Echo Hiding

Daniel Gruhl Anthony Lu
Walter Bender Massachusetts Institute of Technology Media Laboratory

Abstract. Homomorphic signal-processing techniques are used to place information imperceivably into audio data streams by the introduction of synthetic resonances in the form of closely-spaced echoes. These echoes can be used to place digital identification tags directly into an audio signal with minimal objectionable degradation of the original signal.

1 Introduction

Echo hiding, a form of data hiding, is a method for embedding information into an audio signal. It seeks to do so in a robust fashion, while not perceivably degrading the host signal (cover audio).[1] Echo hiding has applications in providing proof of the ownership, annotation, and assurance of content integrity. Therefore, the data (embedded text) should not be sensitive to removal by common transforms to the stego audio (encoded audio signal), such as filtering, re-sampling, block editing, or lossy data compression.

Hiding data in audio signals presents a variety of challenges, due in part to the wider dynamic and differential range of the human auditory system (HAS) as compared to the other senses. The HAS perceives over a range of power greater than one billion to one and a range of frequencies greater than one thousand to one. Sensitivity to additive random noise is also acute. Perturbations in a sound file can be detected as low as one part in ten million (80dB below ambient level). However, there are some "holes" available in this perceptive range where data may be hidden. While the HAS has a large dynamic range, it often has a fairly small differential range. As a result, loud sounds tend to mask out quiet sounds. Additionally, while the HAS is sensitive to amplitude and relative phase, it is unable to perceive absolute phase. Finally, there are some environmental distortions so common as to be ignored by the listener in most cases.

A common approach to data hiding in audio (as well as in other media) is to introduce the data as noise. A drawback to this approach is that lossy data compression algorithms tend to remove most imperceivable artifacts, including typical low dB noise. Echo hiding introduces changes

[1] The adjectives cover, embedded, and stego were defined at the Information Hiding Workshop held in Cambridge, England. The term "cover" is used to describe the original signal. The informatio (data) to be hidden in the cover signal was defined to be the "embedded" signal. The "stego" signal is the signal containing both the "cover" signal and the "embedded" information. The word signal can be replaced by more descriptive terms such as audio, text, stills, video, etc.

to the cover audio that are characteristic of environmental conditions rather than random noise, thus it is robust in light of many lossy data compression algorithms.

Like all good stegonagraphic methods, echo hiding seeks to embed its data into data stream with minimal degradation of the original data stream. By minimal degradation, we mean that the change in the cover audio is either imperceivable or simply dismissed by the listener as a common non-objectionable environmental distortion.

The particular distortion we are introducing is similar to resonances found in a room due to walls, furniture, etc. The difference between the stego audio and the cover audio is similar to the difference between listening to a compact disc on headphones and listening to it from speakers. With the headphones, we hear the sound as it was recorded. With the speakers, we hear the sound plus echoes caused by room acoustics. By correctly choosing the distortion we are introducing for echo hiding, we can make such distortions indistinguishable from those a room might introduce in the above speaker case.

Care must be taken when adding these resonances however. There is a point at which additional resonances severely distort the cover audio. We are able to adjust several parameters of the echoes giving us control over both the degree and type of resonance being introduced. With carefully-selected parameter choices, the added resonances can be made imperceivable to the average human listener. Thus, we can exploit the limits of the HAS's discriminatory ability to hide data in an audio data stream.

2 Applications

Protection of intellectual property rights is one obvious application of any form of data hiding. Echo hiding can place a digital signature redundantly throughout an audio data steam. As a result, a reasonable level of hidden information is maintained even after operations such as extracting or editing. This information can be, but is not limited to, copyright information. With redundantly placed copyright information, unauthorized use of protected music becomes easy to demonstrate. Any clipped portion of an stego audio will contain a few copies of the digital signature (i.e. copyright information). Even "sound bites" distributed over the internet can be thus protected. Before placing an original sound bite on a web site, the creator can quickly run the Echo Hiding encoder. The creator can then periodically send out a web crawler which decodes all sound bites found, and reporting if the given signature is in them. For such applications, detection and modification of the embedded text must be limited to only a select few. The embedded text is only for the benefit of the encoder and is of little use to the end user. We would like it to be immune to removal by unauthorized parties. With the correct parameters, echo hiding can place the data with a very low probability of unauthorized interception or removal.

Another application of audio data hiding is the inclusion of augmentation data. In most cases, this type of data is placed for the benefit of the end user. As such, detection rules are more lenient. Since the data is there for the benefit of all, malicious tampering of the data is less likely. Echo hiding can be used to non-objectionably hide data in these scenarios also. We can place the augmentation data directly into the cover audio in a binary format. One benefit of our technique is that annotations normally require additional channels for both transmission and storage. By hiding the annotations as echoes in the cover audio, the number of required channels can be reduced.

While the inclusion of augmentation data does not require strict control over detection by third parties, echo hiding provides a low interception rate as an option. The uses of augmentation data include closed-captioning (of radio signals and CD's, etc.) and caller-id type applications of telecommunications systems. With echo hiding, the sound signal could contain both the audio information and the closed-captioning. A decoder can then take that signal and output the audio or display the captioning.

More interesting examples are caller-id and secure phone lines. We can use echo-hiding techniques to place caller information during a phone call. A decoder on the receiving end can detect this information revealing who the caller is and displaying other supplemental data (i.e., client information, client history, location of caller, etc.). The information is attached to the callers voice and is independent of the phone or phone service used. In contrast, current caller-id schemes only reveal the number of the device from which the call is placed. With echo hiding, it is possible to attach the information directly to the voice. As such, we have a form of voice identification and voice authentication. This can be useful in large conference calls when many people may try to talk, and identification of the current speaker is difficult due to low bandwidth. Phone calls which require a high degree of assurance of the identity of either party (e.g. oral contracts between an agent and employer) can also benefit from this application of echo hiding.

Echo hiding can also be useful to companies dealing with assuring that audio is played, for example radio commercials. For instance, when a radio station contracts to play a commercial, it can be difficult to know with certainty that the commercial is indeed being played as frequently as contractually agreed upon. Short of hiring someone to listen to the stations 24 hour a day, there is little one can do. Using echo hiding, we can place a "serial number" in the commercial. A computer can be set up to "listen" to the radio station, check for the identification number, and keep a tally of the number of times the commercial was played and how much of it was played (played in its entirety, cut off half way through, etc.). Echo hiding can also be useful when a radio station is multi-affiliated. Given similar commercials by two different companies, the radio station is by law required to play the tape given by each company in order to count for advertising by each company. This holds true even if the commercials are identical. By encoding each commercial using echo hiding techniques, the companies can keep track of which commercial is played. We can encode identical commercials with a different signature for each company.

Finally, tamper-proofing (prevention of unauthorized modification) can also be accomplished using echo hiding. A known string of digital identification tags can be placed throughout the entirety of the cover audio. The stego audio can easily be checked periodically for modified and/or missing tags revealing the authenticity of the signal in question.

3 Signal Representation

In order to maintain a high quality digital audio signal and to minimize degradation due to quantization of the cover audio, we use the 16-bit linearly quantized Audio Interchange File Format (AIFF). Sixteen-bit linear quantization introduces a negligible amount of signal distortion for our purposes, and AIFF files contain a superset of the information found in most currently popular sound file formats. Various temporal sampling rates have been used and tested, including 8 kHz, 10 kHz, 16 kHz, 22.05 kHz, and 44.1 kHz. Our methods are known to yield an acceptable embedded text recovery accuracy at these sampling rates.
Embedded text is placed into the cover audio using a binary representation. This allows the greatest flexibility with regards to the type of data the process can hide. Almost anything can be represented as a string of zeroes and ones. Therefore, we limit the encoding process to hiding only binary information.

4 Parameters

Echo Data Hiding places embedded text in the cover audio by introducing an "echo." Digital tags are defined using four major parameters of the echo: initial amplitude, decay rate, "one" offset, and "zero" offset (offset + delta) (Figure 1). As the offset (delay) between the original and the echo decreases, the two signals blend. At a certain point the human ear hears not an original signal and an echo, but rather a single distorted signal. This point is hard to determine exactly. It depends on the quality of the original recording, the type of sound being echoed, and the listener. In general, we find that this fusion occurs around one thousandth of a second for most sounds and most listeners.
The coder uses two delay times, one to represent a binary one ("one" offset) and another to represent a binary zero ("zero" offset). Both delay times are below the threshold at which the human ear can resolve the echo and the cover audio as different sources. In addition to decreasing the delay time, we can also ensure that the distortion is not perceivable by setting the echo amplitude and the decay rate below the audible threshold of the human ear.

5 Encoding

The encoding process can be represented as a system which has one of two possible system functions. In the time domain, the system functions

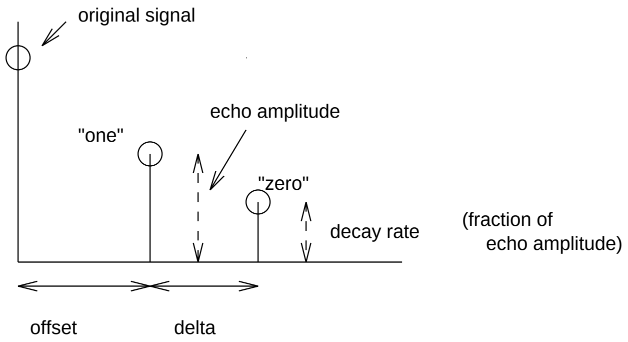

Fig. 1. Adjustable parameters

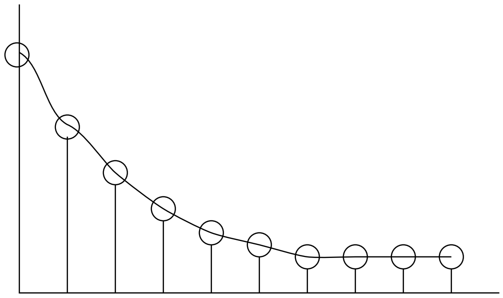

Fig. 2. Discrete time exponential

we use are discrete time exponentials (as depicted in Figure 2) differing only in the delay between impulses.

In this example, we chose system functions with only two impulses (one to copy the cover audio and one to create an echo) for simplicity.

We let the kernel shown in Figure 3(a) represent the system function for encoding a binary one, and we use the system function defined in Figure 3(b) to encode a zero. Processing a signal with either system function will result in an encoded signal (see example in Figure 11).

The delay between the cover audio and the echo is dependent on which kernel or system function we use in Figure 4. The "one" kernel (Figure 3(a)) is created with a delay of δ_1 seconds while the "zero" kernel (Figure 3(b)) has a δ_0 second delay. In order to encode more than one bit, the cover audio is "divided" into smaller portions. Each individual portion can then be echoed with the desired bit by considering each as an independent signal. The stego audio (containing several bits) is the recombination of all independently encoded signal portions.

In Figure 5, the example signal has been divided into seven equal portions labeled a, b, c, d, e, f, and g. We want portions a, c, d, and g to contain a one. Therefore, we use the "one" kernel (Figure 3(a)) as the system function for each of these portions i.e. each is individually convolved with the appropriate system function. The zeroes encoded into

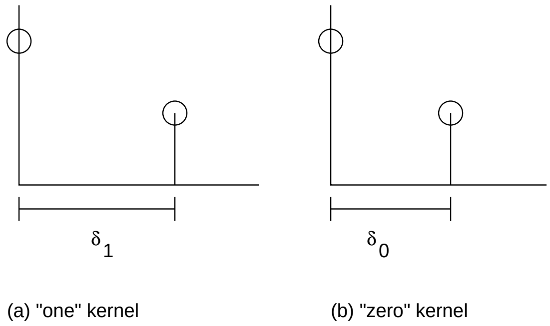

Fig. 3. Echo kernels

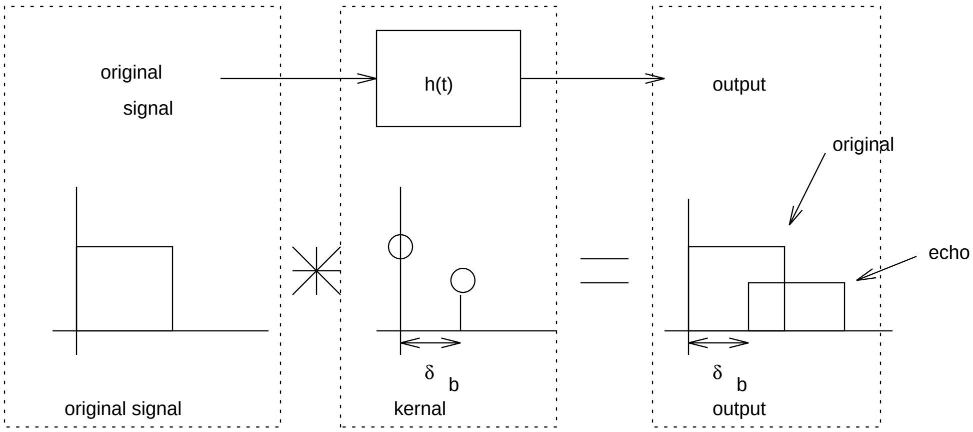

Fig. 4. Echoing example

sections b, e, and f are encoded in a similar manner using the "zero" kernel (Figure 3(b)). Once each section has been individually convolved with the appropriate system function, the results are recombined. While this is what happens conceptually, in practice we do something slightly different. Two echoed versions of the cover audio are created using each of the system functions. This is equivalent to encoding either all ones or all zeroes. The resulting signals are shown in Figure 6.

In order to combine the two signals, two mixer signals (Figure 7) are created. The mixer signals are either one or zero (depending on the bit we would like to hide in that portion) or in a transition stage in-between sections containing different bits.

The "one" mixer signal is multiplied by the "one" echo signal while the "zero" mixer signal is multiplied by the "zero" echo signal. In other words, the echo signals are scaled by either 1 (encode the bit) or 0 (do not encode bit) or a number in-between 0 and 1 (transition region). Then the two results are added. Note that the "zero" mixer signal is the binary

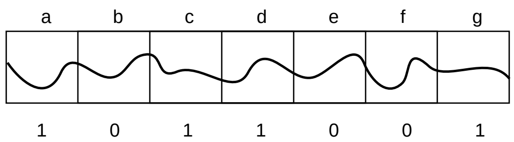

Fig. 5. Divide the cover audio into smaller portions to encode information

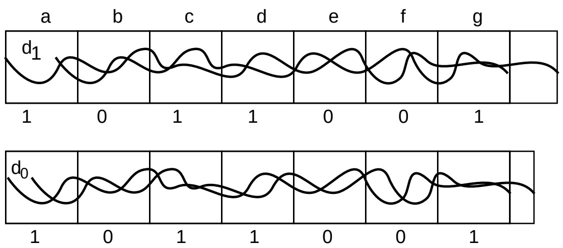

Fig. 6. First step in encoding process

inverse of the "one" mixer signal and that the transitions within each signal are ramps. Therefore, the resulting sum of the two mixer signals is always unity. This gives us a smooth transition between portions encoded with different bits and prevents abrupt changes in the resonance of the stego audio, which would be noticeable. A block diagram representing the entire encoding process is illustrated in Figure 8.

6 Decoding

Information is embedded into an audio stream by echoing the cover audio with one of two delay kernels as discussed in Section 5. A binary one is represented by an echo kernel with a δ_1 second delay. A binary zero is represented with a δ_0 second delay. Extraction of the embedded text involves the detection of spacing between the echoes. In order to do this, we examine the magnitude (at two locations) of the autocorrelation of the encoded signal's cepstrum (Appendix B). The following procedure is an example of the decoding process. We begin with a sample signal which is a series of impulses such that the impulses are separated by a set interval and have exponentially decaying amplitudes. The signal is zero elsewhere (Figure 9).
We echo the signal once with delay δ using the kernel depicted in Figure 10. The result is illustrated in Figure 11.
The next step is to find the cepstrum (Appendix A) of the echoed version. Taking the cepstrum "separates" the echoes from the original signal. The echoes are located in a periodic fashion dictated by the offset of the given bit. As a result, we know that the echoes are in one of two possible locations (with a little periodicity).

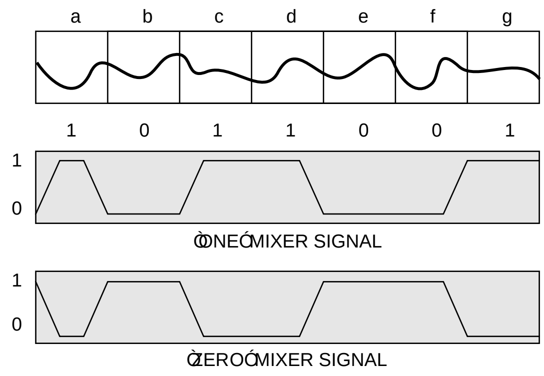

Fig. 7. Mixer Signals

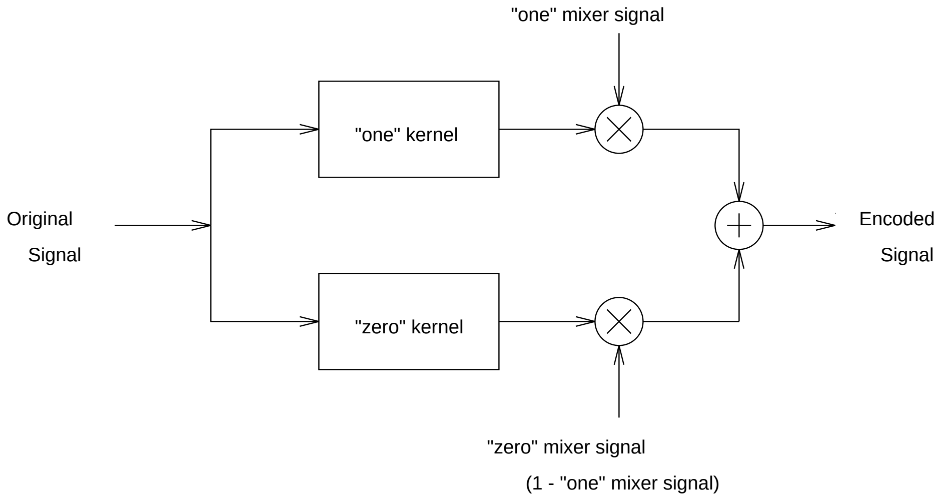

Fig. 8. Encoding process

Unfortunately, the result of the cepstrum also "duplicates" the echo every δ seconds. In Figure 12, this is illustrated by the impulse train in the output. Furthermore, the magnitude of the impulses representing the echoes are small relative to the cover audio. As such, they are difficult to detect. The solution to this problem is to take the autocorrelation of the cepstrum.

The autocorrelation gives us the power of the signal found at each delay. With the echoes spaced periodically every δ_1 or δ_0, we will get a "power spike" at either δ_1 or δ_0 in the cepstrum. This spike is just the power (energy squared) at echo spacings of δ_1 or δ_0. The decision rule for each bit is to examine the power at δ_0 and δ_1 in the cepstrum and choose whichever bit corresponds to a higher power level (see Figure 13).

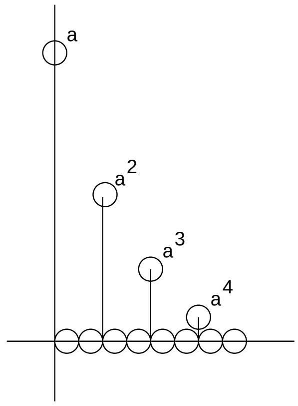

Fig. 9. Example signal: $x[n] = a^n u[n]$; $0 < a > 1$

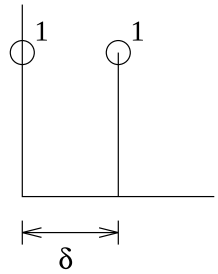

Fig. 10. Echo kernel used in example

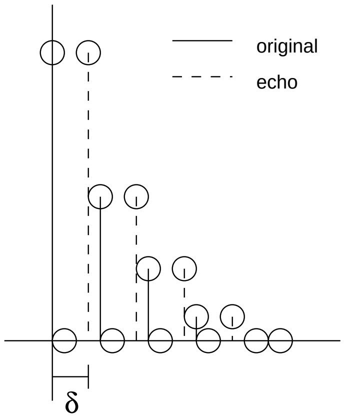

Fig. 11. Echoed version of the example signal

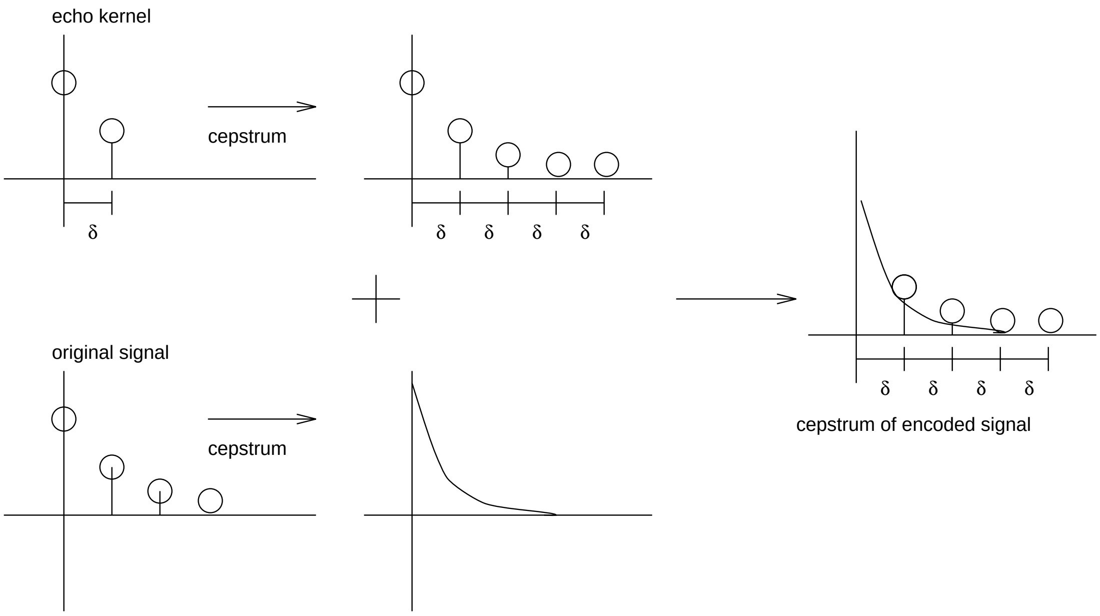

Fig. 12. Cepstrum of the echo-encoded signal

7 Results

Using the methods described, we can encode and decode information in the form of binary digits in an audio stream with minimal degradation at a data rate of about 16 bps[2] By minimal degradation, we mean that the output of the encoding process is changed in such a way that the average human cannot hear any objectionable distortion in the stego audio. In most cases the addition of resonance gives the signal a slightly richer sound.

Using a series of sound clips provided by ABC Radio, we have obtained encouraging results. The sound clips cover a wide range of sound types including music, speech, a combination of both, and sporadic sound (music or speech separated by empty space or noise). We created a tool to test these clips over a wide range of parameter settings in order to characterize the echo hiding process. Running the characterizations on 20 sound clips of varying content and length, we discovered that the relative volume of the echo (decay rate) was the most important parameter with regards to the embedded text recovery rate. With 85% chosen as a minimally acceptable recovery rate (defined in Equation 1) all stego signals showed acceptable accuracy with a decay rate (relative volume of the echo compared to the original signal) between 0.3 and 0.85.

[2] This is dependent on sampling rate and the type of sound being encoded. 16bps is a typical value, but the number can range from 2bps-64bps.

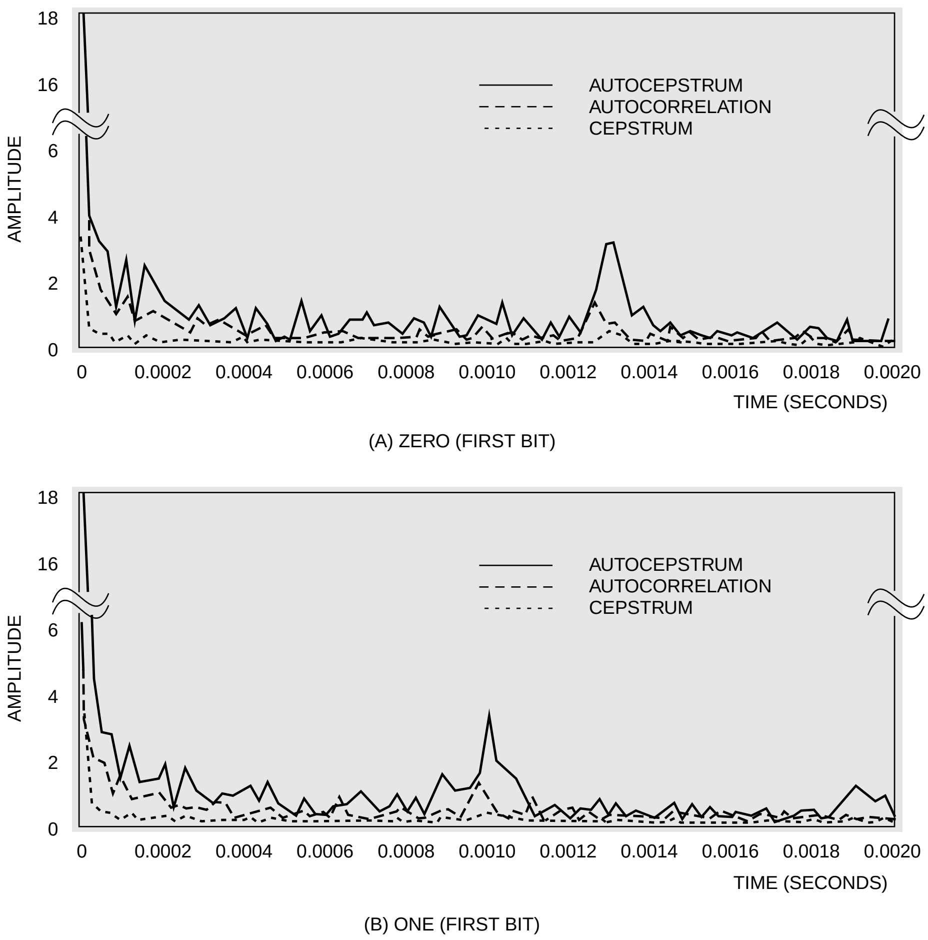

Fig. 13. Result of autocorrelation

$$recovery\ rate = \frac{(number\ of\ bits\ correctly\ decoded) * 100}{number\ of\ bits\ placed} \qquad (1)$$

At 0.5 and 0.6, few can resolve the echoes. While these results are encouraging, we would like to push the relative volume down even more. Between 0.3 and 0.4 even those with exceptional hearing have difficulty noticing a difference. We observed that in general the recovery rate was linearly related to the relative volume. However in certain cases, we observed deviations from this general rule, caused by the particular structure of the specific sound signal. Figures 14 through 17 illustrate the correlation (for three select files) between relative volume and embedded text recovery rate. The sound files chosen are representative of the entire set of sound clips. For the plots provided in this paper, the sample most amenable to encoding by Echo Hiding (**a6**, a segment of popular music), the sample least amenable to encoding (**a1**, a spoken news broadcast), and one mid-range sample (**a14**, spoken advertising copy) were used. In general, the more difficult samples are typically the ones with large "gaps" of silence (similar to **a1**, the example of unproduced spoken word) while those easiest to encode are those without such "gaps" (similar to example **a6**, the popular music clip).

Initially, we tested the process in a closed-loop environment (encoding and decoding from a sound file). The results are illustrated in Figure 14. All the files reached the 85% mark with relative volumes less than or equal to 0.8. **a6** required a relative volume of only 0.3 to recover an acceptable number of bits. By 0.4, we were able to recover 100% of the hidden bits. **a1** and **a14** required a higher relative volume of 0.5 in order to achieve the 85% mark.

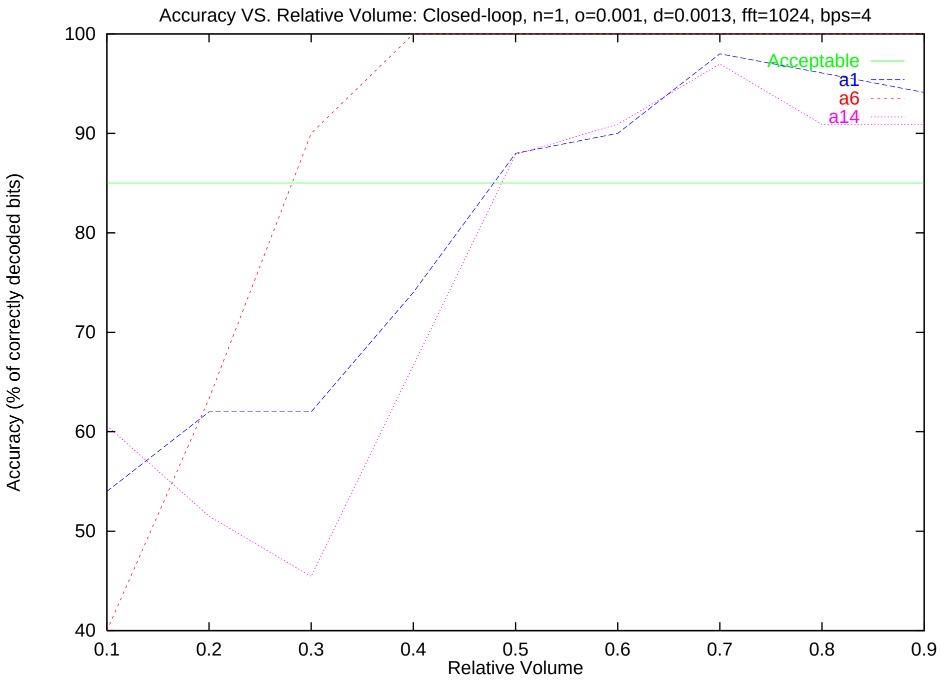

Fig. 14. Accuracy vs. relative volume: closed-loop

We also tried encoding on one machine, transmitting the sound file over an analog wire (with appropriate D/A and A/D conversions, and decoding on another machine (Figure 15). The required relative volume of **a14** increased to 0.8. Both **a1** and **a14** experienced a noticeable decrease in accuracy at higher relative volumes, but an acceptable recovery rate could still be reached. **a6** was approximately the same except that the 100% mark was not reached until 0.5.

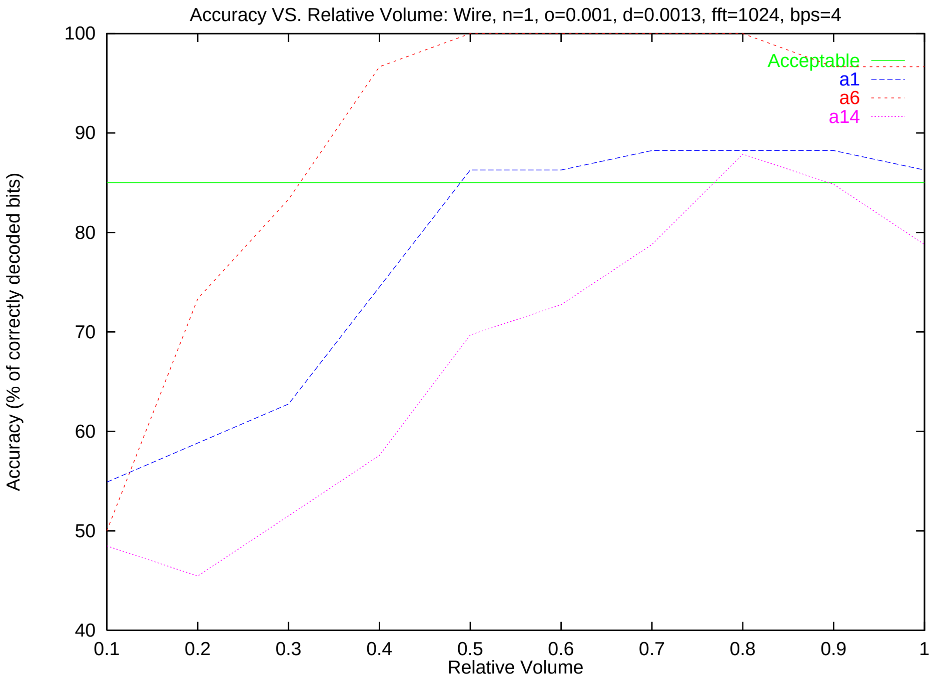

Fig. 15. Accuracy vs. relative volume: Analog wire

After testing an analog connection between two machines, we experimented with compression and decompression before decoding. We used two compression methods: MPEG (Figure 16) and SEDAT (Figure 17). The SEDAT compression was done with a test fixture provided by ABC Radio. In both cases, the recovery rate of **a1** and **a14** significantly decreased. **a6** was only slightly effected by the compression and decompression.

The other parameters (number of echoes, offset, and delta), seemed to produce acceptable results regardless of their value. This does not, by any means, indicate that these parameters are useless. Instead, these parameters play a significant role in the perceivability of the synthetic resonances. These interactions are in some cases highly non-linear, and better models of them are an area of continuing research. As discussed earlier (Section 4), a smaller offset and delta result in an increased "blending" of the resonances with the cover audio making it increasingly difficult for the human observer to resolve the echo and the cover audio as two distinct signals. Offsets greater than 0.5 milliseconds produced acceptable

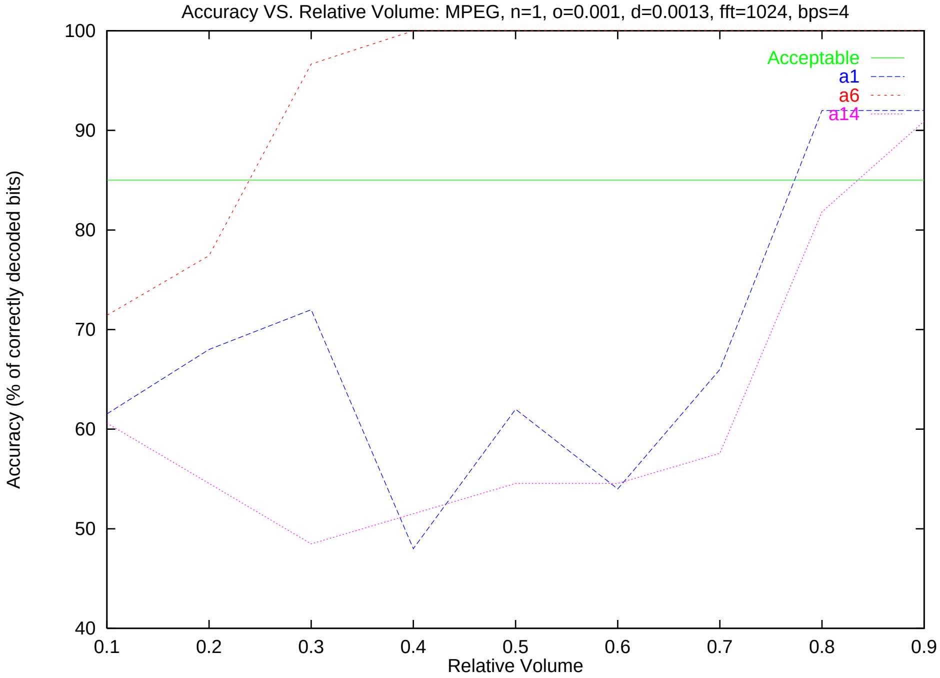

Fig. 16. Accuracy vs. relative volume: analog wire and MPEG

recovery rates. The average listener cannot resolve the echoes with an offset of 0.001 seconds. Below a 0.5 millisecond offset, even the decoder had difficulty distinguishing the echo from the cover audio.

Extensive testing reveals that the two most important echo parameters are relative volume (decay rate) and offset. The relative volume controls the recovery rate. While the offset is the major factor in the perceptibility of the modifications.

The results illustrated in Figures 14 through 17 were obtained at sampling rates of 44.1 kHz (closed-loop) and 10 kHz (wire, MPEG, and SEDAT). Other sampling rates tested include 8 kHz, 16 kHz, and 22.05 kHz all yielding similar (but appropriately scaled) results.

As can be seen, echo hiding performs very well in situations where there is no additional degradation (such as that produced by D/A, line noise or lossy encoding). In this respect, its performance is similar to many existing techniques. It's strength lies in its reasonable performance even in the much more challenging cases where such degradation is present.

At the present time, echo hiding works best on sound files without gaps of silence. This is unsurprising as it is difficult to analyze and recover echoes in regions of silence (such as inter-word pauses in speech). We are working on various thresholding techniques to try to avoid these difficulties by encoding only those areas where there is sound, and skipping areas of

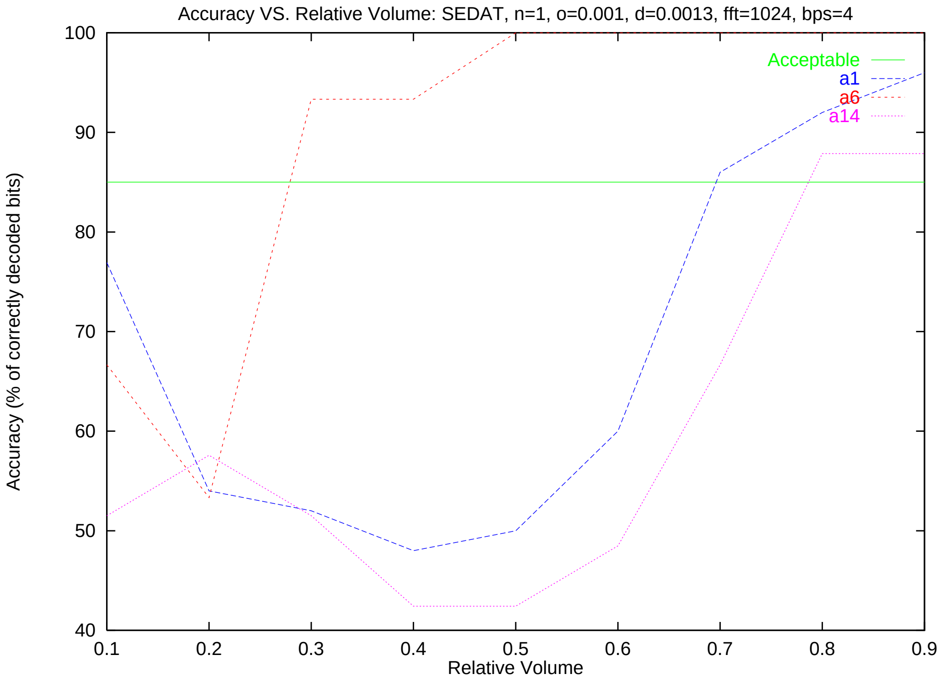

Fig. 17. Accuracy vs. Relative volume: analog wire and SEDAT

silence completely.

8 Future Work

Echo hiding can effectively place imperceivable information into an audio data stream. Nevertheless, there is still room for improvement. We have been examining the use of different echoing kernels and their effect on recovery accuracy and echo perceivability. In particular, we are actively researching both multi-echo kernels (adding another level of redundancy) and pre-echo kernels (echoing in negative time). With the old kernels, we are modifying the encoding process to be self-adaptive. Completion of these modifications will allow the encoding program to decide which parameters yield the highest recovery rate given the user's constraints on perceptibility and sound degradation. In addition, we will use echo hiding as a method for placing caller identification type information in real time over 8-bit, 8 kHz, analog phone lines.

9 References

1. W. Bender, D. Gruhl, N. Morimoto, "Techniques for data hiding," Proc. of the SPIE, 2420:40, San Jose, CA., 1995.

2. R. C. Dixon, Spread Spectrum Systems, John Wiley & Sons, Inc., 1976.
3. L. R. Rabiner and R. W. Schaffer, Digital Processing of Speech Signal, Prentice-Hall, Inc., NJ, 1975.
4. A. V. Oppenheim and R. W. Schaffer, Discrete-Time Signal Processing, Prentice Hall, Inc., NJ, 1989.

Appendix

Much of the following short tutorial was derived from Oppenheim and Schaffer's Discrete-Time Signal Processing. Please refer to the original for a more complete discussion.

A Cepstrums

Cepstral analysis utilizes a form of a homomorphic system which converts the convolution operation to an addition operation. As with most homomorphic systems, the cepstrum can be decomposed into a canonical representation consisting of a cascade of three individual systems. These systems are the fourier transform ($\mathcal{F}$), the complex logarithm (see Section C), and the inverse fourier transform ($\mathcal{F}^{-1}$) as depicted in Figure 18.

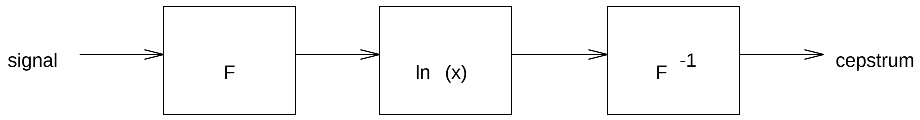

Fig. 18. Canonical representation of a cepstrum

The operational conversion is the result of a basic mathematical property: The log of a product is the sum of the individual logs and multiplication in the frequency domain is identical to convolution in the time domain. To exploit this fact, we use the first system in the canonical representation of the cepstrum to place us in the frequency domain by taking the fourier transform. In the frequency domain, the desired modifications are linear. The next system is a linear, time-invariant (LTI) system which takes the complex logarithm of the product of two functions. This simply becomes the sum of the logarithms. It is analogous to using a slide rule. In fact, the principle is the same. Multiplication becomes simple addition by first taking the logarithm. The final system puts us back in the original (time) domain. In order to express the "conversion" mathematically, let's convolve two finite signals $x_1[n]$ and $x_2[n]$.

$$y[n] = x_1[n] * x_2[n] \tag{2}$$

After taking the fourier transform of y[n], we get:

$$Y(e^{j\Omega}) = X_1(e^{j\Omega})X_2(e^{j\Omega}) \tag{3}$$

Now, we take the complex log of $Y(e^{j\Omega})$:

$$\log Y(e^{j\Omega}) = \log(X_1(e^{j\Omega})X_2(e^{j\Omega})) = \log X_1(e^{j\Omega}) + \log X_2(e^{j\Omega}) \tag{4}$$

Finally, we take the inverse fourier transform.

$$\mathcal{F}^{-1}(\log Y(e^{j\Omega})) = \mathcal{F}^{-1}(\log X_1(e^{j\Omega})) + \mathcal{F}^{-1}(\log X_2(e^{j\Omega})) \tag{5}$$

By the definition of the cepstrum, this becomes (where $\tilde{x}[n]$ is the cepstrum of x[n]):

$$\tilde{y}[n] = \tilde{x}_1[n] + \tilde{x}_2[n] \tag{6}$$

Figure 19 illustrates the entire conversion process.

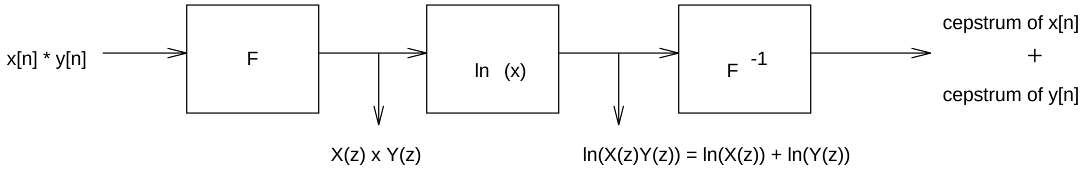

Fig. 19. Conversion of convolution in the time domain to the equivalent cepstral addition while still in the time domain

The inverse cepstrum is the reverse of the process described above and is depicted in Figure 20.

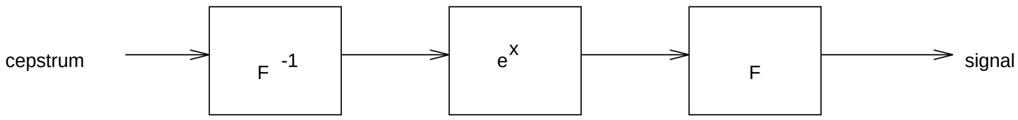

Fig. 20. Inverse cepstrum (canonical representation)

B Autocorrelation using cepstrums

Autocorrelation can be done while taking the cepstrum. Recall that the autocorrelation of any function x[n] is defined as:

$$R_{xx}[n] = \sum_{m=-\infty}^{+\infty} x[n+m]x[m] \tag{7}$$

With a change of variable (letting k=n+m and substituting m=k-n), the equation for the autocorrelation of a given function x[n] becomes:

$$R_{xx} = \sum x[k]x[k-n] \tag{8}$$

Now let's rearrange the second term in the summation (the x[k-n] term) so that:

$$R_{xx} = \sum x[k]x[-(n-k)] \tag{9}$$

Recall that convolution is defined as:

$$x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \tag{10}$$

There is a similarity between the convolution equation (Equation 10) and the "modified" autocorrelation equation (Equation 9). The only difference is the negation of time in the second term of the autocorrelation equation. Mathematically speaking, the autocorrelation equation can be represented as:

$$R_{xx} = x[n] * x[-n] \tag{11}$$

If a signal is self-symmetric, x[-n] is identical to x[n] by definition. Therefore, the autocorrelation of a self-symmetric signal becomes:

$$R_{xx} = x[n] * x[n] \tag{12}$$

In the frequency domain (i.e. after taking the fourier transform of the inputs), this becomes:

$$S_{xx}(e^{j\Omega}) = (X(e^{j\Omega}))^2 \tag{13}$$

Using cepstrums, the autocorrelation of a self-symmetric function can be found by first taking the cepstrum of the function and then squaring the result. The steps in this process are depicted in Figure 21 and Figure 22. Before we square the cepstrum, we first take the fourier transform. Then afterwards, we take the inverse fourier transform. The reason is the same as when we were finding the cepstrum (Appendix A). The fourier transform places us in the frequency domain where modifications are linear. A linear system (x^2) actually performs the operation. Finally, the inverse fourier places us back in the time domain. The inverse fourier transform

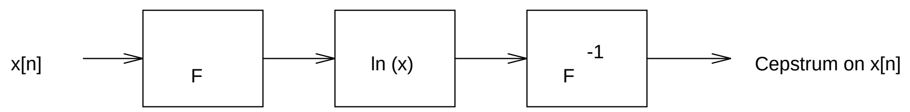

Fig. 21. The first step in finding the Cepstral Autocorrelation is to find the cepstrum of x[n]

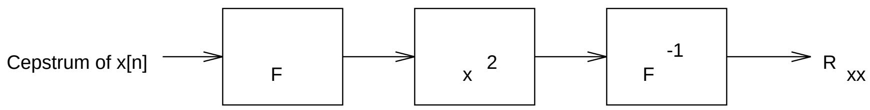

Fig. 22. Once we have the cepstrum, we square it

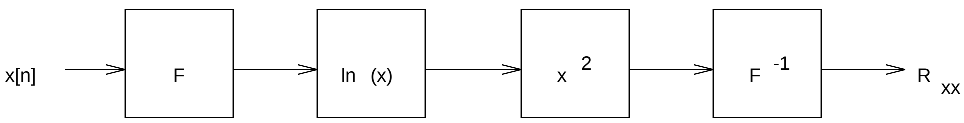

Fig. 23. Systems representation of Cepstral Autocorrelation

from step one (Figure 21) and the fourier transform from step two (Figure 22) will cancel each other when combined. In the end, we are left with the system shown in Figure 23.
Autocorrelation is an order n^2 operation. Using the system in Figure 23, the operation is reduced to a $n\log(n)$ operation. Thus for large n, finding the autocorrelation while taking the cepstrum is much more efficient.

C Complex Logarithm

The fourier transform is a complex function of ω. It can be decomposed into magnitude and phase/angle terms. Thus, if we have some finite signal x[n], the Fourier transform can be represented as a magnitude and an angle:

$$X(e^{j\Omega}) = |X(e^{j\Omega})|e^{jARGX(e^{j\Omega})} \tag{14}$$

ARG (angle modulus 2π) is used instead of arg (angle) since adding 2π (where n is any arbitrary integer) to an angle has no effect:

$$e^{j(x+2n\pi)} = e^{jx}e^{j2n\pi} = e^{jx}(\cos 2n\pi + j\sin 2n\pi) = e^{jx} \tag{15}$$

In most cases, the phase will be a non-zero value. Therefore, we can not use the natural logarithm when taking the cepstrum (Figure 18). Instead, we must use the complex logarithm which is defined as:

$$\log X(e^{j\Omega}) = \log(|X(e^{j\Omega})|e^{jARGX(e^{j\Omega})}) \tag{16}$$

Once again (as in Appendix A) we exploit the fact that the log of a product is identical to the sum of the individual logs:

$$\log X(e^{j\Omega}) = \log(|X(e^{j\Omega})|) + \log(e^{jARGX(e^{j\Omega})}) \tag{17}$$

Exploiting that log and e^x are inverses, we get:

$$\log X(e^{j\Omega}) = \log|X(e^{j\Omega})| + jARGX(e^{j\Omega}) \tag{18}$$

In order to further motivate the idea of converting from convolution to addition, let's mathematically re-examine Appendix A in light of the complex logarithm. We begin by first convolving two finite signals $x_1[n]$ and $x_2[n]$:

$$y[n] = x_1[n] * x_2[n] \tag{19}$$

Convolution becomes multiplication in the frequency domain:

$$Y(e^{j\Omega}) = X_1(e^{j\Omega})X_2(e^{j\Omega})) \tag{20}$$

Taking the complex log:

$$\log Y(e^{j\Omega}) = \log(X_1(e^{j\Omega})X_2(e^{j\Omega}) \tag{21}$$

Finding the mathematical equivalent:

$$\log Y(e^{j\Omega}) = \log(X_1(e^{j\Omega})) + \log(X_2(e^{j\Omega})) \tag{22}$$

Now, we can substitute the result from Equation 17 and rearrange to get:

$$\log Y(e^{j\Omega}) = (\log|X_1(e^{j\Omega})| + \log|X_2(e^{j\Omega})|) + (jARG(X_1(e^{j\Omega})) + jARG(X_2(e^{j\Omega}))) \tag{23}$$

The use of the complex logarithm in cepstral analysis allows the addition of signal components instead of the convolution of the signals.

This article was processed using the LaTeX macro package with LLNCS style