



**UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA
ELETRÔNICA E DE TELECOMUNICAÇÕES**

Gabriel Dos Santos Ferreira De Faria

**COMPARAÇÃO ENTRE OS ALGORITMOS DE CODIFICAÇÃO HUFFMAN E
LEMPER-ZIV PARA COMPRESSÃO DE SINAIS DE ÁUDIO APÓS A FILTRAGEM**

PATOS DE MINAS
2017

GABRIEL DOS SANTOS FERREIRA DE FARIA

**COMPARAÇÃO ENTRE OS ALGORITMOS DE CODIFICAÇÃO HUFFMAN E
LEMPER-ZIV PARA COMPRESSÃO DE SINAIS DE ÁUDIO APÓS A FILTRAGEM**

Trabalho de Conclusão de Curso apresentado à
Faculdade de Engenharia Elétrica da
Universidade Federal de Uberlândia, como pré-
requisito para obtenção de graduação em
Engenharia Eletrônica e de Telecomunicações

Orientador: Prof. Dr. Pedro Luiz Lima Bertarini

PATOS DE MINAS
2017

GABRIEL DOS SANTOS FERREIRA DE FARIA

**COMPARAÇÃO ENTRE OS ALGORITMOS DE CODIFICAÇÃO HUFFMAN E
LEMPERL-ZIV PARA COMPRESSÃO DE SINAIS DE ÁUDIO APÓS A FILTRAGEM**

Trabalho de Conclusão de Curso apresentado à
Faculdade de Engenharia Elétrica da
Universidade Federal de Uberlândia, como pré-
requisito para obtenção de graduação em
Engenharia Eletrônica e de Telecomunicações

BANCA EXAMINADORA

Professor Doutor Pedro Luiz Lima Bertarini (Orientador)
FEELT/UFU

Professora Doutora Karine Barbosa Carbonaro
FEELT/UFU

Professor Mestre Gustavo Nozella Rocha
FEELT/UFU

RESUMO

Os algoritmos de codificação Huffman e Lempel-Ziv são extensivamente utilizados em comunicações digitais para a compressão de dados. Por meio dessa compressão é possível reduzir significativamente o tamanho dos arquivos, possibilitando a redução de gastos com armazenamento de dados e tornando os sistemas mais rápidos e eficientes. Neste trabalho é estudado qual desses algoritmos tem o melhor desempenho de compressão para sinais de áudio filtrados e não filtrados. As métricas utilizadas para a análise de cada desempenho são: relação sinal ruído, comprimento médio de código e taxa de compressão. Em função disso, é utilizado o software Matlab para a simulação dos diferentes panoramas aqui apresentados.

Palavras-chave: codificação Huffman, codificação Lempel-Ziv, compressão, sinais de áudio, filtragem.

ABSTRACT

The Huffman and Lempel-Ziv coding algorithms are extensively used in digital communications for data compression. Through this data compression it is possible to significantly decrease the file size, reducing data storage costs and making the systems faster and more efficient. In this work, it is studied which of these algorithms has the best compression performance for filtered and unfiltered audio signals. The metrics used to analyze each performance are: signal-to-noise ratio, average code length and compression ratio. Moreover, Matlab software is used to simulate the distinct scenarios presented here.

Keywords: Huffman coding, Lempel-Ziv coding, data compression, audio signals, filtering.

LISTA DE FIGURAS

Figura 2.1 - Diagrama de blocos simples do sistema	10
Figura 2.2 - Resposta em frequência de FPB ideal	13
Figura 2.3 - Resposta em frequência de FPF ideal	13
Figura 2.4 - Exemplo de aplicação do algoritmo de codificação Huffman	17
Figura 2.5 - Etapa inicial do Algoritmo Lempel-Ziv 78.....	18
Figura 2.6 - Segunda etapa do Algoritmo Lempel-Ziv 78	20
Figura 2.7 - Codificação Lempel-Ziv 78 finalizada	20
Figura 4.1 - Diagrama de blocos completo do sistema.....	28
Figura 4.2 - Sinal original no tempo, testPadrao.wav	29
Figura 4.3 - Histograma do sinal, arquivo testPadrao.wav	29
Figura 4.4 - Processo de quantização do sinal, testPadrao.wav.....	30
Figura 4.5 - Espectro de frequências do sinal, testPadrao.wav	31
Figura 4.6 - Filtragem do sinal no domínio da frequência, testPadrao.wav	32
Figura 4.7 - Sinal no tempo, recuperado pela IFFT do sinal filtrado, testPadrao.wav	33
Figura 4.8 - Requantização do sinal recuperado pela IFFT após filtragem, testPadrao.wav	33
Figura 4.9 - SNR (à esquerda) e PSNR (à direita), testPadrao.wav	35
Figura 4.10 - Compressão após codificação Huffman e FPB, testPadrao.wav	36
Figura 4.11 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, testPadrao.wav	37
Figura 4.12 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, testPadrao.wav	37
Figura 4.13 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, testPadrao.wav	38
Figura 4.14 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, testPadrao.wav	38
Figura 4.15 - Sinal original no tempo (à esquerda) e espectro de frequências (à direita), testMulher.wav	40
Figura 4.16 - SNR (à esquerda) e PSNR (à direita), testMulher.wav.....	40
Figura 4.17 - Compressão após codificação Huffman e FPB, testMulher.wav	41
Figura 4.18 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, testMulher.wav	42
Figura 4.19 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, testMulher.wav	42
Figura 4.20 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, testMulher.wav	43
Figura 4.21 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, testMulher.wav	43

Figura 4.22 - Sinal original no tempo (à esquerda) e espectro de frequências (à direita), Classic.wav	45
Figura 4.23 - SNR (à esquerda) e PSNR (à direita) Classic.wav	45
Figura 4.24 - Compressão após codificação Huffman e FPB, Classic.wav	46
Figura 4.25 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, Classic.wav	47
Figura 4.26 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, Classic.wav	47
Figura 4.27 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, Classic.wav	48
Figura 4.28 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, Classic.wav	48
Figura 4.29 - Sinal original no tempo (à esquerda) e espectro de frequências (à direita) HarvardSentenceMan.wav.....	50
Figura 4.30 - SNR (à esquerda) e PSNR (à direita), HarvardSentenceMan.wav	50
Figura 4.31 - Compressão após codificação Huffman e FPB, HarvardSentenceMan.wav	51
Figura 4.32 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, HarvardSentenceMan.wav.....	52
Figura 4.33 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, HarvardSentenceMan.wav.....	52
Figura 4.34 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, HarvardSentenceMan.wav.....	53
Figura 4.35 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, HarvardSentenceMan.wav.....	53
Figura 4.36 - Sinal original no tempo (à esquerda) e espectro de frequências (à direita), HarvardSentenceWoman.wav	55
Figura 4.37 - SNR (à esquerda) e PSNR (à direita), HarvardSentenceWoman.wav.....	55
Figura 4.38 - Compressão após codificação Huffman e FPB, HarvardSentenceWoman.wav.....	56
Figura 4.39 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, HarvardSentenceWoman.wav	57
Figura 4.40 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, HarvardSentenceWoman.wav	57
Figura 4.41 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, HarvardSentenceWoman.wav	58

Figura 4.42 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, HarvardSentenceWoman.wav	58
Figura 4.43 - SNR (à esquerda) e PSNR (à direita), Classic.wav	60
Figura 4.44 - Compressão após codificação Huffman e FPF, Classic.wav	61
Figura 4.45 - Compressão após codificação Lempel-Ziv (5 bits) e FPF, Classic.wav	62
Figura 4.46 - Compressão após codificação Lempel-Ziv (6 bits) e FPF, Classic.wav	62
Figura 4.47 - Compressão após codificação Lempel-Ziv (7 bits) e FPF, Classic.wav	63
Figura 4.48 - Compressão após codificação Lempel-Ziv (8 bits) e FPF, Classic.wav	63
Figura 4.49 - FPB de 1750Hz (à esquerda) e FPF de 3250Hz (à direita), Classic.wav	64

LISTA DE TABELAS

Tabela 1 - Arquivos de áudios utilizados	24
Tabela 2 - Resultados do passo a passo, FPB de 2,2kHz e Huffman para testPadrao.wav	34
Tabela 3 - Resultados para testPadrao.wav, codificação Huffman e FPB	36
Tabela 4 - Resultados para testPadrao.wav, codificação Lempel-Ziv e FPB.....	39
Tabela 5 - Resultados para testMulher.wav, codificação Lempel-Ziv e FPB	41
Tabela 6 - Resultados para testMulher.wav, codificação Lempel-Ziv e FPB	44
Tabela 7 - Resultados para Classic.wav, codificação Huffman e FPB.....	46
Tabela 8 - Resultados para Classic.wav, codificação Lempel-Ziv e PFB.....	49
Tabela 9 - Resultados para HarvardSentenceMan.wav, codificação Huffman e FPB	51
Tabela 10 - Resultados para HarvardSentenceMan.wav, codificação Lempel-Ziv e FPB.....	54
Tabela 11 - Resultados para HarvardSentenceWoman.wav, codificação Lempel-Ziv e FPB	59
Tabela 12 - Resultados para Classic.wav, codificação Huffman e FPF	60
Tabela 13 - Resultados para Classic.wav, codificação Lempel-Ziv e FPF	64

LISTA DE SIGLAS

FFT	<i>Fast Fourier Transform</i>	Transformada Rápida de Fourier
LZ-77	<i>Lempel-Ziv 77</i>	
LZ-78	<i>Lempel-Ziv 78</i>	
LZMA	<i>Lempel-Ziv-Markov</i>	
LZW	<i>Lempel-Ziv-Welch</i>	
MPEG	<i>Moving Pictures Experts Group</i>	Grupo de Especialistas em Imagens em Movimento
JPEG	<i>Joint Photographic Experts Group</i>	Grupo de Especialistas em Fotografia
ZIP	<i>Move at high speed</i>	Mover em alta velocidade
GIF	<i>Graphics Interchange Format</i>	Formato para intercâmbio de gráficos
DCT	<i>Discrete Cosine Transform</i>	Transformada Discreta de Cossenos
XML	<i>eXtensible Markup Language</i>	Linguagem de Marcação Extensível
HTML	<i>HyperText Markup Language</i>	Linguagem de Marcação de Hipertexto
AAC	<i>Advanced Audio Coding</i>	Codificação Avançada de Áudio
WSN	<i>Wireless Sensor Network</i>	Rede de Sensores Sem Fio
IoT	<i>Internet of Things</i>	Internet das Coisas
MHC	<i>Modified Huffman Coding</i>	Codificação Huffman Modificada
XOR	<i>Exclusive OR</i>	Ou Exclusivo
PCM	<i>Pulse Code Modulation</i>	Modulação por Código de Pulso
SNR	<i>Signal to Noise Ratio</i>	Relação Sinal Ruído
PSNR	<i>Peak Signal to Noise Ratio</i>	Relação Sinal Ruído de Pico
CR	<i>Compression Ratio</i>	Taxa de Compressão
ACL	<i>Average Code Length</i>	Comprimento Médio de Código
MSE	<i>Mean Squarre Error</i>	Erro Quadrático Médio
FPB	Filtro Passa-Baixa	
FPF	Filtro Passa-Faixa	

SUMÁRIO

1. INTRODUÇÃO	8
2. REFERENCIAL TEÓRICO	10
2.1 REVISÃO DE LITERATURA.....	10
2.1.1 Sinais de Áudio	10
2.1.2 Amostragem e Quantização	11
2.1.3 Transformada Rápida de Fourier.....	12
2.1.4 Filtros.....	12
2.1.5 Teoria da Informação	14
2.1.5.1 Informação.....	14
2.1.5.2 Entropia	15
2.1.5.3 Codificação de Fonte	15
2.1.5.4 Comprimento Médio de Palavra Código	15
2.1.5.5 Limite Fundamental.....	16
2.1.6 Algoritmo de Codificação Huffman	16
2.1.7 Algoritmo de Codificação Lempel-Ziv	18
2.2 TRABALHOS E APLICAÇÕES NA ÁREA.....	20
3. DESENVOLVIMENTO.....	24
3.1 MATERIAIS.....	24
3.2 MÉTODOS	24
3.2.1 Métricas	25
3.2.2 Metodologia.....	26
4. RESULTADOS	28
4.1 PASSO A PASSO.....	28
4.1.1 Sinal no tempo e Histograma.....	28
4.1.2 Amostragem e Quantização	29
4.1.3 FFT e FPB	30
4.1.4 IFFT e Requantização	32
4.1.5 Resultados.....	34
4.2 FILTRO PASSA-BAIXA (FPB)	34
4.2.1 testPadrao.wav	35

4.2.1.1 Huffman.....	35
4.2.1.2 Lempel-Ziv	37
4.2.2 testMulher.wav	39
4.2.2.1 Huffman.....	40
4.2.2.2 Lempel-Ziv	42
4.2.3 Classic.wav	44
4.2.3.1 Huffman.....	45
4.2.3.2 Lempel-Ziv	47
4.2.4 HarvardSentenceMan.wav.....	49
4.2.4.1 Huffman.....	50
4.2.4.2 Lempel-Ziv	52
4.2.5 HarvardSentenceWoman.wav	55
4.2.5.1 Huffman.....	55
4.2.5.2 Lempel-Ziv	56
4.3 FILTRO PASSA-FAIXA (FPF)	59
4.3.1 Huffman.....	60
4.3.2 Lempel-Ziv	62
5. CONCLUSÃO.....	66
REFERÊNCIAS	68

1. INTRODUÇÃO

As comunicações digitais trabalham com a transmissão e a recepção de sinais digitais. Esses sinais são mensagens representadas por sequências de bits de tal forma que ao serem recebidos possam ser interpretados e as mensagens sejam entendidas. Como a necessidade por comunicações vem crescendo significativamente, juntamente com esse crescimento surgem também necessidades por sistemas mais rápidos, mais eficientes e mais confiáveis. Sendo assim, a compressão de dados surge como uma alternativa de resposta à essas necessidades. Por meio da compressão, é possível reduzir consideravelmente o tamanho dos arquivos, resultando em gastos menores com armazenamento e sistemas que conseguem trabalhar com taxas maiores. Para que a compressão possa ocorrer, é necessário que se utilize um algoritmo de codificação que consequentemente poderá resultar em uma redução no número de bits usados para representar determinado arquivo [1].

Dois dos algoritmos mais conhecidos e utilizados para codificação em comunicações digitais são o algoritmo de Huffman [2] e o algoritmo de Lempel-Ziv [3]. Tais algoritmos são a base para a realização deste projeto que faz a comparação de desempenho entre os dois em relação à compressão de sinais de áudio filtrados, ou seja, diferentes filtros são aplicados aos sinais de áudio aqui estudados e são abordadas as consequências desse processo para a compressão desses sinais. A vantagem em relação à filtragem é a capacidade que os filtros têm em selecionar somente a faixa de frequências de interesse dos sinais de áudio, eliminando assim possíveis gastos de armazenamento para frequências que não são utilizadas ou que não fazem com que a mensagem de áudio se torne ininteligível.

Os arquivos de áudio, de maneira geral, requerem largura de banda considerável para serem transmitidos e isso impacta em grande capacidade de armazenamento. Porém, esses mesmos sinais possuem muita redundância, o que proporciona alto grau de compressão com a utilização de algoritmos para esse fim [4]. É esperado que os dois algoritmos tenham desempenhos diferentes, tanto para sinais de áudio filtrados quanto para sinais não filtrados, uma vez que a técnica de codificação utilizada por esses algoritmos é diferente. O algoritmo de Huffman é capaz de alcançar o menor comprimento de palavra código possível para um dado símbolo, dependendo somente do conhecimento prévio da estatística da fonte geradora daquele símbolo. Já o algoritmo de Lempel-Ziv é feito símbolo por símbolo, dependendo da sequência em que aparecem para criar o seu dicionário de códigos [5].

A utilização de filtros associados a algoritmos de codificação para compressão, surge como uma alternativa para que taxas de compressão ainda mais elevadas sejam alcançadas. Outro fator importante é a facilidade de se implementar um filtro digital para áudio em contrapartida ao elevado gasto computacional necessário para suprir o armazenamento de um arquivo não comprimido ou que esteja comprimido a uma baixa taxa de compressão [6]

Portanto, os seguintes aspectos são estudados neste trabalho: Qual o melhor algoritmo de codificação, Huffman ou Lempel-Ziv, para a compressão dos sinais de áudio aqui estudados com e sem a filtragem? Qual é a influência da filtragem dos sinais de áudio nas taxas de compressão desses sinais? Qual algoritmo tem melhor desempenho de compressão para cada configuração de filtro? Qual a melhor relação entre largura de banda, comprimento médio de código e relação sinal ruído para os filtros e algoritmos aqui propostos?

Para isso, este trabalho é dividido da seguinte maneira: na primeira seção é apresentada a introdução, que contém o contexto no qual este trabalho está inserido bem como a sua proposta. Na segunda seção é apresentado o referencial teórico no qual este trabalho é fundamentado. O desenvolvimento contendo os recursos necessários e a metodologia empregada neste trabalho se encontra na terceira seção. Na quarta seção os resultados deste trabalho são apresentados de acordo com a metodologia escolhida e explicada na seção anterior, enquanto nas duas últimas seções, são apresentadas, respectivamente, a conclusão a respeito deste trabalho e as referências utilizadas para a sua realização.

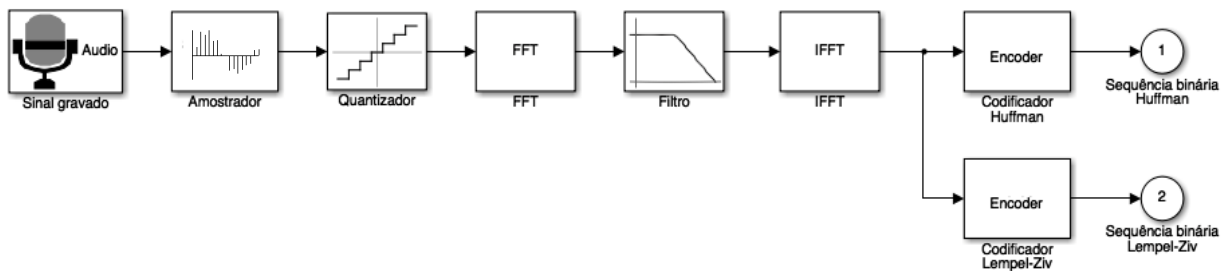
2. REFERENCIAL TEÓRICO

Nesta seção, em um primeiro momento é mostrado todo o embasamento teórico no qual este trabalho de conclusão de curso é fundamentado, e posteriormente é falado sobre os trabalhos e pesquisas desenvolvidos com fundamentação semelhante e aplicações diversas.

2.1 REVISÃO DE LITERATURA

A ordem na qual essa revisão é mostrada segue o mesmo raciocínio mostrado no diagrama de blocos simples da Figura 2.1.

Figura 2.1 - Diagrama de blocos simples do sistema



Fonte: Autor deste trabalho

2.1.1 Sinais de Áudio

O som é caracterizado por um fenômeno ondulatório que exerce pressão sobre o ar. Esse fenômeno é um sinal que assume valores contínuos no tempo. A faixa de frequências audíveis desses sinais para os seres humanos se encontra entre 20Hz e 20kHz, sendo que os limites inferior e superior dessa faixa espectral variam de pessoa para pessoa, bem como de acordo com a faixa etária e as condições do ambiente no qual o som está se propagando. Mesmo sendo aptos a ouvir sons dentro dessa faixa de frequências, os seres humanos são mais sensíveis às frequências entre 2kHz e 5kHz. Essa sensibilidade consiste em poder distinguir os sons mesmo sendo escutados em intensidades muito baixas [7].

Em relação aos sinais de áudio correspondentes à voz humana, o cenário se torna ainda um pouco diferente. Talvez uma das propriedades mais básicas a respeito da forma de onda desses sinais (voz humana) é que elas são limitadas em banda. Essa limitação começa desde o próprio processo de produção da voz nas cordas vocais até contribuições adicionais dos próprios sistemas de comunicação, como no caso dos sistemas de telefonia, cuja largura de banda utilizada na transmissão desses sinais está compreendida entre 200Hz e 3,2kHz [8].

A grande vantagem em se ter uma largura de banda limitada é que esse sinal pode ser amostrado, passando de contínuo para discreto, a uma taxa também limitada. Na prática, como no caso dos sistemas de telefonia mencionados anteriormente, a frequência de amostragem utilizada é 8kHz, o que significa uma frequência ainda um pouco maior que a frequência de Nyquist. Existem algumas vantagens na superamostragem de um sinal, ou seja, utilizar uma frequência de amostragem maior que a frequência de Nyquist, porém, mais importante do que isso são as desvantagens na subamostragem, onde a frequência de amostragem utilizada é menor que a frequência proposta pelo teorema. A principal desvantagem na subamostragem é a degradação do sinal original que é distorcido pelo fenômeno conhecido como *aliasing*. Neste fenômeno, a distorção acontece porque a frequência de amostragem não é suficientemente grande para que amostras consecutivas do sinal original possam ser distinguidas durante a amostragem [8].

2.1.2 Amostragem e Quantização

Em comunicações digitais, a informação deve ser representada por sequências binárias, possibilitando o processamento e a transmissão dessa informação pelo canal de comunicação. Caso essa informação não esteja representada dessa forma, como é o caso de sinais analógicos, ela deve então ser convertida em uma sequência binária tornando-se um sinal digital. Esse processo de conversão normalmente se inicia pela amostragem, onde o sinal contínuo é discretizado, ou seja, é representado por um conjunto de amostras discretas do sinal original a uma certa taxa de amostragem [9].

O limite dessa taxa de amostragem, que permite uma sequência discreta de amostras capturar toda a informação contida em um sinal contínuo no tempo com largura de banda limitada, é conhecido como o Teorema da Amostragem. No teorema, é mostrado que essa sequência discreta de amostras deve ser amostrada a uma taxa no mínimo duas vezes maior que a frequência do sinal original, frequência esta conhecida como a frequência de Nyquist [10;11].

Feita a amostragem do sinal, esse sinal deve ser então quantizado. A quantização é o processo pelo qual esse grande conjunto de amostras do sinal discretizado serão mapeadas em um conjunto menor, facilitando assim a codificação desse sinal. Esse processo gera um erro, chamado de erro de quantização, que basicamente é a diferença entre o valor da amostra antes e depois da quantização. Esse tipo de erro pode ser amenizado aumentando-se a quantidade de níveis de

quantização, ou seja, a quantidade de valores que podem ser atribuídos ao grande conjunto de amostras do sinal discretizado [13].

2.1.3 Transformada Rápida de Fourier

Neste estudo os sinais são trabalhados no domínio da frequência, para que posteriormente os filtros possam ser aplicados a esses sinais e a influência dessa filtragem na capacidade de compressão dos algoritmos Huffman e Lempel-Ziv possa ser investigada. Desta forma, é necessário que o sinal seja convertido de seu domínio original, domínio do tempo, para o domínio da frequência, utilizando a Transformada Rápida de Fourier (FFT – *Fast Fourier Transform*). A FFT é um método largamente utilizado em comunicações digitais por sua capacidade em encontrar as componentes de frequência de sinais com qualquer forma de onda [9].

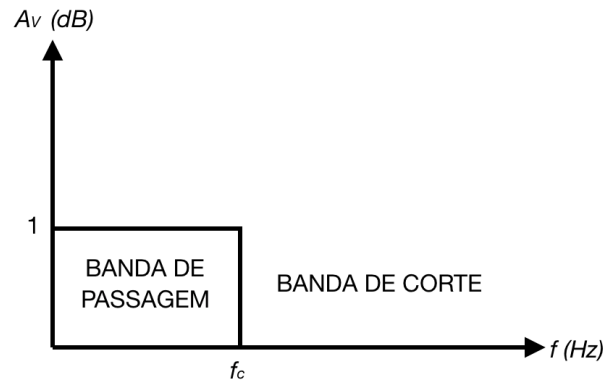
2.1.4 Filtros

Os filtros são dispositivos capazes de permitir a passagem de sinais com certos valores de frequências enquanto bloqueia outros sinais com frequências não desejáveis para o sistema. Desta forma, de acordo com as diferentes respostas em frequência, os filtros podem ser classificados em passa-baixa, passa-alta, passa-faixa, rejeita-faixa e passa-todas [14].

Neste trabalho, é estudada a influência da filtragem dos sinais de áudio, antes da codificação Huffman e Lempel-Ziv, na compressão desses sinais. Para essa filtragem, são considerados o Filtro Passa-Baixa (FPB) e o Filtro Passa-Faixa (FPF), com respostas em frequência muito próximas do ideal.

O filtro passa-baixa é um circuito cuja resposta em frequência se comporta da seguinte maneira: os sinais com frequências abaixo da frequência de corte do filtro (f_c) não sofrem atenuação (para filtros ideais) e, portanto, passam pelo filtro. Já os sinais com frequências acima da frequência de corte do filtro são atenuados e rejeitados pelo circuito. A Figura 2.2 ilustra a resposta em frequência de um filtro passa-baixa ideal.

Figura 2.2 - Resposta em frequência de FPB ideal

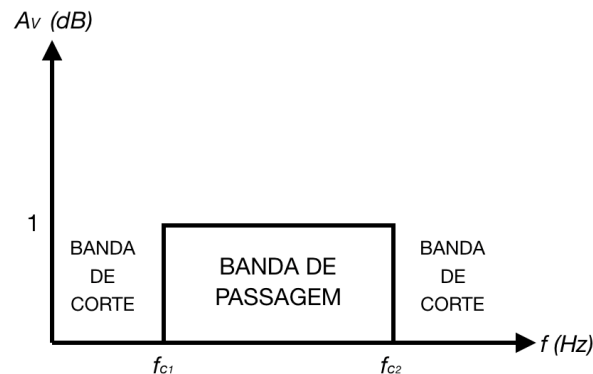


Fonte: Abert Malvino, 2007 [5]

Nota-se que para os sinais com frequências menores que f_c , o ganho é unitário, ou seja, o módulo do sinal de entrada é igual ao de saída. Já para frequências com valores acima de f_c , o ganho é zero, ou seja, o módulo do sinal de saída é atenuado até zero.

O filtro passa-faixa, por sua vez, é um circuito cuja resposta em frequência se comporta de forma diferente à do filtro passa-baixa. Os sinais com frequências dentro do intervalo compreendido entre f_{c1} e f_{c2} não sofrem atenuação (para filtros ideais) e, portanto, passam pelo filtro, já os sinais com frequências abaixo de f_{c1} e frequências acima de f_{c2} são atenuados e rejeitados pelo circuito. A Figura 2.3 ilustra a resposta em frequência de um filtro passa-faixa ideal.

Figura 2.3 - Resposta em frequência de FPF ideal



Fonte: Abert Malvino, 2007 [5]

Nota-se que para os sinais com frequências entre f_{c1} e f_{c2} , o ganho é unitário, ou seja, o módulo do sinal de entrada é igual ao de saída. Já para frequências com valores abaixo de f_{c1} e acima de f_{c2} , o ganho é zero, ou seja, o módulo do sinal de saída é atenuado até zero.

2.1.5 Teoria da Informação

A Teoria da Informação se baseia no importante estudo realizado por Shannon em 1948, o Teorema da Codificação [12]. No teorema é proposto que se uma fonte tem uma taxa de informação menor que a capacidade do canal, existe então um método de codificação tal que a mensagem proveniente dessa fonte pode ser transmitida através deste canal produzindo um erro insignificante. Esse teorema se mostrou extremamente importante para toda a academia na época, pois Shannon explicou que a transmissão e a recepção poderiam ser alcançadas com um erro desprezível mesmo na presença de um canal ruidoso [13]. Para que o método de codificação correto seja escolhido para cada aplicação, o próprio processo de codificação deve ser compreendido bem como alguns conceitos básicos em teoria da informação.

2.1.5.1 Informação

Em teoria da informação, quanto menor a probabilidade de ocorrência de um símbolo, maior será a quantidade de informação que esse símbolo representa. Esse conceito é fundamental para o entendimento de codificação e, posteriormente, compactação de dados. Considerando x_j um evento ou símbolo que ocorre com uma probabilidade $p(x_j)$, se este evento aconteceu, isso quer dizer que a quantidade de informação recebida foi:

$$I(x_j) = \log_2 \left(\frac{1}{p(x_j)} \right) \quad (1)$$

A Equação 1 justifica o conceito de informação mostrado anteriormente, uma vez que a quantidade de informação $I(x_j)$ aumenta a medida em que a probabilidade $p(x_j)$ diminui. Vale observar que, caso o evento ocorra sempre ou nunca ocorra, $p(x_j) = 1$ ou $p(x_j) = 0$, respectivamente, esse evento não possui nenhuma informação útil, ou seja, ou ele é totalmente certo ou ele é inexistente, portanto não existe incerteza em relação a esse evento. O logaritmo na base dois resulta em uma quantidade de informação medida em bits, o que também faz sentido uma vez que a comunicação digital utiliza o sistema binário (0's e 1's) para representar as mensagens.

2.1.5.2 Entropia

A entropia pode ser entendida como a quantidade média de informação por mensagem. Sendo assim, a entropia de uma fonte é o somatório do produto entre a probabilidade de cada símbolo e a quantidade de informação contida em cada símbolo:

$$H = \sum_{j=1}^n p(x_j) \log_2 \left(\frac{1}{p(x_j)} \right) \quad (2)$$

na qual n é o número de símbolos da fonte na Equação 2.

2.1.5.3 Codificação de Fonte

Para que a codificação de uma fonte discreta seja feita de forma eficiente, o codificador de fonte precisa de um conhecimento prévio da estatística da fonte. Desta maneira, esse codificador poderá atribuir palavras código mais simples (curtas) para os símbolos de maior ocorrência e palavras código mais complexas (longas) para os símbolos de menor ocorrência. Esse tipo de código é chamado de código de tamanho variável [5]. Para que se desenvolva um codificador de fonte eficiente nesse sentido, é necessário que duas condições sejam respeitadas:

1. As palavras código produzidas pelo codificador de fonte estejam na forma binária.
2. As palavras código produzidas pelo codificador de fonte sejam inequivocamente decodificáveis.

Sendo assim, é necessário encontrar o comprimento médio das palavras código para o codificador de fonte.

2.1.5.4 Comprimento Médio da Palavra Código

O comprimento médio da palavra código pode ser entendido como o número médio de bits por símbolo da fonte, conforme Equação 3.

$$\bar{L} = \sum_{j=1}^n p(x_j) l_j \quad (3)$$

Na qual l_j corresponde ao comprimento em bits da palavra código atribuída ao símbolo x_j .

O comprimento médio da palavra código, Equação 3, é um importante parâmetro de comparação entre métodos de codificação diferentes [6].

2.1.5.5 Limite Fundamental

O limite inferior para o comprimento médio da palavra código, para uma dada fonte discreta sem memória, é a própria entropia da fonte. Sendo assim, nenhuma palavra código produzida pelo codificador de fonte pode ser representada por um número de bits menor do que o valor da entropia dessa fonte.

Esse limite, Equação 4, é conhecido como o limite fundamental do número médio de bits necessários para representar uma fonte discreta sem memória.

$$\bar{L} \geq H \quad (4)$$

2.1.6 Algoritmo de Codificação Huffman

O algoritmo de codificação Huffman é um código prefixo, ou seja, nenhuma palavra código é prefixa para qualquer outra palavra código, não resultando assim em ambiguidade na decodificação. Uma vez que a entropia de uma determinada fonte é previamente conhecida, a codificação Huffman gera o código com o menor comprimento médio de palavra código, produzindo então, a maior eficiência para tal fonte. A codificação Huffman visa produzir um código cujo comprimento médio de palavra código se aproxime do limite fundamental definido pela entropia da fonte [2].

O algoritmo da codificação Huffman possui os seguintes passos:

1. Todos os símbolos da fonte são organizados em uma coluna em ordem decrescente de probabilidade;
2. As probabilidades dos dois últimos símbolos são combinadas (somadas) em uma única probabilidade;
3. Os símbolos são novamente organizados em uma nova coluna à direita da primeira também em ordem decrescente de probabilidade. Caso o resultado da combinação de duas probabilidades gere um valor de probabilidade já existente na coluna, esse novo valor poderá ser colocado acima ou abaixo do valor já existente, porém o mesmo critério deverá ser adotado até o fim do algoritmo;

4. Os passos 2 e 3 se repetem até que se tenha somente dois símbolos restantes;
5. Esses dois símbolos então serão os primeiros dígitos das palavras código produzidas pelo algoritmo. É assinalado o bit 1 para um dos símbolos e o bit 0 para o outro;
6. Retorna-se então a coluna anterior e os símbolos que foram combinados nessa coluna anteriormente, serão os próximos dígitos das palavras código produzidas pelo algoritmo. É assinalado o bit 1 para um dos símbolos e o bit 0 para o outro seguindo o mesmo critério do passo 5 (se o símbolo superior no passo 5 recebeu o bit 1 e o inferior recebeu o bit 0, assim também será feito no passo 6);
7. O passo 6 é repetido até retornar na primeira coluna que foi organizada no passo 1;
8. As palavras código binárias são obtidas fazendo-se o caminho inverso, ou seja, a partir da última coluna (a direita), em direção a primeira coluna (a esquerda), anotando-se os bits encontrados em cada percurso.

A Figura 2.4 ilustra os caminhos percorridos pelo algoritmo de codificação Huffman, as combinações das probabilidades dos símbolos e os percursos para se encontrar cada palavra código.

Figura 2.4 - Exemplo de aplicação do algoritmo de codificação Huffman

Símbolos	Etapa I	Etapa II	Etapa III	Etapa IV	Palavra Código
S0	0,6	0,6	0,6	0,6 0	S0 = 0
S1	0,2	0,2	0,2	0,4 1	S1 = 10
S2	0,1	0,1	0,2		S2 = 110
S3	0,05	0,1			S3 = 1110
S4	0,05				S4 = 1111

Fonte: Autor deste trabalho

As palavras código geradas pelo algoritmo não são exclusivas e únicas para essa fonte em questão. Na Etapa I da Figura 2.4 por exemplo, o bit 0 e o bit 1 poderiam ser em posições inversas, o que mudaria a posição dos bits nas etapas posteriores, uma vez que o mesmo critério deve ser adotado durante todo o algoritmo. Desta maneira, o algoritmo gera palavras de código diferentes das geradas no exemplo, mantendo o mesmo comprimento médio de palavra código. Outra variação possível, seria o posicionamento da combinação das duas menores probabilidades da

Etapa I (s_3 e s_4) em cima da probabilidade de s_2 , uma vez que representam o mesmo valor (0,1). Mais uma vez o mesmo critério deveria ser adotado durante todo o algoritmo, o que resultaria no posicionamento da combinação das duas menores probabilidades da Etapa II (s_3+s_4 e s_2), também em cima da probabilidade de s_1 . Novamente tais alterações resultariam em palavras código diferentes, mas com um mesmo comprimento médio de palavra código.

2.1.7 Algoritmo de Codificação Lempel-Ziv

O algoritmo de codificação Lempel-Ziv é feito por meio de um algoritmo adaptativo que, diferentemente do algoritmo de Huffman, não necessita do conhecimento prévio da estatística da fonte. Essa não dependência desse conhecimento prévio gera algumas vantagens em determinadas aplicações, uma vez que, na prática, nem sempre é possível prever essa estatística [5].

Esse algoritmo em questão tem algumas versões diferentes, como Lempel-Ziv 77 (LZ-77), Lempel-Ziv 78 (LZ-78), Lempel-Ziv-Markov (LZMA) e Lempel-Ziv-Welch. A versão utilizada neste trabalho é o LZ-78. Este tipo de algoritmo é realizado analisando-se o fluxo de dados da fonte em segmentos que sejam as subsequências mais breves não encontradas anteriormente [3]. Para o melhor entendimento em relação ao funcionamento do algoritmo, será considerada a sequência binária ilustrada na Figura 2.5 e um codificador de 3 bits será utilizado como exemplo.

Figura 2.5 - Etapa inicial do Algoritmo Lempel-Ziv 78

00010111001010010									
Posição Numérica	1	2	3	4	5	6	7	8	9
Subsequência	0	1	00						
Representação Numérica			11						
Blocos Codificados			0010						

Fonte: Haykin, S. Sistemas de comunicação analógicos e digitais [5]

Inicialmente, presume-se que os símbolos binários 0 e 1 já estejam armazenados nessa ordem no *codebook*, que nada mais é do que o dicionário que correlaciona a sequência binária proveniente da fonte com suas respectivas palavras código. Sendo assim, a sequência binária em questão começa a ser analisada e codificada.

Esse processo de codificação se inicia da esquerda para a direita e procura-se a subsequência mais breve e inédita contida na sequência binária que está sendo analisada. Neste exemplo, pode-se observar que a próxima subsequência mais breve e inédita será 00, uma vez que os bits 0 e 1 já estão gravados no *codebook*. Todas as subsequências têm uma posição numérica (linha 2 da Figura 2.5), que representa a ordem das subsequências no *codebook*. Sendo assim, a subsequência 00 recebe a posição numérica 3, uma vez que ela se encontra depois dos bits 0 (posição numérica 1) e 1 (posição numérica 2) no *codebook*.

Outra linha importante da Figura 2.5 é a linha 3, que consiste na representação numérica de cada subsequência. Essa representação numérica funciona da seguinte forma: da esquerda para a direita, é observado quais são as subsequências anteriores que compõem essa subsequência atual. No caso da subsequência que está sendo analisada, subsequência 00 na posição numérica 3, ela nada mais é do que a concatenação de duas subsequências 0 (posição numérica 1), o que resulta na representação numérica 11. As setas da Figura 2.5 auxiliam na compreensão desse processo.

Por fim, a palavra código será gerada na última linha da Figura 2.5, a linha dos blocos binários codificados. Essa palavra código é composta pela representação binária (3 bits – codificador utilizado neste exemplo) do primeiro número da representação numérica, neste caso o número 1, mais o bit da subsequência que aparece como novidade até então no *codebook*, neste caso o segundo bit 0 da subsequência 00 é a novidade, este bit é chamado de símbolo de inovação e é o último bit que aparece na subsequência.

A próxima etapa do algoritmo Lempel-Ziv 78 pode ser observada na Figura 2.6. Nessa etapa, a próxima subsequência mais breve e inédita será 01, pois os bits 0 e 1 sozinhos já estão no *codebook*. Desta maneira, é atribuída a posição numérica 4 para essa subsequência e a representação numérica correspondente é 12, que representa a concatenação dos bits 0 e 1, posições numéricas 1 e 2 respectivamente. Finalmente a palavra código que representa essa subsequência codificada é 0011, que é a representação binária do número 1 mais o símbolo de inovação (último bit da subsequência).

Figura 2.6 - Segunda etapa do Algoritmo Lempel-Ziv 78

00010111001010010									
Posição Numérica	1	2	3	4	5	6	7	8	9
Subsequência	0	1	00	01					
Representação Numérica			11	12					
Blocos Codificados			0010	0011					

Fonte: Haykin, S. Sistemas de comunicação analógicos e digitais [5]

O algoritmo prossegue respeitando as mesmas regras descritas até então, e consequentemente todas as subsequências são identificadas e codificadas. A Figura 2.7 ilustra a codificação finalizada para a sequência de bits utilizada como exemplo.

Figura 2.7 - Codificação Lempel-Ziv 78 finalizada

000101110010100101									
Posição Numérica	1	2	3	4	5	6	7	8	9
Subsequência	0	1	00	01	011	10	010	100	101
Representação Numérica			11	12	42	21	41	61	62
Blocos Codificados			0010	0011	1001	0100	1000	1100	1101

Fonte: Haykin, S. Sistemas de comunicação analógicos e digitais [5]

O processo de decodificação também é simples, uma vez que o algoritmo de Lempel-Ziv usa códigos de tamanho fixo para representar um numero variável de símbolos da fonte. Dessa forma, sabendo o tamanho dos blocos codificados, é possível saber que o último bit corresponde ao símbolo de inovação, ou seja, ele também é o último bit da subsequência em questão. Os bits restantes, nesse exemplo os outros 3 bits, correspondem à posição numérica na qual está o restante da subsequência, consequentemente tem-se a palavra código decodificada.

2.2 TRABALHOS E APLICAÇÕES NA ÁREA

Existem inúmeros estudos voltados para a área de codificação utilizando os algoritmos Huffman e Lempel-Ziv. Em [1], é realizado um estudo comparativo entre os desempenhos de

ambos os algoritmos para compressão de arquivos de texto, imagem e áudio. No estudo, o algoritmo de Huffman demonstrou melhor desempenho em taxa de compressão para texto, enquanto o algoritmo de Lempel-Ziv teve desempenho superior para arquivos de imagens. Já para arquivos de áudio, não foram verificadas diferenças significativas entre os desempenhos dos dois algoritmos e essa parte do estudo foi inconclusiva, o que chama ainda mais a atenção para novos estudos comparativos entre os algoritmos para arquivos de áudio.

Separadamente, são propostas aplicações tanto utilizando algoritmo de Lempel-Ziv quanto utilizando Huffman para compressão de arquivos de áudio. Em [15], Lempel-Ziv é implementado em um estudo com o objetivo de atingir altas taxas de compressão com perdas (compactação), enquanto em [16] Huffman é utilizado como uma alternativa eficiente de compressão para arquivos de áudio MPEG-1, para transferências a taxas de bits reduzidas. Em [17] é analisada a eficiência da codificação Huffman por meio da filtragem de sinais de áudio, mostrando que sinais de distribuições diferentes apresentam comprimento médio diferentes após a compactação e que, além de ocorrer uma alteração no histograma do sinal após a filtragem, a relação sinal ruído também é afetada.

Em relação aos tipos de arquivos que suportam compressão, os mais conhecidos como MPEG, JPEG, ZIP e GIF também utilizam Huffman ou Lempel-Ziv como algoritmos de codificação. O algoritmo de Huffman está presente nos arquivos JPEG (compactação de imagens) e MPEG (compressão de áudio e vídeo) com o objetivo de eliminar parte da redundância gerada nos coeficientes da Transformada Discreta de Cosseno (DCT – *Discrete Cosine Transform*), ajudando a melhorar a taxa de compressão. Já o algoritmo Lempel-Ziv se faz presente nos arquivos ZIP (compressão de arquivos em geral) e GIF (compressão de imagens *bitmap*).

Mesmo a literatura mostrando a superioridade do algoritmo de Huffman sobre o Lempel-Ziv para arquivos de texto, estudos continuam sendo feitos nessa área de aplicação. Em [18], é implementado o algoritmo de Lempel-Ziv para arquivos de texto estruturados, como o XML e o HTML. O estudo mostrou que além de taxa de compressão considerável para arquivos de texto estruturados, facilitando assim o armazenamento, a utilização do algoritmo facilita a manipulação e o acesso randômico desses arquivos.

Aplicações desses algoritmos tem objetivos que vão além de altas taxas de compressão, como as técnicas de esteganografia e marca d'água (*watermarking*), nas quais é escondido algum tipo de informação secreta junto ao arquivo que deve ser enviado, de forma que ninguém além do

transmissor e do receptor saiba da existência de tal informação [19]. Essa informação pode ser uma imagem, um áudio ou até mesmo um vídeo. Em [20], uma imagem é escondida em arquivos de áudio AAC (*Advanced Audio Coding*) através da esteganografia utilizando o código de Huffman. Já em [21], o algoritmo de Huffman é utilizado para criar marca-d'água em arquivos de áudio agregados com o intuito de verificar a autenticidade e a integridade do arquivo. A marca-d'água também é utilizada para documentos de texto como em [22], onde o algoritmo de Lempel-Ziv é implementado com o objetivo de identificar falsificações em documentos, atingindo índices de até 99% de acerto nas identificações, e também com o objetivo de restaurar os documentos modificados. Finalmente, a marca-d'água também é utilizada na área da saúde para evitar fraudes e falsificações em imagens de exames médicos como ultrassonografias e ressonâncias, utilizando o algoritmo de compressão Huffman [23].

A utilização de algoritmos de codificação em aplicações médicas não se restringe somente à verificação de autenticidade das imagens médicas. Imagens de fluoroscopia por exemplo, que consiste em um tipo especial de raio X, no qual várias imagens de raio X são exibidas em tempo real como uma única radiografia, estão sendo comprimidas por meio de codificação Huffman utilizando a correlação entre as imagens sequenciais da fluoroscopia, em um processo sem perdas, visando uma economia em armazenamento de arquivos médicos [24].

Otimização do consumo energético também é um tema relacionado aos algoritmos de compressão e é de extrema importância nas Redes de Sensores sem Fio (WSN – *Wireless Sensor Networks*) que são as redes utilizadas em Internet das Coisas (IoT – *Internet of Things*). Essa otimização pode aumentar o tempo de vida útil tanto das baterias quanto dos sensores utilizados nessa tecnologia. Em [25] é proposto uma versão modificada da Codificação Huffman (MHC – *Modified Huffman Coding*) como uma ferramenta efetiva na compressão de dados para reduzir o consumo de energia dos sensores nas *WSN's*.

Em relação a algoritmos de codificação associados à filtragem, um estudo analisou o algoritmo de codificação Huffman em imagens filtradas por filtros Ou-Exclusivos (XOR – *Exclusive – OR*) [26]. No estudo, o filtro foi utilizado como uma ferramenta de correlação para que altas taxas de compressão sem perdas pudessem ser alcançadas. Ambas imagens, a imagem original e a imagem já filtrada, foram então codificadas pelo algoritmo de codificação Huffman e o aumento da taxa de compressão foi acompanhado pela redução da entropia nas imagens previamente filtradas.

Já em relação a compressão de sinais de áudio associada à filtragem, é mostrado em [27] a influência de filtros passa-baixa em sistemas modulados por código de pulso (PCM – *Pulse Code Modulation*) para sinais de áudio comprimidos.

3. DESENVOLVIMENTO

Nesta seção são discutidos quais os materiais necessários para o desenvolvimento deste trabalho, bem como quais os métodos, por meio dos quais, esses materiais são utilizados para gerar as métricas analisadas nos resultados. A ordem com a qual essa metodologia é abordada, e a sua divisão em etapas, também é discutida nesta seção, para uma melhor compreensão do leitor ao analisar a seção posterior dos resultados.

3.1 MATERIAIS

Este trabalho é todo desenvolvido através do *software Matlab*, que conta com todos os recursos de simulação necessários para a implementação do projeto.

3.2 MÉTODOS

Este trabalho faz a comparação de desempenho entre os algoritmos de codificação Huffman e Lempel-Ziv, para a compressão de sinais de áudio após a filtragem, sendo que a influência dos filtros utilizados na taxa de compressão também é investigada.

Os arquivos escolhidos para serem analisados são todos com extensão WAVE, que é um formato-padrão de áudio da Microsoft e IBM, e é um dos tipos de arquivos aceitos pelas funções do Matlab utilizadas durante o trabalho. Além de todos terem a mesma extensão, os arquivos escolhidos também são todos com 16 bits por amostra e frequência de amostragem de 8kHz. Informações adicionais sobre esses arquivos podem ser visualizadas na Tabela 1 a seguir:

Tabela 1 - Arquivos de áudio utilizados

Nome arquivo	Duração (s)	Descrição
testPadrao.wav	5	Voz do autor deste trabalho, gravada por meio do Matlab
testMulher.wav	5	Voz feminina, gravada por meio do Matlab
Classic.wav	8	Trecho de música clássica
HarvardSentenceMan.wav	8	Voz masculina do repositório de Harvard [28]
HarvardSentenceWoman.wav	8	Voz feminina do repositório de Harvard [28]

A escolha de tais sinais é justificada pela infinidade de aplicações e sistemas que utilizam o sinal de fala humana em seus funcionamentos, como o sistema de telefonia, músicas, filmes, etc. Em relação aos dois arquivos pertencentes ao repositório de Harvard [28], além de ser um

repositório de livre acesso e utilização dos áudios para fins acadêmicos, alguns outros estudos já utilizaram tal repositório para processamento e análise de sinais de áudio, o que pode trazer um benefício maior para futuros estudos que queiram comparar taxas de compressão utilizando outros métodos de codificação ou mesmo Huffman e Lempel-Ziv.

O *Matlab* possui inúmeras funcionalidades que podem ser aplicadas ao processamento de sinais, inclusive a sinais de áudio. As funções *audiorecorder* e *audioreader* por exemplo, que permitem, respectivamente, que as falas sejam gravadas e executadas através do próprio *software*, sem a necessidade de importação de arquivo.

Outra ferramenta importante do *Matlab* que facilita o processamento é o *Simulink*, um ambiente de simulação baseado em diagrama de blocos, que além da própria simulação, suporta geração automática de código, editor de gráficos, testes contínuos e exportação dos resultados de simulação diretamente para o *Matlab* [29].

As métricas utilizadas para a comparação de desempenho dos algoritmos de codificação são abordadas a seguir.

3.2.1 Métricas

As principais variáveis quantitativas escolhidas para a comparação entre os algoritmos de codificação Huffman e Lempel-Ziv para os sinais de áudio serão: Relação Sinal-Ruído (SNR – *Signal to Noise Ratio*), Relação Sinal-Ruído de Pico (PSNR - *Peak Signal to Noise Ratio*), Comprimento Médio de Código (ACL - *Average Code Length*) e Taxa de Compressão (CR - *Compression Ratio*).

A relação sinal ruído é um parâmetro extremamente utilizado em comunicações digitais. Esse parâmetro mede a diferença entre o nível de potência do sinal recebido e o nível de potência do ruído que foi atribuído a esse sinal. Essa atribuição pode acontecer em detrimento de uma série de fatores, desde o ruído do próprio canal de comunicação até o ruído adquirido no processo da quantização, que também é conhecido como erro de quantização e representa o nível mínimo de ruído para um sinal digitalizado e que pode, como visto anteriormente nesse trabalho, ser minimizado com o aumento dos níveis de quantização [6]. A *SNR* utilizada como parâmetro de comparação neste trabalho, Equação 5, é dada em decibel (dB) e o ruído considerado é a diferença entre o sinal de áudio original e o sinal de áudio quantizado.

$$SNR_{(dB)} = 10 \log_{10} \left(\frac{P_{\text{sinale}}}{P_{\text{ruído}}} \right) \quad (5)$$

Uma segunda relação sinal-ruído também serve de parâmetro para este trabalho, que é a relação sinal-ruído de pico. A Equação 6 mostra a PSNR em dB, que utiliza o erro quadrático médio (MSE - *Mean Squarre Error*) em seu cálculo e o número de pontos total (N) do arquivo considerado.

$$PSNR_{(dB)} = 10 \log_{10} \left(\frac{N}{MSE^2} \right) \quad (6)$$

Outro parâmetro importante para a comparação dos algoritmos na codificação dos sinais de áudio é a taxa de compressão. Este parâmetro é a razão entre o tamanho do arquivo comprimido e o tamanho do arquivo antes da compressão (ambos os tamanhos em bits), resultando em um número que significa o quão menor é o arquivo comprimido em relação ao arquivo original. Essa razão pode ser observada na Equação 7 em sua forma de porcentagem, ou seja, seu resultado mostra quantos por cento o arquivo comprimido é menor que o arquivo original.

$$CR_{(\%)} = \left[1 - \left(\frac{\text{tamanho do arquivo comprimido}_{bits}}{\text{tamanho do arquivo antes da compressao}_{bits}} \right) \right] \times 100 \quad (7)$$

E por fim temos o parâmetro do comprimento médio de código que, como mostrado anteriormente neste trabalho na Equação 3, indica o número médio de bits por símbolo, e neste caso é utilizado por ambos os algoritmos como parâmetro essencial à codificação e cálculo de *SNR*. A metodologia utilizada neste trabalho é abordada a seguir.

3.2.2 Metodologia

A metodologia empregada neste trabalho segue as métricas da seção anterior como parâmetros de comparação entre os desempenhos dos algoritmos. As etapas dessa metodologia são divididas da seguinte forma:

1. Demonstração do passo a passo utilizando arquivo testPadrao.wav:
 - a. Huffman (quantização de 7 bits).

2. Filtro Passa-Baixa (FPB): é aplicado o FPB a todos os arquivos da Tabela 1, utilizando os dois algoritmos:
 - a. Huffman (quantização de 5 a 8 bits);
 - b. Lempel-Ziv (quantização de 5 a 8 bits, e comprimento do dicionário de 5 a 8 bits).
3. Filtro Passa-Faixa (FPF): é aplicado o FPF ao arquivo da Tabela 1 que tenha os melhores resultados durante a etapa 2:
 - a. Huffman (quantização de 5 a 8 bits);
 - b. Lempel-Ziv (quantização de 5 a 8 bits, e comprimento do dicionário de 5 a 8 bits).

Em todas as etapas, são utilizados a forma gráfica e tabelas com o objetivo de analisar os resultados.

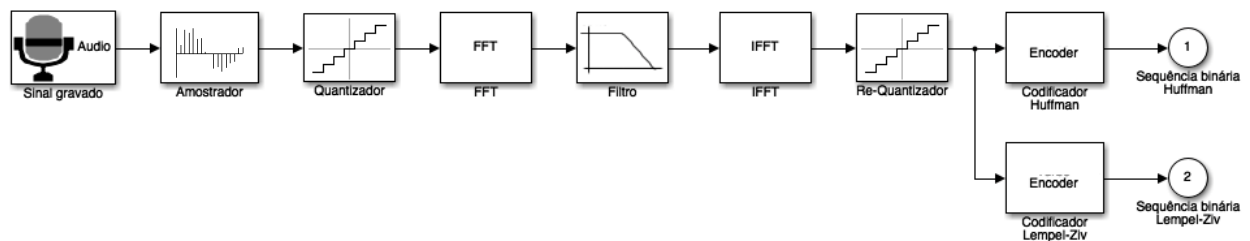
4. RESULTADOS

Nesta seção é mostrado quais os resultados provenientes da metodologia escolhida para este trabalho e explicada anteriormente. Todas as etapas desse processo têm seus resultados analisados com o auxílio de figuras e tabelas. As figuras e os dados contidos nas tabelas são frutos de implementação de código por meio do *software Matlab*.

4.1 PASSO A PASSO

Nesta etapa, será demonstrado através de gráficos, como todo o procedimento base deste estudo é realizado. Esse procedimento segue o mesmo raciocínio do diagrama de blocos da Figura 2.1, mas pode ser melhor compreendido através do diagrama de blocos completo da Figura 4.1.

Figura 4.1 - Diagrama de blocos completo do sistema



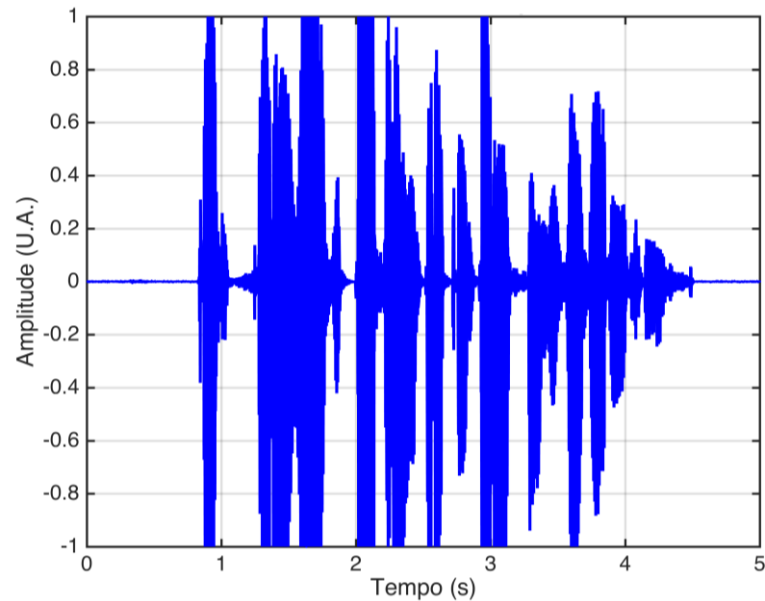
Fonte: Autor deste trabalho

O áudio escolhido para a demonstração do passo a passo foi o *testPadrao.wav* e o algoritmo de codificação foi o Huffman. A seguir, pode ser observado os diversos procedimentos realizados no *Matlab* para o processamento dos dados.

4.1.1 Sinal no tempo e Histograma

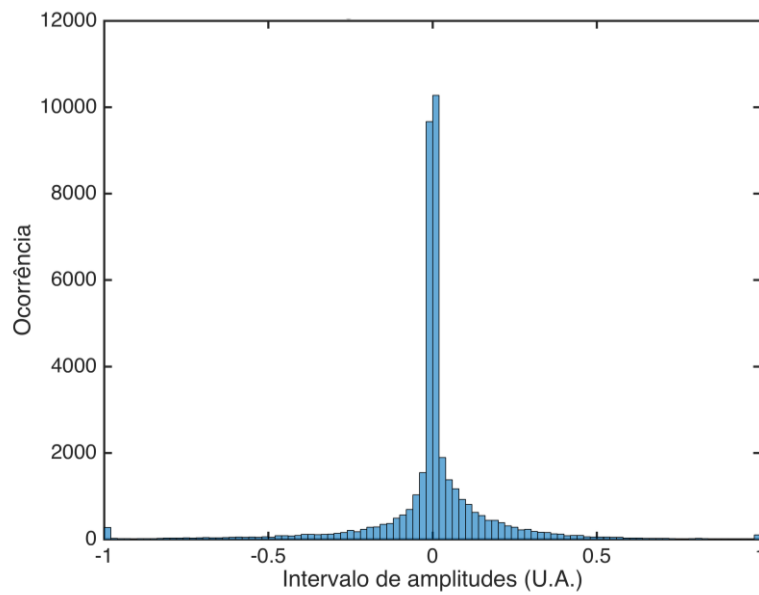
Neste procedimento, o áudio *testPadrao.wav* é lido pelo Matlab por meio da função *audioread* e também é normalizado para que sua amplitude esteja contida no intervalo entre -1 e +1 U.A. (Unidade Arbitrária). A Figura 4.2 ilustra a distribuição desse sinal no tempo, na qual pode-se comprovar o tempo de duração do áudio como sendo de 5 segundos. E a Figura 4.3 mostra o histograma do sinal, ou seja, a frequência com a qual ocorre cada intervalo de amplitude desse sinal.

Figura 4.2 - Sinal original no tempo, testPadrao.wav



Fonte: Autor deste trabalho

Figura 4.3 - Histograma do sinal, arquivo testPadrao.wav



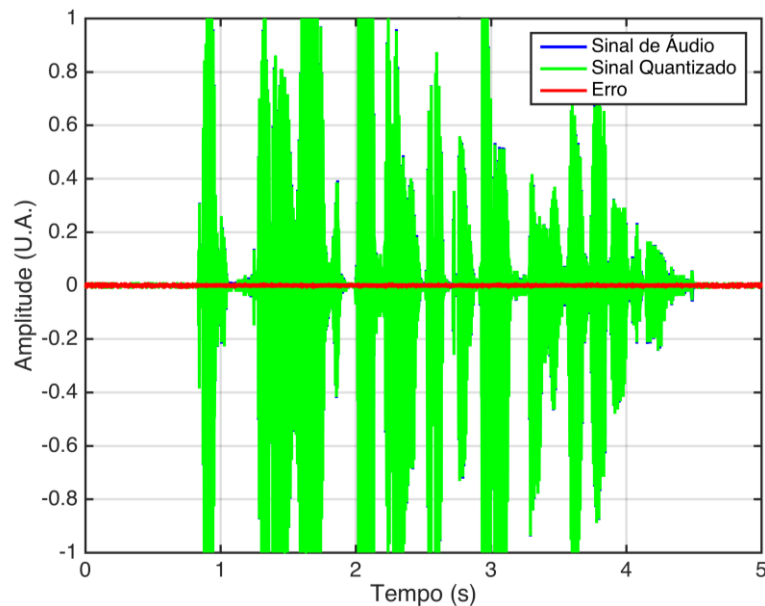
Fonte: Autor deste trabalho

4.1.2 Amostragem e Quantização

Como já discutido anteriormente na seção 3.2, a frequência de amostragem é 8kHz. Sendo assim as maiores componentes de frequência do sinal não devem ultrapassar os 4kHz (Frequência de Nyquist), respeitando o Teorema da Amostragem.

Para a quantização desse sinal, foi escolhido um quantizador de 7 bits, ou seja, 128 níveis de quantização possíveis. A Figura 4.4 ilustra o sinal de áudio original em azul, o sinal quantizado em verde e o erro de quantização em vermelho, que nada mais é do que a diferença entre o sinal original e o quantizado. Pode ser visto que o erro de quantização é muito pequeno, bem próximo de zero, comprovando que para esse sinal, utilizar um quantizador de 7 bits gera um resultado bastante satisfatório em relação ao erro de quantização. Isso também explica a dificuldade em se enxergar o sinal original em azul, uma vez que o sinal quantizado em verde acaba sobrepondo o sinal original em azul, em detrimento dessa grande similaridade.

Figura 4.4 - Processo de quantização do sinal, testPadrao.wav

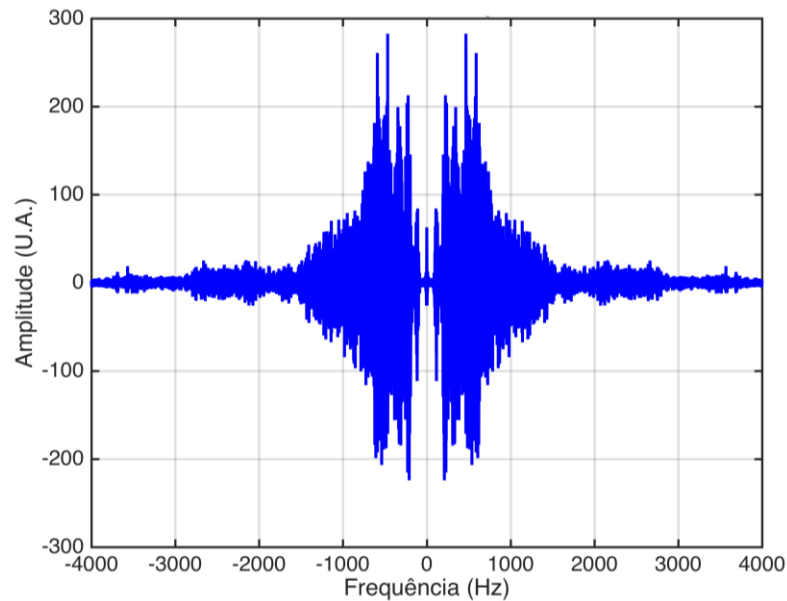


Fonte: Autor deste trabalho

4.1.3 FFT e FPB

Após o procedimento de quantização, o sinal deve sair do domínio do tempo para o domínio da frequência, para que possa posteriormente ser filtrado de acordo com a faixa de frequências desejada e/ou frequência de corte do filtro utilizado. A Figura 4.5 ilustra o espectro de frequências do sinal em questão, obtido através da FFT do mesmo.

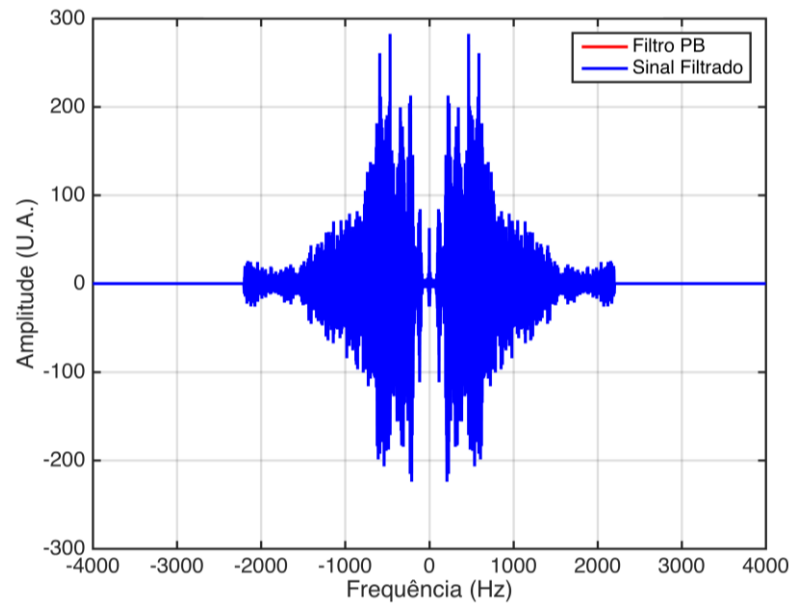
Figura 4.5 - Espectro de frequências do sinal, testPadrao.wav



Fonte: Autor deste trabalho

Pode ser observado, que existe uma maior concentração de informação nas baixas frequências desse sinal. Com o sinal já no domínio do tempo, a filtragem de frequências se faz possível. Neste caso, é utilizado um filtro passa-baixa (FPB) de 2,2kHz, ou seja, somente as frequências abaixo dessa frequência de corte que passam pelo filtro. A Figura 4.6 ilustra o espectro de frequências desse sinal após passar pelo FPB. O sinal em vermelho, que demonstra o comportamento do FPP, não é observado devido a grande diferença de amplitude dele em relação ao sinal filtrado, sendo +1 entre -2,2kHz e +2,2kHz, e 0 para os demais valores.

Figura 4.6 - Filtragem do sinal no domínio da frequência, testPadrao.wav

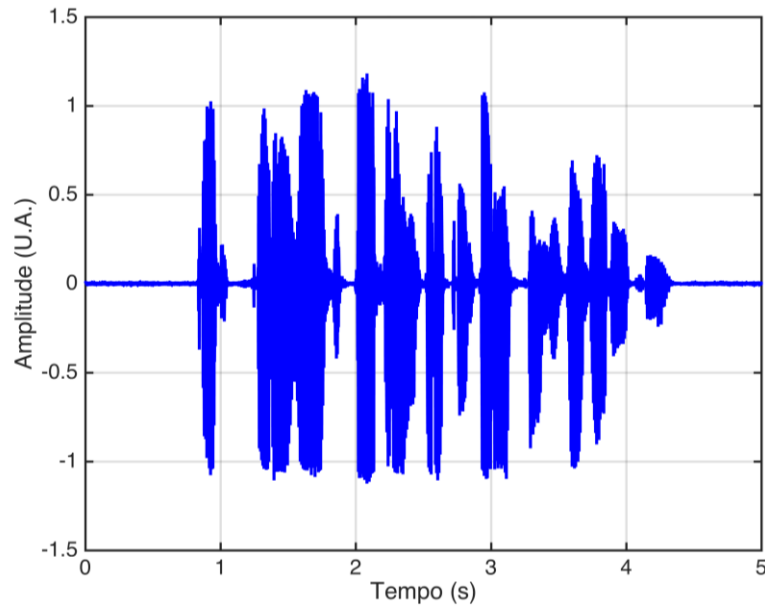


Fonte: Autor deste trabalho

4.1.4 IFFT e Requantização

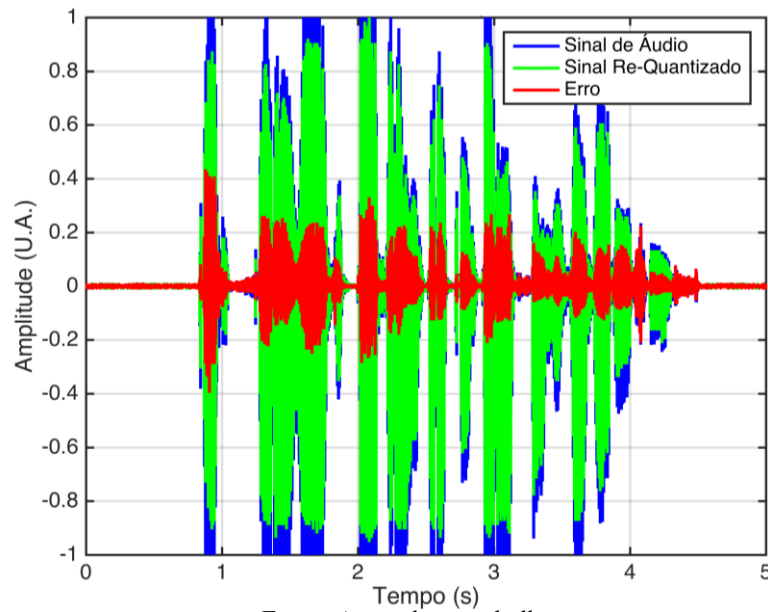
Feita a filtragem do sinal no domínio da frequência, é necessário retornar para o domínio do tempo e então condicionar o sinal para o processo de codificação por Huffman e/ou Lempel-Ziv. A Figura 4.7 ilustra a IFFT do sinal filtrado resultante do procedimento anterior. Pode ser observado que o sinal no tempo recuperado pela IFFT se assemelha ao sinal original da Figura 4.2, porém com alguns valores acima de +1 U.A. e -1U.A. Sendo assim, para que esse sinal seja codificado pelo algoritmo de Huffman, ele deve ser normalizado, e no caso de Lempel-Ziv, além da mesma normalização, o sinal também deve ser somente positivo para que certas funções do Matlab funcionem corretamente. Feito isso, o sinal passa pelo processo de requantização, para que o dicionário criado por Huffman seja suficiente para codificar todas as entradas possíveis do sinal, e que o erro resultante do processo de mudança de domínio e filtragem, não atrapalhe a codificação. A Figura 4.8 ilustra o resultado da requantização, na qual mais uma vez o sinal de áudio original se encontra em azul, o sinal requantizado em verde e o erro de requantização em vermelho.

Figura 4.7 - Sinal no tempo, recuperado pela IFFT do sinal filtrado, testPadrao.wav



Fonte: Autor deste trabalho

Figura 4.8 - Requantização do sinal recuperado pela IFFT após filtragem, testPadrao.wav



Fonte: Autor deste trabalho

Logicamente, o erro de requantização é maior do que o erro de quantização, uma vez que parte do sinal original foi retirada através da filtragem. O sinal ilustrado na Figura 4.8 está condicionado para ser codificado pelo algoritmo de codificação Huffman.

4.1.5 Resultados

Os resultados de toda a simulação demonstrada no passo a passo da seção 4.1, podem ser visualizados na Tabela 2. As nomenclaturas utilizadas para os resultados deste trabalho têm os seguintes significados:

- SNR_1 : Relação Sinal-Ruído em dB, antes da filtragem;
- SNR_2 : Relação Sinal-Ruído em dB, depois da filtragem;
- $PSNR_1$: Relação Sinal-Ruído de Pico em dB, antes da filtragem;
- $PSNR_2$: Relação Sinal-Ruído de Pico em dB, depois da filtragem;
- $AVGLEN_1$: Comprimento médio em bits, antes da filtragem;
- $AVGLEN_2$: Comprimento médio em bits, depois da filtragem;
- $COMP_1$: Compressão em porcentagem, antes da filtragem,
- $COMP_2$: Compressão em porcentagem, depois da filtragem.

Tabela 2 - Resultados do passo a passo, FPB de 2,2kHz e Huffman para testPadrao.wav

SNR_1 (dB)	SNR_2 (dB)	$PSNR_1$ (dB)	$PSNR_2$ (dB)
15,73	7,25	136,81	102,89
$AVGLEN_1$ (bits)	$AVGLEN_2$ (bits)	$COMP_1$ (%)	$COMP_2$ (%)
4,48	4,25	35,92	39,20

Para o cenário demonstrado no passo a passo, é observado por meio da Tabela 1, que a filtragem influencia de forma considerável na compressão do sinal em questão, alcançando uma compressão de 39,20% com filtragem e 35,92% sem a filtragem, ou seja, tem-se um ganho de 9,13% em compressão ao se utilizar o filtro de 2,2kHz para o arquivo testPadrao.wav.

4.2 FILTRO PASSA-BAIXA (FPB)

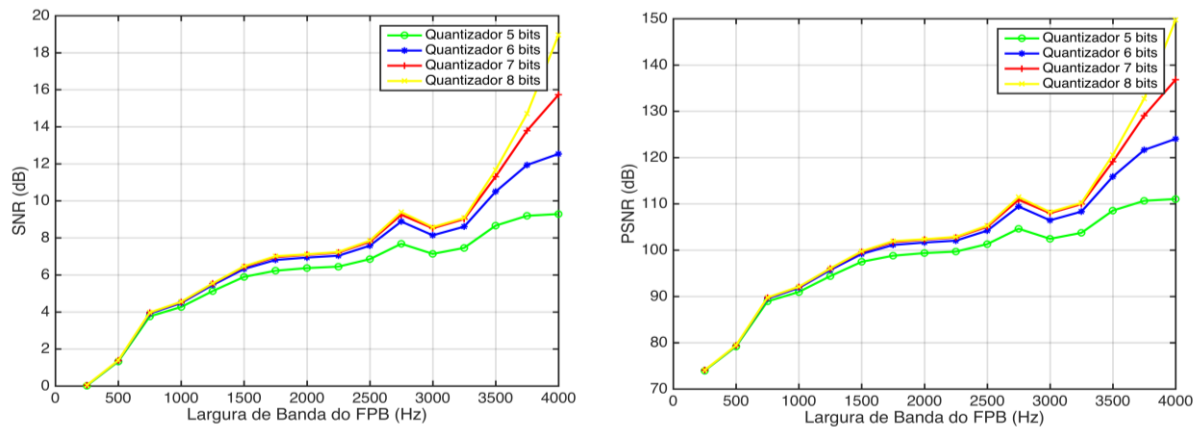
Nesta etapa, todos os arquivos da Tabela 1 são submetidos a um filtro passa-baixa. É utilizado um passo de 250Hz para cada ponto apresentado nos gráficos, tanto para Huffman quanto para Lempel-Ziv, sendo que o primeiro ponto corresponde à primeira frequência de corte, 250Hz, e o último ponto à última frequência de corte do filtro, 4kHz. Vale ressaltar que, para uma frequência de corte de 4kHz, 100% do sinal passa pelo filtro, sendo um ponto de muita importância

na análise dos resultados, pois mostra a compressão do arquivo por Huffman ou Lempel-Ziv sem a ação da filtragem.

4.2.1 testPadrao.wav

Será mostrado, para todos os arquivos submetidos ao filtro passa-baixa, figuras que contêm o gráfico do sinal no tempo, seu espectro de frequências e relação sinal-ruído normal e de pico em função da largura de banda do filtro para melhor análise durante o decorrer da apresentação dos resultados. Como o arquivo test.Padrao.wav foi utilizado na seção 4.1 para demonstrar o passo a passo, a sua representação no tempo e seu espectro de frequências podem ser visualizados nas Figuras 4.2 e 4.5 respectivamente. A Figura 4.9 por sua vez, ilustra a SNR e PSNR deste sinal quando submetido ao filtro.

Figura 4.9 - SNR (à esquerda) e PSNR (à direita), testPadrao.wav

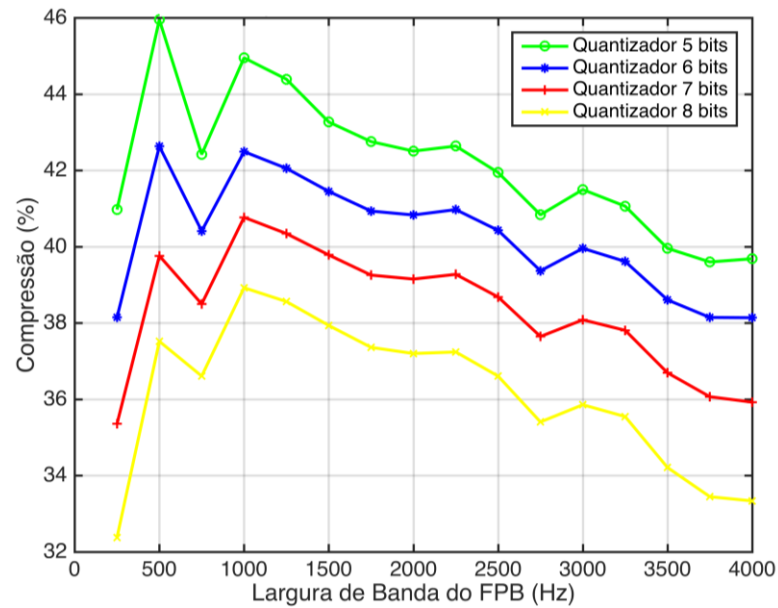


Fonte: Autor deste trabalho

4.2.1.1 Huffman

A Figura 4.10 ilustra as taxas de compressão alcançadas, por meio do algoritmo de codificação Huffman, para cada valor de largura de banda do FPB.

Figura 4.10 - Compressão após codificação Huffman e FPB, testPadrao.wav



Fonte: Autor deste trabalho

A Tabela 3 mostra os pontos que geram os resultados mais interessantes em relação à codificação Huffman aliada ao FPB para o arquivo testPadrao.wav.

Tabela 3 - Resultados para testPadrao.wav, codificação Huffman e FPB

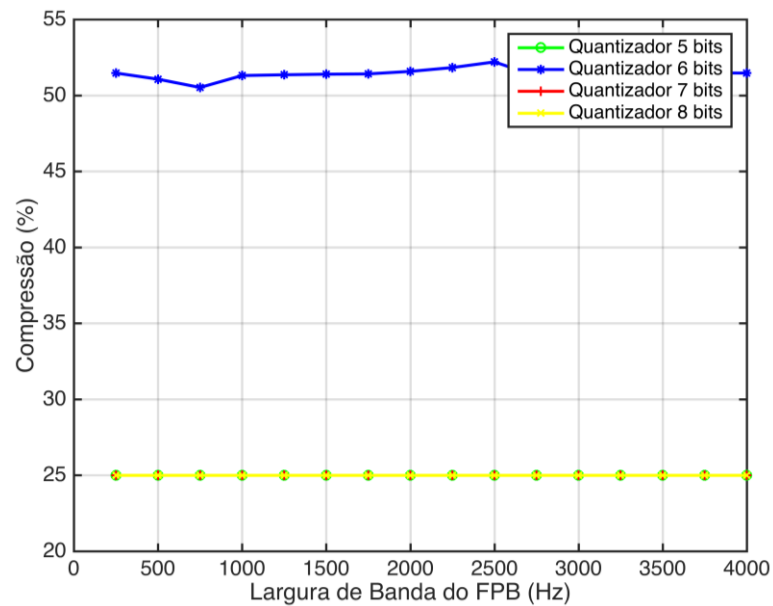
Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
5	500	9,25	1,32	111,10	79,10	39,66	46,00
8	1000	19,10	4,62	149,80	92,42	33,41	38,92
8	2750	19,10	9,51	149,80	112,20	33,41	35,45

Por meio das figuras apresentadas e dos resultados da Tabela 3, é verificado que a filtragem utilizando um FPB de 1000Hz aliada a um quantizador de 8 bits gera um ganho de compressão de 16,49%, sendo que o custo desse ganho é refletido na diminuição da SNR do sinal, de 19,10dB para 4,62dB, porém ainda se tem mais sinal do que ruído e esse processo gera um sinal completamente inteligível para a audição humana.

4.2.1.2 Lempel-Ziv

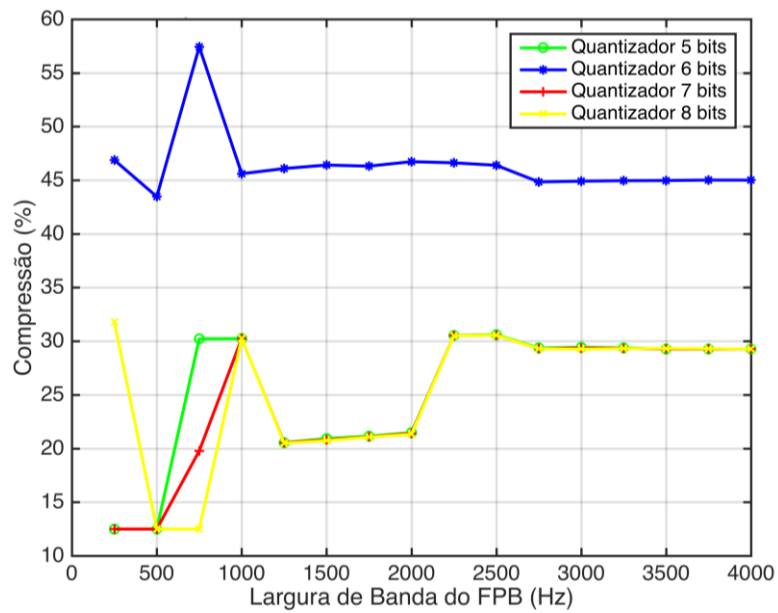
As Figuras 4.11, 4.12, 4.13 e 4.14 ilustram, respectivamente, as taxas de compressão alcançadas, por meio do algoritmo de codificação Lempel-Ziv, para cada valor de largura de banda do FPB, utilizando dicionários de 5, 6, 7 e 8 bits.

Figura 4.11 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, testPadrao.wav



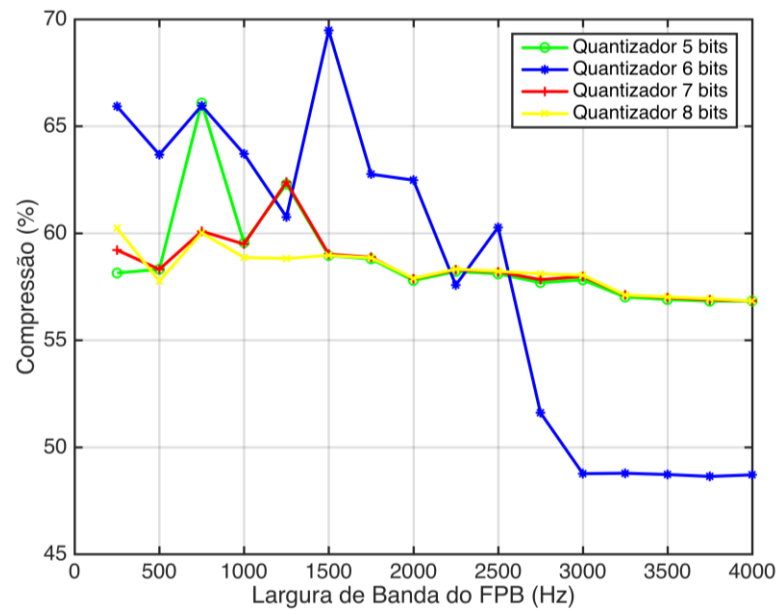
Fonte: Autor deste trabalho

Figura 4.12 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, testPadrao.wav



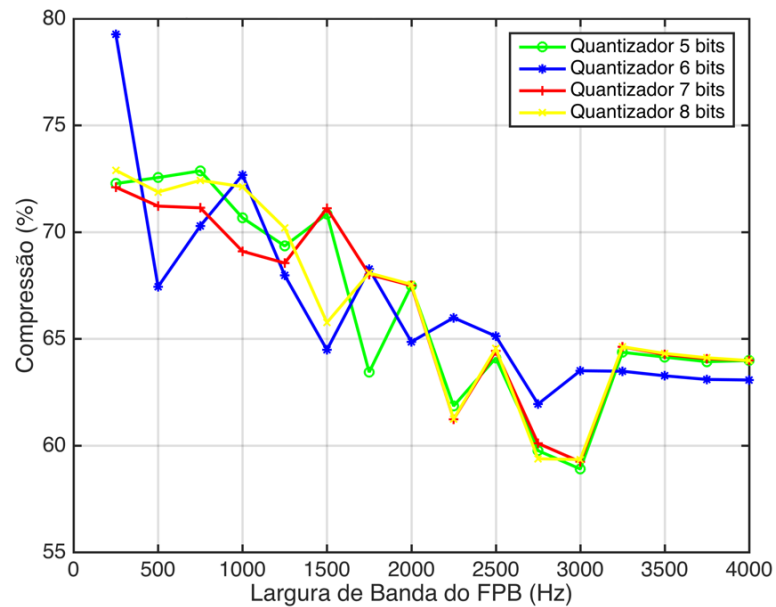
Fonte: Autor deste trabalho

Figura 4.13 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, testPadrao.wav



Fonte: Autor deste trabalho

Figura 4.14 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, testPadrao.wav



Fonte: Autor deste trabalho

A Tabela 4 mostra os pontos que geram os resultados mais interessantes em relação à codificação Lempel-Ziv aliada ao FPB para o arquivo testPadrao.wav.

Tabela 4 - Resultados para testPadrao.wav, codificação Lempel-Ziv e FPB

Dicionário (bits)	Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
6	6	750	12,66	4,01	124,20	90,01	45,00	57,50
7	6	1500	12,66	6,43	124,20	99,10	48,75	69,50
7	5	750	9,25	3,82	111,10	88,81	56,87	66,30
8	6	1000	12,66	9,51	124,20	93,20	63,20	72,27
8	8	1250	19,10	5,51	149,80	96,30	64,10	70,32

Por meio das figuras apresentadas e dos resultados da Tabela 4, é verificado que a filtragem utilizando um FPB de 1500Hz, com codificação Lempel-Ziv de dicionário de 7 bits, aliados a um quantizador de 6 bits gera um ganho de compressão expressível da ordem de 42,56%. Um ponto que gera uma menor diminuição da SNR do sinal em detrimento do ganho de compressão, é a combinação de 1000Hz, dicionário de 8 bits e quantizador de 6 bits, na qual a SNR do sinal diminui de 12,66dB para 9,51dB, embora o ganho em compressão seja menor que o anterior, atingindo 14,35% de ganho com a filtragem.

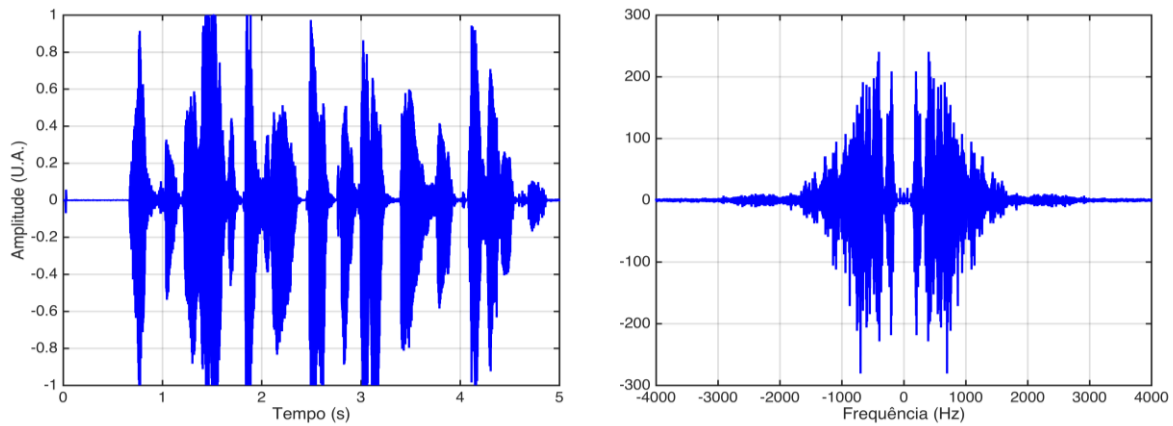
Vale ressaltar que para o arquivo testPadrao.wav, sem o processo da filtragem, a codificação Lempel-Ziv de 7 e 8 bits gera compressões superiores à Huffman para todos os quantizadores. Já para Lempel-Ziv utilizando dicionários de 5 e 6 bits, as taxas de compressão geradas por meio da filtragem são todas inferiores às geradas por Huffman para todos os pontos, menos para Lempel-Ziv com dicionário e quantizador de 6 bits e FPB de 750Hz.

Outro aspecto importante a ser considerado é que os comportamentos distintos das curvas do quantizador de 6 bits ilustrados nas Figuras 4.11 e 4.12 são inconclusivos. Não foi possível descobrir o motivo desses comportamentos, tendo em vista que as curvas para os demais quantizadores seguem um padrão.

4.2.2 testMulher.wav

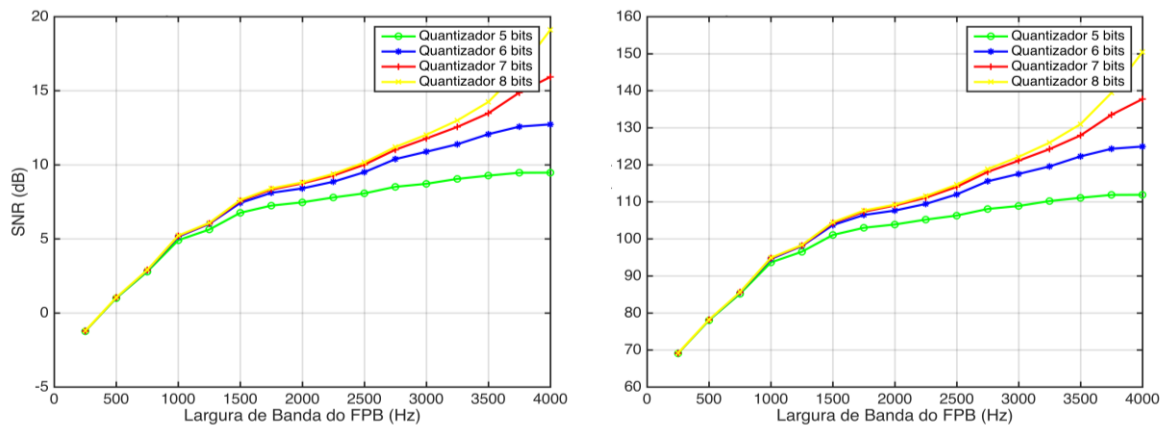
A Figura 4.15 ilustra o sinal representado no tempo e seu espectro de frequências. Já a Figura 4.16 mostra a SNR e PSNR deste sinal quando submetido ao filtro.

Figura 4.15 - Sinal original no tempo (à esquerda) e espectro de frequências (à direita), testMulher.wav



Fonte: Autor deste trabalho

Figura 4.16 - SNR (à esquerda) e PSNR (à direita), testMulher.wav

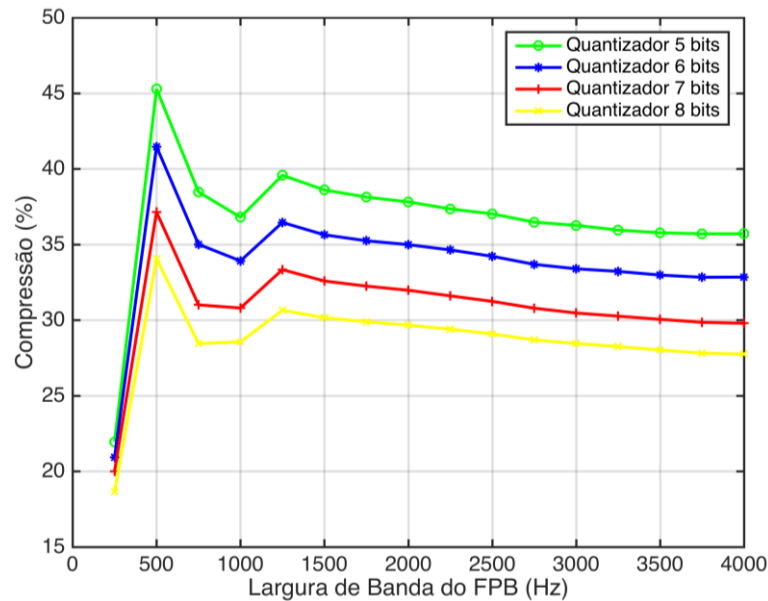


Fonte: Autor deste trabalho

4.2.2.1 Huffman

A Figura 4.17 ilustra as taxas de compressão alcançadas, por meio do algoritmo de codificação Huffman, para cada valor de largura de banda do FFB.

Figura 4.17 - Compressão após codificação Huffman e FPB, testMulher.wav



Fonte: Autor deste trabalho

A Tabela 5 mostra os pontos que geram os resultados mais interessantes em relação à codificação Huffman aliada ao FPB para o arquivo testMulher.wav.

Tabela 5 - Resultados para testMulher.wav, codificação Huffman e FPB

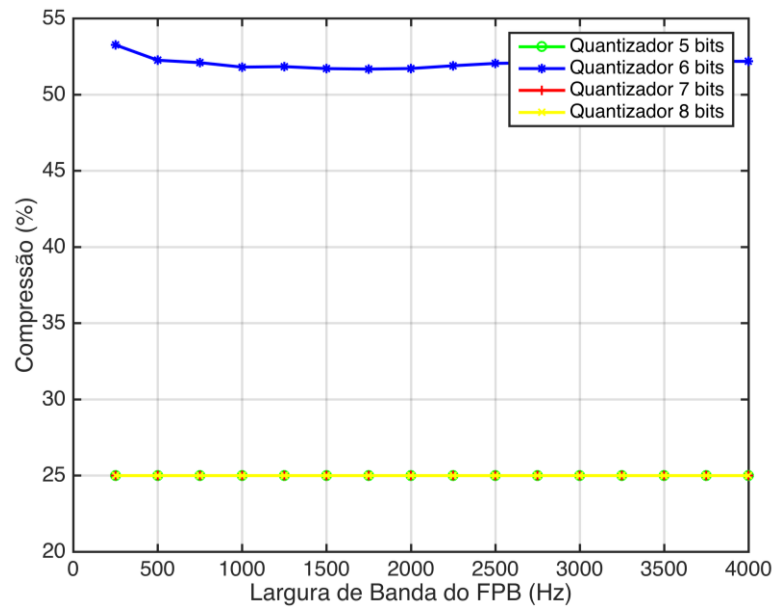
Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
5	500	9,52	1,30	112,08	78,33	35,75	45,05
5	1250	9,52	5,08	112,08	96,50	35,75	39,51

Por meio das figuras apresentadas e dos resultados da Tabela 5, é verificado que a filtragem utilizando um FPB de 500Hz aliada a um quantizador de 5 bits gera um ganho de compressão de 26,01%, sendo que mais uma vez o custo desse ganho é refletido na diminuição da SNR do sinal, de 9,52dB para 1,30dB. O outro ponto em destaque na mesma tabela se refere ao FPB de 1250Hz para um quantizador também de 5 bits, gerando um ganho menor em compressão, 10,51%, porém uma perda menor de SNR, de 9,52dB para 5,08dB.

4.2.2.2 Lempel-Ziv

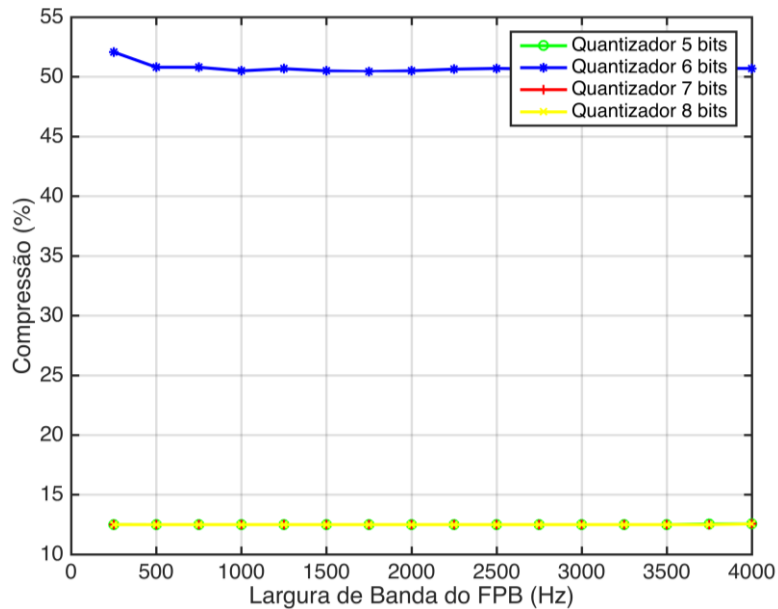
As Figuras 4.18, 4.19, 4.20 e 4.21 ilustram, respectivamente, as taxas de compressão alcançadas, por meio do algoritmo de codificação Lempel-Ziv, para cada valor de largura de banda do FPB, utilizando dicionários de 5, 6, 7 e 8 bits.

Figura 4.18 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, testMulher.wav



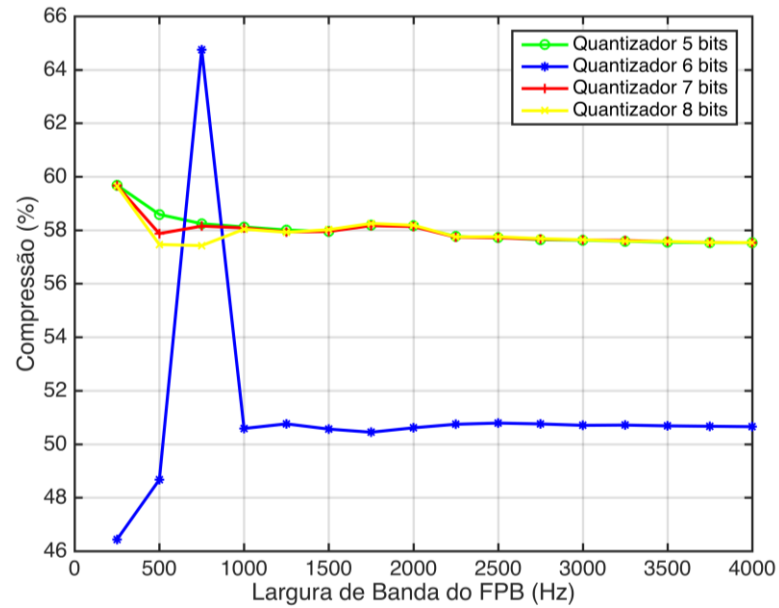
Fonte: Autor deste trabalho

Figura 4.19 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, testMulher.wav



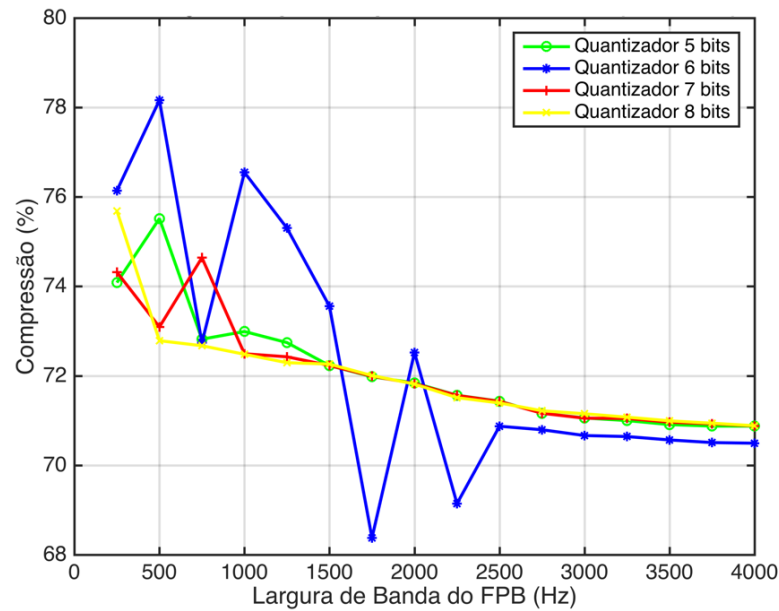
Fonte: Autor deste trabalho

Figura 4.20 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, testMulher.wav



Fonte: Autor deste trabalho

Figura 4.21 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, testMulher.wav



Fonte: Autor deste trabalho

A Tabela 6 mostra os pontos que geram os resultados mais interessantes em relação à codificação Lempel-Ziv aliada ao FPB para o arquivo testMulher.wav.

Tabela 6 - Resultados para testMulher.wav, codificação Lempel-Ziv e FPB

Dicionário (bits)	Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
7	6	750	12,60	3,01	125,20	85,50	50,63	64,80
8	6	500	12,85	1,30	124,80	78,33	70,48	78,35
8	6	1000	12,85	5,22	124,80	95,20	70,48	76,51
8	7	750	15,95	3,01	137,50	96,30	70,90	74,65

Por meio das figuras apresentadas e dos resultados da Tabela 6, é verificado que a filtragem utilizando um FPB de 750Hz, com codificação Lempel-Ziv de dicionário de 7 bits, aliados a um quantizador de 6 bits geram um ganho de compressão de 27,98%, mas para uma queda de SNR de aproximadamente 9dB. Um ponto que gera uma menor diminuição da SNR do sinal é a combinação de FPB de 1000Hz, dicionário de 8 bits e quantizador de 6 bits, na qual a SNR do sinal diminui de 12,85dB para 5,22dB, fazendo com que o ganho em compressão seja menor que o ponto anterior, atingindo 8,12% de ganho com a filtragem.

Importante destacar que para o arquivo testMulher.wav, assim como para o arquivo testPadrao.wav, sem o processo da filtragem, a codificação Lempel-Ziv utilizando dicionários de 7 e 8 bits gera compressões superiores à Huffman para todos os quantizadores. Já para Lempel-Ziv utilizando dicionários de 5 e 6 bits, as taxas de compressões geradas por meio da filtragem são todas inferiores às geradas por Huffman para todos os pontos, menos para quantizadores de 6 bits.

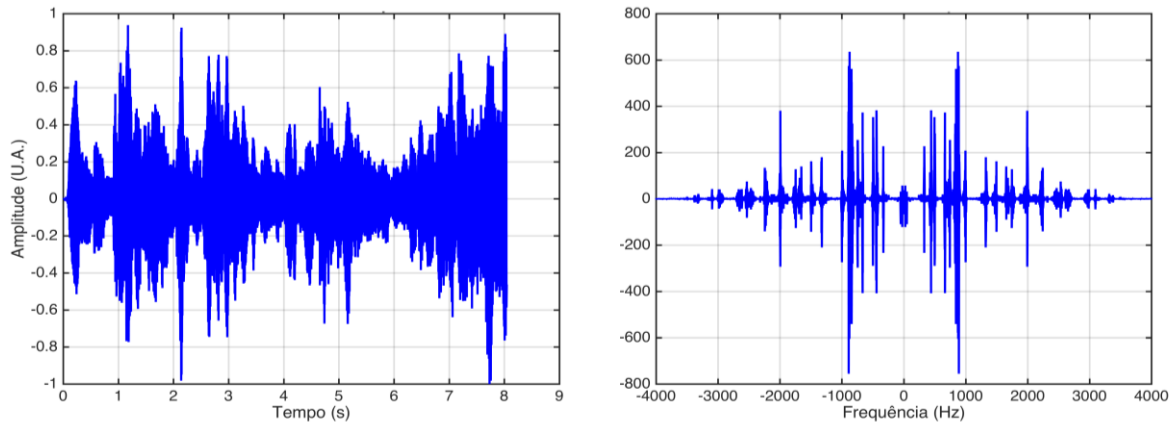
A similaridade dos resultados obtidos para testMulher.wav e testPadrao.wav pode ser fundamentada no fato de que ambos os áudios têm a mesma duração de 5 segundos e se tratam da mesma frase gravada, porém por vozes de gêneros diferentes, como mostrado na Tabela 1.

Mais uma vez é considerado a distinção dos comportamentos das curvas do quantizador de 6 bits ilustrados nas Figuras 4.18, 4.19 e 4.20. Não foi possível descobrir o motivo desses comportamentos. Na tentativa de se encontrar um motivo para o grande pico ilustrado na Figura 4.20, o passo adotado para o filtro foi diminuído, com o intuito de se verificar o comportamento da curva com maior exatidão e mesmo assim o comportamento é inconclusivo.

4.2.3 Classic.wav

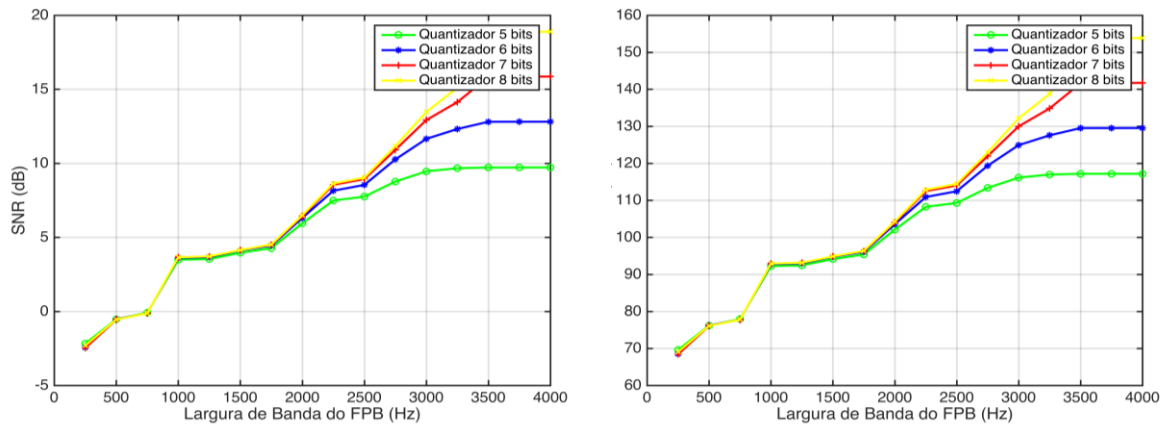
A Figura 4.22 ilustra o sinal representado no tempo e seu espectro de frequências. Já a Figura 4.23 ilustra a SNR e PSNR deste sinal quando submetido ao filtro.

Figura 4.22 - Sinal original no tempo (à esquerda) e espectro de frequências (à direita), Classic.wav



Fonte: Autor deste trabalho

Figura 4.23 - SNR (à esquerda) e PSNR (à direita), Classic.wav

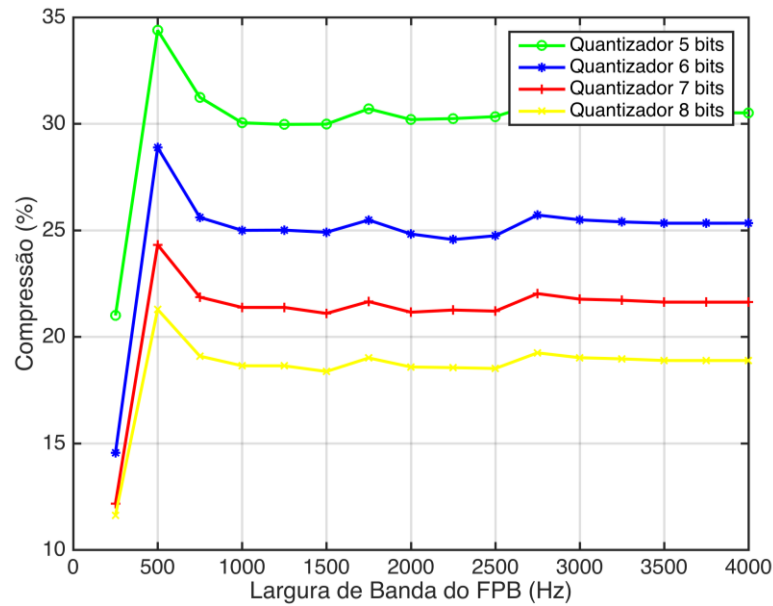


Fonte: Autor deste trabalho

4.2.3.1 Huffman

A Figura 4.24 ilustra as taxas de compressão alcançadas, por meio do algoritmo de codificação Huffman, para cada valor de largura de banda do FPB.

Figura 4.24 - Compressão após codificação Huffman e FPB, Classic.wav



Fonte: Autor deste trabalho

Assim como para os arquivos anteriores, tem-se a Tabela 7 mostrando o ponto que gera o resultado mais interessante em relação à codificação Huffman aliada ao FPB para o arquivo Classic.wav.

Tabela 7 - Resultados para Classic.wav, codificação Huffman e FPB

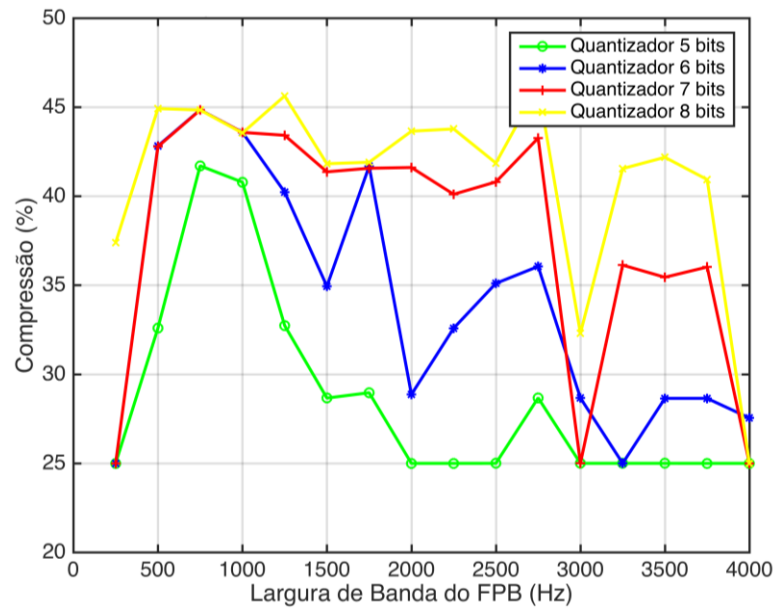
Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
5	500	9,70	-0,35	116,61	76,00	30,52	34,41

Por meio das figuras apresentadas e do resultado da Tabela 7, é verificado que a filtragem utilizando um FPB de 500Hz aliada a um quantizador de 5 bits gera um ganho de compressão de 12,74%, sendo que mais uma vez o custo desse ganho é refletido na diminuição da SNR do sinal, de 9,70dB para -0,35dB. A diminuição da SNR nesse caso, após o processo de filtragem, chegou a um ponto no qual o nível de ruído é maior do que o nível do próprio sinal, resultando em uma SNR negativa. Isso acontece porque o espectro de frequências do sinal Classic.wav é mais espalhado do que o espectro do sinal testPadrao.wav por exemplo, não possuindo uma concentração tão grande do sinal em baixas frequências. Portanto, uma largura de banda de 500Hz corta a maior parte do sinal, gerando uma SNR menor do que zero. Para larguras de banda diferentes de 500Hz, a codificação Huffman aliada à filtragem FPB não gera ganhos de compressão significativos.

4.2.3.2 Lempel-Ziv

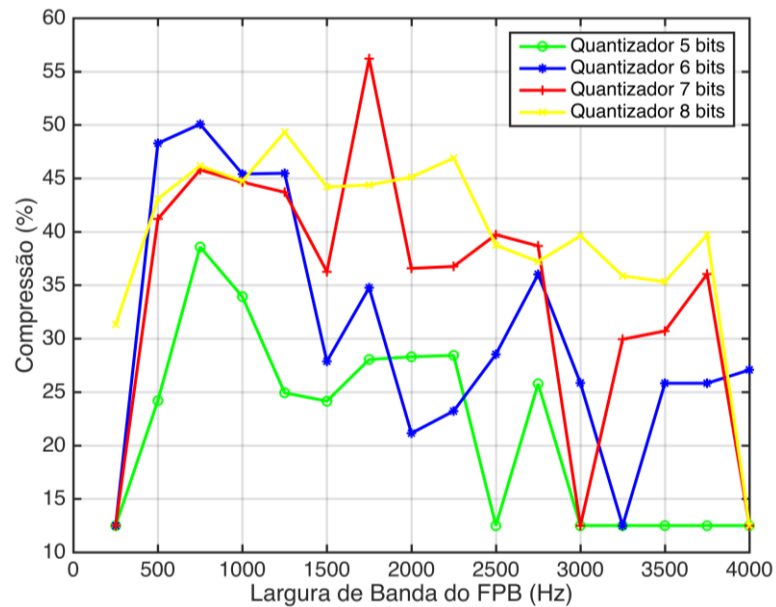
As Figuras 4.25, 4.26, 4.27 e 4.28 ilustram, respectivamente, as taxas de compressão alcançadas por meio do algoritmo de codificação Lempel-Ziv, para cada valor de largura de banda do FPB, utilizando dicionários de 5, 6, 7 e 8 bits.

Figura 4.25 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, Classic.wav



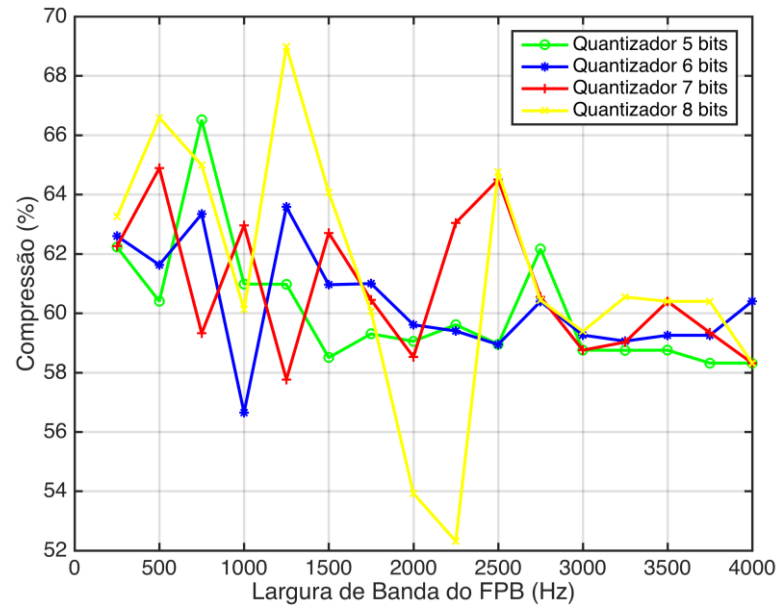
Fonte: Autor deste trabalho

Figura 4.26 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, Classic.wav



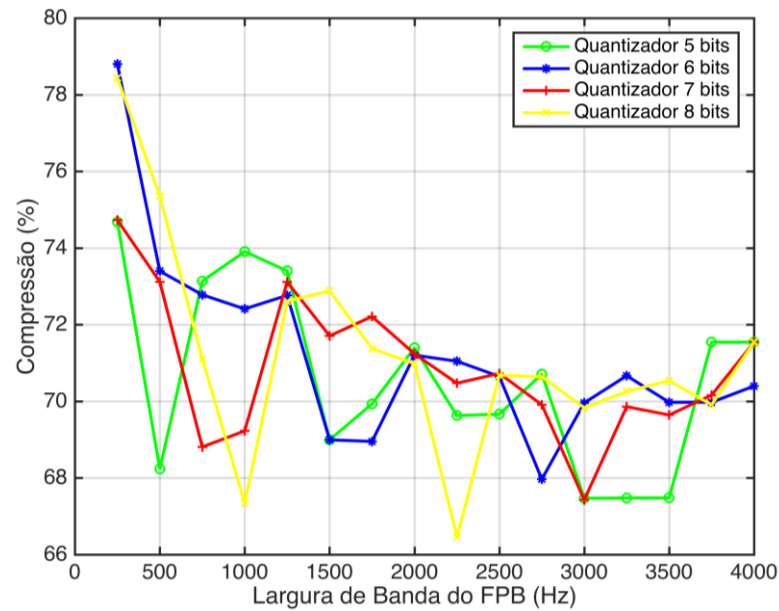
Fonte: Autor deste trabalho

Figura 4.27 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, Classic.wav



Fonte: Autor deste trabalho

Figura 4.28 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, Classic.wav



Fonte: Autor deste trabalho

A Tabela 8 mostra os pontos que geram os resultados mais interessantes em relação à codificação Lempel-Ziv aliada ao FPB para o arquivo testMulher.wav.

Tabela 8 - Resultados para Classic.wav, codificação Lempel-Ziv e FPB

Dicionário (bits)	Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
5	8	1250	18,95	3,75	154,35	92,75	25,00	45,62
5	7	2750	15,95	10,88	141,05	122,05	25,00	43,60
6	7	1750	15,95	4,50	141,05	96,18	12,50	66,25
6	8	2250	18,95	8,62	154,35	113,00	12,50	46,80
7	5	750	9,83	0,00	76,95	117,20	58,30	66,50
7	8	1250	18,95	3,75	154,35	92,75	58,30	68,95
8	5	1000	9,83	3,50	76,95	92,40	71,60	73,90

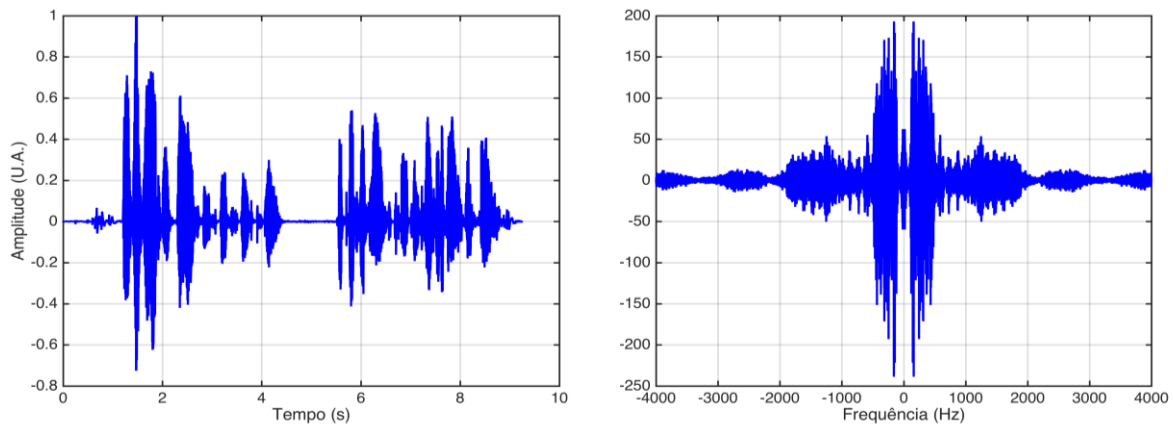
Por meio das figuras apresentadas e dos resultados da Tabela 8, é verificado que a filtragem utilizando um FPB de 1750Hz, com dicionário de 6 bits, aliados a um quantizador de 7 bits gera um ganho de compressão incrível de 430%, para uma queda de SNR de 15,95dB para 4,50dB, ou seja, o nível de sinal ainda é consideravelmente maior que o nível de ruído para um ganho de compressão tão grande. Outro ponto que merece destaque, por gerar ganho de compressão considerável em detrimento de uma baixa diminuição de SNR é a filtragem utilizando um FPB de 2750Hz, com codificação Lempel-Ziv de dicionário de 5 bits e quantizador de 7 bits. Tal configuração alcança um ganho de compressão de 74,4% para uma diminuição de SNR de 15,95dB para 10,88dB.

Um aspecto importante a respeito das variações das taxas de compressão alcançadas em função das larguras de banda do filtro é que, para o arquivo Classic.wav, a variação dessas taxas, para dicionários de 7 e 8 bits, são muito maiores do que as variações visualizadas para os arquivos anteriores, testPadrao.wav e testMulher.wav. Como o sinal estudado nesta seção possui um espectro de frequências mais espalhado se comparado aos anteriores, variações na largura de banda do filtro utilizado causam maior influência nas taxas de compressão, pois mesmo para frequências não tão baixas, existe parcela significativa de sinal a ser comprimida.

4.2.4 HarvardSentenceMan.wav

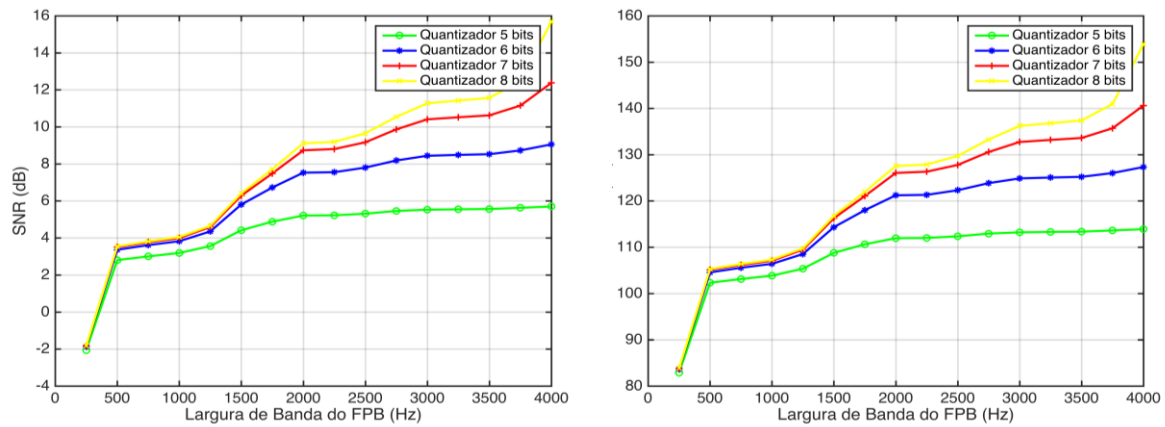
A Figura 4.29 ilustra o sinal em questão representado no tempo e seu espectro de frequências. Já a Figura 4.30 mostra a SNR e PSNR deste sinal quando submetido ao filtro.

Figura 4.29 - Sinal original no tempo (à esquerda) e espectro de frequências (à direita), HarvardSentenceMan.wav



Fonte: Autor deste trabalho

Figura 4.30 - SNR (à esquerda) e PSNR (à direita), HarvardSentenceMan.wav

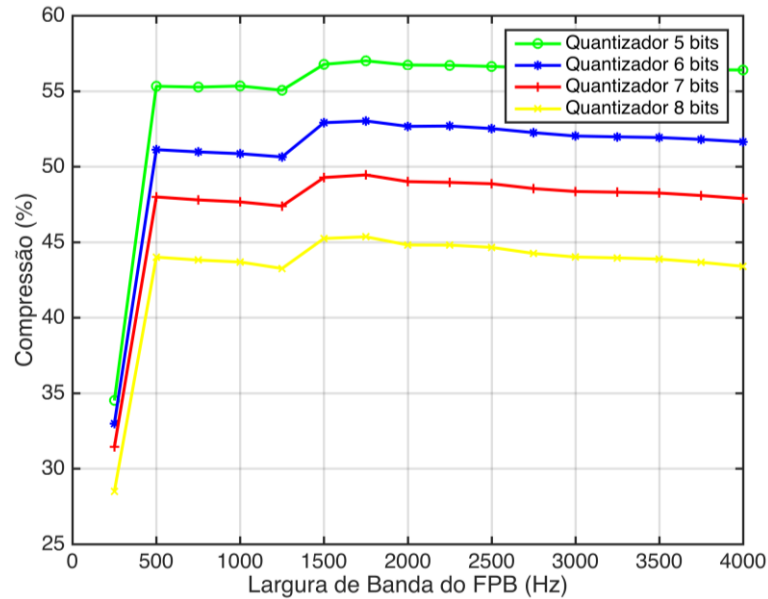


Fonte: Autor deste trabalho

4.2.4.1 Huffman

A Figura 4.31 ilustra as taxas de compressão alcançadas, por meio do algoritmo de codificação Huffman, para cada valor de largura de banda do FPB.

Figura 4.31 - Compressão após codificação Huffman e FPB, HarvardSentenceMan.wav



Fonte: Autor deste trabalho

A Tabela 9 mostra o ponto que gera o resultado mais interessante em relação à codificação Huffman aliada ao FPB para o arquivo HarvardSentenceMan.wav.

Tabela 9 - Resultados para HarvardSentenceMan.wav, codificação Huffman e FPB

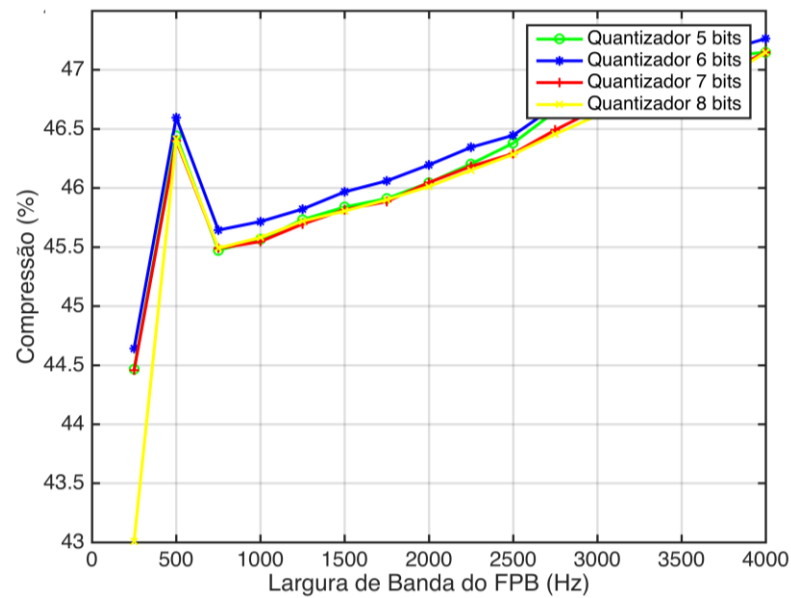
Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
8	1750	15,71	4,75	122,00	153,75	43,3	45,04

Por meio das figuras apresentadas, principalmente da Figura 4.31, é observado que a codificação Huffman sem filtragem gera taxas de compressão entre 40% e 60% para o arquivo considerado. Quanto menor o número de bits do quantizador, maior é a taxa de compressão alcançada. Ao passo que, utilizando o filtro e analisando o comportamento da taxa de compressão em função da largura de banda desse filtro, não são alcançados ganhos de compressão consideráveis. A Tabela 9 traz um único ponto que representa o melhor cenário de ganho de compressão para a configuração em questão. Utilizando um FPB de 1750Hz em conjunto com um quantizador de 8 bits, é gerado um ganho de compressão de 4,02% e uma diminuição da SNR do sinal, de 15,71dB para 4,75dB.

4.2.4.2 Lempel-Ziv

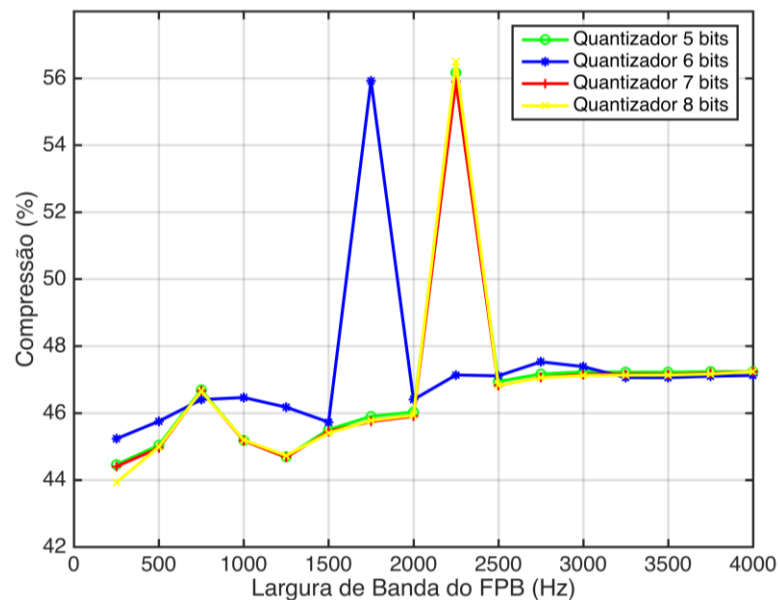
As Figuras 4.32, 4.33, 4.34 e 4.35 ilustram, respectivamente, as taxas de compressão alcançadas, por meio do algoritmo de codificação Lempel-Ziv, para cada valor de largura de banda do FPB, utilizando dicionários de 5, 6, 7 e 8 bits.

Figura 4.32 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, HarvardSentenceMan.wav



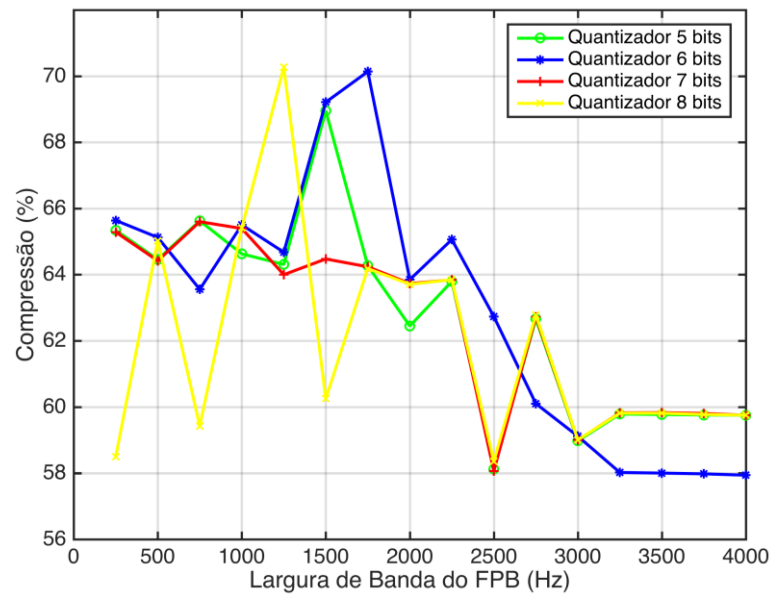
Fonte: Autor deste trabalho

Figura 4.33 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, HarvardSentenceMan.wav



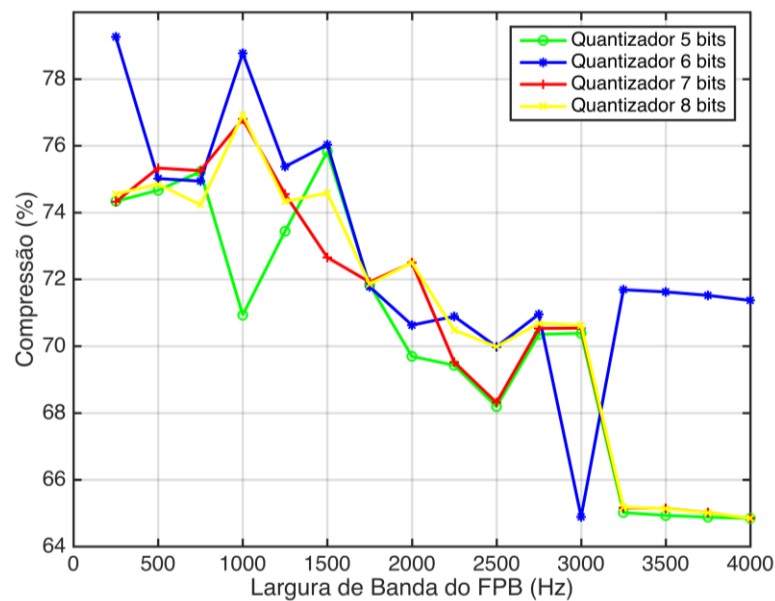
Fonte: Autor deste trabalho

Figura 4.34 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, HarvardSentenceMan.wav



Fonte: Autor deste trabalho

Figura 4.35 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, HarvardSentenceMan.wav



Fonte: Autor deste trabalho

A Tabela 10 mostra os pontos que geram os resultados mais interessantes em relação à codificação Lempel-Ziv aliada ao FPB para o arquivo HarvardSentenceMan.wav.

Tabela 10 - Resultados para HarvardSentenceMan.wav, codificação Lempel-Ziv e FPB

Dicionário (bits)	Quantizador (bits)	BW (Hz)	SNR₁ (dB)	SNR₂ (dB)	PSNR₁ (dB)	PSNR₂ (dB)	COMP₁ (%)	COMP₂ (%)
6	6	1750	9,10	6,72	128,10	118,12	47,10	54,90
6	8	2250	15,75	9,21	153,80	127,75	47,25	56,50
7	6	1750	9,10	6,72	128,10	118,12	57,96	70,20
7	6	2250	9,10	7,62	121,25	113,00	57,96	65,12
7	8	1250	15,75	4,65	153,80	109,75	59,75	70,32
8	5	1500	5,75	4,40	113,78	108,77	64,82	75,85
8	6	1000	9,12	6,90	127,50	106,11	71,37	78,86
8	8	1000	15,75	4,09	153,80	107,01	64,82	76,92

Por meio das figuras apresentadas e dos resultados da Tabela 10, é verificado, por exemplo, que a filtragem utilizando um FPB de 1750Hz, com codificação Lempel-Ziv de dicionário de 7 bits, aliados a um quantizador de 6 bits gera um ganho de compressão de 21,11%, para uma queda de SNR de apenas 2,38dB. Outro ponto que gera um resultado interessante, por gerar uma perda de SNR baixa, é a filtragem utilizando um FPB de 1500Hz, com codificação Lempel-Ziv de dicionário de 8 bits e quantizador de 5 bits, alcançando um ganho de compressão de 17,02% para uma diminuição de SNR de somente 1,35dB.

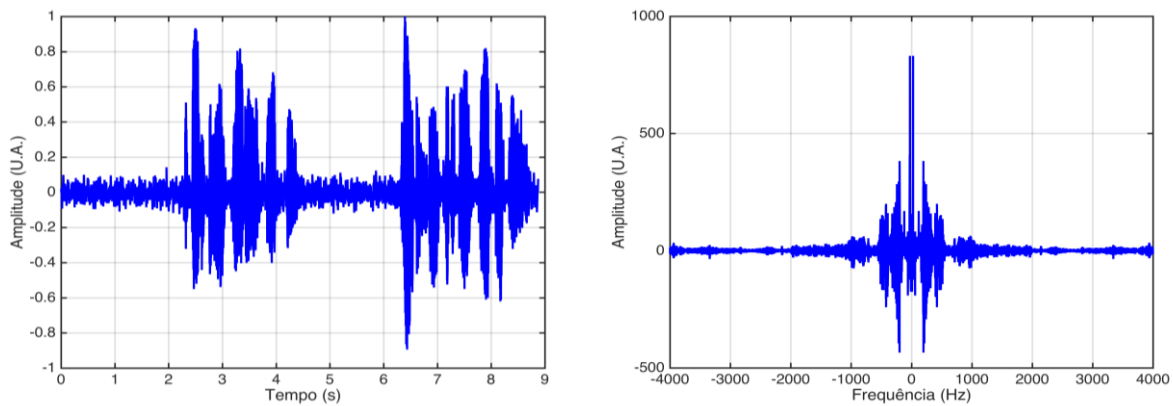
A utilização de codificação Lempel-Ziv com dicionário de 5 bits não gera ganho de compressão para o arquivo HarvardSentenceMan.wav em nenhuma largura de banda do filtro FPB. Observando a Figura 4.30 pode-se observar também que a variação da SNR em função da largura de banda do filtro, para quantizadores de 5 e 6 bits, é menor se comparada às variações dos casos anteriores.

A Figura 4.33 ilustra um grau de variação das curvas de compressão muito diferente das outras curvas geradas para esta mesma configuração que está sendo analisada. Na tentativa de se encontrar um motivo para os grandes picos ilustrados na Figura 4.33, o passo adotado para o filtro foi diminuído novamente, com o intuito de se verificar os comportamentos das curvas com maior exatidão e mesmo assim os comportamentos são inconclusivos.

4.2.5 HarvardSentenceWoman.wav

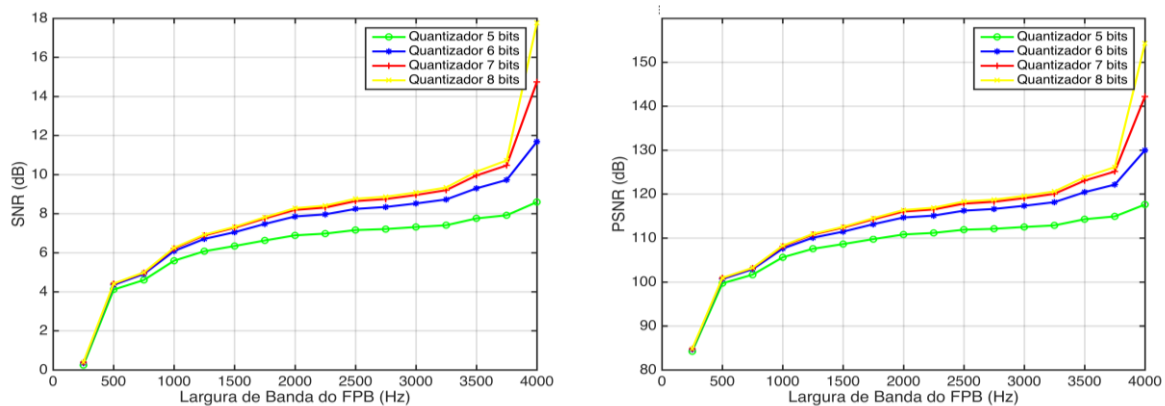
A Figura 4.36 ilustra o sinal HarvardSentenceWoman.wav representado no tempo e seu espectro de frequências. A Figura 4.37 por sua vez, ilustra a SNR e PSNR deste sinal quando submetido ao filtro.

Figura 4.36 - Sinal original no tempo (à esquerda) e espectro de frequências (à direita), HarvardSentenceWoman.wav



Fonte: Autor deste trabalho

Figura 4.37 - SNR (à esquerda) e PSNR (à direita), HarvardSentenceWoman.wav

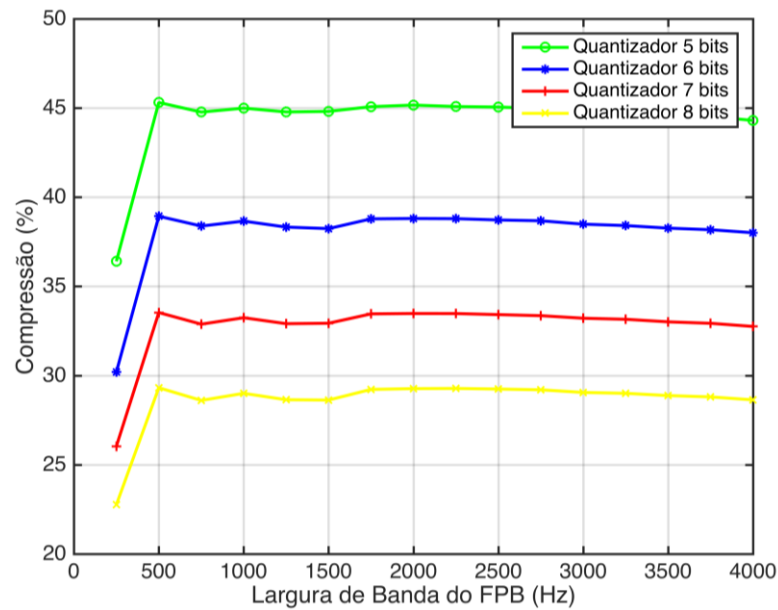


Fonte: Autor deste trabalho

4.2.5.1 Huffman

A Figura 4.38 ilustra as taxas de compressão alcançadas, por meio do algoritmo de codificação Huffman, para cada valor de largura de banda do FPB.

Figura 4.38 - Compressão após codificação Huffman e FPB, HarvardSentenceWoman.wav



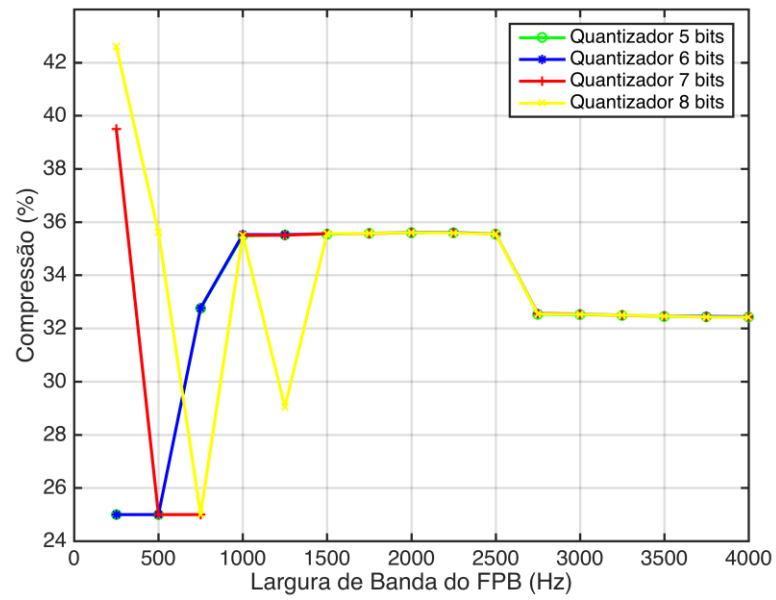
Fonte: Autor deste trabalho

Por meio da Figura 4.38 é observado que a codificação Huffman sem filtragem gera taxas de compressão entre aproximadamente 25 e 45% para o arquivo considerado. Quanto menor o número de bits do quantizador, maior é a taxa de compressão alcançada. O comportamento das curvas geradas na Figura 4.38, em relação à variação, é bastante semelhante ao comportamento das curvas geradas na Figura 4.31 para HarvardSentenceMan.wav. Em relação aos resultados atingidos com a filtragem, pode ser observado que não são alcançados ganhos de compressão por meio desse processo, independentemente da largura de banda do mesmo.

4.2.5.2 Lempel-Ziv

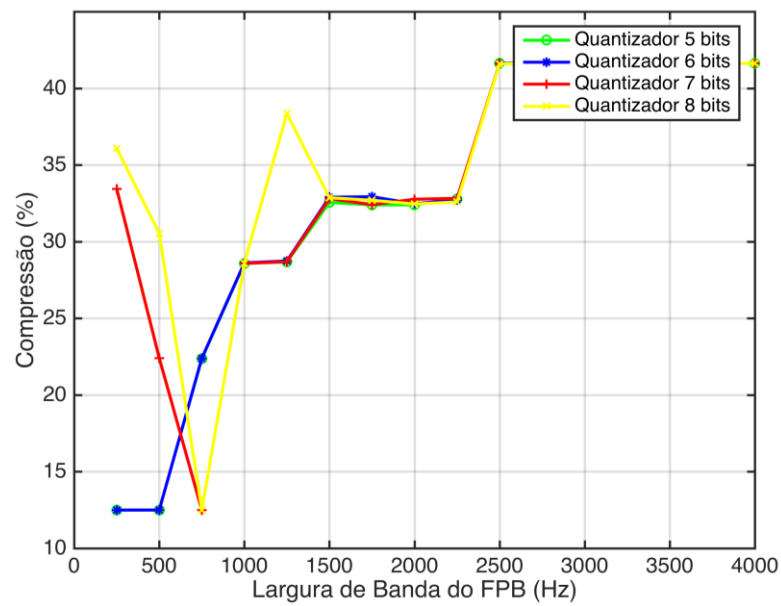
As Figuras 4.39, 4.40, 4.41 e 4.42 ilustram, respectivamente, as taxas de compressão alcançadas, por meio do algoritmo de codificação Lempel-Ziv, para cada valor de largura de banda do FPB, utilizando dicionários de 5, 6, 7 e 8 bits.

Figura 4.39 - Compressão após codificação Lempel-Ziv (5 bits) e FPB, HarvardSentenceWoman.wav



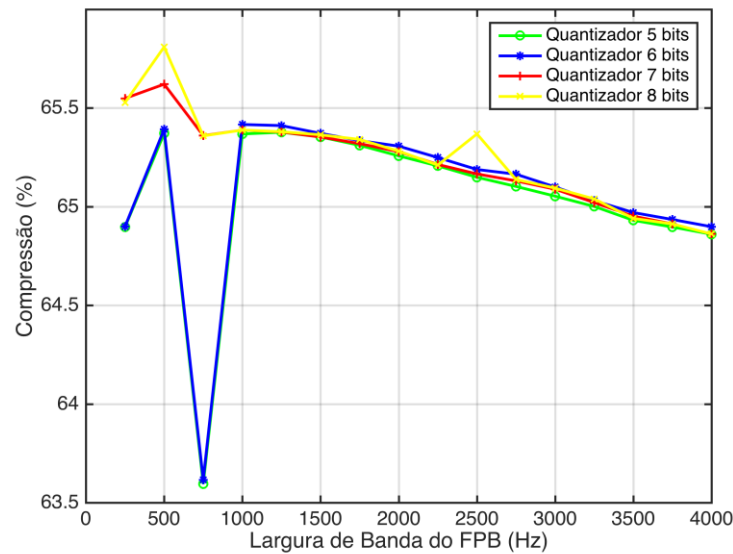
Fonte: Autor deste trabalho

Figura 4.40 - Compressão após codificação Lempel-Ziv (6 bits) e FPB, HarvardSentenceWoman.wav



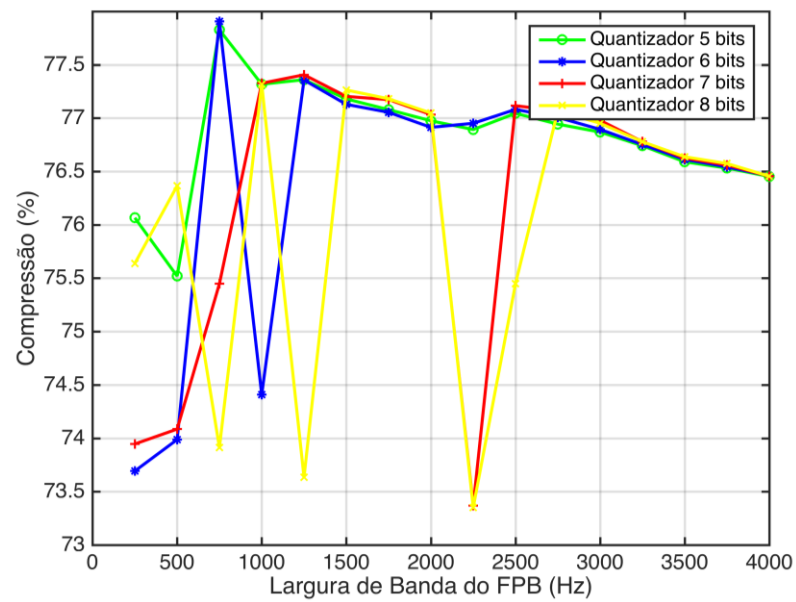
Fonte: Autor deste trabalho

Figura 4.41 - Compressão após codificação Lempel-Ziv (7 bits) e FPB, HarvardSentenceWoman.wav



Fonte: Autor deste trabalho

Figura 4.42 - Compressão após codificação Lempel-Ziv (8 bits) e FPB, HarvardSentenceWoman.wav



Fonte: Autor deste trabalho

A Tabela 11 mostra os pontos que geram os resultados mais interessantes em relação à codificação Lempel-Ziv aliada ao FPB para o arquivo HarvardSentenceWoman.wav.

Tabela 11 - Resultados para HarvardSentenceWoman.wav, codificação Lempel-Ziv e FPB

Dicionário (bits)	Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
5	5	2500	9,10	6,72	128,10	118,12	32,44	35,55
8	6	750	8,57	7,15	130,00	116,31	76,42	77,90

Por meio das figuras apresentadas e dos resultados da Tabela 11, pode ser observado que a filtragem tem pouca influência nas taxas de compressão para o arquivo HarvardSentenceWoman.wav. O melhor resultado em ganho de compressão é observado utilizando-se um FPB de 2500Hz, com codificação Lempel-Ziv de dicionário de 5 bits, em conjunto com um quantizador também de 5 bits, alcançando um ganho de compressão de 9,58%, para uma queda de SNR de 9,10dB para 6,72dB. O outro ponto que consta na tabela e que gera uma perda de SNR muito baixa é por meio de filtragem utilizando um FPB de 750Hz, com codificação Lempel-Ziv de dicionário de 8 bits e quantizador de 6 bits, alcançando um ganho de compressão também muito baixo de 1,93%.

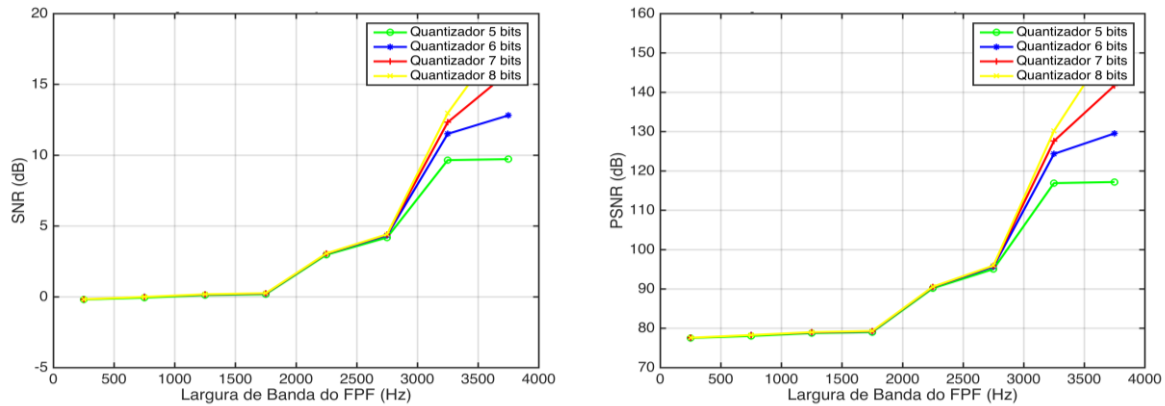
Enquanto a codificação Lempel-Ziv com dicionário de 8 bits tem uma das taxas de compressão menos influenciadas pela filtragem de todo este trabalho, essa mesma codificação gera a maior taxa de compressão atingida sem o processo de filtragem para todos os valores de quantizadores. Além disso o espectro de frequências de HarvardSentenceWoman.wav é o mais concentrado nas baixas frequências, com amplitudes bem mais altas que os outros sinais. Isso explica as grandes variações de taxas de compressão para as frequências mais baixas vistas nas Figuras 4.39, 4.40, 4.41 e 4.42.

4.3 FILTRO PASSA-FAIXA (FPF)

O arquivo escolhido para fazer a análise da influência do filtro passa-faixa na taxa de compressão foi o Classic.wav. O motivo dessa escolha se baseia no fato de que esse arquivo demonstrou a maior influência em taxa de compressão causada pela filtragem FPB, sendo assim, agora será aplicado a ele o FPF para ser verificado a possibilidade de alcançar resultados ainda mais significativos.

O sinal representado no tempo e seu espectro de frequências podem ser visualizados na Figura 4.22, já mostrada anteriormente. Já a Figura 4.43 ilustra a SNR e PSNR deste sinal quando submetido ao FPF.

Figura 4.43 - SNR (à esquerda) e PSNR (à direita), Classic.wav



Fonte: Autor deste trabalho

O filtro passa-faixa, como ilustrado na Figura 2.3, foi implementado com os valores iniciais das frequências de corte f_{c1} e f_{c2} , como sendo 1750Hz e 2000Hz respectivamente, ou seja, a largura de banda inicial desse filtro, primeiro ponto da Figura 4.43 e 4.44, corresponde à 250Hz. O passo utilizado na implementação do filtro continua sendo 250Hz assim como no FPB, porém f_{c2} aumenta de 250 em 250Hz e a f_{c1} , por sua vez, decresce de 250 em 250Hz. Sendo assim, a cada novo ponto nos gráficos, tem-se um aumento de 500Hz na largura de banda do FPF.

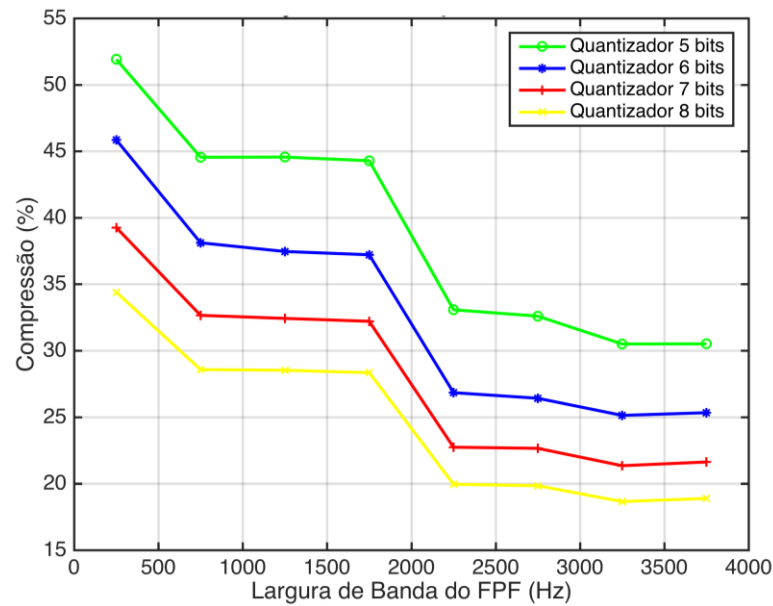
4.3.1 Huffman

A Figura 4.44 ilustra as taxas de compressão alcançadas, por meio do algoritmo de codificação Huffman, para cada valor de largura de banda do FPF. A Tabela 12 mostra os pontos que geram os resultados mais interessantes em relação à codificação Huffman aliada ao FPF para o arquivo Classic.wav, todos eles foram para um quantizador de 5 bits.

Tabela 12 - Resultados para Classic.wav, codificação Huffman e FPF

Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
5	1750	9,71	0,17	117,17	79,00	30,04	44,28
5	2250	9,71	2,96	117,17	90,15	30,04	33,07
5	2750	9,71	4,18	117,17	95,02	30,04	32,59

Figura 4.44 - Compressão após codificação Huffman e FPF, Classic.wav



Fonte: Autor deste trabalho

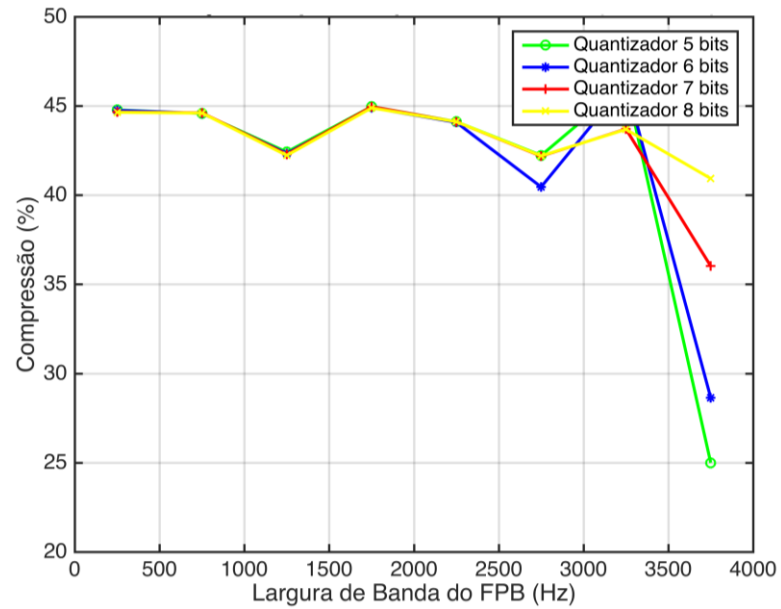
Por meio das figuras e da Tabela 11, pode ser observado que o quantizador de 5 bits gera os melhores ganhos em compressão em função da largura de banda do FPF quando se utiliza Huffman como algoritmo de compressão para o arquivo Classic.wav. Para uma BW de 1750 Hz por exemplo, a filtragem consegue um ganho de compressão de 47,4%, mas a SNR do sinal se reduz a praticamente zero. Com o aumento da BW do filtro, para o mesmo quantizador, consegue-se amenizar essa diminuição de SNR do sinal, mas são alcançadas taxas de compressão menores. Um desses pontos ocorre para uma BW de 2750Hz, no qual a filtragem gera um ganho de compressão de 8,48% para uma pequena redução de SNR de 9,71dB para 4,18dB.

A Figura 4.43, quando comparada à Figura 4.23, ilustra que para o arquivo Classic.wav, as componentes de frequências mais altas não influenciam tanto a SNR do sinal como as componentes de baixas frequências. Como o FPF em questão tem sua primeira BW localizada próximo a parte central do espectro do sinal, as componentes nas extremidades são cortadas no início do processo de filtragem e somente incluídas totalmente para a maior BW do filtro. A Figura 4.43 ilustra que a SNR do sinal não tem variação significativa entre 250Hz e 1750Hz, ou seja, enquanto as componentes das extremidades não são incluídas, a SNR do sinal não melhora. Como o FPF inclui as baixas desde os pontos iniciais da filtragem, o comportamento é o contrário. A Figura 4.23 ilustra que a variação da SNR, para quantizadores de 5 e 6 bits, começa a saturar a medida que a BW do filtro aumenta.

4.3.2 Lempel-Ziv

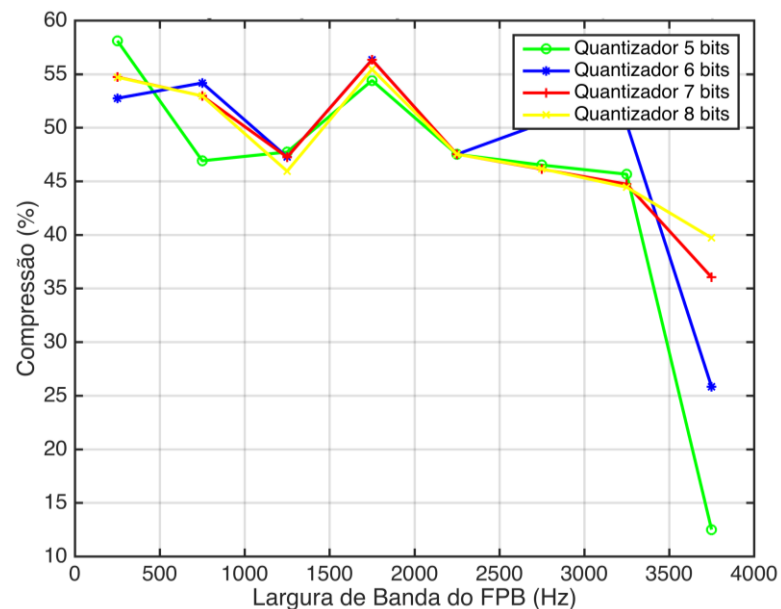
As Figuras 4.45, 4.46, 4.47 e 4.48, ilustram, respectivamente, as taxas de compressão alcançadas por meio do algoritmo de codificação Lempel-Ziv, para cada valor de BW do filtro passa-faixa, utilizando 5, 6, 7 e 8 bits.

Figura 4.45 - Compressão após codificação Lempel-Ziv (5 bits) e FPF, Classic.wav



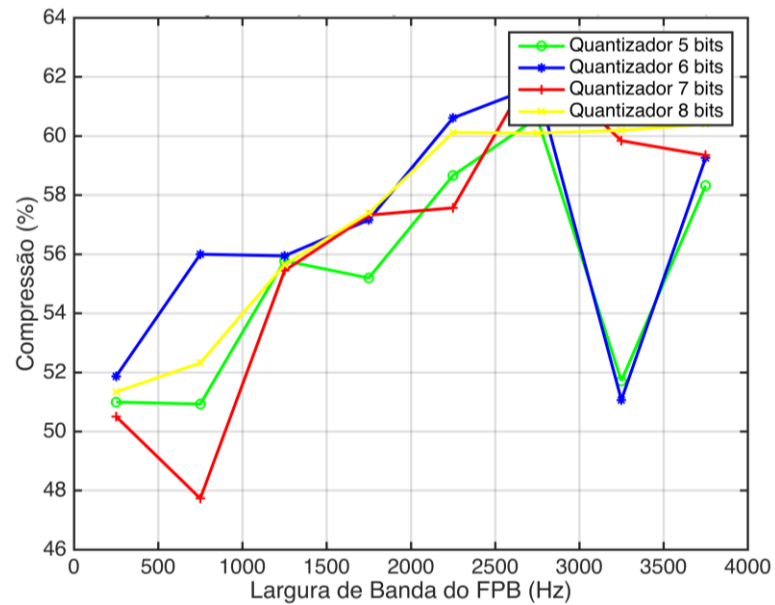
Fonte: Autor deste trabalho

Figura 4.46 - Compressão após codificação Lempel-Ziv (6 bits) e FPF, Classic.wav



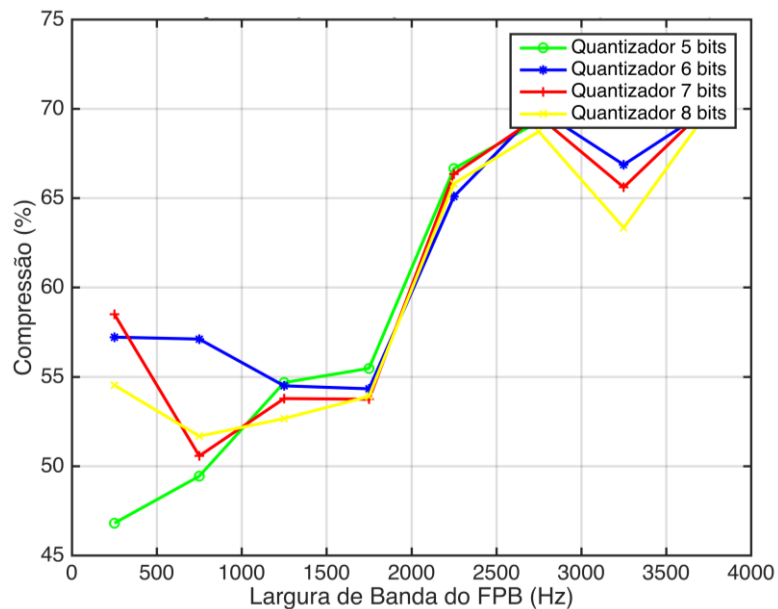
Fonte: Autor deste trabalho

Figura 4.47 - Compressão após codificação Lempel-Ziv (7 bits) e FPF, Classic.wav



Fonte: Autor deste trabalho

Figura 4.48 - Compressão após codificação Lempel-Ziv (8 bits) e FPF, Classic.wav



Fonte: Autor deste trabalho

A Tabela 13 mostra os pontos que geram os resultados mais interessantes em relação à codificação Lempel-Ziv para o arquivo em questão.

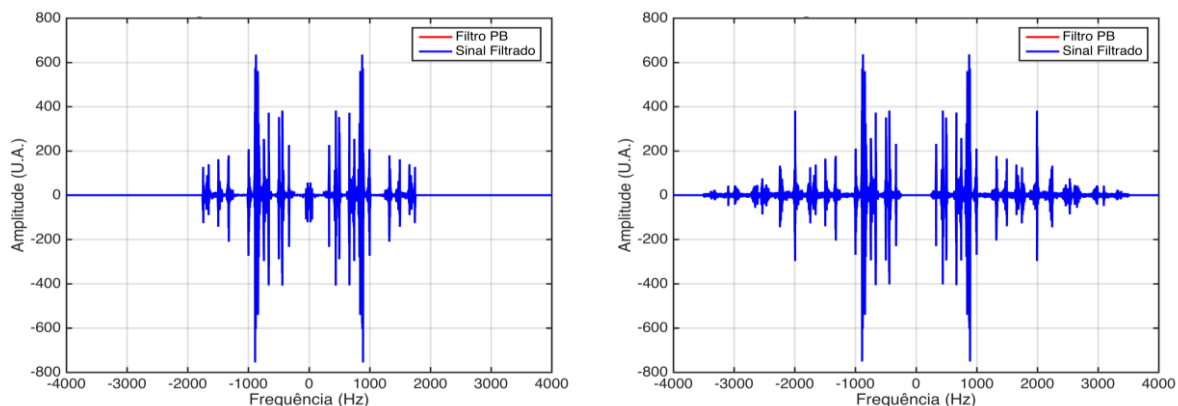
Tabela 13 - Resultados para Classic.wav, codificação Lempel-Ziv e FPF

Dicionário (bits)	Quantizador (bits)	BW (Hz)	SNR ₁ (dB)	SNR ₂ (dB)	PSNR ₁ (dB)	PSNR ₂ (dB)	COMP ₁ (%)	COMP ₂ (%)
5	6	3250	12,67	12,33	129,48	124,47	28,80	46,32
6	5	1750	9,65	0,15	117,20	79,02	12,58	54,41
6	5	3250	9,65	9,58	117,20	117,00	12,58	45,71

Observando as figuras e a Tabela 12, é constatado que a utilização de FPF antes da codificação por Lempel-Ziv gera ganhos de compressão muito interessantes para alguns tamanhos de dicionários. Utilizando um dicionário de 5 bits, associado à um quantizador de 6 bits e filtragem de BW igual a 3250, é possível alcançar um ganho de compressão de 60,83% sendo que a diminuição da SNR é muito baixa, 0,34dB. Um ponto que chega a alcançar ganho de compressão ainda maior é utilização de um dicionário de 6 bits, quantizador de 5 bits e filtro passa-faixa de 1750Hz de largura de banda. Essa configuração garante um ganho de compressão de 332,51% para o arquivo Classic.wav, embora reduza sua SNR para 0,15dB após essa filtragem. O último ponto da Tabela 12, e talvez o mais interessante de todos, se trata da utilização de codificação Lempel-Ziv com dicionário de 6 bits, quantizador de 5 bits e largura de banda do FPF de 3250Hz. Essa combinação garante um ganho impressionante de compressão de 263,35% ao passo que a SNR e PSNR do sinal praticamente não se alteram, diminuindo somente 0,07dB e 0,20dB respectivamente.

A Figura 4.49 contém o espectro de frequências do sinal Classic.wav filtrado para as duas configurações que geraram os resultados mais expressivos.

Figura 4.49 - FPB de 1750Hz (à esquerda) e FPF de 3250Hz (à direita)



Fonte: Autor deste trabalho

Observando a Figura 4.49 é possível notar que algumas componentes de frequência do sinal Classic.wav têm papel fundamental nos resultados alcançados por meio das filtragens. As componentes muito próximas a 2000Hz por exemplo, representam parcela significativa de informação para o sinal, pois quando retirada, como acontece na filtragem passa-baixa de 1750Hz, o ganho de compressão alcançado é muito alto (430%) mas ocorre redução considerável da SNR do sinal, ou seja, o nível de potência do sinal diminui mais que o nível de potência do ruído para tal filtragem. Já as componentes muito próximas de 0Hz provavelmente possuem caráter ruidoso, pois quando retiradas, como acontece na filtragem passa-faixa de 3250Hz, além de se alcançar um ganho de compressão alto (263,35%), a SNR do sinal praticamente não se altera.

5. CONCLUSÃO

Este estudo mostra a superioridade do algoritmo de codificação Lempel-Ziv sobre o algoritmo de codificação Huffman para a compressão dos sinais de áudio aqui estudados. Ao analisar as taxas de compressão alcançadas pelos dois algoritmos sem o processo de filtragem, é possível perceber que, utilizando Lempel-Ziv com dicionário de 7 ou 8 bits, todos os resultados são superiores aos alcançados por Huffman. Ainda para os cenários sem filtragem, na maioria das vezes, quanto maior o tamanho do dicionário, maiores são as taxas de compressão atingidas pelo algoritmo Lempel-Ziv.

Lempel-Ziv também se mostra superior a Huffman quando os dois algoritmos são utilizados para codificar os sinais de áudio após a filtragem. Seja utilizando o filtro passa-baixa ou utilizando o filtro passa-faixa, Lempel-Ziv alcança ganhos de compressão muito mais altos do que os alcançados por Huffman. Sendo assim, entende-se que o desempenho de compressão do algoritmo Lempel-Ziv sofre maior influência do processo de filtragem do que o desempenho de compressão do algoritmo Huffman.

Para o arquivo Classic.wav por exemplo, que dentre os arquivos utilizados neste trabalho foi o que gerou os melhores resultados, utilizando-se Lempel-Ziv associado ao filtro passa-baixa é possível alcançar ganhos de compressão muito altos. Ao se utilizar o filtro passa-faixa na tentativa de melhorar ainda mais o desempenho do algoritmo Lempel-Ziv para o arquivo em questão, é factível que o ganho de compressão ainda atinge níveis que surpreendem e a queda de SNR resultante é praticamente nula.

Apesar deste trabalho mostrar alguns aspectos comuns entre os resultados atingidos para os diferentes arquivos, como por exemplo a relação entre o número de bits do quantizador e a SNR, mostrando que quanto maior o número de bits do quantizador maior é a SNR alcançada, outros resultados dependem de características específicas de cada áudio. Uma dessas características específicas que merece destaque é o espectro de frequências de cada sinal. Na maioria dos casos, não foi observado um comportamento padrão entre taxas de compressão alcançadas em função da largura de banda dos filtros. De acordo com a distribuição espectral de cada sinal de áudio, certas larguras de banda geram resultados mais significativos do que outras, não existindo assim uma largura de banda ótima que atenda a maioria dos sinais.

Salve algumas exceções, quanto maior o ganho de compressão, maior a perda de SNR do sinal, sendo que essa diminuição de SNR é aceitável ou não de acordo com os limites necessários

para cada aplicação. Na grande maioria dos casos aqui estudados, mesmo para resultados com SNR baixa, os sinais de áudio continuam inteligíveis, apesar do nível de ruído agregado durante o processo. Portanto, caso o objetivo da aplicação seja economizar o máximo de banda possível e recursos de armazenamento, se o sinal de áudio resultante é compreensível mesmo com uma SNR considerada baixa, o processo de filtragem antes da codificação gera resultados compatíveis com aplicações que objetivam essa economia.

REFERÊNCIAS

- [1] BEDRUZ, R. A.; QUIROS, A. R. F. Comparison of Huffman Algorithm and Lempel-Ziv Algorithm for Audio, Image and Text Compression. **8th IEEE International Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management (HNICEM)**, 12 Dezembro 2015.
- [2] HUFFMAN, D. A Method for the Construction of Minimum Redundancy Codes, **Proceedings of the I.R.E.**, Setembro 1952. 1098-1101.
- [3] ZIV, J.; LEMPEL, A. A Universal Algorithm for Sequential Data Compression, **IEEE Transactions on Information Theory**, v. IT-23, n. 3, p. 337–343, Maio 1977.
- [4] WYLIE, F. Digital Audio Data Compression, **Electronics and Communication Engineering Journal**, n. 2, p. 5-10, Fevereiro 1995.
- [5] HAYKIN, S. **Communication Systems**. 4th ed. New York: John Wiley & Sons, 2004.
- [6] LATHI, B. P. **Modern Digital and Analog Communication System**. 3rd ed. New York: Oxford University Press, 1998.
- [7] STANLEY, G. **Essentials of Audiology**. 3rd ed. New York: Thieme, 2011.
- [8] FLANAGAN, J. L.; SCHROEDER, M. R.; ATAL, B. S.; CROCHIERE, R. E.; JAYANT, N. S.; TRIBOLET, J. M. Speech Coding, **IEEE Transactions on Communications**, p. 710-737, Maio 1979.
- [9] COUCH, L. W. II. **Digital and Analog Communication Systems**. 8th ed. New Jersey: Pearson Education, 2013.
- [10] NYQUIST, H. Certain Topics in Telegraph Transmission Theory, **Transactions of the American Institute of Electrical Engineers**, vol. 47, p. 617-644, Janeiro 1928.
- [11] SHANNON, C. E. Communication in the Presence of Noise, **Proceedings of the I.R.E.**, vol. 37, p. 10-21, Janeiro 1949.
- [12] Shannon, C. E. A Mathematical Theory of Communication. **The Bell System Technical Journal**, Chicago, v. 27, n. 3, p. 623-656, Outubro 1948. ISSN 10.1002/j.1538-7305.1948.tb01338.x.
- [13] ZIEMER, R. E.; TRANTER, W. H. **Principles of Communication, Systems, Modulation and Noise**. 6th ed. New Jersey: John Wiley & Sons, 2009.
- [14] MALVINO, A. **Eletrônica Analógica 2**. 2^a ed. Porto Alegre: AMGH Editora, v. 1, 2007.

- [15] KIROVSKI, D.; LANDAU, Z. Generalized Lempel-Ziv Compression for Audio, **IEEE 6th Workshop on Multimedia Signal Processing**, Outubro 2004.
- [16] ASHIDA, S.; KAKEMIZU, H.; NAGAHARA, M.; YAMAMOTO, M. Sampled-Data Audio Signal Compression with Huffman Coding, **Proceedings of the SICE Annual Conference**, p. 972–976. 2004.
- [17] ROMÃO, F.; S.; R., Trabalho de Conclusão de Curso, Universidade Federal de Uberlândia. **Análise da Eficiência da Codificação Huffman por Meio da Filtragem de Sinais de Áudio**. Patos de Minas, Dezembro 2016.
- [18] ADIEGO, J.; NAVARRO, G.; FUENTE, P. D. L. Lempel-Ziv Compression of Structured Text, **Data Compression Conference**, Spain, Março 2004.
- [19] INGEMAR, C.; MATTHEW, M.; JEFFREY, B.; JESSICA, F.; TON, K. **Digital Watermarking and Steganography**. San Fransisco: Morgan Kaufmann. 2007.
- [20] ZHU, J.; WANG, R.; YAN, D. The Sign Bits of Huffman Codeword-Based Steganography for AAC Audio, **2010 International Conference on Multimedia Technology**, p. 1–4, Outubro 2010.
- [21] LI, J.; WANG, R.; ZHU, J. A Fragile Audio Aggregation Watermarking Algorithm Based on Huffman Coding, **Electronics, Communications and Control (ICECC) 2011**, Setembro 2011, pp. 626–629.
- [22] SANTOSH, N. A.; NEGI, A. An Approach to Self-Embedding Document Watermarking Based on LZW Algorithm, **IEEE 10 Regions Conference (TENCON) 2007**, Outubro 2007. DOI: 10.1109/TENCON.2007.4428857.
- [23] ADIWIJAYA; FAOZIYAH, P. N.; PERMANA, F. P.; WIRAYUDA, T. A. B.; WISESTY, U. N. Detection and Recovery of Medical Image Watermarking Using Modified LSB and Huffman Compression, **Second International Conference on Informatics and Applications (ICIA)**, Setembro 2013, p.129 - 132. DOI: 10.1109/ICoIA.2013.6650242.
- [24] ARIF, A. S.; MANSOR, S.; KARIM, H. A.; LOGESWARAN, R. Lossless Compression of Flouroscopy Medical Images Using Correlation and the Combination of Run-Lenght and Huffman Coding, **2012 International Conference on Biomedical Enginnering and Sciences (IEEE EMBS)**, p. 759-762, Dezembro 2012.
- [25] MEHFUZ, S.; TIWARI, U.; RATHORE, A.; ARORA, A.; SINGH, D. Huffman Based Lossless Compression Algorithm for Wireless Sensor Networks, **2014 International Conference on Innovative Applications of Computational Intelligence on Power, Energy and Controls with their Impact on Humanity (CIPECH14)**, p. 48-53, Novembro 2014.

[26] COOPER, T. K.; DESOKY, A. H. Huffman Coding Analysis for XOR Filtered Images, **2015 IEEE International Symposium on Signal Processing and Information Technology**, p. 237-241. Dezembro 2015.

[27] OBOLONIN, I. A.; RYGOVSKAYA, N. A. The Analysis of Filtration Influence at Primary Pulsed-Code Conversion on Distortion of Input Signals of Coders with a Compression of Audio Data, **2009 International Conference and Seminar (EDM)**, p. 177-178, Julho 2009.

[28] OPEN SPEECH REPOSITORY. **American English**. Disponível em: <http://www.voiptroubleshooter.com/open_speech/american.html> Acesso em: 05 de setembro de 2017.

[29] GIORDANO, A. A.; LEVESQUE, A. H. **Modeling of Digital Communication Systems Using Simulink**. New Jersey: John Wiley & Sons, 2015.