Analysis of Tanpura Tunings

Report by: Vivek Grover, Guide: Prof. Preeti Rao

***Tanpura* is an integral part of Indian classical music since its invention. There are various versions of *Tanpura* available now e.g. 4-stringed *Tanpura*, 5-stringed *Tanpura* etc. It has droning effect through which the singer can get the reference of the middle *Sa* i.e. the tonic note, for singing. In this report, the aim is to find the tuning of the 4-stringed *Tanpura* and also identifying the key of the tuning using basic music processing techniques in *Python*.**

Introduction

Tanpura is a multi-stringed instrument. It is widely used in the Indian classical as well as Carnatic music for performances to provide harmonic background over which the *raga* melodies evolve. The *Ragas* can be divide into four categories based on which type of *Tanpura* tuning is used [1]:

1. Those which omit *Madhyama*, the Perfect fourth, e.g. Bhupali, Shankara, Desakar, use *SaPa* tuning.
2. Those which used both *Panchama* and *Madhyama*. Here are two types:
   (a) Those dominated by *Panchama*, the Perfect fifth, e.g. Bhairava, Yaman, use *SaPa* tuning.
   (b) Those dominated by *Madhyama*, the Perfect fourth, e.g. Bageshri, Lalit Pancham. use *SaMa* tuning.
3. Those which omit *Panchama*, the Perfect fifth, e.g. Malkauns, Lalit. tanpura is tuned same as in 2b.
4. For many evening *ragas* such as Marva, Poorvi, Puriya, *SaNi* tuning is used.

For this report, we will only talk about the 4-stringed *Tanpura* in which strings are tuned in the following manner:

Tuning name	String 1	String 2	String 3	String 4
SaPa	Pa0	Sa1	Sa1	Sa0
SaMa	Ma0	Sa1	Sa1	Sa0
SaNi	Ni0	Sa1	Sa1	Sa0

The above name convention is taken from the research paper "Beyond Swayambhu Gandhar: An analysis of perceived tanpura notes."[2] by Pandya and Paritosh K. . The '0' above indicates the lower octave, i.e. *Mandra saptak* and '1' indicates the middle octave, i.e. *Madhya saptak* This is also used to name the audio files in the dataset. In this report, we undertake an analysis of these different tuning using *Chromagram* and *band-pass filtering*. We answer the following questions:

1. What is the root of the *Tanpura* in the recording according to the western music notation?
2. What is the corresponding significant note that gives a tuning its particular name?
3. Is the method we use applicable for all the *Tanpura* recordings?

Dataset. We have collected around 108 recordings of the 15 seconds duration from three different mobile *Tanpura* applications, *TanpuraDroid*, *Bandish* and *Singtico*. Some other recordings from *YouTube* and *osf* database was also used to check the validity of the experiments for finding the tuning of *Tanpura* in those recordings. Recordings from *osf* database were full stage performance so the parts containing only *Tanpura* were cut and used in this analysis. These recordings dont have the same tempo between each of them. The dataset is available at https://drive.google.com/drive/folders/1bZzNezpU-IfDip5Ne_xB6nvHxXAVlemp?usp=sharing.

Core Concepts Used

Discrete Fourier transform[3]

Let $x \in \mathbb{C}^N$ be a vector of length $N \in \mathbb{N}$. In the music signal context, x can be interpreted as a discrete signal with samples $x(0), x(1), ..., x(N-1)$ after sampling according to a certain sampling rate. Note that we start indexing with the index 0. The *discrete Fourier transform* (DFT) of x is defined by:

$$X(k) := \sum_{n=0}^{N-1} x(n) \exp(-2\pi i k n/N) \quad \textbf{(1)}$$

for $k \in [0 : N-1]$. The vector $X \in \mathbb{C}^N$ can be interpreted as frequency representation of the time-domain signal x. To obtain a geometric interpretation of the DFT, we define the vector $\mathbf{u}_k \in \mathbb{C}^N$ by

$$\mathbf{u}_k(n) := \exp(2\pi i k n/N) = \cos(2\pi k n/N) + i \sin(2\pi k n/N) \quad \textbf{(2)}$$

for each $k \in [0 : N-1]$. This vector can be regarded as a sampled version of the exponential function of frequency k/N. Then, the DFT can be expressed as inner products

$$X(k) := \sum_{n=0}^{N-1} x(n)\overline{\mathbf{u}_k} = \langle x | \mathbf{u}_k \rangle \quad \textbf{(3)}$$

of the signal x and the sampled exponential functions $\mathbf{u}_k$. The absolute value $|X(k)|$ indicates the degree of similarity between the signal x and $\mathbf{u}_k$.

In the case that $x \in \mathbb{R}^N$ is a real-valued vector (which is always the case for our music signal scenario), we obtain:

$$X(k) := \langle x|\mathrm{Re}(\mathbf{u}_k)\rangle - i\langle x|\mathrm{Im}(\mathbf{u}_k)\rangle$$

There is fast way of calculating the *Discrete Fourier Transform* called *Fast Fourier transform* which utilizes the the observation that applying a DFT of even size $N = 2M$ can be expressed in terms of applying two DFTs of half the size M. Details of this is given in FMP book[3].

Short-Time Fourier Transform(STFT)[3]

DFT gives us the frequency analysis for the all the sample, i.e. the whole audio. There is no temporal aspects related to it. To analyze the audio along the temporal domain, we use *STFT*.

The idea is like this: let $x : [0 : L-1] := \{0,1,\ldots,L-1\} \to \mathbb{R}$ be a real-valued discrete-time (DT) signal of length L obtained by equidistant sampling with respect to a fixed sampling rate F_s given in Hertz. Furthermore, let $w : [0 : N-1] \to \mathbb{R}$ be a sampled **window function** of length $N \in \mathbb{N}$. For example, in the case of a rectangular window one has $w(n) = 1$ for $n \in [0 : N-1]$. The length parameter N determines the duration of the considered sections, which amounts to N/F_s seconds. One also introduces an additional parameter $H \in \mathbb{N}$, which is referred to as the **hop size**. The hop size parameter is specified in samples and determines the step size in which the window is to be shifted across the signal. With regard to these parameters, the **discrete STFT** $\mathcal{X}$ of the signal x is given by

$$\mathcal{X}(m,k) := \sum_{n=0}^{N-1} x(n+mH)w(n)\exp(-2\pi ikn/N) \qquad \textbf{(4)}$$

with $m \in [0 : M]$ and $k \in [0 : K]$. The number $M := \lfloor \frac{L-N}{H} \rfloor$ is the maximal frame index such that the window's time range is fully contained in the signal's time range. Furthermore, $K = N/2$ (assuming that N is even) is the frequency index corresponding to the Nyquist frequency. The complex number $\mathcal{X}(m,k)$ denotes the k^{th} Fourier coefficient for the m^{th} time frame.

Note that for each fixed time frame m, one obtains a **spectral vector** of size $K+1$ given by the coefficients $\mathcal{X}(m,k)$ for $k \in [0 : K]$. The computation of each such spectral vector amounts to a DFT of size N, which can be done efficiently using the FFT which is also explained in the FMP book[3].

Spectrogram[3]

The **spectrogram** is a two-dimensional representation of the squared magnitude of the STFT:

$$\mathcal{Y}(m,k) := |\mathcal{X}(m,k)|^2. \qquad \textbf{(5)}$$

It can be visualized by means of a two-dimensional image, where the horizontal axis represents time and the vertical axis represents frequency. For an audio, the **spectrogram** value $\mathcal{Y}(m,k)$ is represented by the intensity or color in the image at the coordinate (m,k). Note that in the discrete case, the time axis is indexed by the frame indices m and the frequency axis is indexed by the frequency indices k.

In the temporal dimension, each Fourier coefficient $\mathcal{X}(m,k)$ is associated with the time position, T_coef:

$$T_\mathrm{coef}(m) := \frac{m \cdot H}{F_\mathrm{s}}$$

measured in seconds. For a hop size $H = 1$, we get $T_\mathrm{coef}(m) = \frac{m}{F_\mathrm{s}} = m \cdot T$ seconds, resulting in a spectral vector for each sample of the DT-signal x, which significantly increases data volume. To reduce redundancy, the hop size H is often related to the window length N, commonly choosing $H = N/2$, balancing temporal resolution and data volume.

In the frequency dimension, the index k of $\mathcal{X}(m,k)$ corresponds to the frequency, F_coef:

$$F_\mathrm{coef}(k) := \frac{k \cdot F_\mathrm{s}}{N}$$

in Hertz.

Chromagram[3]

Humans have logarithmic perception of frequency whicha is utilised by pooling the frequencies of the **spectral vectors** to represent them using *equal-tempered* scale. The basic idea is to assign each spectral coefficient $\mathcal{X}(n,k)$ to the pitch with a center frequency that is closest to the frequency $F_\mathrm{coef}(k)$. More precisely, we define for each pitch $p \in [0 : 127]$ the set:

$$P(p) := \{k : F_\mathrm{pitch}(p-0.5) \leq F_\mathrm{coef}(k) < F_\mathrm{pitch}(p+0.5)\}. \qquad \textbf{(6)}$$

The frequency range covered by the set $P(p)$ depends on the frequency in a **logarithmic** fashion. We define the **bandwidth** $\mathrm{BW}(p)$ of pitch p by :

$$\mathrm{BW}(p) := F_\mathrm{pitch}(p+0.5) - F_\mathrm{pitch}(p-0.5). \qquad \textbf{(7)}$$

Now we have the pitches in 128 bins. We are only one step away from defining what a *chromagram* is. The equal-temperament tuning has 12 distinct pitches and they just keep on repeating in higher octaves. For example, all the C in different octaves are perceived to ahve the same color or **chroma**. Using that notion, we can define 12 pitch classes such as:

$$\{\mathrm{C}, \mathrm{C}^\sharp, \mathrm{D}, \mathrm{D}^\sharp, \mathrm{E}, \mathrm{F}, \mathrm{F}^\sharp, \mathrm{G}, \mathrm{G}^\sharp.\mathrm{A}, \mathrm{A}^\sharp, \mathrm{B}\}$$

Enumerating the chroma values, we identify this set with $[0 : 11]$ where 0 refers to chroma C, 1 to $\mathrm{C}^\sharp$, and so on. The main idea of **chroma features** is to aggregate all spectral information that relates to a given pitch class into a single coefficient. Given a pitch-based log-frequency spectrogram $\mathcal{Y}_\mathrm{LF} : \mathbb{Z} \times [0 : 127] \to \mathbb{R}_{\geq 0}$, a **chroma representation** or **chromagram** $\mathbb{Z} \times [0 : 11] \to \mathbb{R}_{\geq 0}$ can be derived by summing up all pitch coefficients that belong to the same chroma:

$$\mathcal{C}(n,c) := \sum_{\{p \in [0:127] \,:\, p \bmod 12 = c\}} \mathcal{Y}_\mathrm{LF}(n,p) \qquad \textbf{(8)}$$

for $c \in [0 : 11]$.

Experiments & Results

For all the experimental analysis done in this report, *Sampling rate* is 22050 Hz, window length/ *frame size* is 2048 (approx. 92.88 ms) and hop size/ *hop length* is half of frame size i.e. 1024 samples or 46.44 ms. For all the **STFT** implementation, **Hann window**[4] function is used. The recordings from different applications are named in the following manner: tuning_key_app-name.mp3 e.g. a recording of *SaPa* tuning in the key of *C* from the app *TanpuraDroid* is named as 'SaPa-C-TD.mp3' whereas from *Bandish* it will be 'SaPa-C-Bd.mp3' and from *Singtico* it will be 'SaPa-C-St.mp3'.

Firstly, we will look at the **Spectrogram** of one of the Tanpura recordings, e.g. 'SaPa-C-TD.mp3' and talk about the features that cam be extracted. The **spectrogram** for the duration of 9 seconds is shown below:

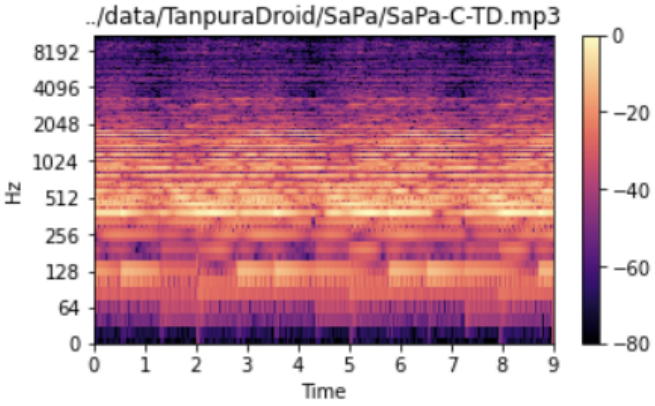

Fig. 1. Spectrogram

The frequency axis is in log scale and the time axis is in seconds scale. We can clearly see the pattern repeating between 2 to 5 seconds and 5 to 8 seconds vaguely. Most of the energy or the enrergy of the harmonic content is present in the frequency band 256-1024 Hz. Here comes the **chromgram** for the above example:

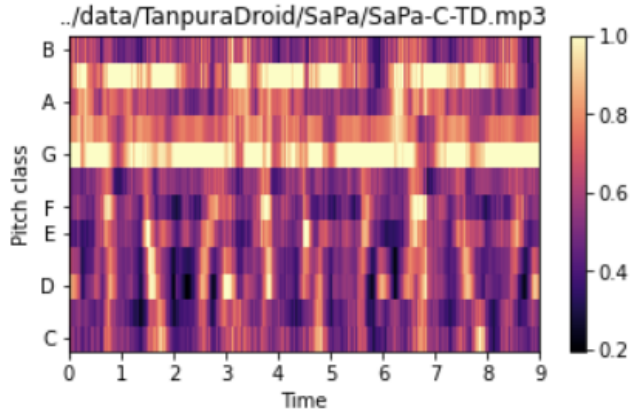

Fig. 2. Chromagram

Here the y-axis becomes the *pitch class* axis and x-axis is still the time axis.We can see that after the pooling of frequencies only the perfect fifth or *panchama* i.e. G, is most noticeable along with some other pitch classes. The root or the *key* is not even significant in the **chromagram**. To get the features for finding the the tuning and key of the *Tanpura* following data processing is done:

1. Band passing the the audio from 60 to 200 Hz.
2. Getting the **Spectrogram** and **Chromagram** of the filtered audio signal.
3. **Median filtering** the chromagram matrix.
4. Putting a **global threshold** on the chromagram matrix.
5. Summing up the **energies** in all the **pitch classes**.

First, let's know thw **median filtering**. The idea of median filtering is to replace each entry of a sequence with the **median** of neighboring entries. The size of the neighborhood is determined by a parameter $L \in \mathbb{N}$, which is the **length** of the median filter applied. Given a sequence $X = (x_1, x_2, ..., x_N)$ of feature vectors $x_n \in \mathbb{R}^K$, we apply median filtering for each dimension for each $k \in [1:K]$. Applying the median filter helps us in **feature smoothing** and preserving sharp transitions.

Second is the **global threshold**. In the end **chromagram** is just a matrix in $R^{Z \times [0:11]}$. We put a threshold at some $p \times max(chromagram)$, where $p \in (0,1)$ and seeting all the components to zero to get the final result. For all the analysis, Tanpura recordings in the key of 'C' will be used for all the tunings for all apps.

SaPa tuning. Below are the visual plots after applying the above mentioned data processing techniques:
From **TanpuraDroid:**

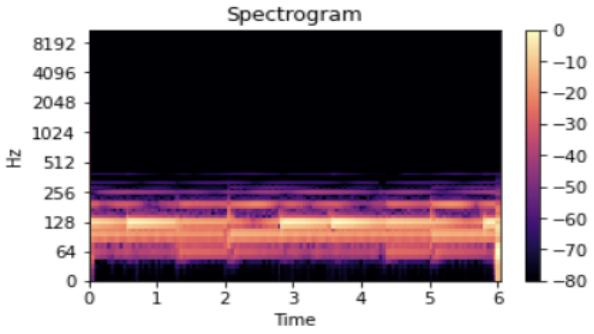

Fig. 3. Spectrogram

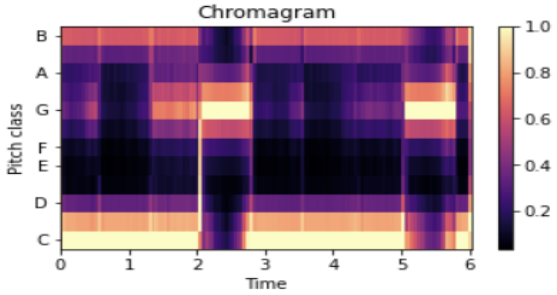

Fig. 4. Chromagram

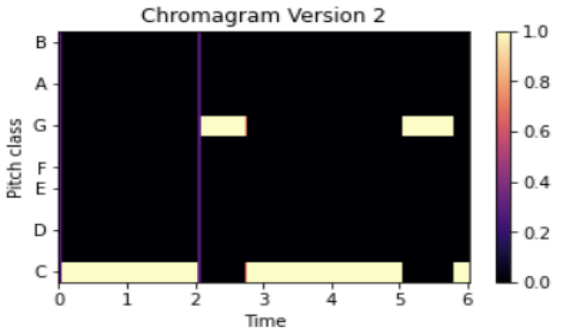

Fig. 5. Chromagram after median filtering and thresholding

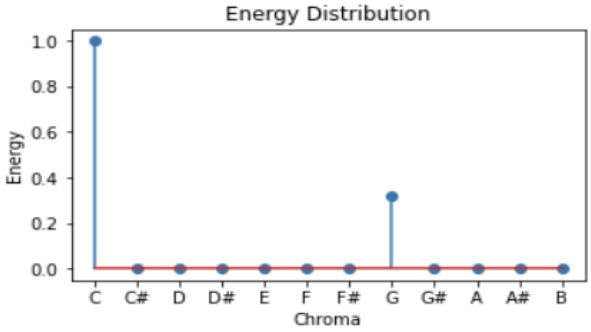

Fig. 6. Energy Distribution

Note: For the other cases only the Chromagram after median filtering & thresholding and energy distribution plot will be shown.

From **Bandish**:

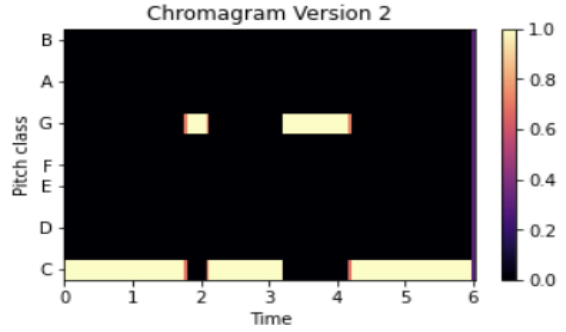

Fig. 7. Chromagram after median filtering and thresholding

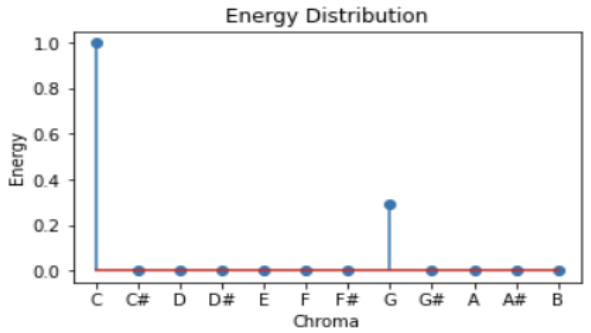

Fig. 8. Energy Distribution

From **Singtico**:

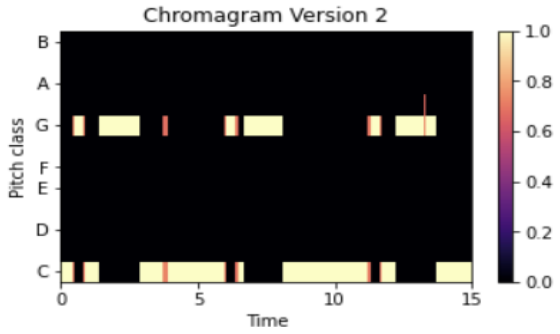

Fig. 9. Chromagram after median filtering and thresholding

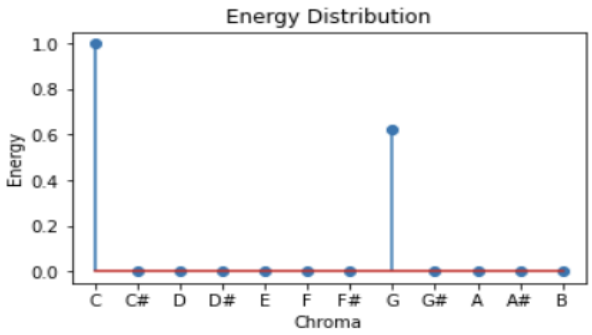

Fig. 10. Energy Distribution

We can see that **chromagram** and **energy distribution** plots give us the tuning and the key.

From chromagram, the brightest pitch class shown are the root i.e. *Sa* and the perfect fifth i.e. the *panchama* with root having the large portion in the duration of 6 seconds. It becomes more clear when the energies for all the pitch classes are summed(also normalised) and displayed in the form of the stem plot. The pitch class with max energy is the root and the second highest energy is the perfect fifth.

The amount of the energies in different pitch classes across different apps is not because they may have used different *Tanpura* samples to make their apps.

SaMa tuning. From **TanpuraDroid**:

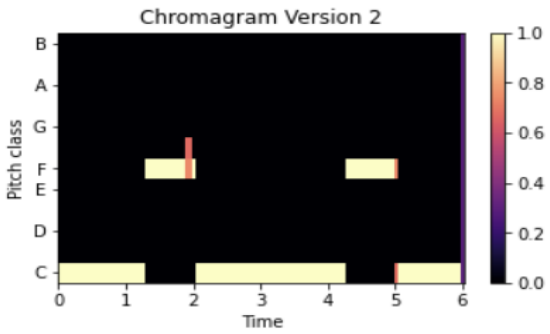

Fig. 11. Chromagram after median filtering and thresholding

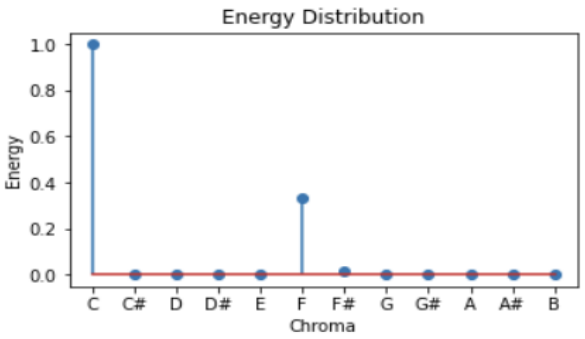

Fig. 12. Energy Distribution

From **Bandish**:

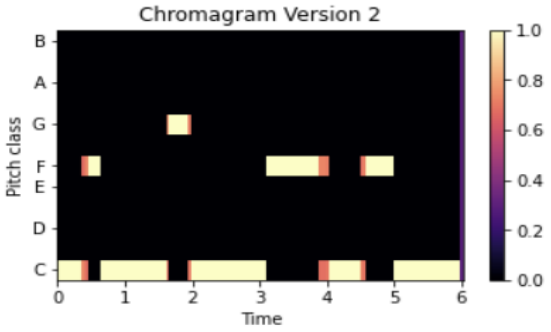

Fig. 13. Chromagram after median filtering and thresholding

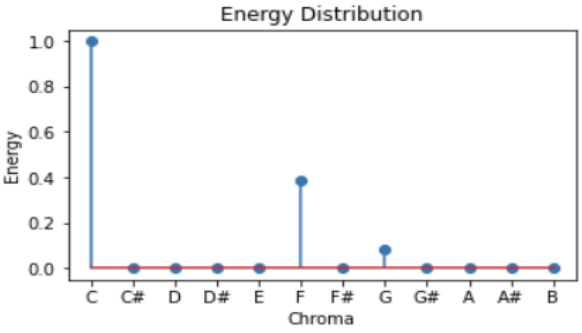

Fig. 14. Energy Distribution

From **Singtico**:

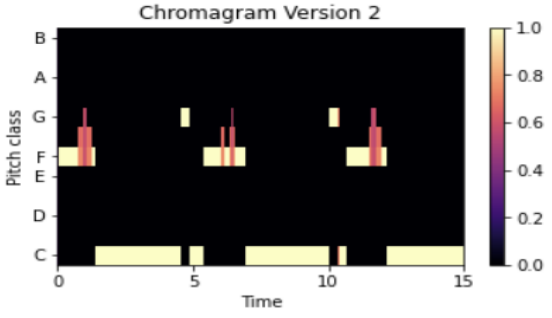

Fig. 15. Chromagram after median filtering and thresholding

Here in the case of *SaMa*, the results are similar to the *SaPa* tuning case that is we are able to get the tuning from the energy distribution plot as the pitch class F is the perfect fourth i.e. *madhyama* a of the root C i.e. *Sa*. We can see that there is

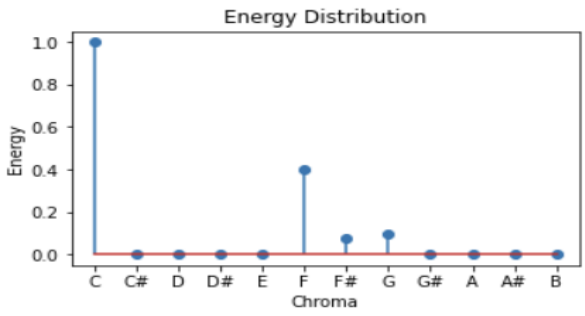

Fig. 16. Energy Distribution

frequency smearing in the pitch classes with -1 and +1 semitone range.

SaNi tuning. From **TanpuraDroid**:

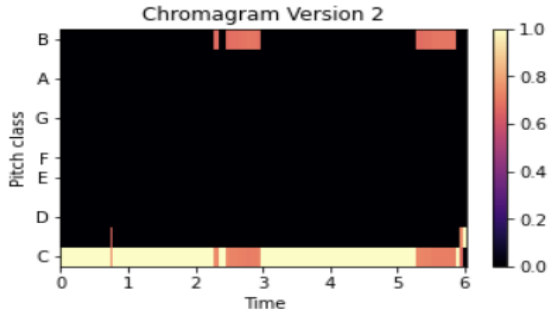

Fig. 17. Chromagram after median filtering and thresholding

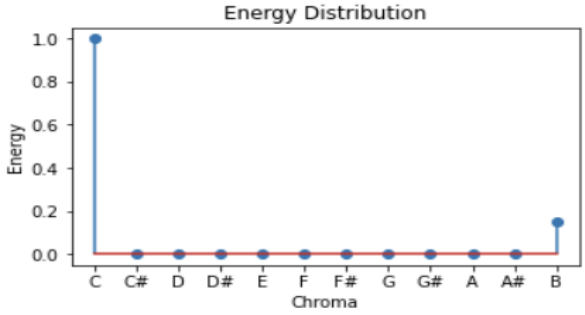

Fig. 18. Energy Distribution

From **Bandish**:

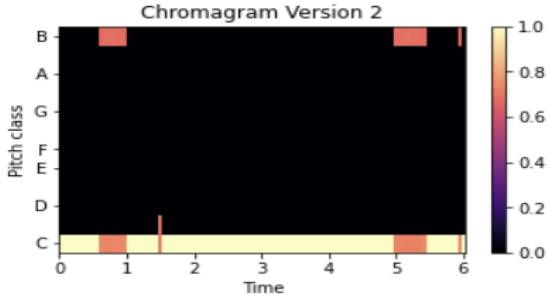

Fig. 19. Chromagram after median filtering and thresholding

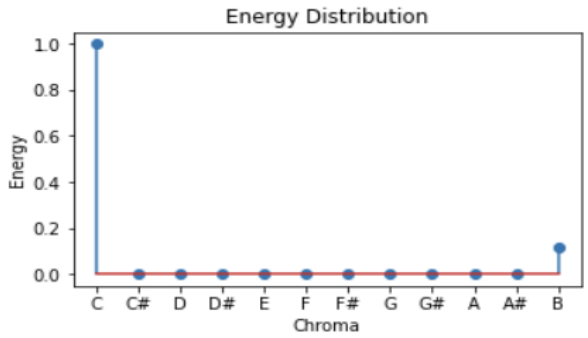

Fig. 20. Energy Distribution

From **Singtico**:

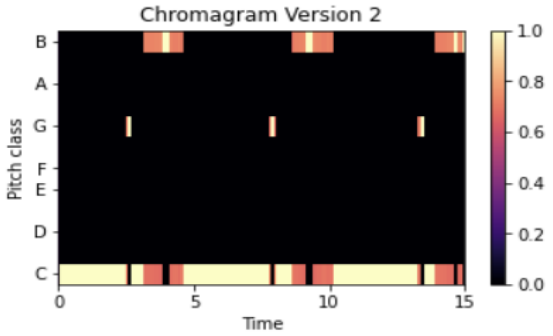

Fig. 21. Chromagram after median filtering and thresholding

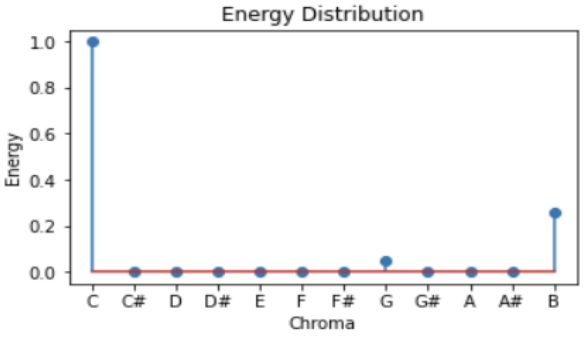

Fig. 22. Energy Distribution

We can clearly see that root i.e. *Sa* and the major seventh i.e. *Nishad/Ni* from the energy distribution plot. We can conclude that this method of determining the tuning of the *Tanpura* works for all the tuning roughly.

After applying this method to all the keys of each tuning, there were some cases where this method seemed to fail generally for the keys of **E, F♯, G** and **G♯** for *SaPa* & *SaMa* tunings mostly. his method still works but the parameters such as **frequency range** for band passing the audio, **global-threshold** and window for **median filtering**.

For all the audios, the parameters were kept the same in the the frequency range for band-passing was 60 to 200 Hz(already mentioned), global threshold i.e. p, was 0.9 and window for median filtering was 3. the following are the energy distribution plots for E and F♯ for **TanpuraDroid** for *SaPa*:

Only the root is coming out. Tuning cannot be determined from this alone.

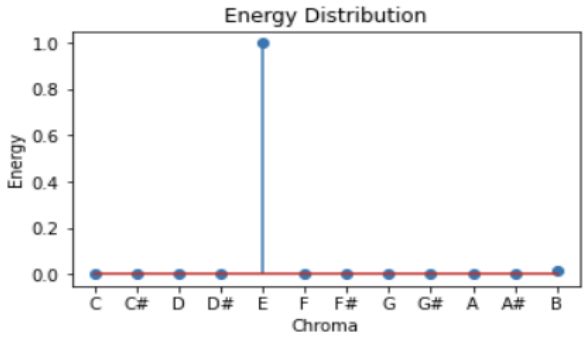

Fig. 23. Energy Distribution for the Key of E

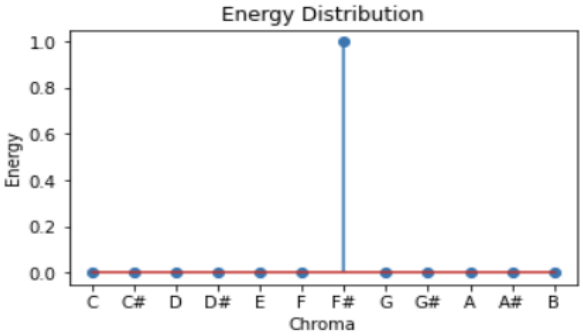

Fig. 24. Energy Distribution for the Key of F♯

See for **Bandish** also but for *SaMa* tuning in the keys of F♯ and G♯:

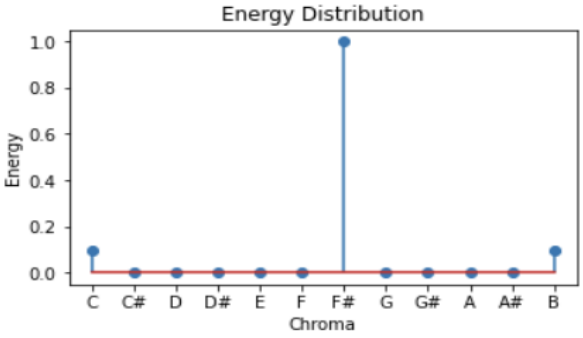

Fig. 25. Energy Distribution for the Key of F♯

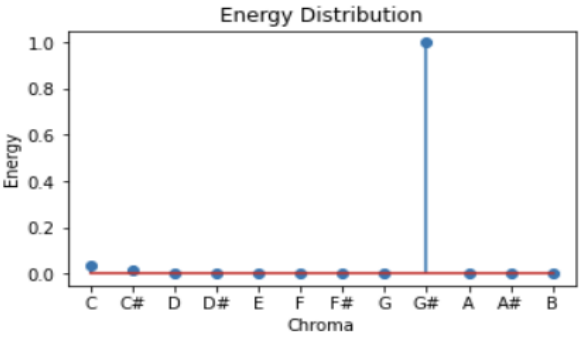

Fig. 26. Energy Distribution for the Key of G♯

For making this method work for these exceptional cases, increasing the frequency range from 60-200 Hz to 50-300Hz seems to work keeping the other parameters same. Below are the results for *SaPa* tuning recording in the key of E(from **TanpuraDroid**) and *SaMa* tuning recording in the key of F♯(from **Bandish**):

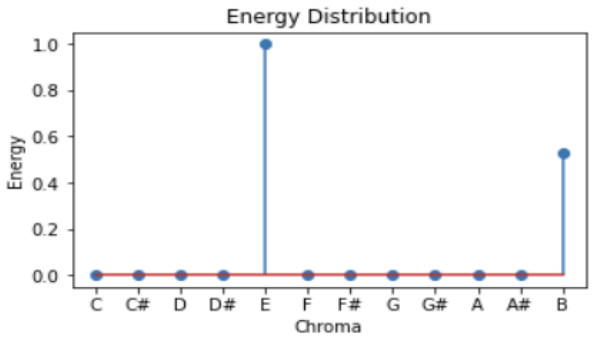

Fig. 27. Energy Distribution for the Key of E (*SaPa* tuning)

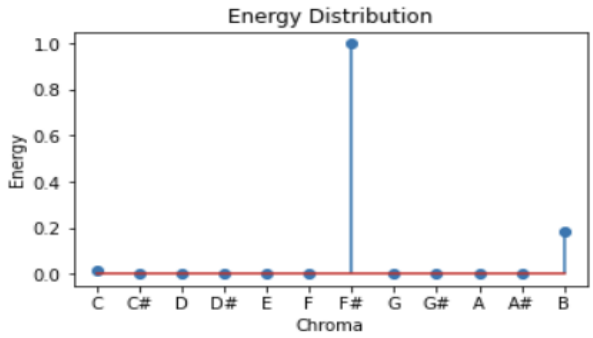

Fig. 28. Energy Distribution for the Key of F♯(*SaMa* tuning)

So for the keys of **E, F♯, G** and **G♯**, we can just increase the frequency range of band-pass filter a to get the tuning and key from the recording of *Tanpura* alone.

This method was applied on some *Tanpura* recordings from *YouTube* too. **Three** recordings for each tuning in the key of C, C♯ and A♯. There results were also quite promising. Following are the energy distribution plots according to different tunings:

SaPa tuning

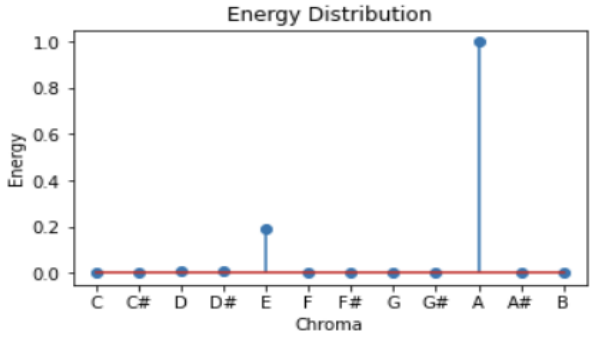

Fig. 29. Energy Distribution for the Key of A

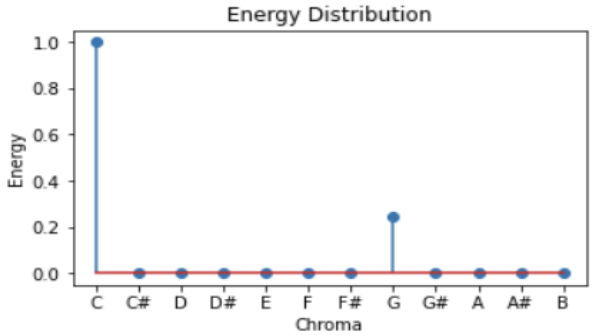

Fig. 30. Energy Distribution for the Key of C

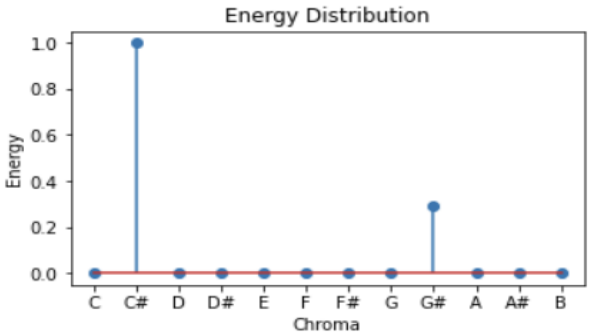

Fig. 31. Energy Distribution for the Key of C♯

SaMa tuning

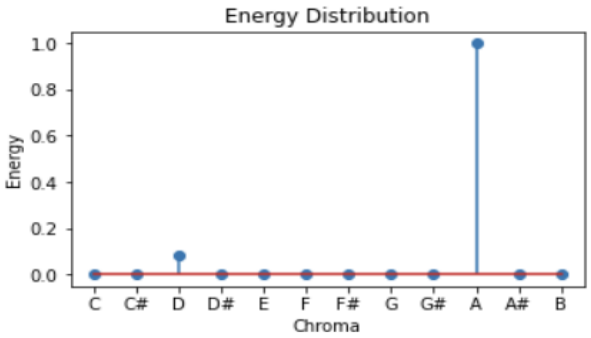

Fig. 32. Energy Distribution for the Key of E

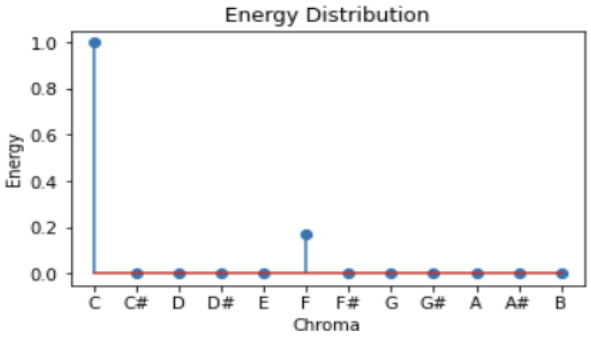

Fig. 33. Energy Distribution for the Key of F

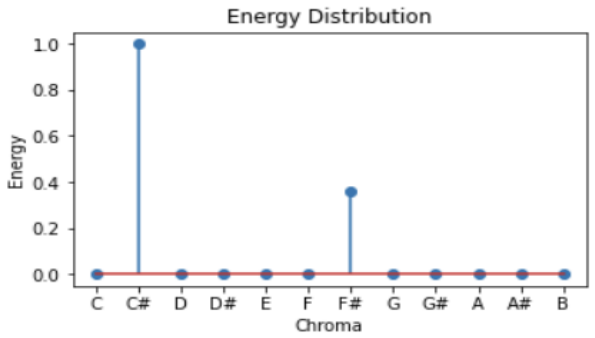

Fig. 34. Energy Distribution for the Key of C♯

SaNi tuning

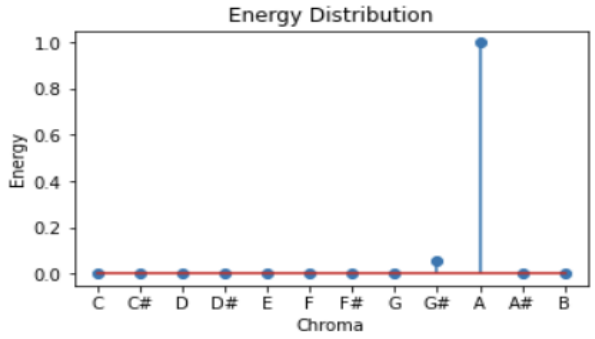

Fig. 35. Energy Distribution for the Key of A

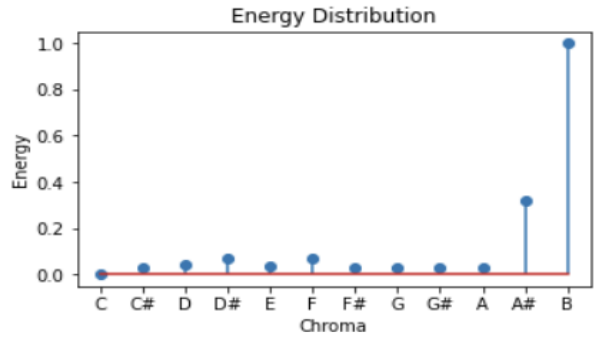

Fig. 36. Energy Distribution for the Key of C

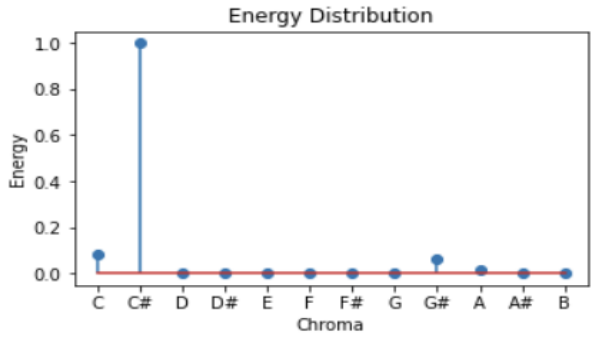

Fig. 37. Energy Distribution for the Key of C♯

Access the dataset for *YouTube* recordings from https://drive.google.com/drive/folders/1bZzNezpU-IfDip5Ne_xB6nvHxXAVlemp?usp=sharing

There are recordings of live performances of various *ragas* on https://osf.io/. Some of them are:

1. MAK_TilakK | Manjiri Asanare Kelkar | Tilak Kamod | 20:41 | Harmonium Accompaniment by Jignesh | [MAK_TilakK_Stereomix.wav](https://osf.io/n5qkc)
2. VS_Bhoop | Veena Sahasrabuddhe | Bhoop | 57:25 | Harmonium Accompaniment by Jignesh | [VS_Bhoop_Stereomix.wav](https://osf.io/9ags7)
3. VK_Multani | Vijay Koparkar | Multani | 49:28 | Harmonium Accompaniment by Jignesh | [VK]_Multani_Stereomix.wav](https://osf.io/k45q2)
4. MAK_Jaun | Manjiri Asanare Kelkar | Jaunpuri | 30:30 | Harmonium solo closed-mic recording from OSF | [MAK_Jaun_Stereomix.wav](https://osf.io/prjq4)
5. MAK_Jhin | Manjiri Asanare Kelkar | Jhinjhoti | 27:46 | Harmonium solo closed-mic recording from OSF | [MAK_Jhin_Stereomix.wav](https://osf.io/dxv76)

Only *Tanpura* parts were cut from these recordings using the **Reaper** DAW(Digital Audio Workstation). all the recordings had singers, harmonium, tabla and *Tanpura*. Some recordings had harmonium and *Tanpura* for the whole performance(**VK_Multani**) while one of these recordings had vocals all the time(**VS_Bhoop**).

Note: There is no prior knowledge of *Tanpura* tuning of these recordings. Comparison with the *Tanpura* recordings in the dataset from different apps reveals correct that the results are indeed actually correct.

For all the audio processing here, the parameters are same as the ones that were used in the previous experiments except for **VS_Multani**. For this recording, the frequency range of band-passing is 60-600 Hz while it is 60-300 Hz for others.

Chromagram and energy distribution plots for **MAK_TilakK**

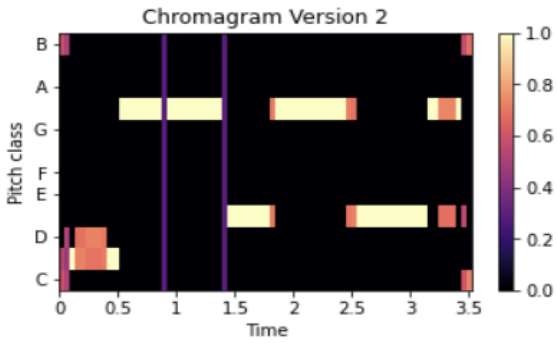

Fig. 38. Chromagram after median filtering and thresholding

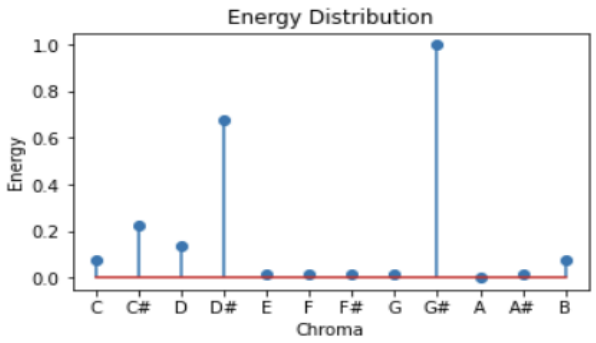

Fig. 39. Energy Distribution

There is frequency smearing into the other **pitch classes** as it is a recording of live performance. From the energy distribution plot, we can clearly say that the tuning of the *Tanpura* is *SaPa* and the key is G♯.

Chromagram and energy distribution plots for **VS_Bhoop**

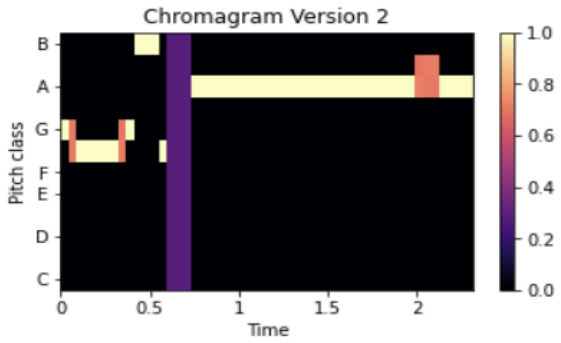

Fig. 40. Chromagram after median filtering and thresholding

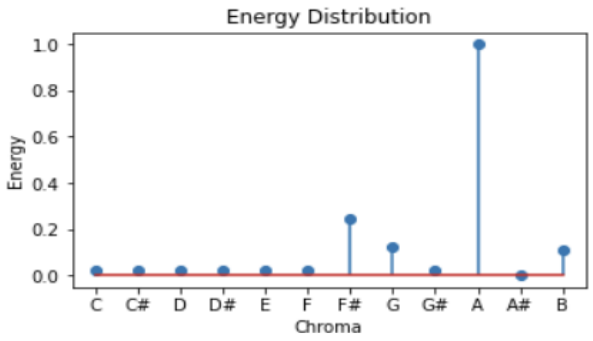

Fig. 41. Energy Distribution

For this recording, the vocals were interfering with the results. That's why we were only able to find the key which is A but not the tuning of the *Tanpura* used in the live performance.

Chromagram and energy distribution plots for **VK_Multani**

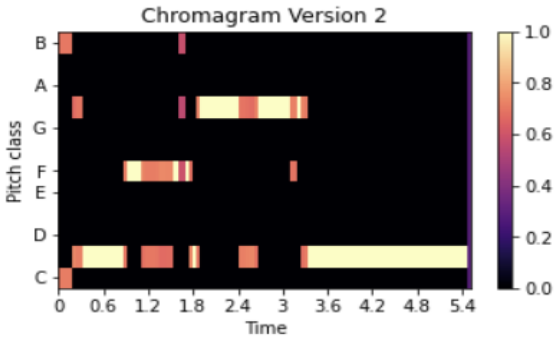

Fig. 42. Chromagram after median filtering and thresholding

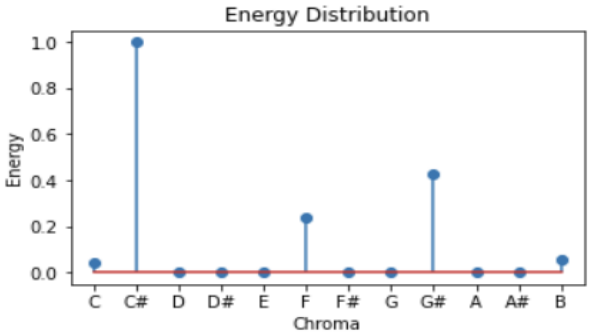

Fig. 43. Energy Distribution

As there was harmonium playing continuously in the background the frequency range for this recording has to be increased to get the tuning of the *Tanpura* which is *SaPa* in the key of C♯.

Chromagram and energy distribution plots for **MAK_Jaun**

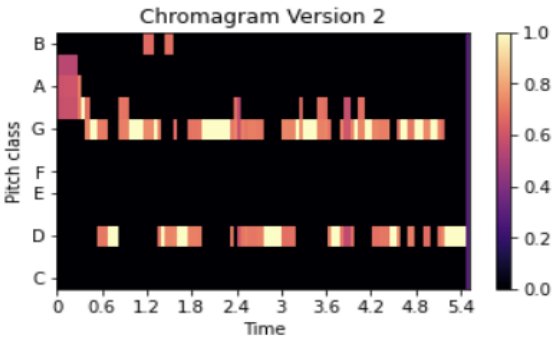

Fig. 44. Chromagram after median filtering and thresholding

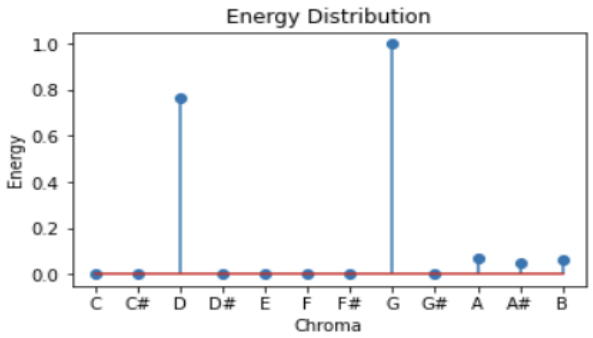

Fig. 45. Energy Distribution

Chromagram and energy distribution plots for **MAK_Jhin**

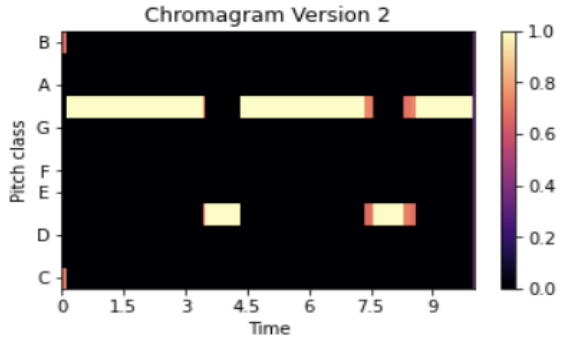

Fig. 46. Chromagram after median filtering and thresholding

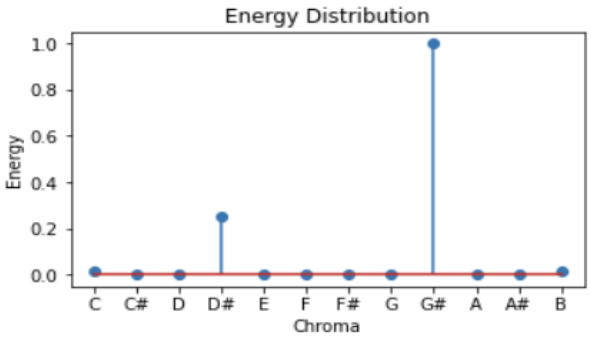

Fig. 47. Energy Distribution

For the last two recordings tuning was easily determined as there was only *Tanpura* in those. This provides the validity for the method used for finding the tuning and the key of *Tanpura* but only when it is alone.

Conclusions & Future Work

We have developed a method for finding the tuning and key of the *Tanpura* using **chromagrams**[3]. This method works for all the tuning with slightly different parameters for some keys i.e. **E, F♯, G** and **G♯**.

This method even works for live recordings of performances where only *Tanpura* is being which indicates the robustness of this technique. For utilising this method on more complex recordings, we can hope to train a **machine learning** model which directly extract the *Tanpura* from the recording and find its tuning and key. As most of today's state of the art **ML/DL** like *Moises.ai*[5], *Fadr*[6] etc., models are only trained on western music dataset, these models don't work well for Indian music which can be an interesting problem to work on in the future.

Bibliography

1. http://www.soundofindia.com/showarticle.asp?in_article_id=816436941
2. Pandya, Paritosh K. "Beyond Swayambhu Gandhar: An analysis of perceived tanpura notes." Journal of the ITC Sangeet Research Academy 19 (2005).
3. Müller, Meinard. Fundamentals of music processing: Using Python and Jupyter notebooks. Vol. 2. Berlin, Germany: Springer, 2021.
4. https://en.wikipedia.org/wiki/Hann_function
5. https://moises.ai/
6. https://fadr.com/