and F. Ismail, "Stability Lobes in Milling," *CIRP Annals, vol. 32, no. 1,* pp. 309-313, 1983.

[7] Y. Altintaş and E. Budak, "Analytical Prediction of Stability Lobes in Milling," *CIRP Annals, vol. 44, no. 1,* pp. 357-362, 1995.

[8] R. P. H. Faassen, N. V. D. Wouw, J. Oosterling, and H. Nijmeijer, "Prediction of regenerative chatter by modelling and analysis of high-speed milling," *International Journal of Machine Tools and Manufacture, vol. 43, no. 14,* pp. 1437-1446, 2003.

doi:10.6342/NTU201903636

[9] E. Kuljanic, G.Totis, and M. Sortino, "Vibrations and Chatter in Machining: State of the Art and New Approaches," in *Advanced Manufacturing Systems and Technology*, 2008.

[10] E. Solis, C. Peres, J. Jiménez, J. Alique, and J. Monje, "A new analytical–experimental method for the identification of stability lobes in high-speed milling," *International Journal of Machine Tools and Manufacture, vol. 44, no. 15,* pp. 1591-1597, 2004.

[11] S. D. Merdol and Y. Altintas, "Multi Frequency Solution of Chatter Stability for Low Immersion Milling," *Journal of Manufacturing Science and Engineering, vol. 126, no. 3*, pp. 459-466, 2004.

[12] T. Insperger and G. Stépán, "Semi-discretization method for delayed systems," *International Journal for Numerical Methods in Engineering, vol. 55, no. 5,* pp. 503-518, 2002.

[13] J. Muñoa, M. Zatarain, Z. Dombovari, and Y. Yang, "Effect of Mode Interaction on Stability of Milling Processes," in *12th CIRP Conference on Modelling of Machining Operations*, San Sebastian, Spain, 2009.

[14] Z. Dombóvári, "Overview of stability analysis in machining processes," *Technical report*, 2008.

[15] Y. Altintas, "Analytical prediction of three dimensional chatter stability in milling," *JSME Int J C-Mech Syst, vol. 44, no. 3,* pp. 717-723, 2001.

doi:10.6342/NTU201903636

[16] Y. Altintas, E. Shamoto, P. Lee, and E. Budak, "Analytical prediction of stability lobes in ball end milling," *J Manuf Sci E-T ASME, vol. 121, no. 4,* pp. 586-592, 1999.

[17] A. Tang and Z. Liu, "Three-dimensional stability lobe and maximum material removal rate in end milling of thin-walled plate," *Int J Adv Manuf Technol, vol. 43, no. 1,* pp. 33-39, 2009.

[18] C. Toh, "Vibration analysis in high speed rough and finish milling hardened steel," *Journal of Sound and Vibration, vol. 278, no. 1-2,* pp. 101-115, 2004.

[19] Z. Han, H. Jin, M. Li, and H. Fu, "An open modular architecture controller based online chatter suppression system for CNC milling," *Mathematical Problems in Engineering,,* p. 13, 2015.

[20] M. C. Yoon and D. H. Chin, "Cutting force monitoring in the endmilling operation for chatter detection," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, vol. 219, no. 6,* pp. 455-465, 2005.

[21] 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.

[22] Y. Sun and Z. Xiong, "An Optimal Weighted Wavelet Packet Entropy Method With Application to Real-Time Chatter Detection," *IEEE/ASME Transactions on Mechatronics, vol. 21, no. 4,* pp. 2004-2014, 2016.

doi:10.6342/NTU201903636

[23] Y. Sun, C. Zhuang, and Z. Xiong, "Real-time chatter detection using the weighted wavelet packet entropy," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2014.

[24] G. G. Yen and K. C. Lin, "Wavelet packet feature extraction for vibration monitoring," *IEEE Transactions on Industrial Electronics, vol. 47, no. 3,* pp. 650-667, 2000.

[25] Melih C. Yesilli, Firas A. Khasawneh, and Andreas Otto, "On Transfer Learning For Chatter Detection in Turning Using Wavelet Packet Transform and Empirical Mode Decomposition," arXiv, 2019.

[26] B. S. Berger I. Minis, J. Harley, M. Rokni, and M. Paradopoulos, "Wavelet based cutting state identification," *Journal of Sound and Vibration, vol. 213, no. 5,* pp. 813-827, 1998.

[27] A. Ordaz-Moreno, R. de Jesus Romero-Troncoso, J. A. Vite-Frias, J. R. Rivera-Gillen, and A. Garcia-Perez, "Automatic online diagnosis algorithm for broken-bar detection on induction motors based on discrete wavelet transform for FPGA implementation," *IEEE Transactions on Industrial Electronics, vol. 55, no. 5,* pp. 2193-2202, 2008.

[28] E. Soliman and F. Ismail, "Chatter detection by monitoring spindle drive current," *The International Journal of Advanced Manufacturing Technology, vol. 13, no. 1,* pp. 27-34, 1997.

doi:10.6342/NTU201903636

[29] Deniz Aslan and Yusuf Altintas, "On-line chatter detection in milling using drive motor current commands extracted from CNC," *International Journal of Machine Tools and Manufacture, vol. 132,* 2018.

[30] R. Du, M. Elbestawi, and B. Ullagaddi, "Chatter detection in milling based on the probability distribution of cutting force signal," *Mechanical Systems and Signal Processing, vol. 6, no. 4,* pp. 345-362, 1992.

[31] S. Tangjitsitcharoen, "In-process monitoring and detection of chip formation and chatter for CNC turning," *Journal of Materials Processing Technology, vol. 209, no. 10,* pp. 4682-4688, 2009.

[32] 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.

[33] Yongjian Ji, Xibin Wang, Zhibing Liu, Hongjun Wang, Li Jiao, Dongqian Wang, and Shouyang 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.

[34] Wei Peng, Zhongju Hu, Li Yuan, and Pingyu Zhu, "Chatter identification using HHT for boring process," in *International Conference on Optical Instruments and Technology*, 2013.

[35] R. Q. Yang and X.Gao, "Hilbert-Huang transform-based vibration signal analysis for machine health monitoring," *IEEE Trans. Instrum. Meas., vol. 55, no. 6,* pp. 2320-2329, 2006.

doi:10.6342/NTU201903636

[36] Tang, J. P., Chiou, D. J., Chen, C. W., Chiang, W. L., Hsu, W. K., Chen, C. Y., and Liu, T. Y., "A case study of damage detection in benchmark buildings using a hilbert-huang transform-based method," *Journal of Vibration and Control, vol. 17, no. 4,* pp. 623-636, 2011.

[37] Michał Szydłowski and Bartosz Powałka, "Chatter detection algorithm based on machine vision," *Int J Adv Manuf Technol, vol. 65,* pp. 517-528, 2011.

[38] Zhenga H, Kongb LX, and Nahavandia S, "Automatic inspection of metallic surface defects using genetic algorithms," *Journal of Materials Processing Technology, vol. 125,* pp. 427-433, 2007.

[39] Lee BY and Tarng YS, "Surface roughness inspection by computer vision in turning operations," *Int J Mach Tool Manuf 41,* pp. 1251-1263, 2007.

[40] Ching-Chih Wei, Meng-Kun Liu, and Guo-Hua Huang, "Chatter Identification of Face Milling Operation via Time-Frequency and Fourier Analysis," *International Journal of Automation and Smart Technology,* pp. 25-36, 2016.

[41] X. Q. Li, Y. S. Wong, and A. Y. C. Nee, "A Comprehensive Identification of Tool Failure and Chatter Using a Parallel Multi-ART2 Neural Network," *Journal of Manufacturing Science and Engineering, vol. 120, no. 2,* p. 433, 1998.

[42] 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.

doi:10.6342/NTU201903636

[43] J. Hino, S. Okubo, and T. Yoshimura, "Chatter Prediction in End Milling by FNN Model with Pruning," *JSME International Journal Series C, vol. 49, no. 3,* pp. 742-749, 2006.

[44] Yang Y, Yu D, and Cheng J, "A fault diagnosis approach for roller bearing based on IMF envelope spectrum and SVM," *Measurement, no. 40, vol. 9–10,* pp. 943-950, 2007.

[45] Tan F, Yin M, Wang L, and Yin G, "Spindle thermal error robust modeling using LASSO and LS-SVM," *Int J Adv Manuf Technol, no. 94, vol. 5,* pp. 2861-2874, 2018.

[46] Bhat NN, Dutta S, Vashisth T, Pal S, Pal SK, and Sen R, "Tool condition monitoring by SVM classification of machined surface images in turning," *Int J Adv Manuf Technol, vol. 83, no. 9,* pp. 1487-1502, 2016.

[47] Y.-C. Yao, Real-time Chatter Detection, Analysis and Suppression Using in Intelligent Spindles Based on One-class Support Vector Machine and Local Outlier Factor (Master thesis), 2018.

[48] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander, "LOF: Identifying Density-Based Local Outliers," in *Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data*, Dalles, TX, 2000.

[49] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, "A Practical Guide to Support Vector Classification," *Techinical report,* 2003.

doi:10.6342/NTU201903636

[50] S. Tangjitsitcharoen, "Analysis of Chatter in Ball End Milling by Wavelet Transform," *International Journal of Industrial and Manufacturing Engineering, vol. 6, no. 11,* pp. 2438-2444, 2012.

[51] R. Brancati, E. Rocca, S. Savino, and F. Farroni, "Gear Rattle Analysis Based on Wavelet Signal Decomposition," in *Proceedings of the ASME 2012 11th Biennial Conference On Engineering Systems Design And Analysis*, 2012.

[52] Sami Ekici, Selcuk Yildirim, and Mustafa Poyraz, "Energy and entropy-based feature extraction for locating fault on transmission lines by using neural network and wavelet packet decomposition," *Expert Systems with Applications, Volume 34, Issue 4,* 2008.

[53] Shaoke Wan, Xiaohu Li, Wei Chen, and Jun Hong, "Investigation on milling chatter identification at early stage with variance ratio and Hilbert–Huang transform," *The International Journal of Advanced Manufacturing Technology, Volume 95, Issue 9–12,* pp. 3563-3573, 2017.

[54] Nayana Gandhi, "FFT based evaluation of cutting forces and chatter vibrations in turning by varying speed, feed, depth of cut and rake angle.," in *GIT-Journal of Engineering and Technology* , 2012.

[55] Haosheng Li, Bin Wu, and Hubert Kratz, "FFT and Wavelet-Based Analysis of the Influence of Machine Vibrations on Hard Turned Surface Topographies," *Tsinghua Science & Technology, vol. 12, no. 4,* pp. 441-446, 2007.

[56] F.B.J.W.M. Hendriks, "Chatter detection in high-speed milling," *Technical report,* 2005.

doi:10.6342/NTU201903636

[57] L.R. Soares, H.M. de Oliveira, R.J.S. Cintra, and R.M. Campello de Souza, "Fourier Eigenfunctions, Uncertainty Gabor PrincipleAnd Isoresolution Wavelets," in *XX Simpósio Brasileiro de Telecomunicações*, Rio de Janeiro, 2003.

[58] Introduction to Wavelet Families (MATLAB online documentation).

[59] F. Safara, S. Doraisamy, A. Azman, A. Jantan, and S. Ranga, "Wavelet Packet Entropy for Heart Murmurs Classification," *Advances in Bioinformatics,* pp. 1-6, 2012.

[60] Eiji Kondo, *US 9285797 B2 (U.S. Patent),* 2016.

[61] S. D. Merdol and Y. Altintas, "Multi Frequency Solution of Chatter Stability for Low Immersion Milling," *Journal of Manufacturing Science and Engineering, vol. 126, no. 3,* pp. 459-466, 2004.

[62] Tamas Insperger and Gabor Stepan, "Semi-discretization method for delayed systems," *International Journal for Numerical Methods in Engineering, vol. 55, no. 5,* pp. 503-518, 2002.

[63] Norden E. Huang and Zhaohua Wu, "A review on Hilbert-Huang transform: Method and its applications to geophysical studies," *Reviews of Geophysics, Volume 46, Issue 2,* 2008.

[64] Norden E Huang and Samuel S P Shen, Hilbert-Huang Transform and Its Applications, World Scientific, 2005.

[65] Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H. Liu, The empirical

doi:10.6342/NTU201903636

mode decomposition and theHilbert spectrum for nonlinear andnon-stationary time series analysis, Royal Society, 1996.

[66] Mathias Johansson, "The Hilbert transform," *Technical report.*

[67] Timothy J. Ulrich, "Envelope Calculation from the Hilbert Transform," *Techinical report,* 2006.

[68] Jason D. M. Rennie, Lawrence Shih, Jaime Teevante, and David R. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," in *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[69] Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.

[70] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell, "Learning to classify text from labeled and unlabeled documents," in *Association for the Advancement of Artificial Intelligence*, 1998.

[71] Schubert, E.; Zimek, A.; Kriegel, H. -P., "Local outlier detection reconsidered: A generalized view on locality with applications to spatial, video, and network outlier detection," *Data Mining and Knowledge Discovery,* 2012.

[72] Yung-Chen Yao, Yu-Hsuan Chen, Chien-Hao Liu, Wen-Pin Shih, "Real-time chatter detection and automatic suppression for intelligent spindles based on wavelet packet energy entropy and local outlier factor algorithm," *The International Journal of Advanced Manufacturing Technology,* pp. 1-13, 2019.

doi:10.6342/NTU201903636

[73] N.Y. Deng, Y.J. Tian, and C.H. Zhang, Support Vector Machines: Algorithms and Extensions, CRC Press, 2012.

[74] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and other Kernel based Learning Methods, Cambridge University Press, 2000.

[75] Chen Bing, Yang Ji, Zhao Ju, and Ren Jingbo, "Milling Chatter Prediction Based on the Information Entropy and Support Vector Machine," in *International Industrial Informatics and Computer Engineering Conference*, 2015.

[76] 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, Volume 95, Issue 1–4,* pp. 775-784, 2018.

[77] Yongqing Wang, Qile Bo, Haibo Liu, Lei Hu, and Hao Zhang, "Mirror milling chatter identification using Q-factor and SVM," *The International Journal of Advanced Manufacturing Technology, vol. 98, issue 5–8,* pp. 1163-1177, 2018.

[78] R. Berwick and Village Idiot, "An Idiot's guide to Support vector machines (SVMs)," *Technical report.*

[79] Tristan Fletcher, "Support Vector Machines Explained," *Technical report.*

[80] Altman, N. S., "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician,* pp. 175-185, 1992.

doi:10.6342/NTU201903636

Appendix A. List of cutting conditions in the dataset

Note: The chip load is fixed at 0.1 mm/tooth.

Experimentally stable cutting conditions:

Spindle speed (rpm)	Depth of cut (mm)	Spindle speed (rpm)	Depth of cut (mm)
4500	0.2	6100	0.5
4500	0.3	6100	0.6
4600	0.2	6200	0.2
4900	0.3	6200	0.3
5000	0.2	6200	0.4
5000	0.3	6200	0.5
5000	0.4	6200	0.6
5000	0.5	6200	0.7
5000	0.6	6200	0.8
5000	0.7	6200	0.9
5000	0.8	6200	1
5000	0.9	6300	0.2
5000	1	6300	0.3
5100	0.2	6300	0.4
5100	0.3	6300	0.5
5100	0.4	6300	0.6
5100	0.5	6300	0.7
5100	0.6	6400	0.2
5200	0.2	6400	0.3
5200	0.3	6400	0.4
5200	0.4	6400	0.5
5300	0.2	6400	0.6
5300	0.3	6500	0.2
5400	0.2	6500	0.3
5500	0.2	6500	0.4
5500	0.2	6500	0.5
5600	0.2	6500	0.6
5700	0.2	6600	0.2
5800	0.2	6600	0.3
5900	0.2	6600	0.4
5900	0.3	6700	0.2
6000	0.2	6700	0.3
6000	0.3	6700	0.4
6000	0.4	6800	0.2
6100	0.2	6800	0.3
6100	0.3	6900	0.2
6100	0.4	7000	0.2

doi:10.6342/NTU201903636

Experimentally unstable cutting conditions:

Spindle speed (rpm)	Depth of cut (mm)	Spindle speed (rpm)	Depth of cut (mm)
4500	0.32	5900	0.4
4500	0.4	5900	0.5
4600	0.3	6000	0.5
4700	0.2	6000	0.6
4700	0.3	6100	0.7
4800	0.2	6100	0.8
4800	0.3	6200	1.2
4900	0.4	6300	0.8
5000	1.1	6300	0.9
5100	0.7	6400	0.7
5200	0.5	6500	0.6
5200	0.6	6500	0.62
5300	0.3	6500	0.7
5300	0.4	6600	0.5
5400	0.3	6700	0.4
5500	0.3	6700	0.5
5600	0.3	6800	0.3
5700	0.2	6800	0.4
5700	0.3	6900	0.3
5800	0.3	7000	0.3

doi:10.6342/NTU201903636

Appendix B. Model training and validation results

Note: The number in the parenthesis indicates the maximum error rate of the 3 validation datasets (in stratified k-fold validation with *k*=3). The other number is the average error rate of the three datasets.

Examples:

(a) Auto. Coeff., window=1.2, overlap=0.8, prominence=50%, a, xyz: The feature is autocorrelation coefficient, window size 1.2 sec, overlap 0.8 sec, prominence 50%, using *x*-, *y*-, and *z*-axes acceleration as feature.

(b) kNN, k=auto[1-200], weight 1/r^3: The classification method is k-nearest neighbors, with k automatically selected between 1 to 200 for the lowest error rate, with weight 1/r^3.

(c) FFT, exp=2.75, n=5, d=100, bw=10: exponent 2.75, tooth pass filter bandwidth 10Hz, only 5 highest peaks are used, with the minimum distance between peaks being 100 Hz.

Feature		Classification		data points	FA (%)	MA %)	ER (%)
Auto. Coeff.	window=1.2, overlap=0.8, prominence=50%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	6.961 (11.03)	11.01 (14.96)	17.97 (18.97)
Auto. Coeff.	window=0.9, overlap=0.6, prominence=50%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	1736	7.258 (10.44)	10.79 (14.54)	18.05 (21.37)
Auto. Coeff.	window=0.6, overlap=0.4, prominence=50%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	2780	6.1 (10.53)	11.24 (14.62)	17.34 (19.78)
Auto. Coeff.	window=0.3, overlap=0.2, prominence=50%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	5895	6.493 (9.336)	11.48 (16.49)	17.97 (23.54)
Auto. Coeff.	window=1.2, overlap=0, prominence=50%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	456	6.895 (13.51)	10.03 (13.42)	16.93 (18.24)
Auto. Coeff.	window=0.9, overlap=0, prominence=50%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	608	6.806 (11.68)	12.33 (13.43)	19.13 (22.84)
Auto. Coeff.	window=0.6, overlap=0, prominence=50%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	972	6.361 (9.873)	12.57 (16.41)	18.93 (23.53)
Auto. Coeff.	window=0.3, overlap=0, prominence=50%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	2010	6.357 (9.035)	10.96 (12.59)	17.31 (17.57)
Auto. Coeff.	window=0.1, overlap=0, prominence=50%, a, xyz	kNN	k=auto[1-2000], weight 1/r^3	6169	6.654 (8.674)	12.08 (14.57)	18.74 (19.79)
Auto. Coeff.	window=0.05, overlap=0, prominence=50%, a, xyz	kNN	k=auto[1-2000], weight 1/r^3	12338	8.793 (11.8)	14.01 (15.78)	22.8 (23.05)
Auto. Coeff.	window=0.9, overlap=0.6, prominence=10%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	1736	7.137 (14.01)	11.45 (16.99)	18.58 (21.59)
Auto. Coeff.	window=0.9, overlap=0.6, prominence=30%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	1736	7.39 (9.591)	11.58 (14.12)	18.97 (19.79)

doi:10.6342/NTU201903636

Auto. Coeff.	window=0.9, overlap=0.6, prominence=70%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	1736	6.805 (10.05)	10.66 (13.38)	17.46 (17.99)
Auto. Coeff.	window=0.9, overlap=0.6, prominence=90%, a, xyz	kNN	k=auto[1-200], weight 1/r^3	1736	6.16 (10.12)	11.77 (14.59)	17.93 (18.8)
Auto. Coeff.	window=0.9, overlap=0.6, prominence=50%, v, xyz	kNN	k=auto[1-200], weight 1/r^3	1736	7.548 (10.55)	8.868 (11.05)	16.42 (20.18)
Auto. Coeff.	window=0.9, overlap=0.6, prominence=50%, x, xyz	kNN	k=auto[1-200], weight 1/r^3	1736	8.846 (9.615)	9.11 (9.464)	17.96 (18.71)
FFT	window=1.2, overlap=0.8, exp=0.5, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	8.539 (9.569)	12.25 (14.25)	20.79 (21.77)
FFT	window=1.2, overlap=0.8, exp=0.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	6.389 (11.67)	12.07 (14.29)	18.46 (20.71)
FFT	window=1.2, overlap=0.8, exp=1, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	4.007 (8.235)	8.993 (11.5)	13 (15.53)
FFT	window=1.2, overlap=0.8, exp=1.25, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.228 (6.921)	6.969 (8.685)	10.2 (14.32)
FFT	window=1.2, overlap=0.8, exp=1.5, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.846 (4.513)	6.136 (7.601)	7.982 (12.11)
FFT	window=1.2, overlap=0.8, exp=1.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.601 (3.294)	5.777 (8.706)	7.379 (12)
FFT	window=1.2, overlap=0.8, exp=2, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.01 (4.245)	5.439 (6.667)	7.449 (8.726)
FFT	window=1.2, overlap=0.8, exp=2.25, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.318 (5.176)	5.525 (7.5)	7.843 (9.647)
FFT	window=1.2, overlap=0.8, exp=2.5, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.332 (5.489)	5.93 (6.361)	8.262 (10.98)
FFT	window=1.2, overlap=0.8, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.808 (2.778)	5.067 (6.921)	6.875 (9.069)
FFT	window=1.2, overlap=0.8, exp=3, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.441 (3.505)	4.953 (9.813)	7.393 (13.32)
FFT	window=1.2, overlap=0.8, exp=3.5, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.643 (6.161)	4.334 (5.45)	6.977 (11.61)
FFT	window=1.2, overlap=0.8, exp=4, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.22 (7.399)	4.45 (5.303)	7.67 (11.46)
FFT	window=1.2, overlap=0.8, exp=5, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.201 (3.066)	4.72 (5.5)	6.92 (7.25)
FFT	window=1.2, overlap=0.8, exp=6, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.089 (4.225)	4.628 (5.736)	6.717 (7.746)
FFT	window=1.2, overlap=0.8, exp=8, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.743 (5.201)	5.375 (8.312)	8.118 (9.848)
FFT	window=1.2, overlap=0.8, exp=10, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.506 (5.742)	4.78 (5.955)	8.286 (9.569)

doi:10.6342/NTU201903636

FFT	window=1.2, overlap=0.8, exp=2.75, bw=2.5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.449 (3.58)	5.216 (7.635)	6.665 (7.635)
FFT	window=1.2, overlap=0.8, exp=2.75, bw=5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.801 (2.837)	5.528 (7.565)	7.329 (10.4)
FFT	window=1.2, overlap=0.8, exp=2.75, bw=20, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.439 (5.314)	5.172 (7.488)	8.611 (12.8)
FFT	window=1.2, overlap=0.8, exp=2.75, bw=40, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.621 (3.095)	4.831 (6.112)	6.451 (7.857)
FFT	window=0.9, overlap=0.6, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1736	1.721 (2.152)	5.851 (7.719)	7.572 (8.772)
FFT	window=0.6, overlap=0.4, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	2780	2.455 (3.459)	5.818 (8.491)	8.273 (11.95)
FFT	window=0.3, overlap=0.2, exp=2.75, bw=10, xyz	kNN	k=auto[1-500], weight 1/r^3	5895	3.582 (5.143)	7.484 (11.05)	11.07 (12.51)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	456	1.705 (4.459)	4.63 (6.579)	6.335 (7.237)
FFT	window=0.9, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	608	1.94 (3.81)	5.862 (8.867)	7.802 (10.48)
FFT	window=0.6, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	972	2.679 (3.481)	5.614 (8.358)	8.293 (11.04)
FFT	window=0.3, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	2010	2.917 (4.512)	7.107 (8.006)	10.02 (12.52)
FFT	window=0.1, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-500], weight 1/r^3	6169	3.803 (5.082)	9.124 (9.545)	12.93 (14.16)
FFT	window=0.05, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-2000], weight 1/r^3	12338	3.827 (4.573)	9.895 (10.83)	13.72 (15.4)
FFT	window=1.2, overlap=0.8, exp=2.75, n=5, d=10, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.906 (5.45)	4.907 (6.234)	7.813 (11.14)
FFT	window=1.2, overlap=0.8, exp=2.75, n=5, d=50, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.531 (3.286)	3.98 (5.634)	6.511 (8.92)
FFT	window=1.2, overlap=0.8, exp=2.75, n=5, d=100, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.108 (3.791)	4.371 (7.109)	7.479 (10.9)
FFT	window=1.2, overlap=0.8, exp=2.75, n=10, d=10, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.581 (5.164)	4.645 (8.291)	8.226 (10.55)
FFT	window=1.2, overlap=0.8, exp=2.75, n=10, d=50, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.565 (7.952)	3.871 (5.911)	7.436 (10.36)

doi:10.6342/NTU201903636

FFT	window=1.2, overlap=0.8, exp=2.75, n=10, d=100, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	8.539 (15.71)	4.297 (6.683)	12.84 (18.1)
FFT	window=1.2, overlap=0.8, exp=2.75, n=30, d=10, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.811 (5.647)	5.017 (7.214)	7.829 (8.458)
FFT	window=1.2, overlap=0.8, exp=2.75, n=30, d=50, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.003 (4.235)	4.437 (7.765)	6.441 (12)
FFT	window=1.2, overlap=0.8, exp=2.75, n=30, d=100, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.926 (4.502)	5.065 (7.214)	6.991 (7.214)
c/tp	window=1.2, overlap=0.8, exp=2.75, n=30, range=5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.295 (2.716)	6.154 (7.092)	8.449 (9.456)
c/tp	window=1.2, overlap=0.8, exp=2.75, n=20, range=5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.081 (5.213)	5.515 (6.733)	8.596 (10.19)
c/tp	window=1.2, overlap=0.8, exp=2.75, n=10, range=5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.251 (4.481)	6.107 (10.61)	8.358 (10.86)
c/tp	window=1.2, overlap=0.8, exp=2.75, n=5, range=5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.53 (3.991)	6.466 (7.277)	9.996 (11.27)
n/all	window=1.2, overlap=0.8, exp=2.75, n=5, range=5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	6.524 (9.39)	8.713 (11.06)	15.24 (17.84)
n/all	window=1.2, overlap=0.8, exp=2.75, n=10, range=5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	6.07 (11.37)	8.051 (8.294)	14.12 (19.67)
n/all	window=1.2, overlap=0.8, exp=2.75, n=20, range=5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	6.819 (7.769)	8.049 (8.772)	14.87 (16.54)
n/all	window=1.2, overlap=0.8, exp=2.75, n=30, range=5, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	6.618 (8.983)	8.203 (9.456)	14.82 (18.44)
WPT	window=1.2, overlap=0.8, exp=2.75, haar, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.934 (3.318)	5.19 (6.923)	7.124 (9.242)
WPT	window=1.2, overlap=0.8, exp=2.75, coif1, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.25 (4.481)	4.953 (6.84)	7.204 (11.32)
WPT	window=1.2, overlap=0.8, exp=2.75, coif2, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.89 (7.16)	6.084 (9.25)	8.974 (12.89)
WPT	window=1.2, overlap=0.8, exp=2.75, coif3, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.649 (5.674)	5.475 (6.856)	8.125 (12.53)
WPT	window=1.2, overlap=0.8, exp=2.75, coif4, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.245 (5.226)	6.229 (7.601)	9.473 (12.83)
WPT	window=1.2, overlap=0.8, exp=2.75, coif8, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.066 (5.938)	6.095 (9.181)	9.162 (11.17)

doi:10.6342/NTU201903636

WPT	window=1.2, overlap=0.8, exp=2.75, coif12, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.955 (6.047)	6.066 (6.512)	9.021 (12.56)
WPT	window=1.2, overlap=0.8, exp=2.75, sym2, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.917 (3.981)	5.327 (6.089)	7.244 (10.07)
WPT	window=1.2, overlap=0.8, exp=2.75, sym3, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.651 (5.14)	4.955 (7.557)	7.606 (11.68)
WPT	window=1.2, overlap=0.8, exp=2.75, sym4, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.415 (4.717)	5.671 (8.255)	8.085 (12.97)
WPT	window=1.2, overlap=0.8, exp=2.75, sym5, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.232 (1.546)	6.517 (8.837)	7.749 (10.23)
WPT	window=1.2, overlap=0.8, exp=2.75, sym6, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.122 (7.857)	5.1 (7.654)	8.222 (11.67)
WPT	window=1.2, overlap=0.8, exp=2.75, sym10, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.301 (2.709)	5.659 (6.888)	7.96 (9.026)
WPT	window=1.2, overlap=0.8, exp=2.75, sym20, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.706 (6.604)	5.927 (6.7)	8.633 (12.03)
WPT	window=1.2, overlap=0.8, exp=2.75, bior1.3, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.958 (7.637)	5.492 (6.931)	8.45 (13.37)
WPT	window=1.2, overlap=0.8, exp=2.75, bior1.5, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.393 (1.733)	5.279 (7.765)	6.672 (9.176)
WPT	window=1.2, overlap=0.8, exp=2.75, bior2.2, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.683 (7.277)	5.719 (7.96)	8.402 (13.38)
WPT	window=1.2, overlap=0.8, exp=2.75, bior2.4, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.781 (9.33)	5.958 (7.379)	9.739 (13.16)
WPT	window=1.2, overlap=0.8, exp=2.75, bior3.1, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.06 (4.976)	6.786 (9.61)	9.845 (13.03)
WPT	window=1.2, overlap=0.8, exp=2.75, rbio1.3, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.432 (4.481)	5.484 (8.019)	7.916 (12.5)
WPT	window=1.2, overlap=0.8, exp=2.75, rbio1.5, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.773 (7.059)	5.071 (6.203)	7.844 (9.882)
WPT	window=1.2, overlap=0.8, exp=2.75, rbio2.2, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.13 (9.39)	5.137 (7.463)	8.267 (12.44)

doi:10.6342/NTU201903636

WPT	window=1.2, overlap=0.8, exp=2.75, rbio2.4, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.218 (2.148)	6.116 (9)	7.334 (9.75)
WPT	window=1.2, overlap=0.8, exp=2.75, rbio3.1, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.747 (2.338)	4.887 (10.07)	6.634 (11.24)
WPT	window=1.2, overlap=0.8, exp=2.75, db1, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.255 (5.516)	5.616 (8.148)	7.871 (8.889)
WPT	window=1.2, overlap=0.8, exp=2.75, db2, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.461 (7.329)	5.611 (7.731)	9.072 (11.58)
WPT	window=1.2, overlap=0.8, exp=2.75, db3, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.01 (7.009)	5.739 (6.983)	8.749 (12.85)
WPT	window=1.2, overlap=0.8, exp=2.75, db4, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.477 (5.924)	5.992 (6.818)	8.469 (12.56)
WPT	window=1.2, overlap=0.8, exp=2.75, db10, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.935 (3.271)	5.156 (8.879)	7.091 (12.15)
WPT	window=1.2, overlap=0.8, exp=2.75, db15, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.929 (7.26)	5.539 (8.586)	8.468 (11.48)
WPT	window=1.2, overlap=0.8, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	1.13 (2.118)	4.89 (6.824)	6.019 (8.941)
WPT	window=1.2, overlap=0.8, exp=2.75, db30, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	3.089 (8.491)	5.87 (7.235)	8.958 (13.68)
WPT	window=1.2, overlap=0.8, exp=2.75, dmey, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.989 (5.425)	6.321 (6.84)	9.311 (12.26)
WPT	window=0.9, overlap=0.6, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	1736	3.094 (6.811)	6.15 (9.683)	9.244 (10.63)
WPT	window=0.6, overlap=0.4, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	2780	3.266 (7.239)	6.391 (8.705)	9.657 (11.27)
WPT	window=0.3, overlap=0.2, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-500], weight 1/r^3	5895	3.375 (5.891)	6.975 (8.277)	10.35 (14.17)
WPT	window=1.2, overlap=0, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	456	1.287 (2.5)	5.883 (8.054)	7.17 (9.375)
WPT	window=0.9, overlap=0, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	608	2.936 (5.742)	6.412 (7.656)	9.347 (13.4)
WPT	window=0.6, overlap=0, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	972	3.248 (6.928)	6.464 (7.53)	9.712 (14.46)
WPT	window=0.3, overlap=0, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	2010	3.825 (7.648)	7.688 (10.63)	11.51 (12.84)
WPT	window=0.1, overlap=0, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-500], weight 1/r^3	6169	4.325 (5.477)	9.005 (11.35)	13.33 (14.69)
WPT	window=0.05, overlap=0, exp=2.75, db20, bw=10, xyz	kNN	k=auto[1-2000], weight 1/r^3	12338	4.531 (5.464)	10.41 (12.65)	14.94 (18.12)

doi:10.6342/NTU201903636

WPT	window=1.2, overlap=0.8, exp=2.75, db20, bw=10, x	kNN	k=auto[1-200], weight 1/r^3	1216	8.892 (13.7)	8.843 (11.25)	17.73 (21.15)
WPT	window=1.2, overlap=0.8, exp=2.75, db20, bw=10, y	kNN	k=auto[1-200], weight 1/r^3	1216	9.433 (13.46)	8.678 (11.76)	18.11 (22.6)
WPT	window=1.2, overlap=0.8, exp=2.75, db20, bw=10, z	kNN	k=auto[1-200], weight 1/r^3	1216	8.077 (10.34)	5.441 (6.971)	13.52 (17.31)
WPT	window=1.2, overlap=0.8, exp=2.75, db20, bw=10, xy	kNN	k=auto[1-200], weight 1/r^3	1216	5.152 (12.5)	5.654 (8.951)	10.81 (15.87)
WPT	window=1.2, overlap=0.8, exp=2.75, db20, bw=10, xz	kNN	k=auto[1-200], weight 1/r^3	1216	4.849 (9.856)	6.501 (8.173)	11.35 (18.03)
WPT	window=1.2, overlap=0.8, exp=2.75, db20, bw=10, yz	kNN	k=auto[1-200], weight 1/r^3	1216	5.449 (8.654)	5.856 (7.161)	11.3 (14.9)
HHT	window=1.2, overlap=0.8, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	4.18 (5.952)	8.043 (9.383)	12.22 (14.05)
HHT	window=1.2, overlap=0.8, w/ WPT, xyz	kNN	k=auto[1-200], weight 1/r^3	1216	2.049 (3.465)	2.907 (5.437)	4.957 (7.092)
HHT	window=1.2, overlap=0.8, w/ WPT, x	kNN	k=auto[1-200], weight 1/r^3	1216	3.497 (4.95)	2.291 (3.073)	5.788 (7.329)
HHT	window=1.2, overlap=0.8, w/ WPT, y	kNN	k=auto[1-200], weight 1/r^3	1216	10.52 (17.02)	11.31 (16.2)	21.83 (24.35)
HHT	window=1.2, overlap=0.8, w/ WPT, z	kNN	k=auto[1-200], weight 1/r^3	1216	3.977 (5.437)	4.613 (7.565)	8.59 (13)
HHT	window=1.2, overlap=0.8, w/ WPT, xy	kNN	k=auto[1-200], weight 1/r^3	1216	3.045 (3.599)	2.287 (3.31)	5.332 (6.619)
HHT	window=1.2, overlap=0.8, w/ WPT, xz	kNN	k=auto[1-200], weight 1/r^3	1216	2.765 (5.693)	2.092 (4.019)	4.856 (7.178)
HHT	window=1.2, overlap=0.8, w/ WPT, yz	kNN	k=auto[1-200], weight 1/r^3	1216	3.474 (6.147)	4.383 (6.619)	7.857 (12.77)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	Thres-hold		456	3.289	3.289	6.579
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	LOF	n=auto[1-200], 10% unstable	456			10.4 (14.8)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	LOF	n=auto[1-200], 15% unstable	456			8.53 (14.4)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	LOF	n=auto[1-200], 20% unstable	456			9.01 (18.1)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	Naïve Bayes	Gaussian	456	2.193	3.07	5.263
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	Naïve Bayes	Bernoulli	456	2.193	3.289	5.482
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight uniform	456	0.646 (1.258)	5.222 (6.918)	5.868 (8.176)

FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r	456	1.065 (2.516)	4.564 (6.289)	5.629 (8.805)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^2	456	1.275 (3.145)	4.354 (5.66)	5.629 (8.805)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^3	456	1.485 (3.774)	4.576 (5.66)	6.061 (9.434)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight 1/r^4	456	1.48 (3.774)	4.349 (5.66)	5.829 (9.434)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight e^-r	456	0.646 (1.258)	5.222 (6.918)	5.868 (8.176)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight e^-2r	456	0.646 (1.258)	5.222 (6.918)	5.868 (8.176)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight e^-5r	456	0.646 (1.258)	4.773 (6.918)	5.419 (8.176)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight e^-10r	456	0.8557 (1.887)	4.354 (5.66)	5.21 (7.547)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	kNN	k=auto[1-200], weight e^-100r	456	2.096 (6.289)	3.691 (5.031)	5.787 (11.32)
FFT	window=1.2, overlap=0, exp=2.75, bw=10, xyz	SVM	C-SVM, C=2.5, kernel=sigmoid, deg=4	456	1.946 (4.487)	3.701 (5.263)	5.647 (8.974)

doi:10.6342/NTU201903636