計算機科学実験 4 音声 レポート 1

安済翔真

2023 年 12 月 26 日

目次

1　機能・プログラム説明

課題 1 では音声の基本的な情報を表示するプログラムを作成した。この章では作成した具体的な機能について説明する。画面全体は図 1 のようになっている。下の Voice Change や Tremolo の部分は課題 2 の機能のため、レポートでは説明しない。

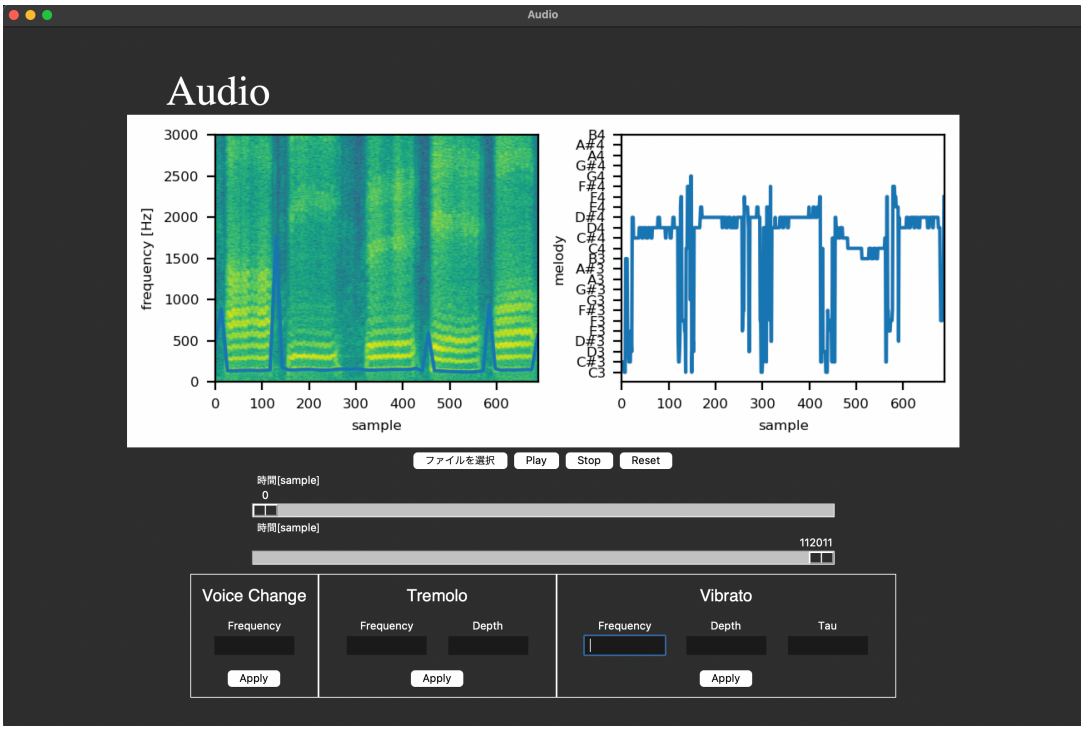

図 1　画面全体

1.1　ファイル選択

ファイル選択機能を実装した。ファイル選択ボタン (図 2) を押すと、ファイル選択ダイアログが表示される。選択したファイルを読み取り、表示対象の音声データとして扱う。

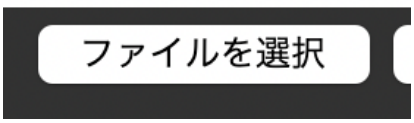

図 2　ファイル選択ボタン

ファイル読み取りのプログラムはコード 1, 2 の通りである。コード 1 は tkinter の filedialog を用いてファイル選択ダイアログを表示し、選択したファイル名を取得する。コード 2 は librosa を

用いて音声データを読み取る。

1　ファイル読み取り 1

```
filename = tk.filedialog.askopenfilename()
self._c.load_file(filename)
```

2　ファイル読み取り 2

```
def load_waveform(filename):
  x, _ = librosa.load(filename, sr=SR)
  return x
```

1.2　スペクトログラムの表示

選択したファイルの音声のスペクトログラムを表示する機能を実装した。コード 3 はスペクトログラムを計算するプログラムである。numpy の hamming 関数を用いて窓掛けを行い、FFT を行う。その後、対数振幅スペクトルを計算し、配列に保存する。

3　スペクトログラム計算

```
def spectrogram(waveform, size_frame, size_shift):
  spectrogram = []
  hamming_window = np.hamming(size_frame)

  for i in np.arange(0, len(waveform) - size_frame, size_shift):
    idx = int(i)
    x_frame = waveform[idx: idx + size_frame]

    # 窓掛けしたデータを FFT
    fft_spec = np.fft.rfft(x_frame * hamming_window)

    # 振幅スペクトルを対数化
    fft_log_abs_spec = np.log(np.abs(fft_spec))

    # 配列に保存
    spectrogram.append(fft_log_abs_spec)
  return spectrogram
```

コード 4 はスペクトログラムを表示するプログラムである。matplotlib の imshow 関数を用いてスペクトログラムを表示する。

4　スペクトログラム表示

```
self.__ax.imshow(
    np.flipud(np.array(spectrogram).T),
    extent=[0, len(spectrogram), 0, SR / 2],
```

```
    aspect='auto',
    interpolation='nearest',
)
self.__ax.set_ylim(0, 3000)
```

1.3 基本周波数の表示

選択したファイルの音声の基本周波数を表示する機能を実装した。コード 5 は基本周波数を計算するプログラムである。まず、numpy の correlate 関数を用いて自己相関係数を計算する。その後、ピークを検出し、ピークのインデックスを取得する。最後に、ピークのインデックスから基本周波数を計算する。

5　基本周波数計算

```
def get_f0(waveform, sampling_rate):
  autocorr = np.correlate(waveform, waveform, 'full')
  autocorr = autocorr[len(autocorr) // 2:] # 不要な前半を捨てる

  # ピークを検出
  peak_indices = [i for i in range(len(autocorr)) if is_peak(autocorr, i)]
  peak_indices = [i for i in peak_indices if i != 0] # 最初のピークは除く

  if len(peak_indices) == 0:
    return 0

  max_peak_index = max(peak_indices, key=lambda index: autocorr[index])

  # 基本周波数を推定
  f0 = sampling_rate / max_peak_index
  return f0
```

コード 6 は基本周波数を表示するプログラムである。matplotlib の plot 関数を用いて基本周波数を表示する。スペクトログラムと同じ x 軸を用いるため、x 軸のデータはスペクトログラムのデータを用いる。

6　基本周波数表示

```
  x_data = np.linspace(0, len(spectrogram), len(f0s))
  self.__ax.plot(x_data, f0s)
```

1.4 音程の表示

選択したファイルの音声の音程を表示する機能を実装した。コード 7 は音程を計算するプログラムである。nn2hz 関数は MIDI ノートナンバーを周波数に変換する関数である。shs 関数はスペク

トルを用いて音程を計算する関数である。候補の音程 (NOTES) の各周波数について、スペクトルの対数振幅スペクトルを用いて尤度を計算する。尤度を計算する際は、対象の音程の「倍音」の強さも考慮した。特に、倍音の強さは 0.8 の指数関数的な減衰を考慮することで精度が大幅に向上した。

7　音程計算

```
def nn2hz(nn):
  return 440.0 * 2 ** ((nn - 69) / 12.0)

def shs(spectrum, sample_rate, size_frame):
  likelihood = np.zeros(len(NOTES))
  for i in range(len(likelihood)):
    base_freq = nn2hz(NOTES[i])
    for j in range(1, 16):
      freq = base_freq * j
      fft_idx = int(freq * size_frame / sample_rate)
      likelihood[i] += 0.8**j * np.exp(spectrum[fft_idx])
  return NOTES[np.argmax(likelihood)]
```

コード 8 は音程を表示するプログラムである。matplotlib の plot 関数を用いて音程を表示する。yticks で y 軸のラベルを設定している。ラベルの内容は C3 から B4 までの音程を表示することで、表示される音程がわかりやすくなるようにした。

8　音程表示

```
  plt.plot(list(map(lambda x: x - NOTES[0], melody)))
  plt.yticks(np.arange(24),
          list(["C3", "C#3", "D3", "D#3", "E3", "F3", "F#3", "G3",
                "G#3", "A3", "A#3", "B3", "C4", "C#4", "D4", "D#4",
                "E4", "F4", "F#4", "G4", "G#4", "A4", "A#4", "B4"])
          )
```

1.5　音声の再生・停止

選択したファイルの音声を再生・停止する機能を実装した。コード 9 は音声を再生するプログラムである。既に音声が再生中の場合は、一旦再生を停止してから再度再生をすることで、同じ音声を連続して再生することができる。

9　音声再生

```
  if self.__play_obj is not None:
    self.__play_obj.stop()
  self.__play_obj = self.__wave_obj.play()
```

コード 10 は音声を停止するプログラムである。音声が再生中の場合のみ、音声を停止する。

10　音声停止

```
if self.__play_obj is None:
    return
self.__play_obj.stop()
self.__play_obj = None
```

1.6　表示区間・再生区間の限定

スペクトログラムや音程の表示、及び音声の再生区間を限定する機能を実装した。コード 11 は表示区間・再生区間を限定するプログラムである。start の値は end の値より大きくならないようバリデーションを行った。

11　表示区間・再生区間の限定

```
def set_start(self, start: int):
    if start >= self.__end - self.MIN_SIZE:
        return
    self.__start = start

def set_end(self, end: int):
    if end <= self.__start + self.MIN_SIZE:
        return
    self.__end = end
```

コード 12 は区間を限定する UI を表示するプログラムである。tkinter の Scale を用いて、スライダーの UI を表示している。このコードにより、図 3 のような UI が表示される。

12　区間を限定する UI

```
self.slider = tk.Scale(
    command=self.__command,
    master=self._frame,
    from_=from_,
    to=to,
    label=u' 時間 [sample]',
    orient=tk.HORIZONTAL,
    length=700,
    width=15,
)
```

図 3　区間を限定する UI

2　実行例とテスト

2.1　ファイル選択

ファイル選択ボタンを押すと、図 4 のようなファイル選択ダイアログが表示される。これにより、ファイル選択ボタンが正しく動作していることが確認できる。

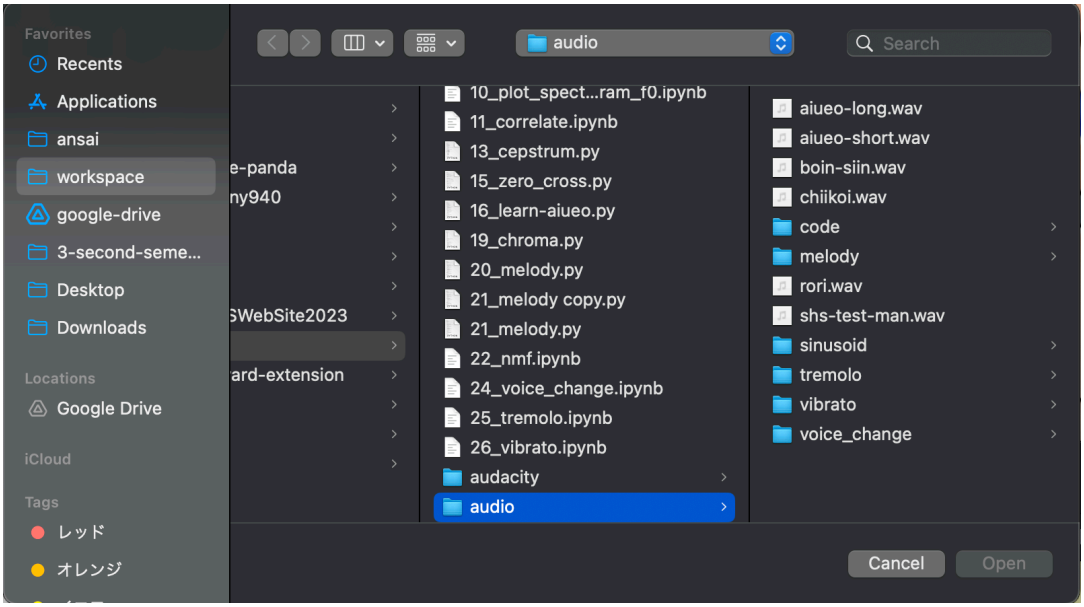

図 4　ファイル選択ダイアログ

2.2　スペクトログラムの表示

ファイルを選択すると、図 5 のようにスペクトログラムが表示される。図 5 は「あいうえお」の音声のスペクトログラムである。これにより、スペクトログラムが正しく表示されていることが確認できる。

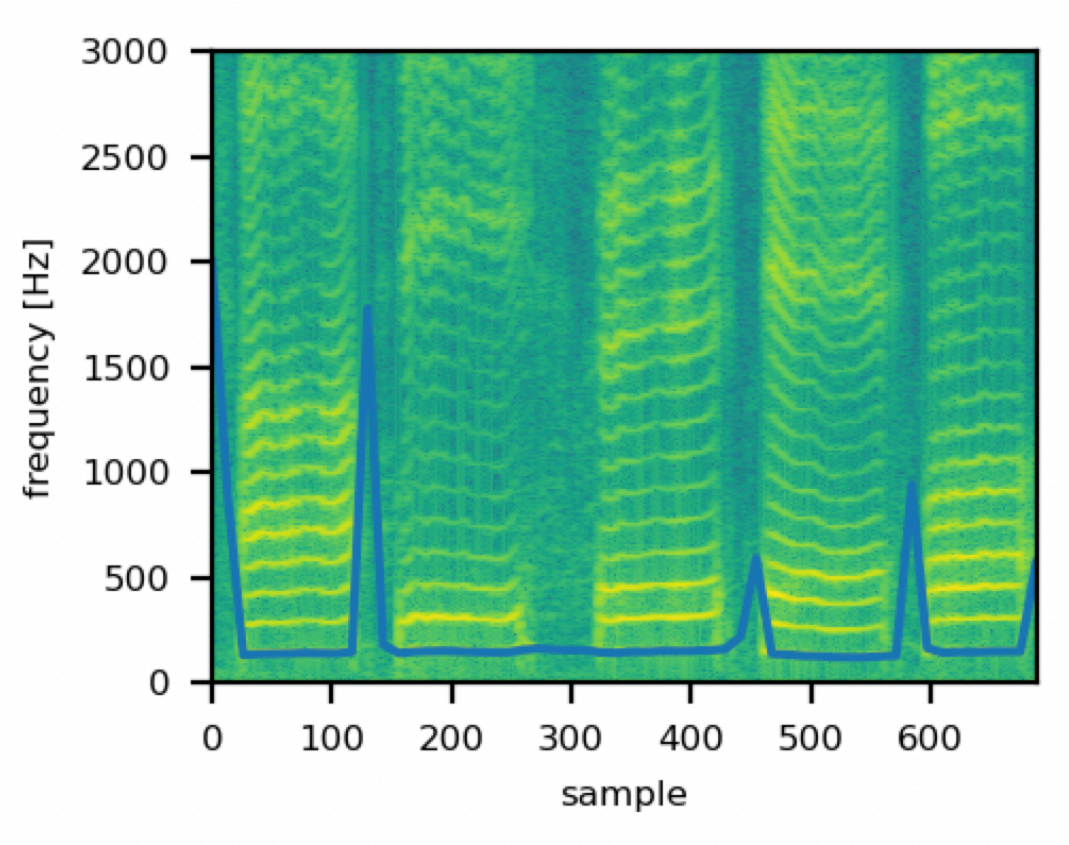

図 5　スペクトログラム

2.3　基本周波数の表示

ファイルを選択すると、図??のように基本周波数が表示される。

2.4　音程の表示

2.5　音声の再生・停止

2.6　表示区間・再生区間の限定

3　工夫点

3.1　ファイル選択機能

3.2　区間選択機能

start の値が end の値より大きくならないようバリデーションを行った。

3.3 音程の表示

3.4 音声の再生・停止機能

4 考察