1-1-2007

Channel estimation for 5.9 GHz DSRC applications.

Harb Abdulhamid
University of Windsor

Channel Estimation for 5.9 GHz DSRC Applications

by

Harb Abdulhamid

A Thesis
Submitted to the Faculty of Graduate Studies and Research through
Electrical and Computer Engineering in Partial Fulfillment
of the Requirements for the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada
2007

Your file Votre référence
ISBN: 978-0-494-42308-0
Our file Notre référence
ISBN: 978-0-494-42308-0

Abstract

Dedicated short range communications (DSRC) was established at a 5.9 GHz band for services involving vehicle-to-vehicle and vehicle-to-roadside communications. Though the standard has yet to be completed, a large amount of bandwidth has already been licensed in the U.S. and Canada. DSRC is one of the fundamental building blocks of the U.S. Department of Transportation "Vehicle Infrastructure Integration" (VII) initiative. VII envisions a nation wide system in which intelligent vehicles routinely communicate with one another and the transportation infrastructure. The purpose is to enable a number of new services that provide safety, mobility, and commercial benefits.

Most researchers have focused on the higher-level challenges with 5.9 DSRC, such as networking issues, while neglecting issues pertaining to the physical layer. Thus far, extending the symbol duration has been the only remedy suggested to resolve physical layer issues stemming from the high delay spread of harsh outdoor environments. Hence, this thesis begins with identifying the challenges in the physical layer by developing a comprehensive system model for DSRC under the channel impairments of wireless access vehicular environments. This was achieved through a worst-case scenario study whereby the conventional DSRC system was tested under varying signal-to-noise ratio, velocity, symbol durations and packet lengths.

After identifying the problem, potential remedies are discussed and analyzed. A design of a novel DSRC receiver is proposed. It will be shown that the proposed receiver has superior design characteristics for the harsh channel conditions of wireless access vehicular environments. The proposed design enables the employment of Viterbi-aided channel estimation, which substantially improves performance of the system. Novel second-order Viterbi-aided channel estimation schemes were also derived and tested. Second-order channel estimation schemes show slightly added enhancements at the cost of higher complexity.

In the name of God, most gracious, most merciful.

Acknowledgments

There are several people who deserve my sincere thanks for their generous contributions to this project. I would first like to express my sincere gratitude and appreciation to my supervisors, Dr. Esam Abdel-Raheem and Dr. Kemal E. Tepe. They have provided me with guidance and constant support throughout the course of this thesis work. They allowed me to pursue a research topic of my interest, making this learning experience an enjoyable one.

I would also like to thank my committee members Dr. Mohammed A.S. Khalid and Dr. Walid Abdul-Kader for their time spent observing my research and for providing comments and suggestions to better my work.

Many thanks to the persons who have encouraged and supported my study. To my Father, Ali, who sacrificed so much to bring his family to this country and provided his children with security and opportunity. To my Mother, Zakie, who made my life easy in every way she could. To my fiancee, Saira Shah, who has been relentlessly patient and supportive. To my little brother and sister, Mohammad and Marwa, they have always been there and I wish them success and happiness.

I would also like to thank all of my friends, who have made graduate studies a pleasant experience. I shared many late nights of both work and play among good friends, including Ali, Bahador, Marwan, Kevin Banovic, Raymond, Amir, Jay, Ashkan, Paymon, Mohammad Poustinchi, Andrew, Matthew, Aws, Omar, Song-Tao, Kevin Biswas, Junson, Mitra, and Mahzad.

Contents

List of Figures

List of Tables

List of Abbreviations

AFD	Average Fade Duration.
AWGN	Additive white Gaussian noise.
BER	Bit error rate.
BPSK	Binary phase shift keying.
dB	Decibels.
DPC	Dynamic Parameter Controlled.
DFT	Discrete fourier transform.
DSP	Digital signal processing.
DSRC	Dedicated short range communications.
DVB	Digital Video Broadcasting.
FEC	Forward error correction.
FFT	Fast fourier transform.
FIR	Finite impulse response.
FSM	Finite state machine.
Hz	Hertz.
IDFT	Inverse discrete fourier transform.
IFFT	Inverse fast fourier transform.
ISI	Intersymbol interference.
IEEE	Institute of Electrical and Electronics Engineers.
LCR	Level Crossing Rate.
LMS	Least mean squares.
LOS	Line of sight.
LS	Least squares.
MAC	Medium access control.
Mbps	Mega-bits per second.
OF	Orthogonal frequency.
OFDM	Orthogonal frequency division multiplexing.
PER	Packet error rate.
PHY	Physical layer.

PLL	Phase-locked loop.
PSK	Phase shift keying.
QAM	Quadrature amplitude modulation.
QPSK	Quadrature phase shift keying.
RMS	Root-mean-square.
SNR	Signal-to-noise ratio.
SOS	Sum-of-sinusoid.
TDMA	Time division multiple access.
USDoT	United States Department of Transportation.
VII	Vehicle infrastructure integration.
WAVE	Wireless access vehicular environments.
WLAN	Wireless local area network.

List of Symbols

The notation used in this thesis is as follows. In general, upper case letters designate information in the frequency domain, lower case letters designate information in the time domain. Also, the scalers are designated by italics while vectors and matrices are designated by bold font. The time index is always denoted as n, while the subcarrier number (i.e. frequency index) is always denoted as k. For example, an OFDM symbol at the nth time index may be represented as a vector, $\mathbf{x}_n$, that has the form,

$$\mathbf{x}_n = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

Note that each row represents a subcarrier, where $k = (0, 1, \cdots, N-1)$. Furthermore, an entire packet of data may be represented as a matrix, $\mathbf{x}$, which is a set of OFDM symbols that has the form,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x_0} & \mathbf{x_1} & \cdots & \mathbf{x_{N_p-1}} \end{bmatrix} = \begin{bmatrix} x_{0,0} & x_{1,0} & \cdots & x_{N_p-1,0} \\ x_{0,1} & x_{1,1} & & x_{N_p-1,1} \\ \vdots & & \ddots & \vdots \\ x_{0,N-1} & x_{1,N-1} & \cdots & x_{N_p-1,N-1} \end{bmatrix}$$

Again, each row represents the subcarrier number, while each column represents the symbol index. The above examples applies to data in the frequency domain as well, simply by switching the lowercase letter 'x' with an uppercase 'X'. Some commonly used symbols are listed below.

Notation	Definition
$E\{\cdot\}$	Expectation operator.
$(\cdot)^*$	Complex conjugation.
$(\cdot)_R$	Real component of a complex number.
$(\cdot)_I$	Imaginary component of a complex number.
$(\cdot)_8$	Octal representation.
$\widehat{x}$	The estimated value.
σ_τ	RMS delay spread.
f_c	Carrier frequency.
f_m	Maximum Doppler frequency.
L_{RMS}	Local RMS signal level of the fading envelope.
m	The number of constellation points in a modulation scheme.
N	Number of OFDM frequency subcarriers.
N_{ds}	Number of OFDM frequency subcarriers for data.
N_p	Number of OFDM symbols per packet.
N_R	Level crossing rate per second.
N_R^{deep}	Deep fading rate per packet.
R_c	Code rate.
R_{data}	Data rate.
T_c	Coherence time.
T_s	OFDM symbol duration.
T_p	Packet duration.
v	Velocity.

Chapter 1

Introduction

In 1999, dedicated short range communications (DSRC) was established at a 5.9 GHz band for the purpose of vehicle-to-vehicle and vehicle-to-roadside communication. The United States Department of Transportation (USDoT) has been considering DSRC for accident prevention, intelligent transport systems, open road tolling, and electronic payment systems [16]. The overall program is being referred to as the "Vehicle Infrastructure Integration" (VII) initiative. In many ways, the deployment of VII could reduce highway fatalities and improve the overall quality of life. DSRC is also being considered by various commercial industries as an opportunity to rapidly advance the development of "Smart Highways" [15]. Future DSRC applications may include in-vehicle internet, streaming multimedia, and voice-over-IP.

The physical layer (PHY) of DSRC was originally adopted from the popular wireless local area network (WLAN) standard IEEE 802.11a [1] in hopes of leveraging existing research and development and eventually to reduce hardware cost. However, IEEE 802.11a was designed for stationary indoor environments, and poses several issues in a harsh outdoor environment, especially at high velocities. Generally, the multipath propagation in urban canyons results in high multipath delay spread [42], which exacerbates the intersymbol interference (ISI). Consequently, the most recent DSRC PHY standard (IEEE 802.11p) has extended the symbol duration to better mitigate the affects of ISI.

This introductory chapter begins by introducing the essential background on this research. Section 1.1 presents a brief overview of multipath propagation in wireless channels. Section 1.2 discusses the merits of orthogonal frequency division multiplexing (OFDM) and it's applications. In Section 1.3, the objectives of this research are stated.

1.1 Channel Overview

In a wireless system with an omni-directional antenna, objects in the surrounding environment reflect and diffract the transmitted signal. This is usually made up of a specular component, i.e., line of sight (LOS), and scatter (non LOS) components. Multiple versions of the signal arrive at the receiver at different time instances, which may cause ISI. *Multipath delay spread* is used to quantify the severity of ISI according to the relations [32],

$$\bar{\tau}_i = \frac{\sum_i P(\tau_i) \cdot \tau_i}{\sum_i P(\tau_i)} \tag{1.1}$$

$$\bar{\tau_i^2} = \frac{\sum_i P(\tau_i) \cdot \tau_i^2}{\sum_i P(\tau_i)} \tag{1.2}$$

$$\sigma_\tau = \sqrt{\bar{\tau_i^2} - (\bar{\tau}_i)^2} \tag{1.3}$$

where $P(\tau_i)$ is relative normalize power obtained from the *power delay profile*, τ_i is the delay spread, σ_τ is the RMS delay spread, and i represents the path number. The channel generally induces ISI when the symbol duration is less than ten times RMS delay spread ($T_s < 10\sigma_\tau$), and it is said to have *frequency-selective fading* characteristics [32]. Channel measurements for 900MHz short range non-mobile vehicle-to-vehicle channels were reported in [42]. Though these results may not accurately reflect a DSRC channel, the report does give a rough range of values for RMS delay spread of harsh outdoor environments in different situations. Mean RMS delay spread was found to range from 20 ns to 70 ns depending on the location of the street and maximum RMS delay spread reaches levels up to 512 ns. For vehicle-to-vehicle communications in the LOS scenario, RMS delay spread would be less than 50 ns. Larger RMS delay spread usually correlates to the intersections. For a communication range greater than 300 m with no LOS present, worst case RMS delay spread reported could be up to 400 ns. In this scenario substantial multipath and high vehicle speeds will have a dramatic impact on the performance of the system.

As the vehicle moves (or any of its surrounding objects) the scatter environment changes. The movement of the vehicle introduces Doppler shift into the incident wave, hence causing envelope fading. The Doppler shift is determined by the following relation [32],

$$f_{D,n} = f_m \cos(\theta_m) = \frac{v \cdot f_c}{c} \cos(\theta_m) \tag{1.4}$$

where f_m is the maximum Doppler frequency, θ_m is the angle of arrival, v represents relative velocity in meters per second, f_c is the carrier frequency in Hertz, and c is the speed of light in meters per second. The time varying nature of the channel is quantified by the coherence time obtained from the following relation,

$$T_c = \sqrt{\frac{9}{16\pi f_m}} \tag{1.5}$$

The channel is characterized as "fast fading" when the symbol duration exceeds the coherence time. This means that the channel impulse response is changing within the symbol duration. Therefore, extending the symbol duration increases the receiver sensitivity to Doppler shift.

Two important statistics of a Rayleigh fading signal are the *level crossing rate* (LCR) and the *average fade duration* (AFD), which are derived as simple expressions by Rice [33]. The LCR is defined as the expected rate at which the Rayleigh fading envelope, normalized to the local RMS signal level (L_{RMS}), crosses a specified level (L_R) in a positive-going direction. This statistic can be used for designing error control codes and diversity schemes [32]. The number of level crossings per second has the form,

$$N_R = \int_0^\infty \dot{r}\, p(L_R, \dot{r}) d\dot{r} = \sqrt{2\pi} f_m \rho e^{-\rho^2} \tag{1.6}$$

where $\dot{r}$ is the time derivation of the received signal $r(t)$, $p(L_R, \dot{r})$ is the joint density function of r and $\dot{r}$ at $r = L_R$, and $\rho = L_R/L_{RMS}$. The maximum LCR occurs at $\rho = 1/\sqrt{2}$.

The AFD is defined as the average period of time for which the received signal is below the specified L_R. This can be computed by the relation,

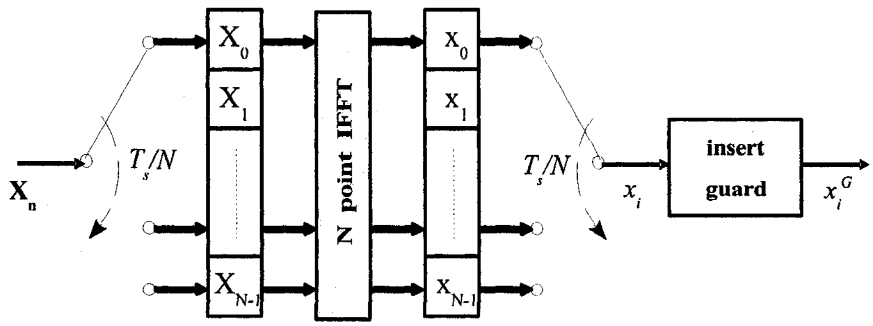
$$\bar{\tau} = \frac{1}{N_R} Pr\left[r \leq L_R\right] = \frac{e^{\rho^2} - 1}{\sqrt{2\pi} f_m \rho e^{-\rho^2}} \tag{1.7}$$

where $Pr\left[r \leq L_R\right]$ is the probability that the received signal $r(t)$ is less than L_R. This statistic can be used to estimate the average number of bits that will be eliminated during a deep fade. Note that the average fade duration is inversely proportional to the velocity.

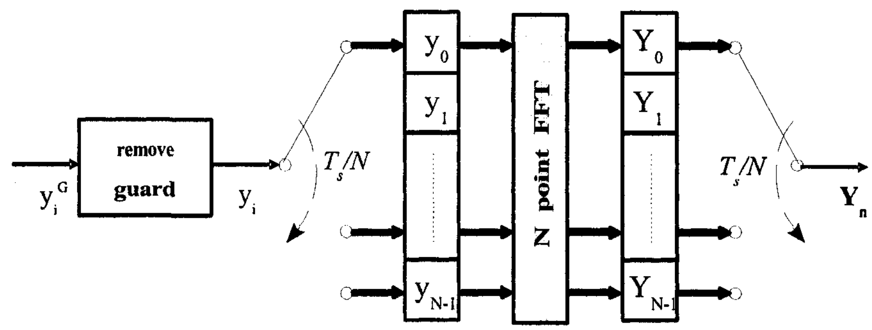
1.2 Orthogonal Frequency Division Multiplexing

The DSRC PHY utilizes OFDM, in which a single serial transmission channel is divided into a number of orthogonal parallel subcarriers to optimize the efficiency of data transmission. This bandwidth efficient signalling scheme was first proposed by Chang for digital communications [10].

To generate a baseband OFDM symbol, serial binary data is first *digitally modulated* or *mapped* using common modulation schemes such as the phase shift keying (PSK) or quadrature amplitude modulation (QAM). These *data symbols* are then converted from serial-to-parallel (S/P) before



(a) OFDM transmitter



(b) OFDM receiver

Figure 1.1: Generic block diagram of an OFDM system

modulating subcarriers. The modulation/demodulation can then be achieved in the discrete domain through the use of Discrete Fourier Transforms (DFT) as shown in Fig. 1.1.

Note that a transmitted OFDM symbol, $\mathbf{X_n}$, is a vector consisting of N data symbols. In this thesis, when the "symbol" or "symbol duration" is discussed, it is the OFDM symbol that should be kept in mind, otherwise the word "data symbol" will be used. Other literature may refer to an OFDM symbol as a "block" or "frame" to avoid confusion.

One of the advantages of OFDM is that it mitigate ISI without utilizing channel equalizers. The ISI can be mitigated using a guard interval in OFDM symbols whose length, G, exceeds the maximum excessive delay. The guard interval consists of a cyclic prefix, which is a copy of the symbol tail that is placed at the front of the symbol, and is later removed at the receiver. This makes OFDM an ideal candidate for multipath scatter environments, where ISI inherently exists. OFDM is a superior modulation technique for high-speed wireless communications. It has also been used to implement digital audio broadcasting (DAB) [3], digital video broadcasting (DVB) [4], WLAN standards (IEEE

802.11a/g and HIPERLAN/2), and new broadband wireless access standard WiMax (IEEE 802.16) [2].

Generally, the guard interval length is no more than 20 percent of the symbol duration [25]. Increasing the guard interval would be at the cost of low spectral efficiency. In IEEE 802.11p, the reason for increasing the symbol duration was to increase the guard interval without sacrificing the spectral efficiency.

1.3 Thesis Objectives

In the early stages of this research, 5.9 GHz DSRC was still a new standard under development. The vast majority of the public literature focused on the challenges in the network layer. There was a lack of research on the challenges in the physical layer. So the initial objective of this thesis is to identify the issues with the current DSRC physical layer design and to determine its limitations in wireless access vehicular environments (WAVE). The goal is to achieve a comprehensive simulation environment in Matlab with various parameters to examine the countless possible scenarios encountered in WAVE. The conventional system model and performance study are discussed in Chapter 2 and 3, respectively.

As this research progressed, it became evident that the main problem with the conventional system was ineffective channel estimation. Consequently, the second objective is to investigate and provide resolutions to the problem at hand, by applying both existing and novel concepts to the system. The final goal is to propose channel estimation schemes that improve the system performance while maintaining the existing protocol. The proposed schemes are thoroughly tested and compared in the simulation environment. The design methodology and performance study of the proposed improvements to the system are discussed in Chapter 4. An additional method for improving performance are proposed in Chapter 5.

This thesis conforms to the objectives stated above. Since the IEEE 802.11p standard is still in the development stage [5], the results of this study could contribute to its development.

Chapter 2

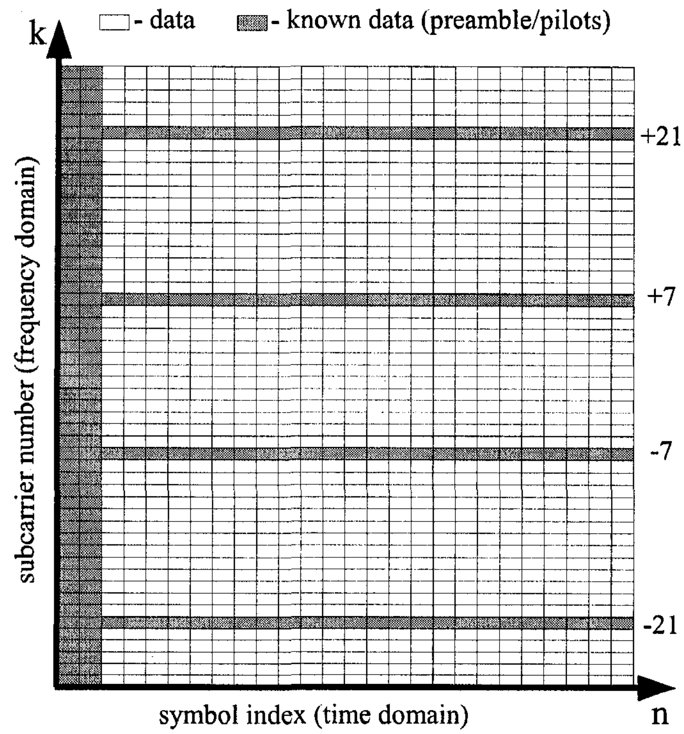
DSRC System Model

The DSRC standard is currently the same as IEEE 802.11a, with an exception to the symbol duration (8.0μs with 1.6μs guard interval) and signal bandwidth (~10 MHz). DSRC uses 64 frequency subcarriers with only 52 subcarriers actually used for signal transmission. Of the 52 subcarriers, 4 are pilots used for phase tracking, while the remaining 48 subcarriers are for data. The DSRC PHY packet structure before guard insertion and after guard insertion is shown in Figs. 2.1(a) and 2.1(b), respectively. Each packet consists of two preambles. The first preamble consists of ten short training symbols for packet detection, frequency offset estimation, and symbol timing based on [35]. The second preamble, consists of two identical training symbols used for channel estimation ($\mathbf{X_{train}}$) subsequent to a long guard interval of length $G_{CE} = 3.2$ μs. The information data rate is calculated by the relation [1],

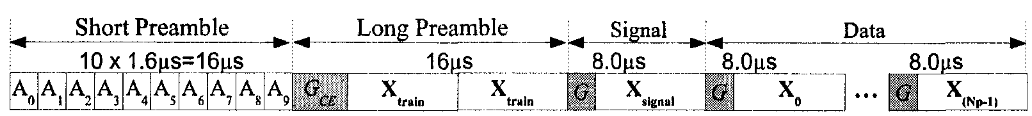
$$R_{data} = \frac{log_2(m) \cdot R_c \cdot N_{ds}}{T_s}, \tag{2.1}$$

where m is the number of constellation points of the modulation scheme, R_c is the channel coding rate, N_{ds} is the number of data subcarriers, and T_s is the OFDM symbol duration. In DSRC, T_s and N_{ds} are fixed at 8μs and 48 subcarriers respectively. The data rate is then determined only by selecting a modulation scheme and channel coding rate. Table 2.1 shows the possible transmission modes available in the current standard. The transmission mode and packet length is determined from $\mathbf{X_{signal}}$. Note that the preamble is always modulated using Binary Phase Shift Keying (BPSK).

In this chapter, an entire DSRC system is modeled based on the latest known standards and literature. Section 2.1 describes in detail the baseband components of the transmitter and receiver.



(a) Before inserting guard interval



(b) After inserting guard interval

Figure 2.1: DSRC transmission packet format

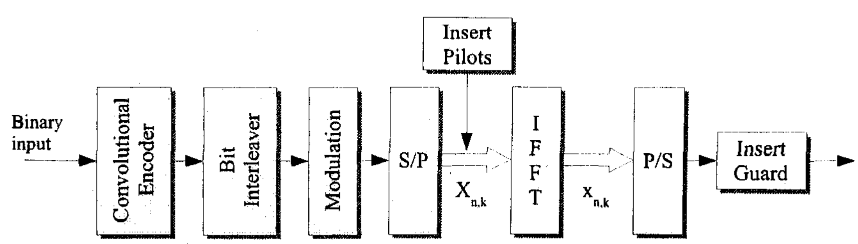
An appropriate channel model is discussed and presented in Section 2.2.

2.1 System Components

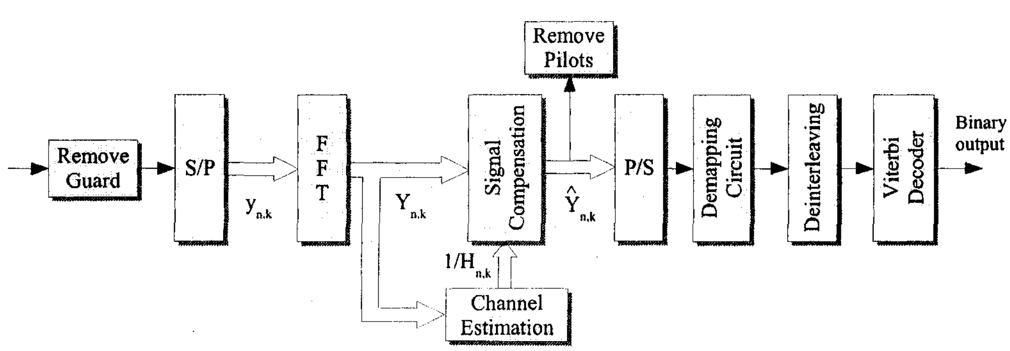
This Section presents the design of a conventional DSRC system. Fig. 2.2(a) and 2.2(b) show the transmitter and receiver configurations respectively. In these figures, the thin line arrows represent the transmission of serial data, while the thick block arrows represent the transmission of parallel data. The baseband components of the system are individually discussed in the following subsections.

Table 2.1: DSRC transmission modes

MODE	MODULATION	INTERLEAVING	CODE RATE	DATA RATE
1	BPSK	6x8	1/2	3 Mbps
2	BPSK	6x8	3/4	4.5 Mbps
3	QPSK	12x8	1/2	6 Mbps
4	QPSK	12x8	3/4	9 Mbps
5	16-QAM	12x16	1/2	12 Mbps
6	16-QAM	12x16	3/4	18 Mbps
7	64-QAM	18x16	2/3	24 Mbps
8	64-QAM	18x16	3/4	27 Mbps

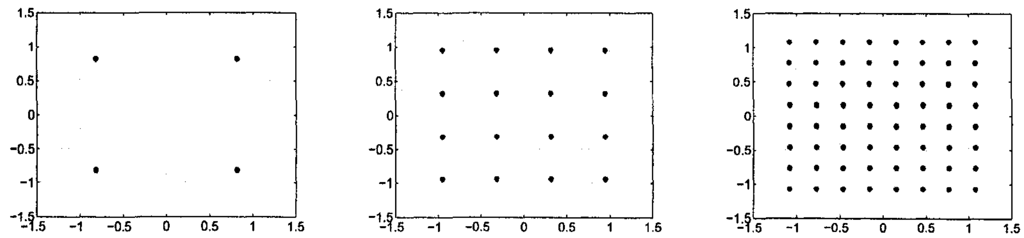


(a) Transmitter



(b) Receiver

Figure 2.2: System model components and configuration



(a) QPSK (b) 16-QAM. (c) 64-QAM.

Figure 2.3: Normalized signal constellations.

2.1.1 Digital Modulation

The digital modulation schemes available for DSRC include binary phase shift keying (BPSK), quadrature phase shift keying (QPSK), 16- quadrature amplitude modulation (16-QAM), and 64-QAM. All these schemes are explained in [26] and [30]. Digital modulation involves converting a set of bits into a complex number (or a *data symbol*) representing a constellation point of the corresponding mapping scheme. In DSRC, the conversions are Gray-coded constellation mappings as described in [1]. After mapping, the symbols must be normalized in order to maintain the same power for all mapping schemes. At the receiver, there are two ways to demodulate the corrupted data symbols. The first method is known as *hard detection* or *slicing*, which simply involves taking the nearest constellation point to the received data symbol. For BPSK and QPSK, hard detection is as simple as looking at the polarity and phase of the received symbol. However, 16-QAM and 64-QAM are more complicated and based on decision-boundaries. The second method is known as *soft detection*, which involves representing the symbol by a higher number of bits [34]. These bits are sometimes called *soft bits*. This passes on more information to the decoder, however can dramatically increase the complexity in cases of larger constellations such as 16-QAM and 64-QAM. The benefits of soft detection will be further discussed in the next section.

2.1.2 Channel Coding

Forward error correction (FEC) code has been a used in digital communications to combat errors due to noise, fading, interference, and other channel impairments. In other literature FEC coding may be referred to as "channel coding" or "error control coding". Error correction coding involves introducing controlled redundancies to a transmitted signal so that they may be exploited at the receiver. The redundancy of the encoded data is quantified by the ratio of b_d data bits per b_c encoded bits [39],

$$R_c = \frac{b_d}{b_c} \text{ where } (b_d < b_c) \tag{2.2}$$

this ratio is known as the *code rate*, which basically means for every b_d data bits input into the encoder, there will be b_c code bits output from the encoder. Generally, channel codes are categorized into two main types: *block codes* and *convolutional codes.*

Block coding is basically generating a "codeword" of b_c coded bits that would be algebraically related to a sequence of b_d data bits. Convolutional coding is generated by the discrete-time convolution of a continuous sequence of input data bits. Both block code and convolutional code are employed in wireless systems. However, convolutional codes have proven superiority over block codes for a given degree of decoding complexity [40].

Consider a rate-$1/b_c$ convolutional encoder with an input sequence $\mathbf{a}$, which generates an output of $\mathbf{b}_i^j$ for each input bit a_i, where $i = (0, \cdots, l-1)$ and $j = (0, \cdots, b_c - 1)$. The convolutional encoder can be described as a set of impulse responses, $\mathbf{g}^j$, where each impulse response is referred to as a *generator sequence* [39]. The generator sequence has the form,

$$\mathbf{g}^j = (g_0, g_1, ..., g_{K-1}) \tag{2.3}$$

where K is the *constraint length*, which is defined as the number of shifts of a single bit through the encoder. In other words, if the encoder has ν-stage binary shift registers, the constraint length is equal to $K = \nu + 1$. The output sequence of the encoder is obtained from the discrete convolution,

$$\mathbf{b}^j = \mathbf{a} \circledast \mathbf{g}^j \tag{2.4}$$

where $\circledast$ denotes modulo-2 convolution. Fig. 2.4 shows a block diagram of the generic convolutional encoder for WLAN systems [1], which is described in detail in [9]. This is a rate-1/2 convolutional encoder has a set of generator sequences,

$$\mathbf{g}^1 = (1011011) = (133)_8 \tag{2.5}$$

$$\mathbf{g}^2 = (1111001) = (171)_8 \tag{2.6}$$

A rate-$1/b_c$ convolutional encoder is basically a *finite state machine* (FSM), where the state of the encoder is defined by content of the shift registers. Therefore, there are a total of 2^ν encoder states and there are two possible state transitions (0 or 1) from each individual state. The *Viterbi Algorithm*, is a popular technique for decoding convolutional code first introduced in [40]. Since an FSM is an example of a *Markov chain* [30], it was later recognized that the Viterbi algorithm was a

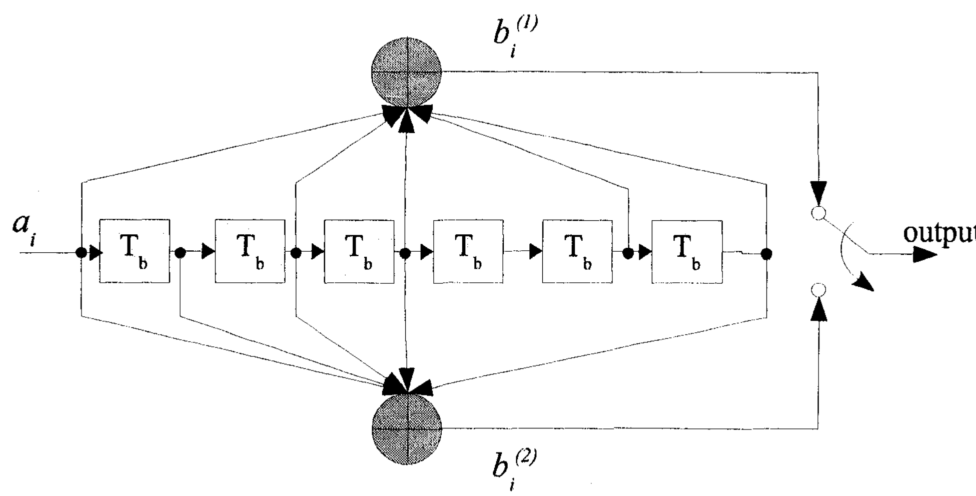


Figure 2.4: Rate-1/2 convolutional encoder with a constraint length of K=7 and a generator matrix of $[133_8, 171_8]$

computationally efficient technique for determining the most probable path taken through a Markov graph [21]. A brief history and overview of the algorithm and its development is presented in [41].

Viterbi decoding involves stepping through a trellis of the Markov graph. For each path through the trellis, the algorithm computes the "distance" as a metric to quantify the discrepancies between the received sequence and the possible coded sequence. Only the paths through the trellis that have minimum distances entering a state are considered. The Viterbi algorithm avoids the need to consider every possible coded sequence and therefore substantially reduces the complexity.

There are two types of decoding: *hard decoding* and *soft decoding.* Hard decoding employs the *Hamming distance* metric, which is the number of bits that differ between two equal length binary sequences. Soft decoding employs the *Euclidean distance* metric, which is the ordinary distance between two points of a constellation by application of the Pythagorean theorem. Soft decoding requires soft detection at the demapping circuit. According to [9], 6-bit quantization of both the real and imaginary components of the received data symbols will suffice in achieving performance close to floating point accuracy.

2.1.3 Block Interleaving

The decoder operates under the assumption that the errors will be random or spaced apart. However, in fading channels, deep fades may cause a long sequence of errors, which may render the decoder ineffective. In order to alleviate bit correlation, the encoded bits are scrambled with a block interleaver. Interleaving can either be done before symbol mapping, and is known as *bit interleaving*, or it can be done after symbol mapping, and is known as *symbol interleaving.* Based on preliminary trial simulation runs, it was determined that bit interleaving had yielded better PER performance in this DSRC system. Recall that the purpose of interleaving is to minimize the bit correlation, while data symbols are essentially groups of bits. Maintaining the bits in groups, in a sense, adds to the bit correlation, especially when using larger constellations. It is for that reason only bit interleaving is considered in this thesis.

The block interleaver is also known as a *row-column interleaver*, since works by grouping a block of IxJ bits into I rows and J columns. It involves "inputting" the rows and "outputting" the columns. The interleaver block size would correspond to one OFDM symbol. The dimensions of the block depend on the modulation scheme selected. The block dimensions are summarized in Table 2.1.

2.1.4 Channel Estimation

At the receiver, the guard interval is removed from the received signal, then the received data is converted to parallel, which is denoted as $y_{n,k}$ in Fig. 2.2(b). At this point $y_{n,k}$ is demultiplexed into the FFT, yielding the following output in the frequency domain,

$$FFT[y_{n,k}] = Y_{n,k} = H_{n,k} \cdot X_{n,k} + W_{n,k}, \tag{2.7}$$

where $H_{n,k}$ denotes the channel frequency response at the n_{th} symbol index of the k_{th} subcarrier, $W_{n,k}$ represents the additive white Gaussian noise (AWGN), and $X_{n,k}$ is the data that was input into the inverse FFT (IFFT) of the transmitter. If it is assumed that $X_{n,k}$ is known, then the channel frequency response Eq. (2.7) can be deduced as follows,

$$\begin{aligned} H_{n,k} &= \frac{Y_{n,k} - W_{n,k}}{X_{n,k}} \\ &= \frac{Y_{n,k}}{X_{n,k}} - \frac{W_{n,k}}{X_{n,k}} \\ &= \widehat{H}_{n,k} + \psi_{n,k} \end{aligned} \tag{2.8}$$

where $\widehat{H}_{n,k}$ is the estimated channel response based on the least squares (LS) method with an error component of $\psi_{n,k}$ due to the AWGN. Therefore, the channel estimate accuracy is reduced with an increase in noise power. This effect is known as *noise enhancement.* In order to reduce the effect to noise enhancement, the channel estimator employs the second preamble, which consists of two identical training symbols presented in Fig. 2.1. The estimated channel response is computed as the average channel response over the first and second received OFDM symbols $Y_{-1,k}$ and $Y_{-2,k}$,

$$\widehat{H}_{0,k} = \frac{Y_{-2,k} + Y_{-1,k}}{2 \cdot X_{train,k}} \tag{2.9}$$

In conventional channel estimation, it is assumed that the channel exhibits time-invariant fading. In other words, the channel response remains relatively constant for the entire packet length, so it is assumed that $\widehat{H}_{n,k}$ is approximately $\widehat{H}_{0,k}$ for the duration of the packet. In this model that translates to,

$$\widehat{H}_{n,k} \approx \widehat{H}_{0,k} \tag{2.10}$$

At the signal compensator, all of received data symbols of the packet are compensated by the estimated channel response. The signal compensator is similar to an equalizer, except that there is one tap per subcarrier. In this conventional model, the tap weights are fixed for each packet. The coefficient vector is the inverse of the estimated channel response,

$$w_{n,k} = \frac{1}{\widehat{H}_{0,k}} \tag{2.11}$$

The received data is compensated as follows,

$$\widehat{Y}_{n,k} = Y_{n,k} \cdot w_{n,k} \tag{2.12}$$

where $n = 0, 1, \cdots, N_p - 1$.

2.2 WAVE Channel Model

At this time, empirical channel models for a 5.9 GHz DSRC system are not publicly available. This study utilizes the appropriate statistical models in representing and simulating the WAVE channel. The WAVE channel can be modeled using statistical models presented in [39]. *Rayleigh fading* channels may represent 2D isotropic scattering environments without a specular component. Under Rayleigh fading, the received complex envelope is treated as a wide-sense stationary Gaussian random process with zero mean. For DSRC applications, it would be appropriate to model the propagation as a scattering environment with a specular component. In that case *Rician fading* is considered, where the received complex envelope is treated as a wide sense stationary nonzero mean Gaussian random process. The angle of arrival distribution of the received signal, which consists of a large number of plane waves, may have the form [6],

$$p(\theta) = \frac{1}{K+1}\widehat{p}(\theta) + \frac{K}{K+1}\delta(\theta - \theta_0) \tag{2.13}$$

where, $\widehat{p}(\theta)$ is continuous distribution, which represents the scatter components, and θ_0 is the angle of arrival of the specular component. The *Rice Factor*, K, is defined as the ratio between the specular power and scatter power. When $K = 0$ the channel exhibits *Rayleigh fading*, when $K = \infty$ there is no fading.

Jake's Sum-of-Sinusoid (SOS) method is used to simulate a 1-path Rayleigh fading channel. As shown in [29], it is an effective method in the range of doppler rates considered in this study. Jake approximates a 2-D isotropic scattering environment by choosing N scatter components uniformly distributed,

$$\theta_i = \frac{2\pi i}{N}, i = 1, 2, \cdots, N. \tag{2.14}$$

where $N/2$ is chosen to be an odd integer. The mathematical model that produces a zero-mean Gaussian envelope obtained by the following relation,

$$\begin{aligned} g(t) &= g_I(t) + j \cdot g_Q(t) \\ g(t) &= \sqrt{2}\left[2\sum_{n=1}^{M} \cos\beta_n \cos 2\pi f_n t + \sqrt{2}\cos\alpha \cos 2\pi f_m t\right] \\ &\quad + j\left[2\sum_{n=1}^{M} \sin\beta_n \cos 2\pi f_n t + \sqrt{2}\sin\alpha \cos 2\pi f_m t\right] \end{aligned} \tag{2.15}$$

there are M low-frequency oscillators with frequencies,

$$f_n = f_m \cos\frac{2\pi n}{N}, n = 1, 2, \cdots, M \tag{2.16}$$

where $M = \frac{1}{2}\left(\frac{N}{2} - 1\right)$. For the simulations in this thesis, a typical Rayleigh faded envelope is obtained by choosing $M = 8$ and $\beta_n = \frac{\pi n}{M}$.

Fig. 2.5 shows the multipath channel simulator used in this study. The simulation includes a specular component and scatter components. The scatter components are modeled based on the power delay profile with a tap delay line and specified gain for each path. Each tap gain must be individually and independently faded in order to achieve wide-sense stationary uncorrelated scattering. According to [19], this can be achieved having random or spaced-out initial phases for each individual fading simulator.

According to [42], multipath RMS delay spreads can reach up to 512 ns, and the largest delay spreads usually correlate to the street intersections. In an urban canyon, the excessive delay may exceed the length of the guard interval of 802.11a. Since empirical channel models for a 5.9 GHz DSRC system are not yet publicly available, the simulations represented the "urban canyon scenario" with a 2-path Rayleigh fading channel whose power delay profile is $P(\tau_k) = [0, -7.5dB]$, with a delay

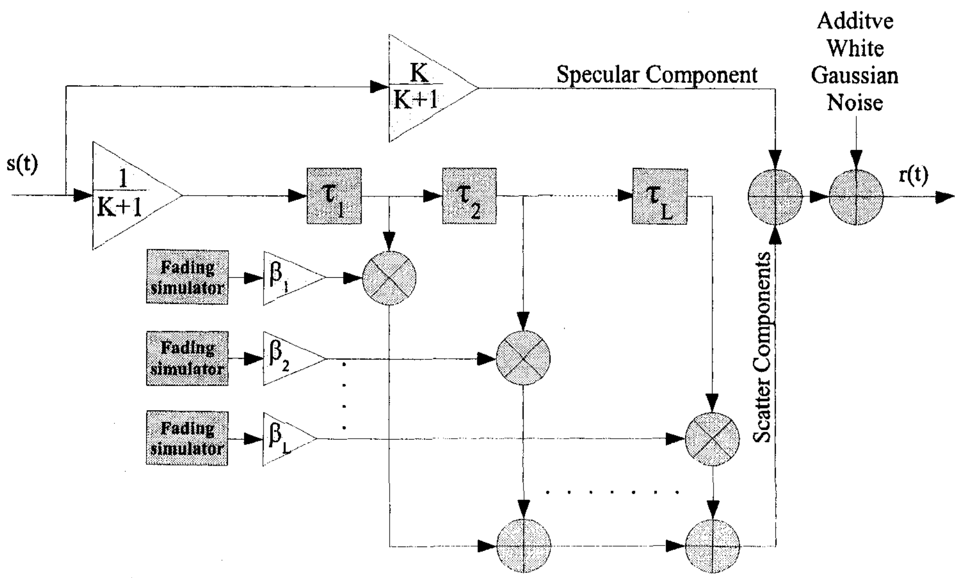


Figure 2.5: WAVE multipath channel simulator

of 1 μs, which yields an RMS delay spread of 358ns. That RMS delay spread is within the range of RMS delay spreads mentioned in [42].

Chapter 3

Performance Limitations of Conventional Channel Estimation

In the feasibility study in [42], the authors conclude that the conventional physical layer design is feasible for DSRC applications. However, only scenarios with very strong specular components are considered in the study. Though in DSRC systems with a strong specular component may be expected in many scenarios, situations will arise where there is no LOS present.

Another feasibility study [36], examined the packet efficiency at varying velocities, but considered Viterbi decoding without channel estimation. The same authors later proposed a resolution in [37] to improve the packet performance at high velocities. However, this required major modifications to the protocol, and would also obstruct the USDoT's plans of achieving inter-operability [16] with other systems.

This chapter considers the feasibility of the conventional system model presented in chapter 2 for DSRC applications. This chapter begins with a review of the static channel assumption by way of analysis in Section 3.1. Section 3.2 presents a comparative study on how different parameters will effect the packet performance. Since the purpose of this chapter is to determine the limitations of the system, only Rayleigh fading (no LOS) was considered since it yields the worst-case performance.

3.1 Problem Statement and Analysis

According to [32], if $T \ll T_c$, then the channel will have static channel characteristics over a period of T seconds. Therefore, at a 5.9 GHz band the channel would have static characteristics over the entire packet duration only when satisfying the requirement,

$$T_p \ll T_c \tag{3.1}$$

where $T_p = T_s \cdot N_p$ is the packet duration and N_p is the number of OFDM symbols per packet. The coherence time is simplified from Eq. (1.5),

$$\begin{aligned} T_c &= \sqrt{\frac{9}{16\pi f_m}} \\ &= \frac{0.423}{f_m} \\ &= \frac{0.423 \cdot c}{v \cdot f_c} \\ &= \frac{0.0215}{v} \end{aligned} \tag{3.2}$$

Therefore, the static channel requirement in Eq. (3.1) can be rewritten as follows,

$$T_s \cdot N_p \cdot v \ll 0.0215 \tag{3.3}$$

where v is the relative velocity in meters per second. If any one of the three parameters (T_s, N_p, or v) is increased while holding the others constant, the assumption in Eq. (2.10) may be rendered invalid. For example, if two vehicles travel on the highway in opposite directions, the vehicles may reach a relative velocity of 250 km/h. Recall that T_s is equal to $4\,\mu s$ and $8\,\mu s$ for IEEE 802.11a and DSRC respectively. In these cases, the assumption in Eq. (2.10) will only hold true if $N_p < 0.386$ for DSRC and $N_p < 0.772$ for IEEE 802.11a. At such a high velocity, the assumption can never be true, even for short packet lengths. At a low velocity of 5km/h, Eq. (3.3) requires $N_p < 1.5$ for DSRC and $N_p < 3$ for IEEE 802.11a. Clearly, a packet length greater than three symbols would be preferred for transmission, but if the limit is exceeded there will be degradation in performance.

Fig. 3.1 illustrates different Rayleigh fading envelopes produced by the channel simulator discussed in the previous chapter. Each plot is a random example of possible fading envelopes across a packet length of 100 OFDM symbols per packet, there the packet duration is $0.8\,ms$. In the figure, at a relative velocity of 2 km/h, it seems reasonable to assume static channel characteristics. However, at higher velocities, the channel is rapidly fluctuating. Aside from the rapid fluctuation, deep fades may also occur, which practically eliminate the signal. Notice in Fig. 3.1, that at 110 km/h only

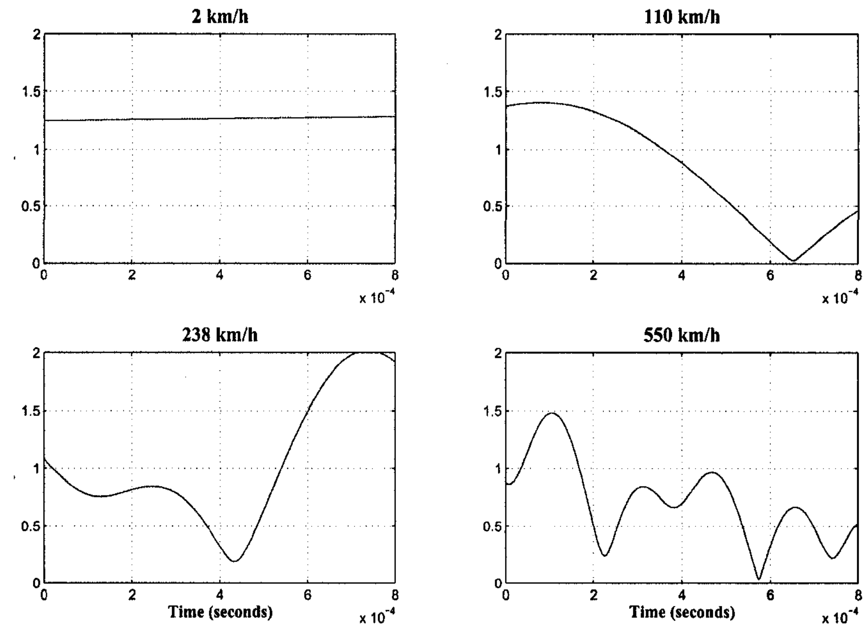


Figure 3.1: Examples of fading envelopes at different velocities for 5.9 GHz DSRC ($T_s = 0.8\,\mu s$) over a packet duration of 0.8 ms ($N_p = 100$).

one deep occurs, while at 550 km/h several deep fades can be seen. Recall that Eq. (1.6) is used to compute the LCR at a specified level. Though the envelope experiences shallow fades frequently, occasionally a very deep fade may occur. As velocity is increased, the LCR will increase, and therefore, the number of deep fades per packet will also increase. Deep fades occur when the envelope drops 75 % below the RMS amplitude ($\rho = 0.25$). The rate of deep fades per packet is denoted as N_R^{deep}. At a 5.9 GHz band, this can be computed by the following relation,

$$N_R^{deep} = 1.104 \cdot f_m \cdot T_p \tag{3.4}$$

For example, when the packet duration is $0.8ms$ and the relative velocity is 110 km/h, the channel will experience 0.53 deep fades per packet. Note that when deep fades occur, very low noise power can generate errors. This means that more than half of the packets transmitted will be prone to errors.

Considering the envelope fading alone, it would seem that DSRC's extended symbol duration would have a negative impact on PER performance. However, the effects of multipath delay spread

Table 3.1: Simulation parameters and values

NAME	ABBREV.	UNIT	VALUES
Signal-to-Noise Ratio	SNR	dB	[1, 2, 3, 6, 10, 15, 20, 25, 30, 35]
Maximum Doppler Freq. (Velocity)	f_m (v)	Hz (km/h)	[10, 200, 400, 600, 800, 1100, 1300] (2, 37, 74, 110, 147, 202, 238)
Symbol Duration	T_s	μs	[0.8, 4, 8]
Packet Length	N_p	symbols/packet	[10, 64, 100, 250, 325]
Rice Factor	K	-	[0 (Rayleigh fading)]
Modulation	m	-	[QPSK]
Decoding	-	-	[none, hard, soft]

must also be considered. Recall the study [42], which claimed multipath RMS delay spreads can reach up to $512ns$. In an urban canyon, the excessive delay may exceed the length of the guard interval of IEEE 802.11a. This would result in ISI corrupting data, which exhibits high PER regardless of high SNR and zero mobility. The doubling of DSRC's symbol duration would remove the ISI as long as the maximum excess delay does not exceed the guard interval. Therefore, doubling the symbol duration was indeed justified. This will be shown by simulation in the following section.

3.2 Simulation Results

The IEEE 802.11a and DSRC physical layer have been simulated in Matlab v7.0.1. The packet performance is measured in terms of packet error rate (PER). A packet error occurs whenever packet has at least one bit error. The bit error rate (BER) is of little importance in packet transmission, while PER is the true metric of the system performance. However, the BER curves can provide an insight to the dispersion of the errors. For the purpose of this feasibility study, only QPSK modulation is considered when obtaining the performance curves. The results of QPSK modulation can be extended to other M-ary QAM modulations without loss of generality.

In each simulation, the PER was determined based on the transmission of 10,000 packets under varying SNR or velocity. The simulation range of SNR was the generic range of 0 to 35 dB. For velocity, it was assumed that 240 km/h would be the legal maximum of relative vehicular velocity, and so that was the maximum simulated. Table 3.1 provides of a summary of the parameters chosen for simulation. The problem statement simplified in Eq. (3.3) will be consolidated by the results presented in the following subsections. Sections 3.2.1 and 3.2.2 focus on the performance of conventional channel estimation without coding. Section 3.2.3 discusses the improvements made by the Viterbi decoder.

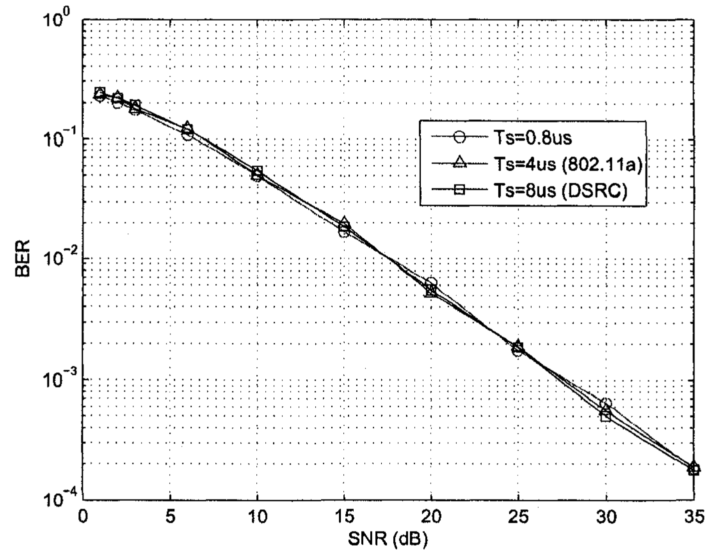
3.2.1 Varying Symbol Duration

The focus of this section is on the effect of increasing or decreasing the symbol duration. The simulations were carried out at varying symbol durations, including $8\,\mu s$ for DSRC, $4\,\mu s$ for IEEE 802.11a, and $0.8\,\mu s$ in order to show the performance trend. The guard interval length was always 20 percent of the symbol duration ($G = 0.20 \cdot T_s$). Recall that the "urban canyon scenario" was simulated as a 2-path Rayleigh fading channel whose power delay profile is $P(\tau_k) = [0, -7.5dB]$, with a delay of $1\,\mu$s, which yields an RMS delay spread of $358\,ns$. The maximum excess delay exceeds the guard interval of IEEE 802.11a to capture the effect of ISI.

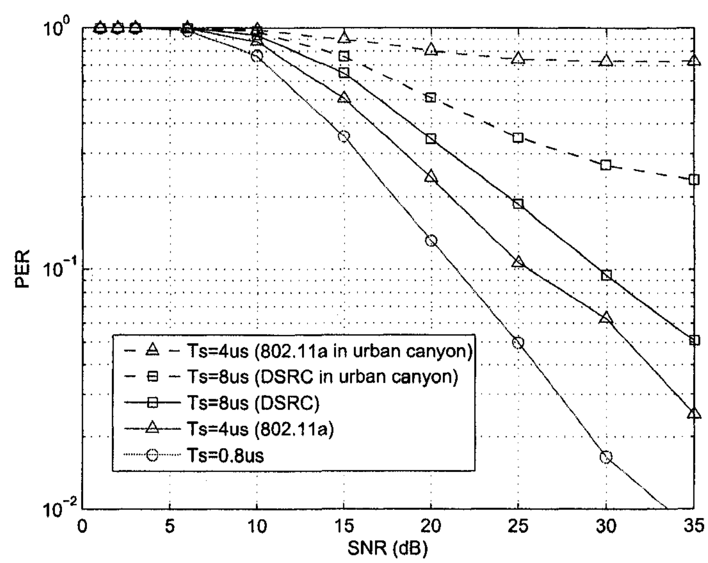
The BER and PER curves under Rayleigh fading for varying symbol durations with a fixed packet length of $N_p = 64$ are shown in Figs. 3.2 and 3.3. The plots in Fig. 3.2 are error curves versus SNR at a fixed velocity of 238 km/h, while the plots in Fig. 3.3 are error curves versus velocity at fixed SNR of 30 dB. Notice from the figures 3.2(a) and 3.3(a) that the BER is independent of the symbol duration.

In Fig. 3.2(b), the PER performance at high velocity in terms of SNR is shown. Increasing the symbol duration in the 1-path Rayleigh fading channel degrades the PER performance. This corresponds to what was expected in Eq. (3.3). However, it can be seen that IEEE 802.11a fails in the urban canyon scenario since the delay exceeds the guard interval. The PER curve converges to a flat line due to the ISI that badly distorts the signal regardless of SNR. In this case DSRC outperforms IEEE 802.11a. Since the guard interval exceeds the delay spread DSRC only experiences "self interference". This only introduces a phase shift and can be completely eliminated using phase tracking [11].

In Fig. 3.3(b), the effects of increasing the symbol duration is illustrated against varying velocity. Notice that the performance degrades as velocity is increased. This was the expected outcome based on Eq. (3.3). Also, when 1-path Rayleigh fading is considered, the performance degrades with the increase of the symbol duration. In the case of zero delay spread, IEEE 802.11a outperforms DSRC at high velocity. However, when high delay spread of the urban canyon scenario is considered, IEEE 802.11a fails even when there is no mobility. As previously discussed, this is due to the fact that the maximum excess delay exceeds the guard length.

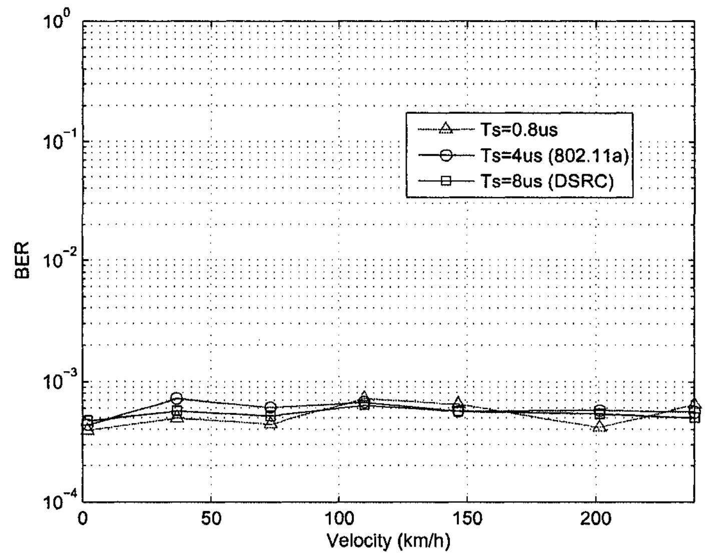


(a) BER vs. SNR

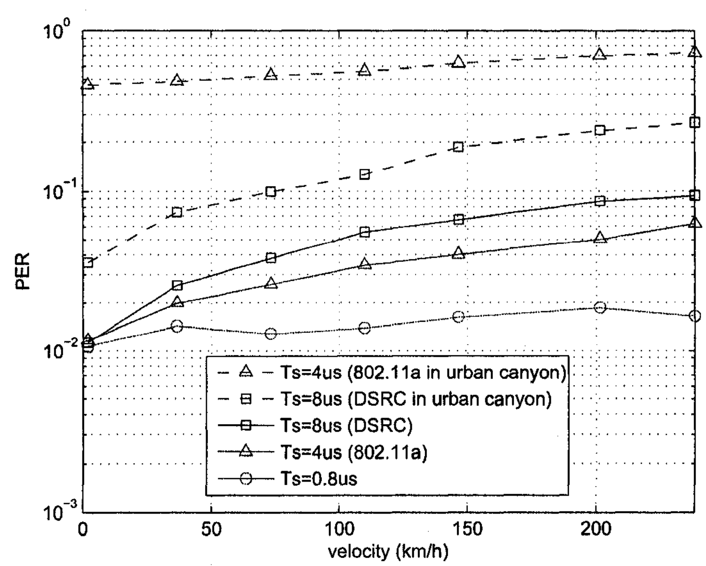


(b) PER vs. SNR

Figure 3.2: Symbol duration comparison against SNR at 238 km/h.



(a) BER vs. Velocity



(b) PER vs. Velocity

Figure 3.3: Symbol duration comparison against velocity at 30 dB.

3.2.2 Varying Packet Length

The focus of this section is on the effect of increasing or decreasing the packet length. The information size of a packet in bytes is obtained from the following relation,

$$\text{Information Size} = N_p \cdot \log_2 m \cdot N_{ds}/8 \tag{3.5}$$

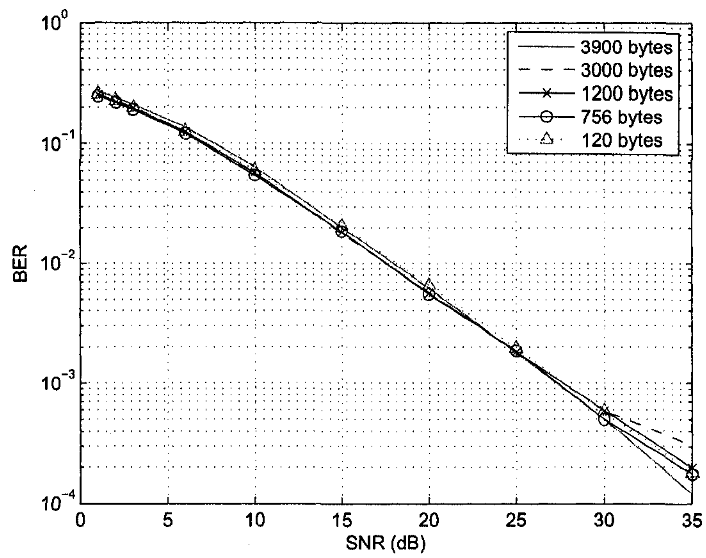
where $\log_2 m$ is the number bits per data symbol and N_{ds} is the number of data subcarriers. The BER and PER curves under Rayleigh fading for varying packet lengths with a fixed symbol duration of $T_s = 8\,\mu s$ are shown in Figs. 3.4 and 3.5. The plots in Fig. 3.4 are error curves versus SNR at a fixed velocity of 238 km/h, while the plots in 3.5 are error curves versus velocity at fixed SNR of 30 dB. Notice from the figures 3.4(a) and 3.5(a) that the BER is independent of the packet length.

In Fig. 3.4(b), the PER performance at high velocity in terms of SNR is shown at varying packet lengths. Increasing the packet length in the 1-path Rayleigh fading channel dramatically degrades the PER performance. Fig. 3.5(b) presents the PER performance in terms of velocity, and again the PER performance degrades with the increase in packet length. These results further confirm Eq. (3.3), since increasing N_p degrades the performance.

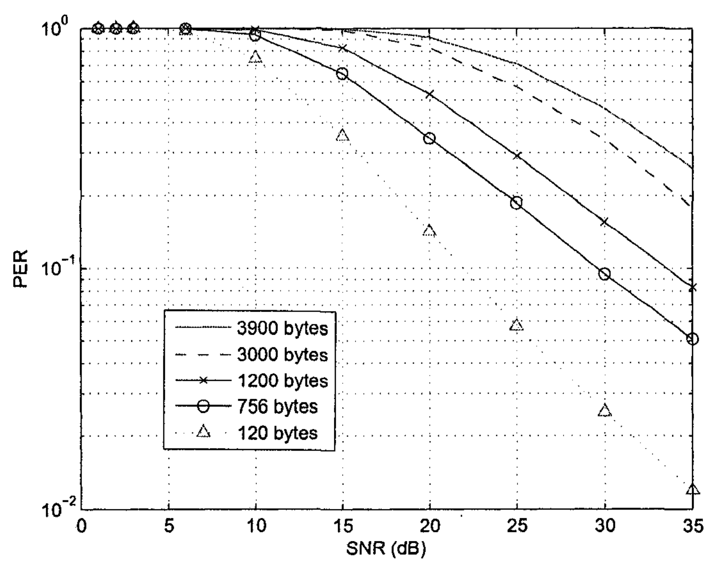
3.2.3 Viterbi Decoding

The previous chapter discussed how FEC codes along with block interleaving may be used to improve the performance of the receiver. This section focuses on the effect of employing the Viterbi Algorithm for decoding with a fixed packet length and symbol duration of 64 and $8\,\mu s$, respectively. It was found that Viterbi decoding does improve the performance under 1-path Rayleigh fading as shown in Fig. 3.6. In Fig. 3.6(a), the PER performance versus SNR is shown at a fixed velocity of 238 km/h. Hard decoding improves the performance by approximately a 5 dB gain, while soft decoding with 6-bit quantization provides an additional 2 dB gain at a cost of higher complexity. Recall that higher modulation schemes such as 16- and 64-QAM require much more processing for soft detection at the demodulator. For this reason, hard decoding was selected for the continued research of this thesis.

A similar relationship can be seen in Fig. 3.6(b). Notice that the PER is reduced but the sensitivity to Doppler still exists. The conventional DSRC system only utilizes the decoder to reduce error, but does not improve the channel estimate. The next chapter will discuss how the Viterbi decoder may be better utilized to improve the channel estimate.

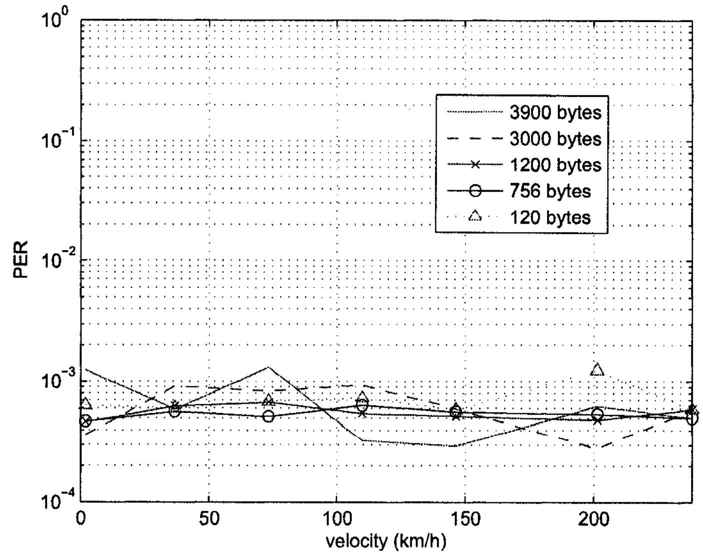


(a) BER vs. SNR

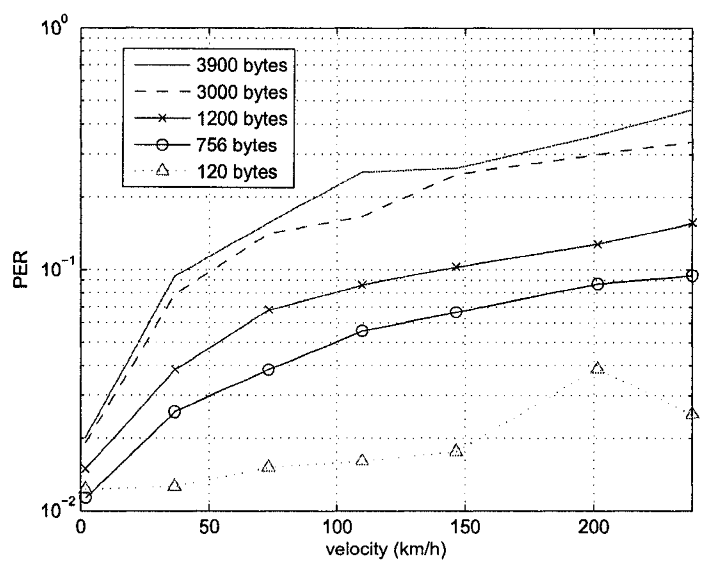


(b) PER vs. SNR

Figure 3.4: Information size comparison against SNR at 238 km/h.

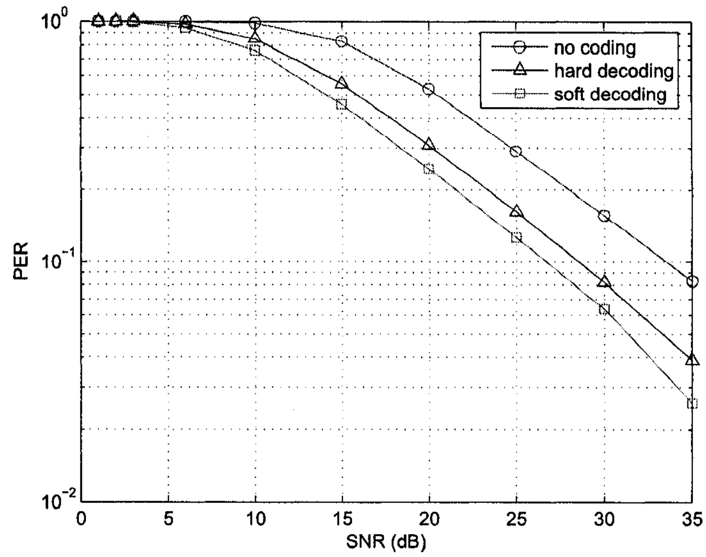


(a) BER vs. Velocity

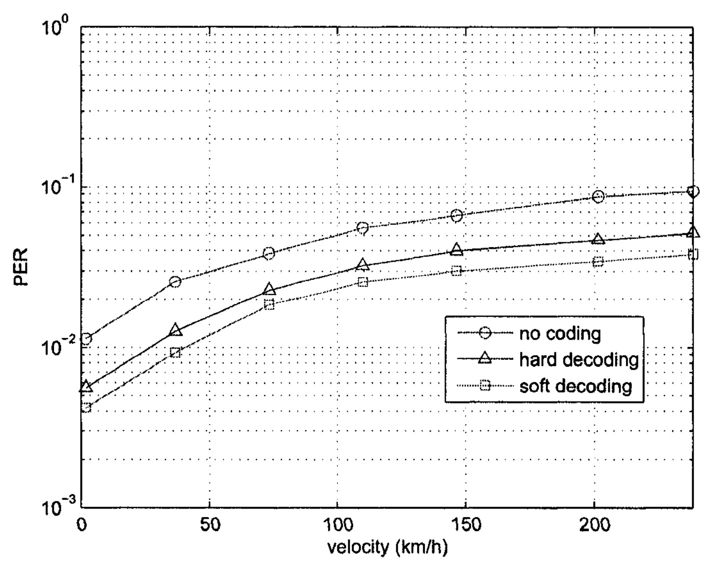


(b) PER vs. Velocity

Figure 3.5: Information size comparison against velocity at 30 dB.



(a) PER vs. SNR



(b) PER vs. Velocity

Figure 3.6: Coding comparison.

All of the above simulation results confirm that the assumption in Eq. (2.10) does not hold true, since the requirement in Eq. (3.1) is not met. These results clearly illustrate that if any two of the variables in Eq. (3.3) are fixed while increasing one of the variables, the assumption becomes less accurate, and hence increases the PER. Ideally, the performance of the system should be relatively independent of velocity and packet length within the simulated ranges discussed.

3.3 Concluding Remarks

Analysis and simulations to better understand the limitations of conventional channel estimation in high relative velocities have been carried out. It was found that, in scenarios with low delay spread, such as a wide open highway, IEEE 802.11a outperforms DSRC at high relative velocity. However, in scenarios with high delay spread, such as dense urban areas, IEEE 802.11a fails, and so extending the symbol duration is justified. The symbol extension also reduced the information data rate, so there is a trade-off between performance and speed. It may be possible to exploit this relationship by selecting IEEE 802.11a for specific DSRC applications that only involve low delay spread.

All the simulation results suggest that the conventional channel estimation scheme of IEEE 802.11a will not suffice in meeting the performance requirements for DSRC applications. Assuming static channel characteristics for DSRC applications is not feasible. Since the DSRC PHY standard is still in development, these simulation results can be used as a benchmark for future research. Future channel estimation schemes should attempt in making the performance more independent of the velocity and packet length.

Chapter 4

Proposed System Enhancements

A conventional DSRC system was previously presented and discussed in chapter 2. The previous chapter presented a feasibility study of the conventional model for DSRC applications. It was found that it is not feasible to assume static channel characteristics, rendering the assumption in Eq. (2.10) invalid. This assumption tends to only hold true in stationary environments (i.e. walking speeds) and is limited to the requirement in Eq. (3.1). However, Eq. (3.1) can not be met in highly mobile DSRC applications. Also, referring back to Fig. 3.1, it can be seen that the channel rapidly varies within the packet duration at vehicular velocities. In order to improve the performance of the DSRC system, it is necessary to track the rapid fluctuation of the channel response.

In this chapter, an effective adaptive channel estimation scheme is derived and tested to enhance the performance of the receiver. This is achieved by applying adaptive signal processing concepts to the system. The receiver design required modification in order to incorporate an accurate channel tracking technique. Section 4.1 provides a brief overview of adaptive filtering and examines the possible reference signals that may be used for channel tracking. Section 4.2 discusses the proposed receiver design based on the selected reference signal. The design modifications will be limited to the receiver alone, so that the system complies with the current IEEE 802.11p protocol. Also, by limiting the modifications to the receiver, the system will also comply with the USDoT's goals of achieving inter-operability [16] with other WLAN systems. The simulation results will illustrate the performance enhancements in Section 4.3. The proposed design is extensively tested against many possible scenarios.

4.1 Channel Tracking Schemes

After the initial channel estimate $\widehat{H}_{0,k}$ is obtained based on the training data, the channel estimator switches to Decision-Directed (DD) mode. Recall from Eq. (2.12), after training, the first received OFDM symbol is compensated by the initial channel estimate,

$$\widehat{Y}_{0,k} = \frac{Y_{0,k}}{\widehat{H}_{0,k}} \tag{4.1}$$

At this point, adaptive signal processing concepts can be considered for tracking the channel variation by updating the estimated channel response $\widehat{H}_{n,k}$, for $n = 1, 2, \cdots, N_p - 1$. An example of an adaptive equalizer is shown in Fig. 4.1. The adaptive algorithm determines the updated tap weights of the filter based on a error signal $e(n)$. The error signal is obtained from the difference between the compensated or equalized data $y(n)$ and a reference signal $d(n)$, which is commonly referred to as the "desired signal". Conventional adaptive equalizers invert the effects of the channel impulse response in the time domain, while the channel estimator inverts the effects of the channel frequency response in the frequency domain. The study in [8] compares channel equalization and channel estimation for OFDM WLAN systems. In that study, it was found that the channel estimator is simpler and performs better than the channel equalizer in fading channels.

In a DSRC system, the static channel assumption must be disregarded and channel variation must be tracked. In order to update the channel frequency response, the channel estimator requires extra information to represent the desired signal, and will be denoted as $\widehat{X}_{n,k}$. As in adaptive equalization, the desired signal will be used to obtain an error vector that can be used for updating the channel coefficients on a symbol-by-symbol basis.

The first step in redesigning the receiver is to identify the possible desired signals. The desired signal should try to accurately replicate the transmitted data $X_{n,k}$. Selecting the desired signal

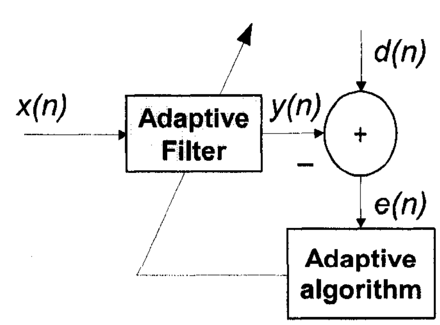


Figure 4.1: Components and configuration of an adaptive equalizer.

may be the most important task in adaptive filtering [14]. The appropriate desired signal usually depends on the application and requirements. There are three possible reference signals that can be considered for a DSRC receiver. The following subsections examine the benefits and disadvantages of each possible reference signal.

4.1.1 Pilot-Aided

After determining the channel response based on the pilot tones, the channel response at the data subcarriers can be obtained through different possible interpolation methods. Pilot-aided channel estimation for OFDM systems has been extensively studied in recent literature, including [12], [20], [27], and [38]. Pilot-aided algorithms are based on the pilot tone arrangement that are designed to exploit certain channel characteristics. There are many different possible pilot arrangements, the most common being vertical and horizontal "comb-type" pilots. Recall that the IEEE 802.11a/p PHY utilizes 4 horizontal comb-type pilot tones. The range of frequencies over which a channel can be considered flat is referred to as the *coherence bandwidth* B_c. According to [32], a conservative estimate of the coherence bandwidth would be,

$$B_c \approx \frac{1}{50\sigma_\tau} \tag{4.2}$$

Vehicular environments involve high delay spreads and require the pilot tone spacing to be at least $200\,KHz$. In the current standard, the pilot tones shown in Fig. 2.1 are spaced apart at $1.875\,MHz$. The number of pilot tones would have to be substantially increased in order to provide the required resolution for interpolation. However, increasing the number of pilot tones would reduce the spectral efficiency and data rate.

Other types of pilot arrangements have been considered for highly mobile OFDM systems. A pseudo-pilot scheme proposed in [37], improves the PER performance at high velocity. However, this scheme worsened the PER performance at low velocity when compared to conventional channel estimation. The proposed scheme in [27] yields similar results, improving performance at high velocity, while degrading performance at low velocity. Aside from the poor performance at low velocity, changing the pilot arrangement would require the protocol to be changed, which is undesirable for the aforementioned reasons. Therefore, pilot-aided channel estimation schemes are not suitable for the current DSRC standard.

4.1.2 Decision-Aided I (Symbol Demapper)

After signal compensation, the data is digitally demodulated with the demapping circuit. This produces hard binary data that can be once again modulated to produce the desired signal ($\widehat{X}_{n,k}$), which would then be compared to the received information $Y_{n,k}$. Decision-directed channel estimation for using the demodulator as the desired signal is covered extensively in [22], [28], [31] and [43]. All these base their research on IEEE 802.11a physical layer specifications. The authors in [31] had claimed that decision-directed channel estimation from the demapping circuit would improve performance with low-computational complexity. The study in [43] suggested a channel variation threshold, so that the channel is only updated if the threshold is exceeded. In [22], a novel quantized decision gradient algorithm is proposed, which was shown to outperform the normalized least mean square algorithm. In [28], it was shown that channel estimation was functionally equivalent to channel equalization, but less complex in terms of hardware implementation. This will be further explained in Section 4.2, where this concept will be applied.

Another benefit of decision-aided channel estimation is that it also incorporates the available pilot tones. This way, the desired signal contains both sliced and known data, which may help produce more reliable estimates of the channel response without reducing the spectral efficiency. This is a major advantage over pilot-aided schemes.

4.1.3 Decision-Aided II (Viterbi Decoder)

Forward error correction (FEC) coding can further mitigate errors. Since the Viterbi decoder will detect and correct errors that the demapping circuit can not, the Viterbi decoder output would produce more reliable decisions to produce a more reliable desired signal. Viterbi-Aided channel estimation has been discussed in [17], [18], [24] and [23]. Viterbi-aided channel estimation for standardized OFDM formats was first proposed in [17]. The same authors later proposed Viterbi-aided channel estimation for a developing Japanese standard called Dynamic Parameter Controlled Orthogonal Frequency and Time Division Multiple Access (DPC/OF-TDMA) [18]. Similarly to DSRC, this standard was established for the purpose of high-speed communications in highly mobile systems. However, this standard utilizes 768 frequency subcarriers, over ten times that of DSRC.

The serial binary output of the Viterbi decoder can be encoded, interleaved, and digitally modulated to be compared with the received information $Y_{n,k}$. All components of the decision feedback circuit are also available in the transmitter. Instead of adding hardware, resources may be shared between the transmitter and receiver in an effective manner. However, in terms of hardware com-

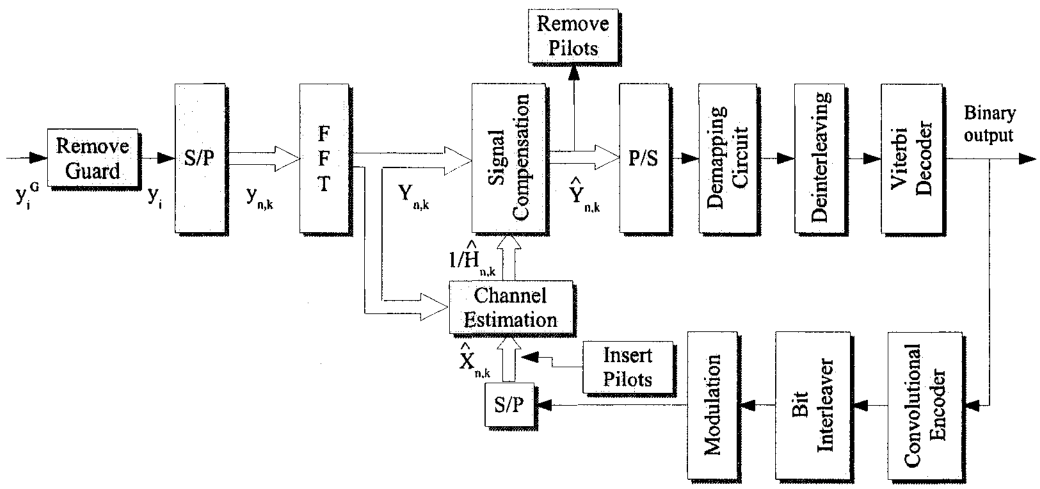


Figure 4.2: Proposed receiver design.

plexity, there would be an increase in the circuit critical time due to the extra required processing for deinterleaving, decoding, encoding, and interleaving. There is an apparent trade off between performance and complexity.

4.2 Proposed Receiver Design

The simulation results in the next section show that the use of hard-decisions from the Viterbi decoder (Decision-Aided II) proves to be more reliable than using the demapping circuit (Decision-Aided I) as the decision device. The serial binary data is not to be directly compared to the parallel data that is input to the channel estimator. In order to make an estimate, the decoded data must be encoded, interleaved, modulated, and finally converted to parallel as shown in Fig. 4.2. These decisions are used after the initial channel estimate based on the training symbols. Since all these components already exist in the transmitter, there is no need for additional hardware.

After training, the received symbol $Y_{0,k}$ is compensated in Eq. (4.1), and the desired symbol $\widehat{X}_{0,k}$ is produced from the feedback circuit. At this point, the channel estimator switches to "decision-directed mode". For the rest of the packet duration, the desired data $\widehat{X}_{n,k}$ is used to update the channel estimate symbol-by-symbol. Initially, a preliminary channel estimate denoted as $\widetilde{H}_{n,k}$ is obtained by the least squares (LS) method,

$$\widetilde{H}_{n,k} = \frac{Y_{n,k}}{\widehat{X}_{n,k}} \tag{4.3}$$

This is only a preliminary channel estimate based on the desired signal, and will later be used to update the actual channel estimate. Error propagation is one of the major problems that should be avoided when using decision-directed channel estimation. A wrong decision due to noise or channel fading can result in unreliable channel estimates. As a result, the signal compensation produces more errors in data that produce another wrong decision. This leads to the loss of the entire packet. Another issue to be considered is the *noise enhancement* caused by the error component in Eq. (2.8) and discussed in more detail in Section 5.1.1. In order to reduce the effects of noise enhancement and error propagation, a proposed step-size referred to as the "forgetting factor" (γ) is introduced into the following recursive channel update equation,

$$\widehat{H}_{n+1,k} = \widehat{H}_{n,k} + \gamma \cdot \triangle H_{n,k} \tag{4.4}$$

where $\triangle H_{n,k}$ is the channel estimation error,

$$\triangle H_{n,k} = \widetilde{H}_{n,k} - \widehat{H}_{n,k} \tag{4.5}$$

hence, 4.4 is written as,

$$\widehat{H}_{n+1,k} = (1-\gamma) \cdot \widehat{H}_{n,k} + \gamma \cdot \widetilde{H}_{n,k} \tag{4.6}$$

where $0 < \gamma < 1$ is a constant with an optimal value that depends on the time invariant property caused by velocity, SNR, and the modulation scheme. The forgetting factor is selected through many trials and tests in ranges of velocity and SNR suitable for DSRC applications. Based on these tests, the best overall forgetting factor is selected and presented in the simulation results illustrated in the following section. The channel estimator is illustrated as a block diagram in Fig. 4.3. First the preliminary channel estimate is obtained based on the desired signal, and then this is added to the current channel estimate. The delay block element with a delay of one symbol (T_s) shows that the new channel update is used to compensate the next received symbol. The concept of recursive channel updates in Eq. (4.4) is similar to that of the gradient algorithm used in adaptive filtering [28]. Let $w_{n,k}$ represent the coefficient vector used for signal compensation at symbol index n. The next coefficient vector $w_{n+1,k}$ is computed as follows,

$$\begin{aligned} w_{n+1,k} &= \widehat{H}^{-1}_{n+1,k} \\ &= \frac{1}{(1-\gamma) \cdot \widehat{H}_{n,k} + \gamma \cdot \widetilde{H}_{n,k}} \\ &= \frac{\widehat{H}^{-1}_{n,k}}{(1-\gamma) + \gamma \cdot \left[\widehat{H}^{-1}_{n,k} \cdot (Y_{n,k}/\widehat{X}_{n,k})\right]} \\ &= \frac{w_{n,k}}{1-\gamma \cdot \left[1 - w_{n,k} \cdot (Y_{n,k}/\widehat{X}_{n,k})\right]} \end{aligned} \tag{4.7}$$

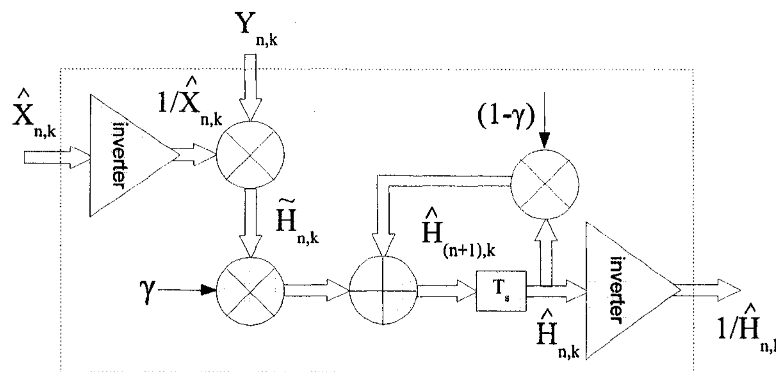


Figure 4.3: Decision-Directed Channel Estimator

Since $w_{n,k}$ approaches the inverse of $\widetilde{H}_{n,k}$, the approximation, $\frac{1}{1-x} \approx 1 + x$ (as x approaches 0) can be used to further simplify (4.7) as follows.

$$\begin{aligned} w_{n+1,k} &\approx w_{n,k} \cdot \left[1 + \gamma \cdot \left[1 - w_{n,k} \cdot (Y_{n,k}/\widehat{X}_{n,k})\right]\right] \\ &= w_{n,k} + \gamma \cdot (\widehat{X}_n - w_{n,k} \cdot Y_{n,k}) \cdot \frac{w_{n,k}}{\widehat{X}_{n,k}} \\ &= w_{n,k} + \gamma \cdot (\widehat{X}_{n,k} - \widehat{Y}_{n,k}) \cdot \frac{1}{Y_{n,k}} \\ &= w_{n,k} + \mu_{n,k} \cdot e_{n,k} \cdot Y^*_{n,k} \end{aligned} \tag{4.8}$$

where * denotes complex conjugate operation. The coefficient update equation in (4.8) is equivalent to the least mean square (LMS) algorithm [14], where e_n is the error vector (i.e. $e_{n,k} = \widehat{X}_{n,k} - \widehat{Y}_{n,k}$) and $\mu_{n,k}$ is the step size vector (i.e. $\mu_{n,k} = \gamma/|Y_{n,k}|^2$). To reduce the complexity, the step size vector may be replace by a constant scalar value μ, where $\mu = \gamma/E\left[Y^2_{n,k}\right]$, where $E\left[\cdot\right]$ is the expectation operation.

4.3 Simulation Results

Since the conventional DSRC system is not feasible for WAVE, the goal of this study is to propose modifications to the receiver that enables the system to achieve acceptable performance under high velocities, high data rates, and large packet lengths. The modified DSRC system was simulated using Matlab v7.0.1 (R14). In each simulation, the PER was determined based on the transmission of 10,000 packets. Several simulation parameters were carefully considered to ensure all possible situations are tested in WAVE. The systems were simulated with varying SNR, velocities, packet

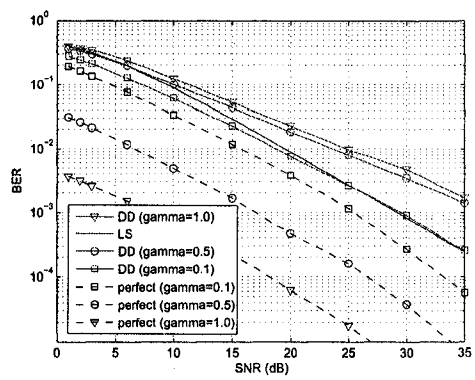
Table 4.1: Simulation parameters and values

NAME	ABBREV.	UNIT	VALUES
Signal-to-Noise Ratio	SNR	dB	[1, 2, 3, 6, 10, 15, 20, 25, 30, 35]
Maximum Doppler Freq. (Velocity)	f_m (v)	Hz (km/h)	[10, 200, 400, 600, 800, 1100, 1300] (2, 37, 74, 110, 147, 202, 238)
Symbol Duration	T_s	μs	[8]
Packet Length	N_p	symbols/packet	[10, 64, 100, 250, 325]
Rice Factor	K	-	[0, 1, 3]
Modulation	m	-	[QPSK, 16-QAM, 64-QAM]
Decoding	-	-	[hard]

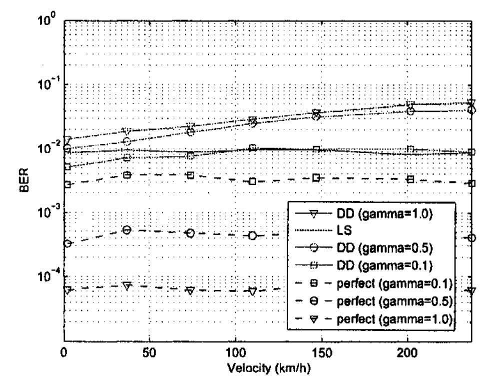
lengths, Rice factors, and modulation schemes (i.e. data rate). The focus was on the effects of envelope fading, since the delay spread issue was resolved by the extension of the guard interval, as shown in [42]. Table 5.1 provides of a summary of the range of values simulated for each variable. The primary focus of the previous chapter was to test the performance under the worst-case scenario, a fading channel with no LOS (i.e., Rayleigh fading). However, scenarios with strong specular components commonly occur, especially in situations involving vehicles traveling on highways [13]. Accordingly, simulations under Rician fading channels are also carried out in this chapter.

4.3.1 Selecting the Forgetting Factor

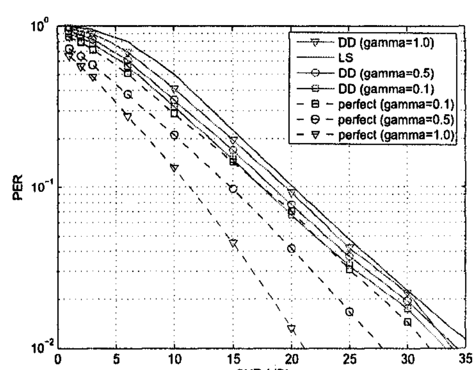
Extensive trials were carried out to select a forgetting factor that yields the best overall PER performance (see Figs. 4.4 - 4.6). All the following simulation results are based on a DSRC system that employs Viterbi-aided (Decision-Aided II) channel estimation. The forgetting factor for QPSK, 16-QAM, and 64-QAM were selected to be 0.1, 0.4, and 0.5 respectively. These forgetting factors don't necessarily yield the best BER, but they would achieve the best overall PER over the range of velocity and SNR discussed. Note that when the forgetting factors are decreased, the performance is improved at higher velocities, while it may slightly degrade at lower velocities. In reality increasing γ may also increase the sensitivity to noise. However, an excessive reduction of γ would render the channel estimator incapable of tracking the fast fading channels. Notice that in the case of perfect decisions, increasing γ improves the performance at higher velocities.



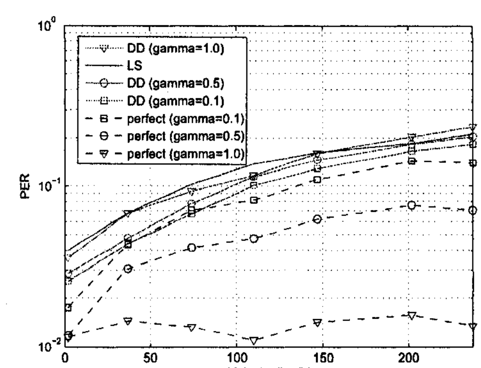
(a) BER vs. SNR at 74 km/h



(b) BER vs. Velocity at 20 dB

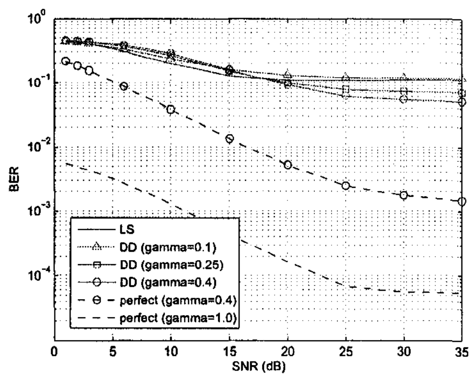


(c) PER vs. SNR at 74 km/h

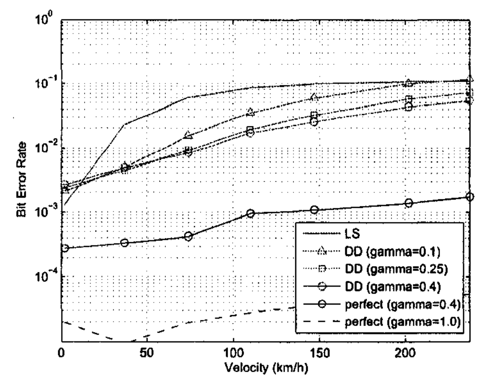


(d) PER vs. Velocity at 20 dB

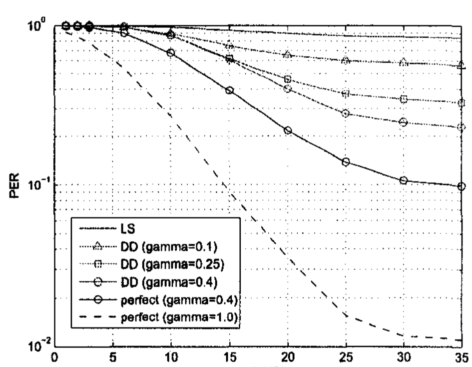
Figure 4.4: Selecting the forgetting factor for QPSK.



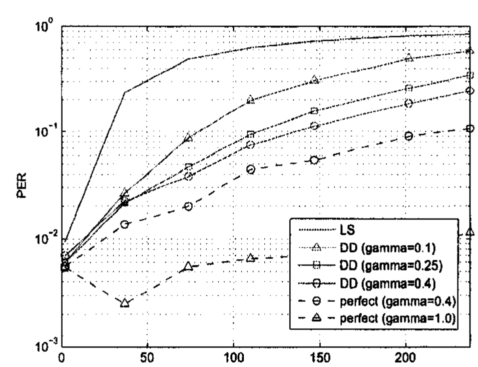
(a) BER vs. SNR at 238 km/h



(b) BER vs. Velocity at 30 dB

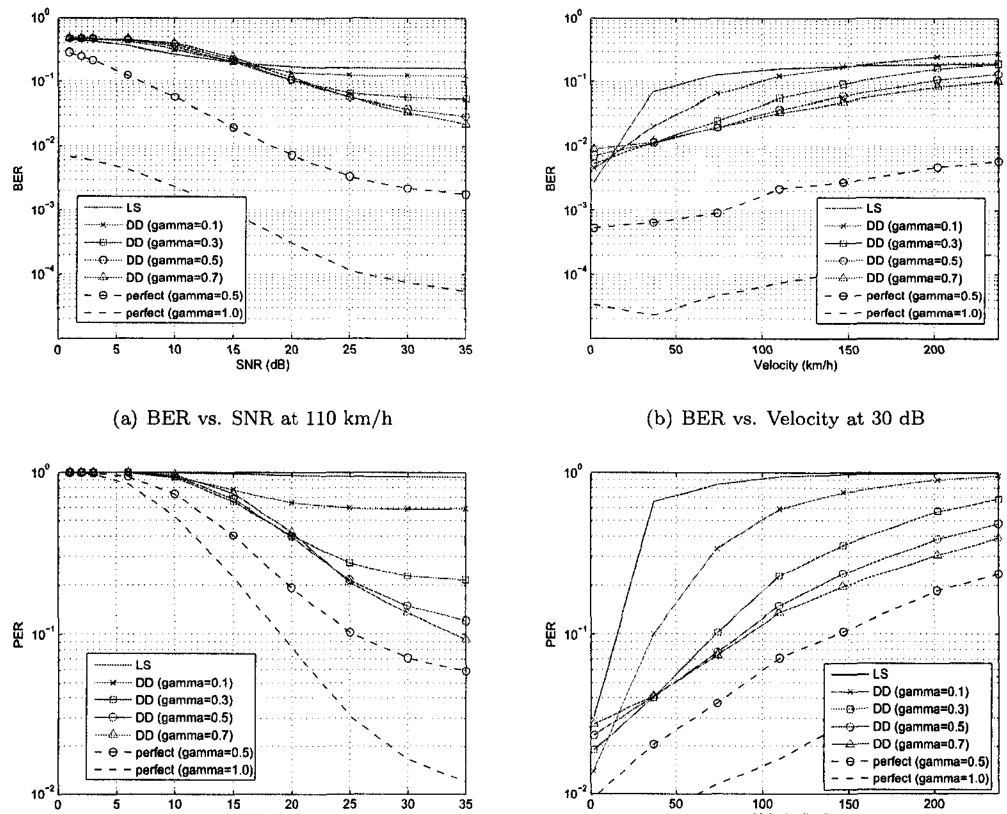


(c) PER vs. SNR at 238 km/h



(d) PER vs. Velocity at 30 dB

Figure 4.5: Selecting the forgetting factor for 16-QAM.



(a) BER vs. SNR at 110 km/h

(b) BER vs. Velocity at 30 dB

(c) PER vs. SNR at 110 km/h

(d) PER vs. Velocity at 30 dB

Figure 4.6: Selecting the forgetting factor for 64-QAM.

4.3.2 Selecting the Desired Signal

The PER performance curves at varying velocities for 16-QAM DSRC are shown in Fig. 4.8. The decision-aided channel estimation schemes are based Eq. (4.4), where the step size (γ) is referred to as the forgetting factor. For 16-QAM, the forgetting factor was selected to be $\gamma = 0.4$ based on many simulated trial runs illustrated in the previous section. It can be seen that both of the decision-aided schemes substantially improve the performance, with the Viterbi-aided scheme having a slightly added improvement of approximately 2 dB gain. The "perfect decision" curves represent the performance when the desired signal is equal to the transmitted data $\widehat{X}_n = X_n$. The "perfect decision" curves show the amount of improvement available if decisions are made more reliable.

In the case of QPSK, the decision-aided schemes only slightly improve performance, while PER performance is substantially enhanced in the case of 64-QAM. In both cases, the Decision-Aided II (Viterbi-aided) scheme provides very little improvement when compared to Decision-Aided I scheme (see Figs. 4.7 and 4.9). Therefore, selecting the Decision-Aided I scheme may suffice in the cases of QPSK and 64-QAM. This may be due to the simplicity of the QPSK constellation and the complexity of the 64-QAM constellation. Also, these results only include hard decoding, it is possible that soft decoding may further improve performance. Regardless, Viterbi-aided channel estimation still provides an overall improvement to the DSRC system, and so the Viterbi decoder was the selected source for the desired signal.

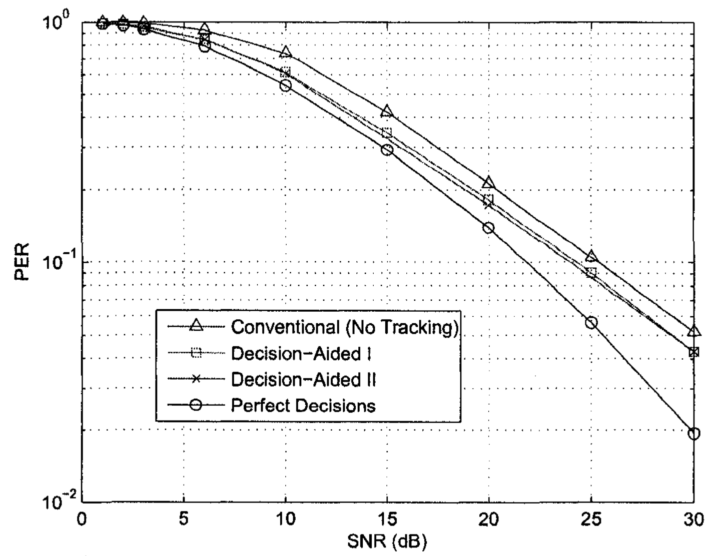
4.3.3 Proposed Design vs. Conventional Design

In this section, a thorough comparison between the conventional and the proposed design are illustrated in the following BER and PER plots. A plethora of results were produced, however, only a few major plots are presented to best illustrate the enhancements made by the proposed design.

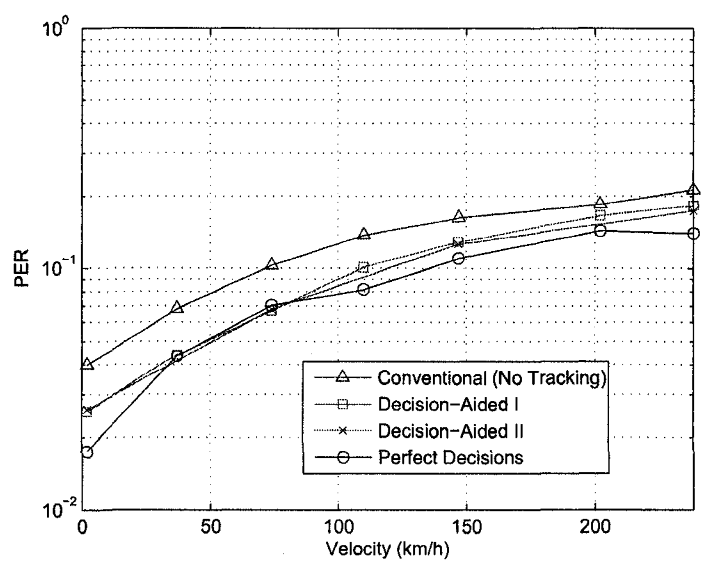
4.3.3.1 Modulation (Data Rate) Comparison

The BER and PER curves under Rayleigh fading for varying modulation schemes with a fixed packet length of $N_p = 64$ are shown in Figs 4.10 and 4.11. The plots in Fig. 4.10are error curves versus SNR at a fixed velocity of 147 km/h. The plots in Fig. 4.11 are error curves versus velocity at a fixed SNR of 30 dB. The BER curves seem to have similar relationships to the PER curves, with the exception of QPSK. Recall that the true performance metric is the PER, so all this shows is that there is a more bit errors in erroneous packets.

Fig. 4.10(b) shows the PER performance at high velocity against SNR. The dashed lines represent

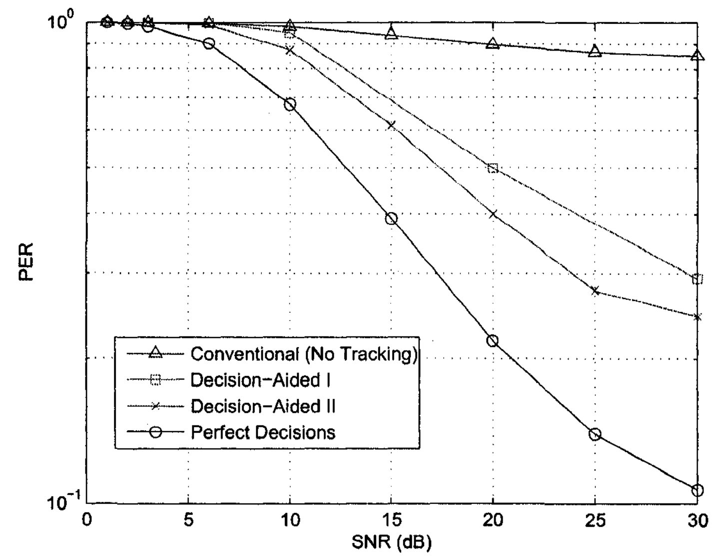


(a) QPSK: PER vs. SNR at 238 km/h

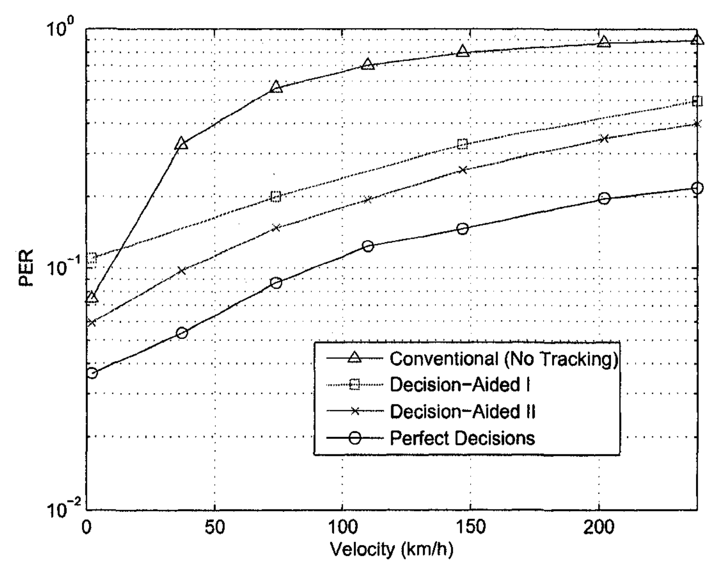


(b) QPSK PER vs. Velocity at 20 dB

Figure 4.7: Desired signal comparison for QPSK.

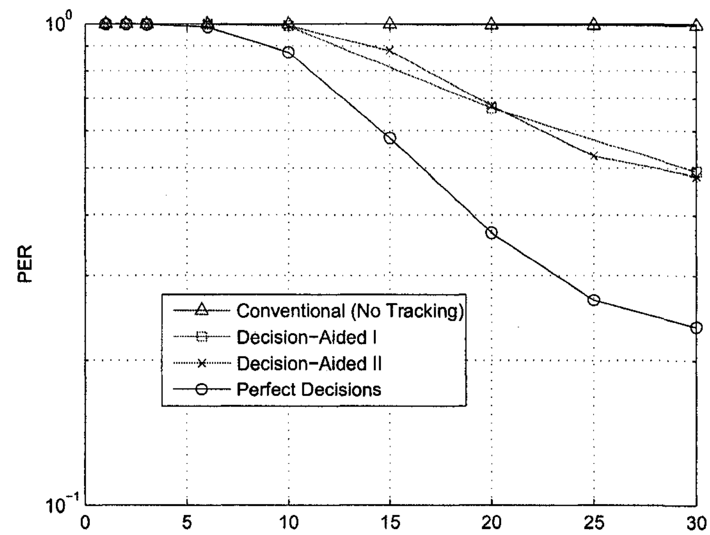


(a) 16-QAM: PER vs. SNR at 238 km/h

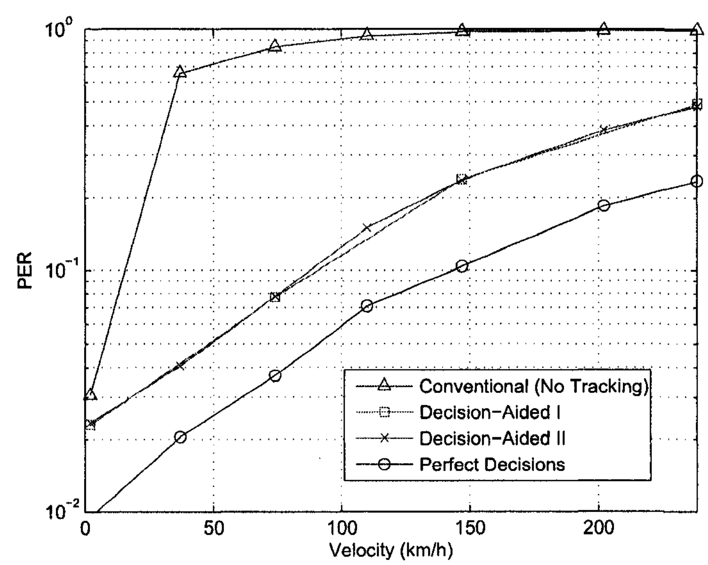


(b) 16-QAM: PER vs. Velocity at 20 dB

Figure 4.8: Desired signal comparison for 16-QAM.

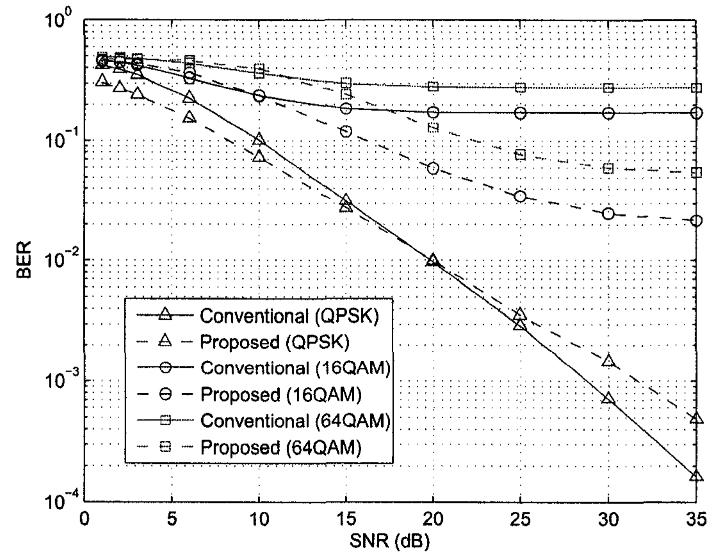


(a) 64-QAM: PER vs. SNR at 238 km/h

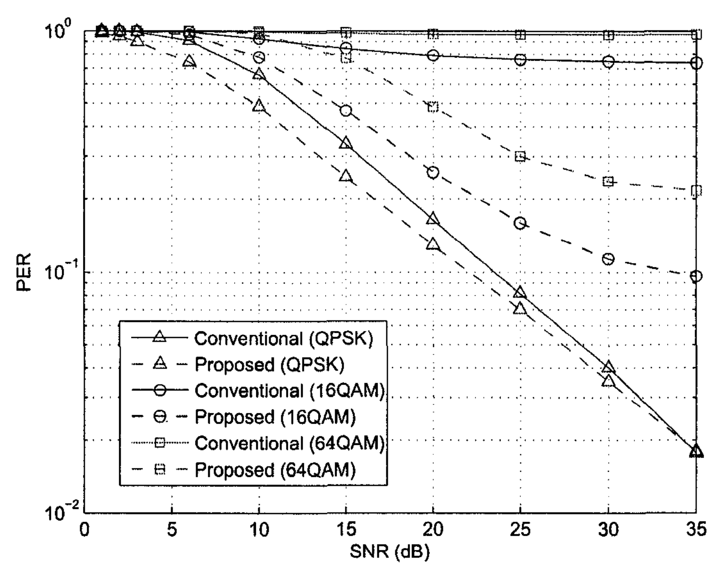


(b) 64-QAM: PER vs. Velocity at 30 dB

Figure 4.9: Desired signal comparison for 64-QAM.

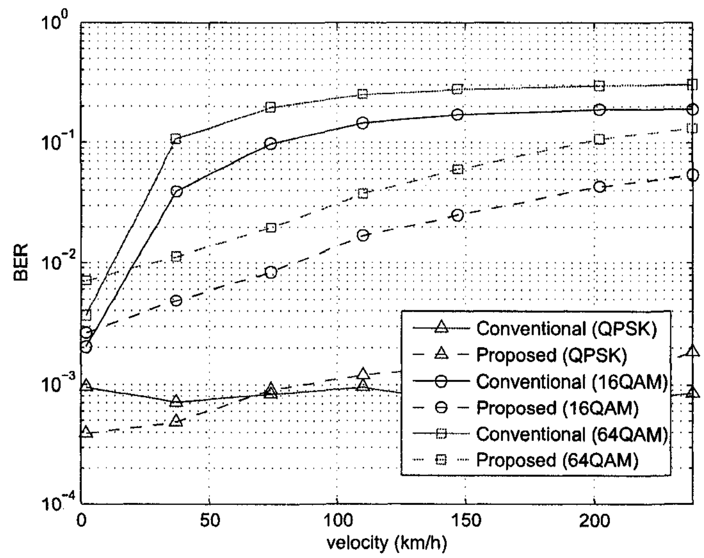


(a) BER vs. SNR

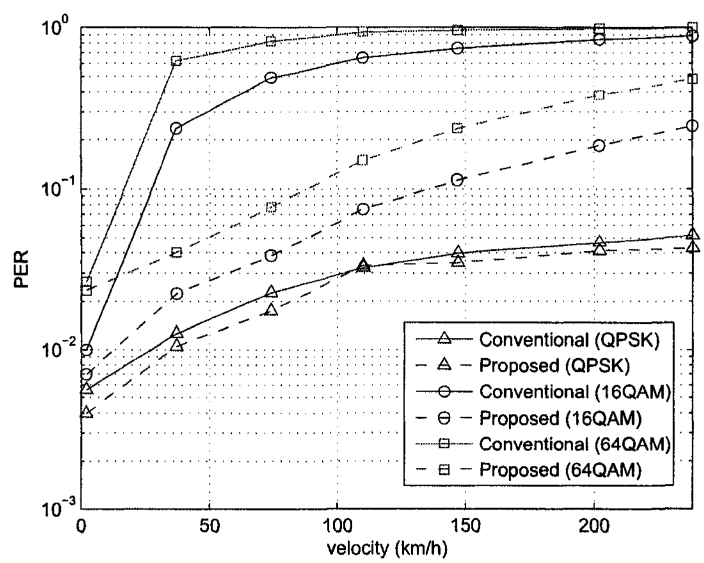


(b) PER vs. SNR

Figure 4.10: Proposed vs. Conventional against SNR under Rayleigh fading at 147 km/h.



(a) BER vs. Velocity



(b) PER vs. Velocity

Figure 4.11: Proposed vs. Conventional against velocity under Rayleigh fading at 30 dB.

the performance of the proposed scheme while the solid lines represent the conventional scheme. The proposed scheme yields a slight improvement at low SNR for QPSK, while it yields substantial improvements at high SNR for 16-QAM and 64-QAM. In the cases of 16- and 64-QAM, the proposed scheme has reduced PER by approximately a factor of 10 compared to the conventional scheme at a SNR of 35 dB.

Fig. 4.11(b) shows the PER performance against varying velocity. It can be seen that the proposed scheme only yields a slight improvement with QPSK, while there is substantial improvement with 16-QAM and 64-QAM, particularly at medium velocities. At approximately 75 km/h, the proposed scheme reduces the PER by more than a factor of 10 compared to the conventional scheme for 16- and 64-QAM transmission. At medium velocities, the performance of 16-QAM is near that of QPSK. This means that higher data rates can be achieved with out a substantial loss in performance.

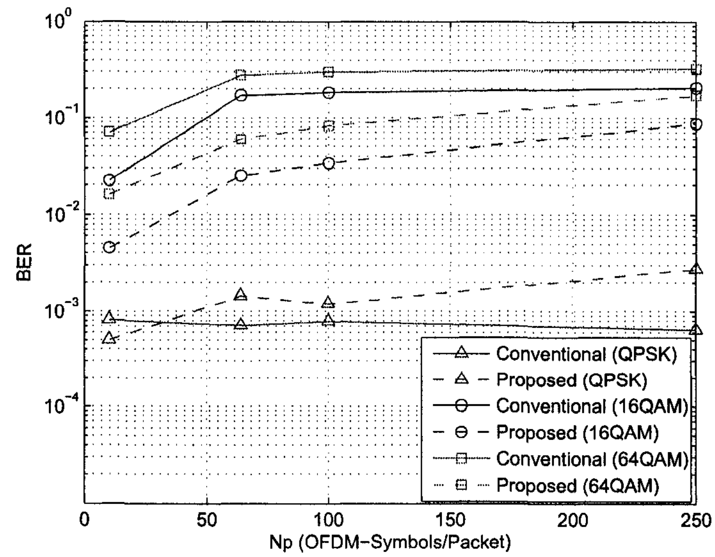
4.3.3.2 Packet Length Comparison

Fig. 4.12 presents the BER and PER performance versus the information size in terms of OFDM symbols per packet. The velocity is fixed at 147 km/h and the SNR is fixed at 30 dB. Again the BER curves have similar relationships when compared to the PER curves with the exception of QPSK. In the case of QPSK, it seems that the bit errors disperse differently between the conventional and proposed schemes. This means that while the packet error rate is reduced, the erroneous packets have a higher concentration of bit errors.

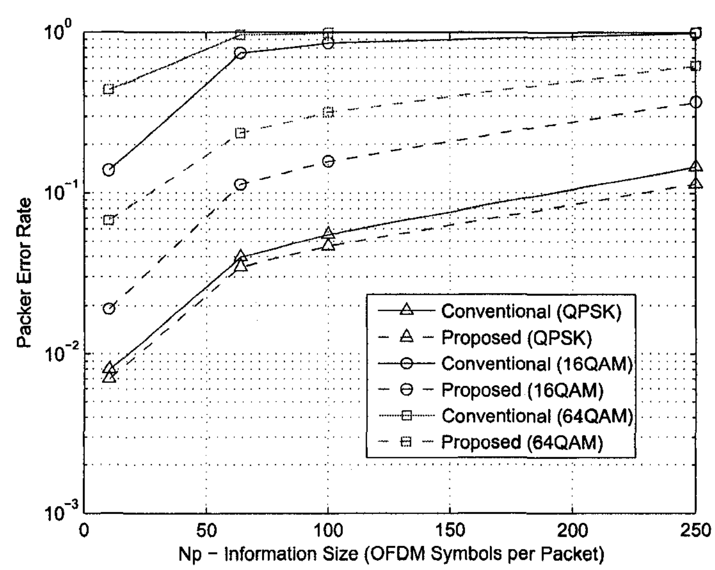
The proposed channel estimation scheme enables the use of larger packet lengths. Again, there is only a slight improvement for QPSK, while there are large improvements for 16-QAM and 64-QAM. For 16- and 64-QAM, at a packet length of 64, the PER in the proposed scheme is again reduced by almost a factor of 10 compared to the conventional scheme.

4.3.3.3 Rician Fading Comparison

Finally, Fig. 4.13 illustrates the performance of both conventional and proposed scheme under Rician fading with a Rice factor $K = 1$ (50% LOS) at fixed velocity of 238 km/h. The performance gap between the proposed and conventional schemes has dramatically increased. The improvement is now more apparent in the case of QPSK. For 16-QAM and 64-QAM, the proposed scheme achieves much lower error rates. For example, in the case of 64-QAM at 25 dB, conventional channel estimation achieves close to 100% error, while the proposed scheme reaches a PER of 10^{-3}. This shows that the proposed scheme achieves PER performance beyond the reach of the conventional scheme. Simulations under Rician fading with Rice factor of $K = 3$ (75% LOS) were also carried out. The

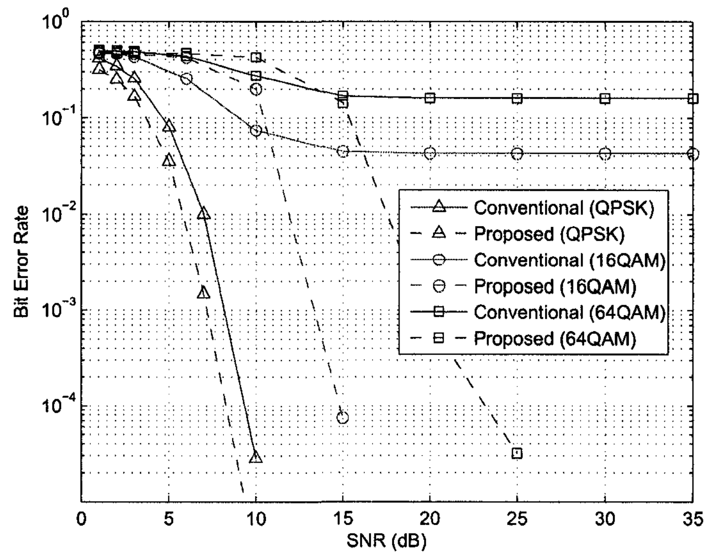


(a) BER vs. Packet Length

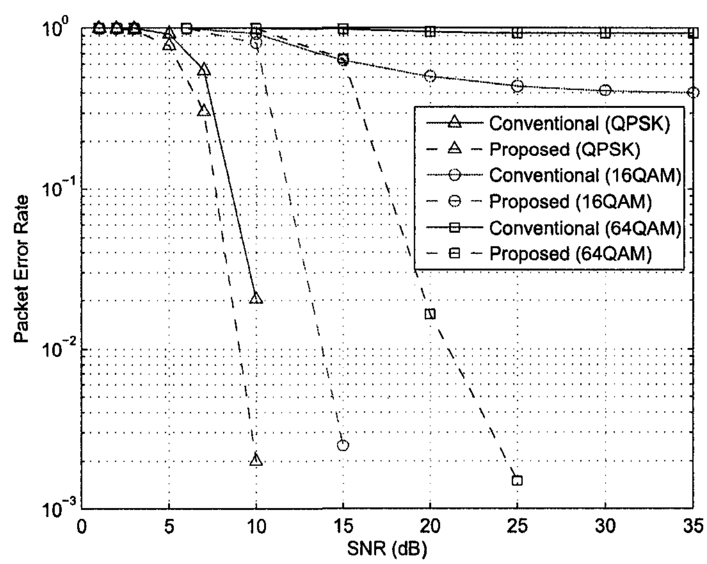


(b) PER vs. Packet Length

Figure 4.12: Proposed vs. Conventional against packet length under Rayleigh fading.



(a) BER vs. SNR



(b) PER vs. SNR

Figure 4.13: Proposed vs. Conventional under Rician fading with 50% Line-Of-Sight.

performance gap was greater, however could not be properly plotted since the computation of the packet error rate is based on the transmission of only 10,000 packets.

4.4 Concluding Remarks

Several channel tracking techniques based on different reference signals were considered for 5.9 GHz DSRC applications. Past literature has shown that pilot-aided channel estimation can improve performance at high velocity, however it is not possible to obtain an accurate channel estimate at the current subcarrier spacing. A much higher number of fixed pilot subcarriers would be needed. Other pilot arrangement schemes have been shown to improve performance at high velocities, however, they degrade performance at low velocities. Also, pilot-aided channel estimation requires modifications to be made to the transmitter and protocol. Decision-aided schemes, which also happen to include the available pilot tones, have been shown to improve PER performance at both low and high velocities. Using the Viterbi decoder output to produce the desired signal has resulted in additional enhancement, particularly in the case of 16-QAM modulation.

The 5.9 GHz DSRC receiver has been redesigned to incorporate Viterbi-aided channel estimation. The proposed receiver is designed with a feedback transmitter to the channel estimator. The proposed design was compared to the conventional design under Rayleigh and Rician fading channels. The proposed channel estimation scheme improved PER performance of the receiver for all modulation schemes, namely QPSK, 16-QAM and 64-QAM. The proposed channel estimator improved 16-QAM and 64-QAM performance 10 fold in Rayleigh fading scenarios. When the Viterbi-aided channel estimator was used in Rician fading channel, i.e., when a LOS component exists, it improved the performance considerably. The proposed scheme achieved performance that was not at all possible with conventional estimator. The proposed design was proven to be very effective and practical for WAVE applications, since it did not require any additional hardware.

Chapter 5

Additional Proposed Enhancements: Second Order Channel Estimation

In this chapter, a novel second-order algorithm is proposed for improving the packet performance of DSRC systems at high velocities. Adaptive signal processing concepts are further explored in order to derive this algorithm. This algorithm is compared to the first order scheme discussed in previous chapters. Since Viterbi-aided channel estimation proved to be the superior scheme, in this chapter the desired signal is obtained from the Viterbi decoder output.

The proposition of second-order channel estimation was driven by the presumption that a fast fading channel can be better represented by a non-linear function. Second-order algorithms utilize additional information to further improve the accuracy of the tap weight estimates. However, achieving higher accuracy usually comes at a cost of higher complexity, which at the hardware level translates to either higher area or critical time.

A second-order algorithm for channel estimation is derived in Section 5.1. The effects of noise enhancements is also discussed in Section 5.1.1. The performance of this algorithm will be illustrated in the simulation results in Section 5.2.

5.1 Second-Order Channel Estimation: Iterative Compensation

As in first-order channel estimation, the initial channel estimate is obtained from the preamble as in Eq. (2.9). This algorithm is derived based on the original recursive function in Eq. (4.6), which is also restated in the following equation,

$$\widehat{H}_{n+1,k} = (1-\gamma)\cdot\widehat{H}_{n,k} + \gamma\cdot\widetilde{H}_{n,k} \tag{5.1}$$

Recall that $\widetilde{H}_{n,k}$ is the channel response based on a least squares estimate in Eq. (4.3). The first-order algorithm is reviewed and scrutinized as it will be used as the base algorithm in deriving the proposed second-order algorithm.

In the original function, the received data, $Y_{n,k}$, is compensated by the last channel estimate, $\widehat{H}_{n,k}$. After compensation the data is demapped and decoded to then produce the desired signal, $\widehat{X}_{n,k}$, by encoding and mapping again. The current desired signal is used to estimate the current channel response $\widetilde{H}_{n,k}$, which is then used to calculate the channel estimation error, $\triangle H_{n,k}$. The updated estimated channel response, $\widehat{H}_{n+1,k}$, is used to compensate the next received data, $Y_{n+1,k}$. Recall that at high velocities the channel may exhibit fast fading characteristics, which means that there may be a substantial change in the channel response during the symbol duration. Therefore, by the time the next symbol is received the estimated channel response may be slightly outdated.

The proposed solution to this problem is to compensate the received symbol a second time with the next channel estimate. So, the received symbol would first be compensated by an estimated channel response based on the previous desired signal, and then the entire process is iterated with the current desired signal. Such iterative schemes like *Turbo Decoding* are known to be very effective in improving performance of communication systems [7]. In that *iterative compensation*, the knowledge on estimated data symbol feedback one more time to improve previous channel estimation, hence improve the overall system performance. The main purpose of iterative compensation is for the channel estimator to "catch up" to the rapid channel variations.

The algorithm is derived as follows. First, the received data, $Y_{n,k}$ is received and compensated by $\widehat{H}_{n,k}$ to be demapped, deinterleaved, and decoded. The Viterbi decoder will reduce errors due to an outdated channel estimate and produce the desired data, $\widehat{X}^{(0)}_{n,k}$. This desired data is used to estimate the preliminary channel estimate $\widetilde{H}^{(0)}_{n,k}$, using the least square method in the following relation,

$$\widetilde{H}^{(0)}_{n,k} = \frac{Y_{n,k}}{\widehat{X}^{(0)}_{n,k}} \tag{5.2}$$

The next step is to obtain a first channel estimate using the following relation,

$$\widehat{H}_{n,k}^{(0)} = (1-\gamma) \cdot \widehat{H}_{n,k} + \gamma \cdot \widetilde{H}_{n,k}^{(0)} \tag{5.3}$$

where $0 < \gamma < 1$ is the forgetting factor which is selected based on simulation practice. At this point, the received symbol $Y_{n,k}$ is compensated a second time with the current channel estimate, $\widehat{H}_{n,k}^{(0)}$. After decoding the compensated data, a new desired signal is produced, $\widehat{X}_{n,k}^{(1)}$. Again, the new desired data is used to estimate the next preliminary channel estimate,

$$\widetilde{H}_{n,k}^{(1)} = \frac{Y_{n,k}}{\widehat{X}_{n,k}^{(1)}} \tag{5.4}$$

The channel response is finally estimated based on the recursive function in Eq. (5.1) using the new estimates,

$$\widehat{H}_{n,k}^{(1)} = (1-\gamma) \cdot \widehat{H}_{n,k}^{(0)} + \gamma \cdot \widetilde{H}_{n,k}^{(1)} \tag{5.5}$$

where $\widehat{H}_{n,k}^{(0)}$ is obtained from Eq. (5.3). The recursive channel update is then simplified as follows,

$$\begin{aligned}
\widehat{H}_{n,k}^{(1)} &= (1-\gamma) \cdot \left[(1-\gamma) \cdot \widehat{H}_{n,k} + \gamma \cdot \widetilde{H}_{n,k}^{(0)}\right] + \gamma \cdot \widetilde{H}_{n,k}^{(1)} \\
&= (1 - 2\cdot\gamma + \gamma^2) \cdot \widehat{H}_{n,k} + (\gamma - \gamma^2) \cdot \widetilde{H}_{n,k}^{(0)} + \gamma \cdot \widetilde{H}_{n,k}^{(1)} \\
&= \gamma^2 \cdot (\widehat{H}_{n,k} - \widetilde{H}_{n,k}^{(0)}) + \gamma \cdot (\widetilde{H}_{n,k}^{(1)} + \widetilde{H}_{n,k}^{(0)} - 2 \cdot \widehat{H}_{n,k}) + \widehat{H}_{n,k}
\end{aligned} \tag{5.6}$$

Finally, the second channel estimate is used at the next symbol index,

$$\widehat{H}_{n+1,k} = \widehat{H}_{n,k}^{(1)} \tag{5.7}$$

where $\widehat{H}_{n+1,k}$ is used to compensate the next symbol $Y_{n+1,k}$, which then will produce a new desired signal $\widehat{X}_{n+1,k}^{(0)}$ and so on. This continues until the last symbol of the packet. The pseudo code in Alg. (1) illustrates the entire procedure.

In summary, the received symbol is first compensated by a channel estimate produced by the last desired signal. After compensation, a new desired signal is produced, which updates the channel estimate. The same received symbol is then compensated a second time by this channel estimate. In the case of fast fading channels, the second compensation is crucial since there may be a substantial change in the channel response. As a result, the second compensation will yield in lower error at high velocities.

5.1.1 Considering Noise Enhancement

An important factor to consider when designing channel estimation and equalization algorithms is the effect of *noise enhancement*. Recall from Eq. (2.8) that the channel estimate has an error

Algorithm 1 Iterative Turbo Compensation (ITC)

procedure ITC($X_{train,k}$, $Y_{n,k}$, γ)
  Compute $\widehat{H}_{0,k}$ ▷ this is the initial channel estimate based on preamble
  Compute $\widehat{Y}_{0,k}$ ▷ the first received symbol $Y_{0,k}$ is compensated
  Generate $\widehat{X}_{1,k}$ ▷ produced by the decision feedback
  Select γ ▷ the forgetting factor
  $\widehat{H}_{1,k}=\widehat{H}_{0,k}$
  for $n = 1 : (N_p - 1)$ **do** ▷ switch channel estimator to "DD" mode
    for $i = 0 : 1$ **do**
      Compute $\widetilde{H}^{(i)}_{n,k}$ ▷ this is a preliminary channel estimate based on the desired signal
      Compute $\widehat{H}^{(i)}_{n,k}$ ▷ update the channel coefficients
      Compute $\widehat{Y}^{(i)}_{n,k}$ ▷ The received symbol $Y_{n,k}$ is compensated
      Generate $\widehat{X}^{(i)}_{n,k}$ ▷ a new desired symbol is generated from the decision feedback
    end for
    $\widehat{Y}_{n,k}=\widehat{Y}^{(1)}_{n,k}$
    $\widehat{H}_{n+1,k}=\widehat{H}^{(1)}_{n,k}$
  end for ▷ End of packet
end procedure

component due to noise.

$$\widehat{H}_{n,k} = H_{n,k} - \psi_{n,k} \tag{5.8}$$

During compensation, this error component will have a multiplicative effect on the received symbol. During training, the initial channel estimate in Eq. (2.9) is averaged across two identical training symbols to reduce sensitivity to noise,

$$\begin{aligned}
\widehat{H}_{0,k} &= \frac{Y_{-2,k} + Y_{-1,k}}{2 \cdot X_{train,k}} \\
&= \frac{2 \cdot H_{0,k} \cdot X_{train,k} + W_{-2,k} + W_{-1,k}}{2 \cdot X_{train,k}} \\
&= H_{0,k} + \frac{W_{-2,k} + W_{-1,k}}{2 \cdot X_{train,k}} \\
&= H_{0,k} + \frac{W_{avg,k}}{X_{train,k}}
\end{aligned} \tag{5.9}$$

where $W_{avg,k}$ is the average noise power. Since AWGN is a zero mean Gaussian process, the average noise power over N symbols would approach zero,

$$\sum_{0}^{N} \frac{W_i}{N} = 0 \text{ (as N} \rightarrow \infty) \tag{5.10}$$

Therefore, it is likely that taking the average channel estimate across two symbols would reduce the error component.

Noise enhancement must also be considered during the channel tracking phase. Recall the recursive channel update equation in Eq. (4.6), the channel estimation error component can be attenuated by reducing the forgetting factor. However, a small forgetting factor limits the channel updates to small variations in the fading envelope. Since the second-order method updates the channel estimate twice for every OFDM symbol, it has the capability of tracking steeper variations in the fading envelope while maintaining the same forgetting factor.

5.2 Simulation Results

The second-order channel estimation method was simulated in a DSRC system using Matlab. In each simulation, the PER was determined based on the transmission of 10,000 packets under varying SNR and velocities. Note that the PER performance curves for second-order channel estimation may have a lower resolution of simulated points. However, enough points are simulated to reasonably observe the performance trend. All simulations were under Rayleigh fading and had a fixed packet length of $N_p = 64$. The forgetting factors are selected for each modulation scheme based on the results of the previous chapter.

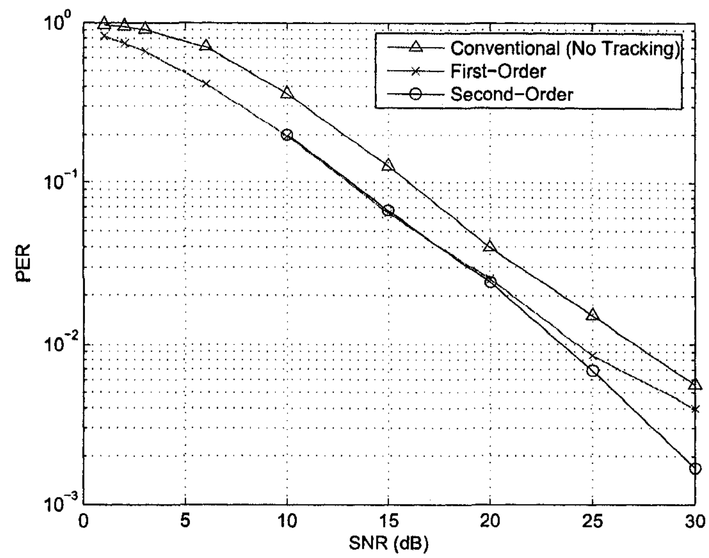
Fig. 5.1 illustrates the PER performance for QPSK modulation. Fig. 5.1(b) shows the PER performance against varying velocity at fixed SNR of 30 dB. Though no improvement is made at high velocities, the second-order method shows quite an improvement at low velocities. Fig. 5.1(a) shows the PER versus the SNR at fixed velocity of 2 km/h. However, it can be seen that the second-order algorithm outperforms the first-order algorithm at high SNR and low velocity.

Fig. 5.2 illustrates the PER performance for 16-QAM modulation. Fig. 5.2(a) shows the PER against varying SNR at fixed velocity of 238 km/h. Iterative-compensation outperforms the first-order channel estimation method at SNR greater than 20 dB. Fig. 5.2(b) plots the PER versus velocity at a fixed SNR of 30 dB. At relative velocities greater than 75 km/h, iterative compensation begins to outperform the first-order method. This was the goal when designing the algorithm. Notice that the added performance enhancement increases with velocity.

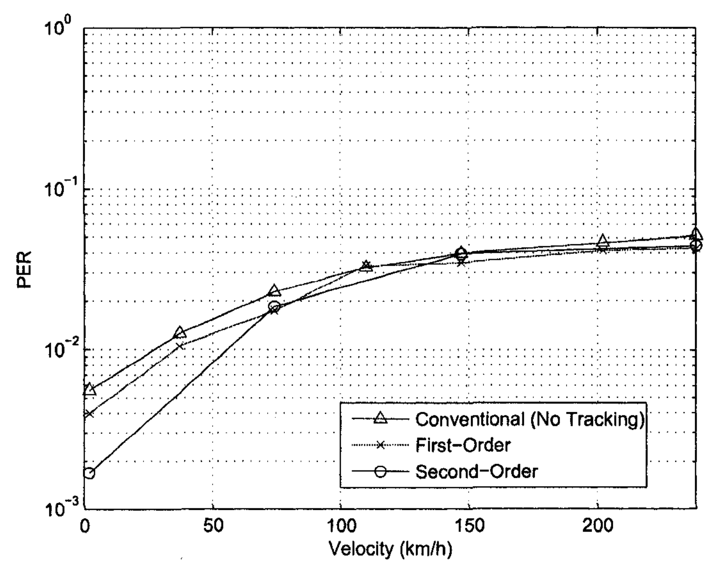
Table 5.1: Simulation parameters and values

NAME	ABBREV.	UNIT	VALUES
Signal-to-Noise Ratio	SNR	dB	[10, 15, 20, 25, 30, 35]
Maximum Doppler Freq. (Velocity)	f_m (v)	Hz (km/h)	[10, 400, 800, 1300] (2, 74, 147, 238)
Symbol Duration	T_s	μs	[8]
Packet Length	N_p	symbols/packet	[64]
Rice Factor	K	-	[0]
Modulation	m	-	[QPSK, 16-QAM, 64-QAM]
Decoding	-	-	[hard]

Fig. 5.3 illustrates the PER performance for 64-QAM modulation. In Fig. 5.3(a) the velocity is fixed at 238 km/h. In Fig. 5.3(b) the SNR is fixed at 30 dB. Again, the second-order method outperforms the first-order method at high velocity when SNR is greater than 20 dB. As in 16-QAM, the second-order method begins to outperform the first-order method at velocities greater than 75 km/h. However, at velocities less than 75 km/h, iterative compensation seems to be slightly outperformed by the first-order method.

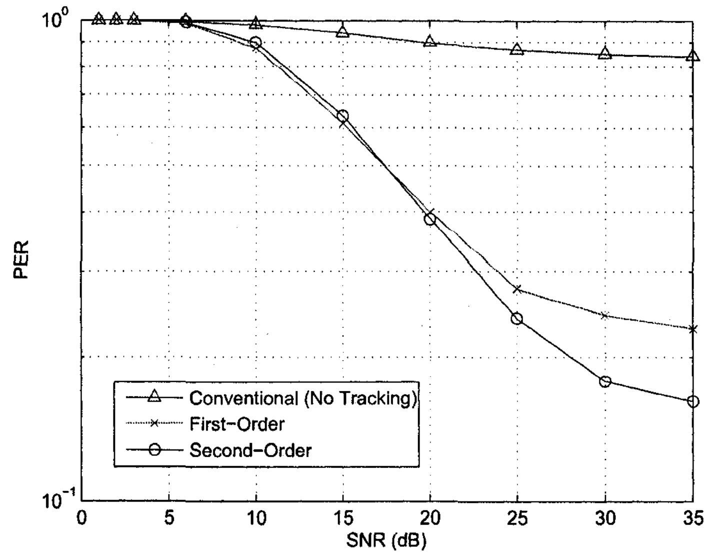


(a) PER vs. SNR at 2 km/h

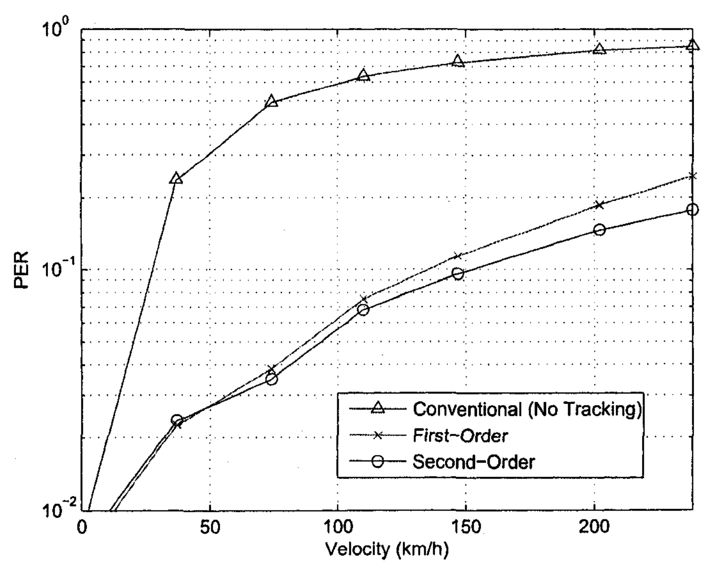


(b) PER vs. Velocity at 30 dB

Figure 5.1: QPSK: First-Order vs. Second-Order under Rayleigh fading.

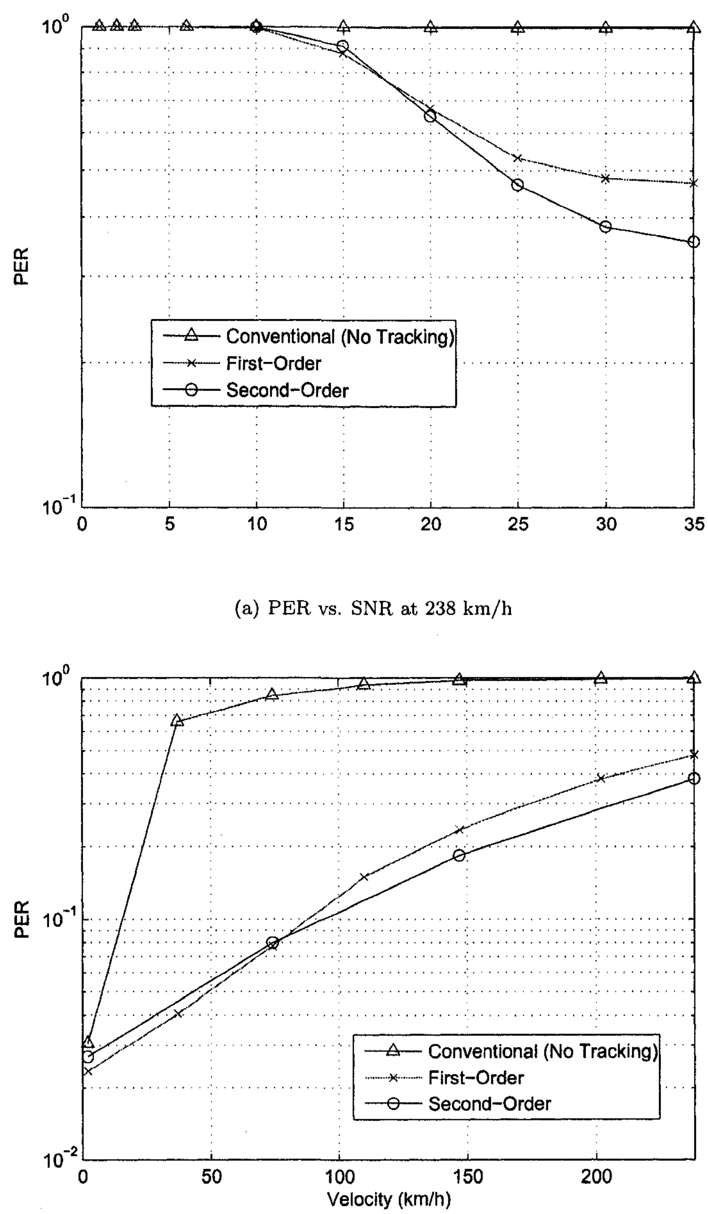


(a) PER vs. SNR at 238 km/h



(b) PER vs. Velocity at 30 dB

Figure 5.2: QAM16: First-Order vs. Second-Order under Rayleigh fading.



(a) PER vs. SNR at 238 km/h

(b) PER vs. Velocity at 30 dB

Figure 5.3: QAM64: First-Order vs. Second-Order under Rayleigh fading.

Chapter 6

Conclusions and Future Work

This thesis provides a comprehensive performance study of a conventional 5.9 GHz DSRC system and identifies the challenges that exist in wireless access vehicular environments. In scenarios with low delay spread and high velocity, 5.9 GHz DSRC is outperformed by IEEE 802.11a. However, in scenarios with high delay spread, IEEE 802.11a completely fails, even at very low velocity. Therefore, 5.9 GHz DSRC is preferred over IEEE 802.11a for harsh outdoor environments. However, due to a lack of channel tracking, conventional 5.9 GHz DSRC system is limited to very low velocities, packet lengths, and data rates. The conventional system would not be suitable for most DSRC applications, which requires high-speed packet transmission at high velocity.

In this thesis, several effective channel estimation schemes are introduced to enhance the system performance. The design of a 5.9 GHz DSRC receiver was proposed with a feedback transmitter from the output of the Viterbi decoder to the input of channel estimator. The feedback transmitter was used to enable channel tracking by generating reliable reference data. The reference data is used by an adaptive algorithm to track the rapid variation of the fading envelope. First-order and second-order adaptive algorithms were proposed to accurately update the channel estimate at every received symbol. First-order channel tracking had made substantial improvements in the packet performance, especially in scenarios where there is a line-of-sight. The proposed method enables the system to achieve higher data rates and larger packet sizes at high velocity with acceptable packet error rate. A novel second-order channel estimation scheme that involves iterative compensation has provided added enhancements to the system, particularly at high velocity.

The deployment of 5.9 GHz DSRC is being planned for the years 2008-2010. In this period, a great deal of advancements can be made by further expanding on the findings of this thesis. Adaptive channel estimation with variable step-size algorithms may be explored. Considering different types of decoders may also be useful. A different type of decoder may be used to generate more reliable reference signals or possibly "soft" feedback. Another addition to the study would be to apply computationally efficient concepts to the proposed schemes for better hardware implementation of the system.

Note that the findings of this thesis may not be limited to 5.9 GHz DSRC. Since the performance was also improved at low-mobility, there are many different type of OFDM systems that may employ such channel estimation techniques.

References

[1] IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, Nov. 1999.

[2] IEEE 802.16: Standard for wireless metropolitan area networks (MAN), Dec. 2001.

[3] DAB System Specifications. http://portal.etsi.org/radio/DigitalAudioBroadcasting/DAB.asp, 2006.

[4] DVB - Digital Video Broadcasting. http://www.dvb.org, 2006.

[5] Status of project IEEE 802.11p. http://grouper.ieee.org/groups/802/11/Reports, 2006.

[6] T. Aulin. A modified model for the fading signal at a mobile radio channel. *IEEE Trans. on Vehicular Technology*, 28:182 – 203, Aug. 1979.

[7] C. Berrou, Glavieux, and P. Thitimajshima. Near shannon limit error correcting coding and decoding: turbo codes. In *IEEE Int. Conf. Communications*, pages 1064–1070, 1993.

[8] S. Boumard and Aarne Mámmela. Channel estimation versus equalization in an OFDM WLAN system. In *Proc. IEEE Vehicular Technology Conference*, volume 1, pages 653 – 657, 2001.

[9] M.R.G. Butler, S. Armour, P.N. Fletcher, A.R. Nix, and D.R. Bull. Viterbi decoding strategies for 5 GHz wireless LAN systems. In *Proc. IEEE Vehicular Technology Conference*, volume 1, pages 77 – 81, Oct. 2001.

[10] R. Chang and R. Gibby. A theoretical study of performance of an orthogonal multiplexing data transmission scheme. *IEEE Trans. on Comm.*, 16:529 – 540, Aug. 1968.

[11] Bing-Leung Patrick Cheung. Simulation of Adaptive Array Algorithms for OFDM and Adaptive Vector OFDM Systems. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2002.

[12] Sinem Coleri, Mustafa Ergen, Anuj Puri, and Ahmad Bahai. Channel estimation techniques based on pilot arrangement in OFDM systems. *IEEE Trans. on Broadcasting*, 48:223 – 228, Sept. 2002.

[13] J.S. Davis and J.P.M.G. Linnartz. Vehicle to vehicle rf propagation measurements. In *Conference Record of the Twenty-Eighth Asilomar*, volume 1, pages 470–474, 1994.

[14] P.S.R. Diniz. *Adaptive Filtering Algorithms and Practical Implementation.* Kluwer, Norwell, Massachusetts, 2002.

[15] Florida Turnpike Enterprise. *ITS Orange Book.* PBS&J, Florida, 2005.

[16] Mike Freitas. Overview of the U.S. DOT Vehicle Infrastructure Initiative. http://ibtta.org/Events/pastpresdetail.cfm, 2005.

[17] Ryuhei Funada, Hiroshi Harada, Yukiyoshi Kamio, Shoji Shinoda, and Masayuki Fujise. A high-mobility packet transmission scheme based on conventional standardized OFDM formats. In *Proc. IEEE 56th Vehicular Tech. Conf.*, volume 1, pages 204 – 208, Sept. 2002.

[18] Ryuhei Funada, Hiroshi Harada, and Shoji Shinoda. Performance improvement of decision-directed channel estimation for DPC-OF/TDMA in a fast fading environment. In *Proc. IEEE 60th Vehicular Tech. Conf.*, volume 7, pages 5125 – 5129, Sept. 2004.

[19] H. Harada and R. Prasad. *Simulation and Software Radio for Mobile Communications.* Artech House, Boston-London, 2002.

[20] Meng-Han Hsieh and Che-Ho Wei. Channel estimation for OFDM systems based on comb-type pilot arrangement in frequency selective fading channels. *IEEE Trans. on Consumer Electronics*, 44:217 – 225, Feb. 1998.

[21] J.D. Forney Jr. The viterbi algorithm. In *Proc. of the IEEE*, volume 61, pages 268 – 278, March 1973.

[22] Sheetal Kalyani and K. Giridhar. Quantised decision based gradient descent algorithm for fast fading OFDM channels. In *Proc. IEEE 60th Vehicular Tech. Conf.*, volume 1, pages 534 – 537, Sept. 2004.

[23] Tideya Kella. Decision-directed channel estimation for supporting higher terminal velocities in ofdm based wlans. In *Proc. IEEE Global Telecomm. Conf.*, volume 3, pages 1306 – 1310, Dec. 2003.

[24] Hyung-Woo Kim, Chae-Hyun Lim, and Dong-Seog Han. Viterbi-decoder aided equalization and sampling clock track of OFDM WLAN. In *Proc. IEEE 60th Vehicular Tech. Conf.*, volume 5, pages 3738 – 3742, Sept. 2004.

[25] Charan Langton. Orthogonal Division Multiplexing (OFDM) Tutorial. http://complextoreal.com/tutorial.htm, 2004.

[26] II Leon W. Couch. *Digital and Analog Communications.* Prentice Hall, Upper Saddle River, NJ, 2001.

[27] Y. Li. Pilot-symbol-aided channel estimation for OFDM in wireless systems. *IEEE Trans. on Vehicular Technology*, 49:1207 – 1215, July 2000.

[28] Zong-Sian Lin, Tze-Lun Hong, and Dah-Chung Chang. Design of an OFDM system with long frame by the decision-aided channel tracking technique. In *Proc. IEEE Sixth Electro/Info. Tech. Conf.*, May. 2006.

[29] M.F. Pop and N.C. Beaulieu. Limitations of sum-of-sinusoids fading channel simulators. *IEEE Trans. on Comm.*, 49:699 – 708, Apr. 2001.

[30] John Proakis. *Digital Communications.* McGraw Hill, New York, NY, 2001.

[31] Jianjun Ran, Rainer Grünheid, Hermann Rohling, Edgar Bolinth, and Ralf Kern. Decision-directed channel estimation method for ofdm systems with high velocities. In *Proc. IEEE 57th Vehicular Tech. Conf.*, volume 4, pages 2358 – 2361, Apr. 2003.

[32] T.S. Rappaport. *Wireless Communications Principles and Practice.* Prentice Hall PTR, Upper Saddle River, New Jersey 07458, 1999.

[33] S.O. Rice. Mathematical analysis of random noise. *Bell System Tech. Journal*, 27:109 - 157, Jan. 1948.

[34] Patrick Robertson, Emanuelle Villebrun, and Peter Hoeher. A comparison of optimal and sub-optimal MAP decoding algorithms operating in the LOG domain. In *Proc. EEE Int. Symp. Circuits and Systems*, pages 1009 - 1013, 1995.

[35] T.M. Schmidl and D.C. Cox. Robust Frequency and Timing Synchronization for OFDM. *IEEE Trans. on Communications*, 45:1613 - 1621, 1997.

[36] S. Sibecas, C.A. Corral, S. Emami, and G. Stratis. On the suitability of 802.11a/ra for high-mobility DSRC. In *Proc. IEEE Vehicular Technology Conference*, volume 1, pages 229 - 234, 2002.

[37] S. Sibecas, C.A. Corral, S. Emami, G. Stratis, and G. Rasor. Pseudo-pilot OFDM scheme for 802.11a and R/A in DSRC applications. In *Proc. IEEE Vehicular Technology Conference*, volume 2, pages 1234 - 1237, Oct. 2003.

[38] O. Simeone, Y. Bar-Ness, and U. Spagnolini. Pilot-based channel estimation for OFDM systems by tracking the delay-subspace. *IEEE Trans. on Wireless Communications*, 3:315 - 325, Jan. 2004.

[39] G.L. Stuber. *Principles of Mobile Communication Second Edition.* Kluwer Academic Publishers, Norwell, Massachusetts 02061, 2000.

[40] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, IT-13:260 - 269, July 1967.

[41] A.J. Viterbi. A personal history of the viterbi algorithm. *IEEE Signal Processing Mag.*, 23:120 - 142, July 2006.

[42] J. Yin, T. ElBatt, G. Yeung, B. Ryu, S. Habermas, H. Krishnan, and T. Talty. Performance evaluation of safety applications over DSRC vehicular ad hoc networks. In *Proc. ACM VANET 2004*, pages 1–9, Oct. 2004.

[43] Qingsheng Yuan, Chen He, Ke Ding, Wei Bai, and Zhiyong Bu. Channel estimation and equalization for OFDM system with fast fading channels. In *Proc. IEEE 60th Vehicular Tech. Conf.*, volume 1, pages 452 - 455, Sept. 2004.

Appendix A

Matlab Code

This appendix contains the code for the major components and configurations of the system. Most system component were coded as individual functions and presented below. An example program for each channel estimation scheme discussed in this thesis is also presented

A.1 Initialization

```
%%%%%%%%%%%%%% MAIN PARAMETERS %%%%%%%%%%%%%%%%%%%%%
para=48;        % number of parallel channels
fftlen=64;      % FFT Length
noc=52;         % number of carriers
sr=125000;      % symbol rate
m=2;            % modulation levels
br=sr*m;        % bit rate
Rc=1/2;         % code rate
n=32/Rc;        % number of OFDM symbols per loop
nloops=10000;% number of loops (packets transmitted)
gi=16;    % guard interval length
kmd=2;          % Number of Training OFDM symbols

%%%%%%%%%%%%% CONSTANTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%
fc=5.9*10^9; % Carrier Frequency (Hz)
c=3*10^8;        % speed of light (m/s)

% generate trellis
K=7;                 % Constraint Length
window=9;            % Trellis Window
softbit=8;           % number of bits for soft input/output
trellis=poly2trellis(K,[171 133]); % Convolutional Encoder
```

```
s=0; % 0--> hard, 1 --> soft decoding

SNR=[1 2 3 6 10 15 20 25 30 35]; % Range of SNR
fd=[10 200 400 600 800 1100 1300]; % Range of Doppler

%%%%%%%%%%%%%%%%%%%% CHANNEL PROPERTIES %%%%%%%%%%%%%%%%%%%%%%%%%%%
    tstp = 1/sr/(fftlen+gi);     % Time resolution
    itime=[0]; % Arrival time (vector size = direct+delayed paths)
    itau = floor(itime./tstp); % Arrival Epoch
    dlvl = [0];     % Mean power for each multipath normalized by direct wave
    M=8;    % Number of oscillators for Sum of Sinosoids
    th=[0];     % Initial Phase of delayed wave
    count_update = n/Rc * (fftlen+gi)*20;     % Number of fading counter to skip
    count=[1000];     % Initial value of fading counter
    npdp=length(itau); % Number of direct+delayed paths in Power Delay Profile
    % Calculate Coherence Time
    Tc = 9/(16.*pi.*fd);
    % Calculate RMS Delay Spread
    tau=0; tau2=0; denom=0;
    for k=1:length(itime)
            tau=tau+10.^(-dlvl(k)/10)*itime(k);
            tau2=tau2+10.^(-dlvl(k)/10)*itime(k).^2;
            denom=denom+10.^(-dlvl(k)/10);
    end
    tau=tau/denom; % mean excess delay
    tau2=tau2/denom; % mean excess squared delay
    sigma=sqrt(tau2-tau.^2); % rms delay spread
    % common rule of thumb Ts > 10 * (sigma)usually means flat fading
    % (1 -> flat , 0 -> freq. selective)
    Ts=1/sr; % symbol period
    if Ts > (10*sigma)
        flat = 1;
    else
        flat = 0;
    end

%%%% Initialize ERROR vectors %%%%
ber_hofdm=zeros(length(fd),length(SNR));
ber_sofdm=zeros(length(fd),length(SNR));
per_hofdm=zeros(length(fd),length(SNR));
per_sofdm=zeros(length(fd),length(SNR));

tic;                    % This begins timing the simulation

fid = fopen('Simulation_Log.dat','w'); % Write data to file
*************************** end of code block *******************
```

A.2 Channel Simulator

A.2.1 Multipath Simulator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: WSSUSricefade.m
% Multipath fading with Uncorrelated 2D isotropic scattering
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% idata : input Ich data
% qdata : input Qch data
% iout  : output Ich data
% qout  : output Qch data
% ramp  : Amplitude contaminated by fading
% rcos  : Cosine value contaminated by fading
% rsin  : Sine value contaminated by fading
% tofa  : Delay time for each multipath fading
% dlvl  : Attenuation level for each multipath fading
% th    : Initialized phase for each multipath fading
% M     : Number of Oscillators for Jakes Simulator
% count : Fading counter for each multipath fading
% npdp  : direct wave (no delay) + number of delayed waves in PDP
% n     : Number of samples to be simulated
% ts    : Minimum time resolution
% fd    : maximum doppler frequency
% count : fading counter
% flat  : flat fading or not
% R     : Ricean Factor (in % LOS)
% ==> 1->flat (only amp fluctuated),0->nomal(phase and amp are fluctutated)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [iout,qout,ramp,rcos,rsin]
         =WSSUSfade(idata,qdata,tofa,dlvl,th,M,count,npdp,n,tstp,fd,flat,R)

% Initialize
iout = R.*idata; qout = R.*qdata;

% total attenuation (power -dB converted to power)
total_attn = sum(10 .^( -1.0 .* dlvl ./ 20.0));

% For the direct and each delay component in the delay power profile:
for k = 1 : npdp

    % 20log(atts)=-dlvl(k) ==> is the attenuation power
    atts = 10.^( -0.05 .* dlvl(k));

    if dlvl(k) >= 40.0
          atts = 0.0;
    end

    theta = th(k) .* pi ./ 180.0;

    [itmp,qtmp] = delay ( idata , qdata , n , tofa(k));
```

```
      [itmp2,qtmp2,ramp,rcos,rsin] = myfade (itmp,qtmp,n,tstp,fd,M,count(k),flat);

    iout = iout + (1-R).*atts .* itmp2 ./ sqrt(total_attn);
    qout = qout + (1-R).*atts .* qtmp2 ./ sqrt(total_attn);

end
% ***********************end of file*********************************
```

A.2.2 Jakes' Rayleigh Fading Simulator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: myfade.m
% Generates Rayleigh Fading Envelope
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% idata : input Ich data
% qdata : input Qch data
% iout  : output Ich data
% qout  : output Qch data
% ramp  : Amplitude contaminated by fading
% rcos  : Cosine value contaminated by fading
% rsin  : Sine value contaminated by fading
% n     : Number of samples to be simulated
% ts    : Minimum time resolution
% fd    : maximum doppler frequency
% M     : number of waves in order to generate fading
% count : fading counter
% flat  : flat fading or not
% ==> 1->flat (only amp fluctuated),0->normal(phase and amp are fluctuated)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function
[iout,qout,ramp,rcos,rsin]=fade(idata,qdata,n,ts,fd,M,count,flat)

%%% Assumptions
% alpha=0

%%%%% Doppler Shift
if fd ~= 0.0

    % Normalize
    ac0 = sqrt(1.0 ./ (2.0.*(M + 1)));    % power normalized constant(ich)
    as0 = sqrt(1.0 ./ (2.0.*M));          % power normalized constant(qch)

    % Parameters
    pai = 3.14159265;
    N = 4.*M + 2;
    wmts = 2.0.*pai.*fd.*ts;


    % Initialize
    xc=zeros(1,n);
    xs=zeros(1,n);
        % here we shift the fading counter
```

```
        % to make the fading of different
        % paths independent (uncorrelated)
    ic=[1:n]+count;

    % Summations
    for nn = 1:M
      cwn = cos( cos(2.0.*pai.*nn./N).*ic.*wmts );
      xc = xc + cos(pai./M.*nn).*cwn;
      xs = xs + sin(pai./M.*nn).*cwn;
    end

    cwmt = sqrt(2.0).*cos(ic.*wmts);
    % Normalized Inphase Component
    xc = (2.0.*xc + cwmt).*ac0;
    % Noramlized Quadrature Component
    xs = 2.0.*xs.*as0;

    % Amplitude Contanimation
    ramp=sqrt(xc.^2+xs.^2);
    rcos=xc./ramp;
    rsin=xs./ramp;


    if flat ==1 % only amplitude fluctuation
        iout = sqrt(xc.^2+xs.^2).*idata(1:n);     % output signal(ich)
        qout = sqrt(xc.^2+xs.^2).*qdata(1:n);     % output signal(qch)
    else % Amplitude and phase fluctuation
        iout = xc.*idata(1:n) - xs.*qdata(1:n);    % output signal(ich)
        qout = xs.*idata(1:n) + xc.*qdata(1:n);    % output signal(qch)
    end

%%%%% No Doppler Shift (No Change)
else
    iout=idata;
    qout=qdata;
    ramp=1;
    rcos=1;
    rsin=1;
end
% ***********************end of file*********************************
```

A.2.3 Additive White Gaussian Noise

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: myawgn.m
% additive white gaussian noise
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% idata : input Ich data
% qdata : input Qch data
% iout  : output Ich data
% qout  : output Qch data
% A     : attenuation level caused by Eb/No or C/N
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [iout ,qout]=myawgn(idata ,qdata ,A)

%output = input + noise
inoise=randn(1,length(idata)).*A; qnoise=randn(1,length(qdata)).*A;

iout=inoise+idata(1:length(idata));
qout=qnoise+qdata(1:length(qdata));
************************** end of code block *******************
```

A.3 Digital Modulation

The following function files are for the mapping and demapping gray-coded constellations.

A.3.1 QPSK Modulation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: qpskmod.m
% Modulation binary data to QPSK
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% paradata   : input data (para-by-n matrix)
% iout       : output I data
% qout       : output Q data
% para       : Number of parallel channels
% n          : Number of data
% M          : Number of modulation QPSK ->2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [iout,qout]=qpskmod(paradata,para,n,M)

m=M./2; paradata2=paradata.*2-1; count=0;

for jj=1:n

    i = zeros(para,1);
    q = zeros(para,1);

    for ii = 1 : m
        i = i + 2.^( m - ii ) .* paradata2((1:para),ii+count);
        q = q + 2.^( m - ii ) .* paradata2((1:para),m+ii+count);
    end

    iout((1:para),jj)=i;
    qout((1:para),jj)=q;
    count=count+M;

end
****************************** end of file **************************
```

A.3.2 QPSK Demodulator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: qpskdemod.m
% Demodulation qpsk to binary data
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% demod      : demodulated data (para-by-n matrix)
% i          : input I data
% q          : input Q data
% para       : Number of parallel channels
% n          : Number of data
% M          : Number of modulation QPSK ->2
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [demod]=qpskdemod(i,q,para,n,M)

demod=zeros(para,M*n); demod((1:para),(1:M:M*n-1))=i((1:para),(1:n))>=0;
demod((1:para),(2:M:M*n))=q((1:para),(1:n))>=0;
****************************** end of file **************************
```

A.3.3 QPSK Soft-Detection

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: qpsksoftdemod.m
% Demodulation qpsk to binary data
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% demod      : demodulated data (para-by-n matrix)
% i          : input I data
% q          : input Q data
% para       : Number of paralell channels
% n          : Number of data
% M          : Number of modulation QPSK ->2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [demod, sdemod] = qpsksoftdemod(i,q,para,n,M)

demod=zeros(para,M*n); sdemod((1:para),(1:M:M*n-1))=i((1:para),(1:n));
sdemod((1:para),(2:M:M*n))=q((1:para),(1:n));
demod((1:para),(1:M:M*n-1))=i((1:para),(1:n))>=0;
demod((1:para),(2:M:M*n))=q((1:para),(1:n))>=0;
****************************** end of file **************************
```

A.3.4 QAM Modulation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: qammod.m
% Modulation binary data to gray-coded 16/64-QAM
% (based on IEEE 802.11a protocol)
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% paradata  : input data (para-by-n matrix)
% iout      : output I data
% qout      : output Q data
% para      : Number of paralell channels
% n         : Number of data
% m         : Number of modulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [iout,qout]=myqammod(paradata,para,n,m)

iout=zeros(para,n); qout=zeros(para,n);

if m==4 % 16QAM
    iv=[-3 -1 3 1];
    k=sqrt(10);
elseif m==6 % 64QAM
    iv =[-7 -5 -1 -3 7 5 1 3];
```

```
    k=sqrt(42);
end

M=m./2; count=0;

for ii=1 : n

    i = zeros(para,1);
    q = zeros(para,1);

    for jj=1 : m

        if jj <= M
            i = i + 2.^( M - jj ).*paradata(:,count+jj);
        else
            q = q + 2.^( m - jj ).*paradata(:,count+jj);
        end

    end

    for p=1:para
        iout(p,ii) = iv(i(p)+1)./k;
        qout(p,ii) =iv(q(p)+1)./k;
    end
    count=count+m;

end
****************************** end of file **************************
```

A.3.5 QAM Demodulator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: qamdemod.m
% Demodulation binary data to 16/64-QAM
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% paradata   : input data (para-by-n matrix)
% iout       : output I data
% qout       : output Q data
% para       : Number of parallel channels
% n          : Number of data
% m          : Number of modulation 16-QAM ---> m=4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [demod]=myqamdemod(idata,qdata,para,n,m)


demod=zeros(para,m*n);

M=m/2; count=0;

if m==4 %%%%%%%%%%%%%% 16QAM

    bin=[0 0; 0 1; 1 0; 1 1];
```

```
    k=sqrt(10);
    idata=idata.*k;
    qdata=qdata.*k;

    for p=1:para
        [iindex, iquant]=quantiz(idata,-2:2:2,[0 1 3 2]);
        [qindex, qquant]=quantiz(qdata,-2:2:2,[0 1 3 2]);
    end

elseif m==6 % 64QAM

    bin=[0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1];

    k=sqrt(42);
    idata=idata.*k;
    qdata=qdata.*k;
    [iindex iquant]=quantiz(idata,-6:2:6,[0 1 3 2 6 7 5 4]);
    [qindex qquant]=quantiz(qdata,-6:2:6,[0 1 3 2 6 7 5 4]);


end

iq=reshape(iquant,para,n); qq=reshape(qquant,para,n);

for ii=1:n
    for p=1:para

        demod(p,(m*(ii-1)+1:m*ii))=[bin(iq(p,ii)+1,:) bin(qq(p,ii)+1,:)];

    end
end
****************************** end of file **************************
```

A.4 Interleaving

A.4.1 Interleaver

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: interleave.m
% this function is a row/column interleaver
% it interleaves data after the encoding stage
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% data     : Input
% out      : Output
% blocksize: size of block data
% rows     : number of rows
% cols     : number of columns
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [out]=interleave(data,blocksize,rows,cols);

numblocks=length(data)/blocksize; out=[];

% interleave one block at a time
for b=0:(numblocks-1)
    % initialize
    block=zeros(rows,cols);
    pos=1;% current position

    %shift
    start=b*blocksize+1;
    temp=data(start:start+blocksize-1);

    % Organize block of data to RxC array
    for r=1:rows
        for c=1:cols
            block(r,c)=temp(pos);
            pos=pos+1;
        end
    end

    % output a block of interleaved data
    for c=1:cols
        for r=1:rows
            out=[out block(r,c)];
        end
    end
end
*************************end of file***********************************
```

A.4.2 De-interleaver

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: deinterleave.m
% this function is a row/column de-interleaver
```

```
% it de-interleaves data before decoding stage
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% data      : Input
% out       : Output
% blocksize : size of block data
% rows      : number of rows
% cols      : number of columns
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [out]=deinterleave(data,blocksize,rows,cols);

numblocks=length(data)/blocksize; out=[];

% de-interleave one block at a time
for b=0:(numblocks-1)

    % initialize
    block=zeros(rows,cols);
    temp=zeros(1,blocksize);
    pos=1;% current position

    %shift
    start=b*blocksize+1;
    temp=data(start:start+blocksize-1);

    % Organize block of data to RxC array
    for c=1:cols
        for r=1:rows
            block(r,c)=temp(pos);
            pos=pos+1;
        end
    end

    % output de-interleaved data
    for r=1:rows
        for c=1:cols
            out=[out block(r,c)];
        end
    end
end
% ***********************end of file**********************************
```

A.5 Other System Components

A.5.1 Parallel-to-Serial and Inserting the Guard Interval

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: insertgi.m
% Insert the guard interval after IFFT
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% idata     : Input Ich data
% qdata     : Input Qch data
% iout      : Output Ich data
% qout      : Output Qch data
% fftlen    : Length of FFT (points)
% gi        : Length of guard interval (points)

function [iout,qout]= insertgi(idata,qdata,fftlen,gi,n);

% idata1=reshape(idata,fftlen,n);
% qdata1=reshape(qdata,fftlen,n);

% Insert Guard Interval
% (Copy End of Symol and Place at Front of Symbol)
idata2=[idata(fftlen-gi+1:fftlen,:); idata];
qdata2=[qdata(fftlen-gi+1:fftlen,:); qdata];

% Parrallel to Serial
iout=reshape(idata2,1,(fftlen+gi)*n);
qout=reshape(qdata2,1,(fftlen+gi)*n);
%************************end of file***********************************
```

A.5.2 Removing Guard Interval and Serial-to-Parallel

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: removegi.m
% Removal the guard interval at the Receiver
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% idata     : Input Ich data
% qdata     : Input Qch data
% iout      : Output Ich data
% qout      : Output Qch data
% fftlen    : Length of FFT (points)
% gi        : Length of guard interval (points)

function [iout,qout]= insertgi(idata,qdata,fftlen,gi,n);

% Serial to Parallel
idata1=reshape(idata,fftlen+gi,n);
qdata1=reshape(qdata,fftlen+gi,n);

% Remove Guard
iout=idata1(gi+1:gi+fftlen,:); qout=qdata1(gi+1:gi+fftlen,:);
```

```
% ************************end of file************************************
```

A.5.3 Inserting Pilots

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: map_pilot.m
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% idata       : Input Ich data
% qdata       : Input Qch data
% iout        : Output Ich data
% qout        : Output Qch data
% fftlen      : Length of FFT (points)
% para        : number of parallel channels
% n          : Number of OFDM symbols
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [iout,qout]=map_pilot(idata,qdata,fftlen,para,n);

iout=zeros(fftlen,n); qout=zeros(fftlen,n);

% set the pilots
iout(8,:)=1;     %+7
iout(22,:)=-1;   %+21
iout(58,:)=1;    %-7
iout(44,:)=1;    %-21

% The rest of the data
iout(2:7,:)=idata(1:6,:); iout(9:21,:)=idata(7:19,:);
iout(23:27,:)=idata(20:24,:); iout(39:43,:)=idata(25:29,:);
iout(45:57,:)=idata(30:42,:); iout(59:64,:)=idata(43:48,:);

qout(2:7,:)=qdata(1:6,:); qout(9:21,:)=qdata(7:19,:);
qout(23:27,:)=qdata(20:24,:); qout(39:43,:)=qdata(25:29,:);
qout(45:57,:)=qdata(30:42,:); qout(59:64,:)=qdata(43:48,:);
% ************************end of file************************************
```

A.5.4 Removing Pilots

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: demap_pilot.m
% Author: Harb Abdulhamid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% idata       : Input Ich data
% qdata       : Input Qch data
% iout        : Output Ich data
% qout        : Output Qch data
% fftlen      : Length of FFT (points)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [iout,qout]=demap_pilot(idata,qdata,fftlen,para);

iout(1:6,:)=idata(2:7,:); iout(7:19,:)=idata(9:21,:);
```

```
iout(20:24,:)=idata(23:27,:); iout(25:29,:)=idata(39:43,:);
iout(30:42,:)=idata(45:57,:); iout(43:48,:)=idata(59:64,:);

qout(1:6,:)=qdata(2:7,:); qout(7:19,:)=qdata(9:21,:);
qout(20:24,:)=qdata(23:27,:); qout(25:29,:)=qdata(39:43,:);
qout(30:42,:)=qdata(45:57,:); qout(43:48,:)=qdata(59:64,:);
% ************************ end of file ***********************************
```

A.6 Example Programs

A.6.1 Conventional DSRC System with QPSK modulation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% QPSKcofdm_interleave.m %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% MAIN PROGRAM %%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;

fid = fopen('QPSKcofdm.dat','w');

for ff=1:length(fd)

    fprintf(fid,'\n\tfd=%g\n',fd(ff));
    fprintf('\n\tfd=%g\n',fd(ff));

 for ss=1:length(SNR)

    % initialize
    be=0; sbe=0;              % total bit errors
    pe=0; spe=0;             % total packet errors
    count=[1000];
    % Calculate Coherence Time (remove added 1 to fd)
    Tc(ff) = sqrt(9/(16*pi*(fd(ff)+1).^2));
    for ii=1:nloops

        be1=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%% TRANSMITTER %%%%%%%%%%%%%%%%%%

        % generate random binary data (serial data)
        input=randint(1,para*n*m*Rc);

        % FEC (convolutional encoder)
        % initial zero state and final zero state and includes puncturing
        encoded=convenc(input,trellis);

        % bit interleaving 12x8 block since One OFDM symbol=96bits
        encoded2=interleave(encoded,para*m,12,8);

        % convert serial data to parallel data
        parainput=reshape(encoded2,para,n*m);

        % QPSK modulation
        [imod,qmod]=qpskmod(parainput,para,n,m);
            % Normalize
            imod2=imod./sqrt(2);
            qmod2=qmod./sqrt(2);

        % CE data generation as seen in 802.11a (L -26 to 26)
        L=[1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1,
```

```
    1, -1, 1, 1, 1, 1, 0,1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, -1, -1,
    -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1];
kndata0=[L;L];
% map CE data
    kndata(:,1:(noc/2+1))=kndata0(:,(noc/2+1):noc+1); % 0 to 26
    kndata(:,(fftlen-(noc/2-1)):fftlen)=kndata0(:,1:(noc/2));%-26to-1
    ceich=kndata; % CE:BPSK
    ceqch=zeros(knd,fftlen);

% data mapping
[imap,qmap]=map_pilot(imod2,qmod2,fftlen,para,n);

% Insert Pilot Symbol
imap2=[ceich.' imap];
qmap2=[ceqch.' qmap];

% IFFT Block
x=imap2+qmap2.*i;
y=ifft(x);
itdata=real(y);
qtdata=imag(y);

% Insert Guard Interval
[itgdata,qtgdata]=insertgi(itdata,qtdata,fftlen,gi,n+knd);

%%%%%%%%%%%%%%%%%%%%%%%%%% CHANNEL %%%%%%%%%%%%%%%%%%%%%%%%

    % Generated data are fed into a fading simulator
    [ifade,qfade,ramp,rcos,rsin]=WSSUSfade(itgdata,qtgdata,itau,
            lvl,th,M,count,npdp,length(itgdata),tstp,fd(ff),flat);

    % Update fading counter
    count = count + count_update;

    % AWGN
        % calculate attenuation
        spow=sum(itgdata.^2+qtgdata.^2)/n./para;
        npow=spow*sr/br*10.^(-SNR(ss)/10);
        A=sqrt(0.5*npow);
    [ichan,qchan]=myawgn(ifade,qfade,A);

%%%%%%%%%%%%%%%%%%%%%%%%%% RECIEVER %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Perfect Compensation
%           ifade2=1./ramp.*(rcos(1,:).*ichan + rsin(1,:).*qchan);
%           qfade2=1./ramp.*(-rsin(1,:).*ichan + rcos(1,:).*qchan);
%           ichan2=ifade2;
%           qchan2=qfade2;

% Remove Guard Interval (for perfect comp use ichan2 and qchan2)
[irdata,qrdata]=removegi(ichan,qchan,fftlen,gi,n+knd);

% FFT Block (removegi function carries out S/P operation)
rx=irdata+qrdata.*i;
ry=fft(rx);
```

```
        ir=real(ry);
        qr=imag(ry);

        % Fading compensation by CE symbols
        [ich7,qch7]=CE(ir,qr,imap2,qmap2,n,knd);

        % CE symbol removal
        ich8=ich7(:,knd+1:n+knd);
        qch8=qch7(:,knd+1:n+knd);

        % Data demapping
        [ir1,qr1]=demap_pilot(ich8,qch8,fftlen,para);

        % Un-Normalize
        ir2=ir1.*sqrt(2);
        qr2=qr1.*sqrt(2);

        % Demodulate data
        [demod, sdemod]=qpsksoftdemod(ir2,qr2,para,n,m);

        % Parallel to Serial
        serdemod=reshape(demod,1,para*n*m);
        sersdemod=reshape(sdemod,1,para*n*m);

        % de-interleave with 12x8 block
        hard=deinterleave(serdemod,para*m,12,8);
        soft=deinterleave(sersdemod,para*m,12,8);

        % Hard Decision Viterbi Decoding
            houtput=vitdec(hard,trellis,window,'trunc','hard');

        % Quantize for soft decoding
            qcode = quantiz(soft,[-1.0:(2/(2^softbits-1)):1.0]);
            % Soft Viterbi Decoding
            soutput=vitdec(qcode',trellis,window,'trunc','soft',softbits);

        % Bit Errors
        hard_be=sum(abs(input-houtput)); % bit errors in this loop
        be=be+hard_be; % total bit errors
        soft_be=sum(abs(input-soutput)); % bit errors in this loop
        sbe=sbe+soft_be; % total bit errors

        % Packet Errors
        if hard_be~=0
            pe=pe+1;
        end
        if soft_be~=0
            spe=spe+1;
        end
        fprintf('fd=%gHz SNR=%gdB nloop=%g\n',fd(ff),SNR(ss),ii);

    end

    ber_hofdm(ss,ff)=be/(para*n*Rc*m*nloops);
```

```
        per_hofdm(ss,ff)=pe/(nloops);
        ber_sofdm(ss,ff)=sbe/(para*n*Rc*m*nloops);
        per_sofdm(ss,ff)=spe/(nloops);

        fprintf(fid,'Hard: ber_ofdm(%g,%g)=%e;  per_ofdm(%g,%g)=%e;\n',ss,
                        ff,ber_hofdm(ss,ff),ss,ff,per_hofdm(ss,ff));
        fprintf(fid,'Soft: ber_ofdm(%g,%g)=%e;  per_ofdm(%g,%g)=%e;\n',ss,
                        ff,ber_sofdm(ss,ff),ss,ff,per_sofdm(ss,ff));
        fprintf('SNR=%gdb\nHard: BER=%e  PER=%e\nSoft: BER=%e  PER=%e\n',SNR(ss),
            ber_hofdm(ss,ff),per_hofdm(ss,ff),ber_sofdm(ss,ff),per_sofdm(ss,ff));

 end

end

fclose(fid);

toc;

 save DSRCcurves/bQPSK_hcode ber_hofdm;
 save DSRCcurves/pQPSK_hcode per_hofdm;
 save DSRCcurves/bQPSK_scode ber_sofdm;
 save DSRCcurves/pQPSK_scode per_sofdm;
% ************************end of file***********************************
```

A.6.2 Decision-Aided I Channel Estimation (from Demapping Circuit) in a DSRC System with 16-QAM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%% QAM_DD4.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% MAIN PROGRAM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ber_ofdm=zeros(length(SNR),length(fd));
per_ofdm=zeros(length(SNR),length(fd));

tic;

fid = fopen('QAM16cofdm.dat','w');

for ff=1:length(fd)

    fprintf(fid,'\n\tfd=%g\n',fd(ff));
    fprintf('\n\tfd=%g\n',fd(ff));

 for ss=1:length(SNR)

    % initialize
    be=0; sbe=0;            % total bit errors
    pe=0; spe=0;            % total packet errors
    % Calculate Coherence Time
    for ii=1:nloops

        be1=0;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% TRANSMITTER %%%%%%%%%%%%%%%%%%%%

    % generate random binary data (serial data)
    input=randint(1,para*n*m*Rc);

    % FEC (convolutional encoder)
    % initial zero state and final zero state and includes puncturing
    encoded=convenc(input,trellis);

    % bit interleaving (12x16-16QAM) 12x16 block
    % where
    encoded2=interleave(encoded,para*m,12,16);

    % convert serial data to parallel data
    parainput=reshape(encoded2,para,n*m);

    % QAM16 modulation
    [imod,qmod]=myqammod(parainput,para,n,m);

    % CE data generation as seen in 802.11a (L -26 to 26)
    L=[1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1,
        1, -1, 1, 1, 1, 1, 0,1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, -1, -1,
        -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1];
    kndata0=[L;L];
    % map CE data
        kndata(:,1:(noc/2+1))=kndata0(:,(noc/2+1):noc+1); % 0 to 26
        kndata(:,(fftlen-(noc/2-1)):fftlen)=kndata0(:,1:(noc/2));% -26to-1
        ceich=kndata; % CE:BPSK
        ceqch=zeros(knd,fftlen);

    % data mapping
    [imap,qmap]=map_pilot(imod,qmod,fftlen,para,n);

    % Insert Pilot Symbol
    imap2=[ceich.' imap];
    qmap2=[ceqch.' qmap];

    % IFFT Block
    x=imap2+qmap2.*i;
    y=ifft(x);
    itdata=real(y);
    qtdata=imag(y);

    % Insert Guard Interval
    [itgdata,qtgdata]=insertgi(itdata,qtdata,fftlen,gi,n+knd);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CHANNEL %%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Generated data are fed into a fading simulator
        if fd==0;
            ifade=itgdata;
            qfade=qtgdata;
        else
```

```
            [ifade,qfade,ramp,rcos,rsin]=WSSUSfade(itgdata,qtgdata,itau,
                    dlvl,th,M,count,npdp,length(itgdata),tstp,fd(ff),flat);
        end
        % Update fading counter
        count = count + count_update;

        % AWGN
            % calculate attenuation
            spow=sum(itgdata.^2+qtgdata.^2)/n./para;
            npow=spow*sr/br*10.^(-SNR(ss)/10);
            A=sqrt(0.5*npow);
        [ichan,qchan]=myawgn(ifade,qfade,A);

%%%%%%%%%%%%%%%%%%%%%%%% RECIEVER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Remove Guard Interval (for perfect comp use ichan2 and qchan2)
    [irdata,qrdata]=removegi(ichan,qchan,fftlen,gi,n+knd);

    % FFT Block (removegi function carries out S/P operation)
    rx=irdata+qrdata.*i;
    ry=fft(rx);
    ir=real(ry);
    qr=imag(ry);

    %%%% Initial Channel Estimate using CE symbols
        % preparation known CE data (Xtrain)
        ice0=imap2(:,1);
        qce0=qmap2(:,1);

        % taking CE data out of received data (Y1 and Y2)
        ice01=ir(:,1:knd);
        qce01=qr(:,1:knd);

        % Taking average over received symbols (Y1+Y2/2)
        ice1=ice01(:,1)./2+ice01(:,2)./2;
        qce1=qce01(:,1)./2+qce01(:,2)./2;

        % calculating initial reverse rotation
        ieq(:,1)=real((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)
                                                .*(ice1-i.*qce1));
        qeq(:,1)=imag((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)
                                                .*(ice1-i.*qce1));

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%% DEMAPPER AIDED CHANNEL ESTIMATION %%%%%%%%%%%%

    fs=0; % initial state of encoder

    for nn=1:n-1;

        % Signal Compensation
        ich7(:,nn)=real((ir(:,nn+knd)+i.*qr(:,nn+knd))
                                    .*(ieq(:,nn)+i.*qeq(:,nn)));
        qch7(:,nn)=imag((ir(:,nn+knd)+i.*qr(:,nn+knd))
```

```
                                        .*(ieq(:,nn)+i.*qeq(:,nn)));

    % Data demapping
    [irl(:,nn),qrl(:,nn)]
            =demap_pilot(ich7(:,nn),qch7(:,nn),fftlen,para);

    % Demodulate data
    demod=myqamdemod(irl(:,nn),qrl(:,nn),para,1,m);

    % Parallel to Serial
    serdemod((para*m*(nn-1)+1):para*m*nn)=reshape(demod,1,para*m);

    % de-interleave with 12x16 block
    hard((para*m*(nn-1)+1):para*m*nn)
            =deinterleave(serdemod((para*m*(nn-1)+1):para*m*nn),
                                                para*m,12,16);

    % Hard Decision Viterbi Decoding
        if nn==1
            [hdec,fme,fst,fin]
                =vitdec(hard((para*m*(nn-1)+1):para*m*nn),trellis
                                        ,window,'cont','hard');
            last=vitdec(encoded(para*m*nn+1:para*m*(nn+1)),trellis,
                                ,window,'cont','hard',fme,fst,fin);
        else
            [hdec,fme,fst,fin]
                =vitdec(hard((para*m*(nn-1)+1):para*m*nn),trellis,
                    window,'cont','hard',fme,fst,fin);
            last=vitdec(encoded(para*m*nn+1:para*m*(nn+1)),trellis,
                    window,'cont','hard',fme,fst,fin);
        end

    houtput((para*m*Rc*(nn-1)+1):para*m*Rc*nn)
                    =[hdec(window+1:length(hdec)) last(1:window)];

    %% Quantize for soft decoding
    % qcode((para*m*(nn-1)+1):para*m*nn)
    %   = quantiz(soft((para*m*(nn-1)+1):para*m*nn),
    %               [-1.0:(2/(2^softbits-1)):1.0]);
    %
    %% Soft Viterbi Decoding
    %    if nn==1
    %        [hdec,fme,fst,fin]
    %               =vitdec(qcode((para*m*(nn-1)+1):para*m*nn),
    %                   trellis,window,'cont','soft',softbits);
    %        last=vitdec(encoded(para*m*nn+1:para*m*(nn+1))*31+32,
    %             trellis,window,'cont','soft',softbits,fme,fst,fin);
    %    else
    %        [hdec,fme,fst,fin]
    %             =vitdec(qcode((para*m*(nn-1)+1):para*m*nn),
    %             trellis,window,'cont','soft',softbits,fme,fst,fin);
    %        last=vitdec(encoded(para*m*nn+1:para*m*(nn+1))*31+32,
    %             trellis,window,'cont','soft',softbits,fme,fst,fin);
    %    end
```

```
            %
            % houtput((para*m*Rc*(nn-1)+1):para*m*Rc*nn)
            %                     =[hdec(window+1:length(hdec))  last(1:window)];

            %%%%%%%%%%%% DECISION FEEDBACK (from Demapping Circuit) %%%%%%%%%%

                % QAM16 modulation
                [himod,hqmod]=myqammod(demod,para,1,m);

                % data mapping
                [himap,hqmap]=map_pilot(himod,hqmod,fftlen,para,1);

                % calculating initial reverse rotation
                ieq_hat(:,nn)=real((1./(ir(:,nn+knd).^2+qr(:,nn+knd).^2))
                            .*(himap+i.*hqmap).*(ir(:,nn+knd)-i.*qr(:,nn+knd)));
                qeq_hat(:,nn)=imag((1./(ir(:,nn+knd).^2+qr(:,nn+knd).^2))
                            .*(himap+i.*hqmap).*(ir(:,nn+knd)-i.*qr(:,nn+knd)));

                ieq(:,nn+1)=ieq(:,nn).*(gamma)+(1-gamma).*ieq_hat(:,nn);
                qeq(:,nn+1)=qeq(:,nn).*(gamma)+(1-gamma).*qeq_hat(:,nn);
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Bit Errors
        soft_be=sum(abs(input(1:length(input)-para*m/2)-houtput)); % errors in loop
        sbe=sbe+soft_be; % total bit errors

        % Packet Errors
        if soft_be~=0
            spe=spe+1;
        end
  fprintf('fd=%gHz SNR=%gdB nloop=%g BE=%g PE=%g\n',fd(ff),SNR(ss),ii,soft_be,spe);

    end

    ber_ofdm(ss,ff)=sbe/(para*n*Rc*m*nloops);
    per_ofdm(ss,ff)=spe/(nloops);

    fprintf(fid,'ber_ofdm(%g,%g)=%e;  per_ofdm(%g,%g)=%e;\n',ss,ff,ber_ofdm(ss,ff),
                                                            ss,ff,per_ofdm(ss,ff));
    fprintf(fid,'n=%g; s=%g; gamma=%g;\n',n,s,gamma);
    fprintf('BER=%e;  PER=%e;\t',ber_ofdm(ss,ff),per_ofdm(ss,ff));
    fprintf('n=%g; s=%g; gamma=%g;\n',n,s,gamma);
  end

end

fclose(fid);

toc;
```

```
bQAM16_DD4_g6=ber_ofdm;
pQAM16_DD4_g6=per_ofdm;
save DSRCcurves/bQAM16_DD4_g6 bQAM16_DD4_g6;
save DSRCcurves/pQAM16_DD4_g6 pQAM16_DD4_g6
% ***********************end of file***********************************
```

A.6.3 Decision-Aided II (Viterbi-Aided) Channel Estimation in a DSRC System with 64-QAM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% QAM64_DD2.m %%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% MAIN PROGRAM %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;

fid = fopen('QAM64cofdm.dat','w');

for ff=1:length(fd)

    fprintf(fid,'\n\tfd=%g\n',fd(ff));
    fprintf('\n\tfd=%g\n',fd(ff));

  for ss=1:length(SNR)

    % initialize
    be=0; sbe=0;            % total bit errors
    pe=0; spe=0;            % total packet errors
    for ii=1:nloops

        bel=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%% TRANSMITTER %%%%%%%%%%%%%%%%%%

        % generate random binary data (serial data)
        input=randint(1,para*n*m*Rc);

        % FEC (convolutional encoder)
        % initial zero state and final zero state and includes puncturing
        encoded=convenc(input,trellis);

        % bit interleaving (12x16-16QAM)
        encoded2=interleave(encoded,para*m,18,16);

        % convert serial data to parallel data
        parainput=reshape(encoded2,para,n*m);

        % QAM16 modulation
        [imod,qmod]=myqammod(parainput,para,n,m);

        % CE data generation as seen in 802.11a (L -26 to 26)

        L=[1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1,
```

```
        1, -1, 1, 1, 1, 1, 0,1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, -1, -1,
        -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1];
    kndata0=[L;L];
    % map CE data
        kndata(:,1:(noc/2+1))=kndata0(:,(noc/2+1):noc+1); % 0 to 26
        kndata(:,(fftlen-(noc/2-1)):fftlen)=kndata0(:,1:(noc/2));%-26to-1
        ceich=kndata; % CE:BPSK
        ceqch=zeros(knd,fftlen);

    % data mapping
    [imap,qmap]=map_pilot(imod,qmod,fftlen,para,n);

    % Insert Pilot Symbol
    imap2=[ceich.' imap];
    qmap2=[ceqch.' qmap];

    % IFFT Block
    x=imap2+qmap2.*i;
    y=ifft(x);
    itdata=real(y);
    qtdata=imag(y);

    % Insert Guard Interval
    [itgdata,qtgdata]=insertgi(itdata,qtdata,fftlen,gi,n+knd);

%%%%%%%%%%%%%%%%%%%%%%%% CHANNEL %%%%%%%%%%%%%%%%%%%%%%

        % Generated data are fed into a fading simulator
        if fd==0;
            ifade=itgdata;
            qfade=qtgdata;
        else
            [ifade,qfade,ramp,rcos,rsin]
                =WSSUSfade(itgdata,qtgdata,itau,dlvl,th,M,count,
                            npdp,length(itgdata),tstp,fd(ff),flat);
        end
        % Update fading counter
        count = count + count_update;

        % AWGN
            % calculate attenuation
            spow=sum(itgdata.^2+qtgdata.^2)/n./para;
            npow=spow*sr/br*10.^(-SNR(ss)/10);
            A=sqrt(0.5*npow);
        [ichan,qchan]=myawgn(ifade,qfade,A);

%%%%%%%%%%%%%%%%%%%%%% RECIEVER %%%%%%%%%%%%%%%%%%%%%%%%%

    % Remove Guard Interval (for perfect comp use ichan2 and qchan2)
    [irdata,qrdata]=removegi(ichan,qchan,fftlen,gi,n+knd);

    % FFT Block (removegi function carries out S/P operation)
    rx=irdata+qrdata.*i;
    ry=fft(rx);
```

```
ir=real(ry);
qr=imag(ry);

%%% Initial Channel Estimate using CE symbols
    % preparation known CE data (Xtrain)
    ice0=imap2(:,1);
    qce0=qmap2(:,1);

    % taking CE data out of received data (Y1 and Y2)
    ice01=ir(:,1:knd);
    qce01=qr(:,1:knd);

    % Taking average over received symbols (Y1+Y2/2)
    ice1=ice01(:,1)./2+ice01(:,2)./2;
    qce1=qce01(:,1)./2+qce01(:,2)./2;

    % calculating initial reverse rotation
    ieq(:,1)=real((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)
                                             .*(ice1-i.*qce1));
    qeq(:,1)=imag((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)
                                             .*(ice1-i.*qce1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% VITERBI AIDED CHANNEL ESTIMATION %%%%%%%%%%%%%

fs=0; % initial state of encoder

for nn=1:n-1;

    % Signal Compensation
    ich7(:,nn)=real((ir(:,nn+knd)+i.*qr(:,nn+knd)).*(ieq(:,nn)
                                                 +i.*qeq(:,nn)));
    qch7(:,nn)=imag((ir(:,nn+knd)+i.*qr(:,nn+knd)).*(ieq(:,nn)
                                                 +i.*qeq(:,nn)));

    % Data demapping
    [ir1(:,nn),qr1(:,nn)]
                =demap_pilot(ich7(:,nn),qch7(:,nn),fftlen,para);

    % Demodulate data
    demod=myqamdemod(ir1(:,nn),qr1(:,nn),para,1,m);

    % Parallel to Serial
    serdemod((para*m*(nn-1)+1):para*m*nn)=reshape(demod,1,para*m);

    % de-interleave with 18x16 block
hard((para*m*(nn-1)+1):para*m*nn)
  =deinterleave(serdemod((para*m*(nn-1)+1):para*m*nn),para*m,18,16);

    % Hard Decision Viterbi Decoding
        if nn==1
            [hdec,fme,fst,fin]
                =vitdec(hard((para*m*(nn-1)+1):para*m*nn),trellis,
                                          window,'cont','hard');
```

```
                    last=vitdec(encoded(para*m*nn+1:para*m*(nn+1)),trellis,
                                        window,'cont','hard',fme,fst,fin);
                else
                    [hdec,fme,fst,fin]=vitdec(hard((para*m*(nn-1)+1):para*m*nn)
                                  ,trellis,window,'cont','hard',fme,fst,fin);
                    last=vitdec(encoded(para*m*nn+1:para*m*(nn+1)),trellis,
                                        window,'cont','hard',fme,fst,fin);
                end

            houtput((para*m*Rc*(nn-1)+1):para*m*Rc*nn)
                             =[hdec(window+1:length(hdec)) last(1:window)];

            %%%%%%%%%%%% DECISION FEEDBACK %%%%%%%%%%%%
                [hencoded fs]
                =convenc(houtput((para*m*Rc*(nn-1)+1):para*m*Rc*nn),trellis,fs);

                % bit interleaving 18x16
                hencoded2=interleave(hencoded,para*m,18,16);

                % convert serial data to parallel data
                hparainput=reshape(hencoded2,para,m);

                % QAM16 modulation
                [himod,hqmod]=myqammod(hparainput,para,1,m);

                % data mapping
                [himap,hqmap]=map_pilot(himod,hqmod,fftlen,para,1);

                % calculating initial reverse rotation
                ieq_hat(:,nn)=real((1./(ir(:,nn+knd).^2+qr(:,nn+knd).^2))
                    .*(himap+i.*hqmap).*(ir(:,nn+knd)-i.*qr(:,nn+knd)));
                qeq_hat(:,nn)=imag((1./(ir(:,nn+knd).^2+qr(:,nn+knd).^2))
                    .*(himap+i.*hqmap).*(ir(:,nn+knd)-i.*qr(:,nn+knd)));

                ieq(:,nn+1)=ieq(:,nn).*(gamma)+(1-gamma).*ieq_hat(:,nn);
                qeq(:,nn+1)=qeq(:,nn).*(gamma)+(1-gamma).*qeq_hat(:,nn);
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Bit Errors
      soft_be=sum(abs(input(1:length(input)-para*m/2)-houtput));%errors in loop
      sbe=sbe+soft_be; % total bit errors

        % Packet Errors
        if soft_be~=0
            spe=spe+1;
        end
        fprintf('fd=%gHz SNR=%gdB nloop=%g BE=%g PE=%g\n',fd(ff),SNR(ss),
                                                         ii,soft_be,spe);

    end
```

```
    ber_ofdm(ss, ff)=sbe/(para*n*Rc*m*nloops);
    per_ofdm(ss, ff)=spe/(nloops);

    fprintf(fid, 'ber_ofdm(%g,%g)=%e;   per_ofdm(%g,%g)=%e;\n', ss, ff,
                                    ber_ofdm(ss, ff), ss, ff, per_ofdm(ss, ff));
    fprintf(fid, 'n=%g; s=%g; gamma=%g;\n', n, s, gamma);
    fprintf('BER=%e;   PER=%e;\t', ber_ofdm(ss, ff), per_ofdm(ss, ff));
    fprintf('n=%g; s=%g; gamma=%g;\n', n, s, gamma);
  end

end

fclose(fid);

toc;

 bQAM64_DD2_g5=ber_ofdm;
 pQAM64_DD2_g5=per_ofdm;
 save DSRCcurves/bQAM64_DD2_g5 bQAM64_DD2_g5
 save DSRCcurves/pQAM64_DD2_g5 pQAM64_DD2_g5
% ************************end of file***********************************
```

A.6.4 Second-Order Method I - Channel Estimation (Iterative Compensation) in a DSRC System with 16-QAM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%% QAM_DD5.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% MAIN PROGRAM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;

fid = fopen('QAM16cofdm.dat','w');

for ff=1:length(fd)

    fprintf(fid, '\n\tfd=%g\n', fd(ff));
    fprintf('\n\tfd=%g\n', fd(ff));

 for ss=1:length(SNR)

    % initialize
    be=0; sbe=0;            % total bit errors
    pe=0; spe=0;            % total packet errors
    % Calculate Coherence Time
    for ii=1:nloops

        bel=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%% TRANSMITTER %%%%%%%%%%%%%%%%%%%%%%%%%

        % generate random binary data (serial data)
```

```
input=randint(1,para*n*m*Rc);

% FEC (convolutional encoder)
% initial zero state and final zero state and includes puncturing
encoded=convenc(input, trellis);

% bit interleaving (12x16-16QAM) 12x16 block
% where
encoded2=interleave(encoded,para*m,12,16);

% convert serial data to parallel data
parainput=reshape(encoded2,para,n*m);

% QAM16 modulation
[imod,qmod]=myqammod(parainput,para,n,m);

% CE data generation as seen in 802.11a (L -26 to 26)
L=[1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1,
    1, -1, 1, 1, 1, 1, 0,1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, -1, -1,
    -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1];
kndata0=[L;L];
% map CE data
    kndata(:,1:(noc/2+1))=kndata0(:,(noc/2+1):noc+1); % 0 to 26
    kndata(:,(fftlen-(noc/2-1)):fftlen)=kndata0(:,1:(noc/2));% -26to-1
    ceich=kndata; % CE:BPSK
    ceqch=zeros(knd,fftlen);

% data mapping
[imap,qmap]=map_pilot(imod,qmod,fftlen,para,n);

% Insert Pilot Symbol
imap2=[ceich.' imap];
qmap2=[ceqch.' qmap];

% IFFT Block
x=imap2+qmap2.*i;
y=ifft(x);
itdata=real(y);
qtdata=imag(y);

% Insert Guard Interval
[itgdata,qtgdata]=insertgi(itdata,qtdata,fftlen,gi,n+knd);

%%%%%%%%%%%%%%%%%%%%%%%%%% CHANNEL %%%%%%%%%%%%%%%%%%%%%%%%%

    % Generated data are fed into a fading simulator
    if fd==0;
        ifade=itgdata;
        qfade=qtgdata;
    else
        [ifade,qfade,ramp,rcos,rsin]=WSSUSfade(itgdata,qtgdata,itau,
                dlvl,th,M,count,npdp,length(itgdata),tstp,fd(ff),flat);
    end
    % Update fading counter
```

```
        count = count + count_update;

        % AWGN
            % calculate attenuation
            spow=sum(itgdata.^2+qtgdata.^2)/n./para;
            npow=spow*sr/br*10.^(-SNR(ss)/10);
            A=sqrt(0.5*npow);
        [ichan,qchan]=myawgn(ifade,qfade,A);

%%%%%%%%%%%%%%%%%%%%%%%% RECIEVER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Remove Guard Interval (for perfect comp use ichan2 and qchan2)
    [irdata,qrdata]=removegi(ichan,qchan,fftlen,gi,n+knd);

    % FFT Block (removegi function carries out S/P operation)
    rx=irdata+qrdata.*i;
    ry=fft(rx);
    ir=real(ry);
    qr=imag(ry);

    %%%% Initial Channel Estimate using CE symbols
        % preparation known CE data (Xtrain)
        ice0=imap2(:,1);
        qce0=qmap2(:,1);

        % taking CE data out of received data (Y1 and Y2)
        ice01=ir(:,1:knd);
        qce01=qr(:,1:knd);

        % Taking average over received symbols (Y1+Y2/2)
        ice1=ice01(:,1)./2+ice01(:,2)./2;
        qce1=qce01(:,1)./2+qce01(:,2)./2;

        % calculating initial reverse rotation
        ieq(:,1)=real((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)
                                               .*(ice1-i.*qce1));
        qeq(:,1)=imag((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)
                                               .*(ice1-i.*qce1));

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%% VITERBI AIDED CHANNEL ESTIMATION %%%%%%%%%%%

    fs=0; % initial state of encoder
    fs_S2=0; % initial state of encoder for second stage
    for nn=1:n-1;

        % Signal Compensation
        ich7(:,nn)=real((ir(:,nn+knd)+i.*qr(:,nn+knd)).*(ieq(:,nn)+i.*qeq(:,nn)));
        qch7(:,nn)=imag((ir(:,nn+knd)+i.*qr(:,nn+knd)).*(ieq(:,nn)+i.*qeq(:,nn)));

        % Data demapping
        [ir1(:,nn),qr1(:,nn)]=demap_pilot(ich7(:,nn),qch7(:,nn),fftlen,para);

        % Demodulate data
```

```
demod=myqamdemod(ir1(:,nn),qr1(:,nn),para,1,m);

% Parallel to Serial
serdemod((para*m*(nn-1)+1):para*m*nn)=reshape(demod,1,para*m);

% de-interleave with 12x16 block
hard((para*m*(nn-1)+1):para*m*nn)
    =deinterleave(serdemod((para*m*(nn-1)+1):para*m*nn),para*m,12,16);

% Hard Decision Viterbi Decoding
    if nn==1
        [hdec,fme,fst,fin]=vitdec(hard((para*m*(nn-1)+1):para*m*nn),
                                        trellis,window,'cont','hard');
        last=vitdec(encoded(para*m*nn+1:para*m*(nn+1)),
                                trellis,window,'cont','hard',fme,fst,fin);
    else
        [hdec,fme,fst,fin]=vitdec(hard((para*m*(nn-1)+1):para*m*nn),
                                trellis,window,'cont','hard',fme,fst,fin);
        last=vitdec(encoded(para*m*nn+1:para*m*(nn+1)),trellis,window,
                                            'cont','hard',fme,fst,fin);
    end

houtput((para*m*Rc*(nn-1)+1):para*m*Rc*nn)
                        =[hdec(window+1:length(hdec)) last(1:window)];

%%%%%%%%%%%% STAGE 1 DETERMINE DESIRED SIGNAL %%%%%%%%%%%
    [hencoded_S1 fs]
      =convenc(houtput((para*m*Rc*(nn-1)+1):para*m*Rc*nn),trellis,fs);

    % frequency interleaving 12x16
    hencoded2_S1=interleave(hencoded_S1,para*m,12,16);

    % convert serial data to parallel data
    hparainput_S1=reshape(hencoded2_S1,para,m);

    % QAM16 modulation
    [himod_S1,hqmod_S1]=myqammod(hparainput_S1,para,1,m);

    % data mapping
    [himap_S1,hqmap_S1]=map_pilot(himod_S1,hqmod_S1,fftlen,para,1);

    % calculating initial reverse rotation
    ieq_hat_S1(:,nn)=real((1./(ir(:,nn+knd).^2+qr(:,nn+knd).^2))
           .*(himap_S1+i.*hqmap_S1).*(ir(:,nn+knd)-i.*qr(:,nn+knd)));
    qeq_hat_S1(:,nn)=imag((1./(ir(:,nn+knd).^2+qr(:,nn+knd).^2))
           .*(himap_S1+i.*hqmap_S1).*(ir(:,nn+knd)-i.*qr(:,nn+knd)));

    ieq_S1(:,nn)=ieq(:,nn).*(gamma)+(1-gamma).*ieq_hat_S1(:,nn);
    qeq_S1(:,nn)=qeq(:,nn).*(gamma)+(1-gamma).*qeq_hat_S1(:,nn);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STAGE 2 - COMPENSATE BY CURRENT ESTIMATE %%%%%%%%%%%%%%%%%%%%%
ich7_S2(:,nn)=real((ir(:,nn+knd)+i.*qr(:,nn+knd)).*(ieq_S1(:,nn)
                                                +i.*qeq_S1(:,nn)));
```

```
            qch7_S2(:,nn)=imag((ir(:,nn+knd)+i.*qr(:,nn+knd)).*(ieq_S1(:,nn)
                                                         +i.*qeq_S1(:,nn)));

            % Data demapping
            [ir1_S2(:,nn),qr1_S2(:,nn)]=demap_pilot(ich7_S2(:,nn),
                                              qch7_S2(:,nn),fftlen,para);

            % Demodulate data
            demod_S2=myqamdemod(ir1_S2(:,nn),qr1_S2(:,nn),para,1,m);

            % Parallel to Serial
            serdemod_S2((para*m*(nn-1)+1):para*m*nn)=reshape(demod_S2,1,para*m);

            % de-interleave with 12x16 block
            hard_S2((para*m*(nn-1)+1):para*m*nn)
              =deinterleave(serdemod((para*m*(nn-1)+1):para*m*nn),para*m,12,16);

            % Hard Decision Viterbi Decoding
                if nn==1
                    [hdec_S2,fme_S2,fst_S2,fin_S2]
                        =vitdec(hard_S2((para*m*(nn-1)+1):para*m*nn),trellis,
                                                  window,'cont','hard');
                    last_S2=vitdec(encoded(para*m*nn+1:para*m*(nn+1)),trellis,
                                window,'cont','hard',fme_S2,fst_S2,fin_S2);
                else
                    [hdec_S2,fme_S2,fst_S2,fin_S2]
                        =vitdec(hard((para*m*(nn-1)+1):para*m*nn),trellis,
                            window,'cont','hard',fme_S2,fst_S2,fin_S2);
                    last_S2=vitdec(encoded(para*m*nn+1:para*m*(nn+1)),trellis,
                            window,'cont','hard',fme_S2,fst_S2,fin_S2);
                end

            houtput_S2((para*m*Rc*(nn-1)+1):para*m*Rc*nn)
                    =[hdec_S2(window+1:length(hdec_S2)) last_S2(1:window)];

            %%%%%%%%%%%% STAGE 3 DETERMINE NEW DESIRED SIGNAL %%%%%%%%%%
                [hencoded_S3 fs_S2]
                    =convenc(houtput_S2((para*m*Rc*(nn-1)+1):para*m*Rc*nn),
                                                          trellis,fs_S2);

                % frequency interleaving 12x16
                hencoded2_S3=interleave(hencoded_S3,para*m,12,16);

                % convert serial data to parallel data
                hparainput_S3=reshape(hencoded2_S3,para,m);

                % QAM16 modulation
                [himod_S3,hqmod_S3]=myqammod(hparainput_S3,para,1,m);

                % data mapping
                [himap_S3,hqmap_S3]=map_pilot(himod_S3,hqmod_S3,fftlen,para,1);

                % calculating initial reverse rotation
                ieq_hat_S3(:,nn)=real((1./(ir(:,nn+knd).^2+qr(:,nn+knd).^2))
```

```
                    .*(himap_S3+i.*hqmap_S3).*(ir(:,nn+knd)-i.*qr(:,nn+knd)));
                qeq_hat_S3(:,nn)=imag((1./(ir(:,nn+knd).^2+qr(:,nn+knd).^2))
                    .*(himap_S3+i.*hqmap_S3).*(ir(:,nn+knd)-i.*qr(:,nn+knd)));

                ieq(:,nn+1)=ieq_S1(:,nn).*(gamma)+(1-gamma).*ieq_hat_S3(:,nn);
                qeq(:,nn+1)=qeq_S1(:,nn).*(gamma)+(1-gamma).*qeq_hat_S3(:,nn);
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             % Bit Errors S1
        soft_be=sum(abs(input(1:length(input)-para*m/2)-houtput));%bit errors
        sbe=sbe+soft_be; % total bit errors

        % Packet Errors
        if soft_be~=0
            spe=spe+1;
        end
        fprintf('fd=%gHz SNR=%gdB nloop=%g BE=%g PE=%g\n',fd(ff),
                                                SNR(ss),ii,soft_be,spe);

        % Bit Errors S2
        soft_be2=sum(abs(input(1:length(input)-para*m/2)-houtput_S2));
        sbe2=sbe2+soft_be2; % total bit errors after stage 2

        % Packet Errors
        if soft_be2~=0
            spe2=spe2+1;
        end
        fprintf('fd=%gHz SNR=%gdB nloop=%g BE2=%g PE2=%g\n',fd(ff),
                                            SNR(ss),ii,soft_be2,spe2);

    end

    ber_ofdm(ss,ff)=sbe/(para*n*Rc*m*nloops);
    per_ofdm(ss,ff)=spe/(nloops);

    ber_ofdm2(ss,ff)=sbe2/(para*n*Rc*m*nloops);
    per_ofdm2(ss,ff)=spe2/(nloops);

    fprintf(fid,'ber_ofdm(%g,%g)=%e;   per_ofdm(%g,%g)=%e;\n',
                   ss,ff,ber_ofdm2(ss,ff),ss,ff,per_ofdm2(ss,ff));
    fprintf(fid,'n=%g; s=%g; gamma=%g;\n',n,s,gamma);
    fprintf('BER=%e;  PER=%e;\t',ber_ofdm2(ss,ff),per_ofdm2(ss,ff));
    fprintf('n=%g; s=%g; gamma=%g;\n',n,s,gamma);

  end

end

fclose(fid);

toc;
```

```
bQAM16_DD5_g6_S2=ber_ofdm2;
pQAM16_DD5_g6_S2=per_ofdm2;
save DSRCcurves/bQAM16_DD5_g6_S2 bQAM16_DD5_g6_S2
save DSRCcurves/pQAM16_DD5_g6_S2 pQAM16_DD5_g6_S2

bQAM16_DD5_g6=ber_ofdm;
pQAM16_DD5_g6=per_ofdm;
save DSRCcurves/bQAM16_DD5_g6 bQAM16_DD5_g6
save DSRCcurves/pQAM16_DD5_g6 pQAM16_DD5_g6
% ***********************end of file**********************************
```

VITA AUCTORIS

Harb Abdulhamid was born in Kfarhamam, Lebanon, on December 9, 1982. He had immigrated to Canada in 1987 with his family. He received his B.A.Sc. degree in electrical engineering in 2004 from the University of Windsor. His Capstone Design Project was titled "Bluetooth Automotive Log-in System". He is currently a candidate in the electrical and computer engineering M.A.Sc. program at the University of Windsor. His research interests include adaptive signal processing, channel estimation, information and coding theory, field-programmable logic, computer arithmetic and VLSI circuit design.