# Digital Audio Watermarking by Magnitude Modification of Frequency Components Using the CSPE Algorithm.

Jian Wang (jwang@cs.nuim.ie), Ron Healy (rhealy@cs.nuim.ie), Dr. Joseph Timoney (jtimoney@cs.nuim.ie)

**ABSTRACT:**

**In this paper we describe a process whereby the magnitude of either one or two frequency components of a signal is modified in order that it may be used to encode a hidden message within a signal in such a way as the casual observer would have no way of noticing the presence of a hidden message. Previous work has used filtering and signal addition to achieve the same goals. The current work improves on this by using a recent super-resolution component-identification technique to isolate the components to modify, limiting the impact on the quality of the signal.**

Keywords: *Signal processing, digital audio watermarking, data hiding, Steganography*

## 1.0 INTRODUCTION

The concept of *Steganography*, defined as "*the art or practice of concealing a message, image, or file within another message, image, or file*" [1] is not new. Steganography may be combined with *Cryptography* in order to make message data more secure even if the presence of the message is discovered. Digital watermarking of audio and video is a form of Steganography, in that the audio/video can be used to 'hide' the presence of other information.

In recent years there has been a marked increase in research in the area of digital watermarking. This has been driven, in part, by the needs of the Entertainment Industry to find means for protecting, tracking or identifying intellectual property such as photographs, music and movies. The SDMI (The *Secure Digital Music Initiative*, a group consisting of more than 200 companies in the fields of I.T., Music and Entertainment, Consumer Electronics, Security and Internet Service Providers) challenge at the turn of the century, with regard to digital music, contributed to much investigation into the area of digital watermarking over the intervening years. Eventually, the SDMI folded, claiming that it was awaiting developments in technology before implementing digital rights management technologies. One of the reasons identified for the SDMI's failure was that the technologies then available were insufficient to achieve the aim of completely hiding an added watermark from those expert or talented listeners described as 'golden ears'. This meant that there was no way of preventing detection and ultimate removal of the watermark. The watermarking technology that the SDMI purported to recommend to the Industry was broken almost immediately [2].

There have been a number of alternative propositions for hiding data in cover signals and most are successful to a certain extent or in a given context. A good overview of the theories in this area can be found in [3]. The basic premise of watermarking schemes is that the information to be watermarked $w$ is added or embedded in the cover or host signal $s$ to produce a watermarked signal $s'$

$$s + w = s' \tag{1}$$

This paper proposes a technique for hiding data in cover audio signals, specifically music or spoken word, by the identification and modification of the magnitude of frequency components in the cover signal itself.

In part, the work is inspired by [4], a technique designed for covert communications across a radio channel for military applications, and follows on from an earlier work which used the addition of multiple frequency components to achieve a similar aim [5]. In [5] it was proposed that the message to be embedded was to be separately generated. This was then added to the host or cover audio. In this paper, however, we instead propose that the host or cover is itself modified in a controlled manner, rather than having potentially destructive and/or detectable content added to it. In both this paper and [5], the primary concern is for inaudibility of the watermark and blind or semi-blind detection, meaning that the decoder does not have any knowledge of either the content of the cover audio or of the embedded watermark prior to decoding. This restriction is guided by the intended use of the technology.

In this paper, we present the results of experiments performed to recover a bit sequence which was embedded in a synthesised cover audio signal consisting of randomly generated components. The decoding was performed

without any reference to the original unwatermarked signal or the watermark itself.

## 2.0 METHOD

A component value is first chosen which is used as the basis for calculating which components to modify to hide the message. The initial component choice may be dependent on various factors, such as the type of audio used as host/cover. For example, human speech generally consists of lower frequency components – and less of them – than a modern Rock or Pop song so hiding data in a recording of speech would naturally limit the component of choice. However, even in such a limited range, there are still thousands of values to choose from.

The value of the chosen component becomes, in effect, a private key and this value is needed in order to decode the watermark – assuming that the *presence* of the watermark has previously been detected. This adds to the security of the technique when used in an environment where *security of the content* of the hidden message is an issue.

The signal intended as the cover or host audio is segmented into frames of uniform length and the frame is then analysed using 'Complex Spectral Phase Estimation' (CSPE) techniques [6] to identify the presence and magnitude of its inherent components. Previously, FFT techniques have been used to approximate the relative strengths of inherent components. This would be inadequate for this project, as exact measurement of components using the FFT is only possible if the component is aligned with an analysis bin. This is an unlikely occurrence in a real-world signal such as recorded music or speech. Therefore, the FFT is an inadequate solution to the problem of identifying exactly the components present.

### 2.1 CSPE INTRODUCTION AND DESCRIPTION

The CSPE algorithm was introduced as a method to accurately estimate the frequency of components that exist within a short time frame. It was also designed to be computationally efficient. It is actually related in some aspects to the cross-spectrogram technique of [7]. The principal of CSPE algorithm can be described as follows:

An FFT analysis is performed twice: firstly on the signal of interest and the second time upon the same signal but shifted in time by one sample. Then, by multiplying the sample-shifted FFT spectrum with the complex conjugate of the initial FFT spectrum, a frequency dependent function is formed from which the exact values of the frequency components it contains can be detected. The procedure of the CSPE algorithm is depicted in block diagram form in Figure 1.

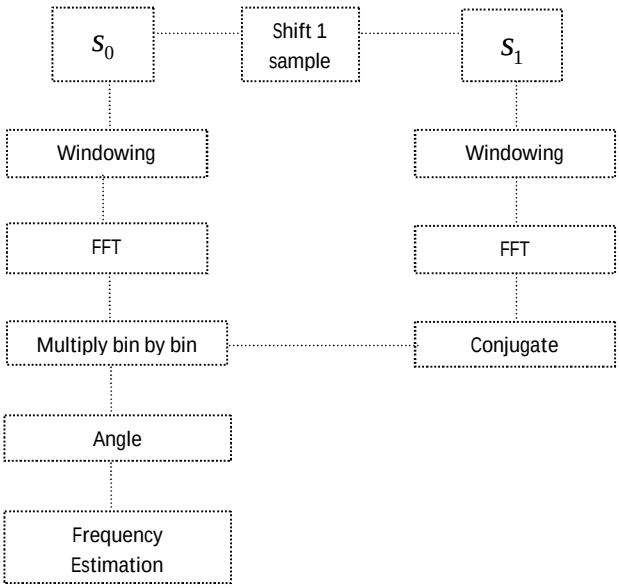


**Fig. 1: The flow diagram of CSPE**

Mathematically, the algorithm can be described as follows. Assume a real signal $s_0$, and a one-sample shifted version of this signal $s_1$. Say that its frequency is $\beta = q + \delta$ where $q$ is an integer and $\delta$ is a fractional number. If $b$ is an initial phase, $w_n$ is the window function used in the FFT, $F_{ws_0}$ is the windowed Fourier transform of $s_0$, and $F_{ws1}$ is the windowed Fourier transform of $s_1$, then, from [6], we find

$$D = e^{\frac{j\,2\,\pi\beta}{N}} \tag{2}$$

The frequency dependent CSPE function can be written as

$$CSPE_w = F_{ws_0} F^*_{ws_1} = (\frac{a}{2})^2 \begin{bmatrix} D^* \left\| F_w(D^n) \right\|^2 \\ + 2\,\mathrm{Re}\left\{ e^{j2b} D F_w(D^n) \otimes F_w^*(D^{-n}) \right\} \\ + D \left\| F_w(D^{-n}) \right\|^2 \end{bmatrix} \tag{3}$$

The windowed transform requires multiplication of the time domain data by the analysis widow, and thus the resulting transform is the convolution of the transform of the window

function, $w_f$, with the transform of a complex sinusoid. Since the transform of a complex sinusoid is a pair of delta functions in the positive and negative frequency positions, the result of the convolution is merely a frequency-translated copy of $w_f$ centred at $+\beta$ and $-\beta$. Consequently, with a standard windowing function, the $\|F_w(D^n)\|$ term is only considerable when $k \approx \beta$ and it decays rapidly when $k$ is far from $\beta$. Therefore, the analysis window must be chosen carefully so that it decays rapidly to minimize any spectral leakage into adjacent bins. If this is so it will render the interference terms, i.e. the second and third terms, to be negligible in Eq.(3). Thus, the CSPE for the positive frequencies gives:

$$CSPE_w \approx \frac{a^2}{4}\left\|F_w(D^n)\right\|^2 D^{-1} \tag{4}$$

From Eq. (4). we find the CSPE frequency estimate

$$\begin{aligned} f_{CSPE_w} &= \frac{-N\angle(CSPE_w)}{2\pi} \\ &= \frac{-N\angle\left(\frac{a^2}{4}\left\|F_w(D^n)\right\|^2 D^{-1}\right)}{2\pi} \\ &= \frac{-N\angle(\frac{a^2}{4}\left\|F_w(D^n)\right\|^2 e^{-j\frac{2\pi}{N}\beta})}{2\pi} = \frac{-N(-\frac{2\pi}{N}\beta)}{2\pi} = \beta \end{aligned} \tag{5}$$

The frequency dependent function as illustrated in Equation (4) produces a graph with a staircase-like appearance where the flat parts of the graph indicate the exact frequencies of the components. The width of the flat parts is dependent on the main-lobe width of window function used to select the signal before FFT processing. An example of the output of the CSPE algorithm is shown in Figure 2. Consider the signal $S_1$ which contains components with frequency values (in Hz) of 17, 293.5, 313.9, 204.6, 153.7, 378 and 423. The sampling frequency is 1024 HZ. A frame of 1024 samples in length is windowed using a Blackman window and is padded using 1024 zeros. The frequency dependent CSPE function is computed as per Equation (5). As shown in Figure 2, each component can be calculated and these are identified with an arrow in the graph. The largest error among all the estimates of the components frequencies is approximately 0.15 Hz.

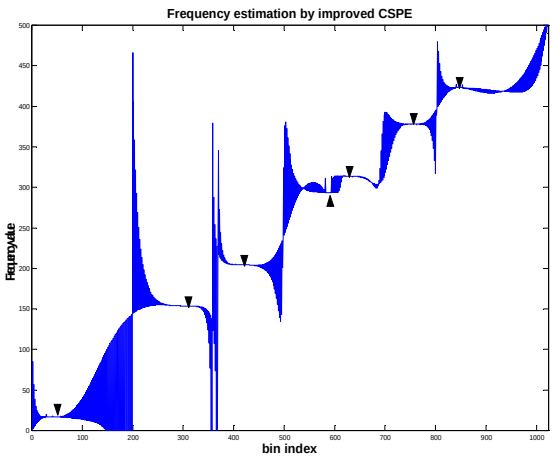


**Fig. 2 Frequency estimation of $S_1$ by CSPE**

Notice too in Figure 2 that at the flat sections in the graph of the CSPE result, the width of flat sections where the arrows point are related to the width of the window's main-lobe in the frequency domain.

In addition, with CSPE, we can get the amplitude and phase of the *k*th frequency component using the following equations, where *W(ω-fcspe(k))* is the Fourier Transform of window function which has been shifted to *fcspe(k)* in frequency domain.

$$Amp_k = \left\|\frac{2 * F_{ws_0}}{W(\omega - fcspe(k))}\right\| \tag{6}$$

$$Phase_k = \angle\left(\frac{2 * F_{ws_0}}{W(\omega - fcspe(k))}\right) \tag{7}$$

## 2.2 EXPERIMENTAL EVALUATION OF CSPE

Experiments were designed to evaluate the performance of the CSPE algorithm in correctly identifying frequency components within a multiple-component signal. In each set of experiments, a total of 500 signals with Sampling Frequency 44100 Hz and containing components across the human hearing range of 100 Hz to 20,000 Hz were generated. Each signal contained many equally spaced frequency components. The number of components in each generated signal was not consistent. For each individual signal, we have a unique, randomly-generated step constant which defines the space between two neighbouring frequency components of the signal. 500 step constants were created range from 169 Hz to 668 Hz for 500 signals. Equation (8) and (9) were designed to assess CSPE accuracy in frequency estimation.

Denoting $Freq_{estk}$ as the value of estimated Frequency components of signal $k$ ; $Freq_{orgk}$ as the value of original Frequency components of signal $k$; $M_k$ as the number of frequency components contained in Signal $k$; *FreqError* as the frequency estimation error between $Freq_{est}$ and $Freq_{org}$ of signal $k$; $MeanError_{cspe}$ as the mean error of the CSPE frequency estimation over $N$ signals, for this experiment, $N$ = 500, $M$ changes with signal step constant.

$$FreqError_k = \frac{\sum_{i=1}^{M_k} \left| Freq_{estk}(i) - Freq_{orgk}(i) \right|}{M_k} \quad (8)$$

The frequency estimation error of each signal as computed using Equation (8) is shown as Figure 3:

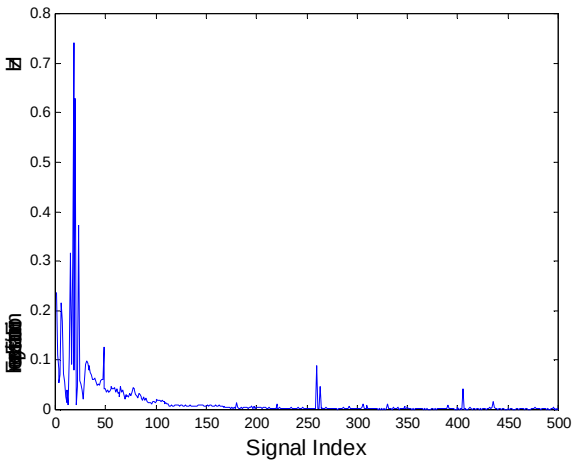


**Fig. 3 CSPE Estimation Error for Each Signal**

The distribution of frequency estimation error (*FreqError)* is shown in Figure 4.

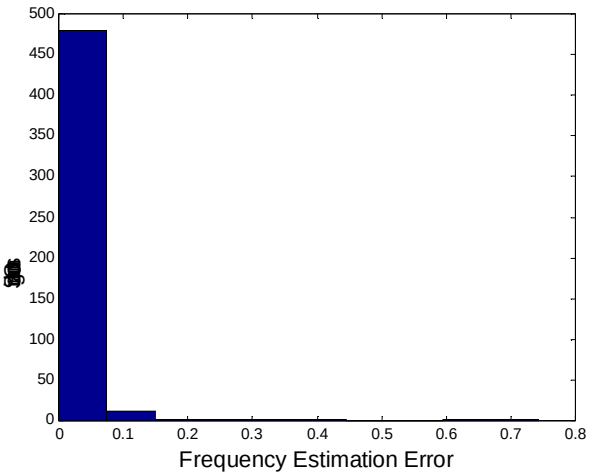


**Fig. 4 The distribution of Frequency Estimation Error**

The mean error is calculated according to Equation （9）

$$MeanError_{cspe} = \frac{\sum_{k=1}^{N} FreqError_k}{N} \quad (9)$$

By data analysis, we note that 97.8% of signals analysed using the CSPE algorithm resulted in a *FreqError* value of less than 0.1 Hz, and the $MeanError_{cspe}$ is 0.0174 Hz, meaning that the algorithm identified the component to within 0.1 Hz in almost all cases. We conclude from these results that the CSPE is extremely accurate in frequency estimation for signals containing constant frequency signal components. With accurate estimation of the frequency, the amplitude and phase can be estimated using Eqs. (6) and (7).

## 3.0 MODIFYING COMPONENTS

Once the user-defined component has been identified in the signal by the CSPE algorithm, its magnitude is then calculated. It is then a matter of modifying the magnitude of this component, weighting it against a second value from within the signal, in order to represent a single bit ‘1’ or ‘0’.

We may choose to weight the user-defined components against the average power of the frame in which the bit is to be embedded. This was the procedure followed in both [4] and [5]. We may also choose to modify the user-defined component against a second component. This method has its advantages and disadvantages but it is not our intention to detail the process in this paper. However, using a second component from within the signal as a comparison against which the first user-defined component was weighted, led to some problems in that, while the CSPE algorithm is very accurate in identifying the components in a synthesised signal with little variation, this may not be the same type of component make-up as would be encountered in real world signals, such as audio and speech.

### 3.1 DYNAMICALLY SELECTING COMPONENTS

We decided to make the process of choosing the component(s) to modify as flexible as possible by making this a dynamically chosen pair of values, dependent on the user-defined value but also dependent on the signal under consideration and reliant on the ability of the CSPE algorithm to detect and identify the components that the watermarking process would use. We defined the components which would be chosen for modification as

being the nearest components above and below the user-defined value by more than a calculated threshold as illustrated in Equation (10) where *compA* is the highest CSPE-detected frequency component that is lower than the user-defined component *u*, by more than the threshold *k* while *compB* is a CSPE-detected frequency component above the user-defined component by the same threshold amount

$$(compA < (u - k)) < u < (compB > (u + k)) \quad (10)$$

What is interesting to note, using the formula in Equation (10) for defining which component we need to modify, and in which frames of the cover signal, is that only approximately half of the frames will require any modification. This is because the relationship between the values of the two chosen components in any given frame may already fit the criteria used for representing a '1' or a '0'. In this case they would not have to be modified in any way. This consideration makes this method far more favourable than [5],

When modifying the amplitude of a frequency component, care must be taken to ensure that we do not introduce any noticeable artefact which would result in an impact on sound quality. Similarly, we must ensure that the alteration we make to the magnitude of the chosen component is not so great as to have a negative impact on the timbre of the original signal.

We define a set of rules that would lead to the modification of only one of the components (*compA* or *compB*) in approximately half the frames. This is achieved by setting the rule (Amp refers to Amplitude)

If *bit*=1 let Amp(*compA)* > Amp (*compB*) + *margin*

If *bit*=0 let Amp (*compB*) > Amp(*compA)* + *margin*

The system would then compare the magnitude of both components (*compA* and *compB*) in any given frame before deciding if any modification would be required in order to satisfy these criteria, depending on the bit to be embedded and the magnitudes of the two components in that particular frame. If they are already in the correct relationship, no modification is required. If, however, they are not in the correct relationship, we must modify at least one of them. The decision to modify a component leads another question. Let us assume that the magnitude of *compA* is lower than that of *compB*, in a frame in which it needs to be of a higher magnitude to represent a '1' bit.

3.2 MODIFYING THE MAGNITUDE

As mentioned in Section 2.0.2, the CSPE algorithm can be used to accurately identify a component within a signal, and then to calculate its phase and amplitude. In order to increase the magnitude of a particular frequency component in the cover signal $S(t)$, we add a component at a defined magnitude and matched to the phase of the component it is being combined with, as illustrated in Equation (11):

$$S(t)= S(t)+(rAmp-lAmp+threshold)\cos(2\pi(compA)t+lp) \quad (11)$$

where *rAmp*, *lAmp*, *compA* and lp define amplitude of *compB*, amplitude of *compA*, Frequency of *compA*, phase of *compA*.

Similarly, if we decide to reduce the magnitude of a component $S(t)$ so that it satisfies the requirements for embedding a '1' bit, we do this by reducing the magnitude of the component to the right of the user-defined component value, by adding in a component that is $180^{o}$ out of phase with the original component in the signal as follows:

$$S(t)= S(t)+(rAmp-lAmp+threshold)\cos(2\pi(compB)t+ \pi-rp) \quad (12)$$

where *compB* and *rp* define amplitude and phase of *compB*.

## 4.0 DECODING

In order to process candidate audio for detection and decoding of a potential embedded watermarked message, the system must first be provided with the user-defined value used as a basis for calculating the embedding values, along with the rules that define a '1' bit and a '0' bit. The candidate audio signal is then segmented into frames using the same frame size as was used for embedding. The system calculates the magnitude of the embedded component, and performs a simple comparison. From this comparison the watermarked bit sequence can be recreated. It would be a comparatively simple matter of applying the CSPE algorithm to identify the two components above and below the user-defined value by more than a pre-defined threshold. These two components would then have their magnitude compared and a '1' or a '0' bit would be determined according to the rules used in their embedding.

## 5.0 EVALUATION OF WATERMARKING SCHEME

A series of experiments was carried out to evaluate the performance of this codec, based on the same 500 signals as introduced in Section 2.2. For each signal, a randomly generated binary bit-sequence of length 150 was embedded by means of modification of the magnitude of components as described in Section 3. The system then decoded the modified signal in order to detect the watermarked code.

The difference between these two code sequences can be calculated in terms of equation below, where *DCode* denotes code sequence obtained in decode side, *ECode* denotes code sequence embedded in the signal. *CodecPrecision* denotes the precision of the decode process with code length *L* for signal *k*, *MeanPrecision* denotes average error of the decode process over *N* signals. In this experiment, *L* and *N* are set to 150 and 500 respectively. The results of this experiment are depicted in Figure 5.

$$CodecPrecision_k = \frac{L - \sum_{i=1}^{L} |DCode(i) - ECode(i)|}{L} \quad (12)$$

$$MeanPrecision = \frac{\sum_{k=1}^{N} (CodecPrecision_k)}{N} \quad (13)$$

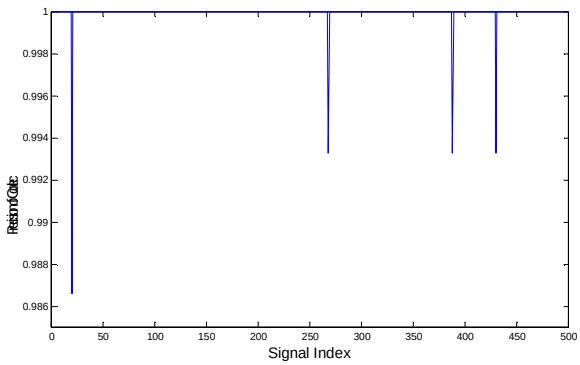


**Fig. 5 Precision of Codec for each Signal**

The distribution of *CodecPrecision* is shown in Figure 6.

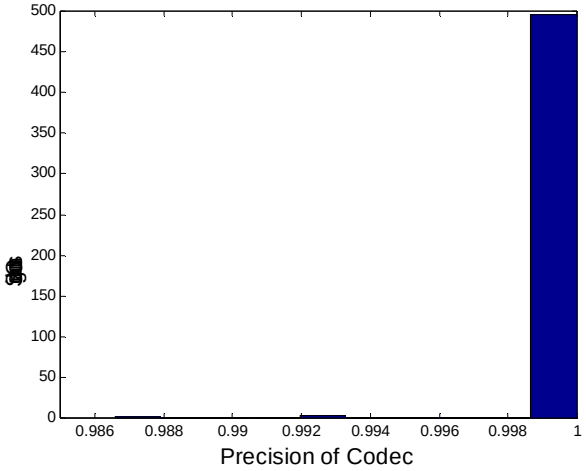


**Fig. 6 The distribution of Precision of Codec**

From the experiment results, it can be seen that 99.2% of signals produce a *CodecPrecision* value of 1 (100%). This means that, from 500 randomly generated signals with multiple components of different frequency spacing, watermarked with a binary bit-sequence of 150 bits, 99.2% of these signals were decoded to the *exact* 150 bit sequence. Only 0.8% (a total of 4) of the 500 signals was not decoded perfectly. Of those not *perfectly* decoded, the bit sequence recovery rate was above 98.66%. The *MeanPrecision* computed using Equation (13) is 0.9999 (99.99%). Therefore, the performance of this codec is almost perfect for this experiment with the synthesised signals.

Furthermore, the decode experiment in this case represented a single iteration of a bit sequence over the length of a signal. Given that any real world use of such a scheme would enable a bit sequence to be embedded repeatedly in a cover signal, it would be possible to increase the effectiveness of the decode process by, for example, repeated decoding and using the *mode* of the results.

## 6.0 CONCLUSION

We have proposed an application that utilises the super-resolution capabilities of the CSPE algorithm to accurately identify individual components of an audio signal, calculate their magnitudes and then alter magnitude as appropriate to represent a particular bit value.

Experimental tests using 500 synthesised signals incorporating multiple randomly generated components embedded with a bit sequence of length 150 showed an

accuracy of completely perfect decoding of 99.2% with an average overall accuracy of 99.999%.

Future work will determine how to calculate and set the magnitude so signal watermarking is perceptually invisible, by evaluating whether to modify the component to the left or right of the user-defined frequency value, or both.

Also, the impact of accidental and deliberate attacks on the watermarked signal will be evaluated.

## 7.0 REFERENCE


[1] *Merriam-Webster Online Dictionary*.

http://www.merriam-webster.com/dictionary/steganography

[2] S. A. Craver, M. Wu, and B. Liu, "Reading between the lines: Lessons from the SDMI challenge," in *10th USENIX Security Symposium.* Washington, DC, 2001.

[3] Moulin, P., & Koetter, R., "Data-Hiding Codes", *Proc. Of the IEEE, Vol. 93, No. 12, Dec. 2005.*

[4] Gopalan, K., et al, 'Covert Speech Communication Via Cover Speech By Tone Insertion', *Proc. of the 2003 IEEE Aerospace Conference*, Big Sky, MT, March 2003.

[5] Healy, R. & Timoney, J. 'Digital Audio Watermarking with Semi-Blind Detection For In-Car Music Identification' *Audio Engineering Society 36th International Conference, Michigan, USA. June 2-4, 2009* (in press)

[6] K. M. Short and R. A. Garcia, "Signal Analysis using the Complex Spectral Phase Evolution (CSPE) Method", *Audio Engineering Society 120th Convention, May 2006, Paris, France*

[7] Douglas Nelson, "Cross Spectral Methods for Processing Speech", *Journal of the Acoustic Society of America, vol. 110, No.5, pt.1, Nov.2001, pp.2575-2592*

[8] The online *Webster* Dictionary. http://www.webster-dictionary.net/definition/interpolation