EE-442: Wireless Receivers: Algorithms and Architectures

Fall 2022 Final Project Report

OFDM Audio Transmission System

Dong Chu,
Hai Miao,
Huan-Ying Yeh

supervised by
Andreas Peter Burg, Joachim Tobias Tapparel, & Sitian Li

Abstract

This project is a MATLAB-based implementation of an OFDM (Orthogonal Frequency Division Multiplexing) acoustic transmission system. Two audio speakers are used to transmit acoustic data bits and a microphone receives the data at a distance. The physical constraints of the microphone transmissions, such as low data transmission rate and susceptibility to fading and phase offsets, require various design techniques such as frame synchronization and phase tracking. Our receiver demonstrates good robustness to noise and interference, and it is able to achieve high data rates and spectral efficiency.

Keywords

OFDM; Wireless Device Algorithms; Audio Signal Transmission, MATLAB Simulations

Contents

1 Introduction

OFDM (Orthogonal Frequency Division Multiplexing) is a type of digital modulation technique that is widely used in wireless transmission systems. Multiple data streams can be sent over a single physical channel and eliminate inter-symbol interference (ISI).

The main advantage of OFDM is its ability to transmit data over a wide frequency band, which makes it resistant to interference and enables high data rates. In this project, we use MATLAB simulations and a two-channel speaker/microphone system to investigate the performance of our design. Performances are evaluated with key metrics such as data rate, spectral efficiency, robustness against noise, and bit error rate.

2 Problems to Solve

- Implement OFDM and phase tracking in two ways: Comb Pilot and Block Pilot.
- Compare the performance of Comb Pilot and Block Pilot OFDM with MATLAB simulations of various fading effects.
- Use the acoustic system to transmit images and ensure good received quality.
- Experiment with various pilot symbol parameters and evaluate the best design choices.
- Analyze the experiment results with statistics, plotting, and model comparisons.

3 Solutions Overview

With techniques such as phase-matching, frame synchronization, and pulse-shape filtering, we accurately estimated the channel in the presence of noise and time-variant fading channels.

The transmitter is a system of two speakers that sends the audio data of the input bits, in our case, an input image. Then, the two speakers' data transmission would face real-world physical constraints such as fading and phase offsets, which would be optimally recovered on the receiver, a microphone, with our OFDM implementation.

Our transmission system is evaluated through simulations under different scenarios, such as signal-to-noise ratio (SNR) vs. bit error rate (BER) statistics, comparing the received constellation of various training methods, and observing experiment results under different microphone setups.

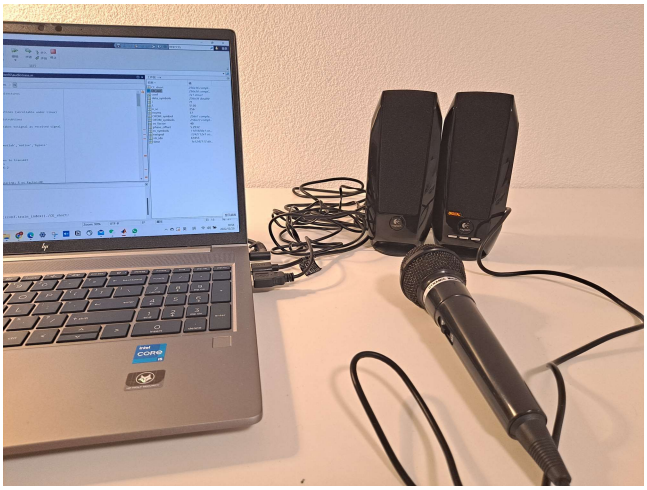


Figure 1: The experiment setup with a laptop running the MATLAB scripts, two speakers transmitting the audio signal, and a microphone recording the signals.

4 Implementation

4.1 Audio Transmission Framework

The key implementation points are:

- A "conf" struct for audio transmission configurations, storing the sampling rate, number of frames, number of bits (2000), modulation order (BPSK or QPSK), etc. According to the project instructions, the default number of sub-carriers is set to 256, and the length of the cyclic prefix is half of that of one OFDM symbol.
- 100 bits are allocated for the preamble and 16 bits per audio sample.
- Note that the sampling rate must be a multiple of the symbol rate. Therefore, the code has error-handling on this section.

After the initial audio transmission parameters are set (frame sizes, number of frames, modulation orders, etc.), a 44 x 51 gray-scale image is loaded and passed into the tx() transmit function, which are implemented in two ways: Comb Pilot and Block Pilot.

Then, the user chooses between three audio system modes: "bypass", "matlab", and "native". In the "bypass" mode, no audio transmission is performed, and the transmitted signal is taken as the received signal. This allows the user to test the receiver without actually transmitting any data. With the other two modes, we set up the respective audio recording parameters so that the speakers can be used properly.

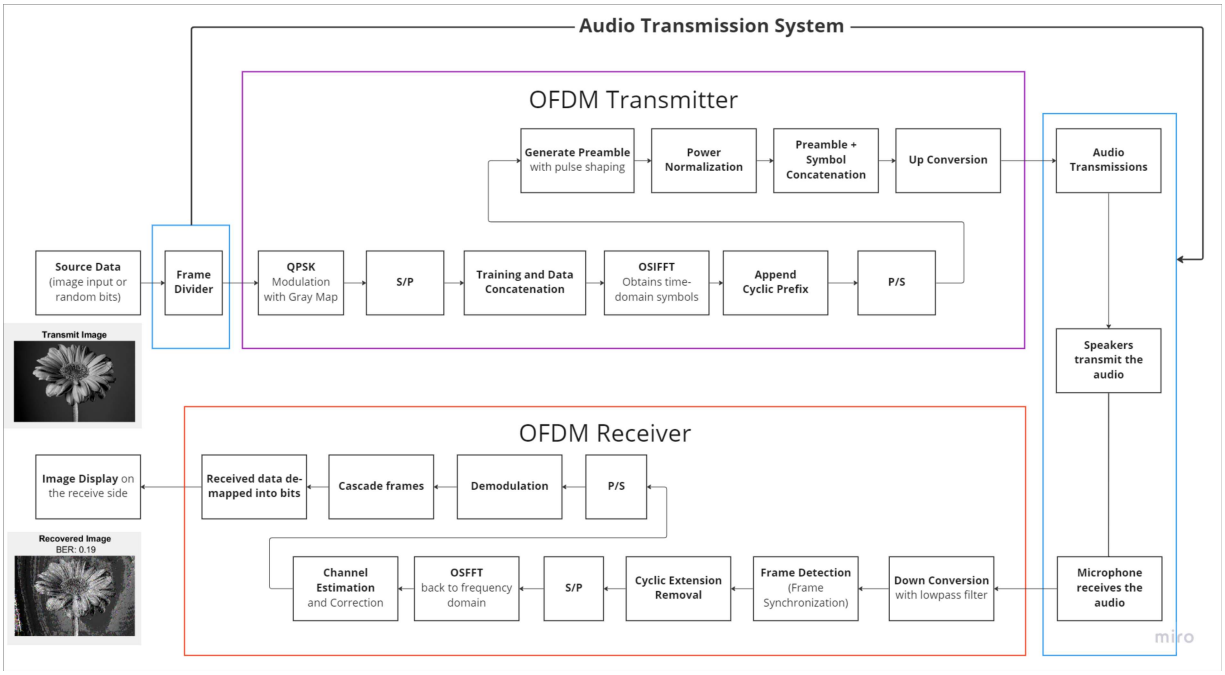


Figure 2: System Block Diagram

As shown in Fig. 2, the system consists of three parts: the OFDM transmitter(purple), the hardware audio transmission equipment (blue), and the OFDM receiver (orange). The transmitter and receiver algorithms are implemeneted in MATLAB. The main goal is to transmit images with the presence of noise and fading.

4.2 Transmitter

The complete transmitter function tx() takes the input vector bits, the config struct defined in the audio transmission script, and the frame index number. Then, it outputs the transmitted signal for the speakers. The main operations include:

- Modulating into pairs of QPSK symbols
- Padding the last incomplete OFDM symbol with random QPSK symbols. This solves the problem of data not being divided equally into symbol chunks
- Splitting serial bits into parallel streams
- Generating BPSK training symbols and concatenating with the data
- Performing IFFT on the OFDM symbols and appending a cyclic prefix
- Split from parallel to serial

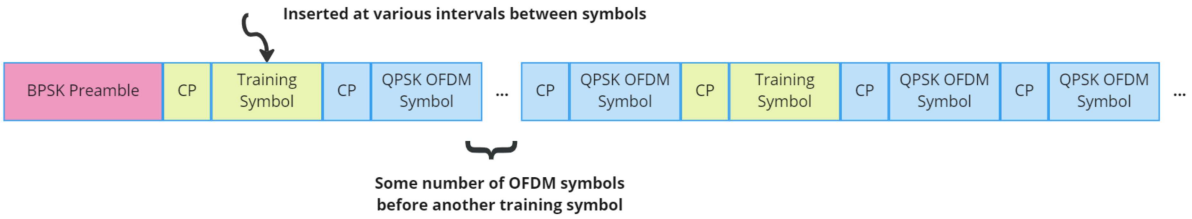


Figure 3: Schematic of the Symbol Blocking Design: Pilot symbols are used for continuous phase-tracking

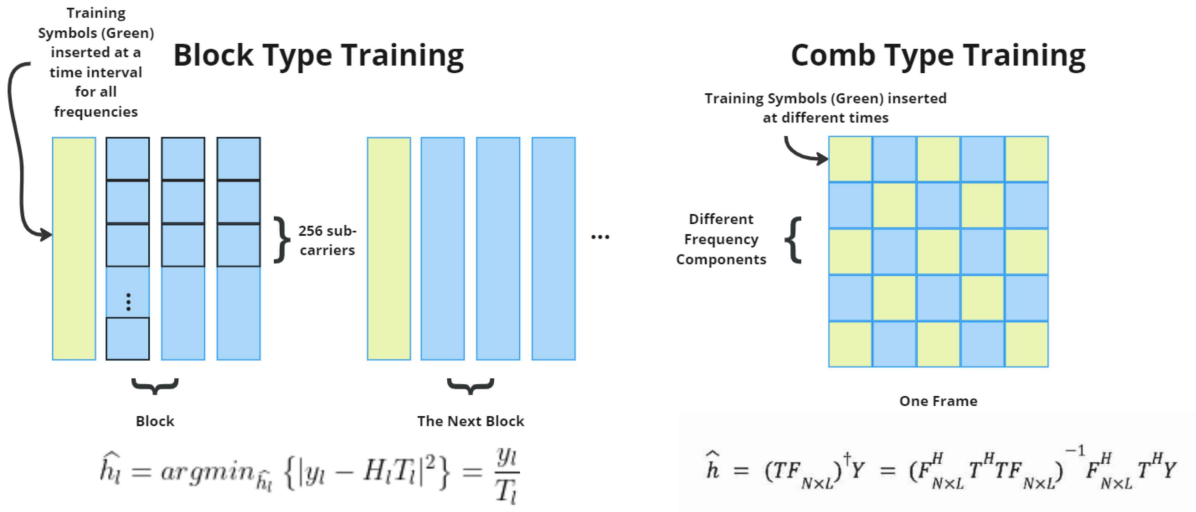


Figure 4: Schematic of Block vs. Comb-Type Training Symbols

- Up-sampling the preamble and concatenating with time-domain OFDM symbols
- Filtering with a low-pass filter to extract the base-band signal

Fig.3 shows a schematic of the transmitted symbol. There is a 100-bit BPSK preamble, cyclic prefix and training symbols inserted at certain intervals (the effects of training symbol insertions will be discussed later), and the QPSK data blocks. The cyclic prefix allows the FFT to be calculated easily and ISI to be avoided; the training symbols, generated with block-type and comb-type, allows the channel gain and phase to be detected at regular intervals of the transmission.

The phase-tracking training methods of the Block Type and Comb Type are shown in Fig.4, which estimates the time-varying channel by comparing the output with known "pilot" inputs (marked in yellow). With the block type, the pilot data occurs at certain time intervals and span across all frequencies. With the comb type, pilot data is inserted in between data packets throughout selected frequencies and times, allowing more consistent "anchors" for channel estimation throughout the transmission.

By changing the pilot data insertion intervals, the length of the preamble, and the training type, we achieve different data transmission efficiency, i.e. the ratio between amount of data transmitted and amount of resources available.

4.3 Receiver

The rx() function implements a complete casual receiver in the digital domain. Using the config struct and the information about frame sizes, over-sampling factor, and channel length, the following steps are carried out to get the received bits from the received signal:

- Down conversion and lowpass filter
- Frame synchronization to find the signal starting point
- Removing the cyclic prefix
- Converting from serial to parallel and transition back to frequency domain with OSFFT()
- Removes channel effects by dividing the OFDM symbols with the estimated $\hat{h}$.
- Parallel to serial conversion
- Demodulation: remove training samples and extra padding, then identify the received bits by determining the closest constellation point that it belongs to.
- Returns the updated config struct and the received bits.
- Displays the output image and the BER of the transmission.

5 Simulations

5.1 Constellation Comparisons

To get an intuitive sense of how well our transmitter receiver implementations worked, we passed in an input image with the QPSK modulation scheme, added simulated noise and fading with AWGN and Exponential Decay Fading with different number of taps.

The lower the SNR, the more noisy the signal. The higher the number of taps, the more fading is applied to the channel.

From Fig.6 and Fig. 7, we see our implementations working as intended. The input image bits are mapped onto QPSK with Gray Mapping. Then, if there's no tracking or phase estimation

```
% Apply fading channel
g = exp(-(0:n_taps-1));
g = g/norm(g);
x_s_noise_fading = conv(x_s_noise,g,'same');
```

Figure 5: Code Snippet for Implementing the Exponential Fading Channel

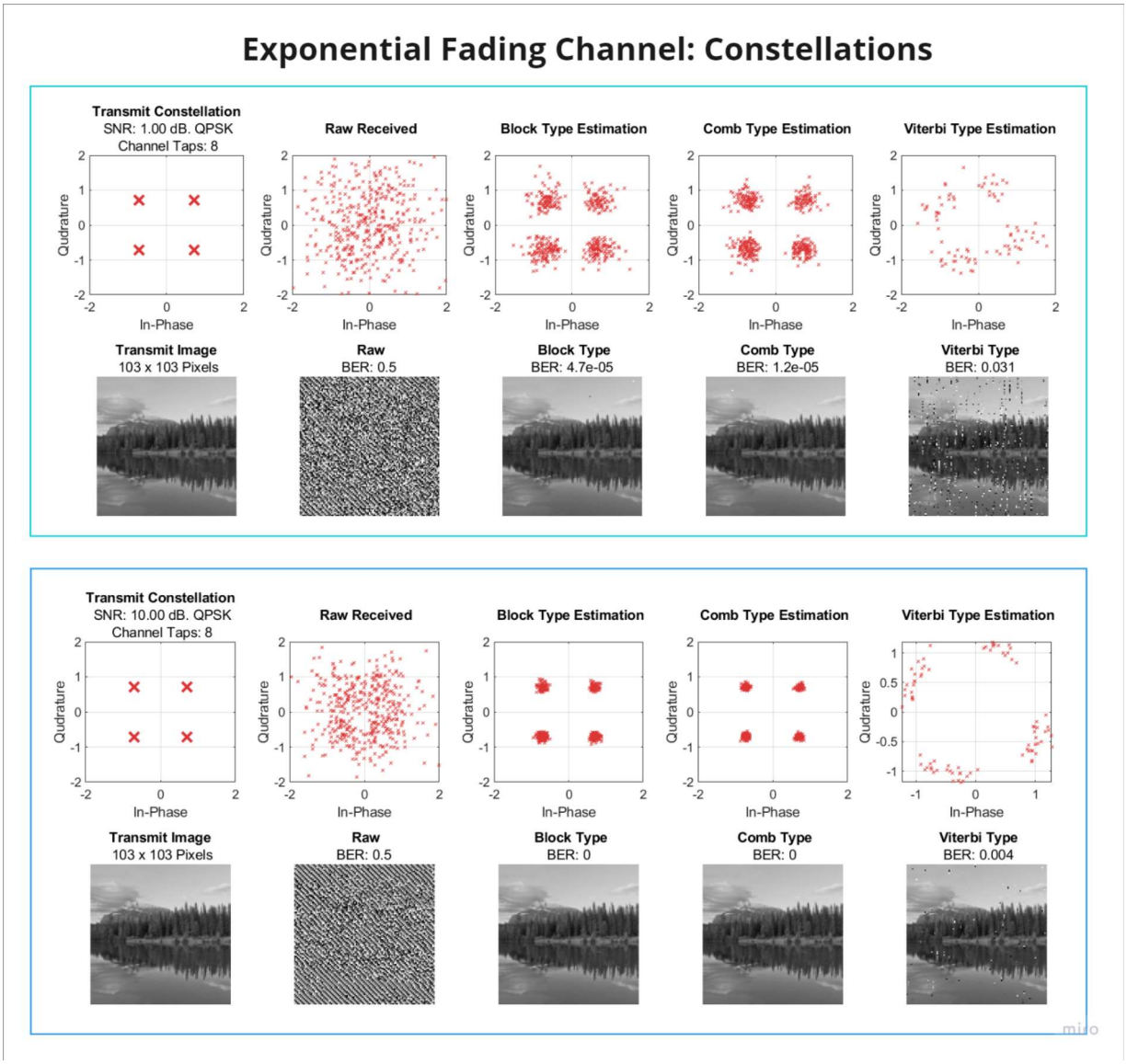


Figure 6: Input vs. Output Constellations of Different Training Types: Exponential Decay Fading Channel

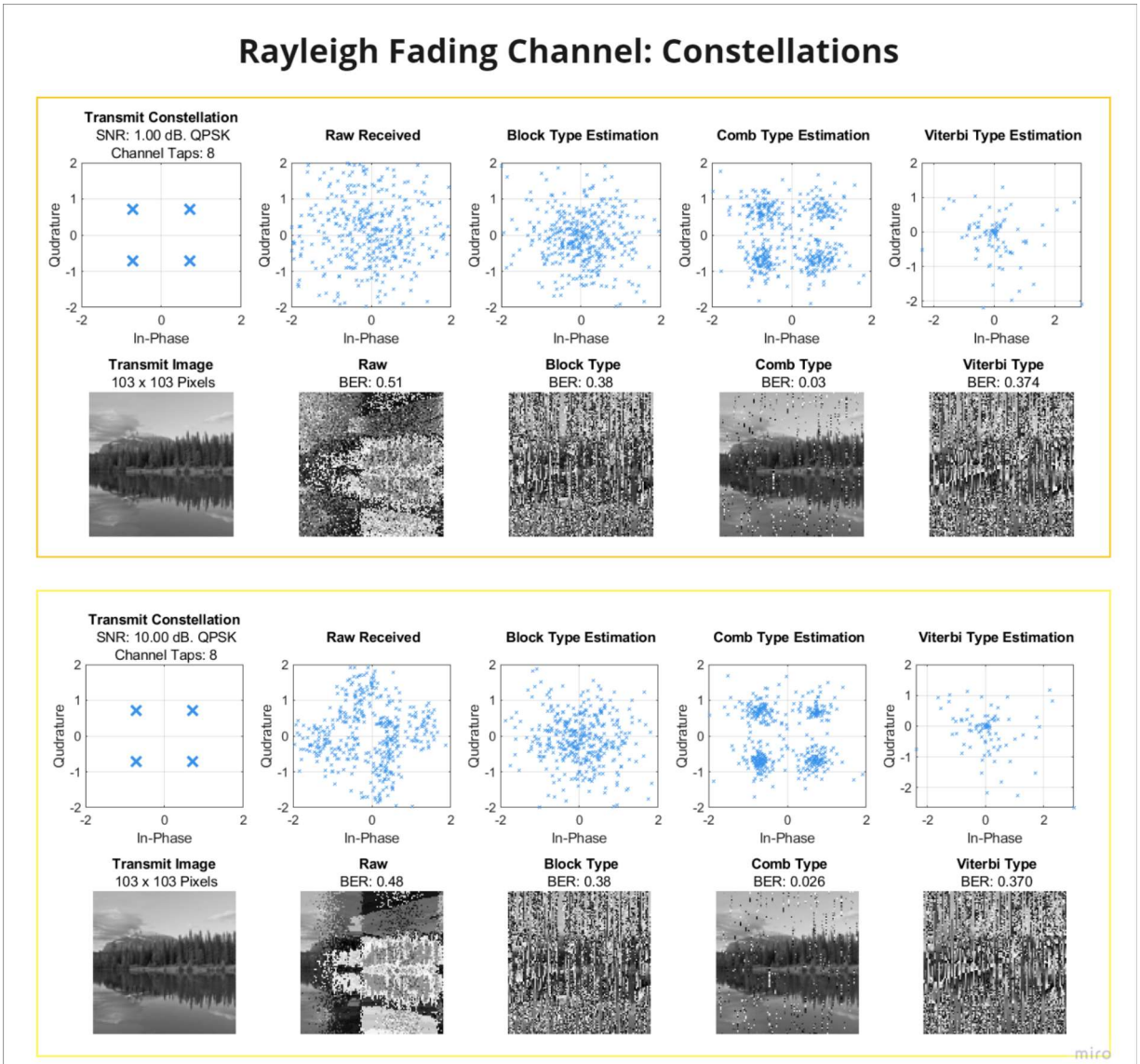


Figure 7: Input vs. Output Constellations of Different Training Types: Rayleigh Fading Channel

(column 2), the BER is very high and the received constellation points are heavily rotated due to fading.

Column 3 through 5 show our three implementations for phase tracking: Block type, comb type, and Viterbi Viterbi. Comb type tracking has the best performance in both Exponential fading and Rayleigh Fading, achieving low BER and high output image qualities under heavy noise. Viterbi Viterbi (column 5) doesn't seem to be working with a Rayleigh fading noise, which will be discussed in the appendix.

```
% Rayleigh Channel initialization
sampleRate  = 48000;     % Sample rate of 20K Hz
maxDopplerShift  = 1; % Max Doppler shift of diffuse components (Hz)
delayVector = [1 2]*1e-8; % Discrete delays of four-path channel (s)
gainVector  = [-1 -1];  % Average path gains (dB)
```

Figure 8: Tunable Parameters in the Rayleigh Fading Object

5.2 Rayleigh Fading

In order to simulate Rayleigh Fading, the MATLAB comm.RayleighChannel object is used. The object simulates a channel with various delay, doppler effects, and gain changes. The sample rate is the same as our audio system, which is 48000.

5.3 Robustness to Various Strengths of Noise and Fading

After verifying that our phase-tracking implementations work, we simulated more noise and fading channel parameters and compared model performances. Mainly focusing on the Block type and Comb type systems, the following metrics are tested for parameters vs. BER:

- SNR (dB): with large delays of 0.01 and 0.2 seconds and 1Hz of Doppler fading, we vary the SNR of the signals. Comb type performs much better than block type throughout every SNR value from 0.01 to 100. Both types' BER slightly decrease with larger SNR, which makes sense. However, the limited amount of accuracy improvements is likely due to the high amount of fading and delay of the channel, which causes phase distortions beyond the correction ability of high SNR.

- Doppler Fading Frequency (Hz): The relative motion between the transmitter and receiver introduces Doppler Shift. When we move the two speakers back and forth during experiments, we see Doppler fading of various strengths that increase the difficulty of channel estimation.

- Channel delay time (s) : We selected Rayleigh channel delay times to be between 0s to 0.0001s, because higher delay times would cause the simulation code to timeout and crash. However, we see a very clear trend of increasing BER for both Block type and Comb type, although both at low rates overall.

- Flat channel SNR: the exponential PDP comes from the log distance path loss model where

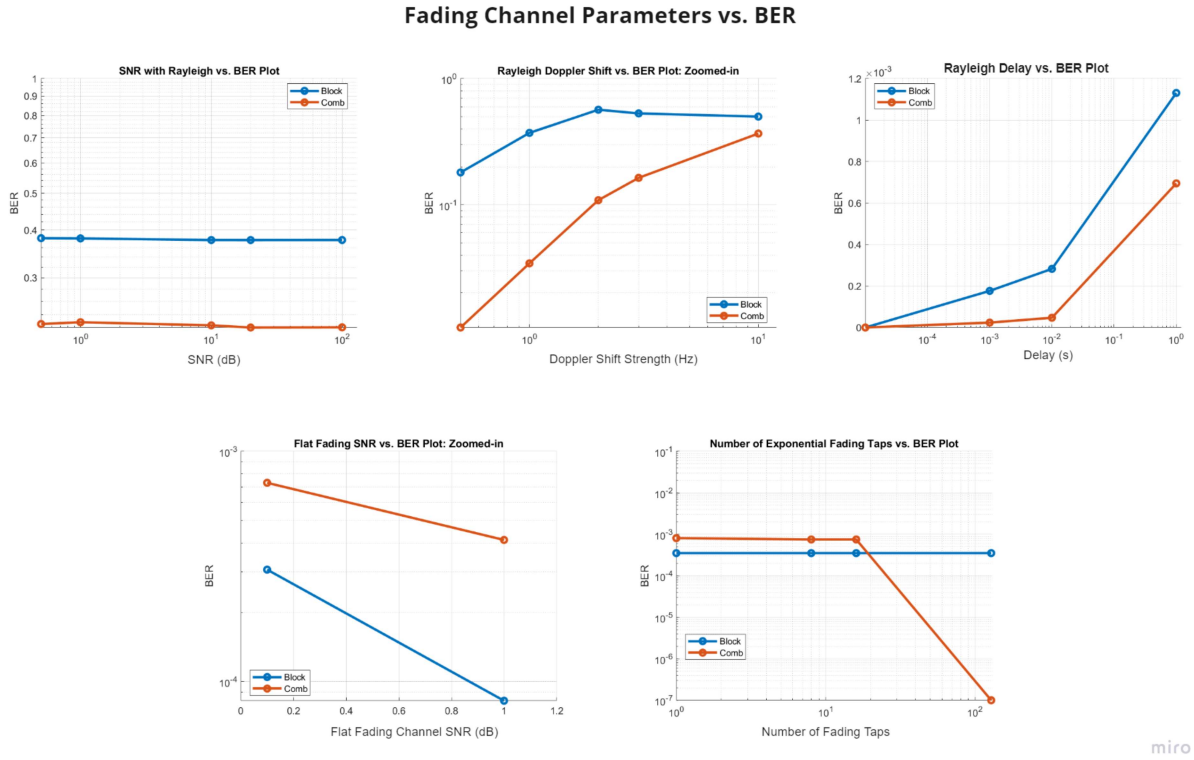


Figure 9: BER Performances Under Various Fading Channel Parameters. a) Rayleigh Channel: SNR vs. BER ranging from 0.01 to 100dB. b) Rayleigh Doppler Shift amount vs. BER. c) Rayleigh delay vs. BER. d) Exponential fading SNR vs. BER. e) Number of exponential fading taps vs. BER.

the power in decibel decrease approximately linearly with time, a phenomenon that arises from real-life measurement. Here, block type training has better performance than comb type, probably due to the more predictable channel that is best estimated for all frequency components at certain time intervals, rather than sparingly probing at selected frequencies and estimating the changes. Higher SNR does improve BER to some extent, but even higher values did not cause significant improvements.

- Number of exponential fading taps: more taps correspond to larger extent of fading. Here, we did not see a clear trend on the model performances when the number of fading taps increased from 2, 4, 8, to 128. The BER are all quite low but showing some irregular trends with respect to tap numbers. Block type performance was consistent, while Comb type, perhaps due to coincidence, had BER = 0 at taps = 128.

5.4 MATLAB Simulation Conclusions

Overall, the Comb type transmission system seems to perform better than Block type one, as for three out of five experiments above, the Comb type BER curve (red) is below the Block type curve (blue). Nonetheless, both systems are extremely robust to channel distortions.

Note that in this project, we also attempted to implement the Viterbi decoder. The Viterbi decoder corrects errors in an OFDM code by using the known error-correcting code to determine the most likely data bits sequence that was transmitted. The simulation results in Viterbi are shown in the appendix. However, after many hours of debugging and re-designing, our implementation doesn't seem to align well with Block Comb type systems, suggesting future work and discussions.

Viterbi solves for the most likely sequence based on the observations at each current block, assuming that the channel distorts the data in a time-varying manner. Therefore, it will naturally have worse performances on a static channel (such as flat fading channel and multi-path fading channel) because it will take all the noise variations into account and estimate different channels constantly, when the channel, in reality, does not change over time. We can see this from Fig. 6, where Viterbi applied to a static noise channel (exponential fading channel) actually made it perform worse than Block pilot.

6 Hardware Experiments

This section discusses hardware experiments with the speaker and microphone system, when we send an audio through Block and Comb-type rx() function, then record audio with a microphone to recover the transmitted bits with the tx() function.

6.1 Microphone Movements

We transmitted a gray-scale image of size 44 by 51 pixels. Note that signal durations are slightly different for block pilot and comb pilot. The lengths of randomly generated data padded in the last OFDM symbol are different because we insert training symbols differently for the previous blocks.

To examine the ability of our system to estimate the channel, we move the microphone in different ranges of motion, which varies the amount of noise and fading distortions. To control the variables on the background noise, we played the same piece of music for every transmission.

Input: 44 x 51, gray-scaled uint8 image

Audio duration: around 25 seconds

Noise： a repeated piece of music

Microphone	BER (Block)	BER (Comb)
No Movement	0.0003	0.0
Slight Movement	0.0432	0.0013
Strong Movement	0.3215	0.0549

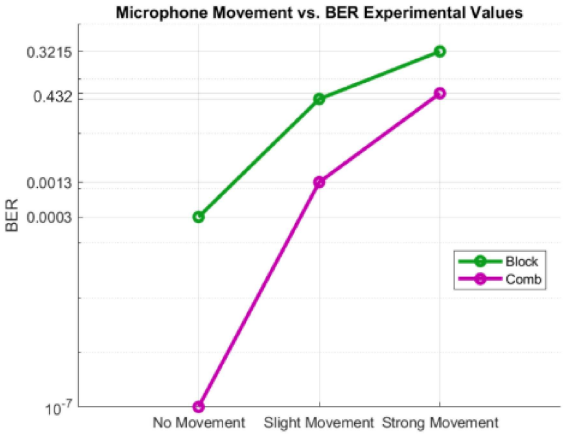


Figure 10: Microphone Movement Ranges vs. Transmission System Performances

When the microphones were stable throughout the transmission, both block and comb pilot show great performance, having BER that are close to 0. But when the movement get strong, the block pilot deceases to 0.3 while the Comb pilot shows only 1 sixth of the BER.

Apart from the BER, we illustrate the received constellation and the reverted image. The received signal is corrected by the estimation channel. Consistent with the BER performance, the boundary of QPSK constellation that should be distributed in the four quadrants is getting blurry. It should be noted that, for experiment with strong movement, the constellation seems to be squeezed into a center. But if we zoom in a bit, it is because of the outliers of some constellation point that has extremely large magnitude.

6.2 Training Symbol Insertion Rates

Above demonstration, the position of inserted training symbols is fixed to be 1 to 1. e.g. In block pilot, one training symbols is inserted after each data symbol. In comb block, it means 128 evenly distributed sub-carries are taken by the training symbols. According to the results, it's clear that the comb pilot more robust against the proportion of inserted training symbols. Even when only one fifth of the overall sub-carriers are training information, the BER is still less than 1%. The training symbol insertion rate vs. performance plot shows that the block pilot performance deteriorates more when one training symbol needs to estimate the channel for a longer duration.

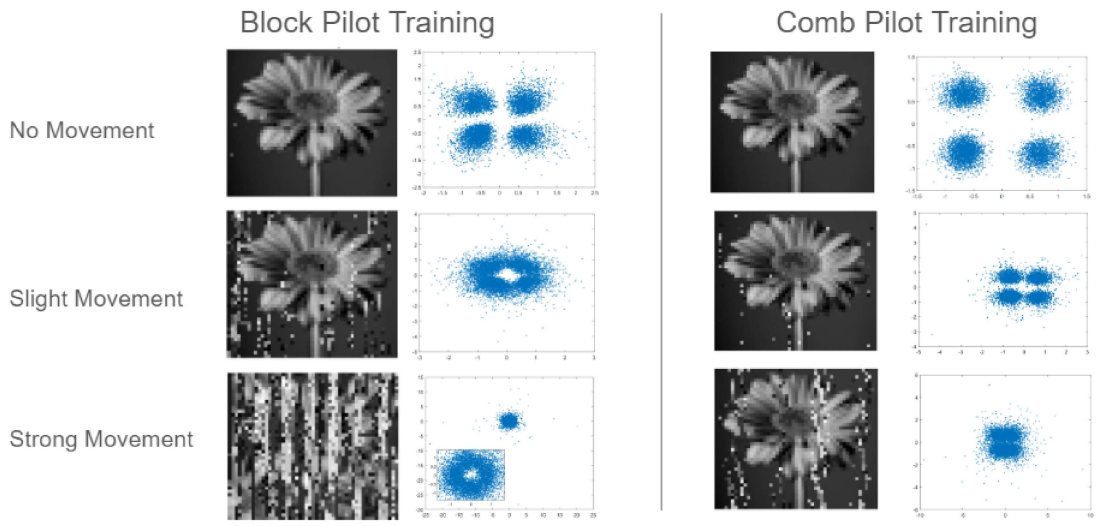


Figure 11: Hardware experiments: received constellations and recovered images of block-type and comb type transmission systems.

Train-Data Ratio: (number of train sub-carriers : data sub-carriers)

Microphone were slightly moving

Train-Data Ratio	BER (Block)	BER (Comb)
1:1	0.0432	0.0013
1:2	0.2931	0.0021
1:4	0.4707	0.0076

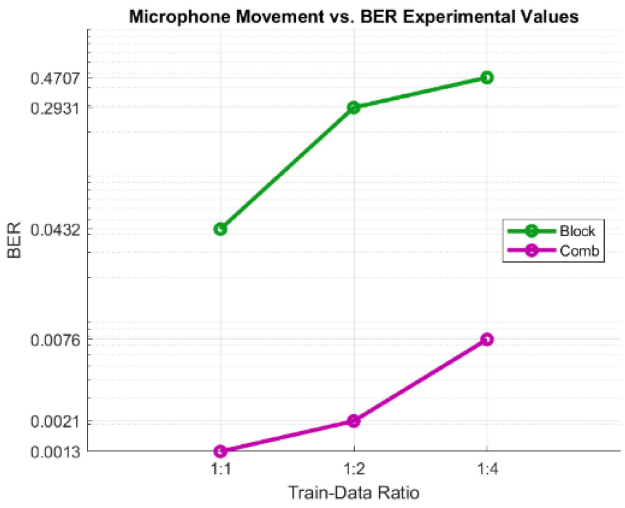


Figure 12: Hardware experiments: received constellations and recovered images of block-type and comb type transmission systems.

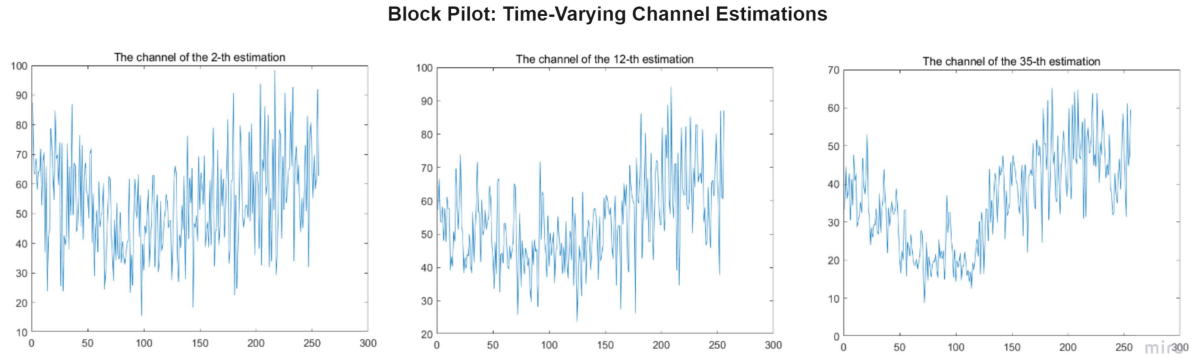


Figure 13: Hardware experiments: block pilot receiver channels at different frames.

6.3 Real-Time Channel Estimations

To verify that the estimated channels do change over time, several channel estimations in the frequency domain are shown in Fig. 13 and Fig. 14. Here, we used train-data ratio as 1 to 1 and moved the microphones slowly. In the block pilot, 36 training symbols were inserted.

For comb pilot, the channel length is tunable. We show two examples of a short and a long channel with lengths of 6 and 80, respectively. A channel estimate is smoother for shorter channel length, as it has fewer anchors to interpolate upon. However, as the channel length increases to 80, we might have over-defined the channel estimate . Therefore, when we increase the length of the channel, there are more spikes in the channel spectrum.

During our empirical experiments, we notice that there is not significant different of BER as long as the channel length is longer than 3. Therefore, we now examine channel in the time domain and use RMS delay spread.

6.4 RMS Delay Spread

The root mean square (RMS) delay spread of a wireless channel reflects the dispersion of the channel's impulse response. It is defined as the square root of the variance of the channel's delay profile, typically measured in seconds. The RMS delay spread determines the coherence time of the channel (or effective channel length), which is the duration over which the channel remains relatively unchanged.

Here the center of delay is not the simple average, but is weighted by the channel magnitude. The center would be close the first channel component. We define x-tau as the effective channel distance here, which is calculated by multiplying delay spread by the speed of sound. By exam-

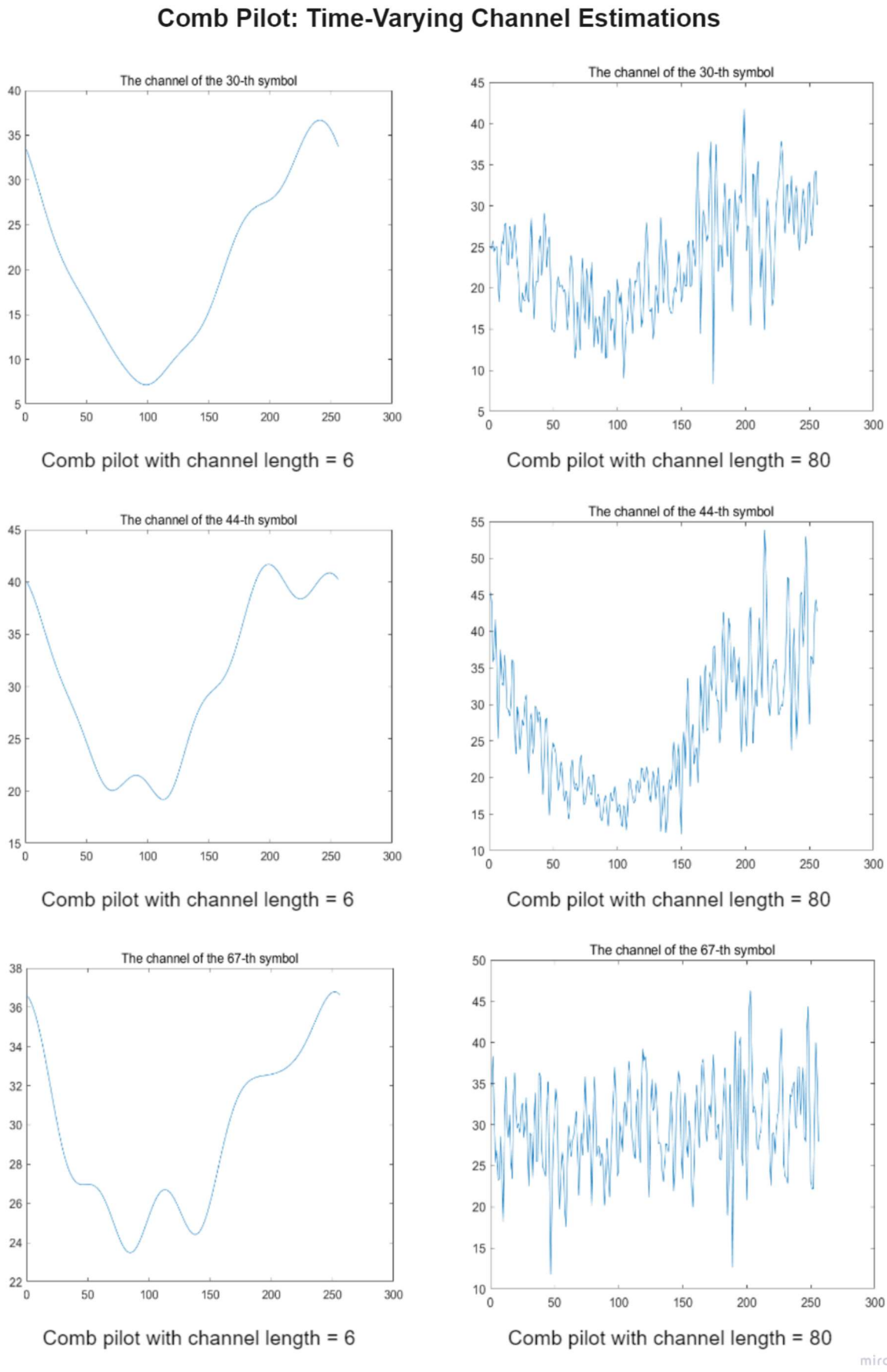


Figure 14: Hardware experiments: comb pilot receiver channels at different frames. Channel length = 6 (left column) and channel length = 8 (right column).

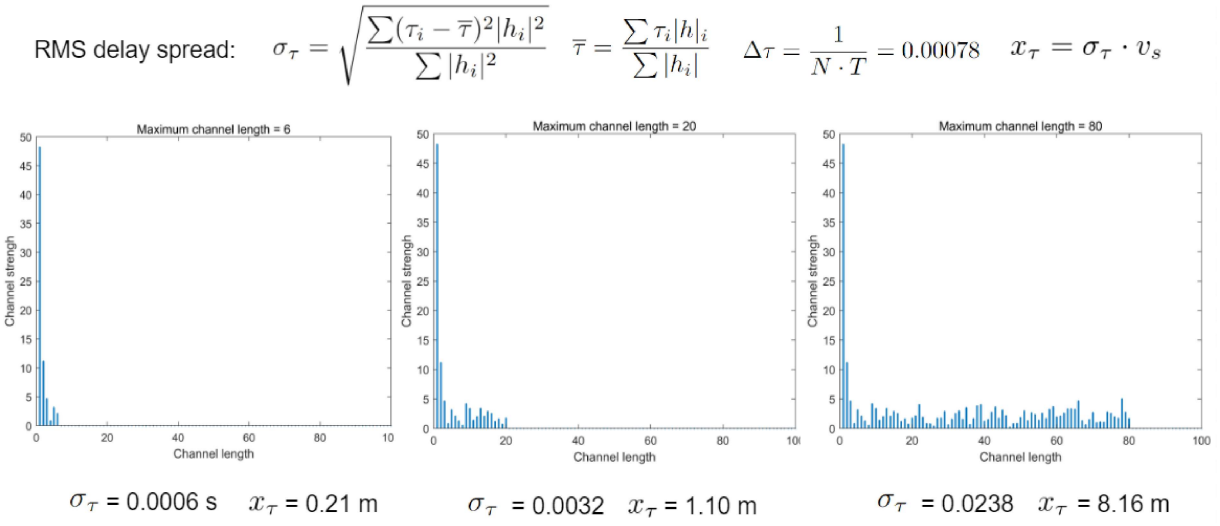


Figure 15: Hardware experiments: Delay spread calculations and channel lengths of 2, 6, and 80.

ining different channel lengths, 6, 20, and 80, we believe that the estimation could correct the deterioration caused by physical channels that are around the same or shorter lengths than the effective distance. However, in our case, all three channel length show great performance. This is because the powers of longer channels are small and could hardly affect the channel estimation.

7 Optimal System Design

From the simulations and experiments, we found that the acoustic system with two speakers and a microphone would transmit data bits best with a comb type pilot configuration and the following default design parameters:

OFDM Comb-Type Optimal Design Parameters	
Parameter Name	Value
Sampling Rate	48000
Symbol Rate	100
Modulation Order	QPSK
Carrier Frequency	8000
Number of Sub-Carriers	256
Sample Spacing	4.6875
Oversampling Factor	40
Length of Preamble	100
Cyclic Prefix Length	5120
Comb-Pilot Insertion Interval	2

Table 1: OFDM Small Model Default Parameters (Control Variables).

This configuration ensures that 1) the system is robust against various levels of noise and

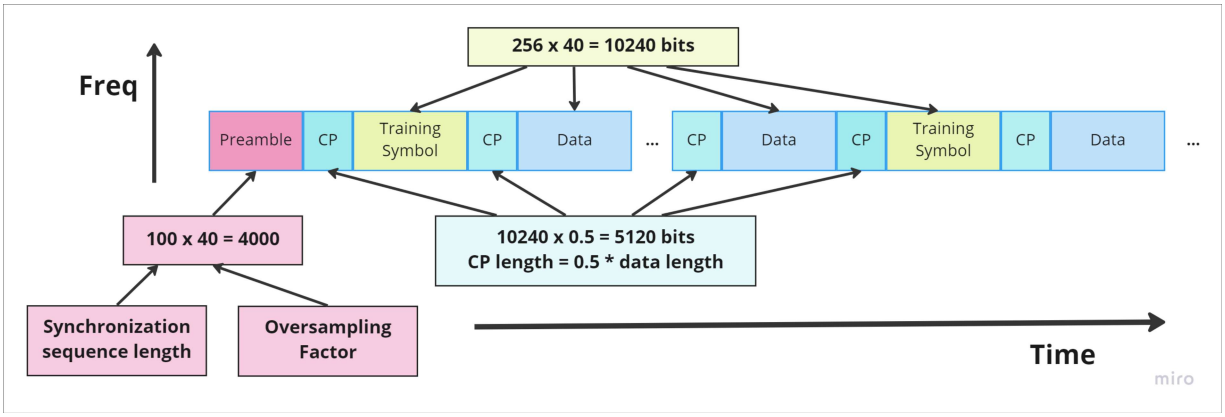


Figure 16: Efficiency Calculations: Size of data chunks following the optimal design parameters

fading, as generated from additional movement of the microphones 2) in simulations, the received constellations and BER also indicate good performance against exponential fading and Rayleigh fading 3) the transmission efficiency is relatively high.

7.1 Efficiency Calculations

We follow the following steps to calculate the efficiency of this design, which is the ratio between the amount of actual data transmitted and the amount of resources available, both in the time- and frequency-domain. According to our data symbol block diagram, the numbers used for calculations are illustrated in Fig. 16.

Since the oversampling factor of 40 is applied to all parts of the data in the numerator and denominator, we omit multiplying the data length by 40 throughout. The calculations are as follows:

The size of preamble is 100 bits. We send 1 preamble every transmission.

In the simulation example with 103 x 103 gray scale image. We over-sample by 40, so the total number of transmitted bits is:

$$103 * 103 * 8 = 84872\ bits$$

In QPSK, this implies this number of data packets, where each of the 256 sub-carriers can transmit 2 bits. For approximating the efficiency, we assume that the number of packets are divisible by 3, allowing a whole-numbered amount of "training + data + data" combinations to be sent.

We also add padding to make sure that the original bit stream is zero-padded and divisible by 256.

$$84872/(256 * 2) \approx 165$$

note that symbols are inserted at $1 : 2$ ratio with the actual data, the total length of the transmitted data packet would be 1.5 times longer than the data alone. In addition, the cyclic prefix is set to half of the length of the data packets, which also extend the overall tx() packet length of 1.5 times the original.

Combining all above steps, we have:

$$\frac{(\frac{picture\ bits\ in\ total}{256*2})}{((\frac{picture\ bits\ in\ total}{256*2} * \frac{1}{3}) + \frac{picture\ bits\ in\ total}{256*2}) * 1.5 + 4000}$$

$$= \frac{165}{(165 * \frac{1}{3} + 165) * 1.5 + 100}$$

$$= 0.384 = 38.4\%$$

Since the block type and comb type used the same numbers of training symbol insertion intervals and other metrics, the efficiency for both implementation are around the same. The overall efficiency is then around 38%.

The trade-off between transmission efficiency and accuracy is evident. Because we are using simple acoustic equipment as the wireless carriers, there are many unexpected fading effects that makes the channel hard to locate and estimate. In order to ensure low BER and robustness, we traded off accuracy with efficiency, adding long cyclic prefix and short pilot insertion intervals.

8 Problems Encountered and Solutions

8.1 Indivisible Data Chunk Lengths

Throughout the project, there were multiple instances where the lengths of the arrays were not divisible by the desired number. WE debugged the issues by

8.2 High Variability with the Physical Environment

The microphone and speaker setup are susceptible to environmental noise, where BER comparisons were not reliable for evaluating the model performances. However, we get a general sense of the channel estimation effectiveness by experimenting with the following:

- Setting up the microphone close together and not moving the microphone
- Slowly move the microphones to add more fading and phase shifts
- Ensure that all implementation modes achieve good BER at static setup (BER < 0.05).

This helped us debug obvious issues, such as using the wrong number of padding for frame synchronizations or forgetting to plug in the microphone.

Later, we also refined the code base and implemented the comm.RayleighChannel object in MATLAB to simulate transmission noise and fading in a controlled manner. This allowed us to run many experiments and thoroughly test the block vs. comb-pilot performances.

8.3 Long Transmission Times of Images

As the transmission rate of the microphones is very limited through audio, we were unable to transmit large images. If an image's size is 250 x 250 pixels, the transmission would take around 50 seconds to complete.

To be able to evaluate results quickly, we selected gray-scale binary images scaled to around 50 x 50 pixels. Then, each experiment trial took about 10s.

8.4 Viterbi Viterbi

This algorithm assumes constant gain and small differences between subsequent symbols. They also assumed normal distributions in the phase constellation.

However, our system does not adapt to Viterbi as well as expected. See the appendix for its simulation results.

9 Discussion

In this study, we implemented and examined the performance of two pilot insertion methods for an Orthogonal Frequency Division Multiplexing (OFDM) wireless transmission system: block pilot and comb pilot. We transmitted images with audio signals in a speaker and microphone setup, comparing the BER and system robustness under various channel conditions and strengths of fading.

In terms of model performance, we found that the comb pilot method provided lower Bit Error Rate (BER) compared to the block pilot method. However, comb pilot has higher overhead

and reduces the overall data capacity of the system. However, the comb pilot method was able to provide more accurate channel estimates, which are beneficial in channel conditions that are highly variable (such as Rayleigh fading with the Doppler effect).

Overall, both pilot insertion methods showed promising results in both the MATLAB simulation and in the physical experiments. Further work includes: 1) Better control the variables for the fading channel parameters and systematically check value ranges that are best suited for model testing. 2) Tune more parameters for the transmission system, such as symbol rate, spacing, and DPSK modulation scheme to learn more about the performances. 3) Design and optimize the systems for the highest efficiency possible.

10 Appendix

10.1 Additional Simulations

In addition to the discussions above, we also ran some simulations during code development to learn more about the OFDM transmission system. However, as some of these divert from the main objective of the project, the results are put in the appendix here.

10.1.1 More Modulation Schemes

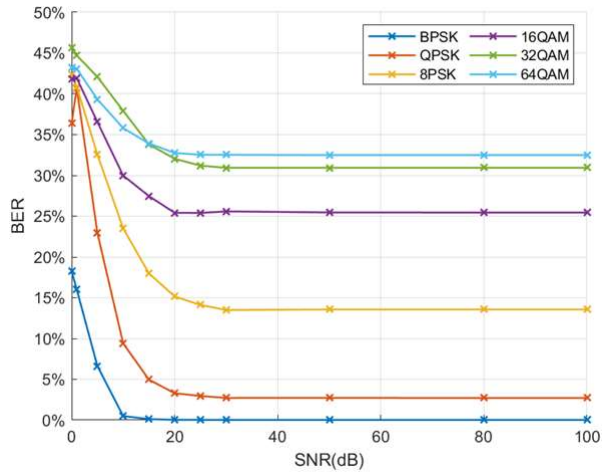
First, we experimented with more modulation schemes and measured the BER and code run-time (Fig. 16 and Fig. 17). These results make sense because as we map input bits onto more constellation points, the more susceptible they are to noise and fading. It would be easier for a symbol to be classified incorrectly at the demodulation step.

We also see decreases in run-time as the DPSK modulation order increases, due to the fact that mapping more data bits into symbols at once would increase the symbol rate and speed up transmission. However, this means higher BER overall, as seen in Fig. 16.

10.1.2 Viterbi Simulations

Here, we explored another trade-off. We could implement our OFDM system with higher modulation orders; however, from the experiments, QPSK turned out to be the most reasonable and efficient choice.

Second, we ran the fading channel parameter tests on the proposed Viterbi implementation, which did not give good results. Fig. 18 shows the Viterbi results alongside Block type and Comb



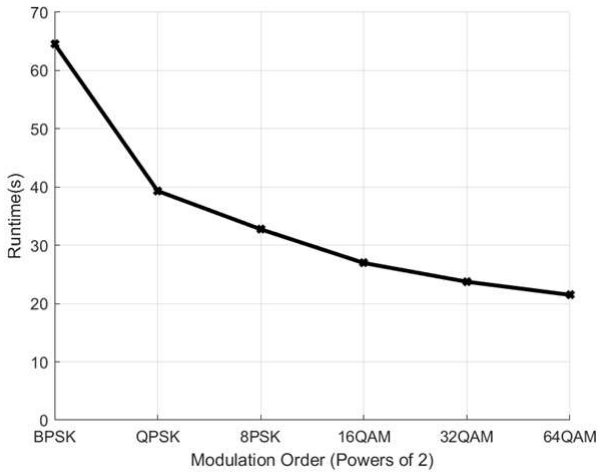


Figure 17: SNR vs. BER plot of each modulation scheme.

Figure 18: Run-time (in seconds) of the six modulation methods.

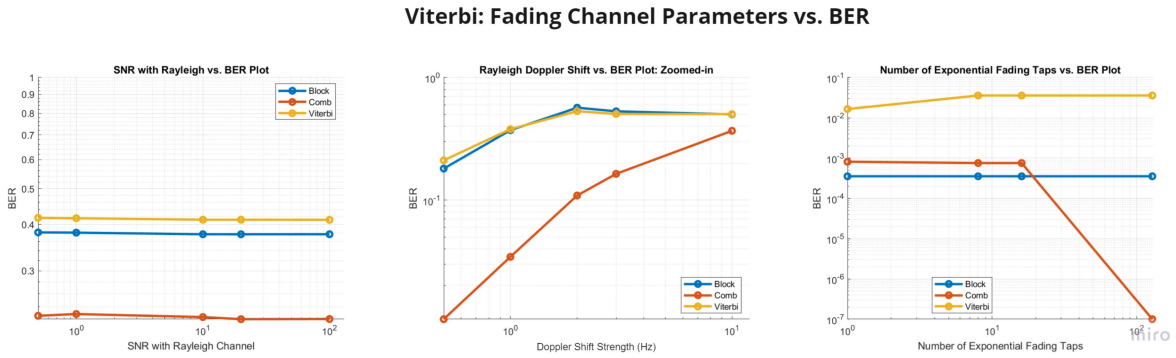


Figure 19: Viterbi Decoder: Channel Parameters vs. BER.

type. With the Rayleigh fading channel (center column of Fig. 18), Viterbi performs slightly better than Block type in dealing with the Doppler fading. This makes sense because Viterbi applies more frequent estimates than pure block type pilot.

10.2 Helper Functions

Here are the detailed descriptions of the helper functions used in our tx() and rx() functions.

10.2.1 Frame Synchronization

The main goal is to identify the beginning of the actual data stream in the received signal, as the transmission would induce time delays. To simulate the noisy environment, a Gaussian White Noise is added to the received signal (rx_signal). Then, the main loop goes through each frame

chunk of the noisy rx_signal and correlate the signal with the preamble data, as specified in the config structure. At the points that are within the over-sampled range AND exceeded the correlator detection threshold with the best (maximum) statistics, we identify the beginning of the signal bits.

10.2.2 Shaped Filter Detection

The code takes as input a received signal with additive white Gaussian noise, and filters it using a root-raised cosine (RRC) filter with a specified roll-off factor and filter length. The filtering is performed using convolution, and the resulting filtered signal is returned as output along with the pulse used for the filtering. Only the 'valid' part of the resulting signal (i.e., the part that does not have any zero padding at the edges) are returned as the output.

10.2.3 LFSR Preamble

This helper function uses a linear feedback shift register (LFSR) to generate and return a pseudo-noise sequence. The length of the PN sequence is specified by the output_length input argument, and the code returns the generated PN sequence as output.

The feedback polynomial of the LFSR is set to $x^0 + x^2 + x^3 + x^4 + x^8$ and with a period length of 255. After generating each new state with the polynomial and shifting the state values until the desired sequence length is reached, a PN sequence is returned.

10.2.4 OSFFT, OSIFFT

These functions take care of the oversampling factor when applying FFT to the parameter signal. The returned output would have the correct FFT/IFFT result and be in the default sample rate.

10.2.5 Low-Pass Filter

Uses the design() function to build a low-pass filter with specified filter coefficients and apply it to the input signal. This extracts the base-band QAM signal and improve the received SNR.