Audio Fingerprinting with Robustness to Pitch Scaling and Time Stretching

Yesenia Díaz Millet
Computer Engineering
Jeffrey Duffany, Ph.D.
Electrical & Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— Current audio fingerprinting systems are becoming increasingly robust against noise and filter distortions, however songs that have been pitch scaled and time stretched are still likely to pass undetected. This research focuses on expanding an existing landmark-based fingerprinting method to identify songs that have been pitch scaled and time stretched to escape current systems while still sounding natural to the human ear. Two feature extraction methods have been explored with the purpose of resolving each task individually. The constant Q spectrogram was used for feature extraction, instead of a conventional spectrogram, to identify songs that have been pitch scaled. Mel-frequency Cepstral Coefficients were used as features for the other task. The goal is to verify whether or not low-level spectral based features alone are capable of handling such transformations in a song instead of needing to use mid-level or high-level musical features as is the case with other Song ID methods.*

Key Terms *— Audio Fingerprinting, Feature Extraction, Music Information Retrieval, Music Similarity.*

INTRODUCTION

Technology advances over the years have changed the way people gather and obtain information. The evolution of the Internet has made it possible for people to share files with others across the world. Different industries have taken advantage of this in order to make their businesses boom. One such industry is the music industry which has increased the amount of music available in digital form. Video and audio sharing sites also make it possible for anyone to upload and share their own multimedia content. This brings about different motivations for music similarity systems such as royalty rights management, personal database organization, music preference list creation, and playlist generation, among others.

This research focuses more on audio identification for personal use as well as for royalty rights management and could even be useful for copyright infringement detection. Because anyone with an account can easily upload their content to video or audio sharing sites, it is important to be able to identify the type of content to make sure that it does not violate others copyrights. Such sites have ways for other users to report offending content but with the increase of users and content that is uploaded, they need to rely on more automated processes. One solution suggested over the years has been the addition of digital watermarks to the content which makes identifying it easy provided that the watermark has not been tampered with. Another solution suggested has been audio fingerprinting. This uses the audio content itself without having to add an undetectable signature and it can be more robust when audio has been tampered with. Both methods require a database to compare the query audio against but they each have their benefits and drawbacks. Gomes et al. [1] explains how both audio watermarking and fingerprinting work and what type of applications they are useful for.

It is public knowledge that such automated systems for identifying content exist. Some users will try to avoid detections by such systems by filtering or compressing the audio and even go as far as trying to alter the pitch and tempo of the audio. Such modifications could alter the signature in watermarking systems so that the offending audio goes undetected and they can alter the quality of the audio so much that even the most robust

audio fingerprinting systems cannot detect it. But how much alteration should be detectable? As Lebossé, Brun, and Pailles [2] stated in their paper "the fingerprint has to be robust against alterations as long as important perceptual properties of the original signal are not significantly altered". This means that alterations that make the signal unrecognizable to the human ear need not be detected especially if they are irreversible.

As this research expands on an already existing audio fingerprinting system which is robust to noise addition and filtering alterations, the focus is mostly on songs that have had the pitch or tempo altered. The motivation behind this is that people are still able to recognize a song even if the pitch or tempo has been altered. This is especially true when listeners do not have a music background and are unable to remember the exact pitch or speed of the song. Even broadcast music is often altered, although minimally, in such a way so that radio stations are able to play more songs in a given amount of time. Therefore making an audio fingerprinting system robust to pitch scaling and tempo stretching is useful for the purpose of simply identifying a song for personal use as well as for royalty rights management.

AUDIO FINGERPRINTING

A fingerprint is usually used as a unique identifier of a person. This idea has been around for a while but it is not limited to using a person's fingerprints to identify them. Multimedia fingerprinting takes from this idea in order to identify multimedia content, such as audio, images, or video, from a fingerprint. As stated by Haitsma & Kalker [3], the main objective of multimedia fingerprinting is "an efficient mechanism to establish the perceptual equality of two multimedia objects: not by comparing the (typically large) objects themselves, but by comparing the associated fingerprints (small by design)." In the case of audio fingerprinting, a fingerprint is defined as "a compact content-based signature that summarizes an audio recording" [4].

Audio fingerprinting is one of the most specific tasks in MIR research. It aims to identify an exact song from a query, preferably small, against a database of known songs. Using a fingerprint brings about some advantages over using the entire audio content. Some of these advantages are reduced memory storage requirements, efficient comparison, and efficient searching. Based on the review by Cano et. al [4], the characteristics of a fingerprint can be summarized as follows:

- Perceptual digest of the recording – It must retain enough information to be distinguishable among a large database.
- Invariant to distortions – must be robust against distortions. Depending on the usage of the system this constraint can be relaxed in order to detect intentional manipulations.
- Compact – the fingerprint obtained must be small.
- Easy computation – the fingerprint calculation should not be overly time-consuming.

Some of these characteristics conflict with others which is why it is important to determine what is an acceptable trade off in order to find a good balance among them.

Most fingerprinting methods use a hash function. However, this hash function is not the same as a cryptographic hash. In order for a cryptographic hash function to be effective, it must be resistant to collisions. This means that no two objects should hash to the same value, even if the two objects are perceptually similar. Conversely, for fingerprinting systems it is important that perceptually similar objects have the same hash value. In the case of audio, it can be said that two objects are perceptually similar if they sound alike.

TOOLS AND ALGORITHM USED

This work uses as its base Dan Ellis's Robust Landmark-Based Audio Fingerprinting [5]. The algorithm is Ellis's own interpretation of the Shazam algorithm as described by Wang [6], which is robust against filtering and distortions, and implemented in the Matlab environment. Other

resources used include a constant Q spectrogram [7] and an MFCC [8] implementation by Dan Ellis. This section will provide a description of the main algorithm used as well as a description of the implementation of the constant Q transform and MFCCs and what they were used for in this work.

Landmark-Based Audio Fingerprinting

The landmark-based [5] approach relies on spectrogram peaks just like the Shazam [6] algorithm it is based on. Using only the maximum peaks results in a reduction of a complicated spectrogram to a sparse set of coordinates which Wang calls "constellation maps". Pairs of spectral peaks within these constellation maps form landmarks and they should be the same for matching segments of audio. Frequencies are paired when they are close enough to each other depending on a threshold.

Before generating a spectrogram, a song is first re-sampled to 8000 Hz. The spectrogram uses a 64ms window and a 32ms overlap which results in a 512 point Fast Fourier Transform (FFT) thus providing good spectral resolution for music. The frequencies with the highest energy amplitudes are taken then paired into landmarks. Some of these frequencies will become "seeds" or "anchor points", that is to say that some frequencies will form more than one landmark pairs with nearing frequencies.

Each landmark pair is represented as a four-tuple containing information about two frequency and two time offset points, which are the rows and column values of the spectrogram as calculated in Matlab. This information is saved as start time (*T1*), start frequency (*F1*), end frequency (*F2*), and time difference (*DT*). The start time is the time column of the seed frequency point, start frequency is the frequency bin of the seed point, end frequency is the closest frequency point which forms a pair with the seed, and the time difference is the difference between the time corresponding to the end frequency and the start frequency. As shown in Figure 1.

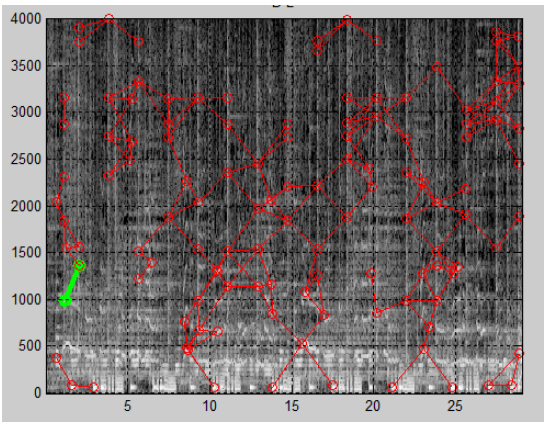

Figure 1
Example of Landmarks within a Spectrogram

Each landmark is then converted to a 20 bit hash value. Since each song has multiple landmarks they also have multiple sub hash values, one for each unique landmark. Each landmark is converted to the hash value by XORing the values of *F1*, frequency difference (*DF*), and *DT*. The first 8 bits of the hash are composed of *F1*, the next 6 bits are composed of *DF*, and finally the last 6 bits are composed of *DT*. The hash is then used to store the song ID and start time of a specific landmark in a type of inverted index hash table. In order to find a match, landmarks are extracted from the query song and converted to unique hashes. The song ID and start time is obtained for every sub hash found within the table. The most repeated song ID with the most popular time offset from the beginning of the audio file until the seed point is found to be the match to the query. Twenty songs are taken which are the closest matches to the query and are each quantified by modalDTcount, modalDT, and hashCount. The value of modalDT is the most common time offset in the current song, modalDTcount is the count of how many landmark hashes between query and reference song have the same time offset, and hashCount is the count of the hashes with a specific song ID.

The Constant Q Transform

For this research, the Constant Q Transform was used as a replacement for the conventional spectrogram to help identify songs which have been pitch scaled. Fenet et al. also use the constant Q

transform in their audio fingerprint method [9] which is also based on Wang's algorithm [6]. The reason for this is that the constant Q transform provides a log frequency representation of the sound which "gives a constant pattern for the spectral components." [10] This makes it more straightforward to identify fundamental frequencies as well as makes it easier to identify the instrumentation within a song.

The conventional spectrogram gives a linear frequency representation in which there is a constant separation between frequency bins. In this representation a change in pitch is equivalent to a multiplicative change in frequency which will cause frequencies to be stretched or compressed. However, frequency bins are spaced logarithmically in the constant Q spectrogram therefore the frequency change becomes additive resulting in a vertical shift. This means that in the conventional spectrogram a change of pitch by a factor of K results in the frequency bins changing to $f*K$ while this change in the constant Q domain results in $f + K$.

Table 1 contains a comparison of variables in the calculation of the discrete Fourier transform (DFT) and the constant Q transform. [10] In the constant Q domain the separation of the frequency bins depends on the number of bins b per octave chosen, where an octave is a series of 8 notes equivalent to 12 semitones in the musical scale. The frequency of the bins will vary from a minimum frequency f_{min} to an upper frequency which must be below the Nyquist frequency.

Table 1
Comparison of Variables Between DFT and CQT

	Constant Q	DFT
Frequency	$f_k = (2^{1/b})^k f_{min}$	$k * \Delta f$
Window	variable = $N[k] = (SR*Q)/f_k$	constant = N
Resolution Δf	Variable = f_k/Q	constant = SR/N
$f_k/\Delta f_k$	constant = $Q = 1/(2^{1/b} - 1)$	variable = k
Cycles in Window	constant = $Q = 1/(2^{1/b} - 1)$	variable = k

Mel-frequency Cepstral Coefficients

Mel-frequency Cepstral Coefficients (MFCC) is a widely used feature in speech recognition. The MFCC has also gained usage in music classification systems because they are believed to encode timbral information which is the quality which what different types of sounds are produced from. In 2000 Beth Logan examined the effectiveness of MFCC features for music modeling. [11] This work demonstrates that the usage of MFCC is not harmful for representing a musical piece.

Li and Chan [12] also experiment with the usage of MFCC for music modeling by using such features in music genre classification. The objective is to determine whether MFCC are invariant to changes in key and tempo. Their experiments show that while MFCCs are not invariant to key, they seem to be invariant to tempo which is valuable for this research.

MFCC features represent a spectral envelope of the analyzed signal. They attempt to capture the most perceptually important parts of the signal and are calculated as follows [11, 13]:

- Convert to Frames – Audio is segmented into small frames. A window size of 20 to 40ms is typically used.
- Calculate the frequency spectrum – The Fourier Transform is used to convert each frame from time domain into frequency domain.
- Take the Log of the amplitude spectrum – The perceived loudness of a signal does not follow the linear scale. It is approximately logarithmic.
- Mel-scaling and smoothing – The signal is passed through a mel filter bank to emphasize the perceptually meaningful frequencies.
- Compute the Discrete Cosine Transform of the logarithm – The DCT decorrelates the components of the Mel-spectral vectors and converts the frequency domain into a time domain signal.

METHODS

While the Shazam algorithm is robust against filtering and some distortions, it is not robust against pitch-scaling or time stretching. Each problem was attacked individually in order to explore the effects of using the two different feature extraction methods discussed in the previous section. The indexing hash of the existing method was also modified to better represent the features extracted from each task.

Increasing Robustness to Pitch-Scaling

The constant Q transform with 36 bins per octave, or 3 bins per semitone, was used as the method for feature extraction in order to increase robustness to pitch-scaling. As with the original algorithm, pairs of spectral peaks are taken as landmarks but this time using the constant Q spectrogram instead of the FFT. Figure 2 shows a comparison of the landmarks obtained with the conventional spectrogram (above) vs. the constant Q spectrogram (below). On the conventional spectrogram, the landmark on the right is shrunken as the song's pitch is lowered; therefore the frequency difference is different for both landmarks. However, on the constant Q spectrogram the landmarks both have the same frequency difference despite the landmark having shifted down as the song's pitch is lowered.

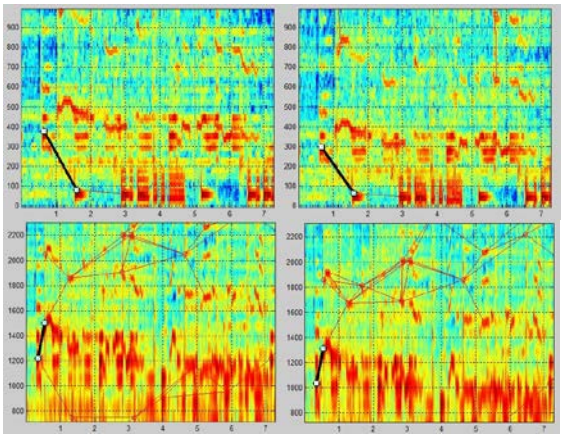

Figure 2
Comparison of the Spectrograms
The images on the left are the original query audio and the images on the right have the pitch lowered by 4 semitones.

The four-tuple representation of the landmark is preserved with the constant Q transform information. In this representation two songs that are identical except in the pitch should have the same value for *DF* for matching landmarks. Despite this advantage, the value of *F1* is still different with the pitch scaling. For this reason it is necessary to change the definition of the hash function.

In the original hash representation the most significant bits are comprised of *F1* which is the seed or reference frequency of the landmark. This presented a problem in identification when the song's pitch was changed because the frequencies are also changed in this situation. Nevertheless it is still important to maintain the information of this frequency so that two landmarks are not confused as being the same for having the same frequency difference *DF*. A relatively effective solution to this problem was to move the reference frequency to the least significant bits of the hash value. The resolution of *F1* was also lowered in order to fit into the last 5 bits of the hash. This was done by dividing by a power of 2 to shift the bits to the right. Fenet et al. also divide this frequency, by a value of 6, in order to obtain a sub-resolved version of *F1* that is invariant with the common pitch-shifting ratios of less than 5%. [9] Even with these changes the new hash value is still calculated with the same information of the landmarks but with the values rearranged. A comparison of the calculation of the hash function for each method is shown in Table 2.

Table 2
Comparison of the Original and Altered Hash Calculation

Original Hash Function	Altered Hash Function
H = F1*2^12 + DF*2^6 +DT	H = DF*(2^14)+DT*(2^6)+F1hat
F1 – reference frequency 8 bits DF – frequency difference 6 bits DT – time difference 6 bits	F1hat = F1/8 reference frequency 5 bits DF – frequency difference 6 bits DT – time difference 6 bits

While these changes helped increase robustness against pitch-scaling, they were only effective up to a change of 1 semitone in either direction which is about a 6% change in pitch. The query was "multiplied" by varying its pitch shift in order to increase robustness when the pitch change is more significant. This is achieved in the following steps:

- Get all the hash values between those that have the frequency bits as all 0 or all 1.
- Find the difference between the original hash and those with altered frequency.
- Group the hash values by their difference, discarding those which do not have the same amount of sub-hashes as the original hash.
- Find all the songs that match the original and pitch shifted query.

Figure 3 demonstrates a simplified example of this process. In this example the hash is only 8 bits long for demonstration purposes. In this example the first step will be obtaining all values between XXXXXX00 to XXXXXX11 which for 125 these are the values between 01111100 and 01111111 (124 ~ 127). The same is done for all the sub-hash values in the song hash. After the hash values are grouped by their difference we are left with two possible queries to find matches for, which results in the query being multiplied by two.

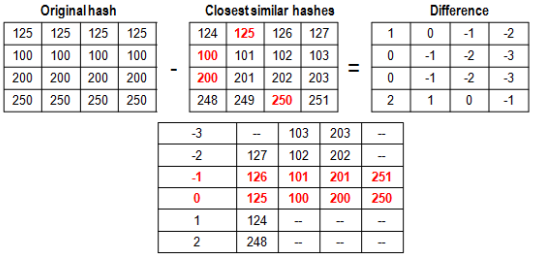

Figure 3
A Simplified Example

Multiplying the query is an attempt at obtaining all the possible matches based on the different pitch changes which can be done to the query audio. This final addition further helps in increasing the accuracy of the search when the query song has been pitch scaled. It is part of the process of redefining the identification of a match.

With these changes in place, it was also necessary to increase the amount of landmarks obtained for each song. There are two important factors that influence the number of landmarks, the density of landmarks and the spreading width of the masking skirt from each found peak. These factors affect the number of local maxima found which are then paired into peaks. The larger the value of the density and the smaller the value of the spreading width, the more landmarks are found. This results in more robust matching but with a greater load on the database.

Increasing Robustness to Time-Stretching

The initial proposition for resolving this problem was to use the same hash function as was modified for increased robustness to pitch-scaling. In this situation, the value of *DT* have a multiplicative change, as opposed to an additive change as is the case of *F1*, so increasing the query presents a bigger problem. We can try following the same steps as above but applying these changes:

- Get all the values between those that have the middle, or time difference (*DT*), bits as all 0 or all 1.
- Find the ratio between the original hash and those with altered *DT*.

Even with these considerations, this implementation did not yield encouraging results when the tempo of the query songs was changed. On the other hand using the MFCC did provide favorable results when the song was time stretched.

With the MFCC as radically different features from the landmarks, it was also necessary to redefine the hash. The redefinition of this hash is loosely based on the method proposed by Haitsma & Kalker [3] which is a process similar to calculating MFCC and delta MFCC features but with the extra step of determining whether a '1' or '0' bit is assigned depending on the sign of the energy difference. In order to be able to use the same indexing table, the hash was redefined the following way:

- The MFCC are calculated with a 64 ms window and 50% overlap returning 20 cepstra coefficients for each frame.
- A 20 bit hash is taken from the cepstra of each frame as follows:
  - Positive values are changed to 1.
  - All other values are 0.
  - The binary string is converted to an integer which is the hash value.

Using the MFCC as a feature does seem to increase robustness against time stretching attack. It is important to be cautious when using these features because MFCCs are not very robust against additive noise although some modifications have been suggested by researchers in order to increase this robustness.

Redefining the Match

In the original landmark-based approach, a match is obtained when there is a song found which has the most amounts of hash values equal to that of the query song. This means that the hash values that match have to be exactly the same, however the hash of altered songs will be different.

Since the pitch-scaling and time-stretching problems were both dealt with separately, each approach defines a match differently. In the first approach, the query was multiplied to account for the possible differences in pitch. On the other hand, the second approach using the MFCC did not require such a redefinition of the match because MFCC values would not change drastically when the query was time-stretched.

The original algorithm returns 20 possible matches to the query song. These songs are ranked based on how many landmarks match between the query and the reference. When the query is multiplied, there are 20 possible matches for every query considered. Returning to the hypothetical simplified example, there would be 40 possible matches for the query song since there are two possible queries. These 40 songs are then sorted by the modalDTcount and the song with the highest count is the accepted match to the query song.

In an attempt to combine both methods for simplicity in running the tests, the MFCC hash was also included among the multiplied queries. This results in an increased accuracy in identification when the query is either pitch-scaled or time-stretched. However this combination required two different hash tables in order to take into account the differences in features extracted and so both methods are still being treated as separate and only joined for the identification step.

RESULTS

Testing has been done using the Magnatagtune database [14] which contains a variety of songs from different musical genres. The songs come divided into multiple clips so the clips pertaining to the same song were pasted together in order to make up the search database. The clips, which are seconds long, were used as queries in order to ensure that the algorithm could in fact work with short queries.

For initial testing, 584 songs were used as the search dataset and 50 clips were used as query. The query clips were altered using Sound eXchange (SoX). The altered queries were tested with the original landmark-based approach and the altered approach. Table 3 contains a summary of the experimental results using the original code with the default parameters. Tables 4 and 5 contain a summary of the results of the experiments using the same parameters for the density and spreading width. This is to ensure that the amount of landmarks taken is not a factor that influences the comparison of the results.

After each test run, the top 20 possible matches are saved in separate files for each of the query songs tested. If the correct song is identified as the first match of the list, then it is considered to be a “perfect match”. If the correct song is not the first on the list but it is contained within the list of the top 20 songs, then it counted as being “within top 20”. The “within top 20” column does not account for “perfect matches”, therefore two dashes “--” were used when there was no count for “within top

20". Finally, if the correct song is not within the list of the top 20 possible matches then it is counted as an "incorrect match" or a false positive.

Table 3
Results with the Original Code Using the Default Parameters for Density and Spreading Width

	Perfect Match	Within Top 20	Incorrect Match
Unaltered query	100%	--	0%
Pitch raised 4 semitones	6%	10%	84%
Pitch raised 3 semitones	12%	12%	76%
Pitch raised 2 semitones	14%	14%	72%
Pitch raised 1 semitone	32%	16%	52%
Pitch lowered 1 semitone	34%	20%	46%
Pitch lowered 2 semitones	14%	16%	70%
Pitch lowered 3 semitones	10%	16%	74%
Pitch lowered 4 semitones	10%	8%	82%
Pitch lowered 10 semitones	0%	10%	90%
Tempo Change 20 %	32%	44%	24%
Tempo Change 15%	38%	40%	22%
Tempo Change 10%	68%	30%	2%
Tempo Change -10%	68%	24%	8%
Tempo Change -15%	52%	28%	20%
Tempo Change -20%	30%	38%	32%
Speed Change 5%	4%	6%	90%
Speed Change -5%	4%	8%	44%

The results on Tables 4 and 5 show that an improved accuracy was achieved with the modifications performed on the landmark-based algorithm. The pitch was changed up to 4 semitones in either direction because a larger change is easily recognizable by the human ear even without the original song for comparison. Even a change higher than 2 semitones is easily recognizable by the human ear but we still wanted to test the limits of the algorithm. The tempo was changed up to 20% for the same reason, to keep the query within the limits of the alterations being recognizable by the human ear.

The use of the CQT and the modified hash with the multiplied query allowed for identification of 98% of the queries when the query song's pitch was lowered down to 4 semitones. Even lowering the pitch down to 10 semitones still achieves a correct identification of 76%. Accuracy is much lower when the query's pitch has been raised but the results are still an improvement over the original algorithm. This decrease in accuracy when the pitch is raised is probably due to *F1* being limited to 5 bits which means larger frequency bins may be lost in the process if they cannot be properly represented in 5 bits.

The use of the MFCC as a feature also increases the identification when the query song's tempo has been altered although the unaltered algorithm does a good job with an increase in the amount of landmarks taken. The identification is 100% when the tempo is altered up to 15% and only drops to 98% when the tempo is altered up to 20%. Despite this good identification rate, the MFCC features alone are not enough to make the identification robust against noise and they are also not robust to pitch changes.

Table 4
Results with the Original Code Using Density of 20 and Spreading Width of 15

	Perfect Match	Within Top 20	Incorrect Match
Unaltered query	100%	--	0
Pitch raised 4 semitones	8%	28%	64%
Pitch raised 3 semitones	4%	36%	60%
Pitch raised 2 semitones	26%	18%	56%
Pitch raised 1 semitone	40%	14%	46%
Pitch lowered 1 semitone	42%	12%	46%
Pitch lowered 2 semitones	24%	20%	56%
Pitch lowered 3 semitones	14%	22%	64%
Pitch lowered 4 semitones	8%	26%	66%
Pitch lowered 10 semitones	2%	20%	78%
Tempo Change 20 %	70%	20%	10%
Tempo Change 15%	94%	4%	2%
Tempo Change 10%	100%	--	0%
Tempo Change -10%	100%	--	0%
Tempo Change -15%	86%	6%	8%
Tempo Change -20%	72%	14%	14%
Speed Change 5%	4%	38%	58%
Speed Change -5%	4%	40%	56%

Table 5
Results with the Altered Code Using a Density of 20 and Spreading Width of 15

	Perfect Match	Within Top 20	Incorrect Match
Unaltered query	100%	--	0%
Pitch raised 4 semitones	18%	24%	58%
Pitch raised 3 semitones	14%	32%	54%
Pitch raised 2 semitones	70%	4%	22%
Pitch raised 1 semitone	98%	--	2%
Pitch lowered 1 semitone	100%	--	0%
Pitch lowered 2 semitones	100%	--	0%
Pitch lowered 3 semitones	100%	--	0%
Pitch lowered 4 semitones	98%	--	2%
Pitch lowered 10 semitones	76%	--	24%
Tempo Change 20 %	98%	2%	0%
Tempo Change 15%	100%	--	0%
Tempo Change 10%	100%	--	0%
Tempo Change -10%	100%	--	0%
Tempo Change -15%	100%	--	0%
Tempo Change -20%	98%	2%	0%
Speed Change 5%	74%	14%	12%
Speed Change -5%	74%	14%	12%

A small speed change was also performed on the queries to verify how well the altered methods performed in this case. Correct identification increased from 4% to about 70% with the alterations. However the identification was done thanks to the alterations for better pitch identification and the usage of MFCC features did not help improve identification in the case of a speed change.

CONCLUSION

The modifications done to the algorithm considerably improved identification when the targeted modifications were performed on the query audio. While the modifications done for increasing robustness to pitch scaling work when the query's pitch has been lowered, the system loses accuracy as the pitch is raised. However, the modifications accomplish their purpose as songs that have the pitch altered too much become noticeable to the human ear thus completely distorting the song in a negative way. The use of MFCC features also proved to be effective for identifying queries that have been time stretched. This further demonstrates the invariance of MFCC features against tempo transformations.

Because the methods are not truly combined the accuracy is lost when the query has had the speed altered which means both a change in pitch and tempo. Running both methods at the same time with two hash tables also increases the time it takes to calculate the features as both the constant Q spectrogram and MFCC are more computationally expensive than calculation of a conventional spectrogram. Still, identification time is within 5 seconds in the Matlab environment and so is an acceptable trade off for the increased accuracy. Despite this, it is important to find a way to more efficiently combine both methods in order to increase speed and accuracy when a song's speed is changed.

ACKNOWLEDGEMENTS

The author would like to acknowledge Dr. Juan Torres for his contribution and guidance during the development of this work. The author would also like to offer thanks to Dr. Jeffrey Duffay and Prof. Arturo Geigel who contributed ideas and encouragement to continue with this research. Special thanks to Dr. Alfredo Cruz for the fellowship opportunity under contract/grant number NRC 27-10-511.

REFERENCES

[1] Gomes, L. d., Cano, P., Gómez, E., Bonnet, M., & Batlle, E. (2003). Audio Watermarking and Fingerprinting: For Which Applications? *Journal of New Music Research , 32* (1), 65-81.

[2] Lebossé, J., Brun, L., & Pailles, J. C. (2006). A Robust Audio Fingerprint for Digital Rights Management.

[3] Haitsma, J., & Kalker, T. (2002). A Highly Robust Audio Fingerprinting System. In *Proc. ISMIR* (Vol. 2, pp. 13-17).

[4] Cano, P., Batlle, E., Kalker, T., & Haitsma, J. (2005). A Review of Audio Fingerprinting. *The Journal of VLSI Signal Processing , 41*, 271-284.

[5] Ellis, D. (2009). *Robust Landmark-Based Audio Fingerprinting*. Retrieved from Laboratory for the Recognition and Organization of Speech and Audio - LabROSA: http://labrosa.ee.columbia.edu/matlab/fingerprint/

[6] Wang, A. (2003). An industrial strength audio search algorithm. In *Proc. Int. Conf. on Music Info. Retrieval ISMIR* (Vol. 3).

[7] Ellis, D. (2004). *Spectrograms: Constant-Q (Log-frequency) and conventional (linear)*. Retrieved from Laboratory for the Recognition and Organization of Speech and Audio - LabROSA: http://labrosa.ee. columbia. edu/matlab/sgram/

[8] Ellis, D. (2005). *PLP and RASTA (and MFCC, and inversion) in Matlab*. Retrieved from Laboratory for the Recognition and Organization of Speech and Audio - LabROSA: http://www.ee.Columbia.edu/~dpwe /resources /matlab/rastamat

[9] Fenet, S., Richard, G., & Grenier, Y. (2011). A Scalable Audio Fingerprint Method With Robustness to Pitch-Shifting. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*.

[10] Brown, J. (1991). Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America , 89*, 425.

[11] Logan, B. (2000). Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval* (Vol. 28, p. 5).

[12] Li, T., & Chan, A. (2011). Genre classification and the invariance of mfcc features to key and tempo. *Advances in Multimedia Modeling , 6523*, 317-327.

[13] Bala, A. (2010). Voice Command Recognition System Based on MFCC and DTW. *International Journal of Engineering Science and Technology*, 12, 7335-7342.

[14] Law, E., & Von Ahn, L. (2009, April). Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1197-1206). ACM.