A StegIbiza implementation using traditional digital signal processing techniques

Steganography in music through tempo modulation

Wai-Yip Tang
LIACS, Leiden University
s1384317

ABSTRACT

StegIbiza is a steganography technique that uses the constant tempo of certain styles of music to hide information. By varying the tempo in certain parts of the song, one can mask information. Inspired by the StegIbiza technique, this paper attempts to implement StegIbiza using traditional digital signal processing techniques and knowledge acquired from the Audio Processing and Indexing course. The results show that the applied methods weren't accurate enough to consistently and correctly decode the message. Furthermore, tempo modulation may not be a good method of hiding information. As there are many restrictions, such as a low message encoding throughput and the message being easily decodable.

1. INTRODUCTION

The term StegIbiza was first coined in the paper 'StegIbiza: New Method for Information Hiding in Club Music' by Krzysztof Szczypiorski[7]. StegIbiza is a combination of two words: Steganography and Ibiza. Steganography is the act of concealing information within other information and Ibiza is the island of whose style of music is used to hide the information within. Krzysztof was inspired by the hypnotic and trance inducing music of a certain style of club music. This style of club music came from what is considered by some as the Mecca of clubbing, namely Ibiza. The style of Ibiza's music is a melting pot of various styles with its main influence coming from Balearic House.

Krzysztof noticed that this style of music had a relative constant tempo with pronounced beats. He speculated that one could hide information within these clubbing songs by varying the tempo of the music. A change of tempo could be encoded as information. A change of lower to higher tempo and higher to lower tempo can respectively represent a positive or negative bit. This change of tempo can be detected by leaving the first part of the song untouched. This first part serves as a reference tempo for the remainder of the song. Preliminary research done by Krzysztof showed that a tempo change of under 1 percent is unnoticeable by any of the participants, a tempo change between 1 and 2 percent is only noticeable by experts and a tempo change above 2 percent is noticed by roughly half of the participants in the preliminary experiment.

Inspired by StegIbiza, this paper attempts to apply the StegIbiza technique using knowledge and techniques acquired from the Audio Processing and Indexing course, and research the feasibility of such techniques[1]. This paper consists of 5 main sections. The background and related work section elaborates on a recent python implementation of StegIbiza using state-of-the-art algorithms and its limitations. This will grant a good overview of what kind of results one can expect. The methodology section discusses on the steps taken to implement StegIbiza. The experiment section elaborates on the experimental set-up used to test the performance of the implementation of this paper. The results section will discuss the results from the experiment. The conclusion and discussion section will summarize and discuss the results of the experiments.

2. BACKGROUND AND RELATED WORK

In 2017, Krzysztof and Wojciech Zydecki published a paper detailing the results of applying the StegIbiza technique in Python using a state-of-the-art digital signal processing library named MADMOM[8][2]. Krzysztof and Wojciech used the MADMOM tempo estimation functions to track the tempo. These functions in the MADMOM library used a small Recurrent Neural Network(RNN) to do the beat tracking necessary for tempo estimation. This RNN is based on the paper 'Enhanced Beat Tracking with Context-Aware Neural Networks' by and Sebastian Böck and Markus Schedl[4].

The results of the python implementation of StegIbiza showed that a satisfactory result can be obtained on songs that have a near constant tempo throughout the whole song. It also, however, showed some severe limitations. First, songs that didn't have a constant tempo yielded a wrong decoding. Second, at least 10 seconds excluding 5 percent of the start and end of the song is needed to estimate the tempo, which results in a very low throughput of information that can be encoded. Third, the state-of-the-art wasn't able to accurately detect a tempo change of 2.75 percent or smaller. Though, through the use of the resultant tempo values from MADMOM's resonating comb filters, Krzysztof and Wojciech could determine if there was a positive or negative tempo change with reasonable accuracy.

Given these limitations, a few conditions can be established. First, the signal doesn't have to be processed in real-time. There is at least a 10 second window in which one can buffer the signal, as this is the shortest window of time needed to accurately estimate the tempo using the state-of-the-art. Second, even with the use of the state-of-the-art, a tempo change of smaller than 2.75 wasn't detectable. Therefore,

the use of the same resonating comb filters originating from MADMOM, may be useful. As one can make use of the resultant tempo's.

3. METHODOLOGY

This section elaborates on the implementation of StegIbiza. The implementation of StegIbiza in this paper consists of 3 main phases: Signal preprocessing, peak extraction and tempo estimation, and finally message decoding.

3.1 Signal preprocessing

The signal preprocessing phase prepares the original signal in a format suitable for tempo estimation. This phase consists of 3 main steps.

1. **Splitting the signal**: From the original signal, the first and last 5 percent of the signal is discarded. These parts are discarded to avoid issues with songs that have a slow tempo or that are silent in the beginning or end. The signal is converted to an mono signal through Librosa. Afterwards, the signal is split into chunks of 10 seconds, which is the minimum time needed for tempo estimation. A thing to note is that the WAV-file meta info should specify how many bits it is, in this experiment, the files were all 16-bit. This is due to the fact that Librosa convert the signal amplitude into a real number in the interval [-1, 1] instead of an integer.

2. **Stretching the chunks**: Each of the chunks are time stretch by a factor of 0.99 or 1.01 using the Librosa `timestretch` function[5]. This function stretches the time duration of the audio by a factor of 1 percent and thus indirectly modulates the tempo to be respectively slower or faster. A thing to note is that depending on the length of a piece, tempo modulation could potentially have a different effect, as the stretching is relative to the length of a piece due to the usage of percentages.

3. **Short Time fourier Transform(STFT)**: The time stretched chunks are split into 'frames' using the `Framed Signal` function from MADMOM with a 'Hamming' window function and overlapping window to reduce leakage of information after a Fast Fourier Transform (FFT). This leakage occurs due to the discrete nature STFT and the periodicity of a signal, which causes a loss of information at the ends of a rectangular window. SFTF is applied on the framed signal using the `ShortTime Fourier Transform` from MADMOM. This function converts the signal from the time domain into the frequency domain for further analysis. This step is not performed for the Librosa method of tempo estimation, as the Librosa function does something similar automatically.

3.2 Peak extraction and tempo estimation

The peak extraction and tempo estimation discusses the various tempo estimation methods used to estimate the tempo of each processed 10 second chunk from phase 1.

3.2.1 Librosa tempo estimation

Librosa tempo estimation uses the Ellis algorithm which uses a dynamic programming approach in tracking beats[6]. First, the strength of the onsets are measured. An onset is a swell of energy in the signal within a short amount of time. The onset can indicate a beat or other high energy events. Depending on the duration of an onset, its peak and shape. One can estimate various characteristics of the audio content, such as the tempo, rhythm or genre.

Second, after measuring the onset strength, the global tempo is estimated from the onset autocorrelation. Autocorrelation is useful for revealing inherently regular and periodic structures. Autocorrelation finds the correlation between itself and a delayed version of itself, as shown in figure 1. In this case, it find a function between an onset and a time lag between an onset. This is done by taking the inner product of an onset envelope with a delayed version of itself. Places where the peak line up will result in high correlation value. Using these values, one can filter and fit how many peak occurs within a set period of time and thus estimate a tempo. The last step is to pick the peaks using the globally estimated tempo to construct a transition cost function. This step of the algorithm is not of importance in this paper, as the interest lies in the estimated global tempo.

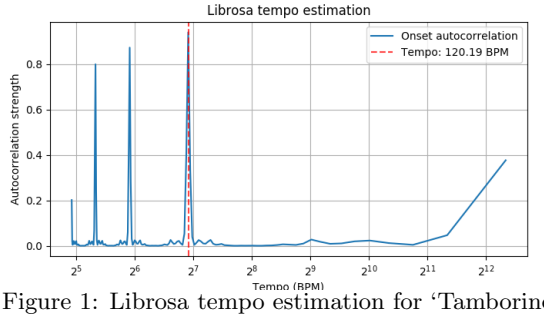

Figure 1: Librosa tempo estimation for 'Tamborine'

3.2.2 Spectral flux and resonating comb filters

This method uses spectral flux and resonating comb filters to estimate the tempo. First, a magnitude spectrogram is created from each 10 second chunk using the `Spectrogram` function from MADMOM. The spectrogram illustrates the energy relationship between the frequency bins and time, as shown in figure 2. The peaks can be seen clearly as vertical green dots.

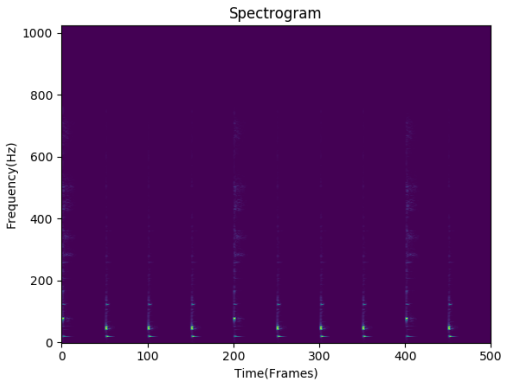

Figure 2: Spectrogram of 'Tamborine'

Afterwards, this spectrogram is used to calculate spectral flux using the `spectral_flux` function from MADMOM. Spectral flux measures the rate of change in a signal. Spectral flux calculates and compares the power spectrum of one frame against its previous frame. It can detect trends of successive increases or decreases in a power spectrum, its slope. These characteristics are useful in determining the start and end of an onset. The spectral flux application is shown in figure 3. The regular peaks can be seen very clearly, including the stronger click every fourth peak.

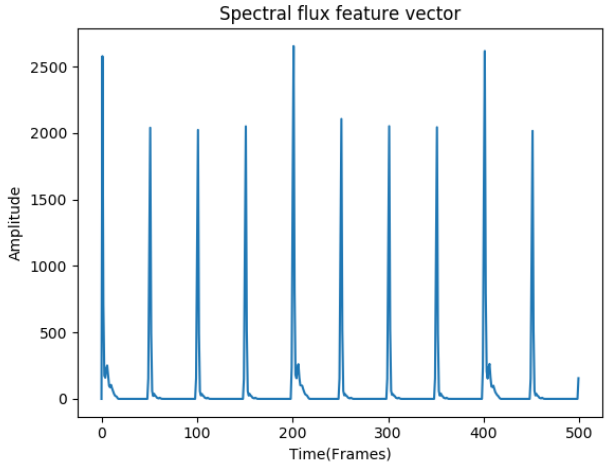

Figure 3: Spectral flux of 'Tamborine'

Lastly, the spectral flux is used as input features for the resonating comb filters to estimate the tempo. The resonating comb filters work very similarly to autocorrelation, it finds periodicities in a signal. Both methods try to estimate the tempo through fitting delayed versions of itself onto itself. The resonating comb filters attempts this through forwarding the input features through a bank of resonating comb filters with different time lags according to formula 1. The difference and advantage of comb filters over autocorrelation is that comb filters can resonate at multiples and fractions. This characteristic emulates the auditory perception behaviour of humans.

$$y(n) = x(n) + \alpha * y(n - \tau) \tag{1}$$

Where n is the signal, α is the scaling factor and τ the delay length. Both α and τ are determined using the knowledge acquired from the paper 'Accurate Tempo Estimation based on Recurrent Neural Networks and Resonating Comb Filters'[3]. *alpha* is set to 0.79, whilst the lag range definition of τ is determined using formula 2 and 3.

$$tMin = \left\lfloor 60 * \frac{fps}{bpm_{max}} \right\rfloor \tag{2}$$

$$tMin = \left\lfloor 60 * \frac{fps}{bpm_{min}} \right\rfloor \tag{3}$$

Where fps is the number of 'frames' or 'windows' in one second of the signal. Where bpm_{min} and bpm_{max} are respectively 40 and 240, defining the BPM interval of which to search within.

3.3 Message decoding

This paragraph discusses the message decoding phase. Message decoding can be done using two simple methods.

1. **Comparing Beats Per Minute(BPM) values**: Methods that result in BPM values are directly compared with each other. The BPM value for the first chunk is the reference BPM. All other BPM values are compared to the reference BPM. If the reference BPM is lower, then this chunk was encoded as a 1. If the reference BPM is higher, then this chunk was encoded as a 0. An encoding of 1 indicates a faster time stretch and an encoding of 0 indicated a slower time stretch.

2. **Comparing resultant BPM values**: Methods that result in multiple estimates of BPM are compared using the method as described in the python implementation paper in section 2. First, all the estimate tempos between the reference chunk and one other chunk are compared. If the absolute difference between the estimate tempo from the reference and other chunk is larger than 5 percent, then this comparison is discarded. Afterwards, each percentile difference between all the comparisons are summed up. A sum that is positive or equal to 0 indicates that the chunk was encoded with a 1 and 0 otherwise.

4. EXPERIMENT

The experiment is set up as follows:

1. **First**: The WAV-file is read in and the first 10 seconds of the song is discarded. This is slightly different than discarding 5 percent, but fulfills the same purpose as most songs are between 2 to 3 minutes long. Afterwards, the chunks are used for encoding, this excludes the first chunk which serves as the reference. This requires the WAV-file to be at least 110 seconds long. To allow flexibility in experimentation, the number of chunks is determined by formula 4.

$$numberOfChunks = 100/duration \tag{4}$$

   Where duration is how long each chunk should be.

2. **Second**: Each of the 10 chunks are time stretched according to their encoding. 1 indicates that the 10 second chunk is stretched with a factor of 0.99 and 0 with a factor of 1.01. The encoding is determined randomly using a random number generator with a seed of 2019 for repeatability. Afterwards, the methods from section3.2 are applied to each WAV-file.

3. **Third**: Five different WAV-files are used in this experiment. The first file is a consistent tick with on the fourth tick a louder tick. It has a BPM of 120. The name of this file is 'tamborine 120 bpm'. The second

file is an Ibiza style club song named 'Night owl' by Broke For Free. The third file is a classical piece called 'Four seasons' by Vivaldi played by violinist Janine Jansen, it contains the Spring Movement 1 (Allegro). Similar to many classical pieces it has a not so static tempo. The last two files are k-pop song called 'Want' by Lee Taemin and 'Really Bad Boy' - by Red Velvet. 'Want' is somewhat more consistent with a hypnotic beat similar to an Ibiza style club song, whilst 'Really Bad Boy' is much more chaotic with random screams. For each WAV-file, the experiment is repeated for a set amount of random encodings for consistency purposes. The amount of repetitions is determined using formula 5.

$$reps = 100/numberOfChunks \tag{5}$$

The amount of repetitions are increased or decreased proportionally to the `numberOfChunks`. This allows for more robust results, as a lesser amount of chunks equates to a more variable result. The average is taken over the random encodings for each song to acquire an accuracy score. An additional repeat of this experiment is done for each song, but this time for 5 chunks with a duration 20 seconds. This experiment is done to research the effect of tempo modulation through time stretching, as in theory, a time stretch over a longer piece will exacerbate the tempo modulation effect and thus could potentially yield more stable results.

5. RESULTS

The results of the experiments are shown in the boxplots below. The light green stripe is the median. The ends of the green box indicate the first and third quartile. The white dot indicates 'outliers'.

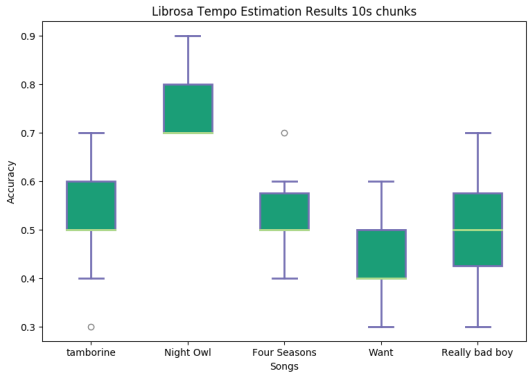

Figure 4: Librosa tempo estimation results for 10s chunks

Figure 4 shows that the Ibiza style music performed the best by a large margin. A surprising result is that 'tamborine' with its constant ticking performed so bad. It also seems that most of the other songs seem to gather around 50 percent accuracy. It seems that Librosa tend to, most of the time ,overestimate the tempo, which causes a high false negative rate.

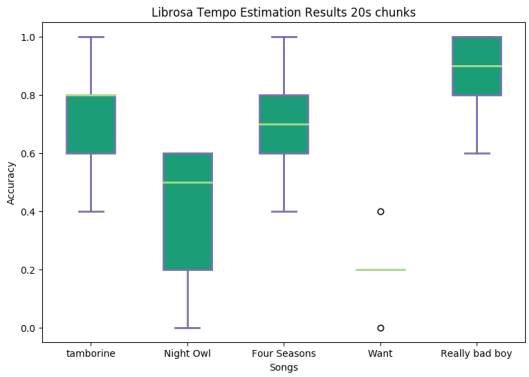

Figure 5: Librosa tempo estimation results for 20s chunks

Figure 5, which shows the 20 second chunk variant has different results compared to the 10 second chunk version. 'Really Bad Boy' seem to perform very well. Same can be said for 'Four Seasons', which is unexpected as these songs are quite varied in their tempo. The graphic result of 'Wan' can be explained due to the fact that the quartiles and the lowest and highest result are the same.

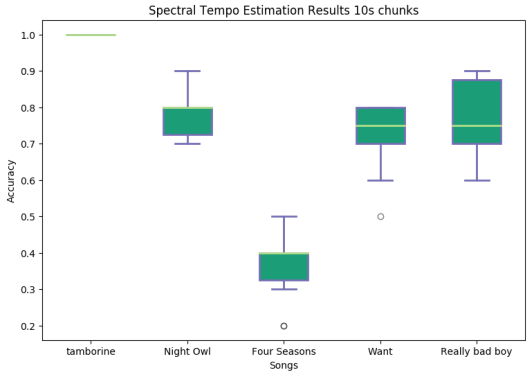

Figure 6: Spectral flux tempo estimation results for 10s chunks

Figure 5 shows that it performed perfectly for the 'tamborine' with its ticking sounds. Furthermore, it performed quite well for all the other songs, except for 'Four Seasons' which has a very irregular tempo.

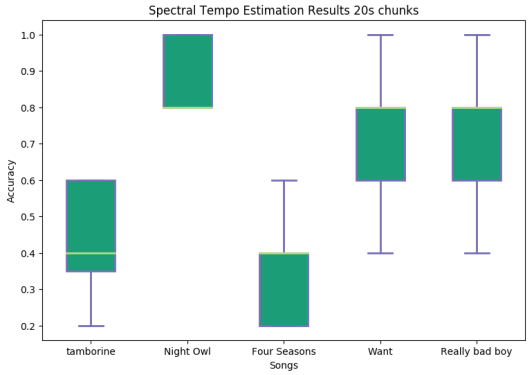

Figure 7: Spectral flux estimation results for 20s chunks

Figure 5 shows that there is a degradation in performance. This is unexpected, it seems that for this experiment that a longer period to estimate BPM yielded worse results. There also seems to be a higher variance between the estimated BPMs for each song.

Songs	Librosa Tempo Estimation	Spectral flux and Comb filters Tempo estimation
Tamborine	0.53 +/-0.12	**1.0 +/- 0.0**
Night Owl	0.55 +/- 0.08	**0.79 +/- 0.07**
Four Seasons	**0.51 +/- 0.16**	0.37 +/- 0.10
Want	0.65 +/- 0.08	**0.72 +/- 0.10**
Really Bad Boy	0.74 +/- 0.08	**0.77 +/- 0.10**

Table 1: The experimental results for the 10 second chunks. The average accuracy and its Standard Deviation(std)

The result in table 1 shows that the spectral flux method outperforms the Librosa method in all but for 'Four Seasons'.

Songs	Librosa Tempo Estimation	Spectral flux and Comb filters Tempo estimation
Tamborine	**0.53 +/-0.16**	0.42 +/- 0.15
Night Owl	0.55 +/- 0.18	**0.88 +/- 0.10**
Four Seasons	**0.49 +/- 0.22**	0.38 +/- 0.14
Want	0.46 +/- 0.22	**0.74 +/- 0.17**
Really Bad Boy	**0.74 +/- 0.16**	0.73 +/- 0.15

Table 2: The experimental results for the 20 second chunks. The average accuracy and its Standard Deviation(std)

The result in table 2 shows inconsistent results for the 20 second chunks. The Spectral flux method seems to perform well for the more stable tempo music types. A thing to note is that 'Tamborine' performed very bad, which is surprising. This could be due to a bad cut-off of the beat. This indicates that spectral flux captures more than just the beat to estimate the tempo.

6. CONCLUSION AND DISCUSSION

The results presented in section 5 shows that a chunk with a longer duration seems to give worse results for almost all songs, with the exception of 'Night Owl' with its Ibiza style music. The usage of spectral flux also seems to outperform the Librosa method. The results also show that the methods applied in this paper aren't accurate enough to consistently decode the message. Furthermore, though StegIbiza is a creative application of steganography, it has many limitation such as a low encoding throughput and being limited to music genres with a constant tempo. The algorithm behind this method is also not very secure, making this an 'ugly' steganography method. Therefore, usage of other steganography methods is recommended.

7. REFERENCES

[1] E. M. Bakker. Seminar audio processing and indexing, 2018. Retrieved from http://liacs.leidenuniv.nl/ bakkerem2/api.

[2] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016.

[3] S. Böck, F. Krebs, and G. Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *ISMIR*, 2015.

[4] S. Böck and M. Schedl. Enhanced beat tracking with context-aware neural networks. In *in Proceedings of the 14th International Conference on Digital Audio Effects (DAFx-11*, 2011.

[5] B. McFee, C. A. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. 2015.

[6] D. P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36:51–60, 03 2007.

[7] K. Szczypiorski. Stegibiza: New method for information hiding in club music. *CoRR*, abs/1608.02988, 2016.

[8] K. Szczypiorski and W. Zydecki. Stegibiza: Steganography in club music implemented in python. *CoRR*, abs/1705.07788, 2017.