Voice Changer Application Document

Program introduction

Voice Changer is a sound processing application that allows users to record voices, import existing sound files, and apply different sound effects, such as gender switching, robot sound effects, and children's voices. The app is designed to provide users with an easy way to explore and play with different sound effects.

Installation guide

- Python 3.x
- Operating system: Windows, MacOS, Linux
- The necessary dependencies: pyaudio numpy keyboard sounddevice scipy librosa

Run the following command from the command line or terminal to install the necessary dependencies: **pip install pyaudio numpy keyboard sounddevice scipy librosa**

Usage Instructions

First make sure voice_changer.py and main.py are in the same directory, then Navigate to the directory containing the program files in the terminal or command line, enter **python main.py** to run the program.

After starting the program, the main menu will be displayed, where users can choose voice data processing (record or import). Select the Sound processing and playback (original, gender transformation, robot effect, child voice).Follow the prompts, such as pressing the Esc key to stop recording when recording sound. The program will play and save the selected sound effect.

Main Features

- Sound Recording and Importing：Allows users to record new sound or import existing WAV files.

- Add effects：users can choose to add gender conversion, robot, child effects to the original sound.
- Play and save：The software can directly play the sound with added effects and save it to the current path.

Developer Guide

Installation library instructions

- pyaudio: used to record sound
- numpy: perform calculations
- keyboard: listen to keyboard events
- sounddevice: play sound
- scipy: signal processing
- librosa: audio analysis and effects processing
- datetime: used when generating file names

application structure

The application comprises two main Python scripts: main.py and voice_changer.py. main.py is the entry point of the application. It handles user interactions through the command line, allowing users to choose between different sound processing options. Voice_changer.py contains the core functionality for audio processing, including recording, effect application, and playback etc.

function description

wav_generator(data, s)

- Function: Use audio data to generate WAV files and save the files in the current directory.
- data: NumPy array, containing audio data.
- s: String, prefix of the generated WAV file.
- Output: None (prints file save path on success, error message on failure).

read_wav_file(filename)

- Function: Read the WAV file at the specified path and return its audio data.

- filename: string, path to WAV file.
- Output: NumPy array, containing the audio data of the WAV file; if the reading fails, None is returned.

low_pass_filter(data, sr, cutoff)

- Function: Apply a low-pass filter to audio data.
- data: NumPy array, original audio data.
- sr: integer, audio sampling rate.
- cutoff: integer, the cutoff frequency of the low-pass filter.
- Output: NumPy array of audio data with low-pass filtering applied.

high_pass_filter(data, sr, cutoff)

- Function: Apply a high-pass filter to audio data.
- data: NumPy array, original audio data.
- sr: integer, audio sampling rate.
- cutoff: integer, the cutoff frequency of the high-pass filter.
- Output: NumPy array of audio data with high-pass filtering applied.\

record_voice()

- Function: Record sound until the user presses the "Esc" key.
- Input: None.
- Output: NumPy array containing the recorded audio data.

play_audio(data, sr=44100)

- Function: Play the given audio data and display the playback progress.
- data: NumPy array, audio data to be played.
- sr: integer (optional), the sampling rate of audio data, the default is 44100Hz.
- Output: None (plays audio and shows progress).

remove_noise(data)

- Function: Remove noise from audio data.
- data: NumPy array, original audio data.
- Output: NumPy array, audio data with noise removed.

analyze_audio(data)

- Function: Analyze audio data to determine the gender of a voice.
- data: NumPy array, original audio data.

- Output: A tuple containing two elements: (1) a string, the gender of the voice; (2) a float, the average pitch of the audio.

change_gender(data, sr=44100, gender=None, avg_pitch=0)

- Function: Adjusts audio data based on a specified gender and average pitch to simulate the voice of another gender.
- data: NumPy array, original audio data.
- gender: string, target gender.
- avg_pitch: Floating point number, the average pitch of the original audio data.
- Output: NumPy array of gender-adjusted audio data.

robot_effect(data, mod_freq=270, gender=None, avg_pitch=0)

- Function: Add robot effects to audio data.
- data: NumPy array, original audio data.
- mod_freq: integer, modulation frequency, default is 270Hz.
- gender: String, the gender of the original audio.
- avg_pitch: floating point number, the average pitch of the original audio
- Output: NumPy array of robot-adjusted audio data.

child_effect(data)

- Function: Adjust the pitch of audio data to simulate a child's voice.
- data: NumPy array, original audio data.
- Output: NumPy array of audio data that has been pitch-adjusted to simulate a child's voice.

Extended Guide

Developers can extend the functionality of their applications by adding new sound processing functions. For example, implement new sound effects or improve algorithms for existing effects. Just add a new function in voice_changer.py and add the corresponding user interaction logic in main.py.