Techniques for Data Hiding in Audio Files

by

Norishige Morimoto

Submitted to the

Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements

for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1995

Author ....
Norishige Morimoto
May 12, 1995

Certified by ....
Walter Bender
Associate Director for Information Technology
Thesis Supervisor

Accepted by ....
Frederic R. Morgenthaler
Chair, Department Committee on Graduate Students
Electrical Engineering and Computer Science

Techniques for Data Hiding in Audio Files

by

Norishige Morimoto

Submitted to the Department of Electrical Engineering and Computer Science on May 12, 1995, in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering and Computer Science

Abstract

Data hiding is a group of techniques for embedding data in media such as images and audio signals. Ideally, the data will be directly embedded into the data stream of the host medium imperceivably, and will also be immune to the modification of the host medium, allowing the data to stay embedded. The process is constrained by the bandwidth of data to be embedded, the need for invariance of the data under conditions where the "host" signal is subject to distortions, and the degree to which the data must be immune to interception, modification, or removal by a third party. Both novel and traditional techniques for data hiding are described and evaluated. This work has focused primarily on audio.

This work was supported in part by the News in the Future Research Consortium and IBM.

Thesis Supervisor: Walter Bender

Title: Associate Director for Information Technology, MIT Media Laboratory

Acknowledgment

I would like to thank the members of the MIT Media Laboratory, especially the members from the News in the Future research consortium for all their supports.

Special thanks to my advisor Walter Bender, for the valuable technical advice and encouragement during my research period, and for providing me the opportunity to conduct my research at the Media Laboratory. It will be one of the most valuable experiences for my future career. Also, great thanks to Daniel Gruhl who helped me with his extensive knowledge of computers both hardware and software. The project could not be accomplished without his support.

I would also like to take this opportunity to thank IBM-Japan for their generous arrangement that made it possible to continue my advanced education at MIT. Especially, thanks to Kenichi Yokoyama, former Manager of STO, IBM Yamato Laboratory and to Masaki Kobayashi, Manager of the Notebook I/O Product Development, IBM Yamato Laboratory, who strongly encouraged me to apply for this overseas scholarship program.

Finally, I would like to thank my wife Atsuko and my son Yuki, who have supported me physically and mentally all the time.

Table of Contents

Chapter 1

Introduction

Digitization of media has greatly improved the portability, efficiency and accuracy of information. Digitized information can easily be reproduced, manipulated, stored and transferred from one place to another without sacrificing the quality of the information. However, this convenience creates several troublesome side effects. In an open network environment, the ease of reproduction and modification makes the violation of copyright and the tampering of media easier. Although all digitized information is protected by copyright law, reproduction and manipulation of digitized media is casually done, and the illegal reuse of the information is almost impossible to track.

A technique that can provide reliable proof of the ownership of media is desirable.

Data hiding[1] is a group of techniques being developed to solve this problem. It is motivated by the need to provide protection of copyright. Because of its unique capabilities, it also has potential for use in a number of other annotation applications.

1.1 What is Data Hiding?

Data hiding is a group of processes that allow users to embed coded information in a "host signal" such as an audio sample or a still image. The primary purposes of data hiding in digital media are to provide proof of the ownership and to assure the integrity of the content. Therefore, the hidden data should stay embedded in the host signal, even if the host signal is subjected to modification, such as filtering, re-sampling, cropping, and/or data compression. Some secondary applications, such as photography and captioning have less stringent requirements. Hidden data need not be inviolate to detection or removal as these data are included for the benefit of both the information provider and the consumer of the information. The amount of data to be embedded, the transparency of the data, and the data's immunity to modification are the main parameters that defining the properties of the data-hiding technique. These parameters vary among different applications. Therefore, instead of using a single technique, a group of processes containing several different techniques are needed to span the entire range of the requirements of the typical application.

The technical challenges of data hiding are formidable. A conflict between data hiding and data compression techniques is the source of the primary challenge. The key to data compression techniques is the elimination of statistically redundant or perceptually irrelevant information from a signal whereas the key of data hiding is to the addition of information to the signal. As you might imagine, whatever "hole" in the host signal one finds to fill with data is likely to be eliminated by data compression techniques. However, if a "hole" is small enough, only providing a few bits of data, it may not be worthwhile for a compression algorithm to exploit. However, a few bits will be more than enough for the purpose of the data hiding. The key to successful data hiding is to find those "holes" that are not large enough for data compression schemes to exploit but that are sufficient for the purpose of data hiding.

Another challenge is to fill these "holes" with data in a fashion that remains inviolate to host signal modifications. This is an important feature because the purpose of data hiding is to protect the copyright of the host signal by affixing the data to the host signal without restricting subsequent manipulation of the host signal.

1.2 Required Capabilities

The requirements of a successful data-hiding system are defined as follows:

(1) Perceivable degradation of the host signal should be minimized -- the embedded data needs to be "invisible" or "inaudible."

(2) The information has to be directly embedded into the data stream of the host signal, rather than into a header or attached file, so that the embedded data will remain encoded across different file formats;

(3) The embedded data should be immune to common modifications ranging from the incidental to intentional, i.e., transmission line noise, compression, filtering, re-sampling, cropping, etc.

(4) Asymmetrical coding[i] is desirable since the decode key may be open to the public.

(5) A data restoration scheme, such as Error Correction Coding[1], should be used to ensure the integrity of the embedded data after the modification of the host signal.

(6) The embedded data should be self-clocking or arbitrarily re-entrant. It is difficult to meet all of the above requirements with a single algorithm. There is always a trade off between the bandwidth of the data to be embedded, the immunity of the embedded information and its imperceibability to the end user.

1.3 Prospective Applications

As mentioned in the previous section, data-hiding techniques have the potential for a wide variety of interesting applications. In this section, several prospective applications, including watermarking, tamper proofing, feature locationing and captioning are discussed. In each particular application, the amount of data to be hidden and the expected level of immunity to modification are different, therefore, different techniques are needed. Since the quality of the encoded file should be lower than the limit of human perception, there are always trade-off between the amount of data hidden and the level of immunity to modification.

i. The coding scheme should have different keys for encoder and decoder, so that the coding scheme will remain unpredictable, even if the decoding key is been widely distributed.

Digital Watermark

The purpose of the digital watermark is to embed an indelible mark on a host signal that can provide solid proof of ownership, for instance, an author's signature or a company logo. In this case, the amount of the data to be embedded is very small, usually as low as one bit[i]. However, since the information is critical and the embedded watermark may face intelligent and intentional attempts at removal, the data-hiding techniques used for this application must be robust to a large variety of modifications.

Digital watermark can be used for the protection of photographs or any other digital image publications. A digital camera equipped with a digital watermarking function could be used to embed the signature of the photographer into pictures as they are taken.

Tamper Proofing

Another low data-rate application is tamper proofing. By embedding data relating to the host signal structure, geometrical modifications, such as affine transforms[ii], can be detected and modelled, possibly allowing the recovery of the original unmodified image. For instance, if file is obtained after the image is re-scaled by an unknown factor, the scaling factor can be detected and the original image can be recovered. One of the techniques used to implement tamper proofing is encoding of a definitive "ruler" into the file. By detecting the change of size and the orientation of the embedded "ruler," changes applied to the host signal can be detected. One of the keys to successful tamper proofing is the ability to find a "ruler" that is resistant to non-geometrical modifications such as filtering, so that the embedded information can be robust to both classes of modification.

i. the information is or is not mine

ii. affine transformation: geometrical transformation such as shift, rotation, flip and re-scaling.

Feature Locating

In this application, the embedded data is hidden to mark specific locations of a host signal. Individual content features, e.g., the name of a person in a picture or a "bookmark" to a specific segment of speech, can be retrieved with this mark. Although the data is not likely to be removed intentionally, the data-hiding scheme for this application needs to be immune to certain common modifications that may be applied during the normal usage of the signal, i.e. re-scaling, compression or transmission noise.

Feature locating enables an editor to encode descriptive information, such as the location and identification of features of interest, directly into a signal.

Feature locating is unlike the use of the annotation or attached files in that the information is spatially located in the file. It is not removed unless the feature of interest is removed from the host file.

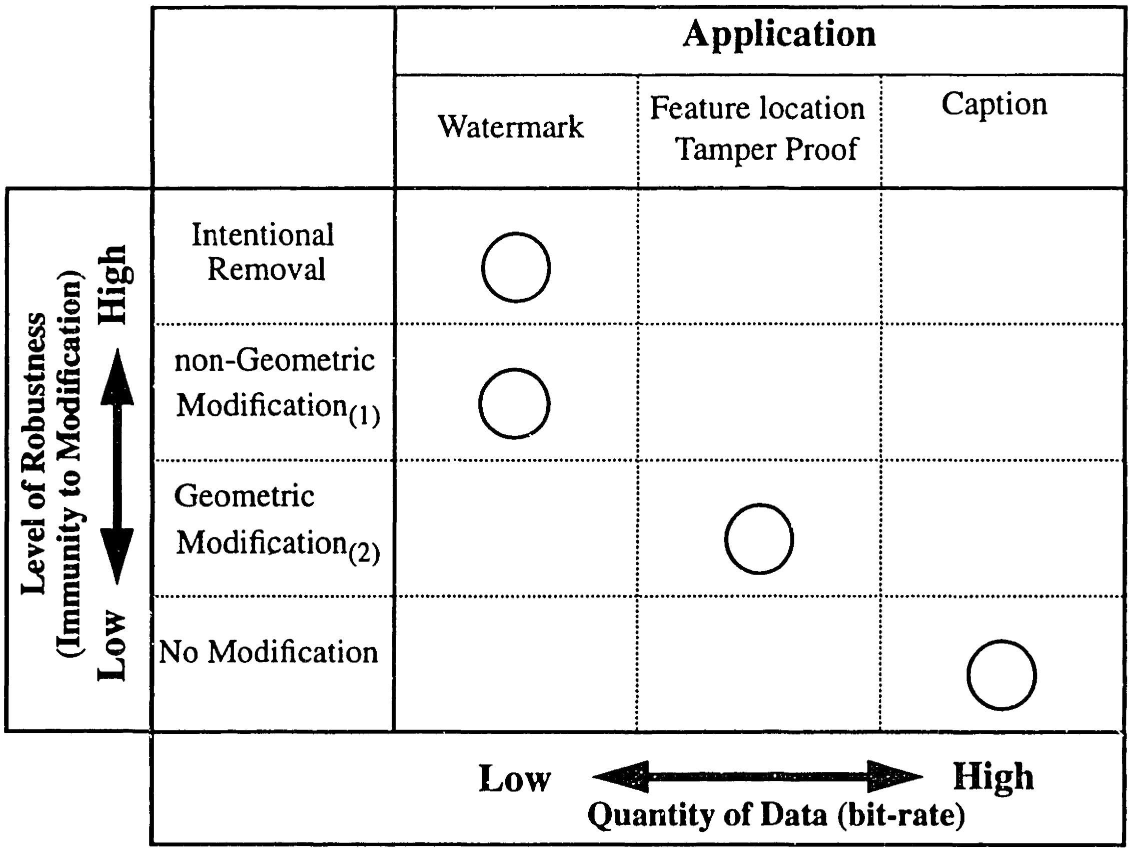
Image and Audio Captioning

One application that requires a large amount of data is captioning. Presently, captions (or annotations) often travel separately from the host signal, requiring additional channels, storage and correlation. Annotations stored in the file header or resource sections may be lost as the file is transmitted or stored in different type of formats. (For example, when image file is transmitted through the internet, the file may be subjected to more than one conversion of the format). If annotations created in a TIFF[i] format image, it will be lost when the image is transformed to PPM[ii] format. Annotations embedded directly in the data stream of the host signal do not suffer these problems.

i. Tagged Image File Format
ii. Portable Pixel Map

1.4 Problem Space Definition

The required capabilities of the data-hiding techniques for each of the above applications vary widely. The relationship between the applications and the required capabilities are illustrated in Figure 1.1. The problem of data hiding can be illustrated in terms of a trade off between two key parameters: the amount of data to be embedded and the level of the immunity to the modifications.



Level of Robustness (Immunity to Modification) Low ↔ High	Application: Watermark	Application: Feature location Tamper Proof	Application: Caption
Intentional Removal	○		
non-Geometric Modification(1)	○		
Geometric Modification(2)		○	
No Modification			○
	Low ↔ High Quantity of Data (bit-rate)		

(1) filtering(kernel application), compression, etc.
(2) affine transformation, cropping, etc.

Figure 1.1: Data Hiding Problem Space (the allowable degradation of the host signal is limited to a certain constant)

The horizontal-axis represents the amount of data to be embedded, and the vertical-axis represents the level of modification to the signal anticipated, (e.g. the level of the immunity of embedded data to modification). The ultimate goal of this project is to fill the problem spaces with appropriate data-hiding techniques. In practice, the requirements of robustness are lower in the applications that require more data space. The realistic problem spaces currently being investigated are indicated by "O" in Figure 1.1. The data-hiding techniques for each indicated problem space are described in the following chapters.

1.5 Previous Attempts at Data Hiding

While there is no existing technique that can meet all of the requirements listed in section 1.2, there are several techniques that allow a substantial amount of data to be hidden in digital media. When hiding data in images, one approach is to use the human eye's varying sensitivity to contrast versus spatial frequency of the image. The human visual system (HVS) has relatively low sensitivity to contrast at the very low and very high spatial frequencies. Adelson[2] uses this "hole" to hide a large amount of data in a still image. Bender[3] extends this idea by using "chaos" as a means to encode the embedded data. However, both of these techniques are not invariant to even a simple modification such as scaling or filtering. In the case of audio, there are several traditional techniques for encryption. Spread Spectrum[4] is a powerful group of techniques to encrypt jam-resistant data in a sound sample, however, it may introduce perceivable distortion to the host sound signals.

Chapter 1 provides an overview of data-hiding techniques, required capabilities, and prospective applications. In Chapter 2, data hiding in still images is discussed. This chapter also provides a broad view of the whole of data-hiding. Chapters 3 through 5 are about data hiding in audio files. The data-hiding system overview is shown in Chapter 3. The detail of the core techniques used for data hiding are discussed in Chapter 4, and several key supplementary techniques used to improve the performance of the core techniques are discussed in Chapter 5.

Chapter 2

Data Hiding in Still Image

Still images and audio samples are two of the major targets for development of data-hiding techniques. In today's multi-media environment, sounds and images are getting more interrelated, and in many cases, they are processed and experienced as a single unit. Because of the importance and the close relationship of the sounds and the images, it is necessary to introduce some approaches and techniques of data hiding for still images for better presentation, and to provide more complete explanation of the whole data-hiding processes. In this chapter, several approaches to data hiding in still images are discussed, and some promising techniques that have been developed as a part of this project are described.

Still image presents a variety of challenges due to the relatively small data space and the wide variety of the modifications to which the images are subject. A typical digital still picture may have a size of 200 X 200 (pixels) with 8-bit gray levels[i] which provides only approximately 40K-byte data space. This is equivalent to the data space provided by a five-second telephone-quality audio file or a one-second CD-quality audio file. Moreover, still images are more likely to be subject to drastic manipulations, ranging from simple affine transformations[ii] to cropping, blurring, filtering, and compressions. These distortions occur when the images are taken and placed in other media. Therefore, the capability of data-hiding techniques to withstand the effects of the modifications and transformations is more critical in the case of still images than in that of any other medium.

i. codeword that contains 8-bit binary code can represent 256 steps
ii. Geometrical transformations such as shift, flip, rotation and etc.

Despite these difficulties, there are some advantages to work with still images. First of all, the human visual system(HVS) has relatively low sensitivity to the contrast at the very high and very low spatial frequency patterns. This can be used as one of the "holes" for the data hiding. The HVS also has low sensitivity to small changes in gray scale (approximately one part in 40 for continuous random patterns). Since a typical digital image file has 8-bit gray levels (256 levels), this provides a good data-hiding space. Another advantage of working with still images is that the structure of the data of the still images are non-causal. Therefore, each pixel (data point) or group of pixels can be accessed in any order for processing. This accessibility provides a great deal of freedom of data processing as compared to other media.

2.1 Low Bit-Rate Coding

Techniques for this group of applications are required to be highly robust to a wide variety of modifications, transformations and compressions. Two different approaches to the data hiding in still images are shown below: a statistical approach, and an exploration of the varying sensitivity of the HVS. Both techniques described here have high level of immunity to modifications but can only embed a minimum amount of data.

Statistical Approach (Patchwork Coding)

The statistical approach, named "Patchwork," is using a pseudo-random process that modifies the gray levels of the selected pixels (data points). In natural pictures, the expected value of the luminance difference between two arbitrarily selected pixels is zero. The cumulative sum of the luminance difference of a number of pairs is also approaching to zero (if a sufficiently large number of pairs are selected and all of the differences of the gray level are summed). The cumulated sum S and its expected value E(S) is shown as follows:

$$S = \sum_{i=1}^{n} \{ai(ui,vi) - bi(li,mi)\} \tag{2.1}$$

$$E(S) = 0 \tag{2.2}$$

a_i and b_i are i-th selected pair of pixels and u_i, v_i, l_i and m_i are the corresponding locations of the pixels from the picture, and n is the total number of pairs selected for the summation. Since all of the images have a limited number of pixels, there is a limitation on the number of points that can be used for the summation; however, in most cases, at least 10,000 pairs of pixels are available and this provides a sufficiently large data space for this kind of application. Now, if the gray levels of the selected pair of pixels are modified, the cumulated sum (S) can be changed. This is shown as follows:

$$\hat{a}i(ui, vi) = ai(li, mi) + \alpha \tag{2.3}$$

$$\hat{b}i(ui, vi) = bi(li, mi) - \alpha \tag{2.4}$$

$$\hat{S} = \sum_{i=1}^{n} \{\hat{a}i(ui,vi) - \hat{b}i(li,mi)\} = \sum_{i=1}^{n} [\{ai(ui,vi) + \alpha\} - \{bi(li,mi) - \alpha\}] = S + 2\alpha \times n \tag{2.5}$$

$$E(\hat{S}) = 2\alpha \times n \tag{2.6}$$

As shown in equation (2.6), the expected value of the cumulated sum after the gray level modification is now converging to (2α x n) instead of zero. This is illustrated in Figure 2.1. The relative variance (σ^2) of the curve of the expected value can be reduced by increasing the number of the pairs, and the distance between two statistic curves (2a x n) can be increased by increasing the value of the modification level (a) so that the two states can be more distinguishable from each other. In other words, the certainty of the code can be adjusted by the number of pairs and the amount of the modification level. In Figure 2.1, if the S in a picture obtained by using the key is near $(2\alpha \times n)$, it means there is very high possibility that the picture is been coded; or, the result of the computation has very low possibility to show the value near $(2\alpha \times n)$ unless it has been coded. (if the $(2\alpha \times n = 3\sigma)$, the probability of that the picture has not been coded will be 1 - 0.997 = 0.003)

By using this technique, one bit of information can be coded in an image and nearly 100% of certainty can be achieved with the parameters of a = 2, n = 10,000. The detection is nearly impossible without the key of the encoding path, and the technique presents great invariance to kernel applications, compressions and geometrical transformations.[i] The major limitation of this technique is its low data capacity. It can code only one bit of data which is as good as to represent "Yes" or "No".

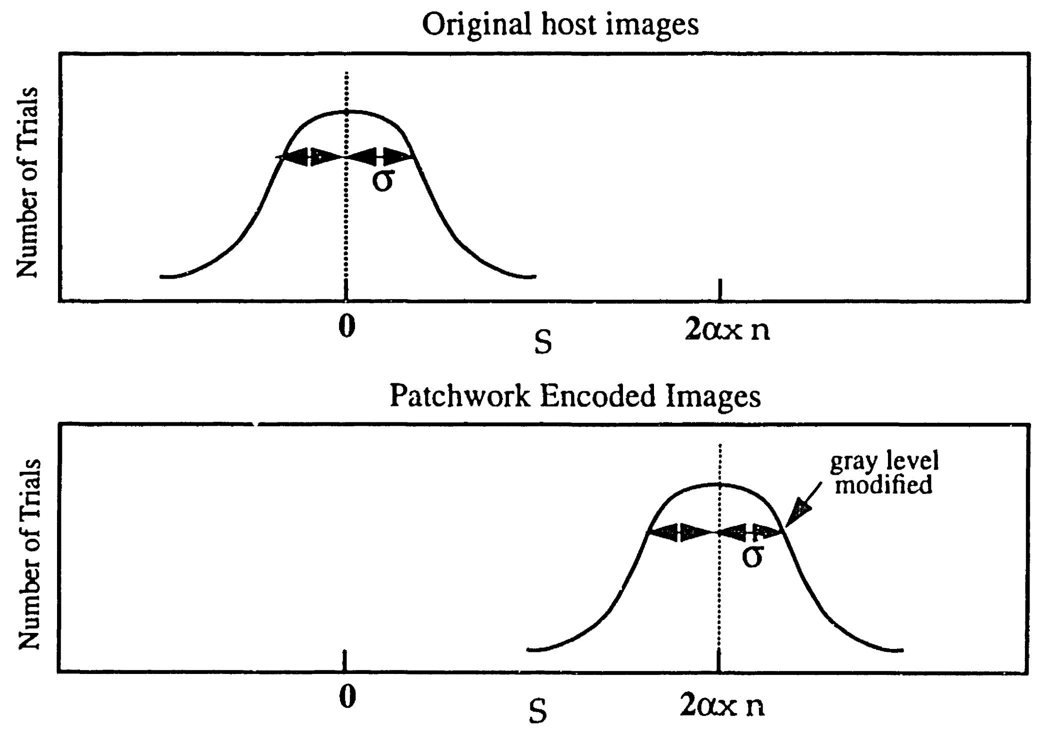


Figure 2.1: Statistical Analysis of original host images and the Patchwork encoded images. The number of trials that yield sum S is shown along the vertical axis.

i. Affine Coding will be used to make the code be resistant to geometrical transformations.

Visual Approach (Texture Block Coding)

Another approach for low bit-rate coding is "Texture Block Coding." This technique uses the characteristics of the low sensitivity of the HVS to the change of random texture patterns. The texture block coding is implemented by copying a block region from a random texture area in a picture and carefully pasting the block to the area that has a similar texture pattern, so that the pasted block can be invisible to the human eye. This operation results in two identical block regions in one picture which is very unusual in natural pictures. Therefore, if the identical region can be detected and identified, it can serve as a proof of the ownership of the picture. The embedded blocks are highly robust to the general modification to the images. Because, if the two blocks are identical, they are modified in the same way when the modification is applied uniformly to the picture. This means, the block can remain identical when it is subject to most of the modifications except for the region selective modifications. One example is shown in Figure 2.2. In this picture, five individual blocks are selected from random texture regions in the picture and each block is carefully pasted to the nearby area that has a similar texture pattern so that the blocks can be un noticeable. The embedded blocks can be detected one at a time by applying two-dimensional auto-correlation followed by the pixel by pixel subtraction at the detected peak index. One of the decoded results is shown in Figure 2 3 (a solid black block appears at the right bottom). The information carried by this technique is the shape of the block. The shape is not limited to square as shown in the example. It can be an arbitrary shape, such as a company's logo or an alphabet letter. However, the robustness of the code can be increased by using larger block. Block size of 16x16 pixels is concerned to be a sufficient size to withstand a combination of a low-pass filtering and J-PEG compression. The major part of the future work of this technique is the automation of the encoding process.

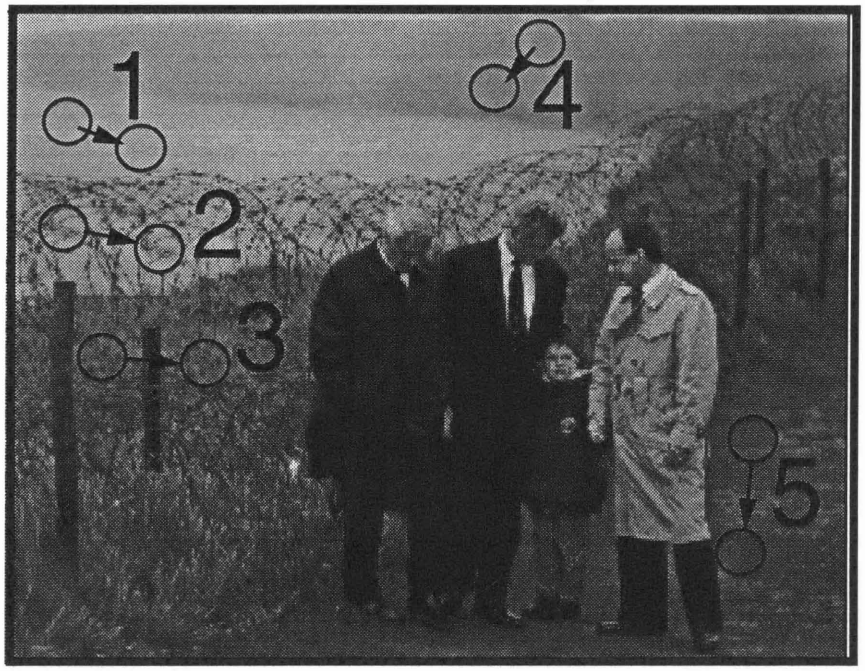


Figure 2.2: Picture encoded by five texture blocks (AP Photo)

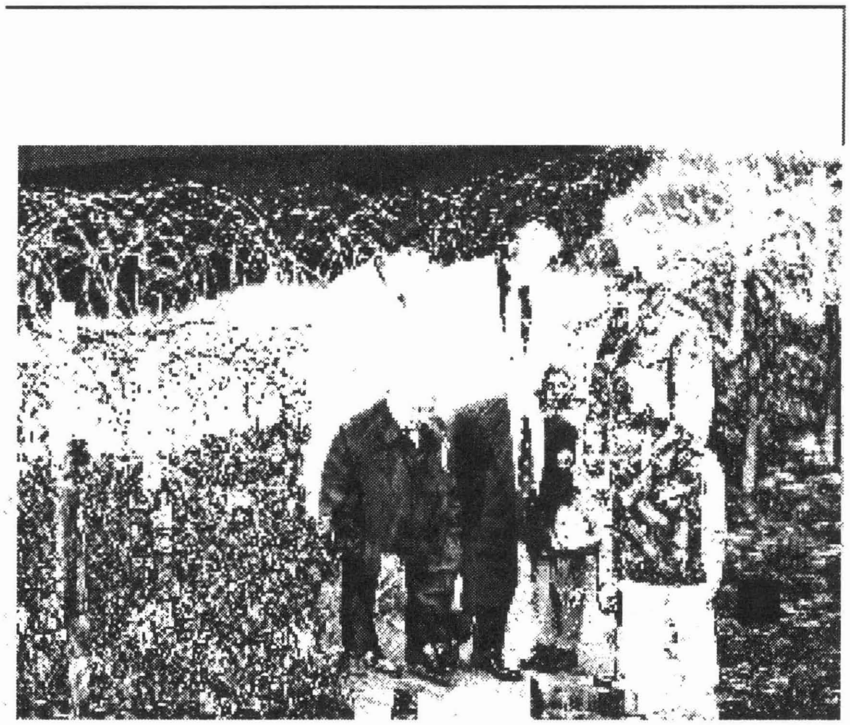


Figure 2.3: Decoded result of one texture block #5 from Fig. 2.2. The encoded picture had subject to low-pass filtering and 75% lossy J-PEG compression before the decoding.

2.2 High Bit-Rate Coding

As discussed in Chapter 1, the major use of the high bit-rate coding techniques is to embed captions; and the capability of coding a large amount of data is of greater concern than an immunity to modifications. One of the common techniques is Least Significant Bit (LSB) Coding.

Generally, the luminance level of each pixel in a digitized image is quantized in 256 levels which is represented by an 8-bit binary code. Thus, the LSB represents 1/256 luminance level of the maximum luminance: one step of the quantized luminance level. The essential idea of this technique is to use the LSB of each data point as a coding space. Since the HVS is less sensitive to the small change of the luminance in general pictures, theoretically, it allows us to have a binary coding space as large as the total data point of the host picture. A coded binary string can be embedded by simply replacing the LSB of the data of the host image by an encoded binary string. [5]

The decoding is as simple as the encoding. Simply extract the LSB of each pixel and map to the original data form as it is coded. If the data is a one dimensional string, map it to a one dimensional string, if the data is a binary image overlapping to the host image, map it to an image space equivalent to the host image.

As mentioned above, since the data points of the coding space also corresponds to the data point of the picture, we can take advantage of this to embed a secondary binary image on top of the host image. The secondary image can then, viewed as a transparency overlaid on the picture, allow us to write comments or pointers or arrows on the picture. This application is called feature locationing (Section 1.3). Another useful application is to embed a "ruler" with a known geometrical dimension that can indicate the type and the amount of the modification that have been applied to the image.

The LSB coding technique gives a desirable performance when the whole process is performed in a closed digital environment because a change in even the smallest level of the luminance will destroy the embedded data. If any distortion or heavy channel noise becomes involved in the process, the embedded data can easily be lost or destroyed. The robustness can be improved by employing the redundancy coding such as an averaging method or Error Correction codling (see Chapter 5). If the information is repetitively coded, it can survive more additive random noise in return for a significant loss of the data space. As mentioned previously, techniques used for high bit rate coding in these cases are only appropriate where it is reasonable to assume the host signal to be in a closed digital environment, so that the absolute value of each data point of the signal is well preserved without any external noise/modification even to its least significant bits.

2.3 Summary of Data-Hiding Techniques for Image

Since the modification of the image file has a wide range, it is difficult to make a data-hiding technique that is resistant to all possible transforms and that meets the required data bandwidth by itself. Therefore, these techniques must be used in combination, with one technique supplementing another, and each compensating for the strengths and weaknesses of the others. For instance, a combination of "ruler" embedding and the "patchwork" will compensate each other if the application requires invariance to geometrical transformations but only a small amount of data to be embedded.

Chapter 3

Data Hiding in Audio

The approach to data hiding in audio signals (or one dimensional signals) is similar in concepts and different in technical implementation from the data hiding in still images described in chapter 2. In this chapter, advantages and difficulties of data-hiding techniques in audio files are described. The prospective applications are described in section 3.1 and the detailed explanations about each developed technique are shown in section 3.2 to section 3.4.

One of the greatest advantages in working on audio is the existence of numbers of sophisticated audio signal processing techniques. As one of the most powerful communication vehicles, many novel and useful techniques have been developed for audio signals. Before today's multi-media environment, the media of communication mainly meant audio signals, therefore many techniques developed in the fields of communication theory and signal processing were based on the audio signal (or one dimensional signal). These techniques have been developed and used for many decades. The efficiency and the security of the information have become much more important in the commercial environment and the techniques such as encryption and cipher developed during the war time for military communication have become enormously useful in our multi-media era.

Although many of the signal processing techniques for audio were developed for analog signals, many of them can be applied to digital signals in practice. Some of the techniques even present better performance because of the characteristics of the digital signals. The discreteness of digital signals eliminates unnecessary complexity and vagueness of the signal and provides definitive synchronization in the time axis. From the data-hiding viewpoint, there are several promising common techniques existing and ready to serve as the basis of our development. The Spread Spectrum technique was developed mainly concerning the security of the data against interception. This is one possible technique that can be used for data hiding. However, in data hiding, stability and invariance of the data is more crucial than the interception.

From the application view point, an audio file is less likely to be heavily modified when it has been reused. In most cases, the heavily modified sound files are no longer as useful as the original one. Therefore, the required level of immunity to the modification may not as high as in the case of the still images.

Technical Challenges

The major challenge to data hiding in audio is the high sensitivity of our human auditory system (HAS). The HAS is extremely sensitive to an additive noises, especially if the subjected host signal is recorded on a perfectly silent background. In that case, the minimum level of the detectable additive noise is as low as -80dB. In other words, if the file is recorded in a 16-bit format, even a single bit of random noise can be detected by HAS. (In a 16-bit format audio file, the minimum quanta is 1/65536 of the dynamic range where the sensitivity of -80dB is equivalent to 1/1Million in magnitude.) However, a human never is as perfect as a machine. The sensitivity of the HAS to the additive noise drops quickly as the amount of the background noise increases. The sensitivity also drops when the dynamic range of the content increases. Thus, the level of maximum allowable additive noise varies greatly depending on the characteristics of the original host signal. In many cases, this allows us to embed data sufficient for our applications by using developed data-hiding techniques.

Another challenge is format transformation. As mentioned previously, sound data is more likely to be used in both analog and digital domains than the still images. Therefore, there is a better chance that the sound signal will be transferred between analog and digital domains (e.g., transmission and recording are mainly done in analog where editing and processing are mainly done in digital). An audio signal that passes through an analog stage and is subsequently re-digitized generally only carries its relative characteristic values. Signals that stay digitally encoded but undergo re-sampling may preserve amplitude and phase accurately but have changing temporal quantization.

The loss of the precision can be a big problem, as in many cases, the embedded data requires precise synchronization and definitive time scale for the extraction of the embedded data string. A specific environment for the system used in one of the development cases is shown in Chapter 4 and, currently, the system has limited capability to deal with the digital to analog or analog to digital conversion stages. More studies need to be done in order to make our techniques more useful in a general audio environment.

Modification to the Audio Signals

The frequent modifications and transformations applied to audio signals are also different from these applied to the still images. Because of the nature of the sound signal, modifications such as low-pass filtering, high-pass filtering, re-sampling and compression are more frequently used when the modification of the sound is done, where the geometrical transformations such as flipping and rotation are less likely to be applied. Since the sound signal is in one dimension and causal, these modifications will destroy the sound.

As the modifications are limited to fewer types, fewer techniques can be used to cover the wide range of applications instead of using varieties of different techniques as in the case of still images. The varying range of the applications requiring different robustness and different data capacities can be covered by adjusting the parameters or employing supplementary techniques such as Error Control Coding[6].

3.1 Prospective Applications

The prospective applications for audio files can be slightly different from the applications for the image files. Although the primary purpose of the data hiding is the protection of copyright, other useful applications can be developed too.

Bookmark and content ID embedding are probably the most important applications for audio files, after copyright protection. Because the sound signals are not visible and the indexing and filing are not intuitive as the image files, it is difficult to provide an appropriate index or file tags. Currently, this is done by using an external file that carries the list of the contents, but there is no common technique that can provide an index of the particular segment in the sound that the user is interested in when the length or the format of the file has changed. By extending the data-hiding techniques, these kinds of the applications can be easily achieved. In this section, several prospective techniques and the required capabilities of each application are discussed.

Captions

Currently, the captions of the digitized sound file are stored in a separate file. They exist independently and, therefore, are easily lost or incorrectly placed. By embedding the captions directly into the sound file, the use of the additional files can be reduced. A picture or other media in the form of a binary string can also be embedded into the sound signal.

The advantage of this technique is that no new hardware or software is required to record, transmit or receive the signal. One application is to embed a picture into a music CD. Since the music CD and the CD-ROM have similar physical format, we can play music CDs on our computers. With such a setting, we could possibly retrieve the pictures that are embedded into the sound signal, such as a picture of the artist.

Watermark

Watermark or copyright protection is one primary purpose of the development of the data-hiding techniques. As is the case of still images, the techniques used for this application need to be highly robust to such modifications as low-pass filtering, high-pass filtering, compressions and re-sampling, but not necessarily to flipping and rotationing, because these modifications do nothing else but destroy the sound. As mentioned previously, since expected modifications are limited, the same technique can be used to cover a wide range of the applications instead of using various different techniques. The techniques used for the watermark can be the same technique used for other applications with different parameter settings.

For copyright protection, an ID number of the registered sound file or coded copyright sentence will be a common data to be embedded into the sound signal.

Bookmarks

The idea of the bookmarks is similar to feature location for still images. In some cases, one would like to put an index tag or "bookmark" on a specific part of the sound signal for future convenience. For instance, the name of the speaker onto the segments that are spoken by the speaker, or the date and the place of the recording and so on. By using data-hiding techniques, one can embed a signal into a particular segment of the sound. The embedded bookmark can serve as an index to search for a particular section in a sound file. The advantage is, since the data is directly embedded into the data structure of the sound, it can stay with the signal even if the length or the format of the sound is changed.

Content ID Embedding

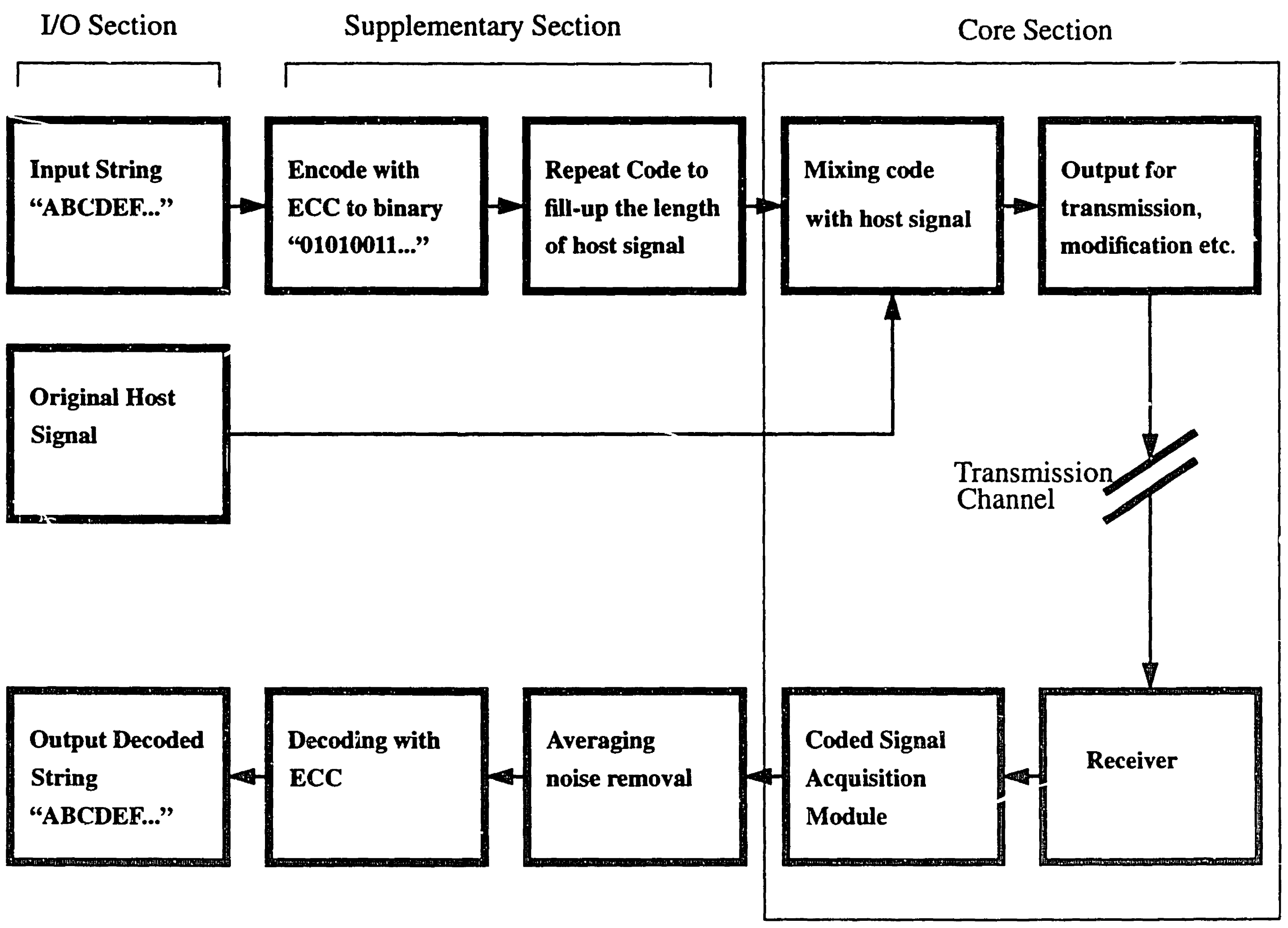
As an extension of the bookmarks application, we can embed a data that carries the information of the content into a sound signal. Unlike the separate descriptor files, the description of the sound segment can be embedded into a location that literally corresponds to the location of the segment in the file. One application is to identify what is being broadcast. The contents of sound segments can be a song of a particular artist, a category of the news, a commercial of a particular sponsor, etc. This allows radio listeners to track and jump to the stations broadcasting the particular content they are interested in (e.g., news tracker, weather tracker, traffic tracker). Or one can automatically track how often a particular song or commercial has been broadcasted.

3.2 System Overview

In this section, an overview of a typical system using data-hiding techniques is described. Figure 3.1 shows a block diagram of a model of a data-hiding system. The system can be divided into three sections: core section, supplementary section and I/O section.

Core Section

The system is mainly characterized by the core section which contains the core data-hiding technique used for the system. The transmission channel and the data acquisition stage are also included in this section. Three core techniques for data hiding are described in Chapter 4. Each technique has its unique property and the technique for a particular system should be selected based on the required capabilities of each application. The transmission channel in Figure 3.1 represents communication channel noise, user modification/ distortion, re-sampling, D/A and A/D transform and any other change applied to the code between the coding and decoding stages.


I/O Section
Supplementary Section
Core Section
Input String "ABCDEF..."
Encode with ECC to binary "01010011..."
Repeat Code to fill-up the length of host signal
Mixing code with host signal
Output for transmission, modification etc.
Original Host Signal
Transmission Channel
Output Decoded String "ABCDEF..."
Decoding with ECC
Averaging noise removal
Coded Signal Acquisition Module
Receiver

Data-Hiding System Block Diagram

Figure 3.1: Data hiding System Block Diagram

The signal acquisition is the most important part of the signal receiver. The main part of signal acquisition is code synchronization and time-scale alignment. The strategy of the data acquisition varies depending on what core technique is used, and it is uniquely designed for each particular system.

Supplementary Section

The purpose of the supplementary section is to support the core technique section and improve the correctness, efficiency and robustness of the embedded code in order to meet the system requirements. Two major techniques are used in this system. One is the Error Correction Coding [ECC (see section 5.2)] and the other is an averaging method. A bit error of the binary code can be corrected with ECC and additive noise to the embedded data can be reduced by the averaging method. Both techniques serve to reduce the error rate of the code or improve the signal to noise ratio (SNR) of the coded sound signal. (e.g., reduce the energy of the embedding information that creates noise) These techniques can be used with any of the core techniques and their effects are independent from the effect of the core techniques. Fine and flexible adjustments can be made by changing the parameters of these supplementary techniques to find an optimum trade-off point for each application.

I/O Section

In this system, an input information (generally the alphanumeric characters) are converted to an 8-bit binary code word character by character. A few bits of ECC code are added to each code word and the entire binary string is thus expanded to the length that matches the length of the host segment by interpolation or replication. Then the coded information is embedded into a host segment by using one of the core techniques described in Chapter 4.

At the receiver, the encoding process is undone in the reverse order. After the signal acquisition, duplicated signals are averaged to reduce the additive noise. ECC decoding is applied to correct the remaining error in the detected binary strings (see Chapter 5). The detected binary code will be converted to a set of alphanumeric characters by using the same code book that is used in the coding stage (see Table 1 in Appendix A).

3.3 Application of Content ID Embedding

One example of the application of Content ID Embedding is shown in this section. The purpose of this application is to automatically detect a commercial broadcast on a radio station and log the number of repetitions, on air time, etc. This will help the sponsor of the radio station to track the practice of the contract and build sponsors' confidence in the radio station.

The challenges of this project are: (1) D/A and A/D conversion are included; (2) an undisclosed data compression/decompression scheme for satellite transmission is involved.

A minimum amount of additive noise or distortion can be expected since the data is not subject to intentional modification other than compression/decompression.

A block diagram of this project is shown in Figure 3.2. The recorded sound sequence is encoded by using a data-hiding technique after it has been digitized. The encoded sound sequence is then recorded to an audio tape and passed through the compression algorithm for the satellite transmission. The sound is decompressed at the receiver in the local radio station. The decoding of the embedded ID may be done at the end receiver or by intercepting the signal directly after the de-compression stage or by receiving the broadcast signal through a radio receiver. This application includes many analog to digital or digital to analog stages.

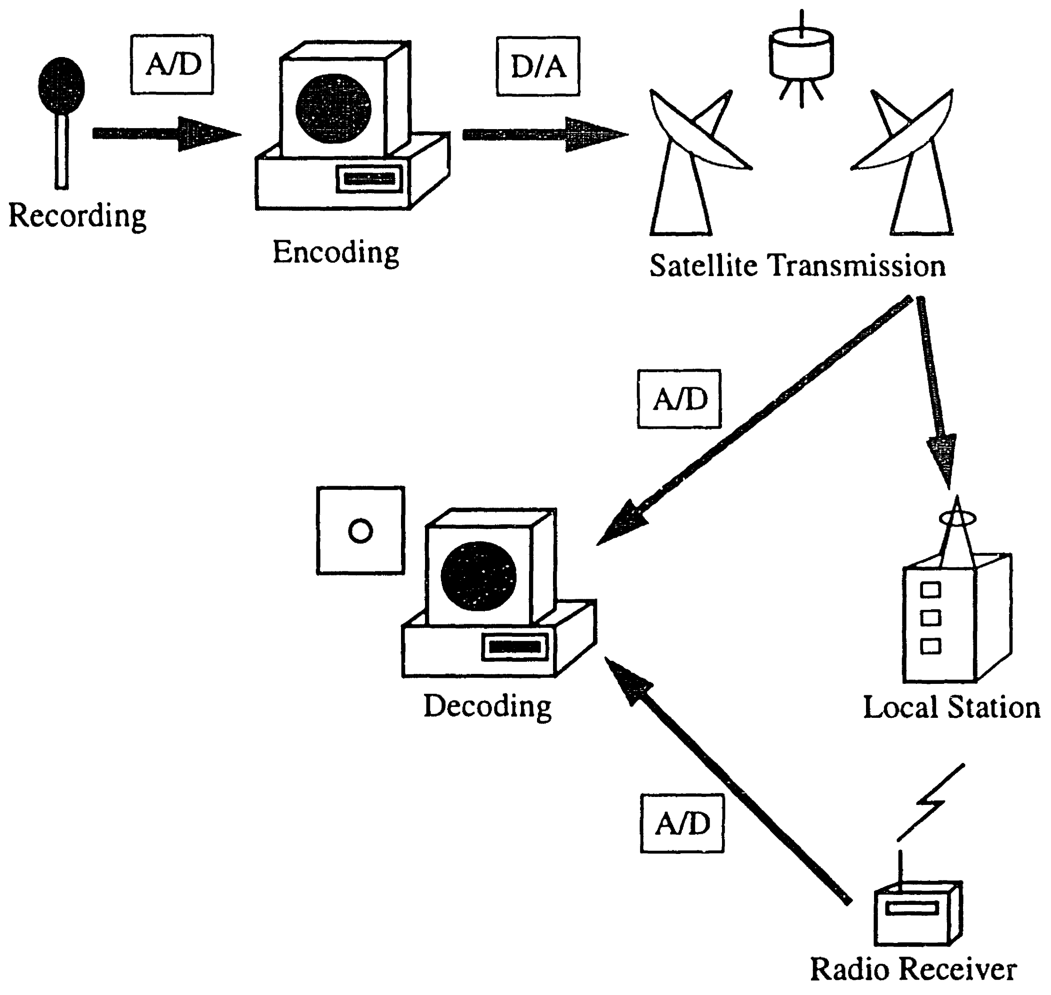


Figure 3.2: Block Diagram of the satellite simulation and test scheme

Chapter 4

Core Techniques

The most important part of the data-hiding system is the core section. The core section contains a core technique that generates and embeds the coded information into the host signal. Each technique has it's own unique property to cover the different aspects of the widespread applications, and thus the selection of the core technique should be made carefully concerning to the required capabilities of each particular application.

Since the coded data is directly embedded into the data structure of the host signal, there is always a trade off between the degradation of the sound and the amount of the modification (which corresponds to the data rate and the robustness of the embedded code). If the allowable degradation of the original signal is concerned to be a fixed value, the trade-off will be in between the robustness of the embedded data and the data rate.

In this Chapter, three core techniques are described, and each of them has its unique property. For instance, the Least Significant Bit (LSB) Coding has capability to code huge amount of data, but the Spread Spectrum (SS) Coding can only code a small amount of data. Phase Coding is invariant to a certain amount of the additive noise, but the LSB coding can be destroyed even with a minimum amount of the noise.

In the following sections, the detail of the coding schemes and the unique properties of each techniques are described. The advantages and disadvantages of each technique are discussed. The mathematical expression of the modification and the core data-hiding processes are attempted to seek for possible enhancement and improvement of the robustness.

4.1 Least Significant Bit Coding

Least Square Bit Coding is one of the simplest techniques to embed data into other data structures. The idea is similar to the Least Significant Bit Coding for images described in section 2.2. First, the least significant bits (LSB) in all of the data points are off-set to zero, and the data in a form of binary string can be embedded into host signal by simply replacing the LSB of each data point by a coded binary string. Since each data point of the host signal carries one bit of information, a sound sequence that is sampled by 8kHz has maximum 8,000bps[i] data rate and a sound sequence that is sampled by 44kHz has maximum 44,000bps data rate.

The LSB Coding is equivalent to generating noise with the magnitude equal to a minimum quanta of the host signal. In an n-bit format audio, the level of one quanta is $1/2^n$ therefore the level of the noise in a 8 bit format is 1/256 of the dynamic range 1/65536 for the 16 bit format audio. As mentioned in Chapter 3, even this small noise is will audible when the background noise of the host sound is relatively low. Since the impact of this noise is a direct function of the content of the host signal, the analysis of the background noise of the host signal and the type of the content are important considerations.

The major disadvantage of this method is the lack of immunity to any signal modification. When the definitive level of the sound signal is changed, the encoded information can be easily destroyed (e.g., channel noise, re-sampling, etc.). In order to provide a certain degree of robustness to the coding, redundancy techniques, which are described in Chapter 5 can be used. However, practically, this technique is useful only if no external noise is involved in the encoding, storing, transmitting and decoding processes. For this reason, the application of the LSB Coding should be limited to applications such as cap-

i. bit per second: a number of bits that can be sent through the communication channel in one second tioning, feature locationing where the process is in closed, digital system, and when it is reasonable to assume there will be no distortion or modification to the original file throughout the process.

4.2 Phase Coding

Phase coding is a coding schemes that introduces least perceptible noise to the host signal. The off-set of the phase of a sound is irrelevant to recognition by the HAS. The HAS has very low sensitivity to changes to the phase components of an audio signal. By using the phase components of the sound segment as a data space, a fairly large amount of data can be coded into the host signal. The embedded data is fairly transparent to the HAS if the relative relations between the phase components of preceding segments are well preserved. (The modification of the off-set of all the phase components results in no distortion to the sound signal.)

Unlike LSB Coding, Phase Coding is robust to small amounts of additive noise, since this noise won't affect to the distortion of the phase in most of the frequency slots. The wave form of the signal is more important than the absolute value of each data point in Phase Coding.

On the other hand, since the phase component is highly sensitive to any distortion of the wave form of the host sound signal, the embedded code may not be robust to manipulations such as a waveform enhancement, restoration and so on. Moreover, the use of the phase components for the data compression has been exploited by numerous audio compression algorithms. Therefore, the data embedded by using the phase coding may be removed by audio compression schemes such as ISO M-PEG[i], as well as any modification schemes that will corrupt the relationship of the phases of the sound wave form.

i. Standard compression scheme for motion pictures and audio defined by ISO.

Encoding Scheme

The basic idea of Phase Coding is to generate the phase matrix of a short segment of the sound and modify the phase matrix with an encoded information string. For the following segments, the phase difference between the consecutive segments extracted from the original sequences are preserved, thus the phase continuity of each frequency components is preserved. As long as the phase continuity is preserved, the wave form of each frequency component is also preserved (except a certain amount of independent phase shift is introduced to each frequency component). Phase Coding will change the relative relationship of the phase between different frequency components. This is called the phase dispersion[7]. This distortion is hard to notice, as long as the amount of the phase shift and the frequency of the change in a phase matrix remained below certain limits discussed in section 4.2.1. A model of the phase coding is illustrated in Figure 4.1.

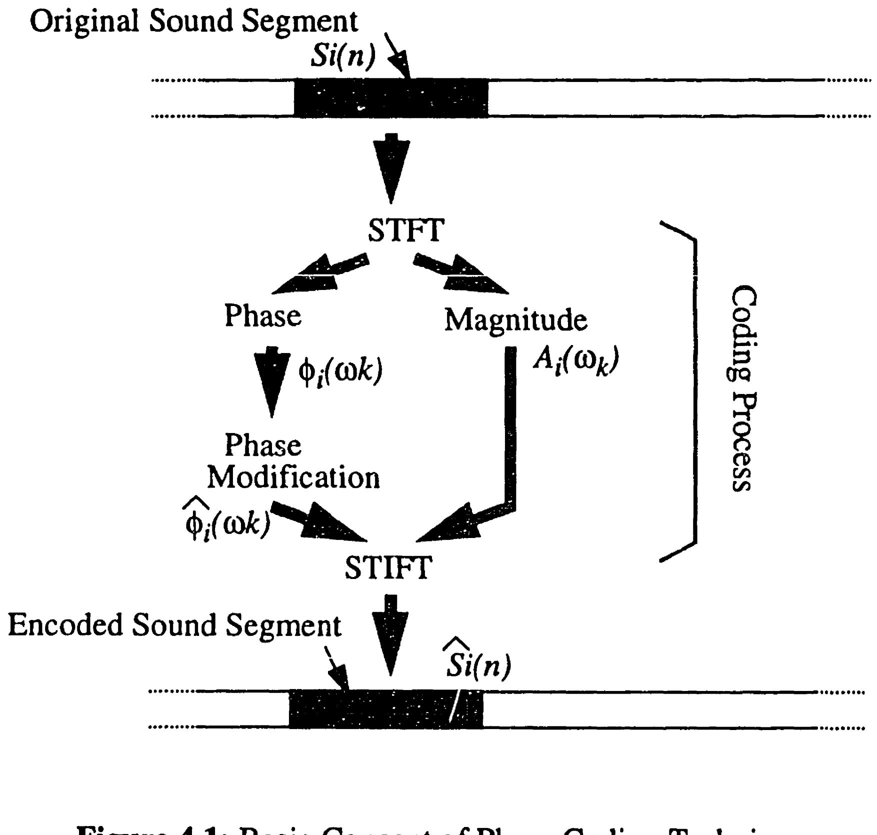


Figure 4.1: Basic Concept of Phase Coding Technique

The original sound sequence is divided into M short segments with N data points in each segment. Then, an N-point Short Time Fourier Transform (STFT)[8] is applied to the first segment [$s_0(n)$] to compute the phase [$\phi_i(\omega_k)$] and magnitude components. The N-point phase matrix, which is the coding space, is modulated by an encoded string. After modification, the sound segment is reconstructed by applying the Short-Time Inverse Fourier Transform (STIFT) with the modified phase components and the original magnitude components. In following segments, the phase differences between the original segments are added to the previous modified phase matrices so that the relative relationship of the phase between the consecutive segments are preserved.

The step by step procedure for the phase coding is shown below:

(1) Break the sound sequence $s[n]$ into M short segments which has N data points in each; $s_0[n], s_1[n] \ldots s_i[n] \ldots s_{M-1}[n]$;

(2) Compute N-points Short Time Fourier Transform (STFT) of each segment, and generate the matrices of the phase, $\{\phi_i(\omega_k)\}$,and magnitude, $\{A_i(\omega_k)\}$ for $(1 \le k \le N)$, where ω_k denotes center frequency for each frequency slot (bin);

(3) Store the difference of the phase of the adjacent segments in each frequency element $(0 \le i \le M-1)$:

$$\Delta\phi_i(\omega_k) = \phi_{i+1}(\omega_k) - \phi_i(\omega_k) \qquad (4.1)$$

(4) Code an information in a form of binary string by using the codebook (see Table 1 in Appendix A) and extend the length of this binary string to N points to generate a phase modifier matrix $d(\omega_k)$;

(5) Apply the modifier matrix d_k to the first set of the phase matrix:

$$\hat{\phi}_0(\omega_k) = \phi_0(\omega_k) + d(\omega_k)_n \tag{4.2}$$

(6) Modify the phase matrices of the follow on segments by adding the phase difference $[\Delta\phi_i(\omega_k)]$ to the previous modified phase components:

$$\begin{aligned}(\hat{\phi}_1(\omega_k) &= \hat{\phi}_0(\omega_k) + \Delta\phi_1(\omega_k))\\ &\dots\\ (\hat{\phi}_i(\omega_k) &= \hat{\phi}_{i-1}(\omega_k) + \Delta\phi_i(\omega_k))\end{aligned} \tag{4.3}$$

(7) Use the modified phase matrices and the original Fourier transform magnitude $A_i(\omega_k)$ to reconstruct the sound segments by computing the inverse STFT.

(8) Concatenate all modified segments to reconstruct the sound sequence.

In (5), if the modified phases exceeds π or below $-\pi$, it will be clipped to $+\pi$ or $-\pi$ because the STFT wrap the phase and the sign of the phase can not be preserved.

The block diagram of the encoding process described above is shown in Figure 4.2.

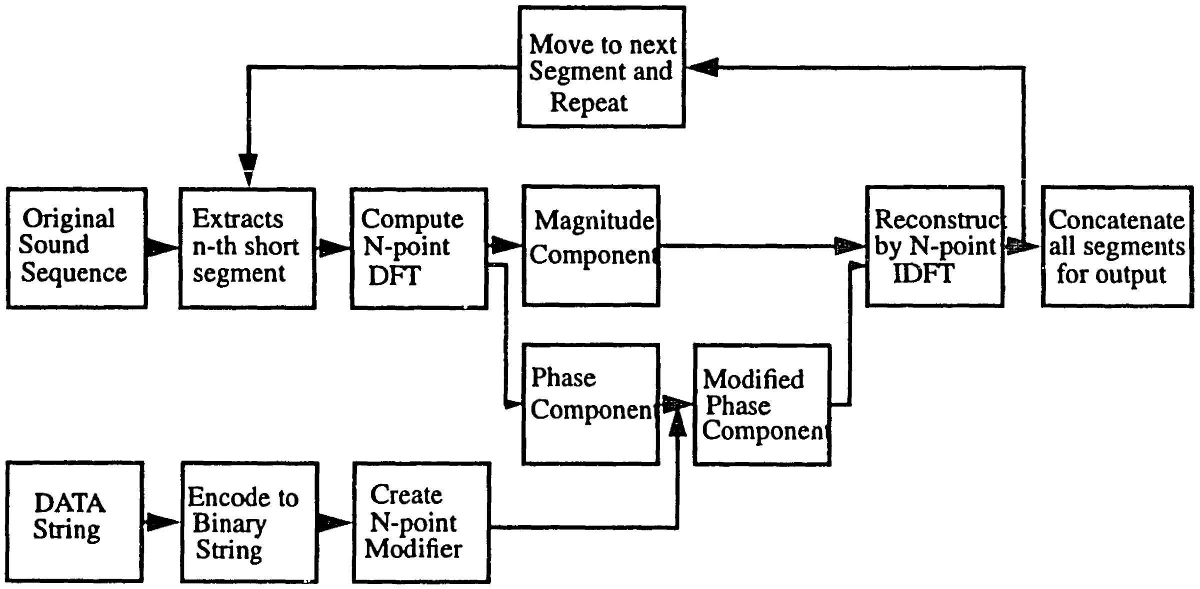


Figure 4.2: Block Diagram of the Phase Encoder

Decoding Scheme

In this coding scheme, a sound sequence consists of several consecutive short segments is used to embed one set of information (a coded binary string with N-points length). Since the same binary string is used for the coding for all segments in the sequence, the modification applies to the phase matrices for all these segments are the same. Thus, the embedded binary string can be detected by analyzing the underlying modification that applied to the phase matrices in all of the coded segments. This is shown in Figure A in Appendix A. (a) is the phase components obtained from one encoded segment. The underlying modification to the phase matrix can be found among a randomly distributed phase values by using the histogram analysis of the phase values. At the decoder, the random noise can be reduced by averaging the phase matrices over several segments. The result of the averaging over seven segments are shown in (b). As the spacing corresponding to one bit is known, the binary string can be obtained by segmenting the averaged result and apply the thresholds. In this example, [1101] was detected as a embedded information.

Since the detected phase components have some amount of noise, it is difficult to code one bit to each frequency slot. In this example, 32 frequency slots are used for each bit in order to increase the robustness of the data. (see section 4.2.1).

The coded sequence is divided into M segments that contain N data points in each as it was divided at the encoding process (the length of the short-time segments (N) and the number of the coded segments (M) must be known as a decoding keys). The modulation that been applied to the sequence is undone, one segment at a time. The phase matrices are obtained by computing N-point DFT of each segment. From these matrices, the coded information can be extracted by averaging it over all coded segments. Finally, the binary string is decoded.

The block diagram of the decoding process is shown in Figure 4.3.

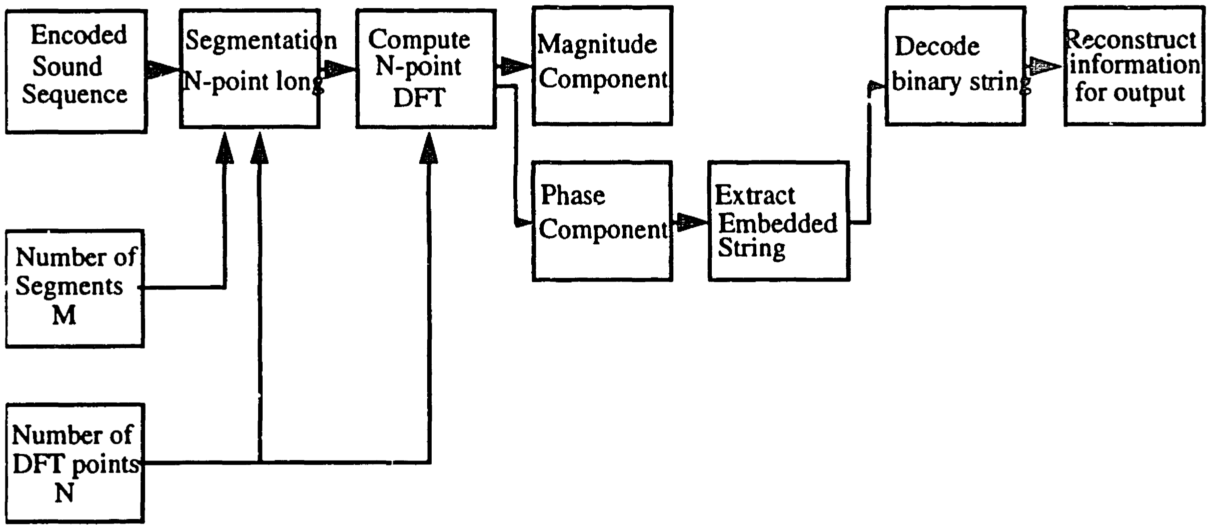


Figure 4.3: Block Diagram of the Decoder

Offset the Initial Phase Matrices

One option to this coding scheme is to offset the entire phase matrix of the initial segment to zero instead of applying the phase modifier to the original phase matrix. Thus, the embedded code can be directly detected from the first segment of the coded sequence without averaging and restorations. Since we are forcing many frequency components to be the same phase, a heavy distortion will be introduced to the first segment. The distortion decays along with the time axis by incorporating the phase difference of the original segments. In order to apply this method: (1) The length of the segments must be short enough so that the distortion of the first segment can be ignored. (2) In order to prevent the persistent resonance effect due to the equalization of the phase components, the frequency content in the subject sound signal should be randomly changed (the distortion persists when apply this method to a music sequence that featuring a single instrument).

Measurement of Perceptible Noise Level

As mentioned previously, the distortion introduced by the Phase Coding technique does not directly associate with the increase of the perceptible noise. The comparison of the Sigma Square Error of the signal and the Perceptible Noise Factor with different data rate are shown in Figure 4.4. The definition of the Q_{local} is shown in equation (5.2). The sigma square error is computed as follows:

$$e^2 = \sum_{k=1}^{L} [x(k) - s(k)]^2 \tag{4.4}$$

where $x(k)$ is encoded sequence and $s(k)$ is the original sequence. The result shows that the error of the signal increases when the data rate of the phase coding increase, however, the perceptible noise does not increase.

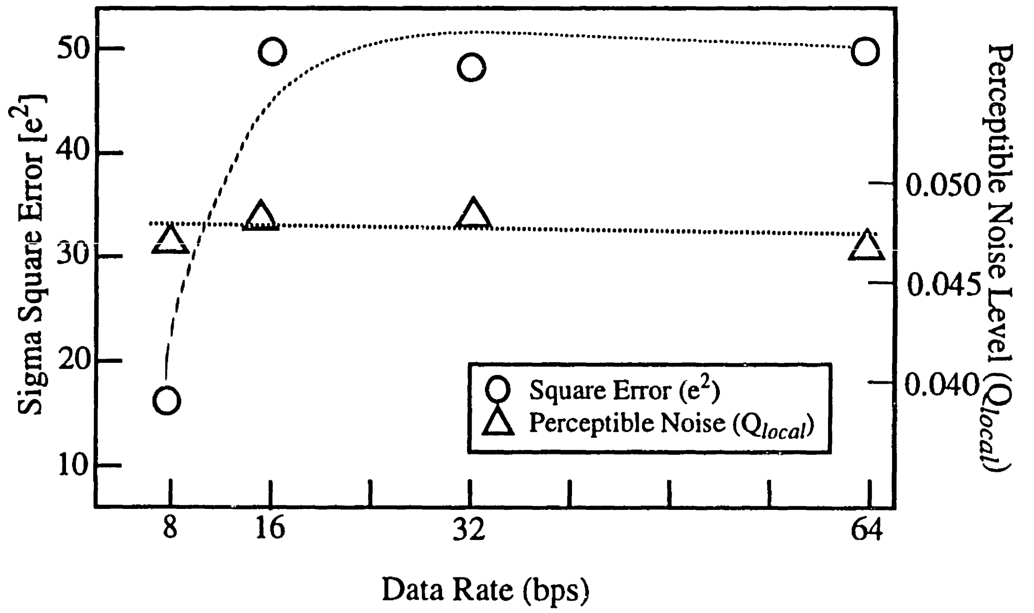


Figure 4.4: Perceptible Noise vs. Square Error

4.2.1 Definition of Coding Parameters

In this section, the definitions and characteristics of several key parameters are described. Since the values of some parameters depend on each other, experiments were needed to determine optimum values. (All of the sound signals used in these experiments are sampled with 8kHz.)

Discrete-Time Fourier Transform (DFT)[i] Points

Since the number of the DFT points directly corresponds to the data space, it is preferable to have as many DFT points as possible. However, the increase of DFT points results in the increase of the length of the DFT segment, which results in the decrease of the frequency resolution of the DFT. As the frequency resolution is one of the crucial parameters of the phase coding, the optimum number of the DFT points needs to be defined by examining the detectability of the embedded code.

The optimum number of DFT points for the phase coding was obtained as follows: First, apply phase coding with the same data rate to several sound segments with different lengths. 5-10% of random noise is added to the coded segments before decoding. The error rate at the decoder is collected. The segments with 800-1,100 DFT points show the best performance. In order to take advantage of the Fast Fourier Transform (FFT) algorithm, 1,024 (2^{10}) was selected. The ratio of the correctly detected bits measured from the test is shown in Figure 4.5.

i. since STFT is performed in discrete time, it is referred as Discrete-time Fourier Transform

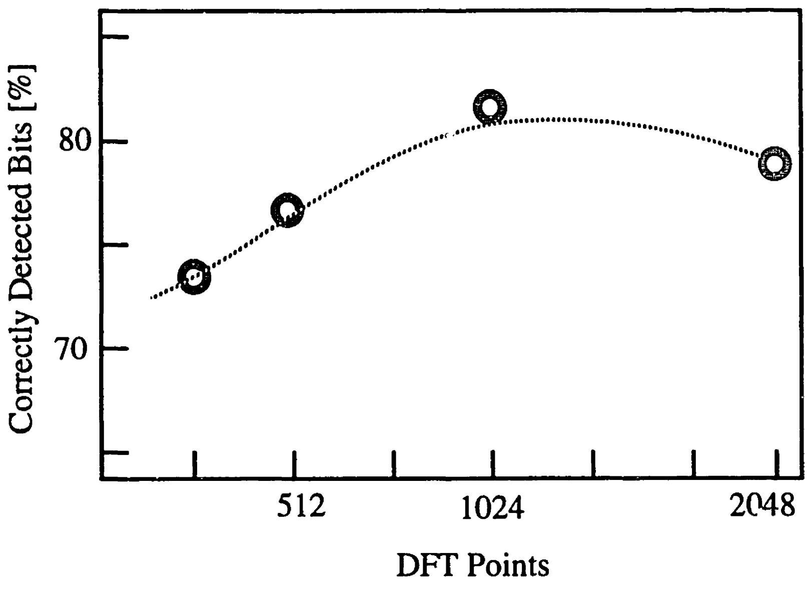


Figure 4.5: DFT Points and The Code Detectability

Phase Modifier

In order to embed data into a phase matrix, a distinguishable modification must be applied. One way to apply the modification is to shift the phase of the frequency slots corresponding to 1 of the coded binary string for certain amount and leave the phase of the rest of the frequency slots unchanged. The amount of the shift should be great enough so that the modified phase components can be distinguished from the unmodified components. By examining the distortion of the sound being introduced by the phase shift, the maximum amount of the shift is determined to 1 (radian). As a result of the linear shifting, the absolute value of the phase may exceed π. In that case, the value of the phase wraps into a value within +/-π. The sign changes. In order to avoid the confusion of the sign of the phase, the phase value is clipped at +/−π.

In general, the phase modifier can be expressed as follows:

$$d(\omega k) = Ab(k) + B \tag{4.5}$$

where A is the attenuation factor of the modifier and B is the amount of the shift. In above case, A = 1 and B = 0 when $b(k)$ is the coded binary string.

Another way to modify the phase is to shift the phase components that correspond to 1 in the coded binary string by $\pi/2$. This is basically same as the previous method except the phase shift is $\pi/2$ instead of 1. The advantage of this method is its simplicity of computation. The modified Fourier Transform F_{new} can be obtained simply by replacing the real part and imaginary part of the original Fourier Transform result as follows:

$$Im\{F_{new}\} = Re\{F\} \tag{4.6}$$

$$Re\{F_{new}\} = -Im\{F\} \tag{4.7}$$

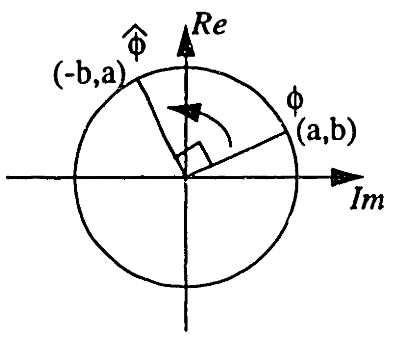


Figure 4.6: Phase Modification

In either case, the phase will b changed from random distribution to dual-peak distribution. The histogram of the phase components for an original and an encoded segment are shown in Figure B in Appendix A. The way to make these peaks distinguishable is either increase the amount of modification or reduce the variance of the distribution of the histogram by averaging the phase matrices over the segments.

Frequency Slots Per Bit

With N-point DFT, theoretically, we can use up to N-frequency slots of the phase matrix for the coding. However, as mentioned previously, because of the noise in the decoded phase in a typical sound waveform, it is almost impossible to code one bit in one frequency slot. Moreover, the modification of the phase done to each frequency component will cause severe phase dispersion. Therefore, it is more realistic to use more than one consecutive frequency slot for each bit of code. More frequency slots can decrease the distortion caused by phase dispersion and provide the better detectability of the embedded code as well. Setting the DFT points to 1,024 and the magnitude of the phase modifier to 1, the minimum number of the frequency slots per bit was found to be 32. This is equivalent to 256bps (the sampling frequency of the sound segment is 8kHz).

4.2.2 Summary of Phase Coding

The key parameters defined in the previous sections are summarized as follows:

Item	Value
DFT Points	1,024
frequency. slots per bit	32
Phase Modifier [radian]	1
net bit per second [bps]	256

Table 4.1: Summary of the Spread Spectrum Coding

In order to make the embedded data robust to the frequency selective filters, the data can be coded redundantly in the high frequency region and low frequency region. In this case, the net bit per second will be 128bps.

4.3 Spread Spectrum Coding

Spread Spectrum (SS) is a powerful technique for jam-resistant coding. The primary strategy of the SS system is to create an orthogonal communication system complex that can replace a numbers of transmit-receiver pairs in different frequency bands.[3,8,9] In other words, SS system is literary spreading the power spectrum of the encoded signal to a wide frequency range in order to make the unexpected interception of the encoded information difficult.

The coded information will be modulated by a stationery carrier wave and a sub-carrier which is a pseudo random sequence (PRS). Since the PRS is a sequence that has wide frequency spectrum, the frequency spectrum of the encoded data can be spread to a wide frequency range (see Figure C in Appendix A). This modulation creates two major advantages against the unexpected detection (e.g., signal jammers): (1) a large frequency band needs to be monitored in order to get all the information (2) there is low power density in each detected signal, and the possibilities of the interception of the signal can be made fairly low. Moreover, the PRS used for the encoder is also used as a "key" for decoding. It is almost impossible to extract any information out of the encoded sequences without this "key".

The primary purpose of the SS technique is jam-rejection where the data hiding is distortion resistance. The similarity between data hiding in audio and the work on SS communication is to hide a signal with lower frequency in a signal with higher frequency. The key to the successful SS coding is to achieve a perfect signal synchronization and alignment in the temporal axis. Without it, the demodulation process can not be done. This is a big challenge with analog signals, because in a continuous signals, it is difficult to obtain a definitive time index. However, a digital signal self-aligns. Each data point has a defined time axis in discrete-time domain. Therefore, the SS techniques adapt well to the data hiding although it is primarily designed for analog signals.

After the study of the various SS techniques, it was concluded that the Direct Sequence Spread Spectrum (DSSS) is best suited to the application of the data-hiding process. DSSS is one of the most simple and powerful techniques among SS techniques. In DSSS, the achievement of the perfect signal alignment is the critical factor related to the data rate. As mentioned previously, in the digital domain, the signal alignment can be virtually ignored since the data of the signal are in discrete-time environment. As a result, a high data rate can be achieved with DSSS technique.

In this section, the concept and the coding schemes of the DSSS techniques are described as well as the definition and the properties of several key parameters.

4.3.1 DSSS Coding Scheme

DSSS Code Synthesis

In DSSS coding, the encoded sequence is generated by adding the synthesized DSSS signal (code) to the host signal. The code is synthesized by using a carrier wave, Pseudo Random Sequence (PRS) and the coded binary string. It is then attenuated to less than 1% of the dynamic range of the host signal before being added to the host signal. The block diagram is shown in Figure 4.7.

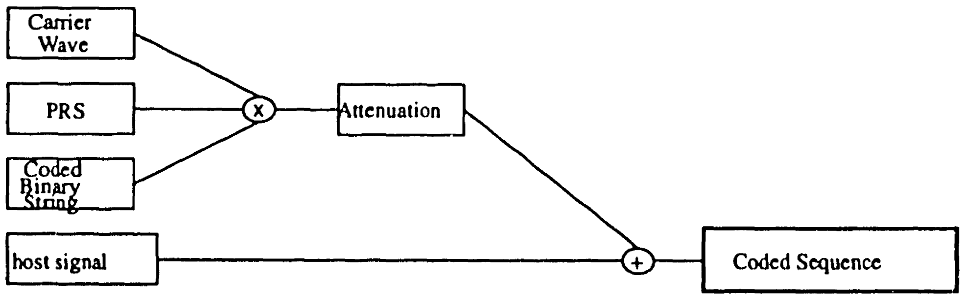


Figure 4.7: Block Diagram of DSSS code synthesis

A carrier wave with a stationery frequency (f_c) and PRS generated with a known "seed" is used for both the modulation of the code at the encoder and the demodulation at the decoder. The PRS modulates the code to spread the power spectrum to a wide frequency range. All of the elements used for the synthesis of the code are range from -1 to +1, so that the multiplication of the "chip" and "information" flips the phase of the carrier wave by 180 degree (or π). The alternated phase of the carrier wave can carry the binary bit. This is called "bi-phase shift keying." An example of the synthesized DSSS code is shown in Figure 4.8.

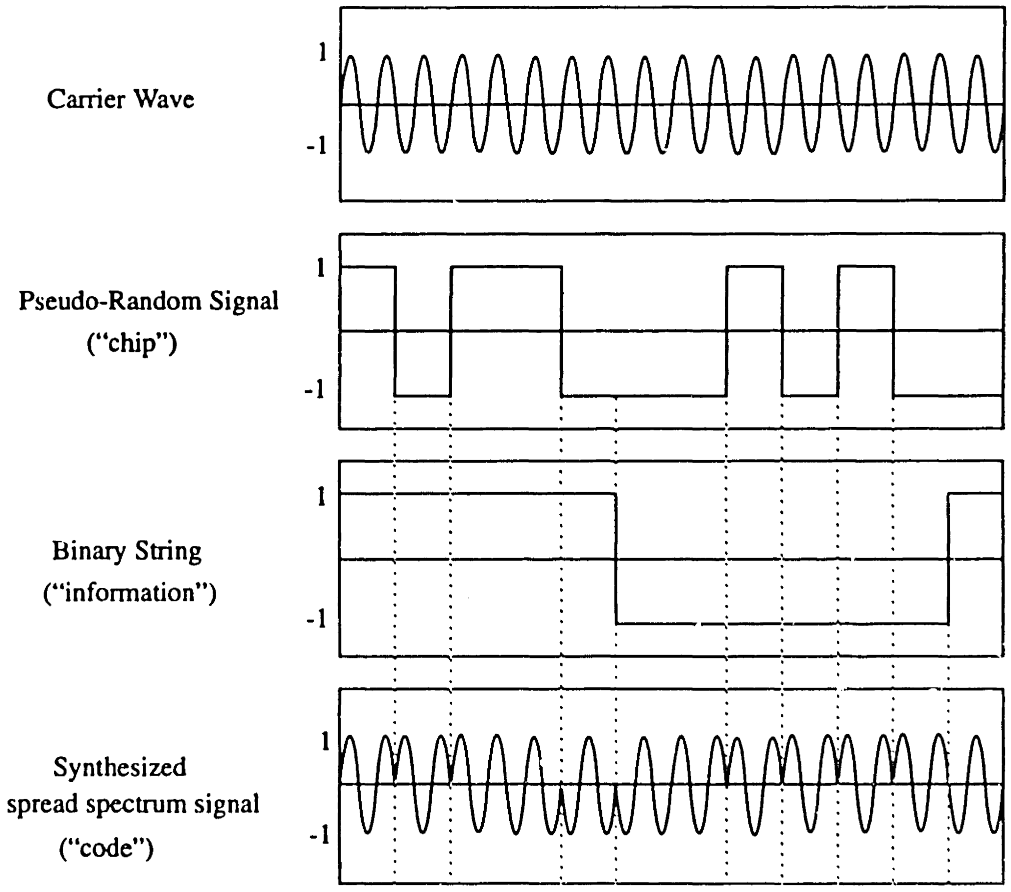

Carrier Wave
1
-1
Pseudo-Random Signal ("chip")
1
-1
Binary String ("information")
1
-1
Synthesized spread spectrum signal ("code")
1
-1

Figure 4.8: Synthesized spread spectrum information encoded by the DSSS.

Decoding Scheme

DSSS employs bi-phase shift keying. The phase of the carrier wave alternates between [ϕ+0/ϕ+π] where the ϕ is the phase value of the original segment. Since the coded carrier wave only has two possible phase values (0 or π), it is easy to extract the data. The vague phase values can be excluded from the decoding to increase the SNR of the decoding process.

The demodulation of the encoded segment is done by multiplying the PRS used for the encoding to a properly aligned segment. This will "unspread" the power spectrum of the encoded segment. A frequency selective filter with pass band of f_c is applied to the segment to extract the carrier wave. The phase of the carrier wave in each chip duration is measured and interpreted to binary code. For the process described above, the following parameters must be known at the receiver: the decoding key (PRS or the seed of the PRS), chip rate, data rate, carrier frequency, and the sampling frequency.

The block diagram of the decoding process is shown in Figure 4.9.

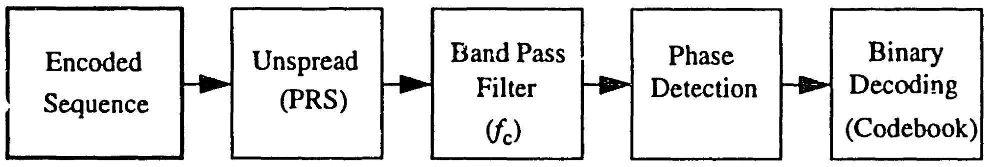


Figure 4.9: Block Diagram of DSSS decoding process

4.3.2 Definition of Coding Parameters

Carrier Frequency

The carrier wave is a perfect sinusoidal wave with a stationery carrier frequency (f_c). The change of the phase of the carrier wave alternating between 0 and π carries the information as a binary string, therefore, an appropriate representation of the wave form is important. More data can be embedded if a higher carrier frequency is used. However, the carrier wave is represented by a limited number of points. At least 8 points are needed for one cycle of the wave form in order to obtain an accurate representation of the waveform. In 8kHz sampling sound, this is equivalent to f_c = 1kHz.

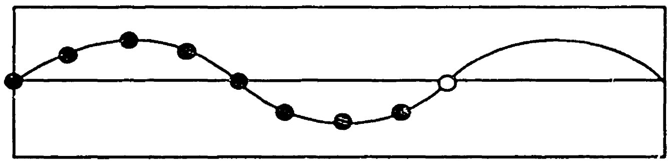


Figure 4.10: Carrier Wave

Chip Rate and Carrier Frequency

Since the data is embedded as the alternating phase of the carrier wave, the measurement of the phase is one of the most important aspects of the Spread Spectrum Coding. As described previously, each cut of the carrier wave to be measured has length which is limited by the length of the chip. The cut of the carrier wave should also contain sufficient cycles for the proper detection of the phase. Therefore, the frequency of the carrier wave should be limited to a certain range regarding to the minimum number of cycles and the length of the chip. Since the detectability depends on the sound content and the amount of the noise, experiments are needed to find the proper number of the cycles needed for the phase computation.

The data for the measured detectability vs. the number of carrier wave cycles is shown in Figure 4.11. The detect rate shows a peak between 2 and 2.5 cycles. Obviously, it is difficult to compute phase correctly if the carrier wave is not long enough. The results also show that too many cycles also makes the phase sensitive to the noise of the waveform.

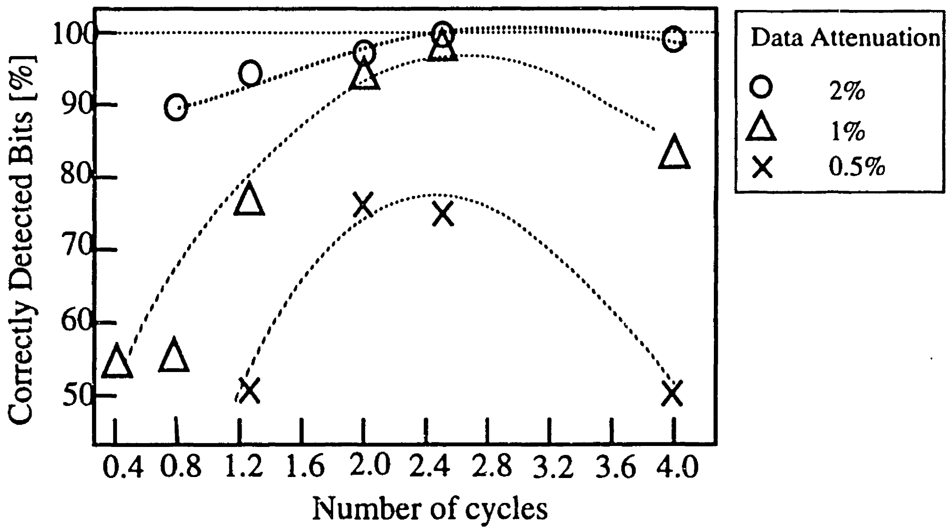


Figure 4.11: Number of Cycle of the Carrier Wave in one chip duration vs. detectability of the embedded binary bit (chip rate = 20)

Data Rate (Data/Chip Ratio)

Data rate is defined as a data/chip ratio which linearly corresponds to the SNR of the embedded data. If we consider the chip duration as the clock rate of the synthesized SS code, the data/chip rate is the number of clocks needed for one bit of information. A greater data/chip rate provides better SNR or higher noise susceptibility to the embedded code. A lower data/chip rate allows more information to be embedded. In order to define the optimum data/chip rate, the following experiments were done:

Figure 4.12 shows the effect of varying the data/chip ratio on the detect rate (without ECC), after the binary code detection with the attenuation factor 0.5%. At both chip rates, the detect rate saturates at Data/Chip = 20.

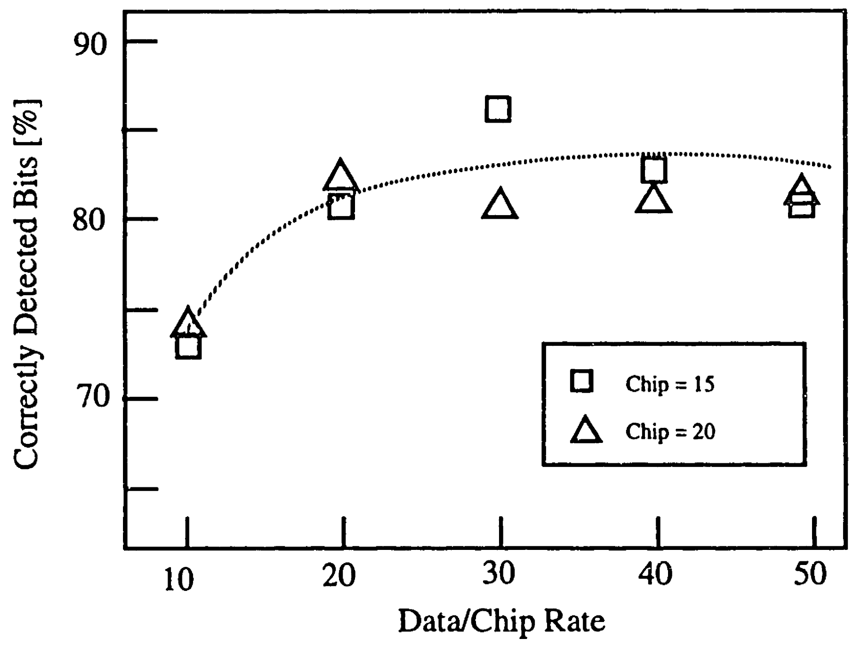


Figure 4.12: Data/Chip Rate vs. Detect Rate

4.3.3 Summary

The parameters defined in previous sections are as follows:

Carrier Frequency	8kHz
Chip Rate	20
Data/Chip Rate	20
Total Data Points/bit	400
net bit per second [bps]	20

Table 4.2: Summary of the Spread Spectrum Coding

The attenuation factor used here is 0.5% of the dynamic range of the sound segment, which is unrecognizable in general recordings such as news programs, interviews and speech. 20 bits of information can be embedded into a sound sequence of one second.

4.4 Analysis of the Code Invariance

In this section, the analysis of the robustness of the embedded data by using data-hiding techniques are discussed. Since the use of the LSB Coding is restricted in a closed digital environment without any signal distortion, the discussion focuses on Phase Coding and DSSS Coding. The modifications are limited to the following: low-pass/high-pass filtering, sub-sampling, quantization, and windowing.

Low-pass/High-pass Filtering

In order to simplify the discussion, the filters are assumed to be a Linear Time Invariant System (LTI System) which the impulse response of the system $h[n]$ is the impulse response of the modification.

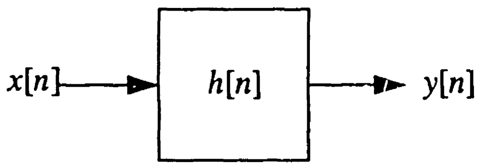


Figure 4.13: Linear Time-Invariant System

$x[n]$ is an encoded sound sequence, $h[n]$ is an impulse response of the modification and $y[n]$ is an out put of the system. In an LTI System, the output $y[n]$ can be expressed as a convolution of the input signal and the impulse response of the system.

$$y[n] = x[n] * h[n] \tag{4.8}$$

In frequency domain,

$$Y(e^{j\omega}) = X(e^{j\omega}) \times H(e^{j\omega}) \tag{4.9}$$

$$X(e^{j\omega}) = |X(e^{j\omega})| e^{j\angle(X(e^{j\omega}))} \tag{4.10}$$

where the $Y(e^{j\omega})$, $X(e^{j\omega})$ and $H(e^{j\omega})$ are the frequency response of the corresponding signals and $<(X(e^{j\omega}))$ is the phase component of the $X(e^{j\omega})$. The equation (4.8) then can be rewrite by using the equation (4.9) as follows:

$$Y(e^{j\omega}) = |X(e^{j\omega})| e^{j\angle(X(e^{j\omega}))} \times |H(e^{j\omega})| e^{j\angle(H(e^{j\omega}))} \tag{4.11}$$

$$Y(e^{j\omega}) = |X(e^{j\omega})| \times |H(e^{j\omega})| e^{j\angle(X(e^{j\omega}) + H(e^{j\omega}))} \tag{4.12}$$

As shown in equation (4.11), when the signal passes through the system, the new phase of the output signal is the summation of the original phase and the phase of the frequency response of the system.

In the Phase Coding technique, the coding applied to the original sequence s[*n*] can be expressed as follows:

$$X(e^{j\omega}) = |X(e^{j\omega})| \times e^{j\angle X(e^{j\omega})} = |S(e^{j\omega})| \times e^{j\angle(S(e^{j\omega}) + \phi_{mod})} \tag{4.13}$$

where the ϕ_{mod} is a phase modifier applied to the phase matrices of the original sequence. By using equation (4.11), the modified sequence with the system $H(e^{j\omega})$ can be expressed as:

$$Y(e^{j\omega}) = |X(e^{j\omega})| \times |H(e^{j\omega})| e^{j\angle(S(e^{j\omega}) + H(e^{j\omega}) + \phi_{mod})} \tag{4.14}$$

This shows that the information embedded in a phase matrices (ϕ_{mod}) can be seriously disturbed by the distortion to the phase due to the modification system.

One example of a filter is a moving average model. A low-pass filter can be modeled by N-point moving average as follows:

$$y[n] = \frac{1}{M1 + M2 + 1} \sum_{k=-M1}^{M2} x[n-k] \tag{4.15}$$

where M_1 and M_2 are the number of points before and after the current index *n* that are used for the moving average. In this case, $M_1 = 0$ and $M_2 = 4$. The phase of the frequency response of the system $H_{lp}(e^{j\omega})$ is shown in Figure 4.14. In a Finite Impulse Response Filter (FIR Filter), if we assume the frequency response is real and non-negative, the filter will have zero-phase characteristics and the phase of the input signal will be modified linearly with the filter. It will result in simple time shift of the signal. However, even in this case, the phase component in the high frequency region will be highly modified, and the information embedded in that region will be missing unless the system function of the modification is known at the receiver.

In order to minimize the effect of the frequency selective filter, the code needs to be embedded into several different frequency bands with relatively small size in each matrix. This will reduce data space.

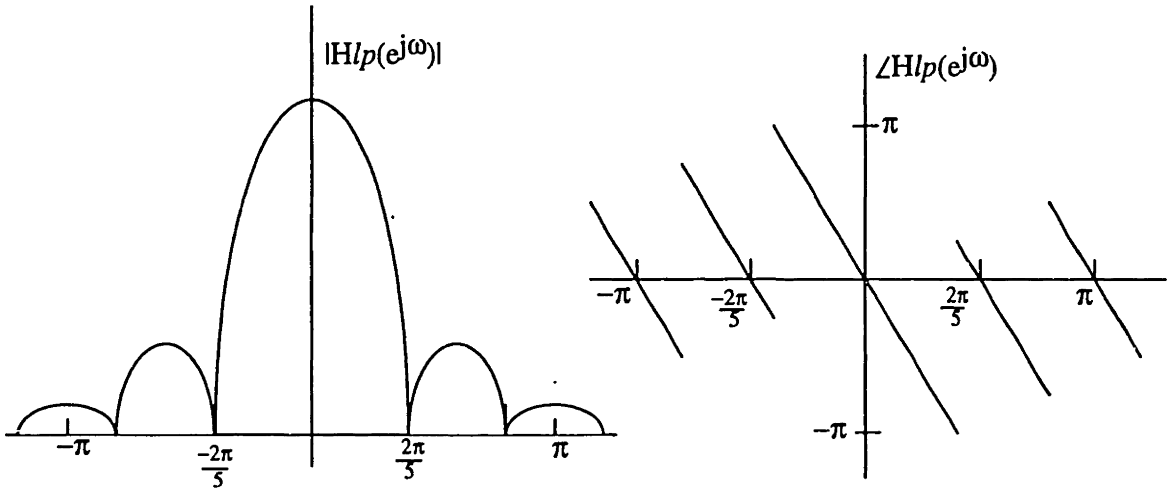


Figure 4.14: Frequency response and the phase of the 4-points moving average filter

In the case of DSSS Coding, if the phase component that carries the information is in one frequency component, the modification will simply shift the phase value linearly. The problem after the modification is to distinguish two peaks in the phase distribution after the modification. This can be done by analyzing the histogram of the phase component in the modified sequence and establish new threshold levels to distinguish two major levels which representing the embedded binary code.

Re-Sampling

Re-sampling or sub-sampling is another common manipulation of an audio signal. The sampling frequency for typical audio signals ranges from 8kHz to 44kHz. Re-sampling does not change the property of the sound waveform as long as the sampling rate is twice as high as the Nyquist rate.

In both Phase Coding and DSSS Coding, the re-sampling will change the number of the data points in a coded segment. Therefore, the length in time of the encoded segment must be known at the decoder for appropriate segmentation.

In case of Phase Coding, the re-sampling will change the DFT points. Frequency slots that were not in the original slots will become noise. When the signal is up-sampled by an integer, the phase will be detected as encoded data plus noise. Since the added noise is random, it can be reduced by averaging an increased number of the segments. If the signal is down sampled, the information embedded in the phase matrices will be completely destroyed.

In case of DSSS Coding, as long as the length in time of the encoded segment is known, the embedded signal can be recovered. The limitation is the representation of the carrier frequency (see section 4.3.2). Assuming the acceptable detection rate is 90%, the appropriate number of cycle of the carrier wave in a duration of the chip is between 1 and 4. If there are two cycles of carrier wave in an original chip duration, the code will have 90% detection rate after either up sampling or down sampling by a factor of two.

Windowing

Windowing is a common manipulation. A complete set of encoded segments must appear in a windowed segment in order to be a valid code. The windowing operation can be expressed as a multiplication of two segments:

$$y[n] = x[n] \times w[n] \tag{4.16}$$

$$w[n] = \begin{cases} 1, & t_1 < n < t_2 \\ 0, & \mathit{otherwise} \end{cases}$$

$w[n]$ is windowing function, t_1 is the upper boundary and t_2 is the lower boundary of the selected segment. In both Phase Coding and DSSS Coding, the segment alignment is crucial. Therefore, synchronization signal must be embedded as often as possible throughout the sequence. Twice a second would be appropriate.

Re-quantization

Signal distortion increases when a digital signal is re-quantized by coarser steps. By looking at the quantization noise as an additive noise $e[n]$, the input and output signal can be expressed as follows:

$$y[n] = x[n] + e[n] \tag{4.17}$$

$$Y(e^{j\omega}) = X(e^{j\omega}) + E(e^{j\omega}) = \left|X(e^{j\omega})\right| e^{\angle X(e^{j\omega})} + \left|E(e^{j\omega})\right| e^{\angle E(e^{j\omega})} \tag{4.18}$$

The frequency of the error signal increases when the frequency of the input signal increases. The phase component in high frequency region is more sensitive to this additional high frequency error signal. Therefore, in Phase Coding, the use of the high frequency region should be reduced in order to avoid serious distortion. In DSSS coding, since the carrier frequency is usually in a relatively low frequency region (1kHz), the re-quantization will not have serious effect on the embedded data (assuming the smallest quantization level is 8-bit).

The summary of the analysis of the code invariance is shown in Table 4.3.

	Phase Coding	DSSS Coding
Low-/High-pass Filtering	need code redundancy in frequency slots	histogram analysis in decoding stage
Re-Sampling	only up-sampling by integer can be recovered	limited by the carrier wave ($\uparrow$2 or $\downarrow$2)
Windowing	more frequent signal synchronization/alignment	
Quantization	utilize smaller subset in low freq. region	not affected if minimum quantization is 8-bit

Table 4.3: Summary of the Code Invariance

4.5 Summary

Three different techniques have been discussed in this chapter and each technique has a unique property that is suitable for different applications. For LSB Coding, the major advantage is its high data-rate and the simplicity of the process. The disadvantage is the lack of the immunity to the signal modification/manipulation. The Phase Coding technique has the advantage that the code can be embedded imperceptibly. However, the code is not robust to signal manipulation such as filtering and re-sampling. The DSSS technique is the most robust technique against the signal modifications among three. The disadvantage of this technique is its low data-rate and the introduction of an audible noise.

Although these technique demonstrate a certain amount of success in some aspect of the data hiding applications, there are severe limitations with each technique. Some supplementary techniques are needed to improve the flexibility and enhance the performance of the coding.

Chapter 5

Supplementary Techniques

In order to apply data hiding to a practical system, several supplementary techniques are needed to support the core techniques described in the previous chapter. These techniques are important in that they provide flexibility to the parameter adjustment and they also improve the integrity of the data, efficiency and robustness of the entire system. In order to build a system that can adapt to real world applications, the following questions need to be answered:

(1) What is the best way to provide a synchronization of the encoded data string?

(2) How can a realistic trade-off point of the optimum redundancy (data rate) and the robustness of the code be defined?

(3) How can the perceptible noise level that is caused by the data-hiding coding be reduced without losing the power of the embedded code?

(4) How can the precise time-scale be regained once the signal is converted to the analog domain?

These questions are all important and need to be answered. In this chapter, several supplementary techniques have been developed in an attempt to answer some aspects of these questions are discussed. The techniques include code synchronization, random channel noise removal, error correction, and adaptive coding.

5.1 Techniques for Code Synchronization

It is crucial to obtain perfect synchronization (in other words, the precise starting indices) of the coded segments at the receiver in order to extract the coded segment correctly. In both phase coding and SS techniques, the decoding of the signal can be done correctly only if the encoded segments are extracted correctly. Although the digital signals hold the advantage of the signal discreteness in time, perfect synchronization is still difficult because the synchronization signal has to be transparent to the host signal. Two techniques developed for the data synchronization are described in this section.

Silent Beep

One simple way to embed a synchronization signal is to provide a "mark" or a beep at the beginning points of the coded segments. The "beep" can be made inaudible if the frequency (f_{beep}) is lower than 40Hz, which is the minimum audible frequency of the HAS. For the detection of the synchronization signal, a frequency selective filter is used to extract the sound wave component with frequency f_{beep}. The extracted sinusoidal waveform should have zero-crossing points separated from each other by half of the cycle; and by knowing the number of the cycles before the start of the signal, the starting point of the coded segment can be indicated. This idea is illustrated in Figure 5.1.

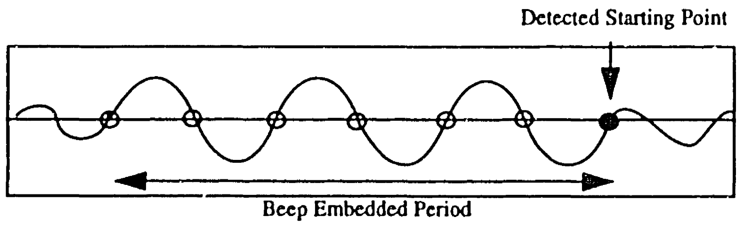


Figure 5.1: Extracted wave form with band pass filter (pass band frequency is f_{beep})

In this example, three cycles of the low frequency beep is embedded into the section indicated as "Beep Embedded Period". The wave form extracted by using the frequency selective filter (pass band frequency $= f_{beep}$) will have several zero-crossing points marked by "o". The adjacent zero-crossing points in the "Beep Embedded Period" will be uniformly spaced, and thus can be distinguished from the segments that has no beep embedded. If the number of the cycle and the frequency of the sinusoidal beep embedded in front of the encoded segment is known as three, the start point of the encoded segment can be easily detected (in this case the starting point of the encoded segment will be seven-th zero-crossing). Since the cycle of the beep is long, it takes a large amount of precious data space away, (3 cycles of 20Hz beep takes 150ms or 1200 points of data in 8kHz sampled sound). Also, the long synchronization is not robust to the cropping or trimming of the segments. Therefore, high beep frequency is preferred, as long as the beep is inaudible (in our case f_{beep} = 40Hz).

A few bits of misalignment may remain after the synchronization by the silent beep technique. In SS technique, the misalignment will result in a spiky noise at the receiver and can be removed by applying a medium filter, as long as the width of the chip is substantially larger than the width of the spike. In the case of phase coding, a few bits of the uniform shift may not cause any significant distortion to the detection since the information is embedded by using a relative relationship of the phase components in different frequencies. A conventional Phase Locked Loop (PLL) technique is also one of the options to acquire the fine synchronization in combination with the rough synchronization done by the silent beep.

The silent beep method is fairly simple and provides sufficient synchronization for all of the core techniques. However, since the beep resides in a narrow, low frequency band, it can be removed by applying a high-pass filter.

5.2 Error Control Coding (ECC)

In data-hiding applications, because of the critical importance of finding an optimum trade-off points between the data capacity and the distortion of the signal, the reduction of error is crucial. ECC is a coding method intends to minimize errors between the transmission and the receiver by adding extra bits to introduce the redundancy for error detection and correction. Since the capacity of the communication channel can be characterized by the capacity at which the information can be transmitted correctly, the effort of error reduction is equivalent to the expansion of the coding capacity.

The basic idea of the ECC is to introduce additional bits (ECC bit) to the code word which can be used as an error detector or corrector. The model of the ECC encoder is shown in Figure 5.2 (In this figure, 4-bit ECC code is added to an 8-bit code word and the total length of the code becomes 12-bit).

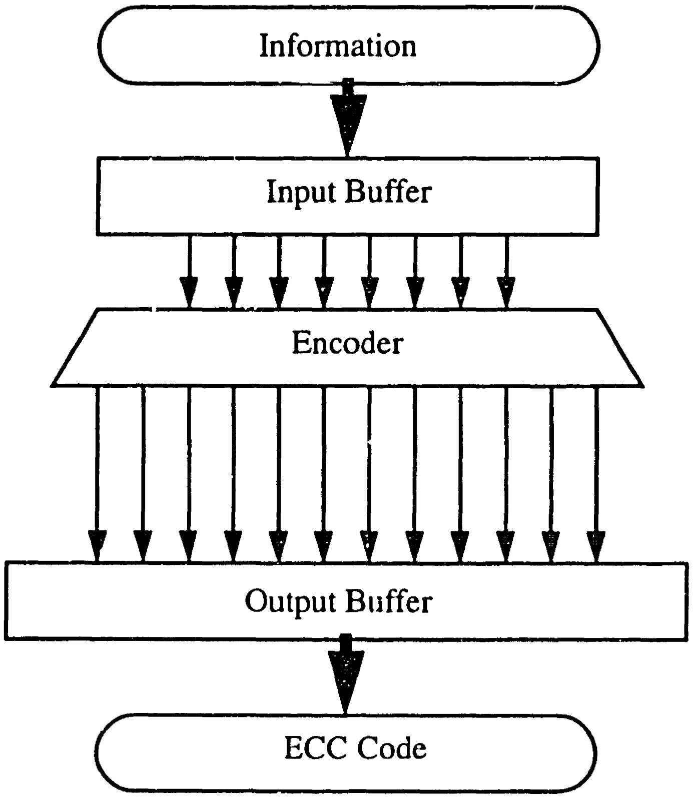


Figure 5.2: ECC Encoder Model

The Hamming Distance and the ECC

The Hamming Distance (d_H) is a total number of bits that are different from each other when comparing two codewords bit by bit. In a codebook, the minimum Hamming Distance (d_{Hmin}) is one of the most important properties. It indicates the distinguishability of each codeword from the others, thus, the codebook which has greater d_{Hmin} has higher capability to detect or coirect its errors. In random codebook, the d_{Hmin} is 1. If the d_{Hmin} of a codebook is 2, one bit of error in a codeword can be detected. In other words, a codeword contains one bit of error can be recognized as an incorrect codeword, instead of been misinterpreted to an incorrect information. If the d_{Hmin} of a codebook is 3, and the detected codeword contains one bit of error, the error can be corrected by choosing the codeword which has smallest Hamming Distance from the detected codeword. In this case, the only codeword which has $d_H = 1$ will be the corrected codeword. In general, the required d_{Hmin} for the correction of e errors is:

$$d_{Hmin} > 2 \times e \qquad (5.1)$$

Create a Code Book (Block Coding Method)

A full ECC code book can be generated by using the block coding method[6]. With this method, the ECC bits can be obtained and added to the end of the existing codewords to create a ECC codebook. One way to generate ECC bits is to set t+1 linearly independent codewords as a generator matrix (where t is the number of bits in the original codeword). The codeword with ECC bits can be generated by a simple mathematical operation of the generator matrix, so that we don't have to look into the whole codebook all the time. A simple way to find t linearly independent matrix is to select all of the codewords which have only one non-zero element. For example, if the original codeword is 4 bits, all we need to find is the following four codewords:

$$\underbrace{\begin{array}{c}0001\\0010\\0100\\1000\end{array}}_{\text{Original Codeword}}\ \underbrace{\begin{array}{c}111\\011\\101\\110\end{array}}_{\text{ECC Bits}}$$

Figure 5.3: Generator Matrix for Block Coding

From this generator matrix, any of the ECC codeword can be generated. For instance, the ECC codeword for [0110] can be generated by adding the second and the third codewords without propagating the carry bit to the next bit. The result of this operation is [0110 110].

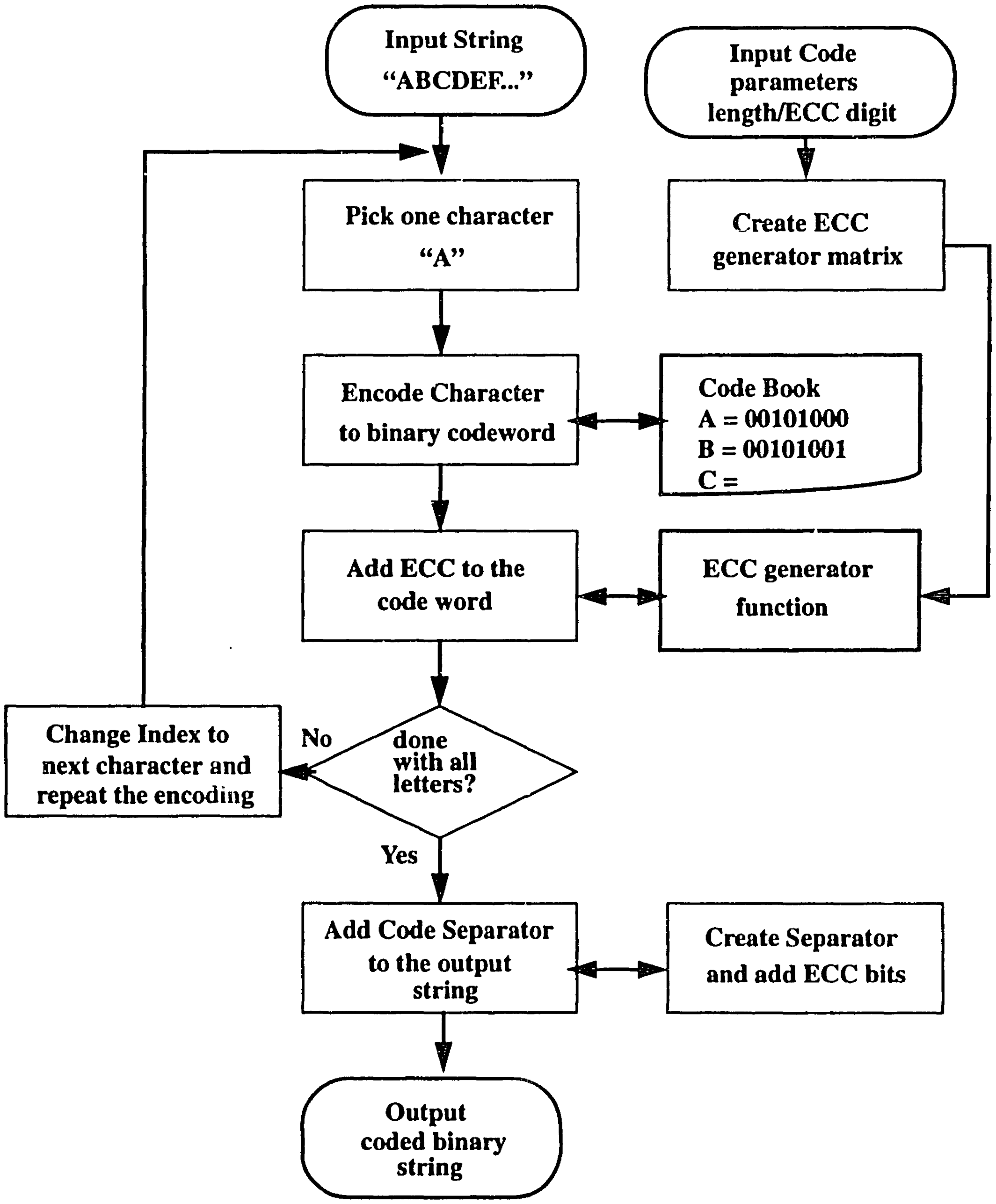
In our data-hiding system, a codebook with 8 bits is used. In that case, seven ECC bits are required for the codebook with $d_{Hmin} = 3$, which will be able to correct one error from each codeword. (see section 5.3). The generator matrix used for the generation of the codebook is shown in Figure 5.4.

```
00000001 1111111
00000010 0111111
00000100 1011111
00001000 1101111
00010000 1110111
00100000 1111011
01000000 1111101
10000000 1111110
```

Figure 5.4: Generator Matrix of ECC block coding for 8-bit codeword

Coding Procedure

The flow chart of the ECC encoder that has been used in this project is shown in Figure 5.5. The flow takes an alphanumeric character as an input, encode characters to binary codes and add ECC bits to the codewords from the generator matrix.



Input String "ABCDEF..."
Input Code parameters length/ECC digit
Pick one character "A"
Create ECC generator matrix
Encode Character to binary codeword
Code Book
A = 00101000
B = 00101001
C =
Add ECC to the code word
ECC generator function
Change Index to next character and repeat the encoding
No
done with all letters?
Yes
Add Code Separator to the output string
Create Separator and add ECC bits
Output coded binary string

Figure 5.5: Error Correction Coding Flow Chart

5.3 Information Block Separator

Since the detected binary string may contain several independent binary string blocks, an appropriate separator is needed to indicate the start/stop of each block. The selection of the separator must be made carefully, so that the string can be uniquely detected from a run-on string. An arbitrarily selected separator may not work if the rest of the codewords are not selected to make the separator unique. For instance, in 4-bit codewords, if [0000] is selected to be a separator, the [0000] should not appear anywhere in the code string. This means, if any of the codewords starting with [0-] is used, all of the codewords ending with [-000] must be avoid, and if any of the codewords starting with [00-] is used, all of the codewords ending with [-00] must be avoided, and so on. As long as this constraint is met, any separator is equally suitable. Therefore, the design of the separator is made by the selection of the codewords.

In our case, [00001111] was chosen as a separator, and the codewords were carefully selected to avoid any possibility of false detection of the separator. The resulting codebook is shown in Appendix A. This codebook contains more than 64 codewords, and can include all of the alphanumeric characters in the standard ASCII character set.

This method provides the coding with an extra capability. That is, with the designed separator, the binary string block can start at an arbitrary point once the synchronization of the bit-level is done

5.4 Adaptive Data Attenuation

As mentioned in Chapter 3, the level of perceptive noise varies largely as the content, the magnitude and the noise level change; and the maximum allowable noise level is different in different parts of the sound signal. Therefore, the magnitude of the modification (coding) can be increased substantially at the loud and noisy section of the sound without increasing the perceptive noise, and the modification(coding) can be suspended at the segment which is completely silent. This is called adaptive data attenuation. A temporal reference level of the modification extracted from a sound segment is shown in Figure 5.6.

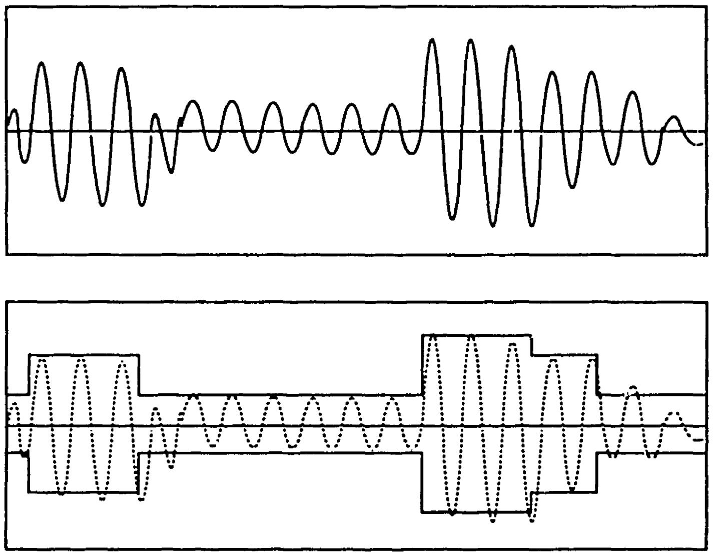


Figure 5.6: Adaptive Attenuation Envelop

This can be implemented by using the temporal magnitude of the sound as a reference level of the magnitude of the modification (coding). Since the maximum modification level of the data is increased, the detectability of the code in that particular segment can be increased. In order to take advantage of this, repetition of the coding should be done so that the larger part of the segment can have a better chance to be coded with greater magnitude, which will lead to a better detectability.

In a general news program recording, the maximum allowable noise level for the coding is approximately 0.05% of the dynamic range without the adaptive coding technique, since the noise level is limited by the relatively quiet part of the entire sound signal. With the adaptive data attenuation, the maximum allowable noise level in relatively loud sections can be raised up to 0.5% (a factor of 10).

Adaptive Coding with Perceptive Noise Factor

Since the allowable level of the additive noise in a entire sound signal depends on the magnitude and the noise content, the temporal noise level can also be used as a reference level of the adaptive coding. This is more useful when the distortion of the host signal is not linearly depending on the level of the modification such as phase coding. In order to apply the noise level to adaptive coding, an optimum way to analyzes the noise content of the sequence is needed. One way to characterize the perceptible noise is to compute the magnitude of the temporal change in the signal. The perceptive noise factor Q_{local} is shown as follows:

$$Q_{local} = \frac{1}{|Savg|} \times \frac{1}{N} \times \sum_{n=1}^{N-1} [s(n+1) - s(n)]^2 \tag{5.2}$$

In equation (5.2), N is the number of sample points in the sequence s(n) and S_{avg} is the average magnitude in the sequence. It simply measures the amount of the sharp spike noises in a sound signal and normalizes by the number of the points and the average magnitude of the sound signal. This provides a rough categorization of the noise level of the original sound and the detected level can be used as a reference level of the temporal adaptive coding.

The correlation between the Q_{local} and the perceptive level of the HAS is shown in Figure 5.7. The test is done by using several typical sequences from radio broadcasting such as news readings, speeches, and interviews. The sequence which has least noise was defined as level 0 and the sequence which has the most noise was defined as level 6 to set the reference level for the perceptive evaluation. Then the evaluated noise level vs. the Q_{local} was collected and plotted in Figure 5.7. The result shows a reasonable correlation between the Q_{local} and the perceptive noise level; therefore, it can be used as a rough index for the analysis of the noise content for the adaptive coding.

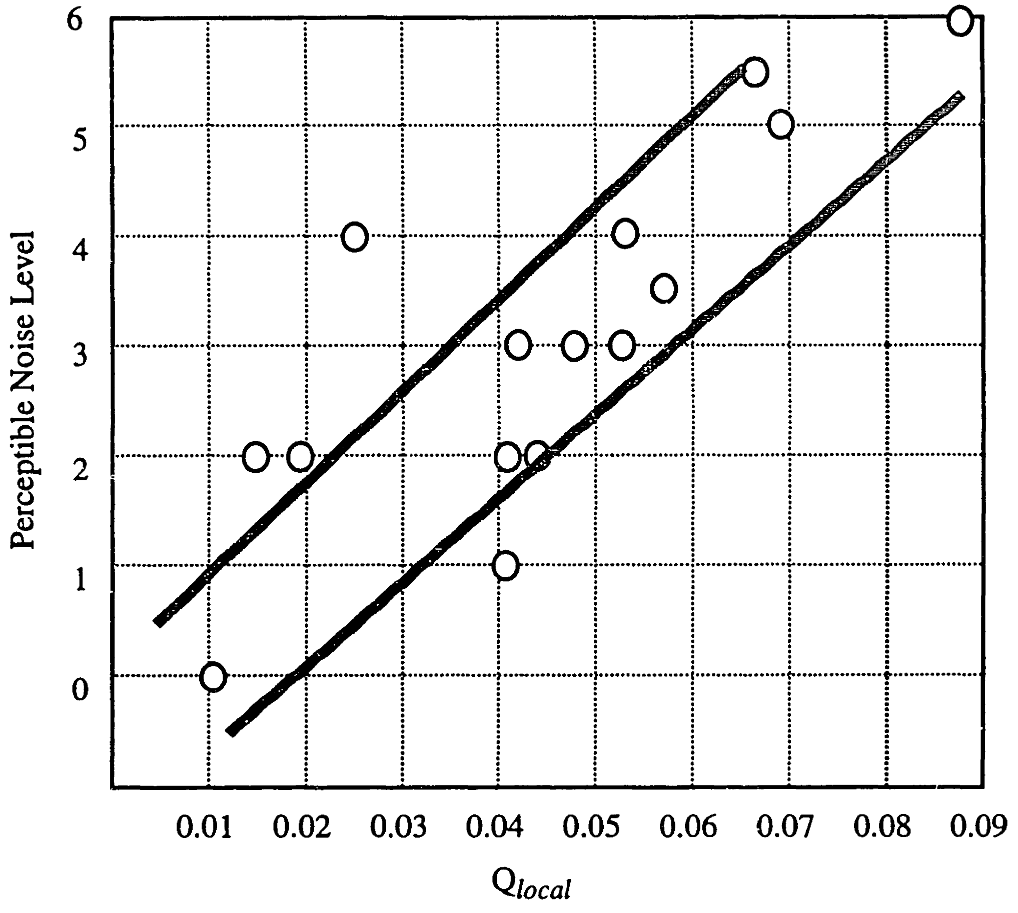


Figure 5.7: Correlation between noise factor (Q_{local}) and the human noise perception level (the noise level is set from 0 to 6 where 6 is the segment with maximum noise in its content)

5.5 Repetitive Coding (Averaging Method)

A sequence contains several segments that are coded by using the same set of the information is a coding unit. As mentioned previously, the noise can be reduced by averaging the detected result over all of the segments in a sequence. More noise can be reduced when the repetition increases, however, the effectiveness of the repetition gradually saturates.

In order to minimize the repetition, the effectiveness of the averaging vs. the number of repetition is measured. In an 8kHz format sequence, the effectiveness of the repetition (averaging) saturates at approximately 10 to 15 repetitio, s. This is shown in Figure 5.8.

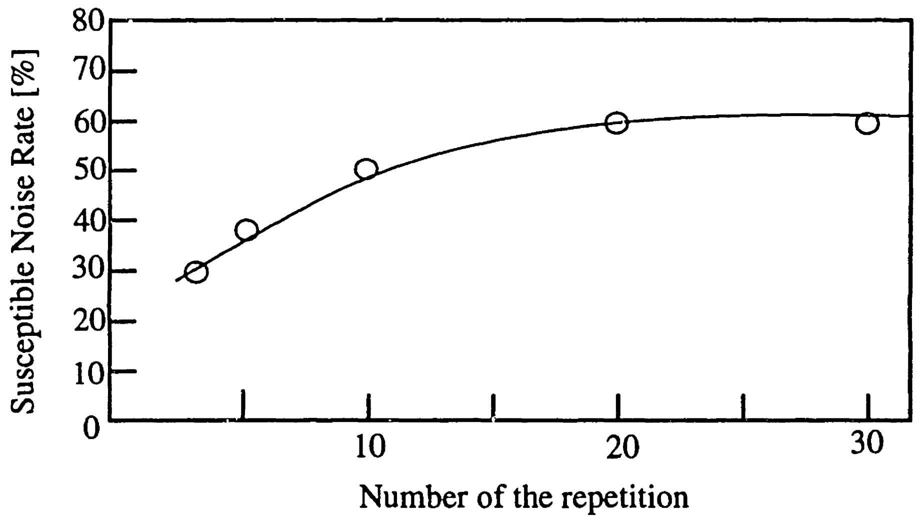


Figure 5.8: Number of the Repetition vs. Susceptible Noise Level

The noise rate is the ratio of the misdetected bits to the total number of bits in a detected binary string. The plotted values are the average of six different sets of codes.

5.6 Summary

The techniques described in this Chapter have been used in the data-hiding system and serving as a critical techniques to improve the performance of the system. Although these techniques are adequate for our experiments, more improvement needs to be done in order to implement a practical system. The system requires highly robust signal synchronization when the conversion to the analog domain is included in its transmission channel.

Chapter 6

Conclusion

Although the issue of the protection of intellectual properties cannot be solved by the technical approach alone, the data-hiding techniques provide ways to install a "lock" on the digitized media instead of leaving the door open.

The concept of data hiding has been introduced, and both the problem space and the required capabilities for widespread applications have been defined. We have also found that a technique suitable for one application is not necessarily suitable for other applications.

Three core techniques developed for the data hiding in audio signal were described. The LSB Coding technique is suitable for applications that require high data-rate and lower robustness such as captioning, whereas the Phase Coding technique is useful for applications that require minimum amount of the distortion to the original signal, and the SS Coding technique is suited to applications that requires certain amount of the robustness in response to the signal modifications. Each technique has its own properties so that can be used in different types of applications.

Besides the core techniques, several supplementary techniques are also developed to improve the performance of the core techniques. These techniques are important when the core techniques are being applied to practical applications. Many conventional techniques for one dimensional signal processing and communication theory are useful as are modification of these techniques. ECC and averaging technique are especially effective in improving the performance of the core techniques.

While we have had some degree of success in data hiding, all of the proposed methods have limitations regarding the robustness in response to modifications. For instance, the Phase Coding technique cannot be used when there is corruption in the phase relationship of the sound wave; and the SS Coding technique introduces audible noise. Therefore, the data-hiding technique developed for the sound signal may not be suitable to the protection of the copyright if it is not robust enough to the high-level interception or destruction. However, unlike the modification made in the case of images, in general, those applied to the sound signals are relatively primitive. Therefore, in many cases, the robustness of the embedded code is sufficient enough for these modifications as well as for many other useful applications. Content ID embedding, indexing, and captioning are probably the most practical applications of these data-hiding techniques.

In order to improve the data-hiding techniques, more work needs to be done to improve the robustness of the embedded data to various aspects of the signal processing. As one initial step toward this goal, more thorough technique for signal synchronization and alignment are needed for the system that including the analog digital conversions.

Appendix A

Eight Bit Code Book for Error Correction Coding

Char	Code	Char	Code	Char	Code	Char	Code
SPACE	00 00 00 00	:	00 10 00 00	T	01 00 00 00	n	01 01 11 01
!	00 00 00 10	;	00 10 00 10	U	01 00 00 10	o	01 10 00 00
"	00 00 01 00	<	00 10 00 11	V	01 00 01 00	p	01 10 00 10
#	00 00 01 01	=	00 10 01 00	W	01 00 01 01	q	01 10 00 11
$	00 00 01 10	>	00 10 01 01	X	01 00 01 10	r	01 10 01 00
%	00 00 10 00	?	00 10 01 10	Y	01 00 01 11	s	01 10 01 01
&	00 00 10 01	@	00 10 01 11	Z	01 00 10 00	t	01 10 01 10
'	00 00 10 10	A	00 10 10 00	[	01 00 10 01	u	01 10 01 11
(	00 00 10 11	B	00 10 10 01	\	01 00 10 10	v	01 10 10 00
)	00 00 11 00	C	00 10 10 10	]	01 00 10 11	w	01 10 10 01
*	00 00 11 01	D	00 10 10 11	^	01 00 11 00	x	01 10 10 10
+	00 00 11 10	E	00 10 11 00	_	01 00 11 01	y	01 10 10 11
,	00 01 00 00	F	00 10 11 01	`	01 00 11 10	z	01 10 11 00
-	00 01 00 01	G	00 10 11 10	a	01 01 00 00	{	01 10 11 01
.	00 01 00 10	H	00 11 00 00	b	01 01 00 01	|	01 10 11 10
/	00 01 00 11	I	00 11 00 01	c	01 01 00 10	}	01 11 00 00
0	00 01 01 00	J	00 11 00 10	d	01 01 00 11	~	01 11 00 01
1	00 01 01 01	K	00 11 00 11	e	01 01 01 00		
2	00 01 01 10	L	00 11 01 00	f	01 01 01 01		
3	00 01 01 11	M	00 11 01 01	g	01 01 01 10		
4	00 01 10 00	N	00 11 01 10	h	01 01 01 11		
5	00 01 10 01	O	00 11 01 11	i	01 01 10 00		
6	00 01 10 10	P	00 11 10 00	j	01 01 10 01		
7	00 01 10 11	Q	00 11 10 01	k	01 01 10 10		
8	00 01 11 00	R	00 11 10 10	l	01 01 10 11		
9	00 01 11 01	S	00 11 10 11	m	01 01 11 00		

Table 1.1: Code Book

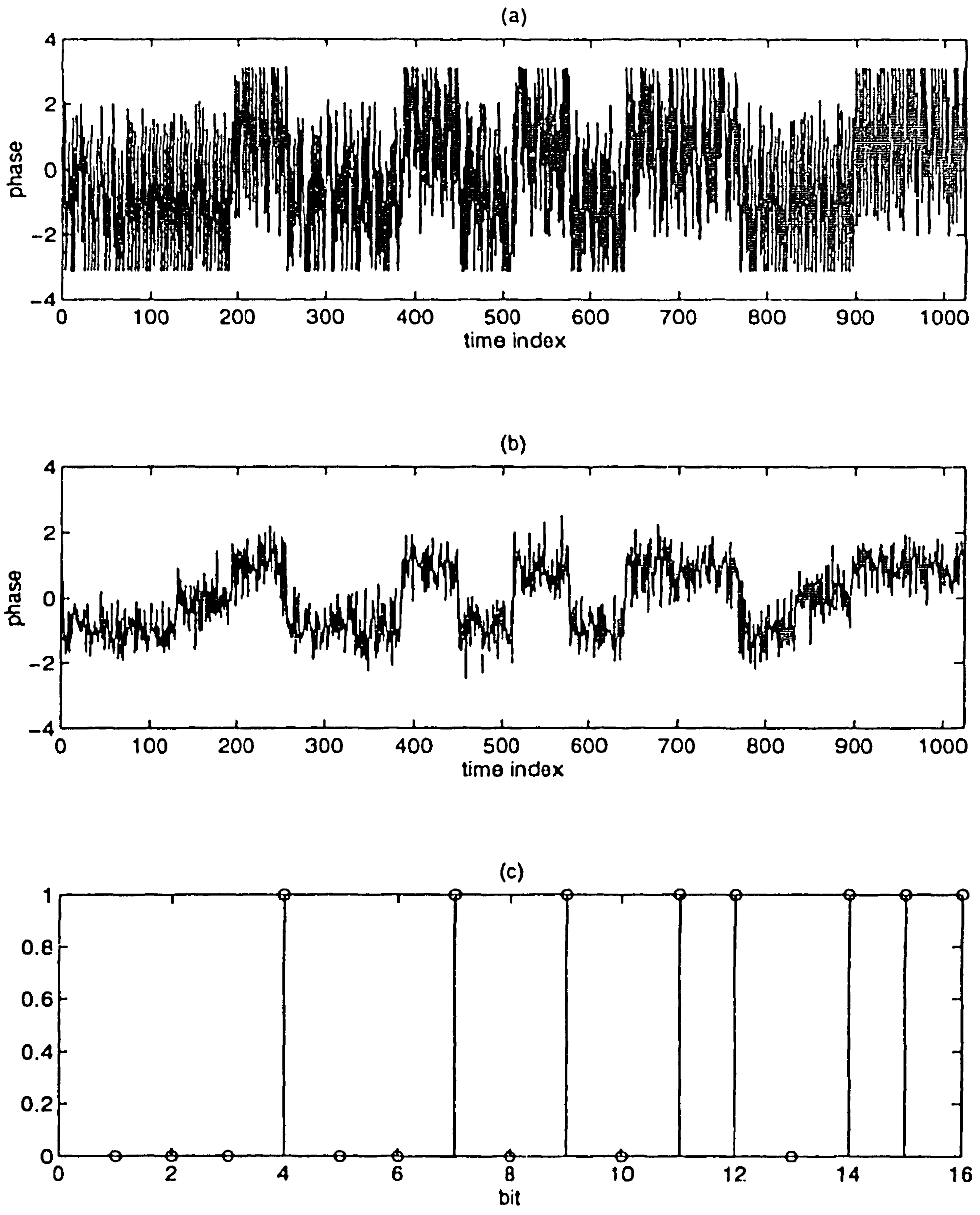
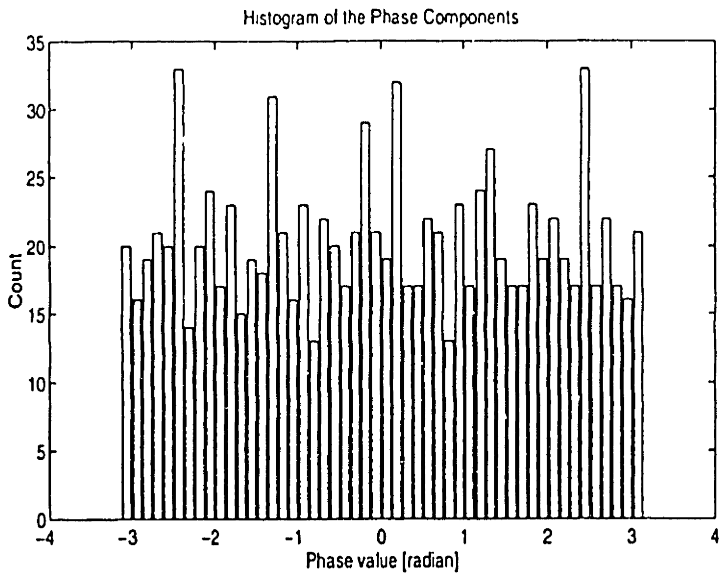


Figure A: (a) detected phase matrix from one segment (b) averaged phase matrices from ten segments (c) detected binary code (16-bits)



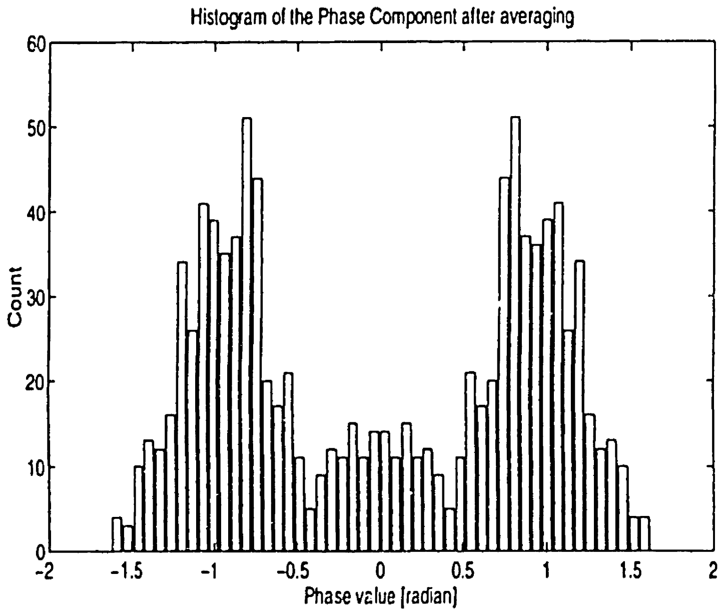
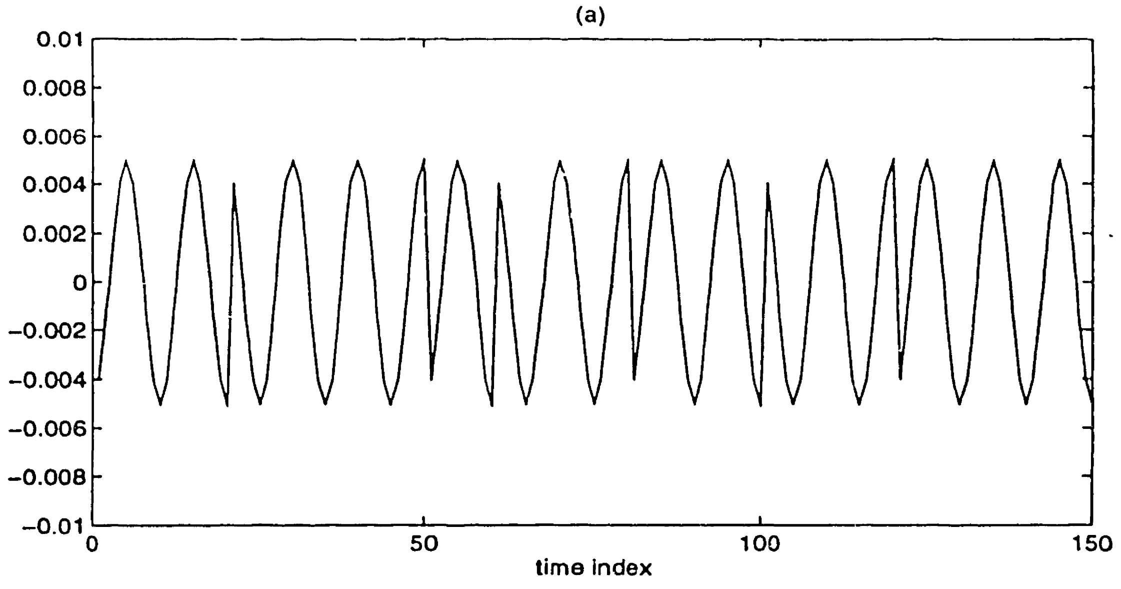


Figure B: Histogram of the phase of the original segment and the encoded segment after averaging



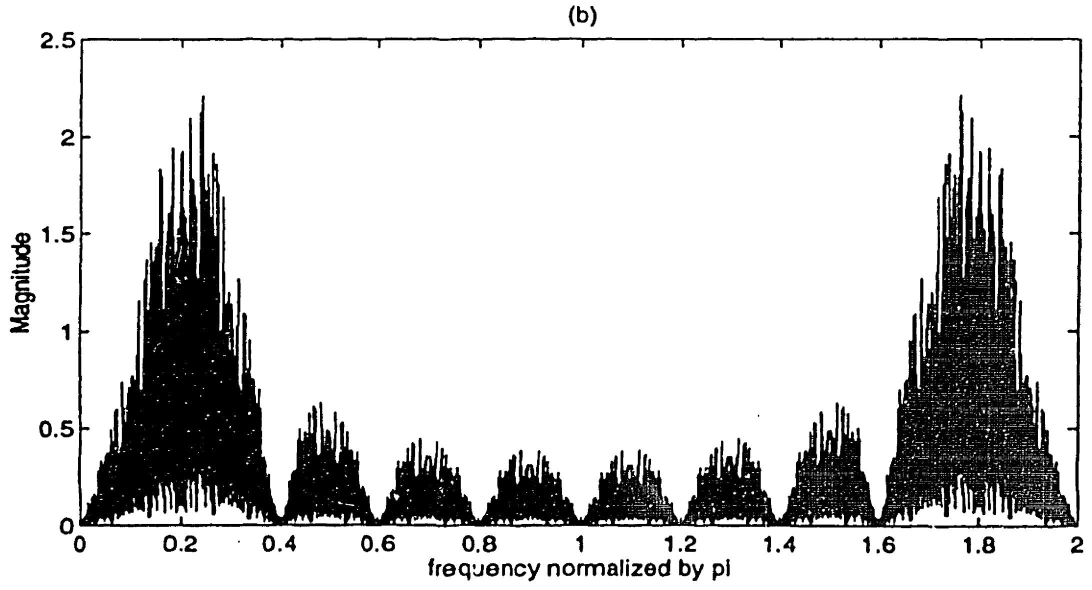


Figure C: (a) Sample segment of synthesized DSSS Code. (b) Power spectrum of the synthesized DSSS Code

References

[1] W. Bender, D. Gruhl and N. Morimoto, "Data Hiding Techniques", *Procedings, SPIE* vol.2420-40. Feb. 1995.

[2] E. Adelson, "Digital signal encoding and decoding apparatus", U. S. Patent 4,939,515, 1990.

[3] W. Bender, "Data Hiding", News in the Future, MIT Media Laboratory, (unpublished lecture notes), May, 1994.

[4] R. C. Dixon, *Spread Spectrum Systems*, John Wiley & Sons, Inc., 1976.

[5] FTP://sumex-aim.stanford.edu/stego.

[6] P. Sweeney, *Error Control Coding (An Introduction)*, Prentice

-Hall International Ltd, 1991.

[7] R. J. McAulay and T. F. Quatieri, "Speech Processing Based on a Sinusoidal Model", Lincoln Laboratory, MIT, 1994.

[8] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Inc., 1975.

[9] S. K. Marvin, *Spread Spectrum Handbook*, McGraw-Hill, Inc., 1985.

[10] P. W. Baier et al., "Spread-Spectrum Receiver Synchronization Scheme Using a SAW-Tapped Delay Line", *IEEE Trans. Commun.*, vol.COM-30, pp.1037-1047, May 1982.

[11] D. Gruhl, *libDSP library*, Version 1.1.0, Apr. 1995.

[12] A. V. Oppenheim and R. W. Schafer, *Discrete Signal Processing*, Prentice-Hall, Inc., 1975.

[13] S. H. Nawab and T. F. Quatieri, *Short-Time Fourier Transform*, (unpublished lecture notes), Sep., 1994.

[14] A. V. Oppenheim, A. S. Willsky and I. T. Young, *Signals and Systems*, Prentice-Hall, Inc., 1983.

[15] K. Matsui and K. Tanaka, "Video-Steganography: How to Secretly Embed a Signature in a Picture", *IMA Intellectual Property Project Proceedings*, 1:1, January, 1994.

[16] D. L. Hecht, "Embedded Data Glyph Technology for Hardcopy Digital Documents", *SPIE* vol.2171-341. Feb. 1995.

[17] S. Walton, "Image Authentication for a Slippery New Age", *Dr. Dobb's Journal*, April 1995. 311997-3-8533-1

[18] J. D. Johnson, "Transform Coding of Audio Signals Using Perceptual Noise Criteria", *IEEE Journal Selected Areas in Communications*, pp. 314-323, Feb. 1988.

[19] N. S. Jayant *et al*, "Signal Compression Based on Models of Human Perception", *Proceedings of the IEEE*, 1993.

[20] T. F. Quatieri *et al*, "Time-scale Modification with Temporal Envelope Invariance", *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Oct. 1993.

[21] J. R. Garrett and J. S. Alen, "Toward a Copyright Management System for Digital Libraries", Copyright Clearance Center, Inc., 1991.