音響課題 1: 音響信号可視化 GUI 作成

学籍番号: 1029332978 氏名: 上野山遼音

2024 年 1 月 12 日

目 次

1　システム概要

今回の課題 1 では，音響信号ファイルを読み込み，音響信号のさまざまな情報を表示する GUI の作成を行った．要求仕様の

- 音響信号のスペクトログラム
- 音響信号の基本周波数
- 母音推定

に加え，python のライブラリである matplotlib と tkinter を用いて指定された位置の音響信号のスペクトルを表示する機能を追加した．スタート画面では処理対象の音響信号ファイルを選択することができる．

アプリ概観

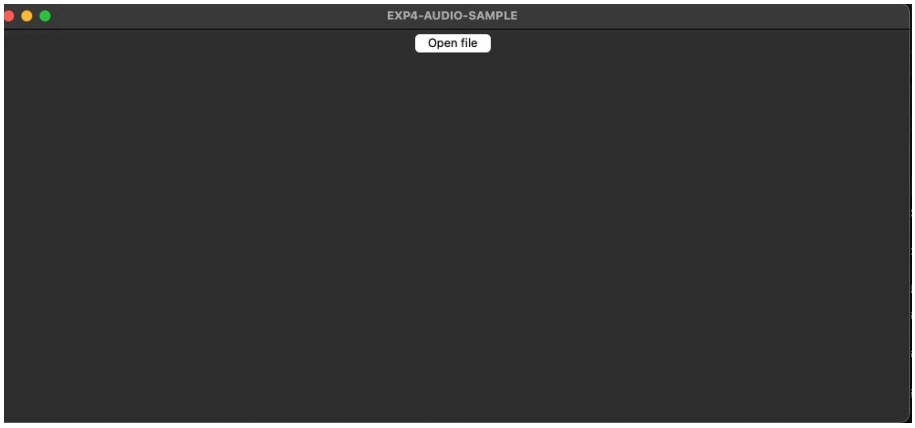

図 1: 初期画面

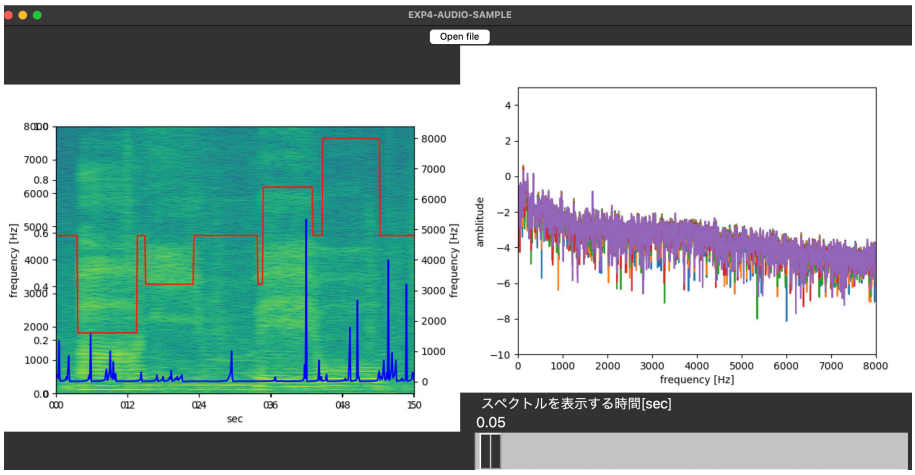

図 2: アプリのホーム画面

2　プログラムの説明

2.1　プログラムの構成

今回の課題 1 では，以下のような関数構成となっている．

```
(main)
├── openfile
  ├── calc
  │   ├──── is_peak
  │   └──── vowel_recognition
  ├──  draw_data
  └── _draw_spectrum
```

2.2　各関数の説明

2.3　main

．GUI アプリケーションのメイン関数．tkinter の初期配置や，ボタンを押した際呼び出される関数を定義している．実際には定義していないが，便宜上 main 関数として扱っているものである．

Listing 1: main 関数

ソースコード

```
1
2 def open_file(event): (略)
3
4
5 # を初期化 Tkinter
6 root = tkinter.Tk()
7 root.wm_title("EXP4-AUDIO-SAMPLE")
8
9 # ファイルを開く
10 open_file_button = tkinter.Button(root, text='Open file')
11 open_file_button.pack(side=tkinter.TOP)
12 open_file_button.bind('<Button-1>', open_file)
13
14 # # 再生ボタン（生やす予定が実装が間に合わなかったものである）
15 # play_music_button = tkinter.Button(root, text='Play')
16 # play_music_button.pack(side=tkinter.END)
17 # play_music_button.bind('<Button-1>', play_music)
18
19 tkinter.mainloop()
```

2.3.1　openfile

openfile 関数は，tkinter の filedialog を用いて音響信号ファイルを選択し，それを元に音響信号のスペクトログラム，基本周波数，母音推定を行う．関数である，内部には is_peak 関数，draw_spectrum 関数が存在する．

Listing 2: openfile 関数

ソースコード

```
def open_file(event):
  # で読み込む tkinter
  fTyp = [("wav file", "*.wav")]
  iDir = os.path.abspath(os.path.dirname(__file__))
  global input_file
  input_file = tkinter.filedialog.askopenfilename(filetypes=fTyp, initialdir=iDir)
  # 音声ファイルを読み込む
  x, _ = librosa.load(input_file, sr=SR)

  # ファイルサイズ（秒）
  duration = len(x) / SR

  # ハミング窓
  hamming_window = np.hamming(size_frame)

  def calc(x):（略）

  def _draw_data(spectrogram, hz_list):（略）

  def _draw_spectrum(v):（略）

  # スペクトルを表示する領域を確保
  # ax2, canvs2 を使って上記のコールバック関数でグラフを描画する
  fig2, ax2 = plt.subplots()
  canvas2 = FigureCanvasTkAgg(fig2, master=frame2)
  canvas2.get_tk_widget().pack(side="top") # "topは上部方向にウィジェットを積むことを
      意味する"

  # スライドバーを作成
  scale = tkinter.Scale(
      command=_draw_spectrum, # ここにコールバック関数を指定
      master=frame2, # 表示するフレーム
      from_=0, # 最小値
      to=duration, # 最大値
      resolution=size_shift/SR, # 刻み幅
      label=uスペクトルを表示する時間'[sec]',
      orient=tkinter.HORIZONTAL, # 横方向にスライド
      length=600, # 横サイズ
      width=50, # 縦サイズ
      font=("", 20) # フォントサイズは 20に設定 px
  )
  scale.pack(side="top")
```

2.3.2 is_peak

is_peak 関数は，基本周波数の推定を行う関数である．基本周波数の推定は，スペクトルのピークを求めることで行うことができるので (cf. 演習 11)，これを用いてシフト幅を変えながら基本周波数を推定する．

Listing 3: is_peak 関数

ソースコード

```
def is_peak(a, index):
if index == 0 or index == len(a)-1:
    return False
```

```
4 if a[index-1] < a[index] and a[index] > a[index+1]:
5     return True
6 else:
7     return False
```

2.3.3 calc

calc 関数は，openfile 関数内で呼び出される関数である．calc 関数は，openfile 関数内で読み込んだ音響信号を元にスペクトログラム，基本周波数，母音推定を行う．引数には librosa で読み込んだ音響信号 x を与え,，返り値にスペクトログラム，基本周波数，母音推定の結果のデータを返す.

Listing 4: calc 関数

ソースコード

```
1 def is_peak(a, index)（略）
2
3
4 # スペクトログラムを保存する list
5   spectrogram = []
6   hz_list = []
7   pred = []
8   autocorr = np.correlate(x, x, 'full')
9
10  # 不要な前半を捨てる
11  autocorr = autocorr[len(autocorr) // 2:]
12  # フレーム毎にスペクトルを計算
13  for i in np.arange(0, len(x)-size_frame, size_shift):
14
15      # 該当フレームのデータを取得
16      start_idx = int(i) # のインデクスはなのでに変換 arangefloatint
17      end_idx = start_idx+size_frame
18      x_frame = x[start_idx: end_idx]
19
20      # スペクトル
21      fft_spec = np.fft.rfft(x_frame * hamming_window)
22      fft_log_abs_spec = np.log(np.abs(fft_spec))
23      spectrogram.append(fft_log_abs_spec)
24
25      # 基本周波数
26      # 区間ごとの自己相関を取得
27      autocorr_interval = autocorr[start_idx:end_idx]
28          # ピークのインデックスを抽出する
29      peakindices = [i for i in range(len(autocorr_interval)) if is_peak(
            autocorr_interval, i)]
30          # インデックス 0 がピークに含まれていれば捨てる
31      peakindices = [i for i in peakindices if i != 0]
32          # 自己相関が最大となるインデックスを得る
33      max_peak_index = max(peakindices, key=lambda index: autocorr_interval[index])
34      # max_peak_index_interval = np.argmax(autocorr_interval)
35      # 区間ごとの周波数を計算して出力
36      freq_interval = SR / max_peak_index
37      hz_list.append(freq_interval)
38
39      # 母音の判定
40      cep = np.real(np.fft.rfft(fft_log_abs_spec))
41      cep = cep[:13]
```

```
42        likelihood_a = calc_likelihood(cep, mu_a, var_a)
43        likelihood_i = calc_likelihood(cep, mu_i, var_i)
44        likelihood_u = calc_likelihood(cep, mu_u, var_u)
45        likelihood_e = calc_likelihood(cep, mu_e, var_e)
46        likelihood_o = calc_likelihood(cep, mu_o, var_o)
47        likelihood = [likelihood_a, likelihood_i, likelihood_u, likelihood_e,
              likelihood_o]
48        pred.append((likelihood.index(max(likelihood))+ 1)* SR / 10)
49    return spectrogram, hz_list, pred
```

2.3.4 draw_data

draw_data 関数は，calc 関数で得られたスペクトログラムを GUI アプリケーションの左側に表示する関数である．引数としてスペクトログラムと基本周波数，母音推定の結果のリストのデータを与え，返り値は持たない．

Listing 5: draw_data 関数

ソースコード

```
1  def _draw_data(spectrogram, hz_list, pred):
2    # まずはスペクトログラムを描画
3    fig, ax = plt.subplots()
4    canvas = FigureCanvasTkAgg(fig, master=frame1) # に対象とするを指定 masterframe
5    ax1 = fig.add_subplot(111)
6    ax1.set_xlabel('sec')
7    ax1.set_ylabel('frequency [Hz]')
8    ax1.imshow(
9        np.flipud(np.array(spectrogram).T),
10       extent=[0, duration, 0, 8000],
11       aspect='auto',
12       interpolation='nearest'
13   )
14   # 続いて右側の軸を追加して，音量を重ねて描画 y
15   ax3 = ax1.twinx()
16   ax3.set_ylabel('frequency [Hz]')
17   x_data = np.linspace(0, duration, len(hz_list))
18   ax3.plot(x_data, hz_list, c='b')
19   ax3.plot(x_data, pred, c='r')
20   canvas.get_tk_widget().pack(side="left") # 最後にに追加する処理 Frame
```

2.3.5 draw_spectrum

draw_spectrum 関数は，指定された位置の音響信号のスペクトルを GUI アプリケーションの右側に表示する関数である．引数としてスライダーの値を与え，返り値は持たない．

Listing 6: _draw_spectrum 関数

ソースコード

```
1  def _draw_spectrum(v):
2
3    # スライドバーの値からスペクトルのインデクスおよびそのスペクトルを取得
4    index = int((len(spectrogram)-1) * (float(v) / duration))
5    spectrum = spectrogram[index]
6
7    # 直前のスペクトル描画を削除し，新たなスペクトルを描画
```

```
    plt.cla()
    x_data = np.fft.rfftfreq(size_frame, d=1/SR)
    ax2.plot(x_data, spectrum)
    ax2.set_ylim(-10, 5)
    ax2.set_xlim(0, SR/2)
    ax2.set_ylabel('amblitude')
    ax2.set_xlabel('frequency [Hz]')
    canvas2.draw()
```

3　工夫した点，今後の展望

既存のコードを参考にしながら，できるだけ拡張性を意識したコードにした．具体的には，関数配置を見直し，関数の再利用性を高めた．tkinter での UI 追加を行う場合，これまでの記法を流用して書きやすいようになっている．また，サンプルコードを参考にしながら，スライダーを作成しその位置に対応するスペクトルを表示する機能もできた．今後余裕があれば，再生ボタンを追加し再生しながらスペクトルを出したり，波形を出したりできるようにしたい．

付録: 全ソースコードはこちら→ https://github.com/Mntisgod/isle4-audio.