A Study of Clarinet Sound Quality and Register Characteristics

Jason-Jiasheng Xu
Computer Music Synthesis and Composition Program

March 3, 2023

Summary

This paper begins with a comprehensive review of literature in the field of clarinet sound analysis and synthesis, identifying valuable references from previous research.

In the study of noise components, I collected samples of E3, E4, and E5 notes through self-recordings and performed preprocessing tasks such as slicing and frequency domain transformation. Thirteen representative features were selected for spectral analysis, and feature extraction was performed on the spectral signals. Utilizing the SVM algorithm based on machine learning, the sound quality was first assessed by human hearing and then labeled based on the prominence of noise components, thereby constructing a sound quality classification model. Although the E5 note cannot be accurately classified, the classification accuracy for E3 and E4 test sets reached 90% and 75%, respectively.

In terms of clarinet sound construction, I compared and analyzed the recorded scales to depict the characteristics of each pitch range of the clarinet. Using filter functions, I filtered the clarinet sound model based on the "clarinet-all" function in Nyquist, creating a more realistic and expressive clarinet sound model. Furthermore, the generated E3 and E4 notes were incorporated into the classification model, both of which were classified as high-quality sounds in the SVM evaluation.

In conclusion, I analyzed the issue of the SVM algorithm's inability to accurately analyze the high-pitched E5 note and proposed a future concept that combines the SVM algorithm with the analysis of timbre in different pitch ranges of the clarinet.

Contents

1 Introduction

1.1 Background

The clarinet is a commonly used woodwind instrument that has a unique sound quality, making it a prominent instrument in solo performances and orchestral music. Therefore, sound quality plays an important role in the perception of clarinet sound. However, for beginners, it is often difficult to discern the quality of the clarinet sound. In the process of playing long notes, they often encounter the problem of unfocused sound. Moreover, issues such as mouthpiece control and the quality of the reed exacerbate the problem of noise components in the basic sound. Therefore, automatic evaluation of clarinet sound has substantial practical value.

Furthermore, the sound characteristics of different regions of the clarinet are distinct. Due to the presence of overtones, the vibration principles of the high and low registers of the clarinet differ significantly. In Nyquist sound synthesis, we cannot use the same sound model to directly adjust the pitch of the clarinet by changing the frequency. Therefore, it is necessary to adjust and modify the sound of the clarinet in different regions using filters for sound synthesis.

In recent years, research on clarinet sound based on machine learning technology has received increasing attention. In this paper, based on the support vector machine (SVM) method, I selected the three individual notes E3, E4, and E5 to study the sound quality of the classical clarinet. Additionally, based on the research results, the Author investigated the sound characteristics of different regions of the clarinet and synthesized a clarinet sound model starts with a built-in clarinet-all model provided with Nyquist.

1.2 Enhancements and Optimizations

In this second edition of the report, I have standardized the citations and expressions from the first edition, optimized the algorithms and models, and provided a more in-depth explanation for certain unclear areas. I also revised the summary and the work I did in the "My Work" section. I elaborated on the recording process, adding photographs of the recording environment and a method for quickly trimming audio based on loop amplitude judgments. Additionally, I re-plotted the frequency domain curve according to a linear axis and shared my understanding of harmonics and pitch.

In the SVM section, I indicated the sources of the images used in the first edition, which greatly helped me understand the algorithmic process and recalibrated the objective function based on my code. To address the poor recognition performance for E5, I used the overlapped spectrograms after the Fourier transform to more intuitively display the differences between the average values of the two types of sounds.

In Work 2, since I could not find the source of the original YAMAHA fingering chart, I cited the updated official version. Regarding the improved clarinet model design in Model 2, I explained that the model is based on the "clarinet-all" function and adjusted the parameters through linear fitting by combining the spectrograms and my auditory perception. I also attempted to more clearly showcase the improvements I made in parameter selection through graphical representation. Furthermore, I converted the originally generated,less applicable syllables into directly callable functions.

Finally, I incorporated the sound effects of this model into the original model for measurement to explore their actual sound performance.

2 Related Work

2.1 Literature Review

In analyzing the quality and sound of clarinet reeds, Wang Y, Guan X, and Du Y used a new method based on harmonic structure and energy distribution to evaluate the quality of clarinet reeds. [1] The researchers decoupled the mass of the reed from the clarinet and found that the mass of the reed is closely related to the even harmonic. Then they constructed a feature set that included even harmonic envelopes and harmonic energy distribution. The researchers recorded annotated clarinet audio data from three levels of performers and classified the sound quality through machine learning support vector machines.

In the study of the effective length of the clarinet tube and its sound, Dickens P, France R, and Smith J used a web-based database to analyze the acoustic details of clarinets containing standard fingerings and some alternative fingerings. [2] The clarinet has a cylindrical tube shape, one end of which is acoustically closed and the other end is open. As it is often used as an example of a closed-open pipe, we show several phenomena that can be clarified by comparing measurements on a cylindrical body with equivalent acoustic length to measurements on a clarinet.

Barthet M, Guillemain P, and Kronland-Martinet R studied the relationship between control gestures, sound color, and their perceptual representations in the clarinet. [3] This study provides great help for synthesis and control, music analysis and perception, and music information retrieval. They used a physics-based model to generate synthetic clarinet tones by changing the main control parameters of the model (related to blowing pressure and lip pressure on the reed) and obtained a low-dimensional spatial configuration that best represents the difference rating through multidimensional scaling and hierarchical clustering of the sound data collected in the experiment. Among them, by comparing the sound of natural and synthetic clarinets, the odd-even ratio was found to be a good indicator for predicting the unique vibrating reed situation of the clarinet.

Almeida A, Lemare J, and Sheahan M studied a simple model of the reed mouth of a single reed instrument and its control parameters. [4] These include intraoral pressure, force applied by the lips to the reed, application position, and damping of the reed. They used an automatic clarinet playing system to control intraoral pressure and control the force and position applied by the lips to the reed through two parameters. By studying the system of clarinet pitch and volume under different parameters, regions can be plotted where the intended notes are produced, the pitch is inaccurate, entering another sound area, slow starting, squeaking or no sound at all.

2.2 Literature Summary

The above literature has studied various aspects of clarinet acoustics and performance, providing important scientific support for our in-depth understanding of the sound characteristics and performance techniques of the clarinet. It also indicates that there have been many rich studies on the sound analysis of clarinets and the clarinet spectrum, especially in the field of the impact of reeds and mouthpiece pressure on clarinet sound.

2.3 Research Gap

Although there have been many rich studies on the sound of the clarinet, specialized research on noise generated by factors such as reeds, mouth shape, and clarinet structure is still lacking. Many times, what we consider as "unfocused" clarinet sound is caused by excessive noise, so the first half of the study will mainly focus on sound quality research. However, in order to obtain a more complete understanding of the clarinet sound, it still needs to conduct further research on noise factors.

3 My work

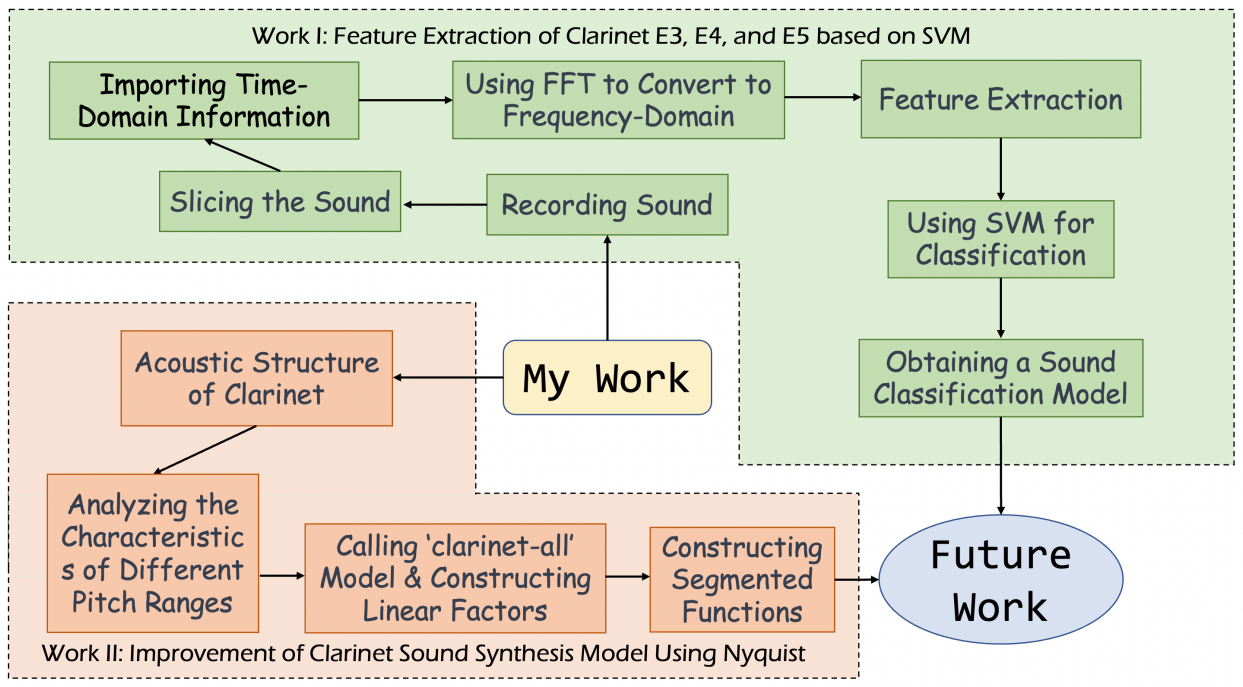
Work I: Feature Extraction of Clarinet E3, E4, and E5 based on SVM
Importing Time-Domain Information
Using FFT to Convert to Frequency-Domain
Feature Extraction
Slicing the Sound
Recording Sound
Using SVM for Classification
Obtaining a Sound Classification Model
My Work
Acoustic Structure of Clarinet
Analyzing the Characteristics of Different Pitch Ranges
Calling 'clarinet-all' Model & Constructing Linear Factors
Constructing Segmented Functions
Future Work
Work II: Improvement of Clarinet Sound Synthesis Model Using Nyquist

Figure 1: Flow Chart of My Work

I do such things:

1. Conducting research to identify potential areas of study.
2. Recording clarinet sounds, both obvious and subtle noises, and utilizing python to take samples automaticly.

3. Analyzing sound frequency using Matlab and categorizing various sound types using machine learning.
4. Investigating the characteristics of clarinet sounds in different tonal ranges and improving the clarinet's sound model starts with the clarinet-all function in Nyquist.
5. Adding sound generated by improved Clarinet Model to the test set and classifying it.

4 Work I

4.1 Construction of the Dataset

Firstly, in the quiet music rehearsal room at midnight, I used the recording device LYRA, placed at a fixed distance of 2 meters, to record multiple long sound samples of the clarinet playing E3, E4, E5, as well as a slow-paced sound scale from E3 to E6 (used for Work II).

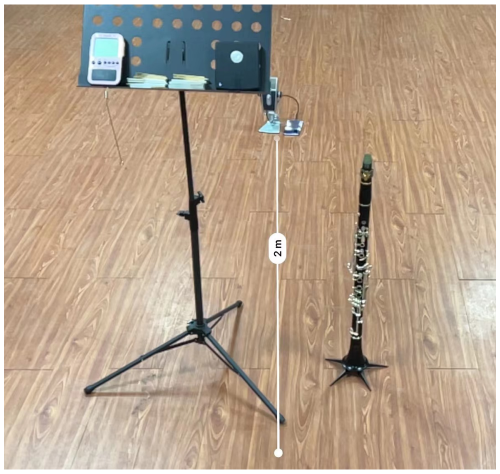

Figure 2: The Recording Process

The next step is to begin the cutting of the samples. The author created 60 samples, each lasting one second, for each of the three types of sustained tones. The sound was converted to mono. The samples used in this experiment were manually cut, and the sound samples were selected from the stable portion of the temporal signal within a sustained tone. The sustained tone of the clarinet is composed of rapidly changing sound heads and a slowly ending sound tail, so the middle section of the sample is the stable portion and is more comparable horizontally.

As the second version of the study, the author developed a plan for cutting through looping automaticly based on the improvement suggestions provided by Professor Dannenberg. During the recording of this sustained tone, multiple sustained tones in the same pitch were recorded in a long audio file, with each sustained tone having almost the same length, but the intervals between them were not fixed. Therefore, in this loop, second-based cutting was considered to achieve automatic recognition and cutting of the stable part of the sustained tone.

Initially, I attempted to implement this algorithm using Nyquist. However, when attempting to use the "snd-maxsamp" function to calculate the maximum frequency after reading the audio file, an error occurred. I thought that this might be related to the format of the file when it was read, but was unsure how to debug it. Therefore, using the same programming ideas, I chose to implement the program using the Python language. The specific reading method is shown below.

- First, read and store the audio.
- Then, traverse the entire audio and calculate the average value of the envelope in each second.
- If the average value of the t-second minus the average value of the (t-1)-second > 0.3 * max_amplitude, store the time point in a list.
- Construct a loop. When the time node t_i coincides with the node in the list, store the audio of this second starting from the time point t_i+1 and ending at the time point t_i+2.

4.2 Frequency Domain Transformation

To more intuitively see the differences between the samples, we need to perform frequency domain conversions on each sample. Due to the large number of samples and the need to use machine learning tools in later processes, Matlab was used as the tool to transform the audio slices into frequency domain signals in matrix form (see the appendix for the specific conversion code $audioFFT.m$).

Discrete Fourier Transform (DFT) is the most classic frequency domain conversion tool:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi \frac{nk}{N}} \tag{1}$$

It converts a finite length time domain sequence into a finite length sequence in the frequency domain that is also discrete. Fast Fourier Transform (FFT) is based on the above formula, using symmetry and repetition properties to reduce the computational complexity from$O(n^2)$ to $O(nlogn)$, greatly improving the calculation speed [14].

The Figure 3 is an example of the FFT transformation result for $E31.wav$(the first sample in E3). This image is an excellent example for understanding the principle of instrument harmonics. Pitch is our subjective perception of the high or low of sound, mainly determined by the fundamental frequency of the sound. Harmonics are a series of frequencies composed of integer multiples of this fundamental frequency[8]. The unique timbre of the instrument we hear is composed of this fundamental frequency and different harmonics. Since all of the harmonics are equally spaced, the frequency domain characteristics of the clarinet can be clearly displayed by adjusting the x-axis range using a standard coordinate system.

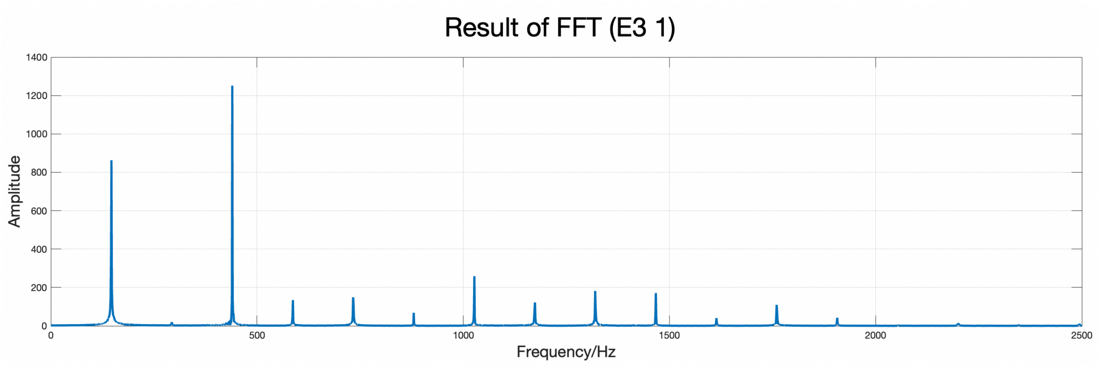

Figure 3: Frequency Domain Signals of E3

In addition, according to the relationship between pitch and fundamental frequency, we know that the fundamental frequency of E4 is twice that of E3[9]. They also have some similarities in their frequency domain plots. The following is the frequency domain plot of E4, which shows the similarity in the frequency domain between E3 and E4. The similarity in their frequency domains contributes to the unique timbre of the clarinet.

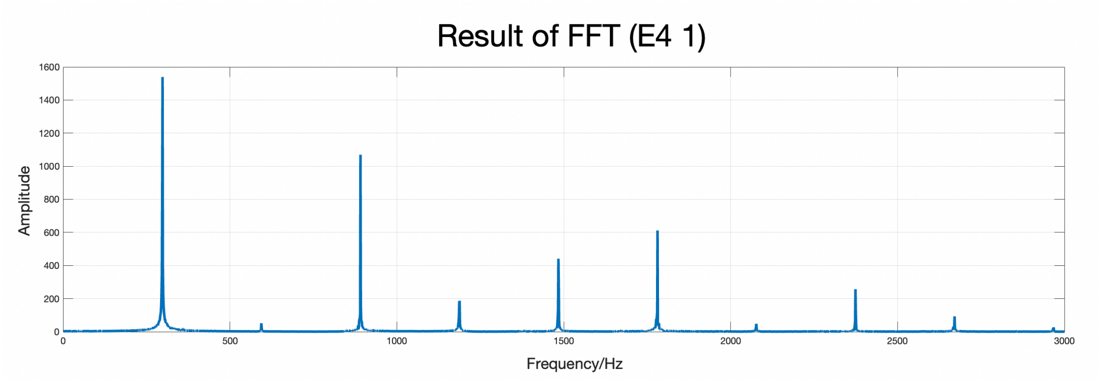

Figure 4: Frequency Domain Signals of E4

4.3 Feature Extraction

After performing the Fast Fourier Transform, the frequency domain data for all samples of E3, E4, and E5 were obtained. According to the sampling theorem, the data was transformed into three matrices of 60 x 22050. To further explore the data, this study used feature extraction to select 13 representative audio features and transformed these three frequency domain matrices into feature matrices of audio samples, as shown in Figure 5.

After the aforementioned frequency domain transformation, each audio data is transformed into a feature vector of the above 13 data, which represents the characteristics of the data. Thus, we can obtain a matrix of 60x13. This is the basis for performing SVM.

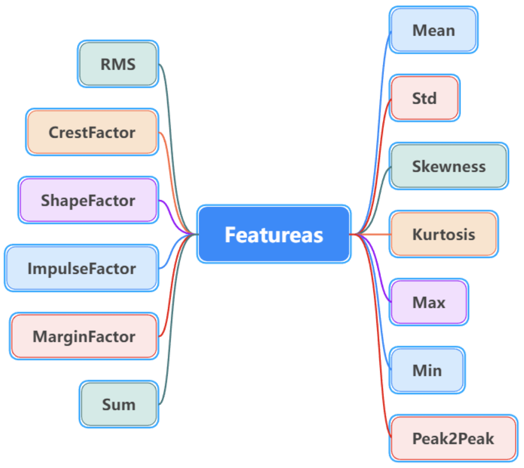

Figure 5: Features Used in SVM

4.4 Processing the Data Using SVM

Support Vector Machine (SVM) is a type of generalized linear classifier that performs binary classification on data in a supervised learning manner, and the decision boundary is the maximum margin hyperplane obtained by solving the learning sample [10]. Figure 6 shows the case of linearly separable data (can be separated by a straight line) in two dimensions:

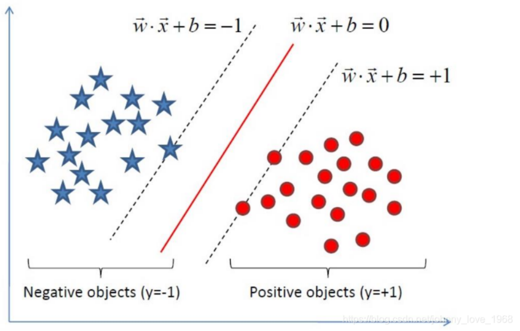
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}\cdot\vec{x}+b=+1$
Negative objects (y=-1)
Positive objects (y=+1)

Figure 6: Two Dimensions Separable Data [7]

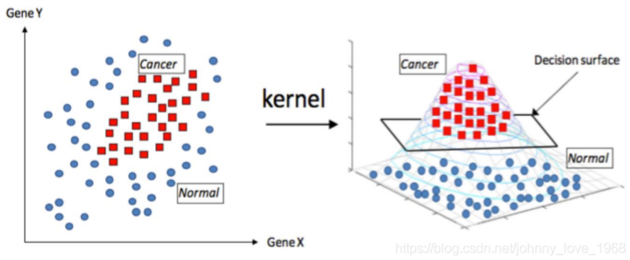

Figure 7: What Kernel Functions Do [7]

In its two-dimensional case, there are two feature vectors. When we have 13 feature vectors, we only need to solve the maximum margin line in 13 dimensions in the same way as in two dimensions.

However, not all datasets are separable, so we need to introduce two concepts: the penalty factor and the kernel function in. The kernel function allows SVM to classify in a high-dimensional feature space without explicitly calculating the feature mapping. The radial basis function (RBF) used in this problem's code is a class of kernel functions [12].

In the specific objective function, $\frac{1}{2}||w||^2$ aims to maximize the classification margin, and $C\sum_i \xi_i$ aims to penalize misclassified samples. In this process, we need to find a suitable set of Lagrange multipliers α values to minimize the objective function. The specific mathematical expression of the objective function is as follows [13]:

$$\begin{aligned} \min_{w,\xi,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y_i(w^T\varphi(x_i)+b) \geq 1-\xi_i \\ & \xi_i \geq 0, \quad i = 1,2,\ldots,N \end{aligned} . \tag{2}$$

When writing the SVM code, the author called the 'svmtrain' function in LIBSVM and set the penalty factor C to 10 and the parameter g of the RBF kernel($\phi(x_i)$) to 0.01. The 'svmtrain' function in LIBSVM completes the optimization and calculation of the objective function internally [11].

In practical applications, I selected 40 samples as the training set, 20 samples as the test set, normalized all the data, constructed a penalty factor and a radial basis function parameter, and conducted simulation tests.

The following are the algorithm steps in the SVM classification process:

(1) Clearing environment variables, turning off warning messages, closing open figure windows, clearing variables, and clearing the command window.

(2) Reading matrix data from an Excel file and storing it in the "res" variable.

(3) Randomly dividing the data into training and test sets, where 40 samples are in the training set and 20 are in the test set.

(4) Normalizing the data using mapminmax function.

(5) Transposing the training and test data to fit the SVM model.

(6) Creating an SVM model using the svmtrain function with a radial basis function kernel and specified penalty factor and gamma values.

(7) Testing the model's performance on the training and test data using the svmpredict function and calculating the classification error.

(8) Sorting the predicted results and the actual results for both the training and test sets.

(9) Plotting the comparison between the actual and predicted values for the training and test sets.

(10) Creating confusion matrices for the training and test sets using the confusion chart function.

The advantage of the above algorithm is that it can judge the training effect of the model by comparing the differences between the test set and the training set, and the confidence matrix can be used to determine its reliability.

4.5 Results and Discussion

As SVM is a classification model based on existing classification, I divided the data in E3, E4, and E5 into two categories based on whether the noise in the sound was obvious, and input them into SVM.m for testing, obtaining the following results:

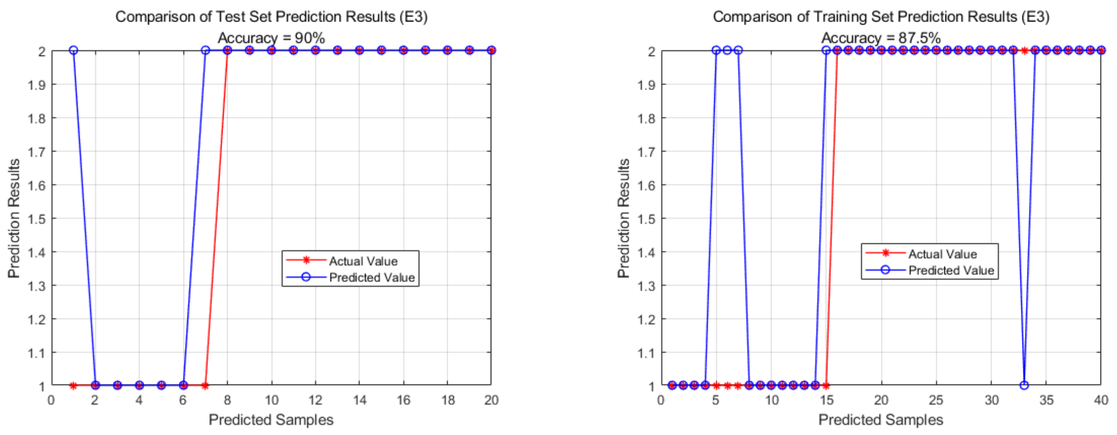

Figure 8: Test and Train Result of E3

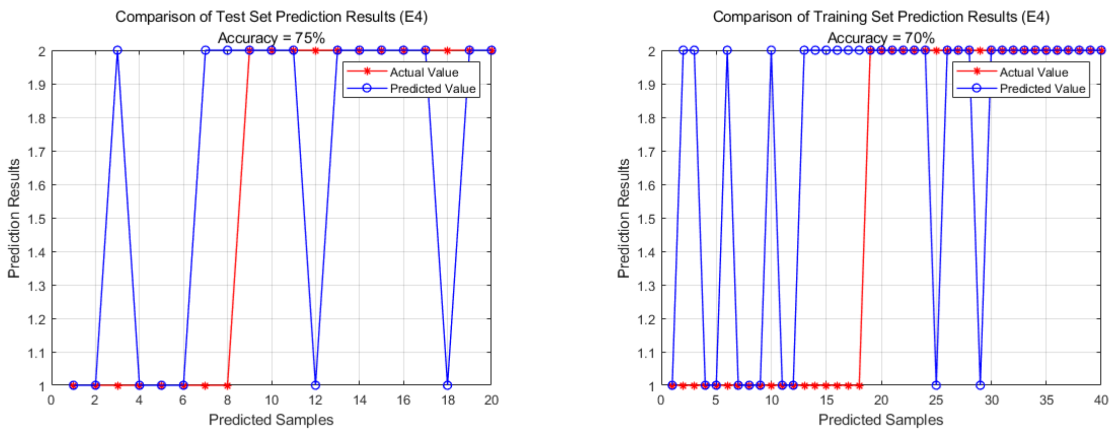

Figure 9: Test and Train Result of E4

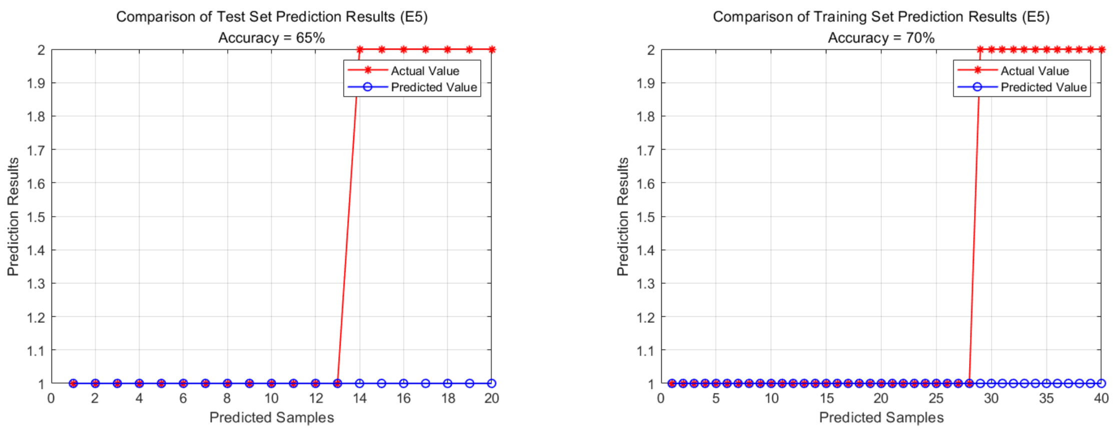

Figure 10: Test and Train Result of E5

During the experimentation process, I found that some parts of E3 and E4 can be distinguished very well into these two categories, with accuracy rates of around 90% for both. However, E5 cannot be distinguished using these feature vectors.

To determine the reason why the sound cannot be distinguished, all E5 data in this study were extracted and their mean data for good E5 sounds and bad E5 sounds were obtained according to their original classification. They were then imported into MATLAB for frequency domain analysis, and their frequency spectrum is shown in Figure 11. Through the comparison and analysis of the frequency spectrum, it can be found that the noise component is not significant and does not affect the features such as skewness and peak value that have been extracted.

Specifically, for the frequency spectrum itself, good performance in playing produces fuller and louder sound at lower frequencies, and the adjacent sound volume is higher around specific frequency of the harmonic. In contrast, the sound with obvious noise has a lower volume at lower frequencies, and the sound is louder than that of the good performance in playing at the harmonic frequencies in the high-frequency range. However, these features cannot be reflected in the feature extraction mentioned above, which is also the reason why this model cannot classify the high-frequency range well.

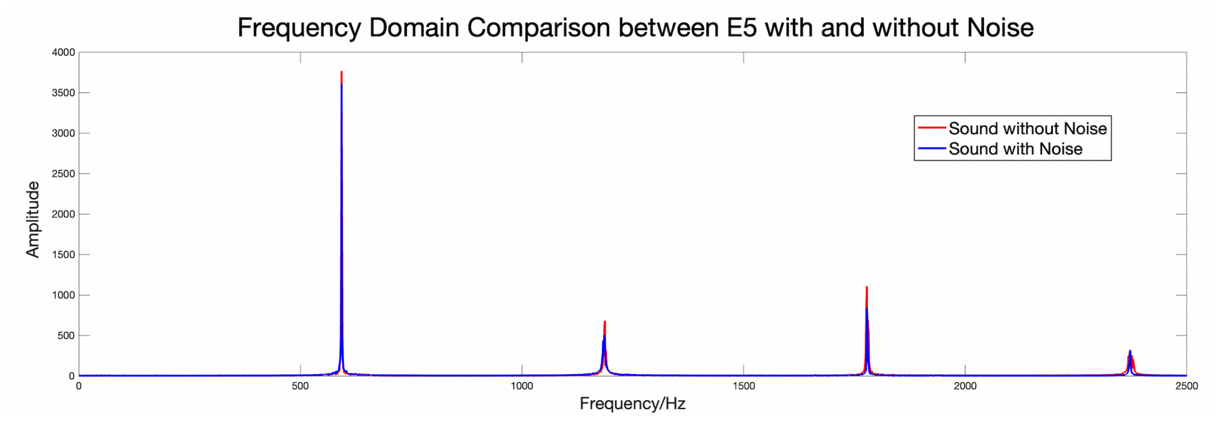

Figure 11: Result of FFT(E3 1)

5 Work II

5.1 Timbre Analysis of Different Vocal Ranges

The sound range of the clarinet is mainly divided by the use of harmonic keys. The clarinet produces a middle to high range sound when using harmonic keys and a middle to low range sound when not using them. Additionally, the clarinet can achieve overblowing through special fingerings, allowing it to play notes ranging from C6 to C7.

In the "throat" area of the clarinet, specifically G4, GS4, A4, and AS4, there are relatively few closed holes in the tube, resulting in significant white noise. Starting from B4, the overall timbre of the clarinet becomes more focused and bright due to the use of a harmonic key, and the sound production principles inside the tube change to a certain extent.

Figure 12 is a chart of commonly used fingerings for the clarinet, it was published by YAMAHA, and we could find the mentioned features in this fingering chart:

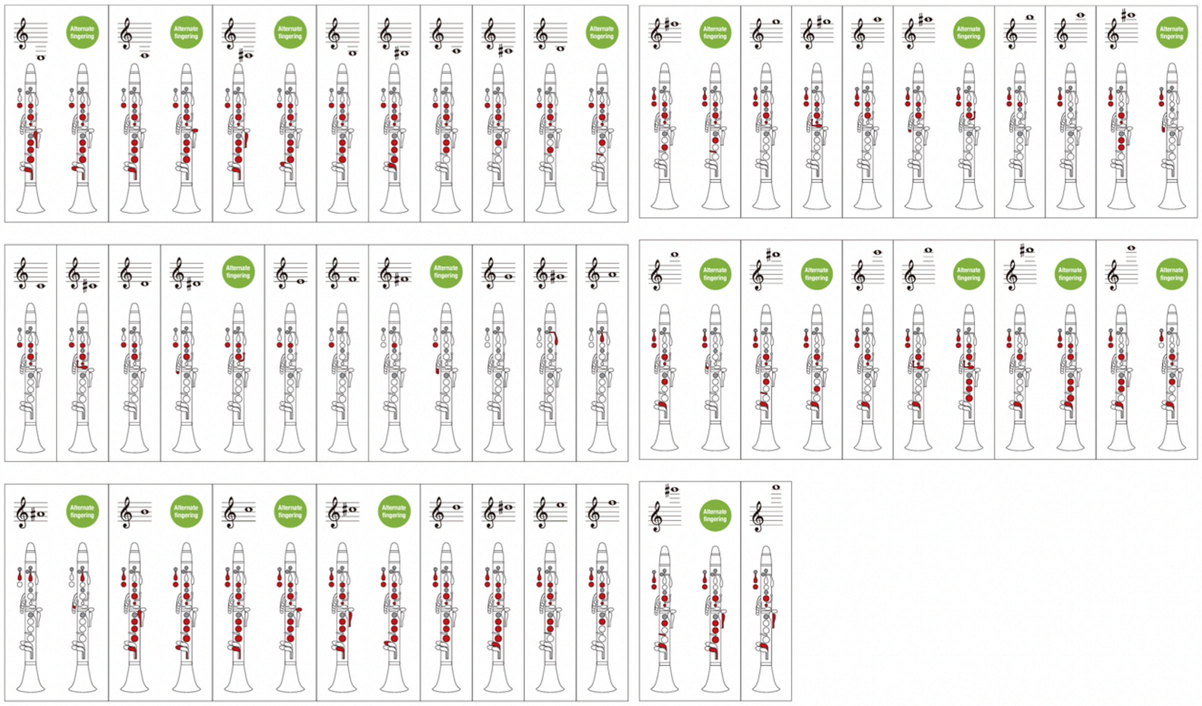

Figure 12: Yamaha Clarinet Fingering Chart[6]

5.2 Model Optimization based on 'clarinet-all' Function

In Nyquist, there is a built-in function called 'clarinet-all', which is constructed based on a simple single reed model and provides many parameters for adjustment, which can effectively showcase the timbre of a single reed instrument. However, since this is a relatively basic model, its principle is based on directly changing the effective tube length of the instrument to alter its pitch [15], it cannot showcase the characteristic timbre differences between different registers of the single reed instrument.

Therefore, after constructing the basic instrument parameters, I improved the above function based on the sound characteristics of modern classical single reed instruments by adjusting the reed hardness and instrument sound noise. Regarding the selection of filters, Nyquist's built-in functions provide low-pass and high-pass filters ranging from first to eighth order. The higher the order of the filter, the more poles it has, the narrower the transition band, and the smoother the frequency response in the passband [16]. Since there is only one instrument here and the spectrum is relatively simple, I chose low-order filters for the mid-to-low range, but for the high-range, which is particularly sensitive to filter selection, I chose high-order filters.

In particular, after Professor Dannenberg proposed modification suggestions, I re-adjusted the fitting parameters for each register and changed the original segmented scale to a function that can be directly called by pitch, and added linear fitting functions to further transition between different registers, making the sound effects between different ranges more natural. For specific parameter selection, I set the threshold for high-frequency sound based on experimental results and my own subjective auditory perception for the low-pass filter, while the high-pass filter mainly protects the instrument's fundamental frequency sound.

Overall, this improved model satisfies the characteristic that the more open the instrument's tone holes, the narrower the sound bandwidth: from the low-register to the BS4 register, the bandwidth gradually narrows, and then there is a jump in the B4 register. BS4 is the note with the most open tone holes in the single reed instrument, and there will be a strong sensation of dryness when playing this note. To solve this problem, performers also use many alternative fingerings to play this note, that is, closing the tone holes in the low-register without changing the instrument's effective tube length to expand the note's bandwidth.

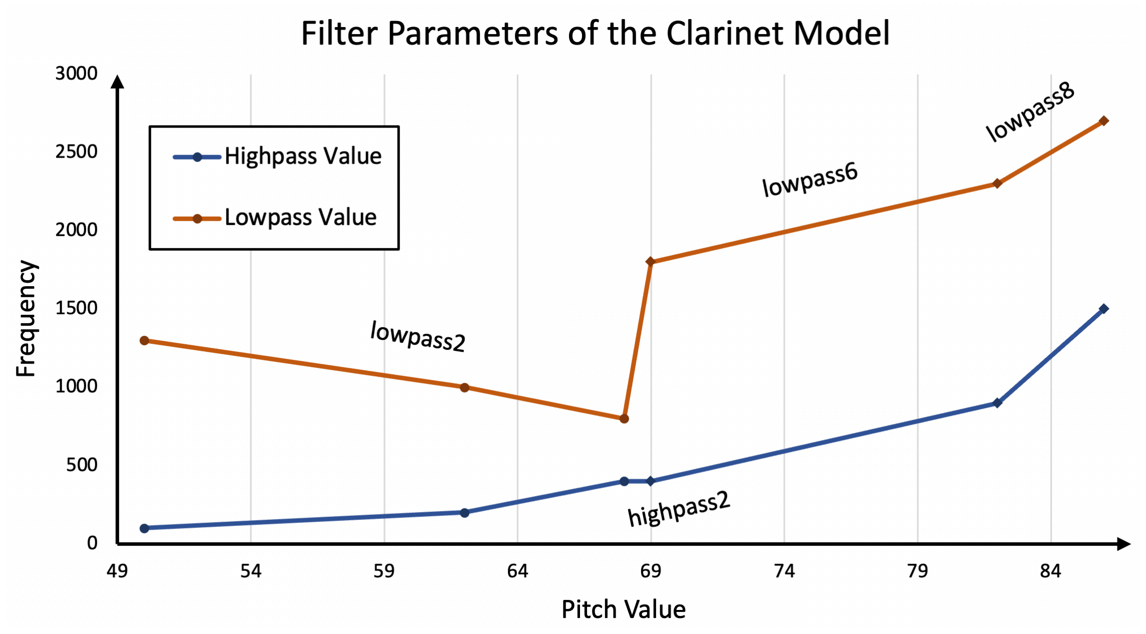

Figure 13: Result of FFT(E3 1)

5.3 Sound Effect Evaluation

After generating the sound, I also selected the stable part for classification. In the SVM process, I added the sounds generated by Nyquist to the test set and used the original training set for training.

Since the classification model for E5 itself cannot achieve good classification, and I have not yet found a better feature for fitting, I did not evaluate the E5 sound generated by Nyquist. Therefore, I used the method mentioned above to bring in the sounds of E3 and E4, perform feature extraction and other operations from the slice, and then input them into the model for classification.

I was very happy to find out that they were all classified as good sounds in this classical clarinet-based classification model. Of course, this is also largely due to the fact that the sounds generated by Nyquist are very pure and have a strong classical clarinet flavor.

Of course, the classification features of my SVM are all composed of frequency domain signals. If the time domain signal features are also added to them, I think it can definitely distinguish which sounds are synthesized by Nyquist, because its time domain spectrum is very smooth.

6 Evaluation

6.1 Strengths

- The SVM method is effective in handling multi-feature problems, and in practical use, I found that this method particularly excels in analyzing the low-pitched sound range.
- The innovation of the model lies in the analysis of the characteristics of different sound ranges of the clarinet, which gives each sound range of the synthesized clarinet sound different parameters, thereby more accurately portraying the sound characteristics of the clarinet.

6.2 Weaknesses

- The model's fitting performance for the high-pitched sound range, especially the range that uses harmonic keys, is not satisfactory, and it is necessary to try selecting other more representative features.
- In addition to the differences between sound ranges, there are also certain differences between each note within the same sound range, but this model is not sufficiently detailed from this perspective. Many people who practice playing the clarinet, even if they do not have absolute pitch, can often judge which note they are playing based on changes in the timbre of each pitch.
- However, the timbre of a clarinet cannot be described by a single criterion of good or bad. The criteria used to judge the quality of clarinet sound in this study are based on classical clarinet and cannot be used as a standard for judging jazz clarinet. Additionally, since the training dataset was based on sustained tones, it may not be able to accurately identify techniques such as tonguing or glissando during testing.

7 Future Work

In the future, I believe that in addition to optimizing the model itself, combining the work I and work II that I have done will be a very meaningful task. This involves studying the characteristics of each note in detail to construct a top-level clarinet sound synthesizer.

Furthermore, I am inspired by Professor Dannenberg's research on trumpet sound synthesis[5], which considers factors such as melodic contour, articulation, and dynamics to synthesize realistic trumpet performances using Spectral Interpolation Synthesis. I also want to conduct similar in-depth exploration and research on the clarinet. This can not only be used in synthesizer performance and software orchestration, but also in fields such as music information retrieval. Furthermore, through digitization, the timbre of the clarinet, as a musical instrument, can be preserved as part of human memory.

8 Acknowledgment

First and foremost, I would like to express my heartfelt gratitude to Professor Dannenberg, Mr. Merrill, and TA Tu for their invaluable guidance and assistance throughout the winter break research project. Their support enabled me to smoothly learn the sal language in Nyquist and gain a preliminary understanding of the grand framework of computer music. In this final project report, I have ventured for the first time to combine my passion for music with my studies in mathematics and computer science. Under the guidance of Professor Dannenberg, I successfully completed the entire process from project planning, experimental design, data processing, to result validation.

Although this report only touches upon the tip of the iceberg in the field of computer music, it has indeed opened the door for me to explore this area further, fueling my intense desire to delve into music information retrieval, automatic accompaniment, and artificial intelligence music generation.

I would like to extend my sincerest thanks to Professor Dannenberg in particular. He provided detailed answers to all my questions, offering me an abundance of ideas and clearing up many of my confusions. He also gave me comprehensive feedback on the first draft of my report, pointing out key mistakes that were crucial for my growth. The version you see now is the second draft of the report. In the first draft, I committed plagiarism, improper citation, and imprecise expression. I deeply recognize the severity of these mistakes and offer my most sincere apologies. In this second draft, I have corrected these errors and pledge never to make similar mistakes in the future.

While making such mistakes is regrettable, the timely identification and rectification of these issues at the beginning of my academic career may serve as a beneficial wake-up call for my long journey ahead in research. I will cherish Professor Dannenberg's teachings, keep the lessons learned from the first draft firmly in my heart, and stay grounded as I continue to make progress.

References

[1] Wang Y , Guan X , Du Y , et al. Harmonics Based Representation in Clarinet Tone Quality Evaluation[J]. IEEE, 2020.

[2] Dickens P, France R, Smith J, et al. Clarinet acoustics: introducing a compendium of impedance and sound spectra[J]. Acoustics Australia, 2007, 35(1): 17.

[3] Barthet M, Guillemain P, Kronland-Martinet R, et al. From clarinet control to timbre perception[J]. Acta Acustica united with Acustica, 2010, 96(4): 678-689.

[4] Almeida A, Lemare J, Sheahan M, et al. Clarinet parameter cartography: automatic mapping of the sound produced as a function of blowing pressure and reed force[C]//Proceedings of International Symposium on Music Acoustics. 2010.

[5] Dannenberg R B , Pellerin H , Derenyi I . A Study of Trumpet Envelopes[J]. International Computer Music Association, 2009.

[6] YAMAHA. (n.d.). Fingering diagram for the clarinet.

https://www.yamaha.com/en/musical_instrument_guide/clarinet/play/play002.html

[7] . (2021, May 9). Introduction to SVM (SVM.) CSDN Blog.

https://blog.csdn.net/johnny_love_1968/article/details/116566101

[8] Chen J M, Smith J, Wolfe J. Saxophone acoustics: introducing a compendium of impedance and sound spectra[J]. Acoustics Australia, 2009, 37(1-19): 18-23.

[9] Stevens S S, Volkmann J. The relation of pitch to frequency: A revised scale[J]. The American Journal of Psychology, 1940, 53(3): 329-353.

[10] Chauhan V K, Dahiya K, Sharma A. Problem formulations and solvers in linear SVM: a review[J]. Artificial Intelligence Review, 2019, 52(2): 803-855.

[11] Hsu C W, Chang C C, Lin C J. A practical guide to support vector classification[J]. 2003.

[12] Jakkula V. Tutorial on support vector machine (svm)[J]. School of EECS, Washington State University, 2006, 37(2.5): 3.

[13] Li Hang( ). Statistical Learning Methods( )[M]. Qing hua da xue chu ban she, 2012.

[14] Brigham E O, Morrow R E. The fast Fourier transform[J]. IEEE spectrum, 1967, 4(12): 63-70.

[15] Dannenberg R. Nyquist reference manual[J]. 2008.

[16] Schaumann R, Mac Elwyn Van Valkenburg X, Xiao H. Design of analog filters[M]. New York: Oxford University Press, 2001.

Appendix A: Program Codes

Here are the program codes we used in our research.

`sliceMaker.py`

```
# This Python script reads an audio file, calculates the average
# amplitude of the envelope per second, stores the time points with
# a significant difference in amplitude, and extracts and saves
# the audio segments.
#
# Author: Jason Xu
# School: Beijing Normal University, China
# Email: 1773117640@qq.com
# Date: 2023/4/21
#
# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

import numpy as np
from pydub import AudioSegment
import os
##
# 1. Read and store the audio file
# You can change the file read path here
audio_file = "/Users/xujiasheng/Documents/Nyquist/projects\
/ICMproject/ICM/projectMaterials/originalAudio/E4.wav"
# Reading audio data
audio = AudioSegment.from_wav(audio_file)

##
# 2. Calculate the average amplitude of the envelope per second
# Get the audio sample rate
samples_per_second = audio.frame_rate
# Convert audio duration from milliseconds to seconds
seconds = len(audio) // 1000
# Store the average amplitude of each sample point
average_amplitude = []
# Iterate over seconds to compute the average amplitude for
# each time segment in the audio file
for i in range(seconds):
    start = i * samples_per_second
    end = (i + 1) * samples_per_second
    # Get the audio data for the time segment from start to end
    segment = audio.get_sample_slice(start, end)
    # Convert the audio sample data to an array containing
    # the amplitude values for all the samples
    # in the current time segment
    segment_numpy = np.array(segment.get_array_of_samples())
    # Generate a list that contains the average amplitude
    # for each time segment.
    average_amplitude.append(np.mean(np.abs(segment_numpy)))

##
# 3. Calculate and store time points
# Create an empty list to store time points
time_points = []
# Get the maximum average amplitude across all time segments
max_amplitude = max(average_amplitude)
# Find time points where the difference between the average amplitude
# of the current time segment and the previous one is greater
# than 30% of the maximum amplitude
for t in range(1, len(average_amplitude)):
    if average_amplitude[t] - average_amplitude[t - 1]\
      > 0.3* max_amplitude:
        time_points.append(t)

##
# 4. Extract and save audio segments
output_directory = "/Users/xujiasheng/Desktop/audio_segments/"
# Check if the output directory path exists.
if not os.path.exists(output_directory):
    os.makedirs(output_directory)
# Save the extracted audio segments that meet the condition
# to new WAV files, using the time points as indices
for i, t_i in enumerate(time_points):
    start_ms = (t_i + 1) * 1000
    end_ms = (t_i + 2) * 1000
    extracted_segment = audio[start_ms:end_ms]
    # Convert stereo audio to mono
    extracted_segment = extracted_segment.set_channels(1)
    extracted_segment.export(f"{output_directory}\
    segment_{i}.wav", format="wav")
```

improvedClarinetModel.sal

```
; This SAL language code improve a clarinet sound model and is based
; on a built-in clarinet-all function in Nyquist.
; It adjusts the reed stiffness, vibration frequency, noise level,
; and uses low-pass and high-pass filters to make the clarinet's
; timbre better match the characteristics of different pitch ranges.
; Author: Jason Xu
; School: Beijing Normal University, China
; Email: 1773117640@qq.com
; Date: 2023/4/21
; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

; Build a basic clarinet sound model, specifying the vibration
; and reed stiffness
define function clar-test (pitch: 80, slide-amt: 0, noise-level: 0,
reed-stiff: 0.64)
```

```
  begin
    with breath = pwlv(1, 0.2, 0.95, 0.9, 0.3) ~ 2, freq-env =
pwlv(0), vib-freq = 1000,  vib-gain = 0.02
      return(clarinet-all(pitch, breath, freq-env, vib-freq, vib-gain,
reed-stiff, noise-level))
  end

; Define a linear transformation function that allows sound to be
; filtered and transformed in different regions by calling this function
define function lin-interp (x, x1, x2, y1, y2)
  begin
    return (y1 + (x - x1) * (y2 - y1) / (x2 - x1))
  end

; Define the clarinet sound playback function.
; According to the range of the Bb clarinet, the lowest note is E3 = 50.
; Although the low frequency range of the clarinet has a low
; fundamental frequency, it often has higher harmonics, making
; the low frequency signal fuller and more penetrating.
; The basic lp and hp functions are not used because they are
; found to lose the original timbre of the clarinet during use.
function clar-play(pitch_value: 60)
  begin
    if pitch_value < 62 & pitch_value >= 50 then
      begin
        with lowpass_val = lin-interp(pitch_value, 50, 62, 1300, 1000),
highpass_val = lin-interp(pitch_value, 50, 62, 100, 200)
          play lowpass2(highpass2(clar-test(pitch: pitch_value,
slide-amt: 0, noise-level: 0, reed-stiff: 0.64) , highpass_val),
lowpass_val) ~ 1
      end

; The mid-range of the clarinet is somewhat dry compared to the low
; range, with the sound mainly concentrated in the fundamental frequency.
; Since almost all holes are open, there are fewer harmonics,
; which can be suppressed by a low-pass filter
    else if pitch_value < 69 then
      begin
        with lowpass_val = lin-interp(pitch_value, 62, 68, 1000, 800),
highpass_val = lin-interp(pitch_value, 62, 68, 200, 400)
          play lowpass2(highpass2(clar-test(pitch: pitch_value,
slide-amt: 0, noise-level: 0.4, reed-stiff: 0.68) , highpass_val),
lowpass_val) ~ 1
        end

; In the high range, the main harmonics of the clarinet come from
; the fundamental frequency during playing.The note 69 corresponds to
; the B4 of the clarinet, which is played using the harmonic key and
; closing all holes (except for the harmonic key). The principle of
; vibration changes at this point, so the sound change is not
; continuous. The lowpass6 function works better here.
    else if pitch_value < 82 then
      begin
        with lowpass_val = lin-interp(pitch_value, 69, 82, 1800, 2300),
highpass_val = lin-interp(pitch_value, 69, 82, 400, 900)
          play lowpass6(highpass2(clar-test(pitch: pitch_value,
slide-amt: 0, noise-level: 0.5, reed-stiff: 0.72) , highpass_val),
lowpass_val) ~ 1
          end

; This range belongs to the super high range of the clarinet. The
; performer often produces more noise with a sharper sound.
; Therefore, the noise level is increased, and the low frequency
; performance is suppressed by the high-pass filter
    else if pitch_value < 87 then
      begin
        with lowpass_val = lin-interp(pitch_value, 82, 87, 2300, 2700),
highpass_val = lin-interp(pitch_value, 82, 87, 900, 1500)
          play lowpass8(highpass2(clar-test(pitch: pitch_value,
slide-amt: 0, noise-level: 0.6, reed-stiff: 0.77) , highpass_val),
lowpass_val) ~ 1
        end
  end

  ; Sound example, modify the pitch_value at the end
  exec clar-play(pitch_value: 52) ~ 1.5
```

audioFFT.m

```
%In this code, the file path for reading can be changed by modifying
%the value of s1, and the
%number of files to be read can be changed by modifying the range of i.
%Author:Jason Xu
%School:Beijing Normal University, China
%Email:1773117640@qq.com
%Date:2023/3/1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Due to the small number of overall files and the fact that the size of
%the matrix
%will change with the number of samples, there is no need to preallocate
%matrix memory.
%l = zeros(50000,200);   %Optional
for i = 1
    s1 = '.\processedSamples\E3\';  %Modify the file path here.
    s2 = string(i);
    ss = strcat(s1, s2);
    s3 = '.wav';
    s = strcat(ss, s3);   %Get the file address.
    %disp(s);
    %The sample data of the audio is read and stored in the matrix x,
    %and the sampling rate Fs of the audio is obtained.
    [x_o,Fs] = audioread(s);
    x = x_o(:,1);
    x = x';
    N = length(x);    %The total number of sample data is calculated.
    y = fft(x);   %Fast Fourier Transform is performed.
    f = Fs/N*(0:round(N/2)-1);
%The horizontal frequency values after Fast Fourier Transform are constructed
    m = abs(y(1:round(N/2)));
%The absolute value of the FFT result is taken.
    m = m';
    l(:,i) = m(:,1);
%Save the value of m into the corresponding column of matrix l.
end
```

featureDomain.m

```
%This is a code for feature extraction on sound spectrogram matrix. A total
%of 13 features are extracted as follows:
%Mean, Std, Skewness, Kurtosis, max, min, Peak2Peak, RMS, CrestFactor,
%ShapeFactor, ImpulseFactor, MarginFactor, Energy
%Author:Jason Xu
%School:Beijing Normal University, China
%Email:1773117640@qq.com
%Date:2023/3/1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Modify the file path here.
load('.\spectrumMatrixFFT\l5.mat')
%Create a table
features = table;
for i=1:60
    v = l(:,i);
    %features of specturms
    features.Mean(i) = mean(v);                            %Mean
    features.Std(i) = std(v);                              %Std
    features.Skewness(i) = skewness(v);                    %Skewness
    features.Kurtosis(i) = kurtosis(v);                    %Kurtosis
    features.max(i) = max(v);                              %max
    features.min(i) = min(v);                              %min
    features.Peak2Peak(i) = peak2peak(v);                  %Peak2Peak
    features.RMS(i) = rms(v);                              %RMS
    features.CrestFactor(i) = max(v)/rms(v);               %CrestFactor
    features.ShapeFactor(i) = rms(v)/mean(abs(v));         %ShapeFactor
    features.ImpulseFactor(i) = max(v)/mean(abs(v));       %ImpulseFactor
    features.MarginFactor(i) = max(v)/mean(abs(v))^2;      %MarginFactor
    features.Energy(i) = sum(v.^2);                        %Energy
end
```

SVM.m

```
%This is a code for SVM classification of a 60*13 matrix with 60 samples
%and 13 audio signal features. It is divided into five steps: data import,
%data processing, model training, model testing, and image drawing.
%Firstly, the features are extracted by taking each column of the
%spectrogram matrix as a vector,
%and then combining them into a feature matrix.
%The model code was referenced from
%https://www.bilibili.com/video/BV1xa411K7aF/?spm_id_from
%=333.337.search-card.all.click&vd_source=7a30b342baf263332cb2825bdf0a44a8
%and was modified by selecting plugins, debugging data, and modifying images.
%Author:Jason Xu
%School:Beijing Normal University, China
%Email:1773117640@qq.com
%Date:2023/3/1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Clear Environment Variables
warning off % Turn off warning messages
close all % Close open figure windows
clear % Clear variables
clc % Clear command window
%Modify the file path here.
res = xlsread('.\featureMatrix\features3.xlsx');
%Divide into Training and Test Sets
temp = randperm(60);
P_train = res(temp(21: 60), 1: 13)';
T_train = res(temp(21: 60), 14)';
M = size(P_train, 2);
P_test = res(temp(1: 20), 1: 13)';
T_test = res(temp(1: 20), 14)';
N = size(P_test, 2);
%Normalize Data
[p_train, ps_input] = mapminmax(P_train, 0, 1);
p_test = mapminmax('apply', P_test, ps_input );
t_train = T_train;
t_test = T_test ;
%Transpose to Fit Model
p_train = p_train'; p_test = p_test';
t_train = t_train'; t_test = t_test';
%Model Creation
c = 10.0; % Penalty factor
g = 0.01; % Radial basis function parameter
cmd = ['-t 2', '-c', num2str(c), '-g', num2str(g)];
%The svmtrain function here requires configuring the C hybrid compilation:
%https://www.csie.ntu.edu.tw/~cjlin/libsvm/, download the libsvm plugin.
model = svmtrain(t_train, p_train, cmd);
```

```
%Simulation Testing
T_sim1 = svmpredict(t_train, p_train, model);
T_sim2 = svmpredict(t_test , p_test , model);
%Performance Evaluation
error1 = sum((T_sim1' == T_train)) / M * 100;
error2 = sum((T_sim2' == T_test )) / N * 100;
%Data Sorting
[T_train, index_1] = sort(T_train);
[T_test , index_2] = sort(T_test );
T_sim1 = T_sim1(index_1);
T_sim2 = T_sim2(index_2);
%Plotting
figure
plot(1: M, T_train, 'r-*', 1: M, T_sim1, 'b-o', 'LineWidth', 1)
legend('Actual Value', 'Predicted Value')
xlabel('Predicted Samples')
ylabel('Prediction Results')
%Modify the file name here.
string = {'Comparison of Training Set Prediction Results (E3)';
['Accuracy = ' num2str(error1) '%']};
title(string)
grid
figure
plot(1: N, T_test, 'r-*', 1: N, T_sim2, 'b-o', 'LineWidth', 1)
legend('Actual Value', 'Predicted Value')
xlabel('Predicted Samples')
ylabel('Prediction Results')
%Modify the file name here.
string = {'Comparison of Test Set Prediction Results (E3)';
['Accuracy = ' num2str(error2) '%']};
title(string)
grid
%  Confusion Matrix
figure
cm = confusionchart(T_train, T_sim1);
cm.Title = 'Confusion Matrix for Train Data';
cm.ColumnSummary = 'column-normalized';
cm.RowSummary = 'row-normalized';

figure
cm = confusionchart(T_test, T_sim2);
cm.Title = 'Confusion Matrix for Test Data';
cm.ColumnSummary = 'column-normalized';
cm.RowSummary = 'row-normalized';
```