Matthew Schumaker
Charlie Fitzgerald
Last Updated: 3/30/2022

Acoustic Analysis User Manual:

NOTE: This is a living document. Features are subject to change and controls to be migrated. Please reference the most up to date user manual for your branch.

Feature Overview:

Currently, this script does spectral analysis using Librosa's STFT method as a Fourier Transform agent. Using the data acquired from this, it is capable of picking out the n loudest partials requested by the user, using a spectral denoising algorithm that checks if the frequency is a local maximum (i.e. amplitude for bin is greater than its immediate neighbor frequency bins). These partial frequencies are then written to an output csv file in order of their amplitudes at each moment in time.

The script also performs a spectral centroid calculation, denoising time frames with maximum amplitudes less than 1.

Command Line Controls:

The typical user input is as follows:

```
basicfft.py <filename> <frames per second> <n loudest partials>
```

Wherein:

`<filename>` designates the .wav file in the `acoustic_feature_analysis/soundfiles` directory to analyze.
`<frames per second>` denotes the number of time frames per second to comb for n_loudest partials (e.g. `10` would denote a value of 10 frames per second, or .1 second increments between frequency representation.

- Note: the fps currently affects the hop length of the fft analysis, which may interfere with time accurate frequency representation
- The `<frames per second>` indicator may be replaced with an input value of `os`, which will cause the program to analyze the loudest partials at onset times detected using Librosa's Onset Detect method and a very small hop length.
    - This seems to be bugged out with the newest peak algorithm implementation

`<n loudest partials>` indicates the number of partial frequencies the user would like to retrieve from the input file.

- Note: If there are fewer peaks for any given frame than the indicated `<n loudest partials>` the program will only output the number of peaks detected that have an amplitude > 1

Inner Program Controls:

This section describes the various controls that have not been migrated to the command line but may be useful for user data manipulation. These controls include the following:

```
18    # controls for csv info
19    write_amplitudes = True
20    write_notes = True
21    write_transcribed_note = True # only applies if write notes is true
```

`write_amplitudes`, `write_notes`, and `write_transcribed_note` are boolean flags indicating to the CSV writer module which fields to include in the output file.

```
def main(filename, fps, n = 2):

    # load signal(s) from default audio directory 'soundfiles/'
    signal_lib, sample_rate_lib = load_audio_librosa(filename) # in stereo?
    # signal_scip, sample_rate_scip = load_audio_scipy('fft_test.wav')

    if(fps == "os"):
        times, frequencies, amplitudes = onset_fft_analysis(signal = signal_lib, sample_rate=sample_rate_lib, n=n)
        csv_write(filename, True, 0, n, times, frequencies, amplitudes)
    else:
        # times, frequencies, amplitudes = librosa_fft_analysis( signal=signal_lib, sample_rate=sample_rate_lib, fps=int(fps), n=n)
        centroid_calculation(signal = signal_lib, sample_rate=sample_rate_lib, fps = int(fps))
        # print(librosa.feature.spectral_centroid(y= signal_lib, sr= sample_rate_lib))
        times, frequencies, amplitudes = n_loudest_peaks( signal=signal_lib, sample_rate=sample_rate_lib, fps=int(fps), n=n)
        csv_write(filename, False, int(fps), n, times, frequencies, amplitudes)
```

The `main` function is relatively small and only executes a couple of commands to perform all of its operations. As you can see there is a switch for the `os` mode mentioned earlier, and in the else statement you can see the default behavior for the script. Currently in the default behavior there is a centroid calculation which is outputted to the terminal, and an `n_loudest_peaks` call which returns the relevant peak amplitude data. It may be useful to replace these values with literals if looking to experiment with various non-control line parameters.

```
def n_loudest_peaks(signal, sample_rate, n_fft=4096, fps=3, n=2):
    hop_length = int(sample_rate/fps)
    fft_set = np.abs(librosa.stft(signal, center=False, n_fft=n_fft, hop_length = hop_length))
    fft_set = np.abs(librosa.stft(signal, center=False, n_fft=n_fft, hop_length = hop_length))
```

The `n_loudest_peaks` method contains various controls in its header, as well as some default literals if none are provided by the call in main. The `n_fft, fps,` and `n` values are all capable of being overwritten by main or changed here, whichever you prefer. By default, `fps` and `n` values are passed in by main and probably shouldn't be changed here.

- Example FFT Bin Calculation:
    - Using a default `n_fft` of 4096 with a default sample rate of 22050 will result in 2048 frequency bins to analyze from 0-11025 Hz at each time frame interval denoted by `fps`

TODO Controls to Implement:

- output mode flags for Houdini/vex with traditional separated array formatting