ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Vertically Segmented Target Zone based Audio Fingerprinting

Author:
Jifar Mekonnen Hunde

Advisor:
Dr. Surafel Lemma Abebe

A thesis submitted in partial fulfillment of the requirements for the Masters of Science in Computer Engineering

April 19, 2022
Addis Ababa, Ethiopia

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Vertically Segmented Target Zone based Audio Fingerprinting

by Jifar Mekonnen Hunde

Approval by Board of Examiners

Dr. Bisrat Derebssa Dean, SECE, AAiT	_____ Signature
Dr. Surafel Lemma Advisor	_____ Signature
_____ Thesis Examiner	_____ Signature
_____ Thesis Examiner	_____ Signature

Addis Ababa, Ethiopia

Declaration of Authorship

I, Jifar Mekonnen Hunde, declare that this thesis titled, "Vertically Segmented Target Zone based Audio Fingerprinting" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a masters degree at this University.
- This work has not been submitted, in whole or in part, for any other degree or professional qualification except as specified.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

Signed:

Date:

Abstract

An audio fingerprint is a set of perceptual features that uniquely identify an audio file. Audio fingerprinting has applications in broadcast monitoring, meta-data collection, royalty tracking, etc. Audio fingerprinting systems suffer a lot from noise, compression, and modifications present in the audio. Pitch shifting is one such audio modification. Common real-world scenarios where pitch-shifting occurs include radio broadcasts, DJ sets, and deliberate alterations. Since pitch-shifting scales the spectral content of the original audio, matching pitch-shifted query audio to its original unmodified version is challenging. This thesis work proposes a Shazam-based audio fingerprinting system resistant to pitch-shifting.

The proposed approach uses CQT to transform the scaling effect of pitch-shifting into vertical translation. From the spectrogram generated by CQT, the proposed approach picks triple spectral peaks to encode pitch-shifting resistant fingerprint hashes. Vertically segmented target zones were employed to organize spectral peaks into triplets. By increasing the locality of the generated fingerprint hashes, vertically segmenting the spectrogram minimizes the effect of pitch-shifting. A fingerprint hashing scheme that leverages vertically segmented target zones is proposed.

A total of 42,000 query audio and a reference database of 3000 freely available songs were used to evaluate the proposed approach as well as the chosen baseline works: Panako and Quad. The result collected shows that the proposed approach handles pitch-shift modifications from -11% to +12% except for modification values of -8, -3, +3, and +9 percent. Panako achieved to identify queries with -6% to +6% pitch shifts except for modification values of -3 and +3 percent. Quad, on the other hand, can handle -12% to +7% pitch shifts with no such drops. The proposed approach is also robust to linear speed modification from -6% to +12%, which is a significant improvement over Panako, which can only handle -4% to +8% modifications. Quad showed better robustness to linear speed modification by handling rates ranging from -16% to +12%. However, Quad took, on average, 3 times more time to query a single audio than the proposed approach. Moreover, the proposed approach shows robustness to common audio effects such as echo, tremolo, flanger, band-pass, and chorus while Quad suffered significant accuracy drop for chorus, flanger and tremolo.

Keywords: *Audio Fingerprinting, Vertically Segmented Target Zone, Pitch Shifting*

Acknowledgements

I would like to thank my advisor Dr. Surafel Lemma for his guidance throughout this research.

Contents

List of Figures

List of Tables

List of Abbreviations

CQT	Constant Q Transform
FFT	Fast Fourier Transform
FLAC	Free Lossless Audio Codec
LFO	Low Frequency Oscillator
ORB	Orient Fast and Rotate Brief
PCM	Pulse Code Modulation
SIFT	Scale Invariant Feature Transform
STFT	Short Time Fourier Transform
WAV	Waveform Audio File Format

Chapter 1

Introduction

An audio fingerprinting is a process of producing a unique and compact digital summary, fingerprint, of an audio signal to identify it from a collection of audios. The audio fingerprint is a summary of audio. The fingerprint should contain the perceptual characteristics of the original audio. If two audios are perceptually similar, then their audio fingerprints should be alike.

The audio fingerprinting process has two distinct phases: the audio fingerprint generation and the fingerprint matching phase. The fingerprint generated in the first phase should take into account the perceptual characteristics of the audio. It should be concise as well as noise and modification-tolerant version of the original audio data. Additionally, the fingerprint should have enough entropy, i.e, information, to yield a high true positive hit and a less false positive hit in the fingerprint matching phase using realistic computational resources such as time and storage.

Audio fingerprinting system should meet numerous specific requirements based on the above general principles, including accuracy, reliability, adaptability, scalability, and robustness [1]. This research focuses on the robustness requirement, which is the ability of fingerprinting system to correctly identify sample audio in the presence of noise, compression, degradation, and alteration.

The three most common challenges for the robustness of a fingerprinting system are speed change, time-stretching, and pitch shifting. Speed change occurs when audio is played faster and at the same time at a higher pitch, as a result, a scaling occurs in both pitch (frequency) and time (tempo) axis. Time stretching on the other hand occurs when audio is speeded up or slowed down without observable changes in pitch. Pitch shifting is said to occur when the frequency (pitch) scale (axis) is modified while keeping the time scale unchanged.

Thus, to correctly identify query audio in the presence of these modifications a robust audio fingerprinting system is vital. Among the different types of changes that challenge audio fingerprinting, this research focuses on handling pitch shifting.

1.1 Problem Statement

The pitch of a sound correlates with the set of frequencies the sound is composed of. When pitch-shifting occurs, all these frequencies in the sound are multiplied by a factor K, which produces pitch-shifted perception to the listener [2]. For example, if we shifted up sound of a man singing, it could be perceived as a woman singing by the listener.

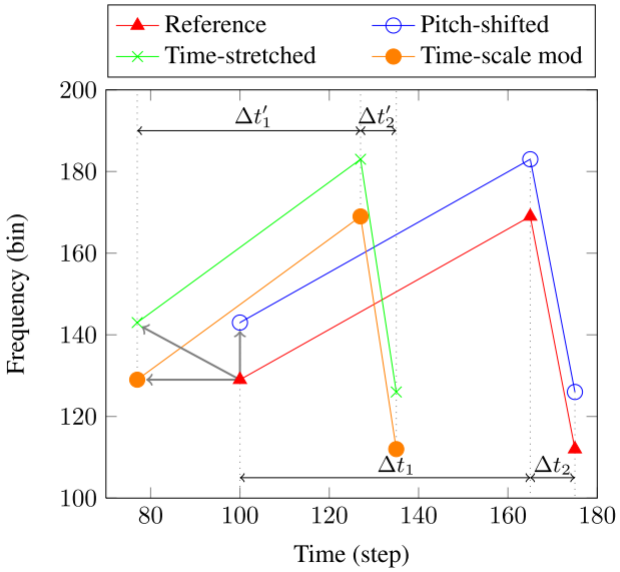

FIGURE 1.1: The effect of time-scale and pitch modification. (Taken From [3])

Figure 1.1 shows the event points, which are points where the magnitude of the signal peaks, in the spectrogram, created using Constant-Q transform (CQT), of the original (red lines) and modified (green, orange, and blue lines) audios. The red triangles represent the event points of the original audio whereas the blue circles, the orange circles, and green crosses represent pitch shifted, linear speed changed and time stretched audios respectively. From Figure 1.1 we can see that the frequency components of peak

points of the pitch-shifted audio, represented by blue circles, are changed while their time components remain the same. Figure 1.1 clearly indicates that pitch shifting is a vertical translation along the frequency axis.

Let denote the event points of the reference audio, represented by the red triangles in Figure 1.1, as $P_1(t_1, f_1)$, $P_2(t_2, f_2)$ and $P_3(t_3, f_3)$ from left to right. Similarly, lets denote their corresponding pitch shifted points, represented by blue circles, as $P'_1(t'_1, f'_1)$, $P'_2(t'_2, f'_2)$ and $P'_3(t'_3, f'_3)$. Since pitch-shifting is vertical translation when CQT is used to generate the spectrogram image [2], the following equation holds true.

$$f'_1 = f_1 + K; f'_2 = f_2 + K; f'_3 = f_3 + K \tag{1.1}$$

Looking at the red triangles and their corresponding pitch shifted white circles, we can see that white circles are vertical translations of the red triangles with some constant value, K. Mathematically, we can put the relationship as follows:

$$f'_1 - f'_2 = (f_1 + K) - (f_2 + K) \tag{1.2}$$

$$f'_2 - f'_3 = (f_2 + K) - (f_3 + K) \tag{1.3}$$

We can say the two values around the equal sign are always the same since we assumed pitch shifting causes every query audio frequency components to translate by K amount from the corresponding original reference audio frequency components.

From Equation 1.2 and 1.3, we can observe that even though pitch shifting translated the original audio's frequency components to newer values the delta between the frequency components is the same for the original and pitch shifted version of it. Therefore, we can use the delta between the frequency components to get pitch shifting invariant audio feature.

Six and Leman [3] used this property while formulating their audio hash. They used the two deltas found in the left sides of Equations 1.2 and 1.3 as well as the slightly modified values of frequency components of P_1 and P_2.

$$Frequency\_Component\_Hash = (f_1 - f_2; f_2 - f_3; `f_1; `f_3) \tag{1.4}$$

Six and Leman [3] included Equation 1.4 as part of their fingerprint hash. $`f_1$ and $`f_3$ are sufficiently rough indications of where f_1 and f_3 land in the spectrogram. They are given as:

$$`f_1 = f_1/16; `f_3 = f_3/16 \tag{1.5}$$

These values, i.e $'f_1$ and $'f_3$, make the hash more discriminative at the same time limiting how much the audio can be pitch-shifted. The limitation on the amount of pitch shift has limited the true positive rate to only 80% for query audio with 3% or less pitch shift [3]. The reason behind this is for large values of pitch shifting the original audio $'f_1$ and $'f_3$ and the pitch shifted (query) audio $'f_1'$ and $'f_3'$ will vary significantly resulting in different hash values. For smaller K, $'f_1' \approx 'f_1$ since $'f_1' = 'f_1 + K$. This thesis work aims to resolve this issue by focusing on the effect of introducing $'f_1$ and $'f_3$ in the hash.

This thesis aims to design and develop pitch shift resistant audio fingerprinting system by focusing on solving the above discussed problems. The research questions this thesis work aims to answer are listed below:

- RQ1: [Effect of using the frequency delta only in the hash]. Is it possible to use frequency delta only to formulate a pitch-shifting invariant fingerprint hash?
- RQ2: [Vertically Segmented Target Zone Assignment Algorithm]. Can use of vertically segmented target zone assignment improve the matching of pitch shifted audio queries?

1.2 Objectives

1.2.1 General Objective

The main objective of the work is to develop pitch shift resistant audio fingerprinting system.

1.2.2 Specific Objectives

- To study the effect of using frequency delta to construct the hash on the fingerprinting system.
- To design and develop new target zone assignment algorithm and fingerprint extraction algorithm.
- To gather, prepare and organize reference and test audio data.
- To compare the developed audio fingerprinting system with other audio fingerprinting systems.

1.3 Scope

The scope of this thesis work is limited to handling pitch shifting in audio fingerprinting. The impacts of linear speed change and time-stretching on the robustness of audio fingerprinting algorithms were not addressed in this study.

1.4 Research Methodology

In order to meet the above research objectives the following research steps are carried out:

- **Literature Review:** Review of related works are carried out here to clearly specify the research problem and to identify the gap in the previous works related to this problem.
- **Propose an Approach:** Based on the gap identified above and related previous works' solutions a new approach was proposed.
- **Data Collection and Preparation:** Here an open source audio data was collected from Jamendo (`https://www.jamendo.com`) data set. From the collected audio data set, a set of test audio query data was generated for audio modifications and audio effects such as linear speed change, pitch shift, time stretch, chorus, tremolo, flanger, echo, and band-pass.
- **Implement the proposed approach:** The proposed approach was implemented here.
- **Analyze and Evaluate the proposed approach:** Different analysis is done on the proposed system using different experiments. The experiments are carried on the collected audio data as well as on the modified query excerpts. The analysis is also conducted on previous baseline works in order to compare them to the proposed system.

1.5 Contributions

The main contribution of this thesis work is developing a new vertically segmented target zone assignment based audio fingerprinting system.

1.6 Thesis Organization

The remainder of the this document is structured as follows. Theoretical backgrounds of audio fingerprinting systems is discussed in Chapter 2. Chapter 3, presents the related previous research works. Chapter 4 discusses each component of the proposed approach. Experimental procedures followed and the results gathered are discussed in Chapter 5. Finally, Chapter 6 presents the conclusion of this thesis work and recommended future works.

Chapter 2

Audio Fingerprinting

Audio fingerprinting is a technology for representing audio in a concise way by extracting perpetual features of the audio. It is similar to how human fingerprinting is used to identify a human. Audio fingerprinting works by first recording the fingerprint of audio in the database. Upon request, It matches the query with the recorded audio. The matching is done on the fingerprints, not on the actual audio object, which has a number of advantages [1]. The first one is the fingerprints contains relevant features of the original audio which will most likely be present in the query audio even if the query audio undergoes various audio compression, modifications, and degradation. The second advantage is that the fingerprint requires less memory than the original audio enabling faster computation, less storage, and faster communications.

The audio fingerprinting process has two distinct phases: the audio fingerprint generation phase and the fingerprint matching phase. The first phase is responsible for generating fingerprints from the original (or in other words reference) audio and storing the fingerprints along with the associated audio metadata in the database. The second phase is tasked to identify the reference audio when presented with the query audio. These phases require three major high-level components depicted in Figure 2.1 [3].

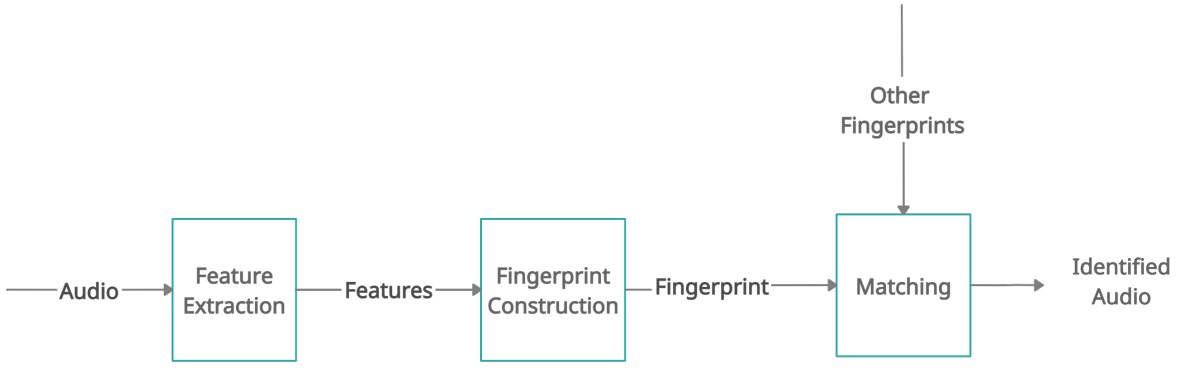

FIGURE 2.1: High level components of an audio fingerprinting system.

The feature extraction component accepts audio and extracts relevant audio features. These features are used in the fingerprint construction component to produce fingerprints. For the fingerprint generation phase, these fingerprints are stored in the database. Whereas for the matching phase these fingerprints are fed to Matching Component. Here, the fingerprints are compared with the other fingerprints in the database to identify the reference audio. Each of the components in Figure 2.1 will be discussed in detail in the following sections.

2.1 Feature extraction

Here an audio signal is converted into sequence of relevant features. The extraction process should take the following points into account [4]:

- The process transforms high dimensional audio in to lower dimensional data, fingerprints, to facilitate indexing and searching.
- It should analyze perceptually relevant data (identical to how the human auditory system functions).
- Robustness to distortions, noise, modifications, and degradation .

The first step in feature extraction is to pre-process the audio. Pre-processing includes: decoding the audio into more general format, such as mono PCM with a fixed sampling rate. After pre-processing, perceptually revelent audio features are extracted. Haitsma and Kalker [1] classifed perpertullay relavent audio features into two broad catergoires: semantic and non-semantic features. Semantic features include genre, beats-per-minute and mood. These features have more direct meaning for human auditory system whereas the non-semantic features are mathematical in nature.

This thesis focuces on the non-semantic audio features. Most common non-semantic audio feature extraction tools include: Mel Frequency Cepstral Coefficients (MFCC) [5], Discrete Cosine Transform (DCT) [6], Short-Time Fourier Transform (STFT) [7], Constant Q-Transform (CQT) [8], and Wavlet Transform [9].

Wang [10] chose spectrogram peaks as non-semantic perceptually relevant audio feature. A time-frequency point in the spectrogram is considered as a peak if it has highest energy content compared to its neighbors in a given defined region. The reason the peaks are chosen as relevant feature is that the peaks are most likely to survive distortions, modifications and, degradation.

2.1.1 Fingerprint Construction

The audio features extracted in the feature extraction step will be turned into audio fingerprints in the fingerprint construction stage. Every audio fingerprinting method organizes audio features into fingerprints in its unique way. This stage has a direct impact on the final fingerprint format as well as the matching phase. To produce translation invariant fingerprints, Shazam [10] based fingerprinting systems utilized two up to four spectral peaks. Panako [3] uses triplet while quad approaches [11], [12] use quadruplet spectral peaks. Philips [1] based systems, on the other hand, use the sign of energy difference (simultaneously along the time and frequency axes) between neighboring bands to build robust fingerprints.

A robust fingerprint, irrespective of the algorithm used to create it, should posses the following qualities [4]:

- It should contain a lot of acoustically relevant information that allows it to be identified out of a large number of fingerprints.
- It should be invariant to distortions, modifications, and degradation.
- It should be compact. Compactness allows for less storage and less computational requirement. However, it shouldn't be over compact since it may not be enough to discriminate among other audio fingerprints.
- It should be easy to compute. Fingerprint construction shouldn't be computationally expensive both in time and storage.

2.2 Matching

In this phase, the audio fingerprinting system tries to identify a potentially distorted, modified, or degraded query audio among the reference database. The matching is done on the fingerprints of the query and fingerprints stored in the database. Wang [10] and Six and Leman [3] employed exact matching technique for matching. They did a search on the reference fingerprint database for every query fingerprint. Audio identifiers with most matches will be subjected to further frequency and time alignment filter. The one with most aligned matches will be selected as final match.

Sonnleitner and Widmer [11] used a fixed-radius near neighbor search instead of exact matching as matching technique. They also adapted a filtering techniques in order to minimize false positives arising from the near neighbor search.

2.3 Audio Fingerprinting System Requirements

Haitsma and Kalker [1] listed parameters to characterize a good audio fingerprinting system. In this section, those qualities will be discussed.

- **Robustness:** It measures the system's ability to correctly identify query audio after modifications and degradation. It's measured with the false negative rate of the system. False negative occurs when the system can't identify query audio while the audio is in the database. Audio fingerprint systems should construct their fingerprint using perceptual features that are invariant to modifications and degradation in order to possess good robustness qualities.
- **Reliability:** This parameter is related with the system false positive rate (FPR). FPR is the rate the system produces incorrect identifier for the given query audios.
- **Fingerprint size:** Size of the fingerprint determines the memory space required for a fingerprint. It has direct implications on the computational requirements of the fingerprinting system.
- **Granularity:** This dictates the length of query audio needed for the correct identification. The length of query audio used in the fingerprinting system depends on the type of application. Applications such as consumer music identification service requires relatively small query length. Whereas, broadcast monitoring applications can ease the requirement for short granularity.
- **Search speed and scalability:** These parameters are related more with the fingerprinting extraction , fingerprint storage and matching mechanisms used. Ideally, the fingerprinting system should be scalable, in both processing time and memory requirements, with the number of audios in the database.

2.4 Robust Audio Fingerprinting System

Challenges for robustness of an audio fingerprinting system can be summarized as signal modifications and degradation. The most common modifications and degradations are:

- **Pitch shifting:** Its a modification type that changes the pitch of an audio without changing its length. All frequencies in the original audio are scaled by a constant factor to achieve pitch shifting. Comparing the spectrogram of the original audio clip with that of +20 percent pitch shifted and -20 percent pitch shifted audios in

Figure 2.2 a to c respectively, we can see that pitch shifting translates the original audio frequency component up or down while keeping the time axis intact. One can clearly observe from Figure 2.2 that the position of the green rectangle is shifted up in Figure 2.2.b relative to the Figure 2.2.a which indicates that +20 percent pitch shifting translates every frequency component in original audio upward (to higher frequency value). Similarly, from the position of the green rectangle in Figure 2.2.c and Figure 2.2.a we can see that -20 percent pitch shifting moves every frequency components in the original audio downward (to lower frequency value).

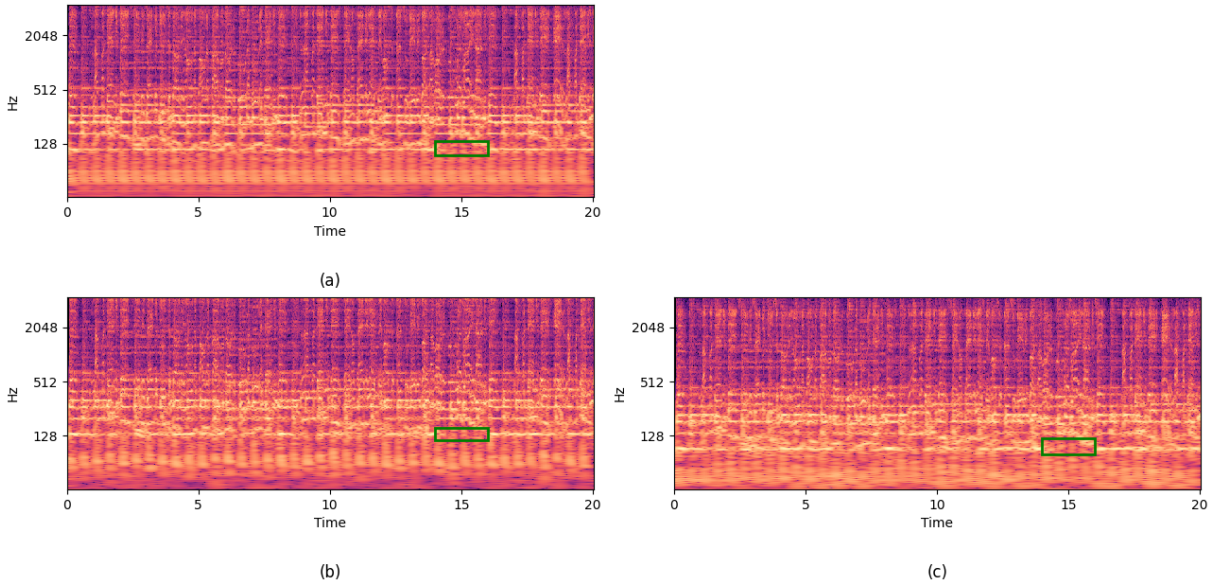

FIGURE 2.2: Spectrograms images of a song called La fraternitÃ© by Mob; (a) is the spectrogram of an original 20s music clip (b) is the spectrogram of +20 percent pitch shifting. (c) is the spectrogram of -20 percent pitch shifting.

- **Time Stretching:** Here the original audio is modified only in the time axis without affecting its pitch [13]. Figure 2.3.a and Figure 2.3.b show that the spectrograms of +20 percent and -20 percent time stretched version of the original audio in Figure 2.2.a. Comparing spectrogram of +20 percent time-stretched audio in Figure 2.3.a and with the original audio spectrogram in Figure 2.2.a, we can observe that the length of time-stretched audios shortened without affecting the frequency component. The position of the green rectangles in both figures witness this fact. The spectrogram of -20 percent time-stretched audio in Figure 2.3.b shows that the length of time-stretched audio lengthen compared to the original. The green rectangles in the figures show this fact clearly.

- **Speed Change:** This modification type affects both the tempo and pitch information of the original audio. Figure 2.3.c and Figure 2.3.d show the spectrograms

of +20 percent and -20 percent speed change of an audio respectively. The effect of speed up can be seen from the positions of the green rectangles. The rectangle in spectrogram of +20 percent speed up audio in Figure 2.3 moved upward in the frequency axis as well as moved towards left indicating speed up affects both in frequency and time components. This fact holds true for -20 percent speed up audio shown in Figure 2.3.d. Here the green rectangle moved downward in the frequency axis and outward in the time axis.

- **Noise and Audio Degradation:** The original audio may undergo thru various channels which will introduce some noise as well as it may degrade the quality of the original audio.

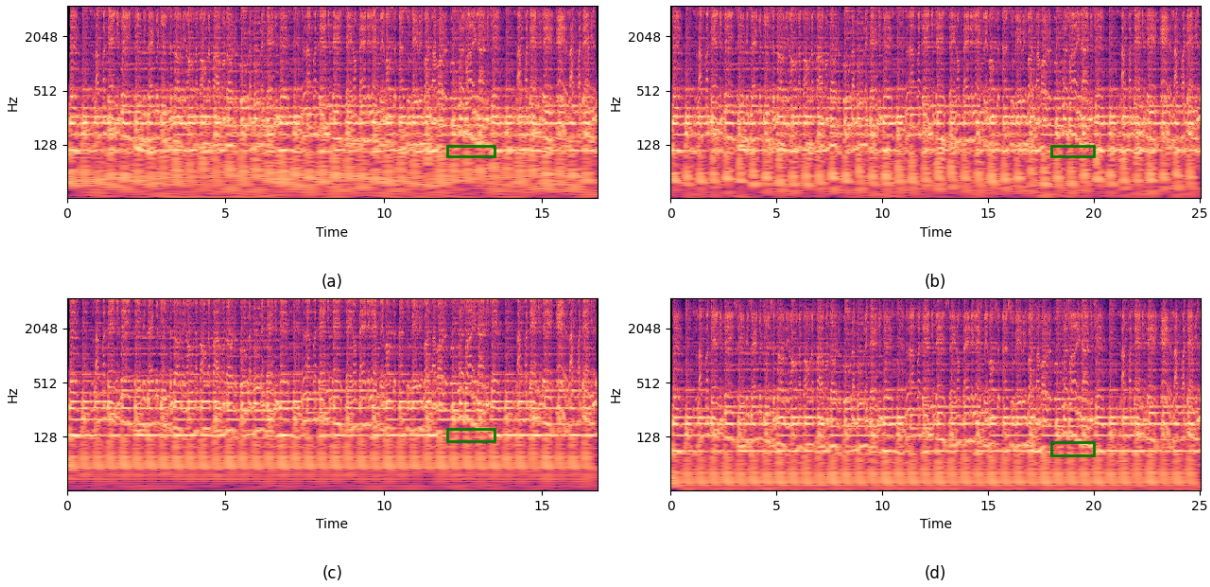

FIGURE 2.3: Spectrogram images of four audio excerpts from the original clip shown in Figure 2.2: (a) -20% time-stretching, (b) +20% time-stretching, (c) +20% linear speed modified, and (d) -20% linear speed modified

- **Common Audio Effects [14]:**
  - Band-pass: this effect allows only specified range of frequencies in the original audio to pass while removing the frequencies outside of the specified range.
  - Chorus: its the result of duplicating an audio signal, delaying, and varying the pitch of the copy or copies over time using low-frequency oscillator (LFO) and mixing the resulting signals back to the original signal. This will enable a single music instrument sound like more than one instruments played at the same time.

- Flanger: its similar with the Chorus effect but here shorter delay times are used.
- Tremolo: its the rapid reiteration of audio tones to produce a trembling effect to the listener.

The visual aspect of common audio effects in the spectrogram is shown in Figure 2.4.

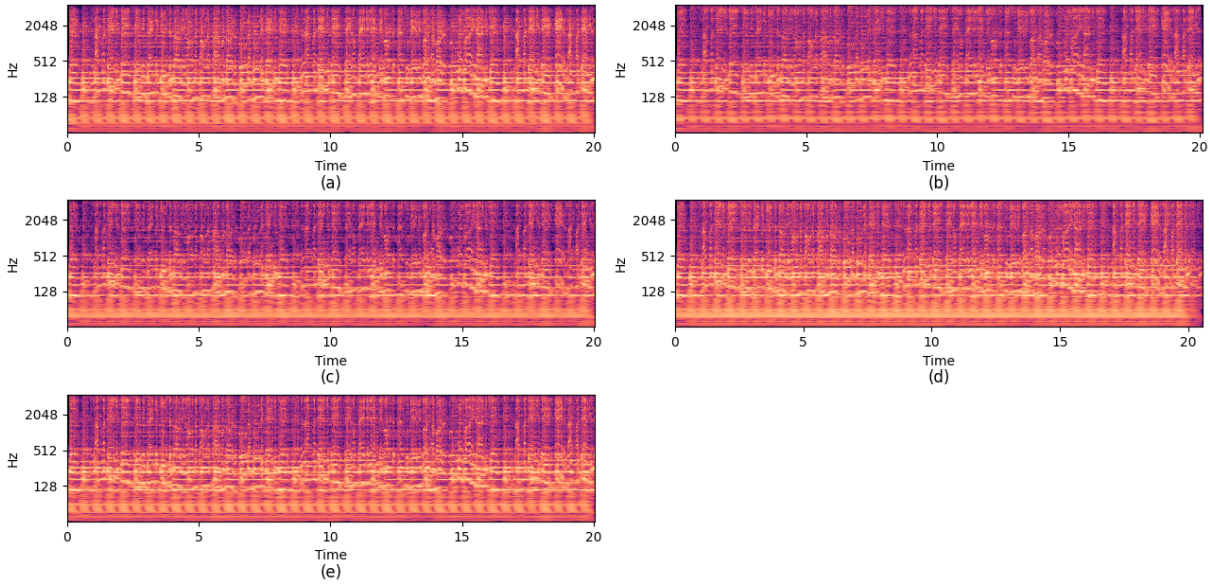

FIGURE 2.4: Spectrogram images of common audio effects. (a) 2KHz band-passed audio, (b) chorus , (c) flanger , (d) echo and, (e) tremolo

Chapter 3

Related Works

Several researches have been conducted to handle pitch shifting. They are organized based on the method they follow into three major categories: Image Processing approach, Philips approach, and Shazam approach.

3.1 Image Processing Approach

Image processing is used in audio fingerprinting because audio signals can be represented by two-dimensional time-frequency representations such as audio spectrograms. These two-dimensional audio representations can be plotted as images, allowing for the use of image processing techniques. Furthermore, most audio transformations, such as time stretching, have image transformations on their audio spectrograms. As a result, audio fingerprinting becomes a problem of identifying different patterns in spectrogram images that are resistant to geometric transformations like vertical and horizontal translations [15].

The first step to take when using image processing techniques for audio fingerprinting is to convert an audio signal into a two-dimensional spectrogram image. Then geometric transformation invariant transforms such as Scale Invariant Feature Transform (SIFT) [16] are applied to the image. SIFT is a local descriptors extraction technique that first finds places of interest in an image and then offers a characteristic description for them. To determine how similar two spectrograms are, a matching technique between their descriptors is used.

Scale invariant feature transform (SIFT) local descriptors derived from a spectrogram image were utilized as sub-fingerprints for music identification by Zhang et al. [17]. The authors stated that SIFT descriptors with local sensitive hashing exhibits a strong robustness against time stretching and pitch shifting. They reported good hit rates for query audios that are time-stretched from -30 percent to +50 percent or pitch-shifted from -50 percent to +100 percent. However, because they used SIFT local descriptors,

their work is computationally intensive, rendering it unsuitable for circumstances requiring quick response times [15].

Zanoni, M. et al [15] compared image descriptors like SIFT, Speeded Up Robust Feature (SURF)[18], Binary Robust Invariant Scalable Key-points (BRISK) [19], and Oriented FAST and Rotated BRIEF (ORB) [20]. The authors used image descriptors such as SURF and ORB to decrease the high computational requirement of SIFT with a slight reduction in hit rates. They manage to identify audios with pitch shifts ranging from -25.08 percent to +49.83 percent with average match rate up to 99 percent.

Williams et al.[21] used ORB descriptor, which is a computationally efficient replacement to SIFT that has similar matching performance [20], [22]. The authors employed ORB to compare the features of the spectrogram image query to a database of reference spectrogram images using a Brute Force matcher. The authors achieved a hit rate of more than 90 percent for query audios with a pitch shift of -20% to +20%. They also reported their work based on the ORB method is more computationally efficient and faster than the SIFT method-based works such as Zhang et al. [17]. However, their use of the brute force matcher to search for query matches from the reference audio database results in high computational requirements. The authors reported a linear relationship between the number of songs in the reference database and the time to search query audio from the database.

The limitation of image processing-based audio fingerprinting systems [15], [17], [21] is that they require a lot of computing power both in time and storage due to computational complexity of the algorithms employed in the image processing.

3.2 Philips approach

This approach to audio fingerprinting is first proposed by Haitsma and Kalker [1]. The main idea behind this approach is using the energy difference of segmented audio frames and frequency bands as invariant audio feature. The task of audio fingerprinting is separated into two key steps: fingerprint extraction and database searching. In the fingerprint extraction stage the input audio is first down sampled to a mono audio stream with a sampling rate of 5kHz. The data is then segmented into 370-millisecond overlapping frames that are weighted using a Hanning window with a 31/32 overlap factor. Every 11.6 milliseconds, one sub-fingerprint, which is a compact representation of a frame of audio, is generated. One fingerprint block is made up of 256 consecutive

sub-fingerprints. Each fingerprint block represents three seconds of audio and can distinguish one audio from another.

Each sub-fingerprint has 32 bit value. In order to calculate this value each audio frame is transformed into frequency domain using Fourier transform and taking only the power spectral since human auditory system is relatively insensitive to phase. The frequency component of the spectrum is sub-divided into 33 non-overlapping frequency bands. Logarithmic spacing is used to arrange the bands from 300Hz to 2000Hz. The use of logarithmic spacing was chosen because the human auditory system operates on a logarithmic scale [1]. Finally, the sign of the energy difference between these frequency bands in neighboring frames is utilized to populate the values of the 32 bit sub-fingerprint, which is empirically proved to be a robust audio characteristic[1] .

The reference database stores the sub-fingerprints created during the fingerprint extraction stage. The saved sub-fingerprints will be organized into reference fingerprint blocks and compared to the query fingerprint blocks in the database searching process. The reference fingerprint block that matches the query fingerprint block and has a lower bit error rate will be chosen as a candidate audio match [1].

Haitsma and Kalker [1] developed a robust audio fingerprinting system that can operate effectively even in the presence of various audio modifications using the aforementioned two steps. However, their work can't recognize pitch shifted audio requests well because sub-fingerprints are extracted from the spectrogram between a fixed starting point and an ending point in frequency. The reason for this is that pitch-shift creates a shift in the frequency component of the spectrogram, resulting in a mismatch in the corresponding sub-fingerprints of the audio with and without pitch-shift, substantially lowering the database searching stage's hit rate.

Yao et al. [23] extends the work of Haitsma and Kalker [1] by proposing a novel sampling and counting retrieval method to improve the query time while keeping the hit rate high. Their work reported a significant memory size reduction by using a new inverted index structure. The authors reported recall rates of over 99 percent for all query audios, including audios with a length of less than 6 seconds. However, their work didn't include query audios with audio modifications such as pitch shifting, time-stretching, and speed-up.

Yao et al. [24] further extend their previous work [23] by handling time-stretching. They proposed a turning point alignment algorithm to enhance the sampling and

counting method introduced in their earlier work [23]. Using this new method the authors achieved a recall value of above 80 percent to time-stretch from 70 percent to 130 percent. However, their work didn't report robustness to pitch shifting.

Chu et al. [25] proposed a peak-point based energy bands computation method (PPEB) to enhance the recall rate of pitch shifted audios in their previous work [24]. The authors proposed that the index of energy bands, which is pitch-shift invariant, be computed by dynamically picking the peak points of each frame among low frequency bins as the starting points. The authors presented a method for computing the energy bands for retaining consecutive sub-fingerprints based on peak points. Although, their work achieved above 80 percent average precision values for audios with pitch shift from -30 percent to +30 percent, their work only achieved average recall rate of above 80 percent for audios with pitch shift percent from -5 percent to +5 percent.

3.3 Shazam Approach

The Shazam-based audio fingerprinting approach is based on the baseline work introduced by Wang [10]. Wang [10] used peak points (points with high energy compared to neighbors) in spectrogram generated with Short-time Fourier transform (STFT) and their associations as robust fingerprints. The authors chose peak points due to temporal locality, translation-invariant, robustness, and sufficiently entropic nature of the peak points and their carefully calculated associations. The temporal locality nature of peak points originates from the fact that these peak points are selected from the small frame of audio samples in time. This nature will enable the fingerprints to be resistant to changes of distant audio samples. Translation-invariant feature of the fingerprints ensures that a query audio results in match irrespective of its position in the original audio track. The fingerprints' robustness assures that fingerprints formed from an unaltered copy of an audio track can be reproduced from a modified, degraded, noisy copy of the same audio. Moreover, the entropy of the fingerprint tokens should be high enough to limit the possibility of false token matches between the query sample and database tracks at non-corresponding locations. In the presence of noise and distortion, insufficient entropy leads to excessive and spurious matches at non-corresponding regions, necessitating additional processing resources to filter the results, while too much entropy usually leads to fragility and non-reproducibility of fingerprint tokens.

The authors chose spectrogram peaks since they most likely survive signal distortion and degradation. Using these spectrogram peaks directly in the fingerprint hash, however, yields low entropy. To avoid this the authors combinatorially associate a pair

of peak points using an anchor peak point and corresponding target zone of peaks. From the pair of peak points, two frequency components namely the first peaks frequency component and the second peaks frequency component are stored directly in the hash. The time difference between the two peaks time components is also stored in the hash. The authors claimed hashes generated this way are quite reproducible even in the presence of noise and various audio compressions. Wang [10] manged to achieve significant robustness to noise, audio compressions while keeping the computational requirements low. However, due to using frequency components of the pair of peaks directly in the hash, this work can't handle query audios with even minor pitch shifting.

Fenet et al [2] extends the original work [10] to specifically address pitch shifting. The authors used Constant Q Transform (CQT) [8] instead of STFT to generate the spectrogram. The reason the authors chose CQT over STFT is that pitch shifting turns into translation instead of scaling in CQT. This property of CQT help the authors to alleviate the problem of storing direct frequency component of the pair of spectral peaks[10]. Relying on this fact, the authors proposed a new fingerprint hashing scheme where they stored the time and frequency difference of the two spectral peaks in the hash. In order to compensate the entropy loss due to storing the time and frequency deltas in the hash, the authors introduced a third parameter in the hash which holds the rough frequency location of the the first peak. Using this hashing scheme the authors able to successfully identifies audio up to 5 percent of pitch shifts. However, due to the introduction of the third parameter , which still depends directly on the frequency components of the first peak, their work failed for pitch shifts more than 5 percent.

Six and Leman [3] extends the work of Fenet et al [2] by using three spectral peaks instead of two peaks. The choice of using three spectral peaks was adopted from Arzt et al.[26]. The authors leverage the power of CQT which make pitch shifting translation. The authors proposed a new hashing scheme based on the work of Fenet et al.[2]. Their hashing scheme takes advantage of the three peak points. The authors store the ratio of the time difference between the first and second peaks to the time difference between the first and last peaks. This ratio enables them to achieve time-scale invariant fingerprints. The authors handled pitch shifting by storing two frequency deltas. The first delta is between the frequency components of the first and second peaks, whereas the second one is between the frequency components of the second and last peaks. To increase the overall hash entropy, the authors introduced two additional parameters. These parameters are sufficiently coarse locations of the frequency components of the

first and third peaks. Overall the authors were able to produce a highly scalable fingerprinting system that can identify songs from 30,000 full-length songs. They manged to handle query audios with pitch shifting up to 6 percents.

Sonnleitner and Widmer [11] further increases the number of peak points to four by adapting findings from the field of blind astrometry. The authors define simple and efficiently representable characteristic feature combinations called quads. To generate these qauds the authors first sample an audio to one-channel at 16 kHz. Then they compute the STFT magnitude spectrogram using a Hann-window of size 2048 samples (128 ms) and a hop size of 64 samples (4 ms). From the spectrogram, peak spectral points are extracted. The extracted spectral peaks are associated by picking an anchor peak and combinatorially associating this anchor peak with three other peaks from its target zone. From each quad peaks a corresponding translation- and scale-invariant hash is generated. Using these translation and scale invariant hashes , the authors managed to fingerprint audios with ±31 percent pitch shift. However, Since they use quads to store the fingerprints, their algorithm suffers from computational and storage inefficiencies [25]. Chu et al.[25] reported that their peak-based Philips fingerprint achieved an average retrieval time of 700ms for pitch-shifted audios while quad-based fingerprint by Sonnleitner and Widmer [11] took 1690ms. Whereas, the proposed approach took 425 ms. Furthermore, Chu et al. [25] reported that quad based fingerprint by Sonnleitner and Widmer [11] is also not robust to GSM compression.

Gebie [27] employed a triple-point geometric hashing algorithm to handle linear speed modification. The authors chose STFT to generate a spectrogram of the time domain audio. From the STFT spectrogram, the authors selected localized triple of spectral peaks. These triple spectral peaks are organized into fingerprint hashes using a geometric hashing algorithm. Geometric hashing technique insensitiveness to transformations such as scaling helped the authors to generate time-scale invariant fingerprint hashes. The authors reported robustness to linear speed change in a range from -30% to 22%. However, due to their use of STFT, the authors could not handle a pitch-shift modification of more than 4%.

Chapter 4

Proposed Approach

The general architecture of the proposed approach is presented in Figure 4.1. The architecture is adapted from [3] and it consists of four major components:

- Feature Extraction
- Fingerprint Construction
- Fingerprint Storage
- Matching

Feature extraction component is responsible for pre-processing the incoming audio, transforming the pre-processed audio into spectrogram , and extracting spectral peaks from the spectrogram. The fingerprint construction component organizes the extracted spectral peaks into fingerprints. Then, if the incoming audio is a reference audio, the fingerprint storage component will be tasked to store the fingerprints into a database alongside the audio's meta data such as audio id, artist and year. Upon an unknown audio query request the matching component do a search on the stored fingerprints for incoming unknown audio query fingerprints. If the match is found in the database the associated meta data will be returned. Otherwise no match is reported. The following sections describe how these components operate in detail.

4.1 Feature Extraction

Here a given query or reference audio is first pre-processed. Then the pre-processed audio sample is transformed into the time-frequency domain using the Constant-Q transform. Points with local maximum energy are then extracted from the transform. Figure 4.2 depicts different stages of the proposed feature extraction component. The following sub-sections will discuss these stages in detail.

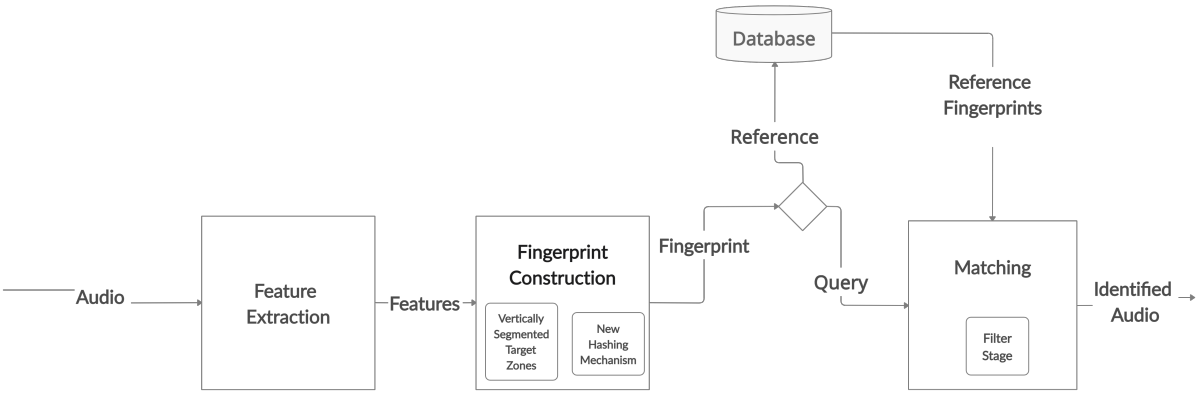

FIGURE 4.1: Architecture of the proposed approach

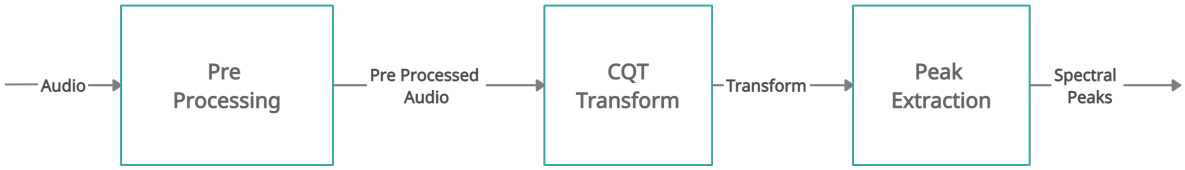

FIGURE 4.2: Stages of feature extraction

4.1.1 Pre-Processing

Pre-processing is required since the audio presented to the audio fingerprinting system can be of different formats. The first pre-processing applied to incoming audio is decoding. Here the original audio coding format such as mp3, Free Lossless Audio Codec (FLAC) and Waveform Audio File Format (WAV) are decoded into 16 bit signed Pulse Code Modulation (PCM). The original audio is also down-mixed to one-channel monaural representations at sampling rate of 44.1 kHz. The audio processing tool we used for pre-processing is ffmpeg [28].

4.1.2 CQT Transform

The row mono channel PCM audio from the pre-processing stage undergoes thru a constant Q transforms (CQT). CQT is chosen over STFT due to the following advantages:

- CQT is widely used for analysis of music signals since the frequency bins in CQT are geometrically spaced resembling how human auditory system perceives sound.
- The constant Q transform produces a constant number of bins for each note or octave [8]

- In the CQT domain, pitch shifting becomes a vertical translation along the frequency axis. In other words, the frequency content found in bin b1 will have a pitch shifted version in bin b1 + K [2].
- CQT solves the problem of fixed time-frequency resolution in STFT by allowing a high spectral resolution at low frequencies and high temporal resolution at high frequencies [8].

As indicated above CQT assigns a constant number of bins for each octave. We used 36 bins per each octave for this work. Increasing the number of bins per octave improves the frequency resolution at the same time increasing the number of bits required to represent each bin. Thus, we picked 36 [3]. The minimum and maximum frequency used to calculate the CQT are 69.2Hz and 9396.35 Hz respectively. Based on these parameters the total number of bins needed will be 255 which can be represented by 8 bits. Equation 4.1 shows the formula used to calculate this.

$$number\_of\_bins = bins\_per\_octave * log_2(max\_frequency / min\_frequency) \tag{4.1}$$

The frequency resolution of each bin is given by cents which is a unit of measure for the ratio between two frequencies. Since 36 bins are used to represent each octave and each octave is 1200 cents[29], the frequency resolution of this work is 33 cents(see Equation 4.2).

$$frequency\_resolution = \frac{1200\,cent/octave}{36\,bin/ocatve} = 33\,cent/bin \tag{4.2}$$

Figure 4.3 shows the CQT spectrogram of a song called La fraternitÃ© by Mob computed using minimum and maximum frequency of 69.2Hz and 9396.35 Hz respectively, and 36 bins per octave. Yellow color indicates regions with high magnitude whereas regions with dark color represent lower magnitudes as shown in the color bar at the right side of Figure 4.3. The reference used to compute the decibels in the color bar is the global maximum magnitude found in the spectrogram. The decibels are calculated using Equation 4.3. For example -20db value represents 10 percent of the maximum magnitude in the spectrogram.

$$db = 20 * log_{10}(\frac{magnitude}{max\_magnitude}) \tag{4.3}$$

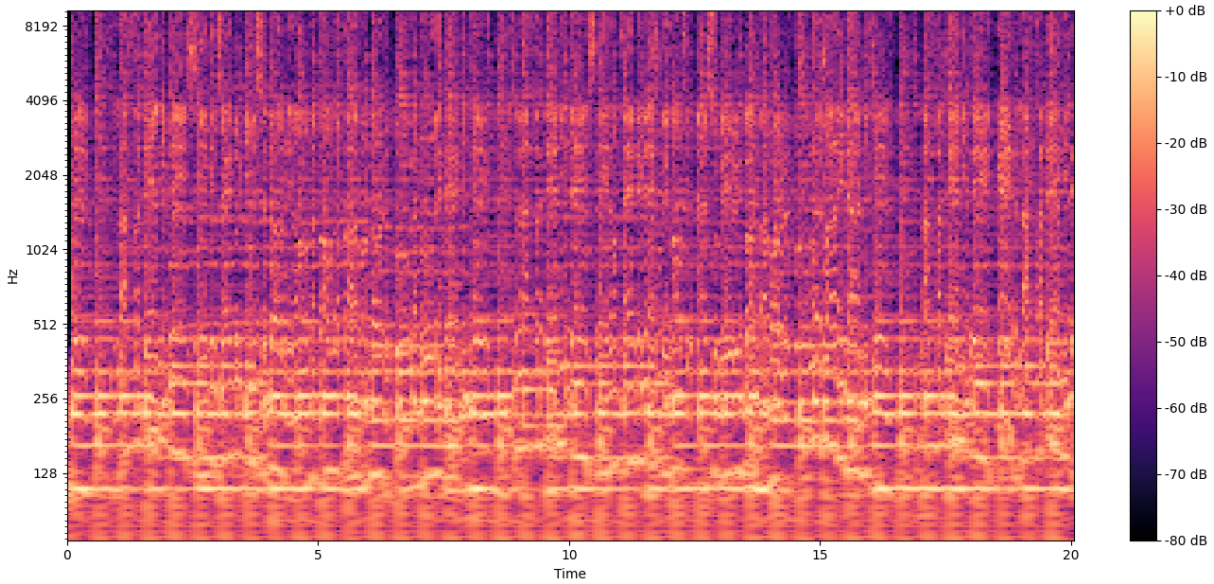

FIGURE 4.3: Spectrogram of a song called La fraternitÃ© by Mob: the x-axis represents tempo and the y-axis represents pitch

4.1.3 Peak Extraction

In this stage event points with local maximum energy, i.e peaks, are extracted from the CQT generated in the previous stage. A tiled two-dimensional peak picking algorithm, which is adapted from previous studies [3], [8], [10], is used to extract peaks from the CQT. The first step in this algorithm is to identify local maxima for each spectral analysis frame. The reason the local maxima is used instead of global maxima is to minimize the effect of distant events on the generated fingerprint [10]. A rectangular tile with dimensions of T and F is then generated for each local maxima points taking their location as the center of the tile. If the local maxima also happens to be the tile's maximum, it's maintained; otherwise, it's eliminated. This will make sure that only one peak is extracted for every tile. The values of T and F are chosen experimentally to allow extraction of 24 up to 60 peaks from one second of audio. Figure 4.4 indicates the spectral peaks , shown as green dots, extracted using the tiled two-dimensional peak picking algorithm described above.

4.2 Fingerprint Construction

Fingerprint construction component deals with creating fingerprints out of extracted spectral peaks. This component is further divided into three main stages namely: target zone assignment, triplet formation and, fingerprint hashing. Target zone assignment stage is responsible for assigning a target zone for the peak under consideration, i.e root peak. Target zone is a rectangular area in spectrogram where a root peak can

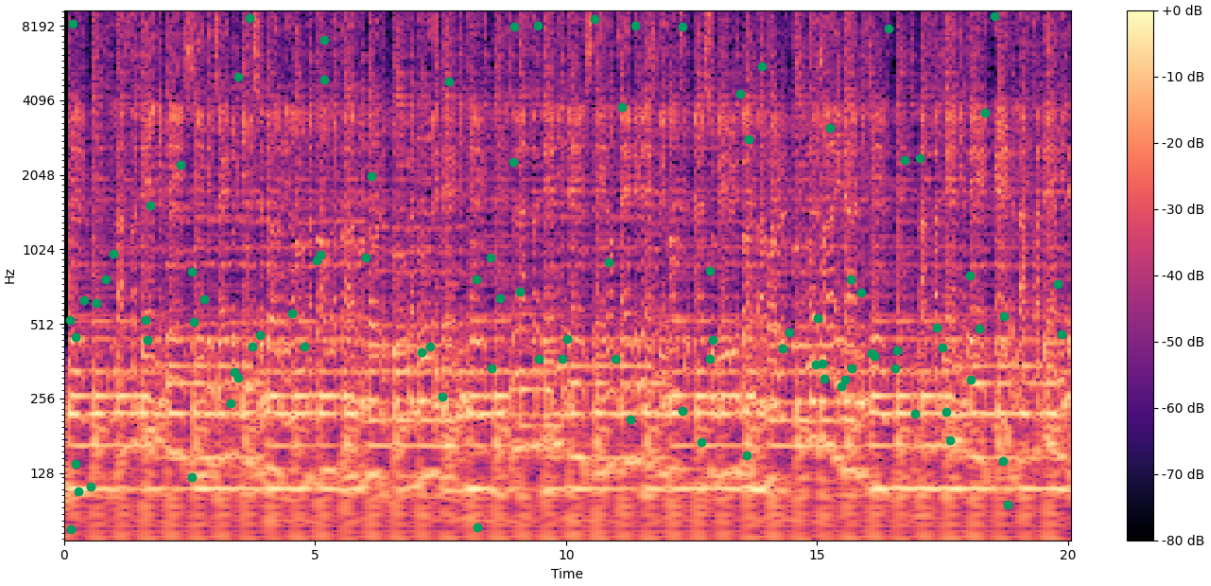

FIGURE 4.4: Spectral Peaks, shown as green dots, generated from spectrogram of a song called La fraternitÃ© by Mob

look for other peaks to form a triplet. Triplet formation block is responsible for creating triples of peaks by associating peaks in the target zone with the root peak based on the selection criteria. For each triplet peaks, the fingerprint hashing stage generates fingerprint hashes based on the proposed hashing scheme described in section 4.2.2. Figure 4.5 shows these stages and their relationships.

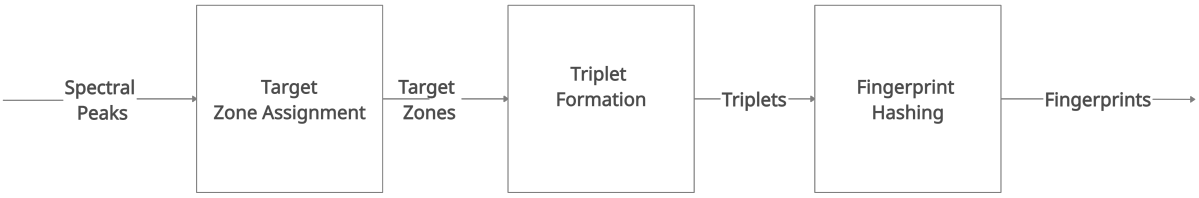

FIGURE 4.5: Stages of Fingerprint Construction

Before discussing the details of these stages, lets first see the effect of pitch shifting on the extracted spectral peaks. Pitch shifting affects the frequency components of the original audio without touching the time components. In Figure 4.6 and Figure 4.7 the yellow dots represent the peaks of +10 percent and -10 percent pitch shifted version of the original audio respectively while the green dots represent peaks of the original unmodified audio.

An audio is said to be pitch shifted by +10 percent when all of its frequencies increased by 10 percent compared to the original unmodified version. In other words

the frequency component of the +10 percent pitch shifted audio is equal to 1.1 times the original audio's frequency component. Similarly, -10 percent pitch shift decreases the frequency component of the the original audio by 10 percent. In other words, frequency component of the -10 percent pitch shifted audio is equal to 0.9 times the original audio frequency component.

The use of CQT as frequency domain transform converts the scaling problem into simple translation. We can see this fact clearly by comparing the positions of the green and yellow dots in the spectrograms shown in figure 4.6 and figure 4.7. Specifically, From figure 4.6, we can see that +10 percent pitch shifting, translates the original unmodified audio peaks upward to higher frequency values, whereas -10 percent pitch shifting translates the original audio peaks downwards to lower frequency values as shown in figure 4.7.

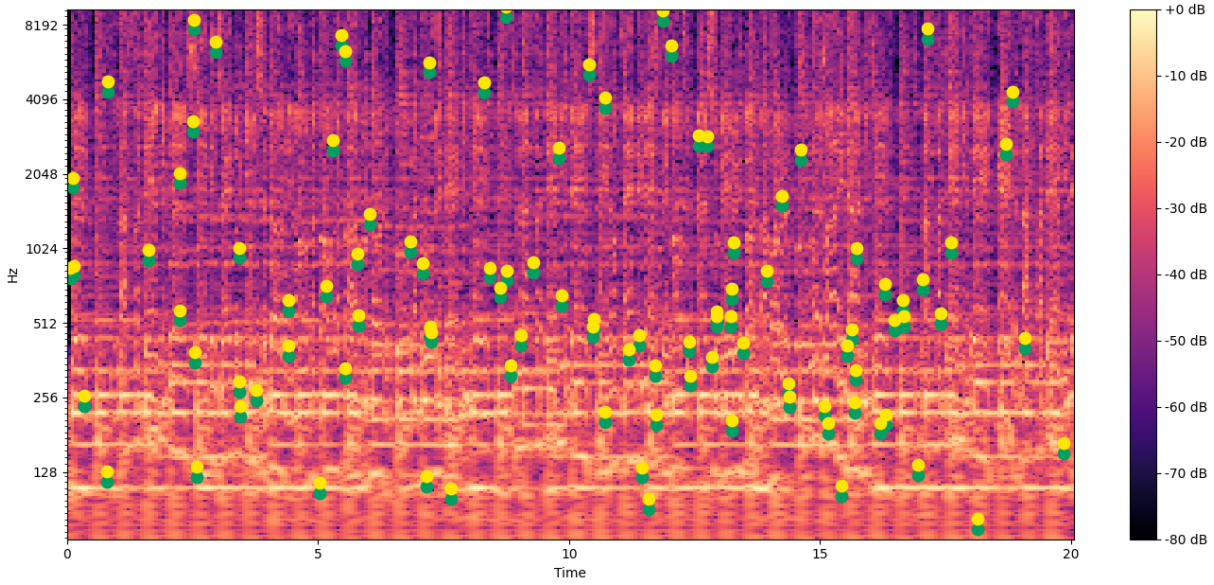

FIGURE 4.6: Spectral peaks of original audio (green dots) vs that of +10 percent pitch shifted audio (yellow dots).

4.2.1 Target Zone Assignment

The first step in the fingerprint construction phase is to organize the extracted local maximum points into triplets. To achieve this task, we propose vertically segmented target zones which are similar to previous works [3], [27]. The target zones utilized in [3], [27] are generated by segmenting the horizontal axis while using the whole vertical axis. Our approach, however, segments the target zones both horizontally, i.e., time-axis, and vertically, i.e., frequency axis. Segmenting the spectrogram in vertical

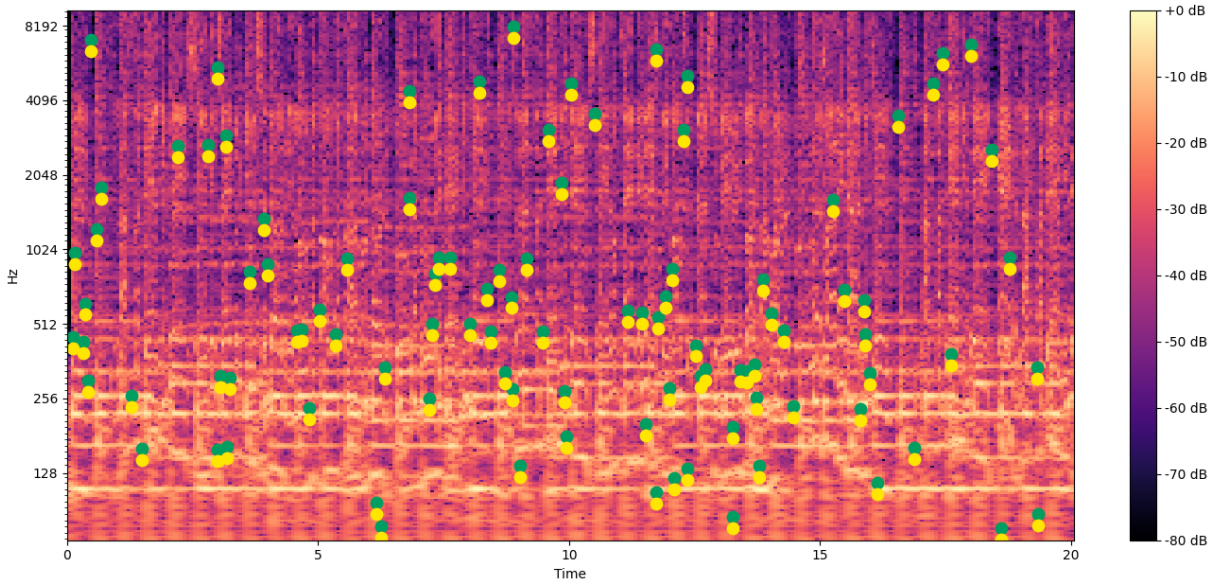

FIGURE 4.7: Spectral peaks of original audio(green dots) vs that of -10 percent pitch shifted audio(yellow dots).

axis increases the locality of the fingerprints by forming triplets from nearby peaks. Increasing the locality of fingerprints in-turn increases their robustness to various audio modification [10]. A sample vertically segmented target zone is shown in Figure 4.8.

The steps followed in the proposed vertically segmented target zone assignment algorithm are listed below.

(A) Select the first peak (i.e root peak): $P_1(t_1, f_1)$
Starting from time zero (far left of the spectrogram) select a peak that will act as root peak. In Figure 4.9 root peak is designated by $P_1(t_1, f_1)$ where t_1 is frame number and f_1 is bin number.

(B) Assign vertically segmented target zone:
Using the selected peak(P_1) in step A, the proposed vertically segmented target zone is calculated as follows. Let us represent the target zone T as a rectangle with dimensions shown in Figure 4.9:

(i) horizontal side ranges between $(t_1 + p - r/2, t_1 + p + r/2)$

(ii) vertical side ranges between $(f_1 - k/2, f_1 + k/2)$

The values of p, r , and k used in this work are 32 time steps(frames), 50 time steps(frames) and 62 bins respectively. These values are determined experimentally using the values suggested in [3], [27] as a starting points.

(C) Triplet Formation: All the peaks, picked two at a time, that fall in the target zone are subjected to the following criteria and the ones that meet the criteria will be

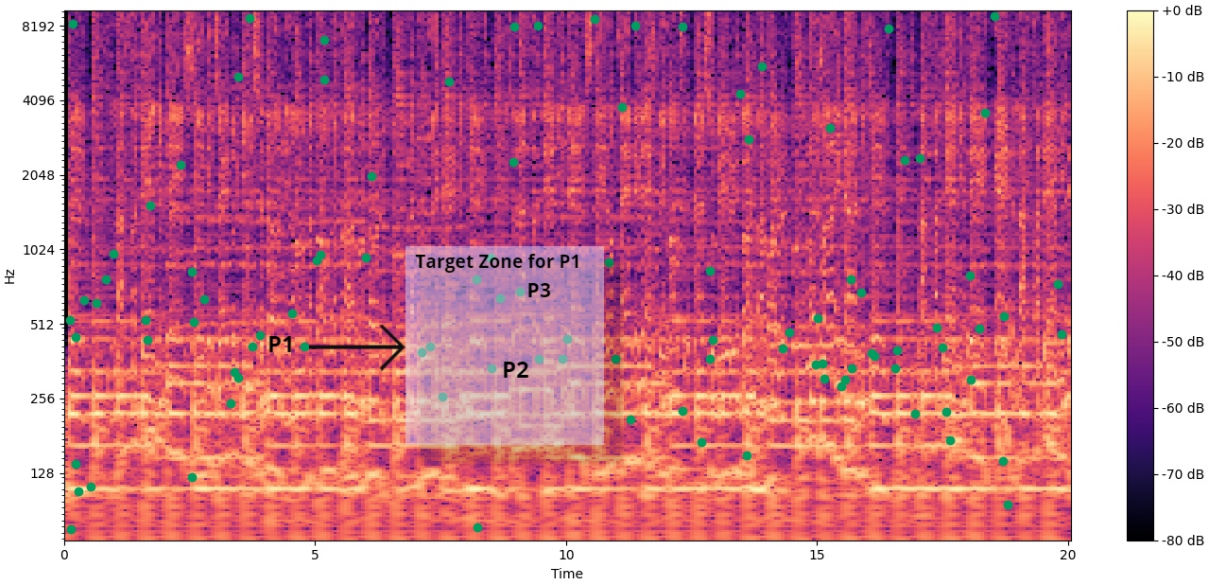

FIGURE 4.8: Spectral Peaks, shown as green dots, with vertically segmented target zone generated from spectrogram of a song called La fraternitÃ© by Mob.

associated with the root peak to form triplets.
The criteria is:- given three peaks: $P_1(t_1, f_1)$, $P_2(t_2, f_2)$ and $P_3(t_3, f_3)$ where $P_1(t_1, f_1)$ is the root peak and $P_2(t_2, f_2)$ and $P_3(t_3, f_3)$ lie inside the target zone.

$$t_1 < t_2 < t_3 \tag{4.4}$$

This criterion states that the first peak's time component must be less than the second peak's time component, which must be less than the third peak's time component. This criterion guarantees that the time deltas between peak2 and peak1, as well as peak3 and peak1, remain positive. As a result, their ratio will always be somewhere between 0 and 1. We used this ratio in the fingerprint hash.

(D) Repeat step A to C for every points from left to right while keeping total count of triplets per a given second from 8 up to 20 fingerprints.

4.2.2 Proposed Hashing Scheme

The next step after the triplets are formed is to construct the fingerprint hash. This work proposes a new hashing scheme based on Panako [3]. Panako's hash is formed from a triplet as follows.
Given:
$P_1(t_1, f_1)$, $P_2(t_2, f_2)$ and $P_3(t_3, f_3)$: triplets from reference audio

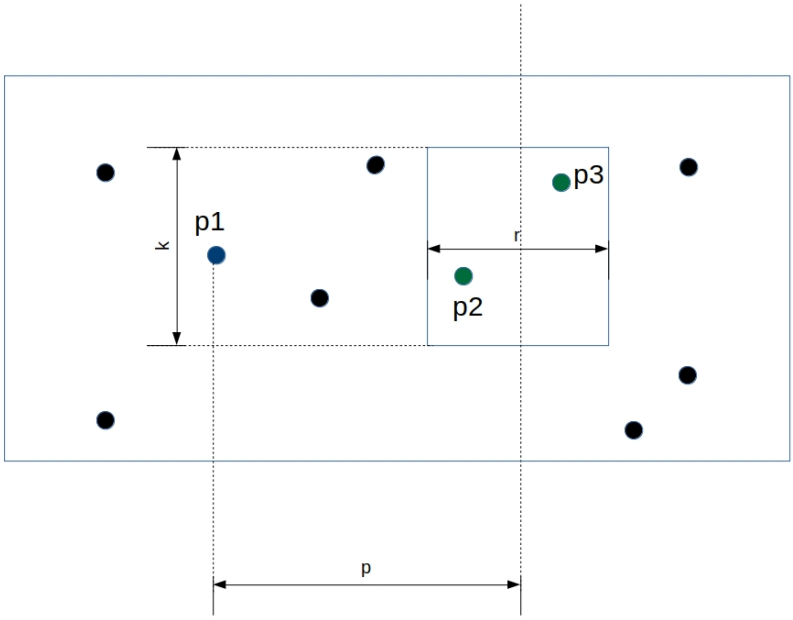

FIGURE 4.9: Proposed vertically segmented target zone.

$P'_1(t'_1, f'_1)$, $P'_2(t'_2, f'_2)$ and $P'_3(t'_3, f'_3)$: triples from query audio

$$Panako\_Hash = (f_1 - f_2; f_2 - f_3; `f_1; `f_3; \frac{(t_2 - t_1)}{(t_3 - t_1)}) \tag{4.5}$$

Since in CQT pitch shifting is a vertical translation, the frequency component of reference peaks will be translated by a constant K when the audio is pitch shifted. The vertical translation could easily be seen in Figures 4.6 and 4.7. In the figures, the green dots in the spectrogram are vertically translated to the yellow dots.
Based on this observation we can say that:

$$\begin{aligned} f'_1 &= f_1 + K \\ f'_2 &= f_2 + K \\ f'_3 &= f_3 + K \end{aligned} \tag{4.6}$$

From Equation 4.6 we can deduce the following equations.

$$\begin{aligned} f_1' - f_2' &= (f_1 + K) - (f_2 + K) \\ f_1' - f_2' &= f_1 - f_2 \end{aligned} \tag{4.7}$$

$$\begin{aligned} f_2' - f_3' &= (f_2 + K) - (f_3 + K) \\ f_2' - f_3' &= f_2 - f_3 \end{aligned} \tag{4.8}$$

From Equations 4.7 and 4.8 we can see that the frequency delta between peaks stays the same for reference audio and its pitch shifted version. Thus, we can use the frequency delta between peaks as a pitch shift invariant audio feature. This leads to inclusion of the deltas in the formulation of the hash in Equation 4.5.

In addition to handling pitch-shifting, this work proposes to enhance the robustness of the proposed approach to time-stretching and linear speed modifications. The following two paragraphs discuss how this work handles these modifications.

When an audio undergoes time-stretching the resulting audio spectrogram will be affected in time component only as shown in Figure 2.3.a and Figure 2.3.b. The frequency component will remain the same [3], [26]. To formulate time-stretch invariant audio hash, we propose to use the ratio of the two time deltas namely: time delta between peak 2 and peak 1 and the time delta between peak 3 and peak 1. As stated in triplet formation step of the proposed target zone algorithm, the criteria stated in Equation 4.4 ensures that these time deltas stay positive and their ratio to always be between 0 and 1. This is shown in Equations 4.9 and 4.10.

$$\begin{aligned} t_2 - t_1 > 0 \; since \; t_2 > t_1 \; from \; Equation \; 4.4 \\ t_3 - t_1 > 0 \; since \; t_3 > t_2 > t_1 \; from \; Equation \; 4.4 \end{aligned} \tag{4.9}$$

Based on Equation 4.9 we can infer that:

$$0 < \frac{(t_2 - t_1)}{(t_3 - t_1)} < 1 \tag{4.10}$$

The effect of speed change is a linear combination of the effect of time-stretching and pitch shifting [3], [26]. We can see this in the spectrogram provided in Figure 2.3.c and Figure 2.3.d. Therefore the linear combination of time-stretching and pitch shifting invariant audio features will automatically give the speed change invariant audio feature[3].

As discussed in the section 1.1, 'f_1 and 'f_3 of the Panako [3] hash add more discriminative power to the hash at the same time limit how much the audio can be pitch-shifted. In order to solve this problem this work proposes to remove these values from the hash. The resulting proposed hash is shown in Equation 4.11.

$$Proposed\_Hash = (f_1 - f_2; f_2 - f_3; \frac{(t_2 - t_1)}{(t_3 - t_1)}) \tag{4.11}$$

4.3 Fingerprint Storage

Before querying the audio fingerprinting system for possible matches, the reference audio should be stored in persistent storage. The hash computed using the proposed Equation 4.11 is stored in the fingerprint storage along with other parameters, discussed below, as a unit of a fingerprint. The frequency component of the reference audio's first peak: f_1 and the time delta between the first and last peak: $t_3 - t_1$ are stored along with the hash. These values are compared with the corresponding values of query audio namely: f_1' and $t_3' - t_1'$ to calculate the amount of pitch shift and time-stretching present in the query audio.

$$Pitch\_Shift\% = e^{((f_1' - f_1) * ln(2)/bin\_per\_octave)} * 100\% \tag{4.12}$$

$$Time\_Stretch\% = \frac{t_3 - t_1}{t_3' - t_1'} * 100\% \tag{4.13}$$

The pitch shift of a query audio exhibited in bins is calculated by simply subtracting the frequency component of the reference audio's first peak, f_1, from the frequency component of the query audio's first peak, f_1' , i.e pitch shift in bins equals to $f_1' - f_1$. Equation 4.12 gives a formula to convert the pitch shift in bins to percent. Here a value greater than 100% indicates that the pitch shift is positive meaning that the original reference audio's pitch is shifted upwards to higher pitch. Whereas, a value less than 100% indicates that the pitch shift is negative, i.e the original reference audio's pitch is shifted down to lower pitch. For example 90% pitch-shift is equivalent to -10% pitch-shift.

The ratio of time delta between the first and third peaks of a reference audio, $t_3 - t_1$, to that of a query audio, $t_3' - t_1'$, indicates the time modification a query audio undergoes [3]. Equation 4.13 is used to calculate the time-stretch of a query audio. Here a value greater than 100 percent indicates the time-stretch is positive and the query audio is compressed (shortened) in time than the reference audio. Whereas, a value less than

100 percent indicates that the time-stretch is negative and the query audio is stretched (lengthen) in time than the reference audio. For example 90 percent time-stretch is equivalent to -10 percent time-stretch.

Another important parameter that should be stored along the hash is the time component of the first peak: t_1. This parameter enables us to calculate the offset at which a match was found for the provided query audio. The complete list of values to store per a fingerprint should, therefore, include the following:

(i) The fingerprint hash: $(f_1 - f_2; f_2 - f_3; \frac{(t_2 - t_1)}{(t_3 - t_1)})$

(ii) f_1

(iii) $t_3 - t_1$

(iv) t_1

(v) Unique audio identifier: *id*

The memory size allocated for each of the values stored per fingerprint has effect on the overall computational requirement of the fingerprint. Hence, we tried to minimize the memory size of each of these values by packing them together. The three parts of the fingerprint hash are packed together into one 32 bit integer. f_1 and $t_3 - t_1$ are also merged into one 32 bit integer. The last two values t_1 and *id* took 32 bit each. Adding these together , the whole fingerprint took 128 bits. Keeping 8 fingerprints per each second of audio, as discussed in the last step of the proposed vertically target zone algorithm, see section 4.2.1, a 4 minute audio will take only 30.7 KB of memory. In this work, a total of 3000 full length mp3 audios were stored in the database taking 187.2 MB of memory.

The database we used to store the fingerprints is MapDB[30]. MapDB uses off-heap or on-disk storage to store basic java collections such as Concurrent Maps, Sets, Lists, Queues and Tuples. MapDB was chosen for its simplicity and efficiency in both space and time, as well as its out-of-the-box and easy integration with java collections.

4.4 Matching

In this stage, the query audio is matched with reference audios. The matching technique employed here is based on the previous works [3], [10], [27]. However, an additional filtering stage is included to accommodate the proposed vertically target zone algorithm and hashing scheme.

The proposed matching algorithm is divided into three main stages namely: fingerprint hash matching, filtering and match alignment stages. Figure 4.10 shows each of these stages. The following sections discuss the details of these stages.

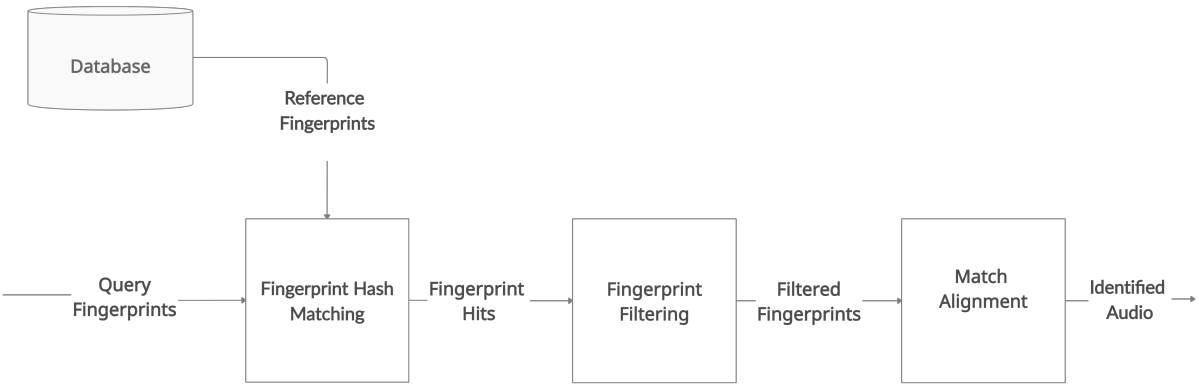

FIGURE 4.10: Stages of fingerprint matching component

4.4.1 Fingerprint Hash Matching

The query audio presented to the fingerprinting system first goes through the feature extraction and fingerprint construction components as shown in Figure 4.1. The result of these components are query audio fingerprints. Given these query fingerprints and reference audio fingerprints from the database, the matching stage iterates query fingerprints and searches for a match among the reference audio fingerprints. For the search to find a match the query fingerprint's hash should be exactly equal with the reference fingerprint's hash. This technique is identical with previous works [3] and [10]. Using exact hash matching technique has computational advantages compared to nearest neighbor search.

However, in the presence of severe audio modifications exact hash matching could lead to two problems. The first problem is the exact hash matching could fail to find matching hashes for query fingerprints among reference fingerprints due to modifications of the original audio peaks. This will lead to false negative results where the system fails to find a match for query audio when there is already a matching reference audio stored in the database. The second problem arises when the exact hash matching found matches for the query fingerprints but most of them are from another audio's fingerprints. This could happen since the entropy or discriminative power of the proposed hash was reduced when exact frequency components ($'f_1; 'f_3$) were removed from the hash (see Equation 4.5 and 4.11). To address these problems, we propose to filter matched fingerprints.

4.4.2 Match Fingerprint Filtering

Given the query fingerprint: $hash, t'_1, f'_1, t'_3 - t'_1$ and reference fingerprint: $hash, t_1, f_1, t_3 - t_1$, the proposed filter makes sure the following conditions are met:

(i) $Tolerance\_lower < t\_mod < Tolerance\_upper$

(ii) $Tolerance\_lower < f\_mod < Tolerance\_upper$

Where:

$$\begin{aligned} t\_mod &= \frac{t_3 - t_1}{t'_3 - t'_1} \\ f\_mod &= e^{((f'_1 - f_1) * ln(2)/bin\_per\_octave)} \end{aligned} \tag{4.14}$$

Tolerance-lower and Tolerance-upper are constants close to one (1) and they are experimentally determined. The values for Tolerance-lower and Tolerance-upper are chosen to be identical since this work aims to handle up to 20% pitch and time modifications. The exact values we used in this work for Tolerance-lower and Tolerance-upper are 0.8 and 1.2, respectively. The first condition filters out reference fingerprints with more than 20 percent time modification. The second condition filters out reference fingerprints with more than 20 percent pitch modification.

4.4.3 Match Alignment

Reference fingerprints that remain after filtering stage are presented to this stage for further alignment checks. Before alignment check is done, how many times each individual audio identifier occurs in the result set is counted. The count is used to eliminate random chance hits. To identify such random hits, we used a threshold value of three. This threshold value is chosen through experiment based on the value used in Panako[3]. Those hits which have a count below three are eliminated. In practice, there is nearly always only one identifier with a large number of matches, with the remainder being chance hits.

The remaining match fingerprints of audio identifiers with count greater-than three are subjected to alignment check both in time and frequency using the values stored along side the hash. We say the query and and reference fingerprints align in frequency if the difference between f_1 component of reference and f'_1 of the query remains constant. Similarly, the time alignment check is carried out using the reference time components: $t_1, t_3 - t_1$ and the query time components: $t'_1, t'_3 - t'_1$ based on Equation 4.15.

For a matching audio time offset, t_o, should be constant[3].

$$t_o = t_1 - (t_1' * (\frac{t_3 - t_1}{t_3' - t_1'})) \tag{4.15}$$

The match alignment algorithm returns the reference audio with most aligned fingerprints both in time and frequency as a match. Additionally, the algorithm also returns the following information with the match.

(i) The time offset at which the query begins in the reference audio using Equation 4.15.

(ii) The time modification the query undergoes using Equation 4.13

(iii) The pitch shift the query undergoes using Equation 4.12.

Chapter 5

Experiment

The datasets, tools, and experimental scenarios employed to evaluate the proposed approach, as well as the results, are discussed in this chapter.

5.1 Dataset Preparation

The reference audio data used in the experiments is gathered from an open source audio dataset, Jamendo [31]. Jamendo is chosen mainly because it has a freely available audio dataset. Furthermore, a number of other works [3], [11], used this dataset. A total of 3000 full length mp3 audios are gathered from this dataset and stored in the reference database. A total of 43200 query excerpts are generated from randomly selected subset of these 3000 audios. These query excerpts are grouped into three groups based on their granularity. The first group contains 20 second audios, the second 40 seconds audios and the last 60 second audios. Each group is subdivided further into five major categories based on the modification type applied on the query audio fragments. The five categories are:

- **Reference query audios:** 200 query audio excerpts are randomly extracted from the reference audio with no modification. These audio excerpts are used to test the proposed system's ability to identify query audios that undergo no modifications.
- **Query audios with flanger, band-pass, chorus, echo, tremolo:** 1000 query audios are generated by applying flanger, band-pass, chorus, echo and tremolo audio effects on randomly selected audios from the reference database. These query audios are used to evaluate the proposed system against the listed common audio effects.
- **Speed up query audios:** 4000 linear speed modified query excerpts are generated by applying linear speed modification with a rate ranging from -20% to +20% with 2% step on randomly selected audios from the reference database.

- **Time stretched query audios:** 4000 time-stretched query excerpts are generated by performing time-stretch modification with a rate ranging from -20% to +20% with 2% step on randomly selected audios from the reference database.
- **Pitch shifted query audios:** 5200 pitch-shifted query excerpts are generated by applying pitch-shifting with the rate ranging from -325 cents (-17%) to +325 cents (+21%) with 25 cents step on randomly selected audios from the reference database.

The query audios are generated using SoX [32]. SoX is a command-line application that can convert several types of computer audio files into other formats. It runs on Windows, Linux, MacOS X, and other platforms. It can also apply numerous effects to sound files. SoX is used to trim the length of the randomly selected reference audio in to the needed granularity, i.e 20 seconds, 40 seconds and 60 seconds. Since any audio in the reference set can be chosen to generate query audio of lengths 20, 40, and 60 seconds, we need to store reference audio with a minimum length of 60 seconds. Thus, we filtered out reference audios with lengths less than 60 seconds.

Identical parameter values are provided to SoX tool for query audio generation as Panako [3] did. For example, to generate query audios with band pass effect a center frequency of 2000 Hz is used. For Chorus effect, the typical parameter values suggested in SoX documentation [32] are used. The gain values of 0.7 and 0.9 are used for gain-in and gain-out respectively. The delay value used is 55 milliseconds whereas the decay and speed are set to 0.4 and 0.25Hz. The modulation signal utilized here is triangular with modulation depth of 2 milliseconds. Similar parameter values as used in Panako [3] are employed for echo too. The gain-in and gain-out values used are 0.8 and 0.9 respectively. The delay and decay values are respectively set to 500 milliseconds and 0.3. A tremolo effect (low-frequency amplitude modulation) is applied to the audio using a speed parameter value of 8 Hz and the default depth parameter value of 40. The default parameter values of SoX flanger tool are used to generate query audios with flanger effect.

The speed up (linear speed modified) query audios are extracted from the reference audios using SoX speed tool. The only parameter provided to this tool is factor which is the ratio of the new speed to the old speed: greater than 1 speeds up, less than 1 slows down. Factor greater than 1 corresponds to modification rate greater than 0 percent. Similarly, factor less than 1 results in negative modification rate. For example to achieve +20 percent speed up a factor of 1.2 is used and for -20 percent speed up a factor of 0.8 is provided. For time stretched query audios, SoX tempo tool is utilized with a factor parameter set to values starting from 0.8 to 1.2 in the interval of 0.2. This

configuration produces query audios with time-stretch of -20 percent to +20 percent with 2 percent step.

Pitch shifted query audios are generated using the SoX pitch tool, which accepts a shift parameter to decide the amount of pitch shift applied. The shift parameter is provided to this tool as positive or negative cents [14] starting from -325 cents to +325 cents. Positive cent values result in the pitch shift of the original audio to higher pitch values, whereas negative cent values result in a pitch shift of the original audio to a lower pitch. The formula given in Equation 5.1 is used to convert the pitch shift in cents to percentages. A pitch shift of more than 100 percent shows that the pitch shift is positive, implying that the original reference audio's pitch has been pushed upwards to a higher pitch. A value of less than 100 percent implies a negative pitch shift, i.e. the original reference audio's pitch has been shifted down to a lower pitch. A 90 percent pitch-shift, for example, is the same as a -10 percent pitch-shift and a 120 percent pitch-shift is identical to a 20% pitch-shift.

$$modification\% = 2^{modification\_cents/1200} * 100\% \tag{5.1}$$

5.2 Experimental Setup

The proposed approach is implemented using Java 1.8. The digital signal analysis functionalities are carried out using, Tarsos-DSP, a java DSP library [33]. The reference hashes and their corresponding information are stored in a key-value store called MapDB [30]. MapDB implements basic collections backed by off-heap or on-disk storage. MapDB is chosen for its out of the box and easy integration with java collections, and for its simplicity and efficiency both in space and time. The hardware and software specifications used for the experiments are listed in Table 5.1.

	Specifications
Manufacturer	HP
Model	HP ProBook 440 G4
Processor	Intel® Core™ i5-7200U CPU @ 2.50GHz
Memory	8GB SODIMM DDR4 2133 MHz (0.5 ns)
Operating System	Ubuntu Linux 20.04 LTS (Focal Fossa)

TABLE 5.1: Hardware and software specifications

5.3 Experimental Scenarios

Several experiments were carried out to test the proposed approach. These experiments are grouped into four main groups. The first two groups evaluate the robustness and granularity of the proposed system against pitch-shifted query audios. The last two groups of experiments assess the robustness and granularity of the proposed approach to other audio modifications and effects. The groups are listed as follows:

- **Experiment Group 1:** includes experiments that evaluate the robustness of the proposed system against pitch-shifted query audios.
- **Experiment Group 2:** includes experiments that evaluate the granularity of the proposed system against pitch-shifted query audios.
- **Experiment Group 3:** includes experiments that evaluate the robustness of the proposed approach to other audio modifications and effects.
- **Experiment Group 4:** includes experiments that evaluate the granularity of the proposed approach to other audio modifications and effects

In these experiments, the baselines used as a reference are Panako [3] and Quad [11]. Panako [3] is chosen as a baseline since it handles pitch-shifting up to 6% with less computational overhead. The other reason we pick Panako [3] is its use of CQT as frequency-domain transform. CQT reduces the scaling effect of pitch-shifting to simple translation, thus allowing frequency deltas between the peaks to be used as a pitch-shift invariant audio feature. Panako's implementation by Joren Six[34] is utilized. Quad [11] is selected due to its capacity to handle pitch shift up to 31%. We used implementation of Quad by Michael Bortnyck et al.[35].

5.3.1 Experiment Group 1: Robustness against Pitch-shifting

Experiments in this group are designed to evaluate the proposed approach against pitch-shifting. For this group of experiments 3000 reference audios collected from Jamendo [31] are fingerprinted and stored in the reference MapDB database.

The proposed approach is evaluated against 5200 pitch-shifted query excerpts. The length of these query excerpts is 60 seconds. Pitch-shift modification rate ranging from -325 cents (-17%) to +325 cents (+21%) with 25 cents step size are utilized. The same queries are presented to Panako [3] and Quad [35]. Table 5.2 presents query audio size used for this experiment.

	Pitch-shifted query excerpts						
	-17%(-325 cents)	-16%(-300 cents)	..	0%	...	19%(300 cents)	21% (325 cents)
Panako	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries
Proposed	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries
Quad	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries

TABLE 5.2: Pitch-shifted query excerpts for audio length of 60 seconds

	Query Audio length	20 seconds	40 seconds	60 seconds
Audio Effects	Panako	1000	1000	1000
	Proposed	1000	1000	1000
	Quad	1000	1000	1000
Speed up	Panako	4000	4000	4000
	Proposed	4000	4000	4000
	Quad	4000	4000	4000
Time-stretch	Panako	4000	4000	4000
	Proposed	4000	4000	4000
	Quad	4000	4000	4000
Pitch-shift	Panako	5200	5200	5200
	Proposed	5200	5200	5200
	Quad	5200	5200	5200

TABLE 5.3: Query excerpts for audio length of 20, 40, and 60 seconds.

5.3.2 Experiment Group 2: Granularity: Pitch-shifting

Experiments in this group are designed to evaluate effect of query audio size, i.e., granularity, on the accuracy value of the proposed approach for pitch-shifted query excerpts. For this experiment 15,600 pitch-shifted query audio excerpts with length of 20, 40 and 60 seconds are used with the rate ranging from -325 cents (-17%) to +325 cents (+21%) with 25 cents step size. The same queries are presented to Panako [3] and Quad [35]. Table 5.3 presents query audio size used for this experiment.

5.3.3 Experiment Group 3: Robustness against Other Audio Modifications

Experiments in this group are designed to evaluate the proposed approach against common audio effects and modifications. This group is further organized into four subgroups of experiments as follow:

- **Experiment 3.a: Robustness against Audio Effects**
  The proposed approach is evaluated against query audios with flanger, band-pass, chorus, echo, tremolo. A total of 1000 query audio excerpts with length of 60 seconds, 200 for each audio effect, are used (see Table 5.4). The same queries are presented to Panako [3] and Quad [35].

Audio Effects	Panako (queries)	Proposed(queries)	Quad(queries)
Flanger	200	200	200
Echo	200	200	200
Tremolo	200	200	200
Chorus	200	200	200
Band-pass(2KHz)	200	200	200

TABLE 5.4: Audio effect and query size used for audio length of 60 seconds

- **Experiment 3.b: Robustness against Linear Speed Change**
  The proposed approach is evaluated against linear speed modified query excerpts. For this experiment, 4000 linear speed modified query excerpts are used for modification rate starting from -20% to +20% with 2% step size. The same queries are presented to Panako[3] and Quad [35]. Table 5.5 presents query audio size used for this experiment.

	Linear speed modified query excerpts						
	-20%	-18%	..	0%	...	18%	20%
Panako	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries
Proposed	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries
Quad	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries

TABLE 5.5: Linear speed modified query excerpts for audio length of 60 seconds

- **Experiment 3.c: Robustness against Time-Stretch**
  The proposed approach is evaluated against time stretched query excerpts. For this experiment, 4000 time-stretched query excerpts with a length of 60 seconds are used for modification rate starting from -20% to +20% with 2% step size. The same queries are presented to Panako [3] and Quad [35]. Table 5.6. presents query audio size used for this experiment.

	Time stretched query excerpts						
	-20%	-18%	..	0%	...	18%	20%
Panako	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries
Proposed	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries
Quad	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries	200 queries

TABLE 5.6: Time stretched query excerpts for audio length of 60 seconds

5.3.4 Experiment Group 4: Granularity: Other Audio Modifications

Experiments included in experiment group 3 are carried out again for 20 second, and 40 second query excerpts to evaluate the granularity of the proposed approach. The time complexity of each experiment is also recorded. Table 5.3 provides the summary of queries used for this experiment.

5.4 Evaluation Metrics

We used a confusion matrix to compute the evaluation metrics. A confusion matrix has four parts: true positives, true negatives, false positives and false negatives. True positive (TP) is where a correct reference is identified by the query. True negative (TN) is where the system says no match for a query because there is no correct reference that matches in the fingerprint database. False positive (FP) refers to cases where the system suggests wrong reference for a query. False negative (FN) refers to cases where the system fails to return result for a query whose matching reference is in the database. Precision and Accuracy (recognition rate) are calculated based on these metrics.

Accuracy indicates the proportion of actual positives (i.e query audios that are in the reference) that are correctly identified. If the system produces no false negatives and no false positives, it will have a accuracy (recognition rate) of one (1).

$$Accuracy = \frac{TP}{TP + FN + FP} \tag{5.2}$$

Precision is given as the proportion of positive identifications which are actually correct. It indicates that out of the total positive results the system gives, how many are actually correct matches. If the there are no false positives the system will have precision of 1.

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

5.5 Results

5.5.1 Experiment Group 1: Robustness Against Pitch-shifting

In this section the results of the experiments conducted to evaluate the proposed approach against pitch-shifting are presented. In this experiment, 5200 pitch shifted query audios with length equal to 60 seconds are presented the proposed system, Panako [3] and Quad [11]. The result, given in Figure 5.1, indicates that the proposed system achieves an accuracy of 80 percent and above for modification values between -11% and +12% except for modification values around specific instances, such as -8, -3, +3 and +9 percents. The reason for the drop in accuracy values around this modifications will be discussed shortly. In contrast Panako achieved an accuracy value of 80 percent and above for modification values starting from -6% to +6%. Quad achieved the same accuracy level for modification values starting from -12% to +7%. Both Panako and proposed work suffer a drop in accuracy at specific pitch-shift modification values, such as -13, -8, -3, 3, 9 and 16 percents. Quad, on the other hand, exhibits no such drops.

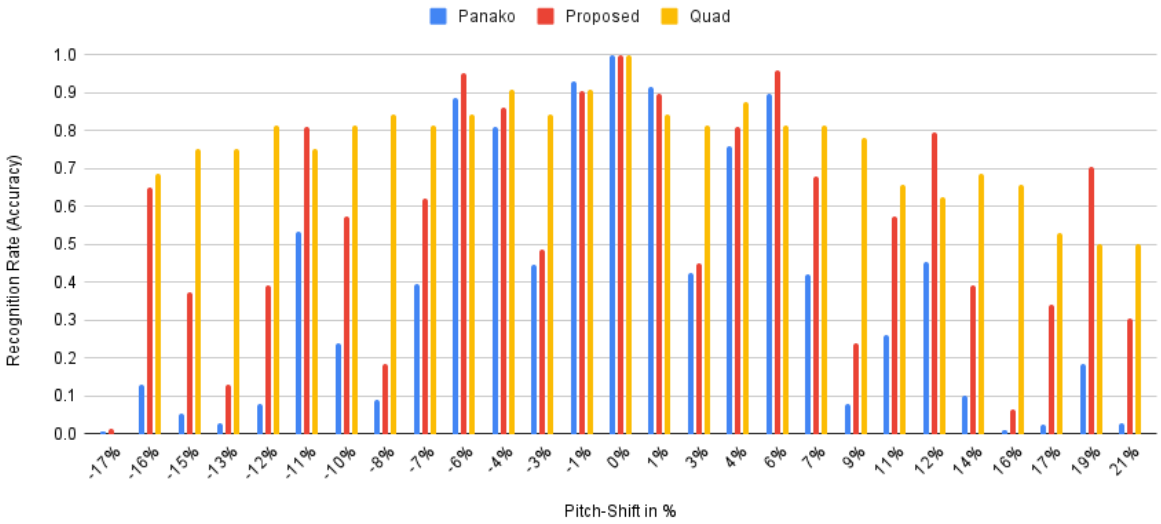

FIGURE 5.1: Accuracy of Proposed, Panako and Quad for pitch shifted query fragments

$$92\% = 2^{-148.5/1200} * 100\% \tag{5.4}$$

The reason for drop in accuracy value on these modification values can be explained

by considering the effect of Constant-Q transform bins. Since there are 1200 cents in 1 octave and we used 36 bins to represent 1 octave, the frequency resolution will be 33 cents per bin. Thus, when an audio undergoes a pitch shift of 33 cents, all of its spectral peaks move to the next bin. Similarly when it undergoes a pitch-shift of 33/2 (16.5) cents, upper half of the peaks move to next bin while the lower half remains intact. This will make the frequency delta in bins for lower half of peaks to be zero while the frequency delta for upper half of peaks to be one. Since the fingerprint hashes directly depend on the frequency deltas (see Equation 4.11), the hashes of the reference and pitch shifted version of the audio will vary significantly. This, inevitably, results in a drop of accuracy value. We can generalize this for other pitch-shift values, by considering that maximum number of peaks spread to two bins occur for pitch-shift values of half of the frequency resolution, which is 33 cents, and its odd multiples. Therefore, when there is pitch-shift of n * 33/ 2 cents for odd values of n, the peak event points will spread over two bins. The pitch-shift step size we used in this experiment is 25 cents, therefore we encountered this scenario for n values:

$$n = -15, -9, -3, +3, +9, +15 \tag{5.5}$$

The corresponding cent values are:

$$c = -247.5, -148.5, -49.5, 49.5, 148.5, 247.5 \tag{5.6}$$

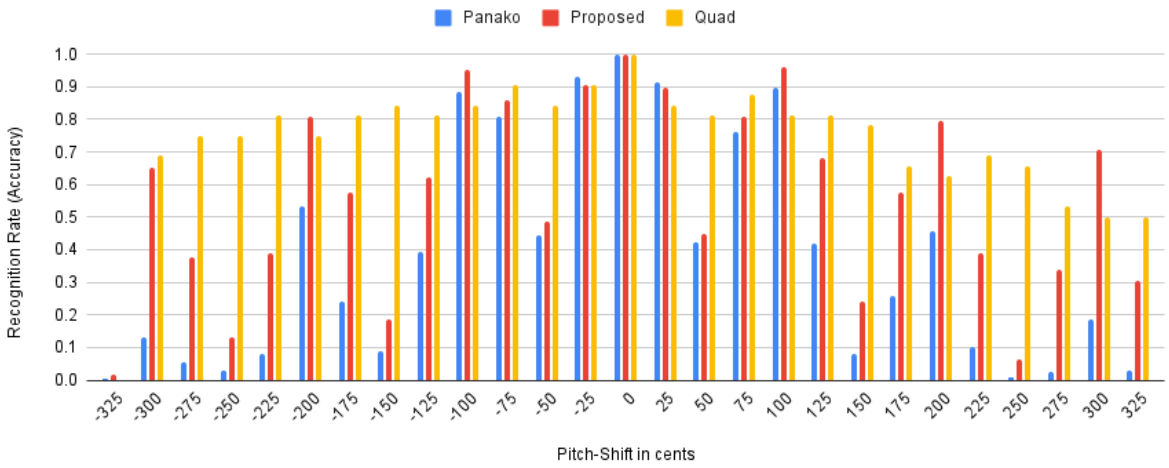

FIGURE 5.2: Accuracy of Panako, Proposed and Quad for 60s pitch-shifted query audios

This is reflected on the accuracy values of both proposed approach and Panako at cent values listed in Equation 5.6 as shown on Figure 5.2. The drop in accuracy value in Figure 5.1 corresponds to the cent values listed in Equation 5.6. Use Equation 5.1 to

convert pitch shift in cents to percentages. For example, the drop in accuracy value at -8% pitch-shift, which is equivalent to (92%) as discussed in data collection and preparation section, corresponds to cent value of -148.5. This is evident from Equation 5.4.

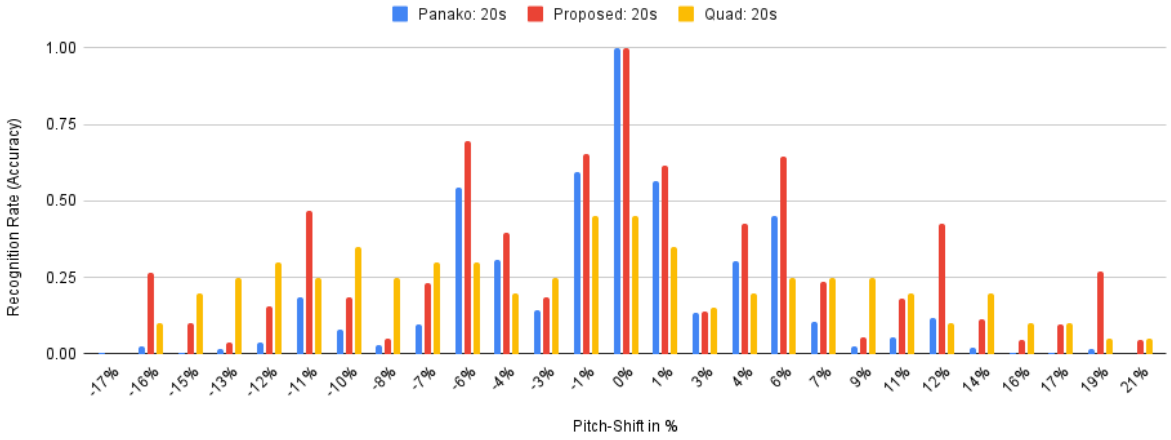

FIGURE 5.3: Accuracy of Panako, Proposed and Quad for 20s pitch-shifted query audios

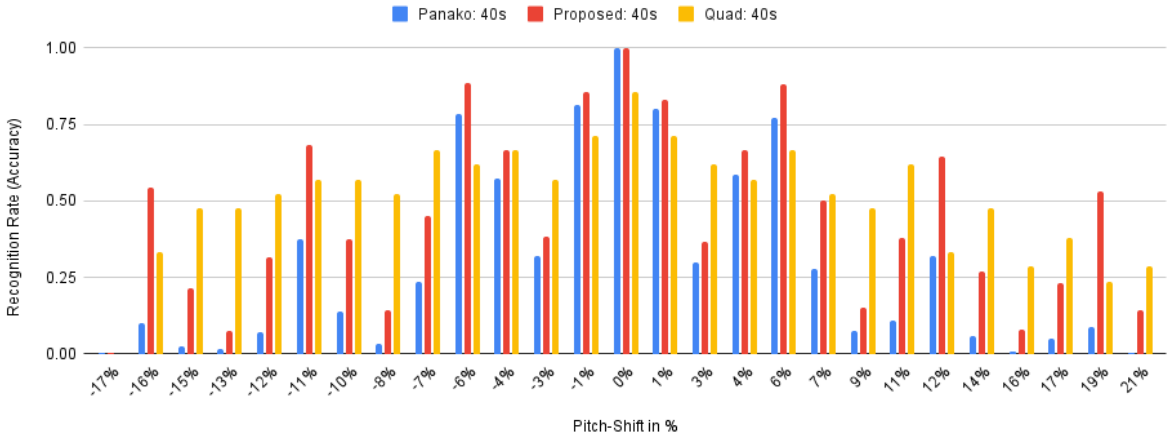

FIGURE 5.4: Accuracy of Panako, Proposed and Quad for 40s pitch-shifted query audios

5.5.2 Experiment Group 2: Granularity: Pitch-shifting

Figures 5.3 and 5.4 show the result for experiment group 2. This group evaluates the effect of audio length on the accuracy value of the proposed system against pitch-shifted query audios. The proposed approach gives better accuracy value as the length of the query audio grows from 20s to 60s (see Figure 5.3 and 5.4). Similar trend is observed for Panako and Quad as well.

In Figure 5.5, the average execution time it took to query pitch-shifted query audios

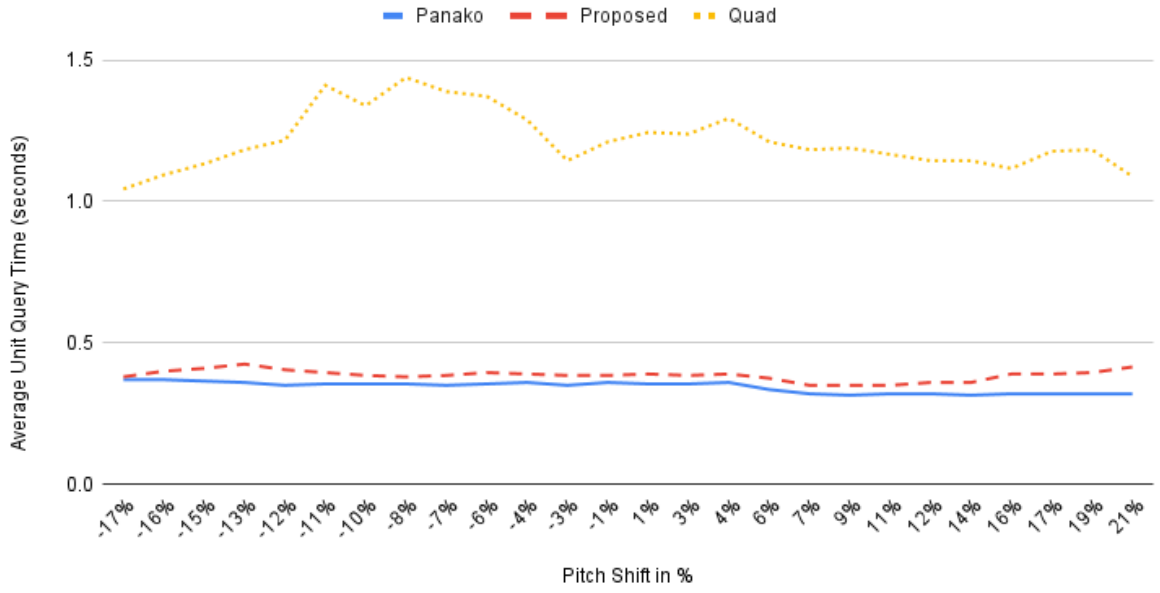

FIGURE 5.5: Average query time for a single query audio of Panako, Proposed system and Quad for 20s pitch shifted query audios

of 20 seconds from reference database of 3000 full length audios is presented. The proposed approach takes less than 0.425 seconds on average to query single audio for all pitch-shift percentages. Although, there is slight increase in query time as modification percentages grows in magnitude, the query time graph given in Figure 5.5 indicates that the proposed system took relatively equivalent amount of time to query an audio despite the amount of pitch-shift modification it goes through. Similarly, Panako took more or less equivalent amount of time to query audio of different pitch-shift modification. From Figure 5.5, its clear that Panako took slightly less time than the proposed approach to query audios of all modification percentages. This, as indicated above, is due to the extra filtering stage introduced in the matching stage of the proposed approach. Quad, on the other hand, took three times longer than the proposed method.

5.5.3 Experiment Group 3: Robustness against other Audio Modifications

Experiment 3.a: Robustness against Audio Effects

The proposed approach is evaluated against 1000 query audios with various audio effects (200 for tremolo, 200 for echo, 200 for chorus, 200 for flanger and 200 for band-pass). The result is given in Figure 5.6. The accuracy of the proposed approach is above

80% for all audio effects except for chorus. For chorus the proposed approach gives a better result than Panako [3]. In contrast, the accuracy of Quad is lower than that of the proposed approach for all audio effects except band-pass.

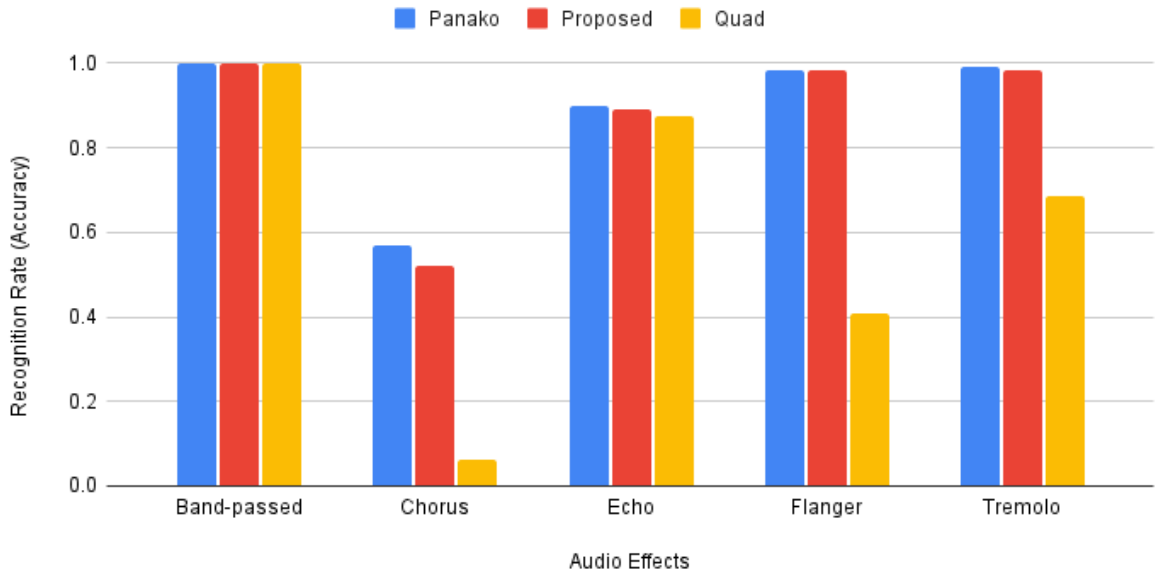

FIGURE 5.6: Accuracy of Panako, Proposed and Quad for 60s query audios with various audio effects.

Experiment 3.b: Robustness against Linear Speed Change

Here, 4000 linear speed modified query audios of length 60 seconds, which are extracted from the reference audios with modification rate of -20 percent to +20 with 2 percent step, are presented to the proposed approach, Panako[3] and Quad[11]. The result, depicted in Figure 5.7, indicates that the proposed system achieves accuracy value of 80 percent and above for speed modification starting from -6% to +12%. In contrast, Panako achieved accuracy value of 80 percent and above for linear speed modification starting from -4% to +8%. Quad, on the other hand, achieved the same accuracy level for modification rate starting from -16% to +12%. This work achieves a slight improvement over Panako for linear speed modified query audios. Quad handled linear speed modification better than proposed approach and Panako.

Experiment 3.c: Robustness against Time-Stretch

In this experiment 4000 time stretched query audios of length 60 seconds, which are extracted from the reference audios with modification rate of -20 percent to +20 with 2 percent step size, are presented to both the proposed approach, Panako[3] and Quad[11]. The result, depicted in Figure 5.8, indicates that the proposed system achieves accuracy

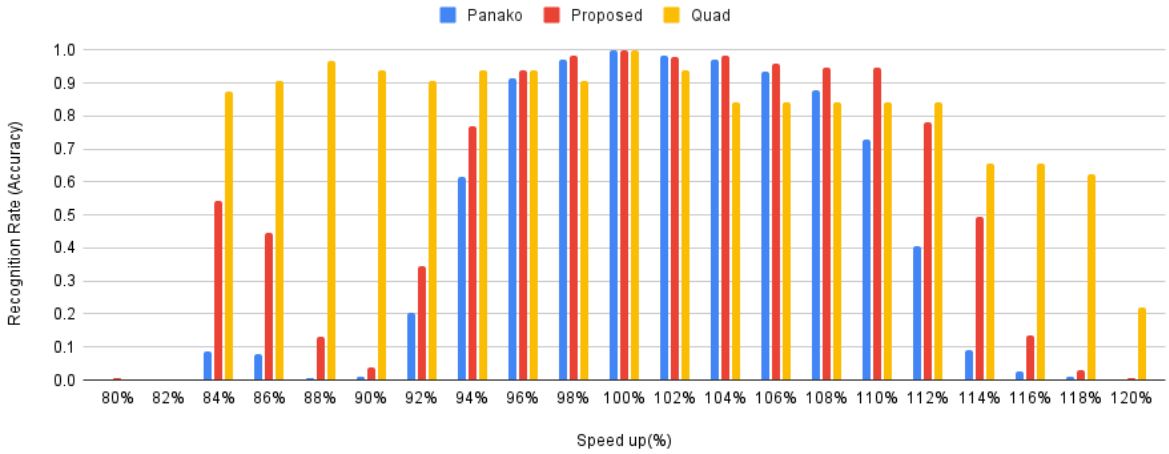

FIGURE 5.7: Accuracy of Panako, Proposed and Quad for 60s linear speed modified query audios.

value of 80 percent and above for modification of -4% to +4%. Panako achieved accuracy of more than 80 percent for modification of -4% to +4%. The overall result in Figure 5.8 indicates that proposed approach has the same result with that of Panako for time-stretch query audios. This is expected as the proposed approach uses identical mechanism to encode time information in the hash (see Equation 4.11) as Panako. Quad achieved above 80% accuracy for time-stretch modification rate of -18% to +18%.

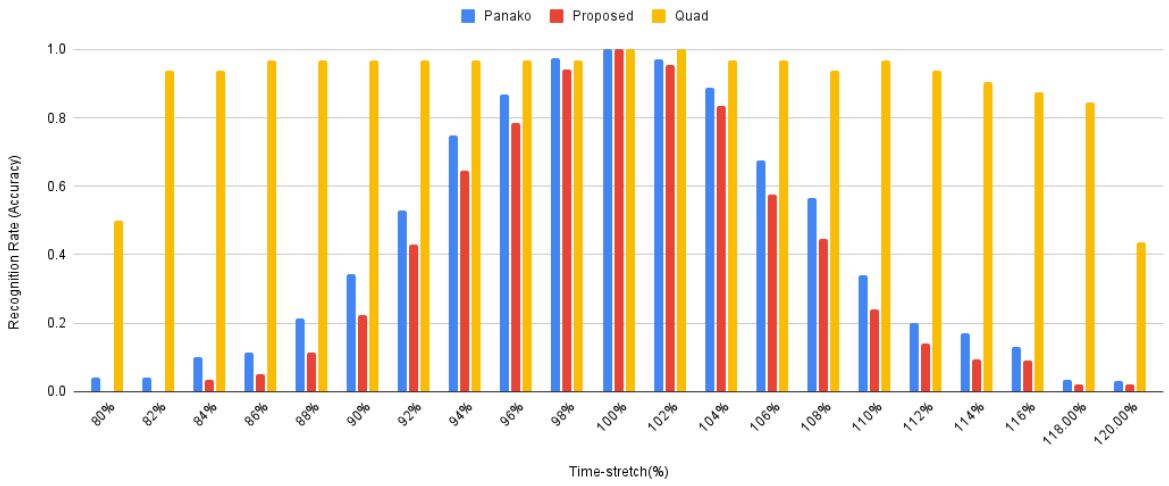

FIGURE 5.8: Accuracy of Panako, Proposed and Quad for 60s time stretched query audios

5.5.4 Experiment Group 4: Granularity: Other Audio Modifications

Figures 5.9 and 5.6 show the result of experiment on the effect of audio length on the accuracy of the proposed system for query audios with common audio effects. Figures

5.9 and 5.6 show that for band-passed, flanger and tremolo audio effects, the proposed system exhibits accuracy value greater than 80% for audio length of 20 seconds, 40 seconds and 60 seconds. For chorus the proposed approach improves the accuracy value of Panako[3]. For echo audio effect, both the proposed approach and Panako achieves above 80% accuracy value for both 40s and 60s length query excerpts. Both works exhibit a slightly lower than 80% accuracy value for 20s query audios with echo audio effect. Figure 5.9 also shows that as the length of audio gets smaller the accuracy value of the proposed system is affected by chorus. This trend is evident for Panako as well. Chorus has a lower accuracy value due to its blurring impact on a spectrogram, making it difficult for both works to detect fingerprint matches[3]. The effect of audio length on Quad's accuracy value is visible on Figure 5.9. As the length of query audio shortens, Quad accuracy values drop significantly for most of the audio effects. Quad can't score accuracy value of above 40% for 20s second queries of all audio effects.

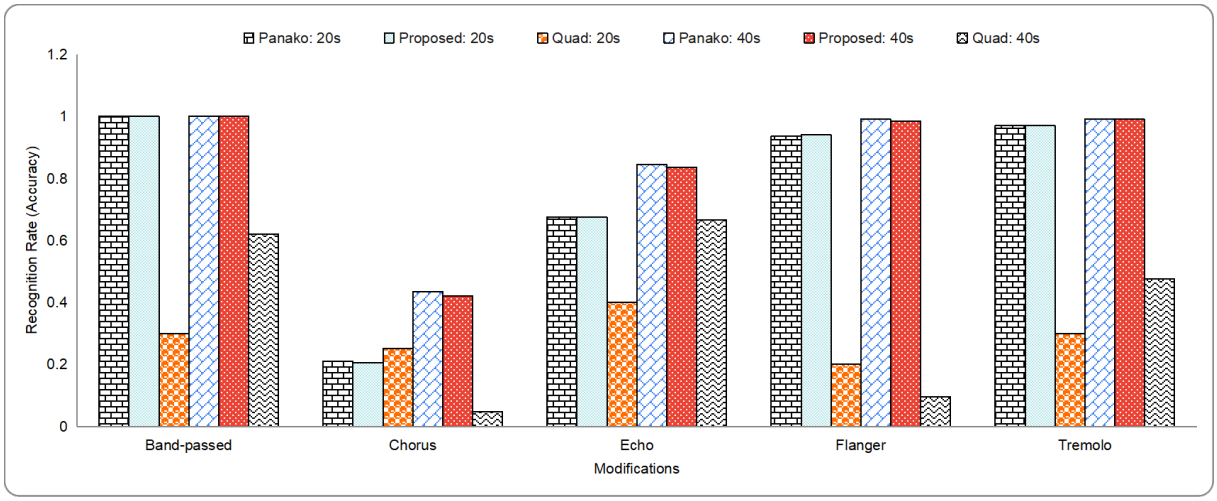

FIGURE 5.9: Accuracy of Panako, Proposed and Quad for common audio effects and different audio lengths

The effect of audio length on the accuracy value for linear speed query audios is shown in Figures 5.10 and 5.11. For lower modification values up to 4%, the effect of query length on the accuracy is minimal. However, as the modification percentage grows, the effect of audio length becomes more evident. As we can see from Figure 5.10, the accuracy value of proposed approach drops below 80% for linear speed modification value of 6%. Over all the proposed approach results in better accuracy value as the length of the query audio grows from 20s to 60s. Similar trend is observed for Panako as well. The effect of audio length is more evident on the accuracy value of Quad as shown in Figures 5.10 and 5.11. The accuracy of Quad for 20s linear speed modified audios is below 40% for almost all modification rates. For 40s linear speed modified

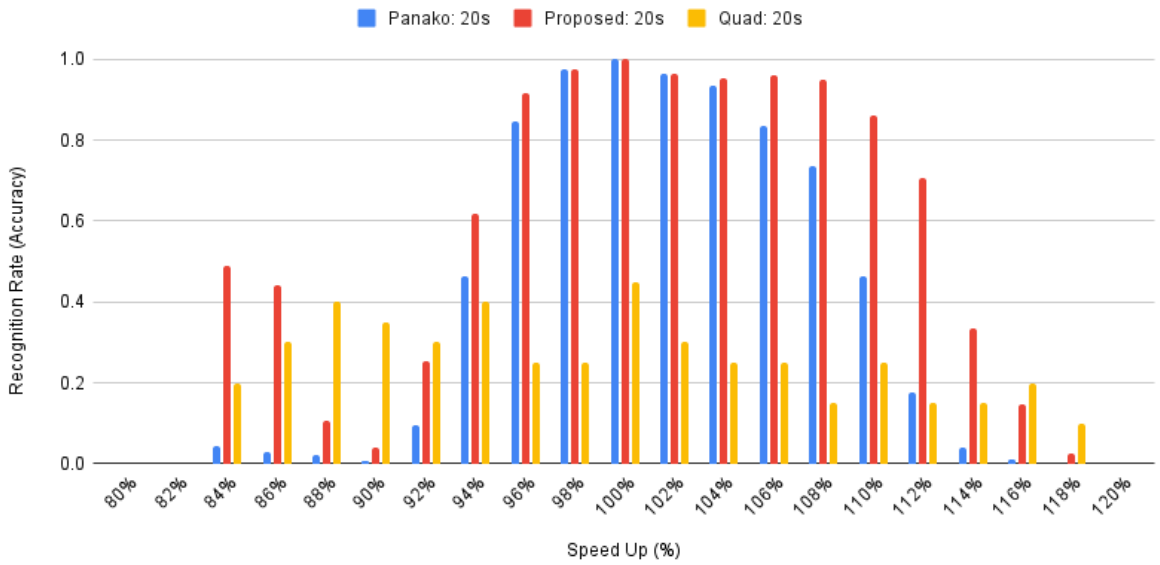

FIGURE 5.10: Accuracy of Panako, Proposed and Quad for 20s linear speed modified query audios

query audios, Quad yields relatively better accuracy results compared to its 20s audios but they all are below 80%.

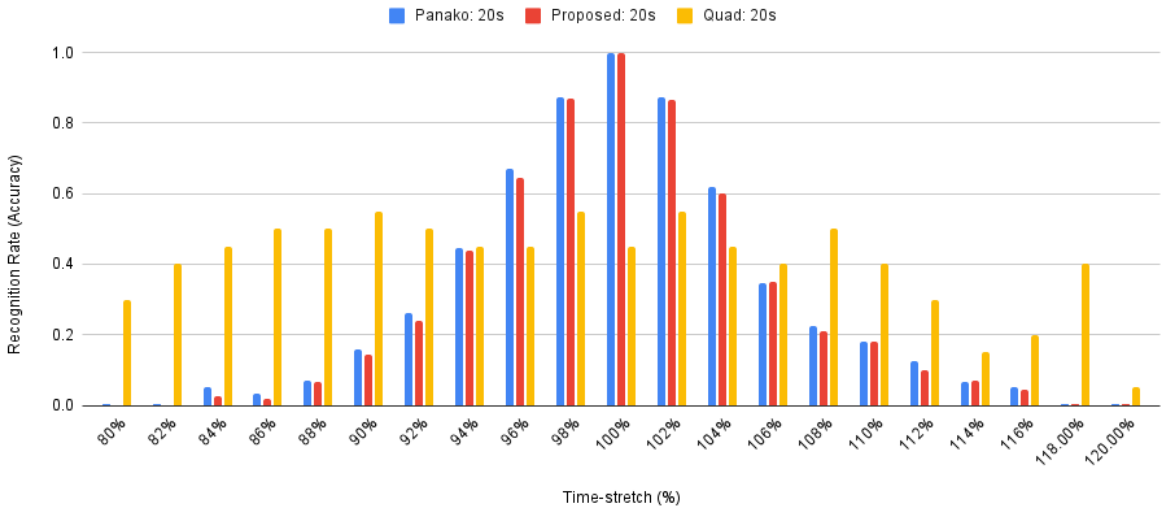

FIGURE 5.12: Accuracy of Panako, Proposed and Quad for 20s time stretched query audios

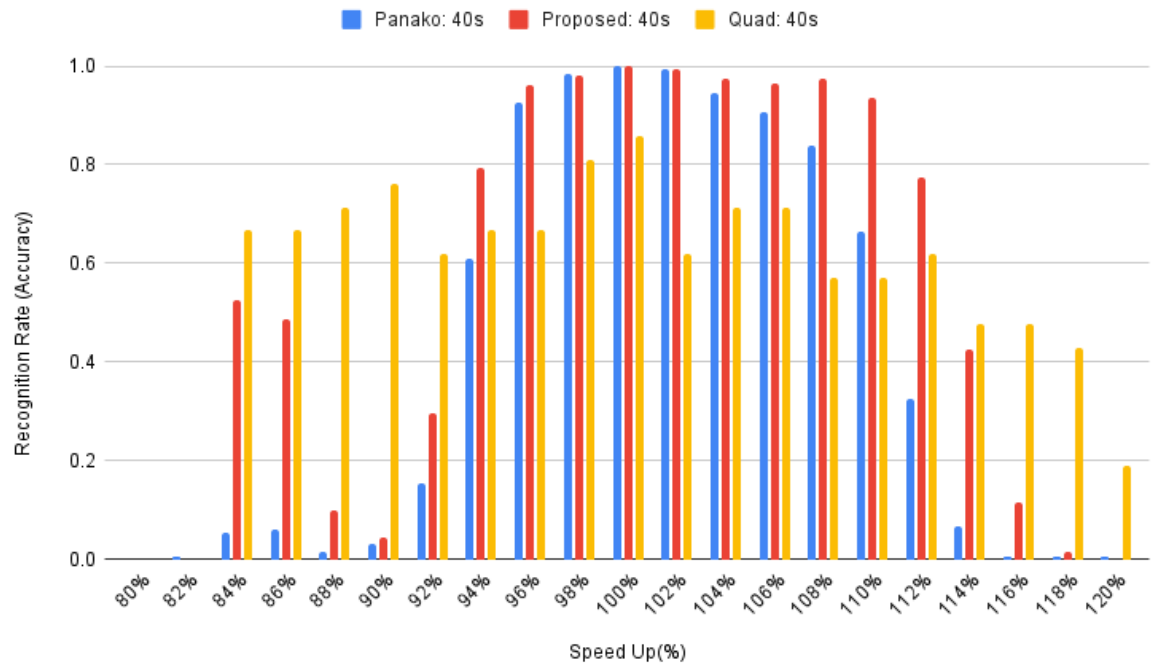

FIGURE 5.11: Accuracy of Panako, Proposed and Quad for 40s linear speed modified query audios

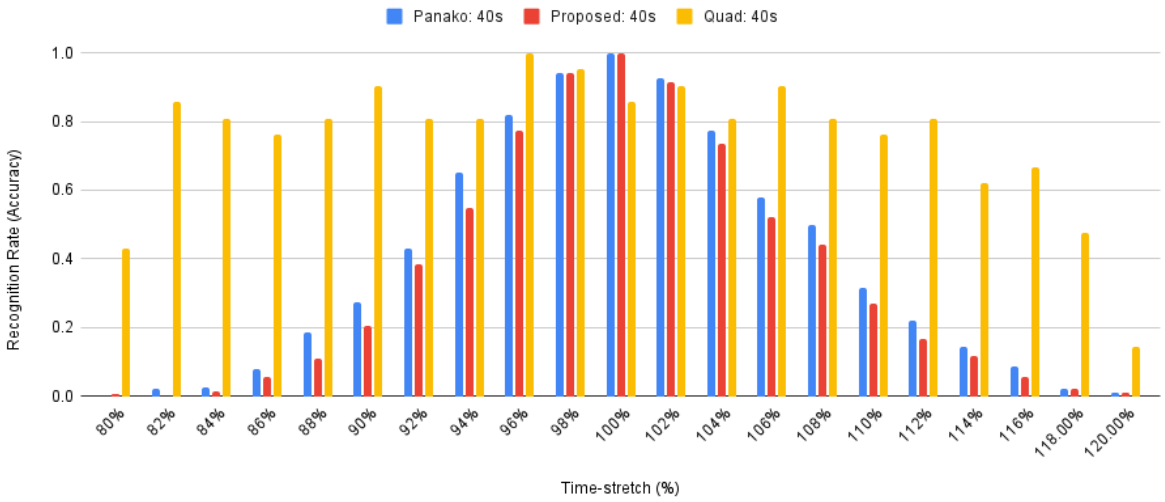

FIGURE 5.13: Accuracy of Panako, Proposed approach and Quad for 40s time stretched query audios

The effect of audio length on the accuracy value for time-stretched query audios is shown in Figures 5.12 and 5.13. The effect of audio length for time-stretched query audio grows with the modification magnitude. For lower modification values up to 2%, the effect of query length on the accuracy is minimal. However, as the modification percentage grows, the effect of audio length becomes more evident. Over all the

proposed approach results in better accuracy value as the length of the query audio grows from 20s to 60s. Similar trend is observed for Panako as well. The accuracy of Quad for identifying 20s query audios is below 80% for all modification rates. This result gets improved for 40s query audios. For 40s queries, Quad achieved more than or equal to 80% accuracy for modification rates starting from -18% to +12%.

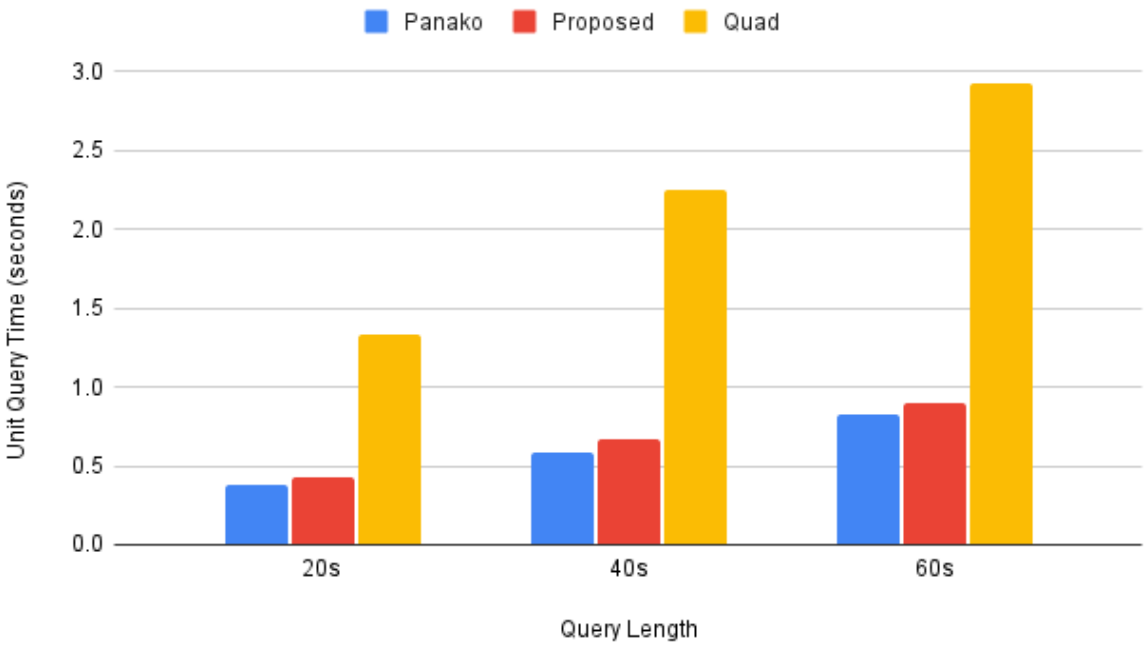

FIGURE 5.14: Average query time for a single query audio of Panako, proposed approach and Quad for reference query audios with audio lengths of 20s, 40s and 60s.

In Figure 5.14, the average execution time taken to query unmodified (reference) audio with length equal to 20, 40 and 60 seconds from reference database of 3000 full length audios is presented. The proposed approach queries a single audio from the reference database for all audio lengths under 0.9 seconds. For 20 second query audio, the proposed approach on average took only 0.426 seconds. For 40 and 60 second query audios, it took 0.67 and 0.9 seconds respectively. Panako, in contrast, took slightly less time than the proposed approach to query audios of all lengths. This is due to the extra filtering stage introduced in the matching stage of the proposed approach. Quad took, on average, 1.34 seconds to query a single 20s from the same database. It took, on average, 2.25 and 2.926 seconds to query a single 40s and 60s query audio respectively. From this figures we can see that Quad took almost three times more time than the proposed approach to query a single query audio irrespective of query audio length.

5.5.5 Results Summary

The results collected from all the experiments are summarized according to the research questions stated in section 1.1:

- **[RQ1]**: The proposed approach showed robustness to pitch shifting with modification rate ranging from -11% to +12% with above 80% recognition rate except for modification values around specific instances, such as -8, -3, +3 and +9 percents. This signifies the use of frequency delta only to formulate fingerprint hash enabled the proposed approach to formulate a pitch-shifting invariant fingerprint hash.
- **[RQ2]**: The results of the proposed system for pitch-shifted query audios of different lengths demonstrated that the use of vertically segmented target zones improves the matching of pitch-shifted queries with a rate ranging from -11% to +12% except for modification values around specific instances, such as -8, -3, +3 and +9 percents.

Overall, using frequency delta only fingerprint hashes and vertically segmented target zones, one can develop an audio fingerprinting system robust to pitch-shifting. Moreover, the results showed these features improved the robustness of the proposed approach to various audio modifications such as linear speed change, time-stretching, and audio effects such as tremolo, echo, flanger, and band-pass.

5.6 Threats to Validity

5.6.1 Internal Validity

Internal validity indicates the researcher's confidence that a cause-and-effect relationship reported in a study cannot be explained by other factors. Threats to internal validity should be identified in advance and dealt with to produce a credible study. Common threats to internal validity relevant to this study include instrumentation and selection bias threats.

Instrumentation threat may arise due to a possible mismatch between the development environment and physical hardware used to evaluate our proposed approach and the baseline. To avoid this threat both works are implemented in the same programming language (Java), and their implementations are executed on the same hardware under the same settings. Selection bias threat may arise due to a possible mismatch between the selected dataset used to evaluate our proposed approach and the baseline. This

threat is dealt with by selecting the same set of reference audios from Jamendo [31] and using the same query audios for both works.

5.6.2 External Validity

Threats to the external validity of this work may arise from the size and quality of the dataset used for evaluation. To avoid this threat, we used a freely available open-source dataset (Jamendo[31]). We stored over 3000 full-length audio in the reference database. The query audios are also generated using an open-source audio processing tool (SoX[32]).

The parameters used to implement the proposed approach can limit the external validity of this work. Sample rate, vertically segmented target zone dimensions, number of bins per octave, number of peaks per second, and threshold values are examples of such parameters.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

Audio fingerprinting has variety of uses such as music information retrieval, broadcast monitoring and royalty tracking. Audio modifications and audio effects are among the most important challenges to design robust audio fingerprinting systems. The influence of pitch shifting in audio-fingerprinting systems was investigated in this thesis. When CQT is utilized, pitch shifting causes spectral peaks to be translated along the pitch axis. A new target zone assignment technique and an additional filtering stage are proposed to deal with the translation impact. Different experiments have been utilized to evaluate the proposed approach's robustness, granularity, and time-complexity.

The proposed approach has been evaluated using a freely available data set of 3000 songs and compared with a baseline work, Panako [3] and Quad [11]. The results show that the proposed system is robust to common audio effects such as echo, tremolo, flanger, band-pass and chorus. The result also shows that the proposed approach is able to withstand pitch-shifting, time-stretching and linear-speed changes. Specifically, proposed system:

- achieved above 80% accuracy value for audio effects under investigation namely: echo, tremolo, flanger, band-pass and chorus.
- achieved above 80% accuracy value for pitch-shifted query audios with modification factors starting from -11% to +12% except for modification values around specific instances, such as -8, -3, +3 and +9 percents. The reason for drop in accuracy value on these specific modification values is due to maximum number of peak spread to two frequency bins. The proposed approach shows improvement over Panako[3], which can only handle -6% to +6% pitch shifts. Quad [11] achieved the same accuracy level for modification values starting from -12% to

+7%. Quad [11] achieved consistent accuracy values compared to both the proposed approach and Panako[3], which suffered drops on specific modification values.

- achieved accuracy level of 80 percent and above for linear speed modification starting from -6% to +12%. In contrast, Panako achieved the same level of accuracy for linear speed modification rate starting from -4% to +8%. Quad, on the other hand, achieved the same accuracy level for modification rate ranging from -16% to +12%. This work achieves a slight improvement over Panako for linear speed modified query audios. Quad handled linear speed modification better than both the proposed approach and Panako.

- achieved accuracy level of 80 percent and above for time-stretch modification rate ranging from -4% to +4%. Panako achieved the same accuracy level for modification rate ranging -4% to +4%. The proposed approach has the same result with that of Panako for time-stretch query audios. This is expected since the proposed approach uses same mechanism as Panako to encode time information in the hash (see Equation 4.11). Quad, however, achieved above 80% accuracy for time-stretch modification rate of -18% to +18% which is significantly better than both the proposed approach and Panako.

The effect of audio length on the robustness of the proposed systems is evaluated for query audios with common audio effects and modifications. The proposed system:

- exhibits small drop in accuracy value as the length of query audio starts to decreases from 60 to 20 seconds for all audio effects under investigation except chorus. For chorus, however, the proposed system showed significant drop in accuracy value as the length of query audio decreases from 60 to 20 seconds.

- results in better accuracy value as the length of the query audio grows from 20s to 60s for query audios with pitch-shifts, time-stretch and linear speed changes.

The effect of audio length on the time complexity of the proposed systems is evaluated for query audios with no modifications and with pitch-shift. The proposed system:

- achieves to query a single audio from the reference database for all audio lengths under 0.9 seconds. For 20 second query audio, the proposed approach on average took only 0.426 seconds. For 40 and 60 second query audios, it took 0.67 and 0.9 seconds respectively. Panako, in contrast, took slightly less time than the proposed approach to query audios of all lengths. This is due to the extra filtering stage introduced in the matching stage of the proposed approach. Quad took, on

average, 1.34 seconds for a 20s query and took, on average, 2.25 and 2.926 seconds to query a single 40s and 60s query audio respectively. Quad took almost three times more time than the proposed approach to query a single query audio irrespective of query audio length. The reasons for Quad's slow query time are related to how it formulate the fingerprint hash, the use of quads instead of triplets and the audio hash search algorithm.

- achieves to query a single audio from the reference database for all pitch-shift percentages on average in under 0.425 seconds. Although, there is slight increase in query time as modification percentages grow in magnitude, the proposed approach took equivalent amount of time to query an audio despite the amount of pitch-shift modification it goes through. Panako took more or less equivalent amount of time to query audio of different pitch-shift modification. Panako took slightly less time than the proposed approach to query audios of all modification percentages. This is due to the extra filtering stage introduced in the matching stage of the proposed approach. Quad, on the other hand, took more than three times when compared to the proposed method.

6.2 Future Works and Recommendations

This thesis work can further be enhanced by considering the following points:

- Replacing exact matching technique used in fingerprint hash matching stage 4.4.1 with techniques like local sensitive hashing (LSH), which allows nearest neighbour searches in sub-linear time, would be a clear improvement and allow much larger databases.

- Formulating better filtering technique by storing f_3 along f_1 in the reference fingerprint to allow frequency delta equivalence check among reference and query fingerprints. Equation 6.1 could be used to do this check.

$$f\_mod = \frac{|f'_3 - f'_1|}{|f_3 - f_1|} \tag{6.1}$$

- Figuring out a solution for significant drop in accuracy value at specific pitch-shifts values discussed in Section 5.5.1.

References

[1] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system with an efficient search strategy," *Journal of New Music Research*, vol. 32, no. 2, pp. 211–221, 2002.

[2] S. Fenet, G. Richard, and Y. Grenier, "A scalable audio fingerprint method with robustness to pitch-shifting.," in *ISMIR*, 2011, pp. 121–126.

[3] J. Six and M. Leman, "Panako: A scalable acoustic fingerprinting system handling time-scale and pitch modification," in *15th International Society for Music Information Retrieval Conference (ISMIR-2014)*, 2014.

[4] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A review of audio fingerprinting," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 41, no. 3, pp. 271–284, 2005.

[5] B. Logan, "Mel frequency cepstral coefficients for music modeling.," in *ISMIR*, vol. 270, 2000, pp. 1–11.

[6] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[7] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[8] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant q transform," *The Journal of the Acoustical Society of America*, vol. 92, no. 5, pp. 2698–2701, 1992.

[9] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE transactions on information theory*, vol. 36, no. 5, pp. 961–1005, 1990.

[10] A. Wang, "An industrial strength audio search algorithm.," in *Ismir*, Washington, DC, vol. 2003, 2003, pp. 7–13.

[11] R. Sonnleitner and G. Widmer, "Robust quad-based audio fingerprinting," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 3, pp. 409–421, 2015.

[12] R. Sonnleitner and G. Widmer, "Quad-based audio fingerprinting robust to time and frequency scaling.," in *DAFx*, Citeseer, 2014, pp. 173–180.

[13] S. M. Bernsee, "Time stretching and pitch shifting of audio signals," *The DSP Dimension*, 2003.

[14] L. Trandafir. "Audio effects: The beginner's guide to shaping your sound." (2021), [Online]. Available: https://blog.landr.com/audio-effects-plugins-guide/ (visited on 09/01/2021).

[15] M. Zanoni, S. Lusardi, P. Bestagini, A. Canclini, A. Sarti, and S. Tubaro, "Efficient music identification approach based on local spectrogram image descriptors," in *Audio Engineering Society (AES) Convention*, 2017.

[16] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[17] X. Zhang, B. Zhu, L. Li, *et al.*, "Sift-based local spectrogram image descriptor: A novel feature for robust music identification," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, p. 6, 2015.

[18] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008, Similarity Matching in Computer Vision and Multimedia, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314207001555.

[19] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[20] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "Orb: An efficient alternative to sift or surf.," in *ICCV*, Citeseer, vol. 11, 2011, p. 2.

[21] D. Williams, A. Pooransingh, and J. Saitoo, "Efficient music identification using orb descriptors of the spectrogram image," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2017, no. 1, p. 17, 2017.

[22] A. V. Kulkarni, J. Jagtap, and V. Harpale, "Object recognition with orb and its implementation on fpga," 2013.

[23] S. Yao, B. Niu, and J. Liu, "A sampling and counting method for big audio retrieval," in *2016 IEEE Second International Conference on Multimedia Big Data (BigMM)*, 2016, pp. 307–313. DOI: 10.1109/BigMM.2016.27.

[24] S. Yao, B. Niu, and J. Liu, "Enhancing sampling and counting method for audio retrieval with time-stretch resistance," pp. 1–5, 2018.

[25] R. Chu, B. Niu, S. Yao, and J. Liu, "Peak-based philips fingerprint robust to pitch-shift for massive audio retrieval," in *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, 2019, pp. 314–320. DOI: 10.1109/BigMM.2019.000-3.

[26] A. Arzt, S. Böck, and G. Widmer, "Fast identification of piece and score position via symbolic fingerprinting," Jan. 2012.

[27] E. Gebie, "Triple point geometric hashing based audio fingerprinting," *Addis Ababa University*, 2020.

[28] FFmpeg. "Ffmpeg." (2021), [Online]. Available: https://www.ffmpeg.org/ (visited on 09/01/2021).

[29] Wikipedia. "12 equal temperament." (2021), [Online]. Available: https://en.wikipedia.org/wiki/12_equal_temperament.

[30] MapDB, "Mapdb," *https://mapdb.org/*, 2020. [Online]. Available: https://mapdb.org/.

[31] Jamendo, "Jamendo," 2020. [Online]. Available: https://www.jamendo.com/?language=en.

[32] SoX, "SoX - Sound eXchange," 2020. [Online]. Available: http://sox.sourceforge.net/.

[33] J. Six, O. Cornelis, and M. Leman, "Tarsosdsp, a real-time audio processing framework in jav," in *Proceedings of the 53rd AES Conference*, 2014, AES53–0022–1–AES53–0022–7. [Online]. Available: http://0110.be/posts/TarsosDSP_Paper_and_Presentation_at_AES_53rd_International_conference_on_Semantic_Audio.

[34] J. Six, "Jorensix: Panako," 2020. [Online]. Available: https://github.com/JorenSix/Panako.

[35] M. Bortnyck, V. Pal, and I. Timeev, "Quad implementation," 2022. [Online]. Available: https://github.com/mbortnyck/qfp.