



成果発表 チーム 6

下田音里・瀧和樹・相田陽香

開発環境（JUPYTER NOTEBOOK ・ VSCODE）

使用言語：Python

- pyaudio
- Pydub
- librosa
- Scipy

処理の流れ

音声を録音

フィルター処理

音声加工

フィルター処理

音声出力

開発環境（JUPYTER NOTEBOOK ・ VSCODE）

使用言語：Python

- pyaudio
- Pydub
- librosa
- Scipy

処理の流れ

音声録音

フィルター処理

音声加工

フィルター処理

音声出力

プログラム説明

- ピッチ変更
- フィルタ（ノイズ除去）
- 付加効果



プログラム(一部抜粋)

- ピッチ変更

- 値の正負で音域を操作
- 1オクターブ上 = 12
- 1オクターブ下 = -12

```
wavf = 'criminal.wav'
wr = wave.open(wavf, 'r')
fr = wr.getframerate()
ch = wr.getnchannels()
print("Channel: ", ch)

wavf = 'criminal.wav'
fn = wr.getnframes()

def stretch(data,rate=1):
    input_length = fn

    data=librosa.effects.time_stretch(data,rate)
    return data

#1オクターブ上げる
def pitch(data,sr,pitch_factor):
    return librosa.effects.pitch_shift(data,sr,pitch_factor)

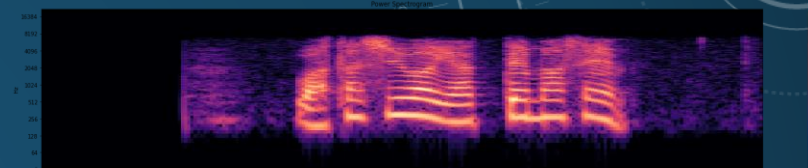
def load_audio_file(file_path):
    input_length = fn
    data = librosa.core.load(file_path,sr=fr)[0]
    |
    if len(data)>input_length:
        data = data[:input_length]
    else:
        data = np.pad(data, (0, max(0, input_length - len(data))), "constant")
    return data

sr=fr
data = load_audio_file("criminal.wav")

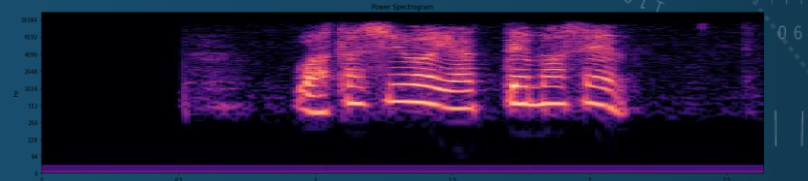
#1オクターブ上げる
pitch_data=pitch(data,sr,12.0)

# wavファイル保存
sf.write("output2.wav", pitch_data, sr, subtype="PCM_16")
```

元音声



1オクターブ上げる



プログラム(一部抜粋)

- フィルタ
 - ハイカットフィルター
 - ローカットフィルター
- 一定以下の振幅を持つデータの
カット

```
fft = fftpack.fft(wave)    # FFT

# 元データの隔離,実際に使うのはfft
fft_original = copy.copy(fft)

#フィルタ処理
lf = 100    #Hz下限
hf = 5000   #Hz上限
ac = 0.5    #振幅下限
#サンプリング周期
samplerate = N / AudioLength
fft_axis = np.linspace(0, samplerate, N)    # 周波数軸を作成
fft_amp = np.abs(fft / (N / 2))    # 振幅成分を計算
|

#上下のカット
fft[(fft_amp < ac)] = 0
fft[(fft_axis < lf)] = 0
fft[(fft_axis > hf)] = 0

# IFFT処理
ifft_time = fftpack.ifft(fft) #この時点ではまだ複素数
```

プログラム(一部抜粋)

- 残響付加
- 様々な音
- 動物や物音効果音など
- ボイスチェンジにどう変化を与えるか

```
input_length = 480000
data = librosa.core.load(file_path, sr=18000)[0] #, sr=16000
if len(data) > input_length:
    data = data[:input_length]
else:
    data = np.pad(data, (0, max(0, input_length - len(data))), "constant")
return data

sr=18000
data = load_audio_file("haji.wav")
pitch_data=pitch(data,sr,-3.0)

sf.write('change.wav', pitch_data,sr)

#wavファイル読み込み
fs1, x = wav.read("change.wav")
fs2, h = wav.read("tu.wav")
x = x / 32768
h = h / 32768

#フィルタ設計
filter = sig.firls(13, (0, 8000, 12000, 24000),
                  (1, 1, 0, 0), fs = 48000)

#畳み込み
y = sig.convolve(x, h)
y_int = (y * 32768).astype('i2')

#wavファイル保存
wav.write("change3.wav", fs, y_int)
```

The background is a solid dark blue. On the left side, there is a large, semi-circular scale with tick marks and numbers ranging from 140 to 260. Several concentric circles and arcs are scattered across the image, some with arrows indicating a clockwise or counter-clockwise direction. The overall aesthetic is technical and modern.

実演

実演

1. 元音声
 2. フィルター1回目（ノイズ除去）
 3. ピッチ変更
 4. フィルター2回目（ノイズ除去）
-
1. フィルタ1回目を行わなかったもの
（ピッチ変更のみ）



まとめ

- ・入力音声データを加工するプログラムを作成した
 - ・リアルタイム変換は実装せず
 - ・GUI実装せず
 - ・フーリエ変換したものをもう一度WAVへ変換する作業が難しかった
 - ・環境構築・プログラム統合に手間取った
-
- ・**一週間では足りなかった！**
(やりたいことはまだあったが、手が回らなかった…)

The background is a solid dark blue color. It features several faint, light blue circular patterns. On the left side, there are concentric circles with radial lines, resembling a protractor or a circular scale. Some of these circles have degree markings: 150, 160, 170, 180, 190, 220, 230, 240, 250, and 260. There are also smaller circles with arrows indicating a clockwise direction. The overall design is technical and geometric.

ご清聴ありがとうございました