DISTORTION TOLERANT SOURCE CODE USING THE VITERBI ALGORITHM
DISTORTION TOLERANT SOURCE CODE USING THE VITERBI ALGORITHM

An Undergraduate Honors College Thesis

in the

Department of Electrical Engineering
College of Engineering
University of Arkansas
Fayetteville, AR

by

Christopher Chad Hankins
ABSTRACT

Convolutional codes are used in digital communication systems in order to protect information from distortion and increase system reliability. One of the most efficient methods for decoding convolutional codes is based on the Viterbi Algorithm. Considering the fundamental similarities between these channel codes and source codes, it is logical to postulate that the Viterbi Algorithm may also provide a basis for the implementation of an efficient lossy source code. In this thesis, the Viterbi Algorithm is used to compress digital data from a symmetric Bernoulli source. The Viterbi source code is simulated and shown to produce results which are reasonably close to the theoretical limit provided by rate-distortion theory. The results presented in this thesis highlight the correlation that exists between channel codes and lossy source codes.
This thesis is approved.

Thesis Advisor:

[Signature]

Thesis Committee:

[Name]

[Name]

[Name]
# TABLE OF CONTENTS

## CHAPTER 1: INTRODUCTION
  1.1 BACKGROUND INFORMATION 1  
  1.2 MOTIVATION 4  
  1.3 EXISTING WORK 5  
  1.4 DEVELOPMENT OF LOSSY SOURCE CODE USING VITERBI ALGORITHM 8  
  1.5 ORGANIZATION OF THESIS 9

## CHAPTER 2: DIGITAL COMMUNICATION THEORY 10
  2.1 CHANNEL CHARACTERISTICS 10  
  2.2 MODULATION TECHNIQUES 14  
  2.3 SIGNAL DETECTION AND ERROR PROBABILITY 21  
  2.4 CHANNEL CODE 23  
  2.5 SOURCE CODE 28

## CHAPTER 3: CHANNEL CODE USING VITERBI ALGORITHM 30
  3.1 HISTORY 30  
  3.2 METHODOLOGY 32  
  3.3 CHANNEL CODE SIMULATION 40

## CHAPTER 4: LOSSY SOURCE CODE USING VITERBI ALGORITHM 42
  4.1 MOTIVATION 42  
  4.2 METHODOLOGY 42  
  4.3 SOURCE CODE SIMULATION 46

## CHAPTER 5: CONCLUSION 48
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Example Analog Signal</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Example Digital Signal</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>FBD of Communication System with VA-Based Source Code</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Normalized PDF of Gaussian Noise</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>Nonreturn-to-zero PCM Waveforms</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>Return-to-zero PCM Waveforms</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>Phase Encoded PCM Waveforms</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>Multilevel Binary PCM Waveform</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>Shift Register Representation of Convolutional Encoder</td>
<td>26</td>
</tr>
<tr>
<td>10</td>
<td>State Transition Representation of Convolutional Encoder</td>
<td>26</td>
</tr>
<tr>
<td>11</td>
<td>Input-Output Table Representation of Convolutional Encoder</td>
<td>27</td>
</tr>
<tr>
<td>12</td>
<td>Trellis Representation of Convolutional Encoder</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>First Transition</td>
<td>36</td>
</tr>
<tr>
<td>14</td>
<td>Second Transition</td>
<td>37</td>
</tr>
<tr>
<td>15</td>
<td>Third Transition</td>
<td>38</td>
</tr>
<tr>
<td>16</td>
<td>Fourth Transition</td>
<td>38</td>
</tr>
<tr>
<td>17</td>
<td>Survival Path</td>
<td>39</td>
</tr>
<tr>
<td>18</td>
<td>Performance of Viterbi Decoding</td>
<td>41</td>
</tr>
<tr>
<td>19</td>
<td>Performance of Channel Codes with Various Code Rates</td>
<td>41</td>
</tr>
<tr>
<td>20</td>
<td>Rate-Distortion Function</td>
<td>45</td>
</tr>
<tr>
<td>21</td>
<td>System Model for Source Code Simulation</td>
<td>46</td>
</tr>
<tr>
<td>22</td>
<td>Viterbi Source Code Performance</td>
<td>47</td>
</tr>
</tbody>
</table>
CHAPTER 1: INTRODUCTION

1.1 BACKGROUND INFORMATION
Communication systems are categorized as either analog or digital in nature, depending on the manner in which information is represented and interpreted within the given system. In an analog system, the transmitted signal, or message, can take on infinitely many values. Examples of analog signals include the human voice and the temperature reading of a glass thermometer. Conversely, digital systems involve the transmission and reception of signals which are limited to a finite number of possible values. Morse code and smoke signaling both transmit information using digital signals. Example signals for each system are shown below.

![Figure 1: Example Analog Signal](image-url)
As a result of this fundamental difference, analog and digital systems do not always exhibit the same level of performance. Furthermore, the relative simplicity of digital signals allows for a greater level of data manipulation and system optimization within digital systems than would ever be possible using analog signals. As a result, digital communication provides a number of advantages over analog communication and is used extensively in technologies such as telecommunications, computer networking, and satellite communication. Due to the demand for such systems to transmit data in the most efficient, secure manner possible, researchers continue to search for ways to optimize the performance of modern digital systems. A number of techniques have been developed that improve various characteristics of system performance by means of manipulating the original signal. Coding theory is one of the most popular areas of study within this field and is the focus of this paper.

Basically, coding involves the transformation of information within a communication system from one form to another, more desirable form, using a strict set of rules and procedures. These rules and procedures depend upon which specific coding method is utilized and can vary greatly.
from code to code. However, all codes fall into one of two general categories, depending upon the code’s function within the overall system. These two subgroups, channel codes and source codes, are discussed below.

Channel coding is most commonly used to protect transmitted data from the negative effects of channel noise. Using this method, redundancy is added to the original signal before it is transmitted and is then removed upon signal reception. This structured redundancy reduces the distortion caused by the channel and results in a lower overall error rate for the system. However, this increase in transmission reliability comes at the expense of bandwidth efficiency. Since channel codes increase the total number of bits transmitted, systems in which this method is utilized require more bandwidth to transmit information bits. Therefore, it is vital that the designer of a given communication system understand the performance trade-offs associated with channel codes.

Source codes are used to increase communication efficiency in situations where it is not necessary or practical to transmit the original signal in its entirety. These codes operate in a manner exactly opposite to that of channel codes, eliminating the redundancy found in a given signal. Some common applications of source coding involve analog-to-digital conversions or the compression of digital information for the purpose of increasing data usage efficiency. Source codes not only provide a method for representing analog signals in a digital system, but also increase bandwidth efficiency by decreasing the total number of bits that are transmitted. However, the major risk associated with source codes is that some of the original information may not be recovered correctly, leading to an increase in the overall error rate. The ability of a
particular source code to perfectly recover the original signal, or lack of this ability, dictates whether the code is considered lossless or lossy. Lossy source codes and lossless source codes will be discussed in more detail in later chapters.

One of the most popular coding techniques in modern communication systems is a channel code method known as convolutional coding. In convolutional codes, data is protected from distortion by an encoding method in which the output of the encoder depends not only on the current input, but also on previous inputs. As with any channel code, convolutional codes involve a decoding stage in which the received, coded signal is converted back to its original form. Several techniques exist which provide the functionality necessary to implement such a decoder. One such method, known as Viterbi decoding, uses the Viterbi Algorithm to achieve impressive levels of reliability and signal recovery in communication systems that use convolutional codes [1].

1.2 MOTIVATION

Source codes are extremely powerful tools and can greatly improve the performance of many communication systems. By removing redundancy from a given data source and transmitting the fewest bits possible, source coding leads to an increase in the bandwidth efficiency of the associated system. However, there is a fine line between only removing the unnecessary redundancy in a particular signal and over-compressing the data to the point that the original signal cannot be recovered. For this reason, it is crucial that the encoding and decoding methodologies of a given source code result in sufficiently low bit error rates for the system while operating under various conditions. These conditions include the noise characteristics and
The capacity of a given transmission channel, commonly referred to as the Shannon Limit, describes the maximum rate at which data can be transmitted error-free through a particular channel. This equation for channel capacity, developed by renowned mathematician and electronics engineer Claude Shannon, uses the bandwidth and signal-to-noise ratio for a given channel to determine its capacity [2]. If any system operates at a data rate greater than this maximum value, error will be unavoidable and system reliability will suffer to some degree. Clearly, the Shannon Limit serves a restrictive purpose in communication system design. However, this concept also plays another role that may be less obvious at first glance. By clearly identifying the highest possible level of performance for a particular channel, Shannon’s equation fosters advancement in the field of digital communication, specifically coding theory, by providing the target on which researchers can set their sights. As is the case for most research in this field, the motivation for the work presented in this paper is the desire to develop a coding method capable of achieving maximum system performance as defined by the Shannon Limit.

1.3 EXISTING WORK
Proof of the extensive efforts devoted to research in the field of coding theory can be seen in the wide variety of codes in existence today, each operating in the manner dictated by the unique set of protocols associated with that particular code. Some of these codes are fairly generic, finding use in a broad range of situations, while others are tailored for specific applications. However, despite the diversity of the various coding methods, every code can be classified as either a source code or channel code. A few of the most prevalent codes in modern systems are
discussed below. The technical operation of these codes will be discussed more thoroughly in Chapter II.

1.3.1 LINEAR BLOCK CODE (LBC)

Linear Block Code is a channel coding technique used to convert information vectors of length $k$ bits to codeword vectors of length $n$ bits for transmission. This transformation is achieved by multiplying each of the individual information vectors by the matrix $G$, known as the generator matrix. The number of redundancy bits in each resulting codeword vector is equal to the difference ($n - k$), where the first $k$ bits of each codeword are the same as the information vector. Thus the data rate for LBC depends upon the ratio of $k$ to $n$ and is given by $r = k / n$.

A benefit of such systems is that Linear Block Codes use error detection to increase transmission reliability. Upon reception of the encoded signal $y$, a parity-check matrix $H$ is used to detect any error which may have occurred during signal transmission. Specifically, $y$ is multiplied by the parity-check matrix $H$, and if the result is nonzero, an error has occurred. In certain situations in which error has been detected, syndrome detection may be used to actually correct the error at the receiver. The limitations of error detection and error correction functionalities for a given LBC depend upon its specific system design and will be discussed in the following section.

1.3.2 CONVOLUTIONAL CODE (CC)

Another very popular channel code is Convolutional Code. Compared to LBC, Convolutional Codes exhibit better power efficiencies as well as superior bandwidth efficiencies. However, Convolutional Codes are much more complex than LBC, making the design and implementation
of such systems more difficult. Most of this increased complexity is seen in the decoding process required for Convolutional Codes.

Convolutional encoders produce an $n$ bit codeword that is based not only on the current input $k$, but also on some number of past inputs. This characteristic of memory is one of the main differences between Convolutional Codes and many other channel coding methods. Convolutional encoders may be represented by one of four different models including the shift register, input-output table, state transition, and trellis representations. The shift register model is the most widely used by hardware implementation for encoders whereas the trellis representation provides the most advantageous characteristics for the decoding process.

As mentioned previously, the crux of Convolutional Codes is the ability to implement a reliable decoder capable of recovering the original signal with minimal error. The Viterbi Algorithm has been used for decades in the development of such decoders and has proven to offer an efficient and robust solution to the complex problem of signal decoding within Convolutional Codes. The great success of the Viterbi Algorithm as a convolutional encoder is one of the motivating factors for the work presented in this paper.

1.3.3 LINEAR PREDICTIVE CODE (LPC)

In contrast to the previous two methods discussed, Linear Predictive Code is a form of source code. As such, it is used in applications which call for a reduction in the redundancy of an information source. LPC is especially useful in the field of speech coding. Instead of transmitting the entire signal produced by the source, Linear Predictive Codes transmit only the
key parameters of the signal. Upon reception, the signal is reproduced using *a priori* knowledge of speech characteristics. Linear Predictive Codes provide very low data rates and are less robust than many other source codes, but also lead to high bandwidth efficiency. These type codes are used heavily in modern cellular systems in which minimizing user bandwidth is crucial.

1.4 DEVELOPMENT OF LOSSY SOURCE CODE USING VITERBI ALGORITHM

The purpose of this paper is to discuss the development and performance of a lossy source code based on the Viterbi Algorithm. In contrast to the Viterbi Algorithm's common role as a decoder for convolutional codes, this paper promotes its potential use as an encoder. The figure below depicts a very basic system in which this method is used.

![Diagram of Communication System with VA-Based Source Code](image)

*Figure 3: FBD of Communication System with VA-Based Source Code*

To demonstrate the performance of a VA-based source code, a communication system model was designed and simulated. The results presented in this paper were obtained through the
successful completion of several individual tasks. The methodology used to obtain the results discussed in this paper is outlined below.

1. Development of code blocks for Viterbi Algorithm decoder and convolutional encoder for a system with any given trellis structure.
2. Simulation of communication system using convolutional coding and binary signaling.
3. Design of communication system in which existing Viterbi decoder block and convolutional encoder block are interchanged, thus implementing proposed source code.
4. Simulation and analysis of VA-based source code.

1.5 ORGANIZATION OF THESIS

To present all relevant information in the most efficient manner possible, the remainder of this paper is organized as follows: Chapter II discusses some of the fundamental theories and methods associated with digital communication theory including various modulation schemes and methods of coding. Special attention will be given to those concepts which are most closely tied to the work discussed within this paper. Chapter III focuses on the Viterbi Algorithm and the role it plays in the decoding of convolutional codes. Also included in this chapter will be a brief explanation of the techniques used to develop and simulate a communication system model in which Viterbi decoding is utilized. Chapter IV will be centered on the development and simulation of a lossy source code that uses the Viterbi Algorithm for signal encoding, the method proposed in this thesis. Finally, in Chapter V, the paper closes with a discussion of all relevant findings as well as comments on the possible implications of the results obtained.
CHAPTER 2: DIGITAL COMMUNICATION THEORY

2.1 CHANNEL CHARACTERISTICS

2.1.1 NOISE

In an ideal communication system, the signal arriving at the receiver would exactly match the signal that was transmitted. There would be no signal distortion and the transmitted information would be recovered perfectly without any error. Creating such a system is not possible, however, due to the presence of thermal noise in every communication channel. Also known as Johnson noise, this effect is caused by the thermal motion of electrons in the dissipative components within a given communication system. Thermal noise often masks the original signal and leads to error in signal detection at the receiver. Since it is present in all communication systems, it is important to understand the characteristics of thermal noise as well as the techniques available for reducing its detrimental effects on system performance.

Thermal noise can be modeled as a zero-mean random process with Gaussian distribution [3]. This means that the value of the noise function $n(t)$ for any given time $t$ can be characterized by the following probability density function:

$$p(n) = \frac{1}{\sigma_0 \sqrt{2\pi}} \exp \left[ -\frac{1}{2} \left( \frac{n}{\sigma_0} \right)^2 \right].$$

(1)

In order to normalize this probability density function to for analytical purposes, it may be assumed that the standard deviation $\sigma$ is equal to 1. The resulting function, known as the standardized Gaussian density function, is plotted in the figure below.
If we let $a$ represent the transmitted symbol and we let $n$ represent the thermal noise with the PDF shown above, the received signal can be written as the sum of these two components.

$$z = a + n$$  \hspace{1cm} (2)$$

Furthermore, according to [3], the probability density function for the received signal $z$ can be written as

$$p(z) = \frac{1}{\sigma_0\sqrt{2\pi}} \exp\left[ -\frac{1}{2}\left(\frac{z-a}{\sigma_0}\right)^2 \right].$$  \hspace{1cm} (3)$$

These equations are used extensively in the analysis of various communication systems. They aide in the calculation of expected error rates for various communication methods and are used by system designers in the pursuit of optimal communication performance.
One of the major figures of merit in analog communication is the signal-to-noise ratio, SNR. In digital communication, however, it is more helpful to use the normalized version of this relationship in system analysis. The normalized SNR is shown below as the energy of one bit, $E_b$, divided by the noise power spectral density, $N_0$ [3].

\[
\frac{E_b}{N_0} = \frac{S}{N/W} = \frac{S/R_b}{N/W}
\]  

(4)

Here, the energy per bit is shown to be equivalent to signal power multiplied by the period of one bit and the noise power spectral density is represented by noise power divided by bandwidth. The subscript $b$ on the data rate variable $R_b$ is somewhat redundant because, in digital communications, data rate values are almost always presented using the unit of bits per second. Therefore, the equation above can be rearranged to highlight the fact that $E_b/N_0$ is simply a normalized version of SNR. The resulting equation is given as

\[
\frac{E_b}{N_0} = \frac{S}{N} \left(\frac{W}{R}\right).
\]  

(5)

A very common tool in the analysis of digital systems is a plot of bit-error probability, $P_b$, as a function of $E_b/N_0$. The derivation of bit-error probabilities of various systems are presented later in this chapter.

### 2.1.2 BANDWIDTH
Another performance-limiting characteristic associated with digital communication is channel bandwidth. With respect to transmission channels, bandwidth refers to the range of frequencies that can be used to transmit information through the channel without resulting in distortion levels greater than the acceptable value. The bandwidth for a given channel could be limited by the physical properties of the medium used for signal transmission, or it could be intentionally limited through the introduction of a filter.

In the design of communication systems, bandwidth efficiency is one of the main factors considered. A high bandwidth efficiency allows for a system to achieve error-free transmission at a higher data rate than would be permitted in a system with low bandwidth efficiency. However, there are tradeoffs associated with bandwidth efficiency that must be balanced by the system designer according to the specific performance requirements for the system. For example, channel codes decrease the overall error rate for a given system by adding redundancy to the signal before transmission. However, the addition of a channel code increases the total number of bits transmitted, thus increasing the bandwidth needed for transmission and reducing system efficiency in this respect.

2.1.3 CAPACITY

The term channel capacity describes the maximum data rate at which error-free transmission can occur through a given channel. This value is dependent upon the two channel characteristics previously discussed, noise and bandwidth. In 1949, Claude E. Shannon proved that this maximum data rate, for a channel in the presence of white thermal noise, is given as
\[ C = W \log_2 \left(1 + \frac{S}{N}\right) \]  \hspace{1cm} (6)

where \( W \) is the channel bandwidth and \( S \) and \( N \) are the average values of signal power and noise power respectively [2]. Therefore, error-free transmission is not possible for any data rate higher than \( C \), regardless of the efficiency of the encoding method used.

2.2 MODULATION TECHNIQUES

Modulation involves the transformation of a given signal into waveforms that are compatible with the channel through which transmission occurs. Two types of modulation are discussed below and examples of each are presented.

2.2.1 BASEBAND MODULATION

In baseband modulation, digital information is transformed into waveforms consisting of various pulse shapes. These waveforms usually exhibit relatively low frequency components. Some common examples of such modulation techniques are discussed below.

2.2.1.1 PULSE CODE MODULATION (PCM)

Pulse Code Modulation involves is the process by which a quantized PAM signal is converted to a string of binary digits and then modulated to create a signal waveform capable of carrying information through the channel. The resulting PCM waveform will take on one of four possible forms, depending upon the specific procedure used to encode the digital information. The four types of PCM waveforms include nonreturn-to-zero (NRZ), return-to-zero (RTZ), phase encoded, and multilevel binary waveforms. The encoding methodologies for each type of waveform are discussed below [3].
Nonreturn-to-zero waveforms are the most commonly used of the four. In NRZ waveforms, a binary one is represented by one voltage level and a binary zero is represented by another voltage level. Waveforms of this type can be further classified as either NRZ-L, NRZ-M, or NRZ-S waveforms. In NRZ-L waveforms, where the L stands for "level", the voltage level of the signal changes every time the data changes from a one to a zero or from a zero to a one. For NRZ-M waveforms, M meaning "mark", the voltage level only changes for each interval in which the data bit is a one. Thus, in this type waveform, a binary zero is represented by no change in the voltage level. Finally, in waveforms of the type NRZ-S, where the S stands for "space", the signal voltage changes when the data to be transmitted is a binary zero but remains constant during the transmission of a one. The figure below shows waveforms of this form for a given binary data sequence.

![Figure 5: Nonreturn-to-zero PCM Waveforms](image)

Return-to-zero waveforms can also be divided into even more specific subgroups. These include unipolar-RZ waveforms, bipolar-RZ waveforms, and those waveforms classified as AMI-RZ, or "alternate mark inversion"-RZ waveforms. In unipolar-RZ waveforms, a binary one is
represented by a half-bit-wide pulse and a zero is represented by the lack of such a pulse. Therefore, the signal alternates between a voltage level equal to the pulse amplitude and zero, hence the name return-to-zero. In bipolar-RZ waveforms, as in the unipolar case, a binary one is represented by a half-bit-wide pulse. However, a zero is represented by a half-bit-wide pulse with a voltage level opposite to that of the pulse which occurs for a one. Therefore, a pulse occurs for every bit in the given data sequence when using bipolar-RZ modulation. Finally, for AMI-RZ waveforms, binary ones are represented by alternating equal-amplitude pulses. Zeros in the data sequence are represented by the lack of a pulse. Examples of each RZ waveform are shown in the figure below.

![Figure 6: Return-to-zero PCM Waveforms](image)

Phase encoded waveforms carry binary information using the phase of individual pulses within the waveform. Variations of this type modulation include bi-phase-level, bi-phase-mark, bi-phase-space, and delay modulation. Bi-phase-level modulation, also known as Manchester coding, is fairly simple. Using this scheme, ones are represented by a half-bit-wide pulse occurring during the first half of a bit interval while zeros are represented by a half-bit-wide pulse of equal amplitude occurring during the second half of the bit interval. In bi-phase-mark
modulation, there is a level change at the beginning of every bit interval. For an interval in which a one is to be transmitted, the level changes again halfway through the bit interval. If a zero is to be transmitted during an interval, there is no level change. Bi-phase-space modulation is simply the opposite of the bi-phase-mark scheme. That is, intervals in which a one is transmitted do not involve a level change but those containing a zero display a level change halfway through the bit interval. Delay modulation, commonly referred to as Miller coding, is a bit more complex. Using this scheme, ones are represented by a transition at the middle of the bit interval. Intervals in which a zero is to be transmitted lack this transition. However, if a zero is followed by another zero, a transition occurs at the end of the interval containing the first zero. The figure below shows how these waveforms would look for a sample sequence of binary symbols.

![Phase Encoded PCM Waveforms](image)

*Figure 7: Phase Encoded PCM Waveforms*

The last type of PCM waveform is the multilevel binary waveform, and includes dicode-NRZ and dicode-RZ waveforms. In each of these schemes, there exist three distinct voltage levels. These levels include two equal-amplitude voltages of opposite polarity and a zero voltage level.
For dicode-NRZ waveforms, a transition in the data sequence from one to zero or zero to one results in a polarity shift of the signal. If a transition does not occur, a voltage of zero is sent. Dicode-RZ is basically the same as dicode-NRZ except for the fact that the pulses in this type waveform last only half of the given bit interval and then return to the zero level. Examples of these multilevel binary waveforms are shown below.

![Figure 8: Multilevel Binary PCM Waveform](image)

2.2.1.2 BIPOLAR SIGNALING

Bipolar signaling is a method for transmitting binary information using antipodal waveforms. The term antipodal refers to a pair of waveforms that are mirror images of one another. Thus the transmitted signal of such a system takes the form of the NRZ-L PCM waveform discussed in the previous section. Upon reception at the system receiver, this waveform has been distorted to some extent by the effects of channel noise. In order to minimize the probability that this additive noise causes error in signal detection, certain parameters must be taken into account during system design. The various techniques used for signal detection and recovery in systems using bipolar signaling will be discussed later in this chapter.
2.2.2 BANDPASS MODULATION

Often times, the low frequency signals produced by baseband modulation are not of the most advantageous form in terms of transmission efficiency. In order to achieve more efficient communication, bandpass modulation is used to transform these low frequency baseband signals into higher frequency signals using a sinusoid signal known as the carrier wave. There are many techniques available for achieving bandpass modulation, each of which transmits information by manipulating the characteristics of the carrier wave in a particular way [3].

The general form of a radio frequency carrier signal is given by

\[ s(t) = A(t)\cos(2\pi f_c t + \varphi(t)). \]  \hspace{1cm} (7)

2.2.2.1 PHASE SHIFT KEYING (PSK)

In Phase Shift Keying, each digital symbol is represented by a particular signal phase. The general form of a PSK modulated carrier signal is given by

\[ s_i(t) = \sqrt{\frac{2E}{T}} \cos(\omega_0 t + \varphi_i(t)) \quad 0 \leq t \leq T \quad i = 1, ..., M, \]  \hspace{1cm} (8)

where the signal phase is given by

\[ \varphi_i(t) = \frac{2\pi i}{M}. \] \hspace{1cm} (9)

2.2.2.2 FREQUENCY SHIFT KEYING (FSK)
In Frequency Shift Keying, each digital symbol is represented by a particular signal frequency. The general form of a FSK modulated carrier signal is given by

\[ s_i(t) = \sqrt{\frac{2E}{T}} \cos[\omega_i t + \varphi] \quad 0 \leq t \leq T, \quad i = 1, ..., M. \] (10)

### 2.2.2.3 AMPLITUDE SHIFT KEYING (ASK)

In Amplitude Shift Keying, each digital symbol is represented by a particular signal amplitude. The general form of an ASK modulated carrier signal is given by

\[ s_i(t) = \sqrt{\frac{2E_i(t)}{T}} \cos[\omega_0 t + \varphi] \quad 0 \leq t \leq T, \quad i = 1, ..., M. \] (11)

### 2.2.2.4 AMPLITUDE PHASE KEYING (APK)

In Amplitude Phase Keying, each digital symbol is represented by a unique phase-amplitude pair. The general form of an APK modulated carrier signal is given by

\[ s_i(t) = \sqrt{\frac{2E_i(t)}{T}} \cos[\omega_0 t + \varphi_i(t)] \quad 0 \leq t \leq T, \quad i = 1, ..., M. \] (12)

### 2.2.3 DEMODULATION

Clearly, since modulation transforms the original binary information into waveforms, it is necessary that communication systems also implement a demodulating function. The demodulator is responsible for reversing the changes made by the modulator and returning the data to its original form. Once this has occurred, the recovered binary information can be passed
to the next stage of the communication system. Often times, demodulation is followed by a decoding process in which the source information is determined based on the encoded signal and the method of encoding that was used prior to transmission.

The actual design and implementation of the demodulator for a given system depends on several factors. Clearly, demodulator design for a given system will depend heavily on the modulation method used therein to convert the original binary symbols to information-carrying waveforms. Also, demodulator structure depends on whether a communication system uses coherent or incoherent detection. In coherent detection, the receiver is aware of the carrier signal's phase and uses this information to improve signal detection accuracy whereas incoherent detection requires the receiver to operate without knowledge of carrier phase. This operational difference results in higher error rates for systems in which incoherent detection is used.

2.3 SIGNAL DETECTION AND ERROR PROBABILITY

After the received signal has been demodulated, the resulting baseband pulse must be evaluated in order to determine which binary symbol was transmitted. This step involves a decision process in which test samples of the baseband pulse are compared to a design-specific threshold value to determine which channel symbol was most likely transmitted. Clearly the value of this threshold plays a large role in determining the performance of communication systems. The process for determining the minimum-error-producing value of the decision threshold, \( \gamma_0 \), is shown below. Detection is assumed to occur in the presence of Gaussian noise.

Let \( z(t) \) represent the baseband pulse resulting from the demodulation of the received signal in a given system. This signal can be written as
\[ z(t) = a_i(t) + n_0(t) \]  

(13)

where \( a_i(t) \) is the signal component of \( z(t) \) and \( n_0(t) \) is the noise component. After sampling occurs at time \( T \), the test statistic is produced in the form given by

\[ z(T) = a_i(T) + n_0(T). \]  

(14)

Now, to determine which symbol was transmitted, the test sample is compared to the threshold in the manner shown below.

\[
\text{Transmitted Symbol} = \begin{cases} 
0 & \text{if } z(t) < \gamma_0 \\
1 & \text{if } z(t) > \gamma_0.
\end{cases}
\]  

(15)

Clearly, error occurs when the detected symbol does not match the transmitted symbol. This occurs when a one is transmitted and a zero is detected, or when a zero is transmitted and a one is detected. Therefore, the goal in selecting a value for \( \gamma_0 \) is to minimize the probability that either of these situations occur. It has been shown that for digital communication in which the probability of a zero being transmitted is equal to that of a one being transmitted, this optimal value is obtained by calculating the threshold in the following way:

\[ \gamma_0 = \frac{a_1 + a_2}{2}. \]  

(16)
where $a_1$ and $a_2$ represent the signal component of $z(T)$ for the transmission of a binary one or a binary zero respectively [4]. After choosing this value, it is possible to estimate the overall error rate for a given system.

Using the probability density function shown in Equation (iii), the probabilities are determined for each of the error-causing situations mentioned above. Here, $s_1$ represents the transmitted signal for a binary one and $s_2$ represents the signal for a binary zero.

$$P(E|s_1) = \int_{-\infty}^{\gamma_0} p(z|s_1)dz$$  \hspace{1cm} (17)$$

$$P(E|s_2) = \int_{\gamma_0}^{\infty} p(z|s_2)dz$$  \hspace{1cm} (18)$$

Using the equation for conditional probability, we calculate the total bit error probability for a given digital system in the following manner [4]:

$$P(E) = P(s_1)P(E|s_1) + P(s_2)(P(E|s_2).$$  \hspace{1cm} (19)$$

In the equation above, $P(s_1)$ and $P(s_2)$ represent the probability of transmission for a binary one and zero respectively. For digital communication, these values are both equal to one half.

2.4 CHANNEL CODE

2.4.1 ERROR CONTROL
Channel codes are used to protect data from the deleterious effects of channel noise. This is accomplished through the addition of redundancy to the signal before transmission. The protocols which govern this process in a given system will vary from one type of code to another. In all channel codes, however, these added bits are used for the purpose of reducing error in the decoding process, thus improving system reliability. The two basic methods for implementing error control in digital communication systems are discussed below.

2.4.1.1 ERROR DETECTION AND RETRANSMISSION

The first method, error detection and retransmission, involves several steps: First, an error in the received signal is detected by the receiver. Next, the receiver notifies the transmitter that the error has occurred. Finally, the transmitter retransmits the specified information. A two-way link between transmitter and receiver is required for this type of error control. Automatic Retransmission Queries (ARQ), the name given to error control methods of this type, are further divided into three groups: stop-and-wait ARQ, continuous ARQ with pull back, and continuous ARQ with selective repeat. Stop-and-wait ARQ, as suggested by its name, requires that the transmitter receive acknowledgment of successful message reception before transmitting the next packet. In contrast, both of the other ARQ methods involve constant transmission for both the transmitter and receiver. However, a difference between these methods can be found in the retransmission step of ARQ. For continuous ARQ with pull back, the transmitter sends the problem packet and all of the subsequent messages, whereas continuous ARQ methods with selective repeat only require the retransmission of the problem packet. Not surprisingly, the latter is much more complicated.
2.4.1.2 FORWARD ERROR CORRECTION (FEC)

In the case of forward error detection methods, only a one-way link is required between transmitter and receiver. FEC is implemented with either a block code or a convolutional code. The receiver detects an error and corrects the error to the best of its ability. This ability is provided through the use of structured sequences in the encoding process and the receiver's knowledge thereof. Compared to ARQ methods, FEC techniques provide somewhat lower levels of error correction but have the benefit of being less complex.

2.4.2 CONVOLUTIONAL CODE (CC)

Convolutional Code is one of the most widely used channel code in modern technology. As explained in the previous chapter, this method uses an encoder to transform \( k \) bit input vectors into \( n \) bit codewords based on current inputs as well as previous inputs. The number of previous inputs considered depends upon the value of \( K \) which denotes encoder memory. A convolutional encoder can be represented in a number of ways. Depending upon the particular application, one representation may be more beneficial than the others [3]. The following examples are representations of an encoder with the following parameters; \( k = 1, n = 2 \), and \( K = 3 \).

2.4.2.1 SHIFT REGISTER

The shift register representation is commonly used for hardware implementations of CC encoders. The register size is determined by the depth of encoder memory, \( K \). The \( n \) output bits are determined based by the code definition for the given encoder. This representation is shown in the figure below.
2.4.2.2 STATE TRANSITION DIAGRAM

In state transition diagram, boxes represent all of the possible encoder states. Two lines leave each box. One line leads to the state the encoder will be in if the next input is a one, and the other leads to the state the encoder will be in if the next input is a zero. Also, the lines are labeled in the manner "Current Input Value / Current Output". This method is depicted below.
2.4.2.3 INPUT-OUTPUT TABLE

This method for encoder representation uses a table to show the next state and output value for all possible combinations of current states and input values. An example is given below.

<table>
<thead>
<tr>
<th>Input (k)</th>
<th>Current State (R2,R3)</th>
<th>(R1, R2, R3) (R1–k)</th>
<th>Next State (After Shift)</th>
<th>Output (n1,n2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(00)</td>
<td>(000)</td>
<td>(00)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>0</td>
<td>(01)</td>
<td>(001)</td>
<td>(00)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>0</td>
<td>(10)</td>
<td>(010)</td>
<td>(01)</td>
<td>(1,0)</td>
</tr>
<tr>
<td>0</td>
<td>(11)</td>
<td>(011)</td>
<td>(01)</td>
<td>(0,1)</td>
</tr>
<tr>
<td>1</td>
<td>(00)</td>
<td>(100)</td>
<td>(10)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>1</td>
<td>(01)</td>
<td>(101)</td>
<td>(10)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>1</td>
<td>(10)</td>
<td>(110)</td>
<td>(11)</td>
<td>(0,1)</td>
</tr>
<tr>
<td>1</td>
<td>(11)</td>
<td>(111)</td>
<td>(11)</td>
<td>(1,0)</td>
</tr>
</tbody>
</table>

*Figure 11: Input-Output Table Representation of Convolutional Encoder*

2.4.2.4 TRELLIS

The trellis structure is the most beneficial encoder representation for decoding convolutional codes. Using this method, a matrix of nodes is created in order to visually represent state transitions and their respective outputs. Each row represents an encoder state and the number of columns is determined by the number of output symbols. The figure below shows the trellis representation for the data sequence [01101].
2.5 SOURCE CODE

Source codes are used to eliminate unnecessary redundancy in the source information of a given communication system. Analog-to-digital conversion is one of the most common tasks in which this technique is required. Source codes are also used to compress digital data for the purpose of reducing it to a more appropriate or manageable size.

2.5.1 LOSSY VS. LOSSLESS

One of the most significant characteristics used to describe source codes is whether or not the code allows for the perfect recovery of the original information. Codes that result in some level of data loss are classified as lossy source codes, and, for obvious reasons, those allowing for perfect signal recovery are called lossless. By definition, digital signals can only take on a finite number of values. As a result of this limitation and the fact that analog signals are defined on a continuous spectrum, all source codes used in the implementation of analog-to-digital conversion are lossy codes. For digital compression applications, either type of code may be used.
2.5.2 SAMPLING AND QUANTIZATION

Analog-to-digital conversion is accomplished through techniques known as sampling and quantization. During sampling, the value of a given source signal is evaluated at specified intervals. The results obtained are then individually assigned to the appropriate quantization level, each of which is represented by a unique digital word. This step allows the original signal to be represented by a digital signal. The accuracy of signal recovery for a particular sampling and quantization method depends upon sampling frequency $F_s$ and the number of quantization levels.

The minimum sampling frequency necessary for complete signal recovery depends upon that of the source signal. Commonly referred to as the Nyquist rate, this lower bound for sampling frequency is related to the source signal frequency $W$ as shown below [2].

$$F_s > 2W$$

(20)

Any sampling frequency less than this value will result in discrepancies between the original signal and the recovered information. For a given quantization method, the number of discrete levels $L$ plays a large part in determining the system's error rate. For efficiency, this value is usually chosen to be a power of two since binary values use a base of two. For any given number of quantization levels, the required number of bits per symbol, $q$, can be calculated using the equation provided in [3]:

$$q = \log_2(L).$$

(21)
One major tradeoff that must be considered in determining the value for $L$ is between accuracy and bandwidth. A greater number of discrete quantization levels allows for more accurate reproduction of the original signal, but also requires a greater number of bits per symbol, thus decreasing bandwidth efficiency.

**CHAPTER 3: CHANNEL CODE USING VITERBI ALGORITHM**

**3.1 HISTORY**

In a 1967 paper entitled "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", Andrew J. Viterbi introduced an original method for the decoding of convolutional codes [5]. At that time, Viterbi viewed his new technique as a useful learning tool, capable of increasing one's "general understanding of convolutional codes and sequential decoding through its simplicity of mechanization and analysis". In the original paper, Viterbi goes so far as to describe his algorithm as "clearly suboptimal", and deny any possible real-world application, claiming that the "algorithm is rendered impractical by the excessive storage requirements" [5]. However, it doesn't take long for Viterbi to learn that he had drastically underestimated the importance of his discovery.

One of the most influential people in helping Viterbi realize the true value of this new algorithm was an MIT professor by the name of G. David Forney, Jr. The following information is paraphrased from his account of the Viterbi Algorithm's evolution [1]. In an effort to better understand the reasons for convolutional code's superiority over block code in practical applications, Forney had started experimenting with trellis diagrams around the same time he
received an early draft of Viterbi's paper. In a time during which tree diagrams dominated the analysis of sequential codes, his approach was very much 'outside the box'. Interestingly, it was precisely this unique perspective which allowed Forney to the uncover the algorithm's hidden potential. After thoroughly analyzing Viterbi's method as it related to his own trellis structure, Forney found that "the Viterbi algorithm was an exact recursive algorithm for finding the shortest path through a trellis, and thus was actually an optimum trellis decoder" [1]. These results were published in [6] and researchers began examining the VA to determine exactly what benefits it offered.

One of the leading contributors to research surrounding the Viterbi Algorithm was Jerry Heller. Heller studied the performance of convolutional codes with relatively short constraint lengths, and published his findings in [7] and [8]. His research proved that the VA could be used to achieve coding gains as large as 6dB in convolutional codes having only 64 states, disproving Viterbi's original claim that the algorithm's storage requirements made it impractical. Soon thereafter, Heller was hired as the first full-time employee of Linkabit, a company started by Viterbi and two of his colleagues in order to obtain government funding for their research. In 1969, using grants from NASA and the Navy, Heller and his associates began building a VA prototype. Two short years later, Jacobs and Heller published a journal article discussing the 2 Mb/s, 64-state Viterbi decoder that Linkabit had successfully implemented [9]. Due to the combined efforts of Linkabit and the Jet Propulsion Laboratory in the 1970's, the VA became a staple in codes used for deep-space communication. Since then, the popularity of the VA has continued to rise, as proven by the fact that "VA decoders are currently used in about one billion cellphones" and countless other applications [1].
3.2 METHODOLOGY

3.2.1 CONVOLUTIONAL ENCODING AS MARKOV PROCESS

In an article published in 1973 and simply titled "The Viterbi Algorithm", G. David Forney, Jr. presents a concise explanation of the process through which the VA is used to implement optimum decoding in convolutional codes. He defines the Viterbi Algorithm as "a recursive optimal solution to the problem of estimating the state sequence of a discrete-time finite-state Markov process observed in memoryless noise" [6]. In this paper, the Markov process of interest is convolutional encoding and the optimal solution is obtained by determining the most likely source signal based on the encoded data and code structure. Justification for the use of the Viterbi Algorithm as a means for solving this specific problem is given below.

A Markov process is defined as one in which the future of the process, or the next state, depends only on the current state. Additionally, all Markov processes are finite-state discrete time processes. Convolutional encoding satisfies these conditions, thus is may be treated as a Markov process [6]. Furthermore, if thermal noise is assumed to be the only cause of distortion in a particular system, the last component of Forney's definition is satisfied. Therefore, the Viterbi Algorithm can be used to successfully implement a convolutional decoder in the presence of Gaussian noise. The following discussion addresses the key concepts of Viterbi decoding.

3.2.2 STATEMENT OF THE PROBLEM

The goal of virtually all coding methods is to decrease, or even eliminate the possibility of detection error. In other words, an ideal decoder would produce a signal that is identical to the
source signal. In the case of Viterbi decoding for convolutional codes, this requires the decoder to use its knowledge of the system's code structure to reverse the effects of encoding. Specifically, the decoder is responsible for identifying the source signal which most likely resulted in the encoded data sequence observed by the decoder.

In convolutional codes, the protocols followed during the decoding process depend completely upon the encoder's method of operation. Therefore, a discussion of Viterbi decoding must begin with an understanding of convolutional encoding.

### 3.2.3 ENCODER CHARACTERISTICS

Convolutional encoding transforms $k$ bit information vectors into $n$ bit output vectors. It follows then that the coding rate for a given encoder is given by the equation below [3].

$$ r = \frac{k}{n} $$

(22)

The $n$ output bits are determined through the evaluation of a parity check on a subset of the relevant data bits. The specific set of information bits used for the calculation of each different output bit is controlled by the structure of the convolutional code. Encoder memory $K$ denotes the number of previous information bits which are stored, thus making them available for consideration during the determination of the output bits. The value of $K$ also plays a role in determining the total number of states $M$ for a given encoder. The current state of an encoder is defined by the previous $(K-k)$ information bits. In the shift register representation of an encoder, the encoder's state is obtained through the concatenation of the right-most $(K-k)$ register values.
In any case, the number of states for a given encoder may be calculated in the following manner [3]:

\[ M = 2^{(K-1)k}. \]  \hspace{1cm} (23)

In summary, the state of an encoder gives a description of past events and provides clues as to how the code will respond to future inputs.

### 3.2.4 VITERBI DECODING

In [10], the process of decoding a convolutional code is shown to be analogous to "choosing a path in the code tree whose coded sequence differs from the received one in the fewest number of places". The Viterbi Algorithm provides an efficient method by which this goal may be accomplished. The general steps involved in Viterbi decoding are explained below.

1. Based on the structure of the given convolutional code, the received sequence is broken into individual channel symbols of length \( n \) bits. Then, an \((M \times z)\) trellis diagram is created where \( M \) is the number of states and \( z \) is related to the total number of transitions.

2. Starting at the all zeros state, map all possible transitions to states in the next column and label the distance associated with each. This distance is calculated as the hamming distance between the encoded symbol and the output symbol associated with the given transition. The distance for each possible path is stored as its metric.
3. Extend all paths by one time period according to the encoder's state transition behavior, and calculate the total length of each path as the total hamming distance between the encoded symbols and output symbols associated with that path. For each ending state, where multiple paths terminate, the path with the shortest metric will be kept. This group of \( M \) paths, known as the survivors, represent the most likely paths for each ending state based on the encoded sequence.

4. Once the survivors have been determined for the final column of the trellis diagram, the ending state with the smallest metric is chosen as the most likely ending state. The state sequence associated with this ending state is then used to determine the most likely transition sequence. This transformation is made possible by the one-to-one correspondence between state sequences and transition sequences that is characteristic of convolutional encoders [6].

5. Finally, this transition sequence is passed to the destination as the decoded sequence.

An example of Viterbi decoding is shown below for the convolutional code associated with the input-output table in Figure 11. The encoded sequence is \( \{00110110\} \).
Beginning with the all-zeros state A, the two possible paths are shown above. The state transition from A to A results in the output 00 whereas the state transition from A to C produces the output 11. The values shown for the two ending nodes represent the metric of its survivor path. For the first transition, there is only one possible path for each of the ending states A and B. Therefore, the length of each path is simply the hamming distance between the branch output and the current output of the given path. Hamming distance is calculated as

\[
d(S, \hat{S}) = \sum_{i=1}^{n} (S_i \neq \hat{S}_i),
\]  

(24)

where \(S_i\) and \(\hat{S}_i\) represent the source signal and recovered signal respectively and \(n\) is the length of the sequences [3].
Following the second transition, there exist one possible path to each of the four possible ending states. Again, the metrics for each survivor path are shown in parenthesis. The general method used for calculating this value for a given node is shown below:

\[ \Gamma(x_k) = \Gamma(x_{k-1}) + d(\text{current output, branch output}), \]

where \( \Gamma(x_k) \) represents the total length of the survivor path \( x_k \) which ends after \( k \) transitions [6].
After the third transition, there exist two possible paths for each ending state. The shortest path for each ending state is selected as the survivor path. This path, along with its metric, is stored so that it may be used to calculate the survivor paths following the next transition. Therefore, there are a total of \( M \) survivor paths associated with each column of nodes, where \( M \) is the number of possible ending states.

Figure 15: Third Transition

Figure 16: Fourth Transition
This iterative method is carried out until the survivor paths reach the terminal node. The next step is to select the ending state with the smallest total length. The term length is used for simplicity's sake, but this value actually represents the total number of discrepancies between the original sequence and the maximum-likelihood decoded sequence.

![Figure 17: Survival Path](image)

The figure above shows the state sequence with the highest likelihood of producing the encoded message received by the decoder. Using the known behavior of the convolutional encoder, the input sequence which produced these state changes can be determined. In the example above, the encoder state shifts from A to A, A to C, C to D, and then D to D. Therefore, the decoded sequence is \{0111\}. 
3.3 CHANNEL CODE SIMULATION

In order to determine the benefits of Viterbi decoding, simulations were conducted for two, almost identical communication systems. The only difference between the two systems is that one uses a Viterbi-decoded convolutional code whereas the other system uses no form of channel coding. In each case, binary signaling is used to transmit data over a channel in which Additive White Gaussian Noise (AWGN) is assumed to be the only source of distortion. Using the method discussed in Chapter II for determining the decision threshold \( \gamma_0 \), maximum-likelihood detection is implemented in each simulation. Error rates at various \( E_b/N_0 \) values are calculated for each system and the results are plotted for comparison.

A popular figure of merit for determining the benefit associated with a particular channel code is called coding gain. Coding gain is defined as the difference between the \( E_b/N_0 \) values for the uncoded system and the coded system at a particular bit-error-rate. The equation for coding gain is given as

\[
G(dB) = (\frac{E_b}{N_0})_{uncoded} dB - (\frac{E_b}{N_0})_{coded} dB,
\]

and is usually measured in decibels [3]. This concept is one of several which allowed for the comparison of the two systems: the first using a Viterbi-decoded convolutional code and the second using no form of channel coding at all.

The figures below show some of the results for these simulations. Figure 18 shows the bit error rates for a system with no channel code and a system using a Viterbi decoded channel code. Figure 19 shows error rates for several codes with different code rates.
Figure 18: Performance of Viterbi Decoding

Figure 19: Performance of Channel Codes with Various Code Rates
CHAPTER 4: LOSSY SOURCE CODE USING VITERBI ALGORITHM

4.1 MOTIVATION

The impressive coding gains that have been achieved through the use of Viterbi decoding suggest that the VA may be useful in other applications. One of these potential applications involves the implementation of a lossy source code. Although source codes and channel codes perform seemingly opposite functions, the conceptual foundations for these two techniques are surprisingly similar.

Matsunaga references this theory in [11], stating that "channel coding can be considered as the dual problem of lossy source coding in rate-distortion theory". In [12], Gupta expounds on this idea, explaining that this "duality exists both in the sense of evaluating the capacity and rate distortion functions, as well as optimal coding schemes". It is well-known that the Viterbi Algorithm can be used to produce a capacity-achieving channel code [1]. It follows then, that the VA may also be used to implement an efficient source code.

4.2 METHODOLOGY

The technique used for transforming a channel code into a source code is fairly straight-forward. For the Viterbi-decoded convolutional code discussed in Chapter III, the transformation is accomplished by adjusting the order in which certain functions are carried out during the coding process. Specifically, the Viterbi decoder and convolutional encoder functions are interchanged. In the resulting system, the Viterbi decoder compresses the source signal based on the given code structure and the convolutional encoder block decompresses the information at the appropriate time. In a sense, the original signal is first "decoded" by the Viterbi Algorithm, and then "encoded" based on the convolutional code structure. As the quotations suggest, this is not
technically how the system operates but the concept remains valid. The fact that convolutional encoding and Viterbi decoding are operational inverses allows for their interchange within the system.

4.2.1 STATEMENT OF THE PROBLEM

The performance of a given source code is determined by how accurately the source signal is reconstructed from the compressed signal. The method used to determine this value depends upon the nature of the signal being compressed. In order to evaluate the efficiency of the source code proposed in this paper, simulations are conducted in which a symmetric Bernoulli source is assumed. A symmetric Bernoulli source consists of independent and identically distributed fair coin tosses [13]. In the case of digital communication, this is represented by binary sequences in which zeros and ones are equally probable. Wainwright supports the use of Bernoulli sources in simulation, writing that "Effective coding techniques for solving this binary compression problem ... serve as a building block for tackling compression of more complex sources" [13]. For this type compression, code efficiency is most aptly measured using Hamming distortion.

4.2.2 RATE DISTORTION THEORY

It has been proven that linear error correcting codes, usually associated with channel coding, can be used to implement lossless source codes for memoryless sources [12]. However, more interesting is the case in which some level of distortion is tolerated, as this allows for higher levels of data compression. Rate distortion theory suggests that for a source code with a given distortion level, there exists a maximum achievable compression rate. Furthermore, the theory states that a capacity achieving channel code can be used to create a source code capable of
achieving this rate-distortion function. As proposed in [14], the rate-distortion function is given as

\[ R(D) = 1 - h(D), \quad 0 \leq D \leq 0.5. \] (27)

In the equation above, \( D \) represents the hamming distortion between the source signal and the recovered signal. \( h(D) \) is the binary entropy function defined as

\[ h(D) = -D \log_2(D) - (1 - D) \log_2(1 - D). \] (28)

The coding rate \( R(D) \) represents the extent to which the source data is compressed. For example, if \( n \)-bit information vectors are compressed to form \( m \)-bit codewords, then \( R(D) = \frac{m}{n} \).

To illustrate this relationship graphically, the rate-distortion function has been plotted below.

*Figure 20: Rate-Distortion Function*
4.2.3 EFFICIENT CONVOLUTIONAL CODES

As some code structures provide more protection against distortion, even with the same constraint length, it is important to use the most efficient code possible. In [15] and [16], the authors solve this issue and provide the best codes for various rates. The codes which were used during the simulation of the Viterbi source code are described in the table below.

Table 1: Best Codes for Various Rates ([16], [15])

<table>
<thead>
<tr>
<th>Rate</th>
<th>Constraint Length</th>
<th>Code (octal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8</td>
<td>8</td>
<td>275 275 253 371 331 235 313 357</td>
</tr>
<tr>
<td>1/4</td>
<td>8</td>
<td>235 275 313 357</td>
</tr>
<tr>
<td>3/8</td>
<td>8</td>
<td>274 045 124 216 357 245 216 334</td>
</tr>
<tr>
<td>1/2</td>
<td>8</td>
<td>247 371</td>
</tr>
<tr>
<td>5/8</td>
<td>8</td>
<td>320 026 213 034 116 270 065 377</td>
</tr>
<tr>
<td>3/4</td>
<td>8</td>
<td>045 124 216 357</td>
</tr>
<tr>
<td>7/8</td>
<td>8</td>
<td>003 004 011 020 041 100 201 377</td>
</tr>
</tbody>
</table>
4.3 SOURCE CODE SIMULATION
The performance of the proposed source code was analyzed using the method explained below.

\[
S = \text{[binary data sequence]}
\]

\[
\hat{S} = \text{[recovered binary data sequence]}
\]

\[
C = \text{[compressed binary data sequence]}
\]

\[
\text{Length of } C = (\text{Length of } S) \times \left(\frac{k}{n}\right)
\]

\[
\hat{S} = \text{[recovered binary data sequence]}
\]

\[
S = \text{[binary data sequence]}
\]

\[
\hat{S} = \text{[recovered binary data sequence]}
\]

\[
C = \text{[compressed binary data sequence]}
\]

\[
\text{Length of } C = (\text{Length of } S) \times \left(\frac{k}{n}\right)
\]

\[
\hat{S} = \text{[recovered binary data sequence]}
\]

\[
S = \text{[binary data sequence]}
\]

\[
\hat{S} = \text{[recovered binary data sequence]}
\]

\[
C = \text{[compressed binary data sequence]}
\]

\[
\text{Length of } C = (\text{Length of } S) \times \left(\frac{k}{n}\right)
\]

\[
\hat{S} = \text{[recovered binary data sequence]}
\]

1. Generate a random sequence of binary digits in which the probability of generating a binary one is equal to the probability of generating a binary zero. This binary sequence represents the signal produced by a symmetric Bernoulli source.

2. Use the Viterbi decoder block developed in Chapter III to compress the source signal at a rate of \( \frac{k}{n} \). The decoder's exact operation depends upon the system's code structure.

3. Then, use the convolutional encoder block from Chapter III to reconstruct the original data sequence using the compressed data sequence and knowledge of the system's code structure.

4. Calculate the Hamming distortion between the source signal and the recovered signal for the various compression rates. Plot compression rate as a function of distortion.
Compare the resulting curve to the rate-distortion function in order to determine the code's relative efficiency.

Using the codes specified previously and the method outlined above, the performance of the source code was evaluated. The figure below shows these results along with the rate-distortion function provided in [14].

![Figure 22: Viterbi Source Code Performance](image)

CHAPTER 5: CONCLUSION

It has been shown in this paper that the Viterbi Algorithm may be used to create an efficient lossy source code for the purpose of compressing digital data. This conclusion comes from simulation results in which the performance of the Viterbi source code is seen to approach the theoretical rate-distortion function discussed in [14]. These results provide further support for
the duality between channel code and source code. Future work may focus on the performance of Viterbi source codes having different constraint lengths and the quantitative benefit of implementing this source code in systems that use various other modulation and coding techniques.
WORKS CITED


