3 3

3

Quick Submit

Quick Submit

College of Engineering, Pune

Document Details

Submission ID	trn:oid:::1:2764649497
Submission Date	Nov 28, 2023, 1:21 PM GMT+5:30
Download Date	Nov 28, 2023, 1:50 PM GMT+5:30
File Name	PIJET_3_Mukta_Takalikar.docx
File Size	711.2 KB

6 Pages

1,665 Words

9,979 Characters

How much of this submission has been generated by AI?

72%

of qualifying text in this submission has been determined to be generated by AI.

Caution: Percentage may not indicate academic misconduct. Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

What does the percentage mean?

The percentage shown in the AI writing detection indicator and in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was generated by AI.

Our testing has found that there is a higher incidence of false positives when the percentage is less than 20. In order to reduce the likelihood of misinterpretation, the AI indicator will display an asterisk for percentages less than 20 to call attention to the fact that the score is less reliable.

However, the final decision on whether any misconduct has occurred rests with the reviewer/instructor. They should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in greater detail according to their school's policies.

How does Turnitin's indicator address false positives?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be AI-generated will be highlighted blue on the submission text.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

What does 'qualifying text' mean?

Sometimes false positives (incorrectly flagging human-written text as AI-generated), can include lists without a lot of structural variation, text that literally repeats itself, or text that has been paraphrased without developing new ideas. If our indicator shows a higher amount of AI writing in such text, we advise you to take that into consideration when looking at the percentage indicated.

In a longer document with a mix of authentic writing and AI generated text, it can be difficult to exactly determine where the AI writing begins and original writing ends, but our model should give you a reliable guide to start conversations with the submitting student.

MUSICAL FREQUENCY NOTE DETECTION

Manish Godbole[1], Kaustubh Joshi[2], Aditya Kadu[3], R.A. Kulkarni[4] & M.S. Takalikar[5]

1 Pune Institute of Computer Technology, Computer Engineering, Pune, Maharashtra, India, manishgodbole02@gmail.com
2 Pune Institute of Computer Technology, Computer Engineering, Pune, Maharashtra, India, knj621311@gmail.com
3 Pune Institute of Computer Technology, Computer Engineering, Pune, Maharashtra, India, adityakadu1203@gmail.com
4 Pune Institute of Computer Technology, Computer Engineering, Pune, Maharashtra, India, rakulkarni@pict.edu
5 Pune Institute of Computer Technology, Computer Engineering, Pune, Maharashtra, India, mstakalikar@pict.edu

Abstract:

Accurate and rapid detection of musical notes is crucial for tasks such as automatic tuning, transcription, and instrument recognition. The proposed detector employs advanced signal processing techniques to analyze audio input and determine the fundamental frequency (pitch) of the predominant musical note being played.

The system utilizes a combination of time-domain and frequency-domain analysis to extract relevant features from the input audio signal. These features are then fed into a machine learning-based classifier that identifies the closest musical note corresponding to the detected frequency. To ensure robustness and accuracy, the system has been trained on a comprehensive dataset covering a wide range of musical instruments, playing styles.

Our work presents the design, implementation, and evaluation of a novel musical frequency note detector aimed at instrumental applications in various musical contexts.

Keywords: Audio input, fundamental frequency, pitch, time-domain, frequency-domain, novel musical frequency note detector.

1 Introduction

Music, a universal language that transcends borders and cultures, has always been a subject of fascination and study. The ability to analyze and understand the nuances of music, such as detecting individual musical notes within a composition, has far-reaching applications in fields ranging from music theory and education to audio processing and digital signal analysis. This research paper delves into the realm of musical frequency note detection, a crucial area of study in the realm of music technology. By exploring the principles and methodologies behind this process, we aim to shed light on the underlying mechanics of musical notes, their frequencies, and how advanced technology can facilitate their precise identification. In doing so, we hope to unlock new avenues for creativity, education, and innovation in the world of music.

2 Literature Survey

As per the work by Jay K. Patela et al. [1] A song basically consists of two things, vocal and background music. The distinctiveness of the voice depends on the singer and in case of background music, it is combination of different musical instruments like piano, guitar, drum, etc. To extracting the attributes of a song becomes more important for various objectives like learning, teaching, and composing. The experiment is done with the quite a few number of piano songs where the notes are already known, and recognised notes are compared with original notes until the detection rate goes higher. And then the experiment is done with piano songs with unknown notes with the proposed algorithm.

The article by John Glover et al. [2] provides a review of some of the most commonly used techniques for real-time onset detection. The author's suggest ways to improve these techniques by incorporating linear prediction as well as presenting a novel algorithm for real-time onset detection using sinusoidal modelling. As well as provides comprehensive results for both the detection accuracy and the computational performance of all of the described techniques, evaluated using Modal

In the research by Allabakash Isak Tamboli et al. [3], the authors developed a musical note recognition method based on an optimization-based neural network (OBNN) within a classification framework. The study involved an extensive review of existing approaches for musical note recognition. The use of OBNN for recognizing musical notes was explored. The document comprehensively analyzes recent investigations related to musical note recognition, summarizing their findings and classifications, with the aim of advancing the effectiveness of this recognition process through diverse methodologies.

The paper by Smith et al. [4] gives seminal work in the field of digital audio processing. This paper delves into the principles and methodologies of physical modeling, which simulates the behavior of real-world musical instruments and sound effects in the digital domain. It explores the mathematical and computational foundations of physical modeling, allowing for the creation of highly realistic virtual instruments and audio effects. By emphasizing the accurate emulation of physical interactions and acoustic phenomena, Smith's research paper has been pivotal in advancing the quality and authenticity of digital music synthesis and audio processing. It remains a foundational reference for researchers and engineers in the field.

The H. Purwins et al. [5] paper gives comprehensive overview of the application of deep learning techniques in the field of audio signal processing. It explores the use of neural networks and deep learning architectures for tasks such as speech recognition, music analysis, and sound synthesis. The paper discusses various deep learning models and their effectiveness in handling complex audio data. It serves as a valuable resource for researchers and practitioners interested in leveraging deep learning for advanced audio processing applications.

The paper by S. A. Shedied et al. [6] presents Critical problem of accurately estimating pitch in speech signals contaminated by noise. The authors propose a novel pitch estimation method tailored for noisy conditions, focusing on the challenging scenario of adverse environmental or recording conditions. Their approach combines adaptive filtering and signal processing techniques to enhance the accuracy and robustness of pitch estimation in the presence of noise. This paper presents an essential contribution to speech signal processing, particularly in contexts where noise interference poses a significant challenge, making it valuable for applications like speech recognition and enhancement.

The paper by S. Wang et al. [7] senses self-supervised learning approach that leverages audio-visual data with spatial alignment to enhance audio representation learning. The proposed method combines visual information and audio signals to train deep neural networks without explicit annotations. By exploiting spatial alignment cues, the model learns robust and informative representations, which have applications in areas such as speech and sound analysis, offering potential benefits for improving the accuracy of audio-based tasks using multi-modal data.

3 Methodology

3.1 Audio Loading

The code begins by loading an audio file ('test.mp3') using the `librosa.load` function, obtaining the raw audio waveform and its sampling rate (SR). This step prepares the data for subsequent analysis.

3.2 Waveform Visualization

It proceeds with visualizing the audio waveform using Matplotlib. This visualization represents the amplitude of the audio signal over time, providing a visual understanding of the audio's characteristics.

3.3 Amplitude Envelope

To analyze the signal's variations, two functions, `amp_env` and `fancy_amp`, are used to compute the amplitude envelope. The amplitude envelope captures the maximum amplitude within specified frame sizes, which is a crucial feature for various audio processing tasks.

3.4 Time and Frame Calculation

The code calculates the time and frame indices for the amplitude envelope using the `librosa.frames_to_time` function, enabling alignment of the envelope with time for visualization.

3.5 Visualizing the Envelope

Another Matplotlib plot is generated, displaying the audio waveform and overlaying the amplitude envelope in red. This visualization helps in understanding how the amplitude changes over time.

3.6 Short-Time Fourier Transform (STFT)

To delve into the audio's time-frequency characteristics, the code computes the STFT using `librosa.stft`. The STFT provides a detailed representation of the audio signal in the time and frequency domains.

3.7 Pitch Detection

For pitch analysis, the code uses the `librosa.piptrack` function to identify pitch frequencies in each frame. This process is achieved by tracking the peaks in the magnitude of the STFT.

3.8 Note Mapping

A dictionary, `note_mapping`, is defined to map detected frequencies to their corresponding musical note names, allowing for easier interpretation of the pitch information.

3.9 Average Pitch Calculation

The code calculates the average pitch within specific frame intervals to provide a more generalized view of the audio's pitch characteristics. It calculates the mean pitch, ignoring any NaN values in the pitch data.

3.10 Displaying Results

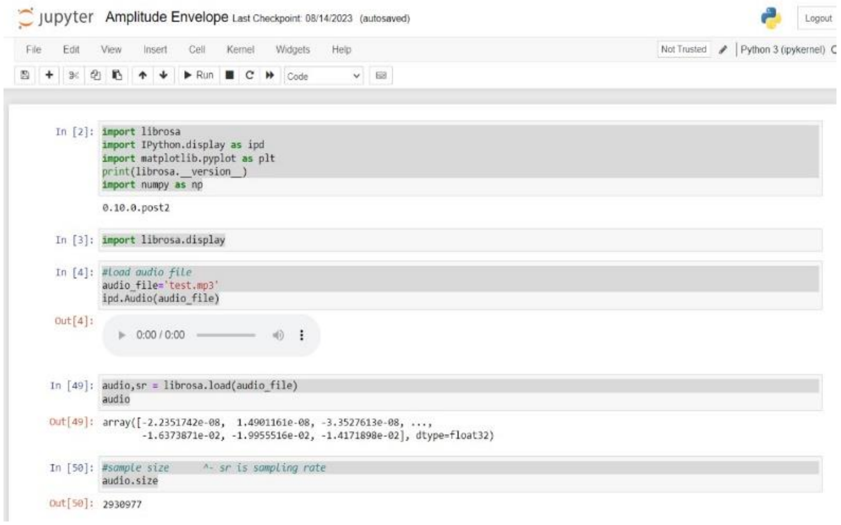
Finally, the code prints and displays the average frequency and corresponding musical notes for each set of frames at specified intervals.

4 Implementation

We integrate signal processing, visualization, and feature extraction techniques to analyze audio data, highlighting attributes such as amplitude variations and pitch characteristics in a structured and visually informative manner.

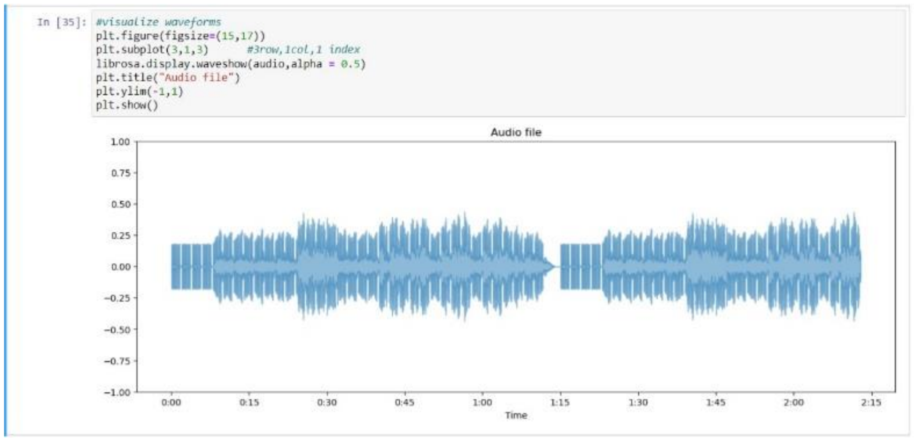
The methodology primarily relies on traditional audio analysis techniques and visualization, rather than machine learning models or classifiers.

5 Results


jupyter Amplitude Envelope Last Checkpoint: 08/14/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
0:00 / 0:00

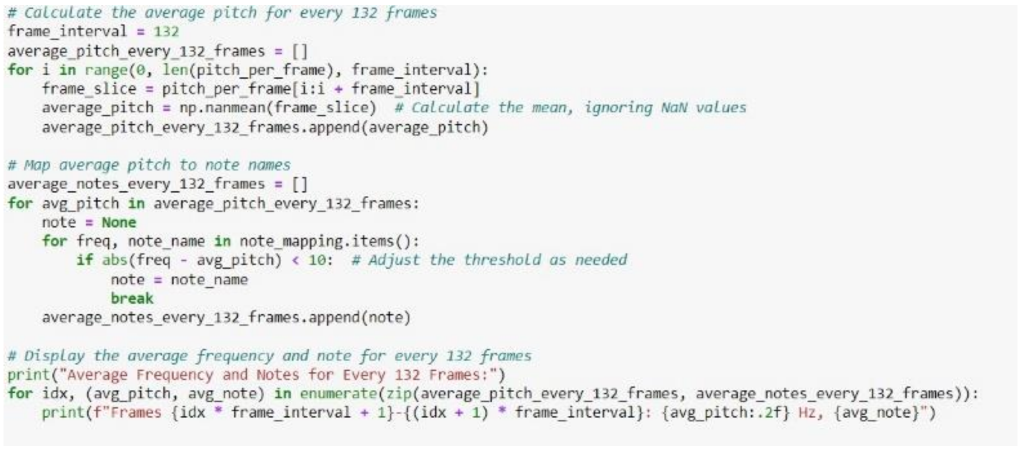
```
In [2]: import librosa
        import IPython.display as ipd
        import matplotlib.pyplot as plt
        print(librosa.__version__)
        import numpy as np

        0.10.0.post2

In [3]: import librosa.display

In [4]: #load audio file
        audio_file='test.mp3'
        ipd.Audio(audio_file)

Out[4]:

In [49]: audio,sr = librosa.load(audio_file)
         audio

Out[49]: array([-2.2351742e-08,  1.4901161e-08, -3.3527613e-08, ...,
                -1.6373871e-02, -1.9955516e-02, -1.4171898e-02], dtype=float32)

In [50]: #sample size      ^- sr is sampling rate
         audio.size

Out[50]: 2930977
```

Fig. 1 Amplitude Envelope Jupyter Notebook Code



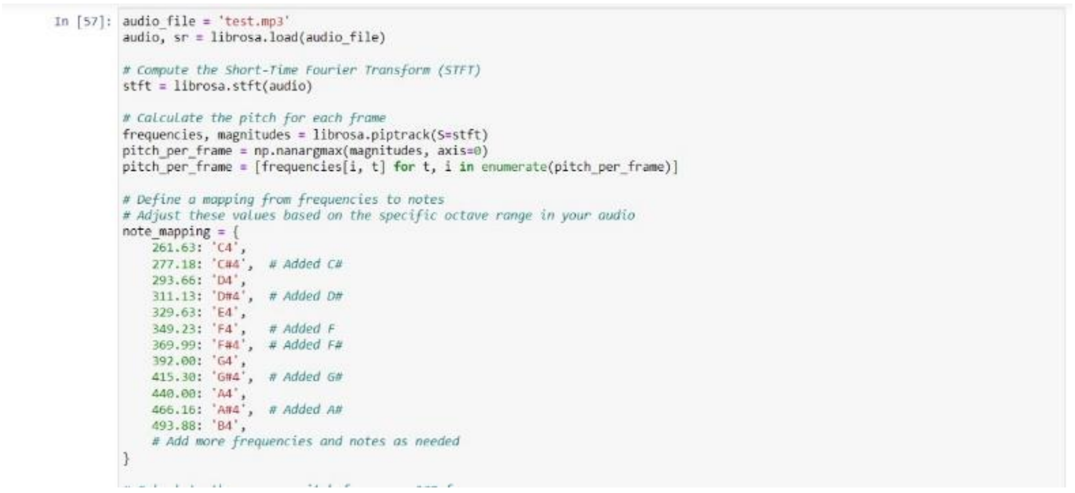
```
In [35]: #visualize waveforms
         plt.figure(figsize=(15,17))
         plt.subplot(3,1,3)       #3row,1col,1 index
         librosa.display.waveshow(audio,alpha = 0.5)
         plt.title("Audio file")
         plt.ylim(-1,1)
         plt.show()
```

Fig. 2 Visualizing Waveforms



```python
# Calculate the average pitch for every 132 frames
frame_interval = 132
average_pitch_every_132_frames = []
for i in range(0, len(pitch_per_frame), frame_interval):
    frame_slice = pitch_per_frame[i:i + frame_interval]
    average_pitch = np.nanmean(frame_slice)  # Calculate the mean, ignoring NaN values
    average_pitch_every_132_frames.append(average_pitch)

# Map average pitch to note names
average_notes_every_132_frames = []
for avg_pitch in average_pitch_every_132_frames:
    note = None
    for freq, note_name in note_mapping.items():
        if abs(freq - avg_pitch) < 10:  # Adjust the threshold as needed
            note = note_name
            break
    average_notes_every_132_frames.append(note)

# Display the average frequency and note for every 132 frames
print("Average Frequency and Notes for Every 132 Frames:")
for idx, (avg_pitch, avg_note) in enumerate(zip(average_pitch_every_132_frames, average_notes_every_132_frames)):
    print(f"Frames {idx * frame_interval + 1}-{(idx + 1) * frame_interval}: {avg_pitch:.2f} Hz, {avg_note}")
```

Fig. 3 Calculating average pitch and mapping to note names



```python
In [57]: audio_file = 'test.mp3'
         audio, sr = librosa.load(audio_file)

         # Compute the Short-Time Fourier Transform (STFT)
         stft = librosa.stft(audio)

         # Calculate the pitch for each frame
         frequencies, magnitudes = librosa.piptrack(S=stft)
         pitch_per_frame = np.nanargmax(magnitudes, axis=0)
         pitch_per_frame = [frequencies[i, t] for t, i in enumerate(pitch_per_frame)]

         # Define a mapping from frequencies to notes
         # Adjust these values based on the specific octave range in your audio
         note_mapping = {
             261.63: 'C4',
             277.18: 'C#4',  # Added C#
             293.66: 'D4',
             311.13: 'D#4',  # Added D#
             329.63: 'E4',
             349.23: 'F4',   # Added F
             369.99: 'F#4',  # Added F#
             392.00: 'G4',
             415.30: 'G#4',  # Added G#
             440.00: 'A4',
             466.16: 'A#4',  # Added A#
             493.88: 'B4',
             # Add more frequencies and notes as needed
         }
```

Fig. 4 Calculate pitch and map frequencies to notes

6 Conclusion

In the pursuit of understanding the intricate world of musical frequency note detection, this research has illuminated the potential for innovative applications and the broader impact on the field of music and technology. Our exploration has unveiled several key findings and insights.

First and foremost, we have highlighted the significance of accurate note detection in various domains, including music education, transcription, and audio processing. The ability to precisely identify musical notes is essential for musicians and music educators, as it can enhance the teaching and learning of music, providing valuable tools for musicians to fine-tune their performance and compositions.

Moreover, our investigation has uncovered the advancements in technology, such as digital signal processing and machine learning, that have made automated note detection more efficient and accessible. These technologies open doors to developing software tools and applications that can assist both amateur and professional musicians in their creative processes.

Furthermore, we have emphasized the importance of rigorous research in this field, including the need for large and diverse datasets, robust algorithms, and continuous refinement of techniques to improve the accuracy and reliability of note detection systems.

In conclusion, the pursuit of musical frequency note detection represents not only a technical endeavour but also a creative and educational one. It harmonizes the worlds of music and technology, offering new dimensions for artistic expression and learning. As we continue to refine and innovate in this area, we can anticipate a future where music becomes more accessible, comprehensible, and enriched for all. This research is but a glimpse into the vast potential that lies ahead in the realm of musical note detection.

7 References