HW #1
Audio Signal Processing & Musical Key Detection

Li Su

March 18, 2024

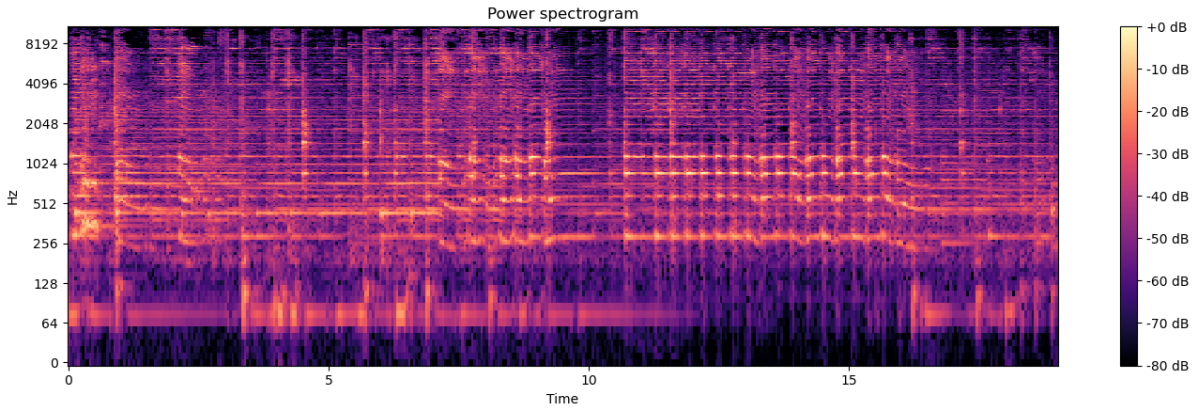
Prerequisite: the following Python libraries are suggested for this assignment.

- `librosa`, a Python library for music and audio signal processing.
- `pretty-midi`, a Python library for MIDI signal processing
- `mir_eval`, a Python library for MIR evaluation
- `soundfile`, a Python library for audio I/O

1. Basic audio signal processing

   (a) (10%) Find an audio file you like and plot the spectrogram. You may directly use the `librosa` functions. Explain the audio content shown on the spectrogram. Be aware of the x-axis and y-axis grids and units; they should be correct and informative.

   (Example: This example[1] is a solo excerpt of *sitar*, a kind of plucked string instrument in Indian classical music. There is also a percussion instrument, *tabla*, in the accompaniment. Sitar has 19 strings: only two are used for playing melody, while other strings are used for playing accompaniment or just for providing a resonating harmonic background. In this recording, we can hear that the melody is mainly at f_0 = D4 (293.66 Hz), which is the note the first string is tuned. This can be clearly seen in the spectorgram. Also, we can see a persistent harmonic patterns (horizontal lines) at $2f_0$, $3f_0$ and $4f_0$. It seems that the resonance also produce at a low frequency (a sub-harmonic at $f_0/4$). The plucking of sitar and the tabla both form the vertical patterns in the spectrogram. The recording quality is bad, and this can also be observed by the low color contrast between the spectral peaks and floors (around 20 dB).)



[1] https://drive.google.com/file/d/1rA0vMUlfsRKenP_b5t4QmZrllmRYx6EQ/view?usp=sharing

(b) (5%) **Instantaneous frequency.** It was mentioned in our class that the frequency of a time-domain sinusoidal signal $\sin(2\pi f_0 t)$, where t representing the time, is exactly f_0. Here, let us discuss a more general case. Consider a time-domain signal $x(t)$:

$$x(t) = \sin(\phi(t)), \tag{1}$$

where $\phi := \phi(t)$ is the *phase function*. The frequency of this signal, denoted as f, is defined as the time derivative of phase:

$$f(t) := \frac{1}{2\pi}\frac{d\phi}{dt}. \tag{2}$$

It should be noted that $f := f(t)$ is also a function of time. Therefore the frequency at time t is also called the *instantaneous frequency* (IF) of x at t. Based on this definition, let us consider the following two signals:

$$x_1(t) = \sin\left(2000t^2\right), \tag{3}$$

$$x_2(t) = \sin\left(2000t + 10\sin(2.5t^2)\right). \tag{4}$$

According to (2), the IF of x_1 as a function of time is $f_1(t) = \frac{2000t}{\pi}$ Hz – you may say the frequency of x_1 is 2 kHz at 3.1415 sec and 4 kHz at 6.283 sec. Such a kind of signal is called a *chirp signal*, meaning that its IF increases with time and sounds like a bird's chirp. Please derive the IF of x_2. (Hint: if you already forgot the claculus course you took, Wolfram alpha[2] is your good friend.)

(c) (15%) For $t \in [0, 10]$ (that means, the signal is of 10 seconds length from $t = 0$), write x_1 and x_2 into .wav files with a sampling frequency at 8 kHz. You may use the `write` function in `soundfile`. Listen to both of them and describe what you hear. Can you hear the changes of IFs in them? Can you hear the IF of x_1 keep increasing throughout the 10 seconds? If not, what happens on it?

(d) (15%) Plot the spectrograms of x_1 and x_2 using the window size as 2048 samples and hop size as 512 samples. You may use the `stft` function in `librosa`. Describe what you see (bonus: you may also overlay the IFs of the two signals on the two spectrograms). Does the patterns on the spectrograms fit the true IF curves?

2. Musical key detection

Music key or tonality are related to two fundamental aspects of a musical scale: the *tonic* and the *mode*. Considering a musical scale adopted in a piece of music, the tonic is the first note of the scale, while the mode refers to the type of the scale. The major and minor modes (or, major and minor scales) are two most often seen mode types in the diatonic scale system. A musical key is identified by the tonic note and the tonic chord. There are some simplified yet imprecise ways to identify a musical key. For example, the tonic is usually recognized as the first or the last note of a music piece. Moreover, if the chord corresponding to the tonic (i.e., the tonic chord) is a major chord, the music piece is then in major key. On the other hand, if the tonic chord is a minor chord, the music piece is then in minor key. However, such over-simplified ways in finding key usually fails in most of the real-world musical data. As a high-level concept, musical key is long-term and context dependent. According to the musical context, a music piece may have a global key as its main key, and local keys which may change several times throughout the music piece. The detection of global and local keys is still not yet a solved problem in MIR.

[2] https://www.wolframalpha.com/

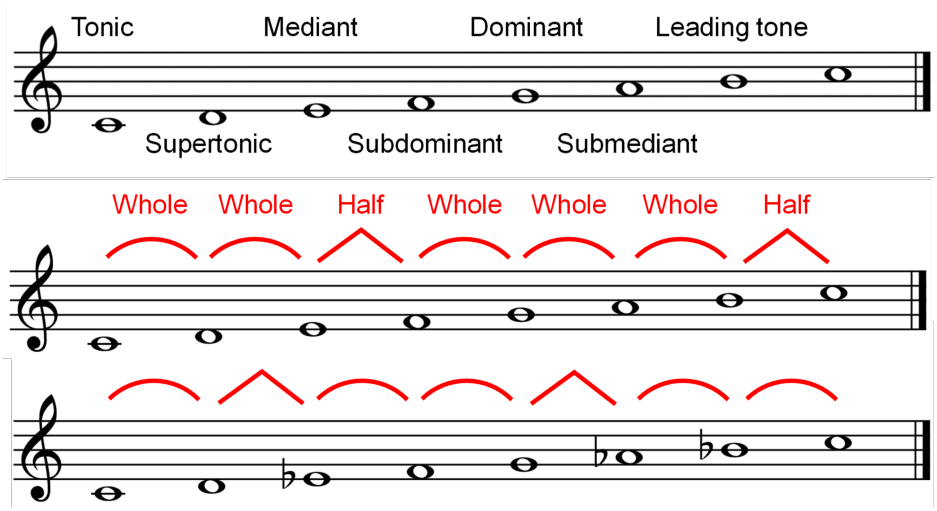


Figure 1: The functions of notes in a diatonic musical scale (upper), the configuration of a C major scale (middle), and the configuretion of a C minor scale (lower). Figure from Meinard Müller, *Fundamentals of Music Processing*, Chapter 5, Springer 2015.

In this assignment, let us design some global and local key detection algorithms for global and local music key detection for both audio and symbolic data, with full or limited contextual information.

The concept of musical key

- Denote T as a whole step and S a half-step (i.e., a semitone), a major scale is a note sequence represented as T-T-S-T-T-T-S while a minor scale is T-S-T-T-S-T-T, from low to high. The major and minor scales are the two most commonly seen diatonic scale. These seven notes functions as the tonic, supertonic, median, subdominant, dominant, submediant, and leading tone, respectively (see Figure 1).
- If the tonic (i.e., the first note) of a major scale is C, we then call it a C major scale. If the tonic of a minor scale is C, we then call it a C minor scale. There are 24 different keys in Western classical music (i.e., two modes time 12 tonic notes).
- A major scale and a minor scale that have the same tonic are called *parallel keys*. For example, the parallel minor of a C major key is a C minor key. A major scale and a minor scale that have the same key signatures (i.e., the same set of notes) are called *relative keys*. For example, the relative minor key of the C major key is the A minor key, while the relative minor of E major is C# minor.
- Do not confuse the major/minor key with the major/minor chord. A chord is the co-occurrence of (usually 3) notes, like the major triad and the minor triad, while a key represents the structural information in a diatonic scale.

(a) (25%) **The Krumhansl-Schmuckler key-finding algorithm.** We consider the Krumhansl-Schmuckler (K-S) profile as the template for key detection. The K-S profiles of the major and minor scales, as shown in Table 1, are empirical values of human perceptual saliency of each note in the scale. The K-S experiment was done by playing a set of context tones or chords, then playing a probe tone,

major	degree	1		2		3	4		5		6		7
	K-S	6.35	2.23	3.48	2.33	4.38	4.09	2.52	5.19	2.39	3.66	2.29	2.88
minor	degree	1		2	3		4		5		6		7
	K-S	6.33	2.68	3.52	5.38	2.60	3.53	2.54	4.75	3.98	2.69	3.34	3.17

Table 1: The scale degrees and the K-S templates of major and minor scales.

and asking a listener to rate how well the probe tone fit with the context (see Carol L. Krumhansl, *Cognitive foundations of musical pitch*, Oxford University Press, 2001.).

In this assignment, we consider using the cosine similarity between the audio chroma features and the K-S profile for key detection. Denote the zeroth index of the K-S template as the C note. The K-S template values of the C major key (denoted as $\mathbf{t}_{\mathrm{M}}$) and C minor key (denoted as $\mathbf{t}_{\mathrm{m}}$) are then the values shown in Table 1. The K-S template of C# major is then `np.roll`$(\mathbf{t}_{\mathrm{M}}, 1)$, a circular shift of $\mathbf{t}$ by one, where `np.roll` is a `numpy` function performing circular shift. By circular shifting $\mathbf{t}_{\mathrm{M}}$ and $\mathbf{t}_{\mathrm{m}}$ from 0 to 11 we then have the templates of all the 24 keys. The key of a music recording is then determined by the maximal cosine similarity between the chroma feature of the recording and the 24 templates. You may use `scipy.stats.pearsonr` in `scipy` to find the correlation coefficients. For more information of this key finding algorithm, see http://rnhart.net/articles/key-finding/.

In global key detection, the algorithm should output one estimated key for each piece of music. You may either take average pooling of the chromagram over time and then compute the correlation coefficients (this is called *early fusion*), or compute the correlation coefficients first then summarize the estimated keys into one results by voting (this is called *late fusion*).

Sketch of the K-S key finding algorithm

i. Generate the K-S templates $\mathbf{t}^{(k)} \in \mathbb{R}_{+}^{12}$ of the 24 keys, $k \in \mathcal{K}$, where $\mathcal{K}$ denotes the set of keys, namely {C:maj, C#:maj, D:maj, $\cdots$, B:maj, C:min, C#:min, $\cdots$, B:min}.

ii. Compute the chroma feature $\mathbf{z} \in \mathbb{R}_{+}^{12}$ of the input audio. $\mathbf{z}$ can be the temporal average of the chromagram (early fusion) or frame-level chroma (late fusion).

iii. The estimated key $\hat{k}$ is then

$$\hat{k} := \operatorname*{argmin}_{k \in \mathcal{K}} \operatorname{corr}(\mathbf{t}^{(k)}, \mathbf{z}) \,. \tag{5}$$

iv. Late fusion or post-processing (optional).

To evaluate the performance of global key estimation, the raw accuracy (RA) is simply

$$\mathrm{RA}_{global} = \frac{\text{\# correct detection}}{\text{\# all music pieces in the dataset}} \tag{6}$$

The raw accuracy is however unable to resolve the ambiguity in key perception. For example, the C major key is easily to be detected as G major key (a perfect-fifth error), A minor key (a relative-major/minor error), or C minor key (a parallel-major/minor error), because these erroneous keys are intrinsically "close" to C major keys. To address this, we also consider the weighted accuracy (WA), which gives relative weights to the results having relation to the ground truth key:

$$\mathrm{WA}_{global} := \frac{\text{\# same+0.5(\# fifth)+0.3(\# relative)+0.2 (\# parallel)}}{\text{\# all music pieces in the dataset}} \tag{7}$$

You can directly use the evaluation function `mir_eval.key.evaluate` in the `mir_eval` library.

Please implement the K-S key finding algorithm and evaluate the RA and WA on the SC06 audio recordings in the Schubert Winterreise Dataset (SWD).[3] SWD is a multimodal dataset comprising various representations and annotations of Franz Schubert's song cycle *Winterreise*. *Winterreise* is Schubert's seminal work, which constitutes an outstanding example of the Romantic song cycle—a central genre within Western classical music. Please compare the performance of global key detection (i.e., RA and WA) using the following three chroma features: 1) STFT-based chromagram, 2) CQT chromagram, and 3) CENS chromagram. They are `librosa.feature.chroma_stft`, `librosa.feature.chroma_cqt` and `librosa.feature.chroma_cens` in `librosa`. You may take either the early fusion and late fusion approaches. Discuss your experiment results.

(b) (20%) **Local key detection.** In SWD, *key modulation* can be found in some pieces. That means, the key of a music piece may change over time. Please then design a local key detector that outputs the key of the music every 0.1 second. That means, there is a key detection output for every 0.1 second. In the early fusion approadch, to get the local key feature, you may consider the average-pooled chroma of a segment (maybe 30 seconds or so), not of the whole music piece. For example, the feature representing the local key at the 60th second can be obtained by summing up the chorma vectors from the 45th to the 75th second. You may also try different segment sizes. Late fusion is also worth trying.

Perform your method on the SC06 audio in SWD and evaluate your results against the local keys given by annotator 1. The raw accuracy and weighted accuracy of local key finding can be defined by counting the number of time instances rather than the number of pieces.

$$\text{RA}_{local} = \frac{\text{\# correct time frames}}{\text{\# time frames (detections) in all music pieces}} \tag{8}$$

$$\text{WA}_{local} := \frac{\text{\# same+0.5(\# fifth)+0.3(\# relative)+0.2 (\# parallel)}}{\text{\# time instances (detections) in all music pieces}} \tag{9}$$

(c) (10%) The local key detection problem can be regarded as a segmentation problem. There has been evaluation metrics for the segmentation performance in the chord recognition problem, but such metrics have not been applied in local key detection. Please apply the over-segmentation, under-segmentation and average segmentation measures (please refer to the directional Hamming divergence in Lecture 3 slides) on the local key detection of SWD. (Hint: these metrics have been implemented somewhere in `mir_eval.chord`). Under which cases does your algorithm over-segment or under-segment the audio? How to improve it?

(d) (Bonus) Propose any better global and local key detection algorithms here. Show that they are better than the above ones.

Please submit your .zip file containing the report (.pdf) and your code (.py or .ipynb), with the file name "HW1_[your ID]" to the course website. Please note that, as a report, a detailed discussion on your result is encouraged. The deadline of Assignment #1 is April 9.

[3] Dataset and annotation available at: https://zenodo.org/record/4122060#.YituDHpBy5e