BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

(An autonomous institution, Affiliated to VTU, Belagavi)

Avalahalli, Doddaballapura Main Road, Yelahanka, Bangalore-64

Department of MCA

Major Project [22MCA403]

Student Name: Sathwik P S USN: 1BY22MC046

Date of Submission: 18TH May 2024

Project Title: Tune Master

Sl. No	Particulars	Remarks
1.	Introduction	
2.	Literature Survey	
3.	Features	
4.	Proposed System	
5.	Modules Identified	
6.	Hardware and Software Requirements	

Recommendations: Accepted (Y/N)

Suggestions by the Guide:

Signature of Guide

INTRODUCTION

The project focuses on developing an advanced audio tuner application using Python and various digital signal processing (DSP) techniques. The primary objective is to create a tool that can accurately detect and analyse musical notes and frequencies in real-time, providing valuable feedback to musicians and audio engineers. The application leverages libraries such as Tkinter for the graphical user interface (GUI), pyaudio for audio capture, and specialized algorithms for frequency detection and noise reduction.

The tuner application aims to address common challenges in audio tuning, such as handling noisy environments and providing real-time feedback. By incorporating robust pitch detection algorithms and noise reduction techniques, the project ensures high accuracy and reliability. The GUI allows users to interact with the application easily, offering functionalities like recording, setting noise reduction levels, and displaying detected frequencies and notes.

This project is particularly relevant for musicians who need precise tuning tools for practice and performance, as well as audio professionals who require accurate frequency analysis in various environments. Through the integration of advanced DSP methods and user-friendly interfaces, the tuner application enhances the overall audio tuning experience, making it a valuable tool in the field of music and audio engineering.

LITERATURE SURVEY

S.NO	Title	Methodology Identified	Conclusion
1.	Fundamentals of Audio Signal Processing	Provides detailed explanations of DSP techniques such as Fourier transforms, filtering, and spectral analysis.	Offers foundational knowledge for implementing and understanding audio signal processing crucial for tuner applications.
2.	Evaluation of Pitch Detection Algorithm	Comparative analysis of pitch detection algorithms like autocorrelation, cepstrum, and harmonic product spectrum.	Highlights the effectiveness of each algorithm, with cepstrum and harmonic product spectrum being particularly accurate for music applications.
3.	Real Sound Synthesis for Interactive Applications	Discusses various real-time audio synthesis and processing techniques, including physical modeling and sample-based synthesis.	Provides practical approaches for creating interactive and responsive audio applications, enhancing real-time performance.
4.	YIN, a Fundamental Frequency Estimator for Speech and Music	Introduces the YIN algorithm, which improves pitch detection by reducing pitch ambiguity and increasing robustness.	Demonstrates higher accuracy and lower error rates compared to traditional methods, making it suitable for both speech and music applications.

5.	Musical Tuning Systems	Explores the mathematical and theoretical foundations of various musical tuning systems, including just intonation and equal temperament.	Offers insights into the relationship between frequency ratios and perceived musical harmony, aiding in the accurate mapping of frequencies to notes.
6.	A Review of Single-Channel Noise Reduction Methods	This paper provides an extensive review of various single-channel noise reduction techniques that are crucial for audio processing. Key methods discussed include: Spectral Subtraction, Wiener Filtering	Essential for improving audio quality and frequency detection accuracy, especially in noisy environments.
7.	Deep Learning for Music Information Retrieval	Examines the use of deep learning techniques, including convolutional and recurrent neural networks, for music information retrieval tasks.	Demonstrates that deep learning significantly enhances pitch detection and classification accuracy compared to traditional methods.
8.	The PyDub Library: A Simple and Easy Audio Processing Library in Python	Introduction and practical applications of the PyDub library for audio manipulation, including format conversion and signal processing.	PyDub provides an accessible and efficient tool for audio data manipulation in Python, suitable for developing audio applications.

9.	A Comparison of Time-Domain Pitch Detection Algorithms	Comparative study of time-domain pitch detection methods like autocorrelation, average magnitude difference function (AMDF), and zero-crossing rate.	Autocorrelation and AMDF are found to be more accurate and computationally efficient, making them suitable for real-time applications.
10.	Pitch Detection Using the Harmonic Sum Spectrum	Introduces the Harmonic Sum Spectrum (HSS) method for pitch detection, which sums harmonics to emphasize the fundamental frequency.	HSS method effectively reduces errors in pitch detection, particularly in complex musical signals, and offers high accuracy.

FEATURES

1. **Real-Time Frequency Detection:**
    - The application captures audio in real-time and analyses the frequency content to determine the current pitch

2. **Accurate Note Identification:**
    - Converts detected frequencies into musical notes, providing users with precise information about the pitch

3. **Graphical User Interface (GUI):**
    - Utilizes Tkinter to create an intuitive and user-friendly interface.
    - Displays current frequency, detected note, and a timer for recording sessions.

4. **Recording and Playback:**
    - Allows users to record audio sessions and save them for later analysis.
    - Stores recordings in WAV format for high-quality playback.

5. **Logging and History:**
    - Logs detected notes and frequencies along with timestamps.
    - Provides a history view to review previously detected notes and frequencies.

6. **Noise Reduction Settings:**
    - Includes adjustable noise reduction settings to improve accuracy in various environments.
    - Users can customize volume division and noise reduction levels through the settings menu.

PROPOSED SYSTEM

1. Audio Input and Processing:

Microphone Input: The system captures audio input using the computer's microphone.

Pyaudio Library: Utilized for accessing and managing the audio stream, ensuring real-time capture and processing of audio data.

2. Frequency Detection Algorithm:

FFT (Fast Fourier Transform): Applies FFT to convert time-domain audio signals into their frequency components.

Frequency Analysis: Identifies the dominant frequency from the FFT results, which corresponds to the pitch of the input sound.

3. Note Identification:

Mapping Frequencies to Musical Notes: Converts detected frequencies into corresponding musical notes using predefined frequency ranges for each note.

Note Stabilization: Implements logic to stabilize note detection, preventing rapid fluctuations in the displayed note due to minor frequency variations.

4. Graphical User Interface (GUI):

Tkinter Library: Provides a user-friendly and visually appealing interface.

Real-Time Display: Continuously updates and displays current frequency, detected note, and elapsed time.

Control Buttons: Includes buttons for starting/stopping recordings, accessing settings, and selecting output directories.

5. Recording and Playback:

Audio Recording: Records audio sessions and saves them as WAV files.

File Management: Allows users to choose the output directory for saving recordings and logs.

6. Settings and Customization:

Volume Division and Noise Reduction: Users can adjust settings for volume division and noise reduction levels.

Settings Window: Provides an interface for users to configure these parameters according to their preferences.

7. Multi-threading for Performance:

Concurrent Processing: Uses threading to handle audio processing and GUI updates simultaneously, ensuring a responsive user experience.

MODULES IDENTIFIED

1. **Audio Input Module** captures the sound and sends it to the Frequency Detection Module.
2. **Frequency Detection Module** analyses the audio and sends the detected frequencies to the Note Identification Module.
3. **Note Identification Module** maps these frequencies to musical notes and sends the information to the GUI Module for display.
4. **Noise Reduction Module** processes the input sound to enhance frequency detection accuracy and sends the cleaned signal back to the Frequency Detection Module.
5. **Recording and Playback Module** allows users to start and stop recording, saving the audio files, and works with the GUI Module to provide control and feedback.
6. **Logging and History Module** records and displays detected notes and frequencies during the session, interacting with the GUI Module to update the display.

HARDWARE AND SOFTWARE REQUIREMENTS

1. Hardware Requirements:

- Processor: Quad-core Processor (e.g., Intel i3 and above or AMD Ryzen 5 and above)
- RAM: Minimum 4 GB
- Storage: SSD or HHD with 256 GB Capacity
- Microphone

2. Software Requirements:

Operating System:

- Windows 7 or later
- macOS 10.10 or later
- Linux distributions (Ubuntu, Fedora, etc.)

Programming Language:

- Python

Libraries Used:

- tkinters
- pyaudio
- numpy
- wave
- threading
- note

S.NO	Title	Methodology Identified	Conclusion
1.	Fundamentals of Audio Signal Processing	Provides detailed explanations of DSP techniques such as Fourier transforms, filtering, and spectral analysis.	Offers foundational knowledge for implementing and understanding audio signal processing crucial for tuner applications.
2.	Evaluation of Pitch Detection Algorithm	Comparative analysis of pitch detection algorithms like autocorrelation, cepstrum, and harmonic product spectrum.	Highlights the effectiveness of each algorithm, with cepstrum and harmonic product spectrum being particularly accurate for music applications.
3.	Real Sound Synthesis for Interactive Applications	Discusses various real-time audio synthesis and processing techniques, including physical modeling and sample-based synthesis.	Provides practical approaches for creating interactive and responsive audio applications, enhancing real-time performance.
4.	YIN, a Fundamental Frequency Estimator for Speech and Music	Introduces the YIN algorithm, which improves pitch detection by reducing pitch ambiguity and increasing robustness.	Demonstrates higher accuracy and lower error rates compared to traditional methods, making it suitable for both speech and music applications.
5.	Musical Tuning Systems	Explores the mathematical and theoretical foundations of various musical tuning systems,	Offers insights into the relationship between frequency ratios and perceived musical

		including just intonation and equal temperament.	harmony, aiding in the accurate mapping of frequencies to notes.