令和 6 年度　卒業研究

ベースの基礎練習支援システムの開発

Development of a support system for those who practice the basics of bass

函館工業高等専門学校
生産システム工学科　情報コース
東寺 琉之介
指導教員　東海林 智也

目次

1.1 概要

1.1.1 和文

ベースはギターに比べて弦の本数が少なく、覚えるべきコードもないため、楽器経験がない人でも手軽に始めやすいといわれている。しかし、初心者にとってはギターよりも弦が太いため、押さえるのが難しく、正しく音を鳴らすのが困難な場合がある。

そこで、本研究では、音声データの高速フーリエ変換（FFT）とデータ解析を用いて、弦が正しく押さえられているか、正しく音が鳴らせているかを判別し、ベース初心者の挫折を軽減することを目指す。

1.1.2 英文

The bass guitar is often considered easier to start with than the guitar, as it has fewer strings and does not require learning numerous chords. This makes it more accessible for those with no prior musical experience. However, beginners may find it challenging to press down on the strings properly and produce a clear sound, as bass strings are thicker compared to guitar strings.

Therefore, this study aims to reduce frustration among bass players and beginners by using fast Fourier transform (FFT) and data analysis on audio data to determine whether the strings are being pressed correctly and producing the intended sound properly.

1.2　研究背景

エレキベースは、重厚感のある低音で曲全体を支え、ライブ会場全体を包み込むほどのパワーを持つ、バンドにおいて重要なパートである[1]。ギターと比べて弦の本数が少なく、覚えるべきコードもないため、楽器経験がない人でも手軽に始められる。しかし、基礎練習や曲の練習が単調で飽きやすいため、ベース人口が少ないのが現状である。実際に、筆者自身もベースを始めたばかりの頃は、練習に飽きてしまい、日によって触る時間にばらつきがあった。

そこで本研究では、音声データの高速フーリエ変換（FFT）とデータ解析を用い、効率的な練習を可能にするシステムの開発を提案する。本システムでは、チューニング機能に加え、ビビり音やミュート音の判定を行うことで、ベース初心者をサポートできると考えられる。

ベース初心者が挫折しやすい要因としては、正しく弦を押さえる力が不足していたり、適切な弾き方が分からなかったりすることが挙げられる。また、ギターに比べ練習用アプリケーションが少ないため、効率的な基礎練習にたどり着きづらいという課題も存在する。

1.3　研究目的

本研究では、ベースの演奏音に対し、高速フーリエ変換（FFT）などの信号処理やデータ解析を行い、正しく演奏できているかを判別することで、効率的な練習を支援するシステムを開発する。これにより、初心者がより早く上達できるようにし、ベースをより手軽に始められる環境を目指す。

1.4　開発環境

PC：VivoBook_ASUSLaptop
OS：Windows11
CPU：Intel(R) Core(TM) i5-1035G1
実装 RAM：8GB

使用言語：Python3.11.2
開発環境：pip、 ffmpeg
使用ライブラリ： librosa、matplotlib、 numpy、ffmpeg

2. 関連技術

2.1 高速フーリエ変換

本研究では、高速フーリエ変換（Fast Fourier Transform: FFT)を用いて周波数解析を行った。この高速フーリエ変換は離散フーリエ変換(Discrete Fourier Transform :DFT)の計算時間を短くするためのアルゴリズムであり、DFT の定義は下記の通りである[2]。

$$f_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}kn} \quad ,(k = 0, \dots, N-1)$$

e は自然対数の底、j は虚数単位、π は円周率である。このように DFT では一つの周波数データに対してN^2のオーダーで演算を繰り返し行う必要があるため計算時間が長くなる問題がある。

一方 FFT では演算回数$N * \log_2 N$のオーダーまで減らすことができる。音声処理ではデータ数が膨大なため FFT が用いられることが多い。FFT の定義は下記の通りである[3]。

$$f_k = \sum_{n=0}^{M-1} x_{2n} w_M^{nk} + \sum_{n=0}^{M-1} x_{2n-1} w_M^{nk}$$

ここで $W_M = e^{-j\frac{2\pi}{M}}$は回転子と呼ばれ単位円を M 分割した点として定義されている。

さらに FFT を実行した後でパワースペクトルを求めて周波数解析を行った。P(f)をパワースペクトル、X(f)をフーリエスペクトルとするとその関係は下記の通りとなる。

$$P(f) = |X(f)|^2 = X(f) * X(f)$$

2.2 Numpy

Python では通常リストを用いて演算処理を行うため処理に時間がかかる。一方 Numpy を使用することで配列を使った計算が行えるため処理速度を向上できる[4]。そこで計算処理速度向上のため今回の研究では Numpy を使用した。

2.3 Librosa

Librosa は、音声および音楽信号処理を行うための Python ライブラリである。今回の研究では、音声データの読み込みと短時間フーリエ変換による周波数スペクトル解析をするために Librosa を用いた。

2.4 Matplotlib

Matplotlib は Python におけるグラフ描画のライブラリである。今回の研究では解析した音声データの周波数スペクトルを横軸が周波数、縦軸がパワーのグラフとして可視化するために Matplotlib を用いた。

3. 研究結果

3.1 実験内容

まず、ベースの開放弦の音を鳴らし、正しく判別できるかを検証する実験を行った。図 1 のプログラムを実行し、録音した音声データに対して Numpy を用いて高速フーリエ変換（FFT）を実施した。その後、作成したリストの周波数と比較し、結果を表示するとともに、Matplotlib を使用してグラフを描画した。

次に、ベースのミュート音とビビリ音をグラフで可視化し、それぞれの特徴を分析した。得られた特徴をもとに、ミュート音とビビリ音を判別するための条件を作成した。図 2 および図 3 が、そのプログラムである。

```python
# 特定の周波数に基づいて弦を検出する関数
def detect_string(frequencies, power_spectrum):
    # 閾値を超えるパワーを持つ周波数を検出するための閾値
    threshold_power = 20
    # 検出された周波数と最大パワーを初期化
    detected_frequency = None
    max_power = 0
    # 各周波数とパワーを比較して、閾値を超える周波数を検出
    for freq, power in zip(frequencies, power_spectrum):
        if power > threshold_power:
            # 閾値を超える周波数とパワーを記録
            detected_frequency = freq
            max_power = power
            break
    # 定義された弦の周波数リスト
    string_freqs = {
        'E': 75.35,
        'A': 96.88,
        'D': 139.9,
        'G': 183.0
    }
    # 検出された周波数に最も近い弦を特定
    detected_string = min(string_freqs,
    key=lambda s: abs(string_freqs[s] - detected_frequency))
    return detected_string, detected_frequency, max_power
```

図 1

```python
def detect_mute(frequencies, power_spectrum,
                power_threshold=5, count_threshold=40):

    count = 0
    for freq, power in zip(frequencies, power_spectrum):
        # 0から700Hzまでの周波数で、かつパワーが閾値を超える周波数をカウント
        if 0 <= freq <= 700 and power >= power_threshold:
            count += 1

    # 高周波数の数が閾値を超えた場合、ミュート音と判定
    mute_detected = count > count_threshold
    return mute_detected, count
```

図 2

```python
# 全体のパワー最大値に基づいてビビり音を検出
def detect_buzz_from_max_power(power_spectrum, high_power_count,
    max_power_threshold=80, count_threshold=40):

    # 全体のパワー最大値を取得
    max_power = np.max(power_spectrum)
    # 高パワー周波数の数が閾値を超える場合、ビビり音ではないと判定
    if high_power_count > count_threshold:
        buzz_detected = False
    else:
        # 最大値が閾値未満ならビビり音と判定
        buzz_detected = max_power < max_power_threshold
    return buzz_detected, max_power
```

図 3

3.2　実験結果と考察

弦の開放弦の判別結果を図 4・5 に示す。リストに登録された周波数は 75.35Hz であり、1 弦の開放弦の周波数も 75.35Hz であるため、正しく周波数が表示されていることが確認できた。

次に、録音したミュート音とビビリ音の判別を行った。それぞれの判別結果の詳細を図 6・図 7 に示す。どちらも正しく判別され、グラフから各音の特徴を正確に捉えられていることが分かった。

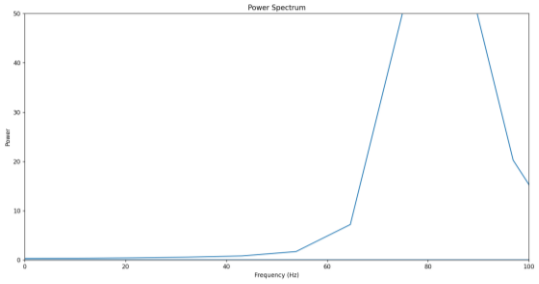

図 4 四弦のグラフ

```
検出された音程: E (Frequency: 75.35 Hz)
パワーの最大値: 385.1117858886719
ビビり音: False
高パワー周波数 (>= 5): 32
ミュート音: No
```

図 5 音の詳細

```
検出された音程: E (Frequency: 86.11 Hz)
パワーの最大値: 123.79864501953125
ビビり音: NO
高パワー周波数 (>= 5): 62
ミュート音: Yes
```

図 6 ミュート音の詳細

```
検出された音程: E (Frequency: 75.35 Hz)
パワーの最大値: 71.67320251464844
ビビり音: YES
高パワー周波数 (>= 5): 27
ミュート音: No
```

図 7 ビビリ音の詳細

4.　今後の展望

本研究では、ベースの音に対し、高速フーリエ変換（FFT）などの信号処理やデータ解析を行い、どのような音が鳴っているかを判別できるシステムを開発した。本システムによって、初心者が狙った音を出せているかを手軽に確認でき、チューナーとしても利用可能になった。しかし、細かい音の違いを判別できない課題が残るため、アルゴリズムの再検討が必要である。

今後の課題は、音声の判定精度を向上させること、さらにリアルタイムで判別できるようにすることである。

謝辞

本研究を進めるにあたり、ご指導くださいました指導教員の東海林智也准教授に感謝いたします。

参考文献

[1]ベースがバンド内で重要なパートだと言われる理由
https://25reuse.com/blog/788/

[2]高速フーリエ変換の並列処理に関する研究
https://core.ac.uk/download/pdf/229757255.pdf

[3]FFT を使ったバイオリン音のピッチ検出について
https://www.jstage.jst.go.jp/article/inctkiyoupre/41/0/41_KJ00005074107/_pdf/char/ja

[4]Numpy
https://helve-blog.com/posts/python/numpy-fast-fourier-transform/