PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E PROCESSOS INDUSTRIAIS

Joel Giordani Pereira

ANÁLISE COMPARATIVA EM RELAÇÃO À QUALIDADE E DESEMPENHO DE CODECS USANDO COMPRESSÃO POR CELP E WAVELETS

Santa Cruz do Sul, dezembro de 2009.

Joel Giordani Pereira

ANÁLISE COMPARATIVA EM RELAÇÃO À QUALIDADE E DESEMPENHO DE CODECS USANDO COMPRESSÃO POR CELP E WAVELETS

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas e Processos Industriais - Mestrado, Área de Concentração em Controle e Otimização de Processos Industriais, Universidade de Santa Cruz do Sul, como requisito parcial para obtenção do título de Mestre em Sistemas e Processos Industriais.

Orientador: Prof. Dr. Rafael dos Santos
Co-orientador: Prof. Dr. Ruben Edgardo Panta Pazos

Santa Cruz do Sul, dezembro de 2009.

Joel Giordani Pereira

ANÁLISE COMPARATIVA EM RELAÇÃO À QUALIDADE E DESEMPENHO DE CODECS USANDO COMPRESSÃO POR CELP E WAVELETS

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas e Processos Industriais – Mestrado, Área de Concentração em Controle e Otimização de Processos Industriais, Universidade de Santa Cruz do Sul, como requisito parcial para obtenção do título de Mestre em Sistemas e Processos Industriais.

Dr. Rafael dos Santos

Professor Orientador

Dr. Ruben Edgardo Panta Pazos

Professor Co-Orientador

Prof. Dr. Philippe Olivier Alexandre Navaux (UFGRS/RS)

Prof. Dr. Rolf Fredi Molz (UNISC/RS)

AGRADECIMENTOS

Agradeço a minha família, em especial a minha esposa, pela dedicação, apoio e compreensão, que me auxiliaram na conclusão deste trabalho.

Agradeço aos colegas do mestrado e do GPSEM pela amizade, companheirismo e troca de conhecimento que a convivência proporciona.

Agradeço especialmente ao meu orientador, Prof. Dr. Rafael Santos, e ao co-orientador, Prof. Dr. Ruben Panta, pelos ensinamentos, dedicação, empenho e paciência.

Agradeço ao CNPq pela concessão da bolsa DTI.

Agradeço também ao Prof. Dr. Philippe Olivier Alexandre Navaux e ao Prof. Dr. Rolf Fredi Molz, pela participação como membros da banca, e suas contribuições que enriquecem este trabalho.

RESUMO

Neste trabalho, foi realizado o desenvolvimento de dois *codecs* de áudio, usando a teoria de *wavelets,* e uma análise comparativa entre os mesmos e o *codec open-source Speex*, que implementa o algoritmo CELP. Os tipos de *wavelets* usados foram *wavelets Haar* e *wavelets Coiflet* com 30 coeficientes. A aplicação alvo dos *codecs* é em um dispositivo embarcado específico para comunicação *VoIP,* sendo que os mesmos foram compilados para o uso em processadores da arquitetura ARM920T. Os *codecs* foram avaliados através da análise da taxa de compactação do sinal, a qualidade após aplicação do *codec* e a necessidade de processamento para a codificação e decodificação. Para obter a taxa de compactação foi analisada a quantidade de *bits* dos dados de entrada e saída do codificador. Os indicadores de qualidade foram obtidos através do método subjetivo MOS e método objetivo PESQ. O uso do simulador *QEmu* possibilitou analisar e comparar o desempenho dos *codecs* através da extração dos dados do traço da execução das instruções. De acordo com os resultados obtidos, o *Speex* obteve uma melhor aceitação quanto à qualidade do sinal em ambos os métodos de avaliação, seguido do *Speex-Wavelet Coiflet* e, por último, o *Speex-Wavelet Haar.* A taxa de compactação foi maior no *Speex*, que operou a 15 *kbps*, em comparação a 40 *kbps* dos *codecs wavelet*. Quanto à necessidade de processamento, o *codec Speex-Wavelet Haar* executou menos instruções no processador e ocupou menos ciclos de processamento, seguido do *Speex-Wavelet Coiflet* e, por último, o *codec Speex.*

Palavras chave: Codec; ARM; CELP; *Wavelet; S*imulação orientada a traço; MOS; PESQ.

ABSTRACT

This dissertation presents a comparative analysis between the opensource codec Speex (using the CELP algorithm) and two developed audio codecs based on the wavelet's theory. The implemented wavelets are respectively the Haar and the Coiflet wavelet with 30 coefficients. The application/scenario was a VoIP embedded device, thus both codecs have been compiled and ported for compatibility with the ARM920T's architecture. They have been evaluated according to the following criteria: signal's compression rate, quality at the output of the codec and the processing effort required for coding and decoding. To get the compression rate, the amount of bits at the codec's input and at the output were analyzed. The quality indexes were calculated using the MOS and the PESQ methods (subjective and objective respectively). By applying the QEmu simulator it has been possible to compare the codecs' performance (by means of tracing the data execution). According to the results of both evaluation methods, the signal quality of the Speex achieved better results. Speex is followed by Speex-Wavelet Coiflet and then Speex-Wavelet Haar. The compression rate has been higher for the Speex, that operated at 15kbps, comparing to 40kbps for the wavelet codecs. Regarding the processing overhead, the Speex-Wavelet Haar codec has executed less instructions and allocated less cycles of the processor, followed by the Speex-Wavelet Coiflet and then by the Speex.

Keywords: Codec; ARM; CELP; Wavelet; Trace-driven simulation; MOS; PESQ.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

IP	*Internet Protocol*
VoIP	*Voice over IP*
SIP	*Session Initiation Protocol*
IAX	*Inter-Asterisk Exchange*
AMR	*Adaptative Multi-Rate*
CELP	*Code Excited Linear Prediction*
WMA	*Windows Media Áudio*
LSF	*Line Spectral Frequencies*
LSP	*Line Spectral Pairs*
PCM	*Pulse Code Modulation*
LPC	*Linear Predictive Coding*
AMDF	*Average Magnitude Difference Function*
LPAS	*Linear Prediction Analysis-by-Synthesis*
AbS	*Analysis-by-Synthesis*
LTP	*Long-Term Prediction*
STP	*Short-Term Prediction*
WPT	*Wavelet Packet Transform*
MOS	*Mean Opinion Score*
DRT	*Diagnostic Rhyme Test*
DAM	*Diagnostic Acceptability Measure*
SNR	*Signal-to-Noise Ratio*
PSQM	*Perceptual Speech Quality Measure*
ITU	*International Telecommunications Union*
PAMS	*Perceptual Analysis / Measurement System*
PESQ	*Perceptual Evaluation of Speech Quality*
JTAG	*Joint Test Action Group*
MMU	*Memory Management Unit*
PC	*Program Counter*
RLE	*Run-Lenght encoding*
CS-ACELP	*Conjugate-Structure Algebraic-Code-Excited Linear-Prediction*
TDBWE	*Time Domain Band Width Extension*
SPIHT	*Set Partioning In Hierarchical Tree*

SUMÁRIO

1 INTRODUÇÃO

Uma ligação voz sobre IP (*VoIP - Voice Over Internet Protocol*), em funcionalidade, é equivalente a uma ligação usando o sistema de telefonia normal, sendo que a diferença está no modo em que os dados são transmitidos. Usando *VoIP*, o sinal de entrada analógico é digitalizado e os dados gerados pela digitalização do sinal são compactados. Estes dados compactados são transmitidos através de uma rede de computadores e descompactados no destino. Após a descompactação, o sinal é recuperado e reproduzido. Este processo está ilustrado conforme a Figura 1 (DOHERTY; ANDERSON, 2006).

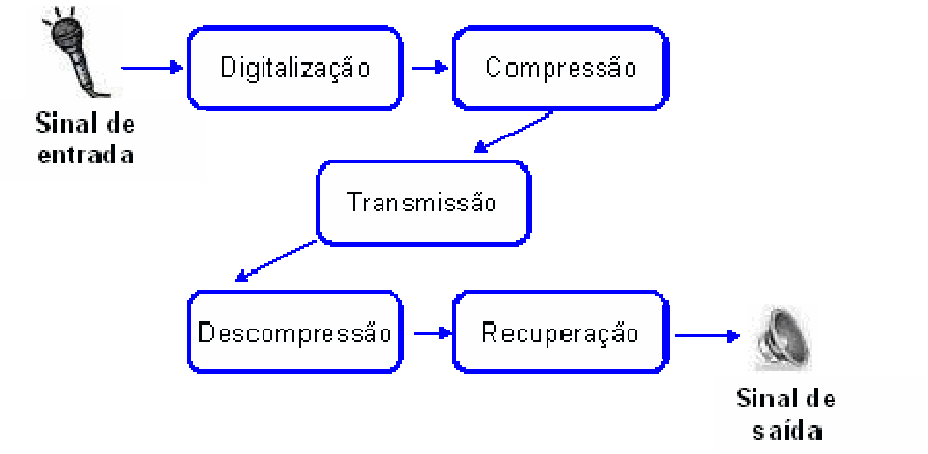


Figura 1 – Diagrama de funcionamento de uma ligação *VoIP*.
Fonte: Elaborado pelo autor

Existem várias implementações livres de *royalties* de protocolos de comunicação usados em aplicações *VOIP*, como o *Session Initiation Protocol* (SIP) e o *Inter-Asterisk EXchange* (IAX). No entanto, tratando-se de *codecs* de áudio, alguns dos melhores e mais usados, como o G.729 e o *Adaptative Multi-Rate* (AMR), necessitam o pagamento de *royalties* para serem usados.

Um *codec* livre de *royalties* é o *Speex* (VALIN, 2006), um *vocoder* híbrido baseado no funcionamento CELP (*Code Excited Linear Prediction*), suportado pelos programas *open-source Linphone, Ekiga* e *Asterisk*.

No contexto do funcionamento de uma aplicação *VoIP*, o *codec* realiza as seguintes funções: a) compactar o sinal de áudio para posterior transmissão; b) descompactar o sinal após a transmissão. Neste processo de compactação e descompactação não se deve perder informações relevantes para a qualidade da ligação (SALOMON, 2004).

O uso de um *codec* que realize a compactação/descompactação do sinal de áudio com alto grau de compactação sem perder qualidade para posterior transmissão possui grande importância, pois possibilita transmitir mais dados sem a necessidade de aumentar a velocidade e a capacidade do canal de comunicação. Isto reduz custos com transmissão de voz, permitindo que sejam realizadas várias ligações simultâneas ou que outros serviços possam ser usados pelo mesmo canal de comunicação.

Entretanto, convém analisar a necessidade de processamento durante a compactação e descompactação, que devem ser rápidos o suficiente para ocorrer em tempo real.

Considerando o uso de aplicações *VoIP* em sistemas embarcados, o processamento necessário para a codificação e decodificação em tempo real, é um fator crucial para a definição dos *codecs* a serem usados no dispositivo embarcado.

Convém lembrar que, dispositivos embarcados são diferentes de um computador de propósito geral, pois estes dispositivos são desenvolvidos com a finalidade de realizar uma tarefa específica. No caso do projeto VoipWIFI, que tem como objetivo desenvolver o protótipo de um dispositivo embarcado com uso específico para *VoIP,* e tecnologia *wireless* para a transmissão de dados na rede, o desenvolvimento da plataforma de *hardware* sofre limitações como, por exemplo, custo dos componentes, tamanho do dispositivo e consumo de energia (BARR, 1999) (VOIPWIFI, 2009).

No dispositivo embarcado estão sendo usados preferencialmente *softwares open-source,* como o *kernel* do sistema operacional *Linux 2.6.x,* e o programa de *softphone* usado para a comunicação *VoIP* será uma versão modificada do *Linphone*. O processador deste dispositivo é o *Cirrus Logic* EP9302, de 200 Mhz de

clock, e pertence à família de processadores ARM920T.

Este trabalho apresenta uma análise comparativa entre um *codec* que usa o algoritmo CELP e dois *codecs* que usam a transformada discreta *wavelet.* Os *codecs* usaram a transformada discreta *wavelet,* pois é uma alternativa à predição linear e análise de *Fourier,* e apresentou bons resultados na compressão e remoção de ruídos do sinal, segundo Nagaswamy (2005) e De Meuleneire (2006).

Para fazer tal análise, foi necessário desenvolver um *codec* baseado na estrutura do *codec Speex*, usando teoria de *wavelets* para compactar e descompactar o sinal, pois, até o momento, não existe *codec* de áudio em funcionamento para uso em aplicações *VoIP* que utilizem a teoria de *wavelets* na sua implementação. Existem apenas descrições de implementações realizadas em produções científicas, que apresentaram resultados referentes ao potencial compactação e a qualidade do sinal.

Este trabalho está organizado conforme descrito a seguir. O capítulo 2 apresenta definições gerais sobre codificadores de áudio, representação digital de sinais de áudio e tipos de codificadores usados para codificação de voz. O capítulo 3 mostra as características do processo da produção da fala e o funcionamento de um modelo matemático usado para reproduzir sinteticamente a fala humana. O capítulo 4 descreve o funcionamento do modelo matemático que usa predição linear para a codificação e decodificação, assim como o funcionamento dos *codecs LPC10e* e *CELP*, que usam este modelo de predição linear. O capítulo 5 possui definições e características da teoria de *wavelets* relevantes para este trabalho. O capítulo 6 apresenta métodos usados na medição da qualidade dos *codecs*. O capítulo 7 mostra formas de analisar e mensurar o desempenho de programas. O capítulo 8 apresenta a metodologia e resultados de trabalhos correlatos, com a finalidade de apontar as diferenças e contribuições presentes neste trabalho. O capítulo 9 mostra a metodologia usada por este trabalho no desenvolvimento de dois *codecs wavelet* e na análise da necessidade de processamento, qualidade e taxa de compactação entre os *codecs wavelet* e o *codec* CELP, usado no codificador open-source *Speex.* No capítulo 10 contém os resultados obtidos das análises da necessidade de processamento, qualidade e taxa de compactação. O capítulo 11 mostra as conclusões e possíveis trabalhos futuros.

2 CODIFICADORES E DECODIFICADORES DE ÁUDIO

Um *codec* (*COder/DECoder*) de áudio tem a função de codificar e decodificar um sinal de áudio. Este trabalho aborda somente a conversão entre formatos binários, buscando a compactação do sinal, o que é feito representando o sinal com o mínimo possível de *bits*, preservando a qualidade do sinal após a descompactação (SALOMON, 2004).

A compactação pode ser feita com ou sem perda de informações. Na compactação sem perda, o sinal obtido depois da codificação e decodificação é idêntico ao sinal original. Na compactação com perda, o sinal obtido não é idêntico ao sinal original após a codificação e decodificação. O seu principal objetivo é tornar o sinal perceptualmente idêntico, eliminando informações consideradas irrelevantes (SALOMON, 2004). Exemplos de *codecs* para a compactação de áudio sem perda são o FLAC (FLAC, 2009) e *Shorten* (SALOMON, 2004) e para compactação de áudio com perda são o MP3 e WMA.

C*odecs* de áudio são utilizados para diversos fins, entretanto, os *codecs* analisados e desenvolvidos neste trabalho são usados em ligações *VoIP*. A próxima seção apresenta mais informações sobre função de um *codec* em uma ligação *VoIP* e o processo de conversão de um sinal analógico para digital.

2.1 Codificação de áudio para aplicações *VOIP*

Na telefonia digital, assim como *VoIP*, a voz não é transmitida de forma analógica. Primeiramente, o sinal analógico captado pelo microfone deve ser convertido para um *stream* digital de dados, que são transmitidos para o receptor, onde o sinal digital é convertido de volta para um sinal analógico (DOHERTY; ANDERSON, 2006).

O processo realizado pelo *codec* de áudio é realizado entre a conversão do sinal e a transmissão do *stream* de dados, onde o sinal é codificado e compactado. Estes mesmos *codecs* realizam a descompactação do sinal após o recebimento da *stream* de dados pelo receptor (SALOMON, 2004).

A conversão de sinais de áudio analógico em digital é realizada nos processos de amostragem e da quantização do sinal.

Durante a amostragem, o sinal é discretizado no domínio do tempo, onde a voltagem do sinal analógico é medida e representada por um número, durante várias vezes em um intervalo de tempo, como mostra Figura 2 (RABINER; SCHAFER, 1978).

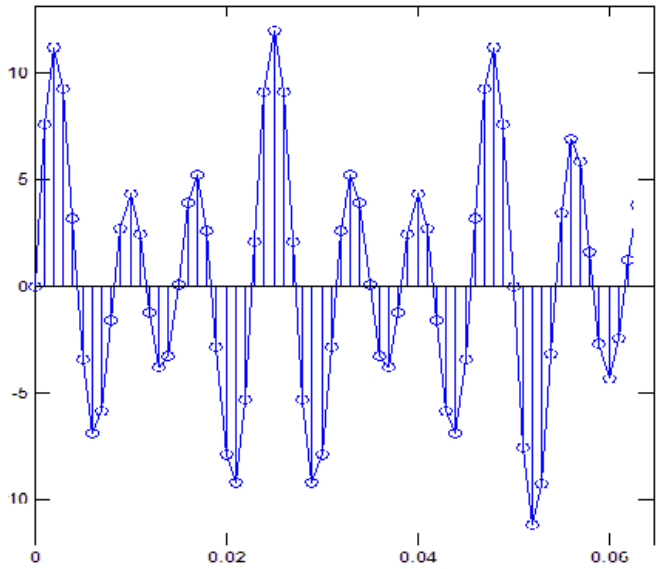


Figura 2 – Amostragem de um sinal de áudio

Fonte: Elaborado pelo autor

A quantidade de medições no intervalo de tempo define a frequência de amostragem, sendo este um fator importante para a futura reconstrução do sinal. Se a quantidade de amostras for muito pequena, o sinal pode não ser reconstruído corretamente, conforme exemplos na Figura 3. Este erro na amostragem é conhecido como *aliasing* (GOLDBERG; RIEK, 2000) (RABINER; SCHAFER, 1978).

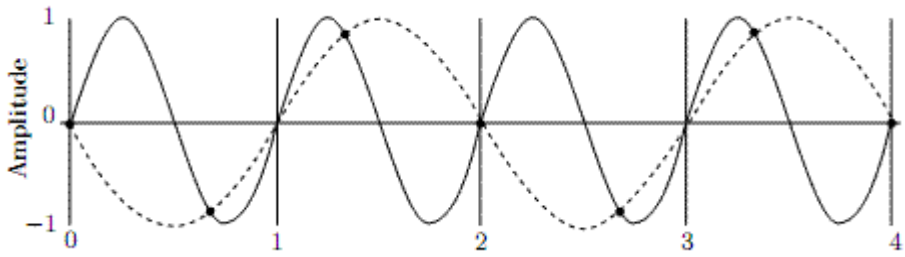


Figura 3 – Erro em frequência de amostragem
Fonte: (SALOMON, 2004, p. 696)

A frequência mínima de amostragem é definida através do Teorema de *Nyquist-Shannon*, onde a frequência de amostragem do sinal dever ser no mínimo duas vezes a frequência do sinal (GOLDBERG; RIEK, 2000) (RABINER; SCHAFER, 1978).

A quantização tem a função de discretizar os valores da amplitude obtidos na amostragem do sinal. O objetivo a ser alcançado com a quantização é representar o valor da amostra com o menor número possível de *bits*, minimizando os efeitos da perda de informação ocorrida pela discretização da amplitude do sinal. A quantização das amostras de um sinal pode ser realizada de forma escalar ou vetorial (SPANIAS, 1994) (GOLDBERG; RIEK, 2000).

Na quantização escalar, a quantidade de valores possíveis para representar a amplitude de uma amostra depende da quantidade de *bits* disponível para a representação do sinal. Se “n” *bits* estão disponíveis para representar o valor de uma amostra de um sinal, então existem 2^n valores possíveis para representar este sinal (SPANIAS, 1994) (GOLDBERG; RIEK, 2000).

Quanto ao intervalo entre os valores possíveis de quantização, é possível classificá-los em: quantização escalar uniforme e quantização escalar não-uniforme. Na quantização escalar uniforme, o intervalo entre os valores da quantização é constante. Já na quantização escalar não-uniforme, o intervalo entre os valores da quantização é geralmente definido for uma função (GOLDBERG; RIEK, 2000).

Um exemplo de quantização não-uniforme é a quantização logarítmica realizada pelo *codec* G.711 (SALOMON, 2004).

Conforme pode ser visto na Figura 4, os sinais de baixa amplitude são quantizados com um intervalo menor e os sinais de maior amplitude, com intervalos maiores. Isso porque as baixas amplitudes contêm mais informação que as altas amplitudes (HUANG; ACERO; HON, 2001).

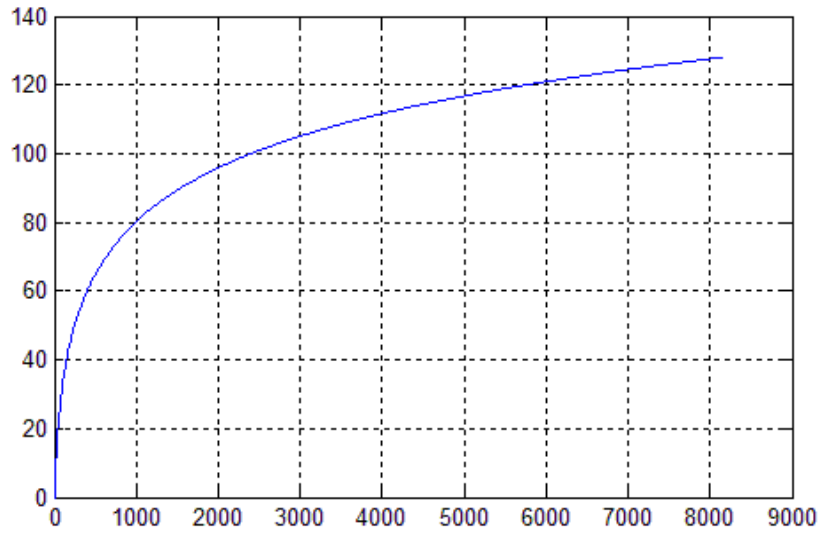


Figura 4 – Quantização logarítmica
Fonte: Elaborado pelo autor

Além da quantização escalar, existe também a quantização vetorial, onde um conjunto de valores é codificado, ao invés dos valores individuais. É utilizado como componente central dos codificadores de voz paramétricos, quantizando e codificando parâmetros do modelo do trato vocal, geralmente representado em forma de *Line Spectral Frequencies* (LSF) (GOLDBERG; RIEK, 2000).

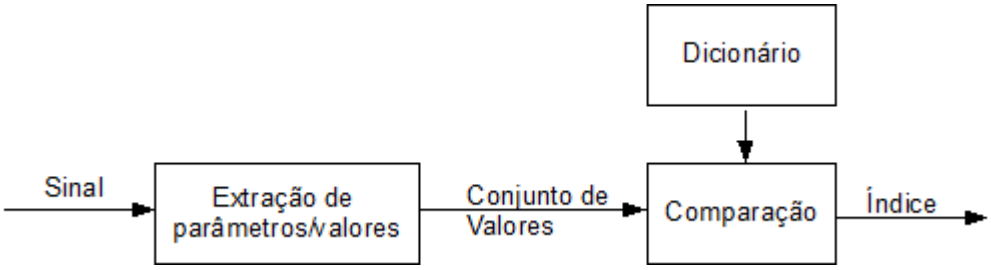


Figura 5 – Diagrama de funcionamento da quantização vetorial
Fonte: Elaborado pelo autor

O diagrama da Figura 5 ilustra o funcionamento de uma quantização vetorial, onde o sinal de entrada é analisado ou transformado para extrair um conjunto de valores. Este conjunto de valores é comparado a um dicionário de valores (*codebook*) e transmitido o índice desse dicionário. No decodificador deve existir a mesma versão do dicionário, que usa o índice transmitido para recuperar os valores quantizados pelo codificador (GOLDBERG; RIEK, 2000).

Para otimizar a codificação de ligações *VoIP*, são usados codificadores de finalidade específica para voz, conforme descrito a seguir.

2.2 Codificadores de voz

Em codificadores de voz, a compressão é otimizada e se limita a codificar e decodificar sinais produzidos pela fala humana (SPANIAS, 1994). Os codificadores de voz são geralmente classificados como:

— Codificadores *waveform*: Os codificadores *waveform* buscam reproduzir o sinal decodificado de forma mais semelhante possível ao sinal original, sem ser aplicado exclusivamente para sinais de voz. Não fazem uma análise específica para sinais de voz, portanto, são mais robustos, e capazes de representar qualquer tipo áudio, necessitando menor capacidade de processamento, com qualidade variando de boa a excelente. No entanto, quando aplicados a codificações de voz possuem uma taxa de compactação menor se comparada a outros tipos de codificadores específicos para voz. Um

exemplo de codificador *waveform* é o PCM (*Pulse Code Modulation*), onde é realizada amostragem pela amplitude do sinal e o tipo de quantização é escalar uniforme. Outro codificador muito usado é o G.711, que usa quantização não-uniforme (SALOMON, 2004) (SPANIAS, 1994) (GOLDBERG; RIEK, 2000).

— Codificadores de transformação: Usam a redundância do sinal para aplicar uma transformação matemática no sinal original, obtendo um novo sinal a ser usado na compactação. Podem ser transformadas contínuas, discretas ou semi-discretas. Como os sinais de áudio usados na codificação são sinais discretos, serão usadas transformadas discretas. Em transformadas discretas, numa função complexa, no caso o sinal original, é representado como um conjunto de funções simples. Esse conjunto de funções é chamado de função base e a função que divide a função base em segmentos menores são representadas por uma escala da função base. Alguns exemplos de transformadas discretas que podem ser usadas na codificação de áudio são: Transformada Discreta de Cosseno (DCT), Transformada Discreta de *Fourier* (DFT), Transformada Discreta *Wavelet* (DWT), entre outras (NAGASWAMY, 2005) (SPANIAS, 1994).

— Codificadores paramétricos: Também chamados de *vocoder* ou *source coders*, modelam matematicamente a fonte dos dados, no caso o trato vocal humano, para fazer a codificação e decodificação. O modelo necessita de parâmetros que são obtidos através da análise dos dados de entrada no codificador. Esses parâmetros são quantificados, transmitidos e usados para reconstruir o sinal original no decodificador. O codificador por LPC (*Linear Predictive Coding*) é um exemplo de codificador paramétrico. (SPANIAS, 1994) (SALOMON, 2004).

— Codificadores híbridos: Codificadores híbridos geralmente possuem características de dois ou mais tipos de codificadores para voz. Um exemplo é o CELP que possui características de codificador *waveform* e paramétrico (SPANIAS, 1994).

Neste capítulo foram apresentados conceitos sobre codificadores de áudio, uso de codificadores de áudio em ligações *VoIP*, representação digital de sinais de áudio e tipos de codificadores usados para codificação de voz. Com base nestes

conceitos, o próximo capítulo inicia com as definições do funcionamento do processo de produção da fala, usado na codificação específica de voz em codificadores do tipo paramétricos.

3 PRODUÇÃO DA FALA

Através da compreensão do processo de produção da fala e suas características, é possível otimizar *codecs* para o uso específico em sinais de fala. Exemplos deste tipo de *codec* são os paramétricos e os híbridos, que representam a produção da fala através modelos matemáticos, para reproduzi-la sinteticamente ou analisar suas características (GOLDBERG; RIEK, 2000).

A fala humana é resultado da passagem do ar originado dos pulmões pelo trato vocal. O trato vocal humano é composto pela faringe, glote, cordas vocais, palato mole, cavidade bucal, língua, dentes e lábios (RABINER; SCHAFER, 1978) (GOLDBERG; RIEK, 2000).

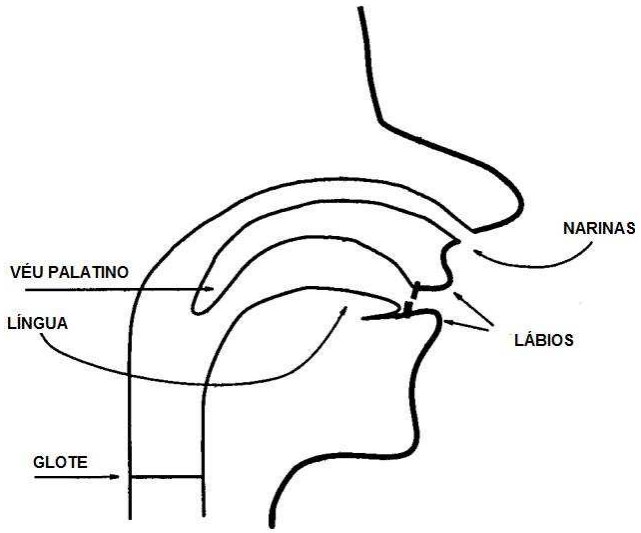


Figura 6 – Trato vocal humano
Fonte: Elaborado pelo autor

3.1 Classificação dos sinais de fala

O tipo de som produzido pela fala humana pode ser classificado em dois grupos principais: vozeados ou sonoros e não vozeados. Quando ocorre a vibração das cordas vocais localizadas na glote, o som produzido é vozeado. O som das vogais é um exemplo desse tipo de som (GOLDBERG; RIEK, 2000).

Sons não vozeados são produzidos pela constrição da passagem do ar pela língua, dentes ou lábios, como é o caso do som das consoantes “s” e “f” (GOLDBERG; RIEK, 2000).

Existem sons que são produzidos através da vibração das cordas vocais e da constrição da passagem do ar, sendo é classificados como mistos. Como exemplo, temos o som produzido pela fala da consoante “z” (GOLDBERG; RIEK, 2000).

Analisando os sinais de fala do tipo vozeado, é possível perceber uma natureza quasi-periódica, enquanto que no som não vozeado, o sinal é semelhante a um ruído. A Figura 7 e a Figura 8 ilustram exemplos de sinais produzidos por tipos de sons sonoros e não sonoros. A Figura 7 é o sinal correspondente ao som da vogal “a”, e a Figura 8 é o sinal correspondente ao som da consoante “s”. Estes sinais foram gerados a partir de gravações da voz do autor, com taxa de amostragem de 44 Khz. O arquivo contendo o áudio foi aberto na ferramenta *open-source Octave* para gerar a representação gráfica do sinal.

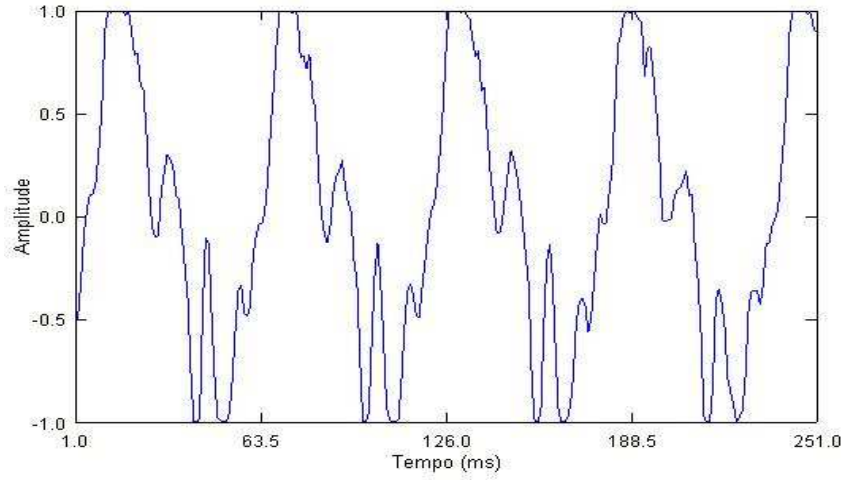


Figura 7 – Exemplo de sinal de voz do tipo sonoro.

Fonte: Elaborado pelo autor

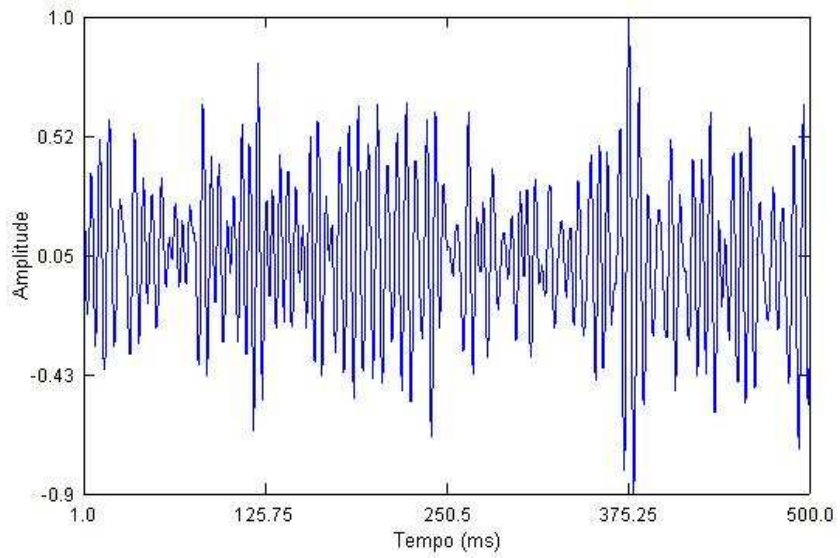


Figura 8 – Exemplo de sinal de voz do tipo não sonoro.

Fonte: Elaborado pelo autor

Uma característica importante usada em codificadores paramétricos é a frequência fundamental e as suas formantes, que será detalhada a seguir.

3.2 Frequência fundamental e formantes

A frequência fundamental (F0), ou frequência de *pitch,* está presente em sons vozeados, sendo definida pela frequência da vibração das cordas vocais. Esta frequência da vibração das cordas vocais depende da característica fisiológica do tamanho da abertura da glote, local onde estão situadas as cordas vocais (GOLDBERG; RIEK, 2000).

Devido a esta característica fisiológica, o valor da frequência fundamental de uma criança atinge até 500 Hz, de um homem adulto atinge frequências variando entre 50 a 250 Hz e a frequência de uma mulher varia de 120 a 300 Hz (GOLDBERG; RIEK, 2000) (SPANIAS, 1994).

Analisando um sinal vozeado, é possível perceber o período de *pitch*, onde o valor de maior amplitude marca o início do período de *pitch*, conforme ilustrado na Figura 9. (GOLDBERG; RIEK, 2000)

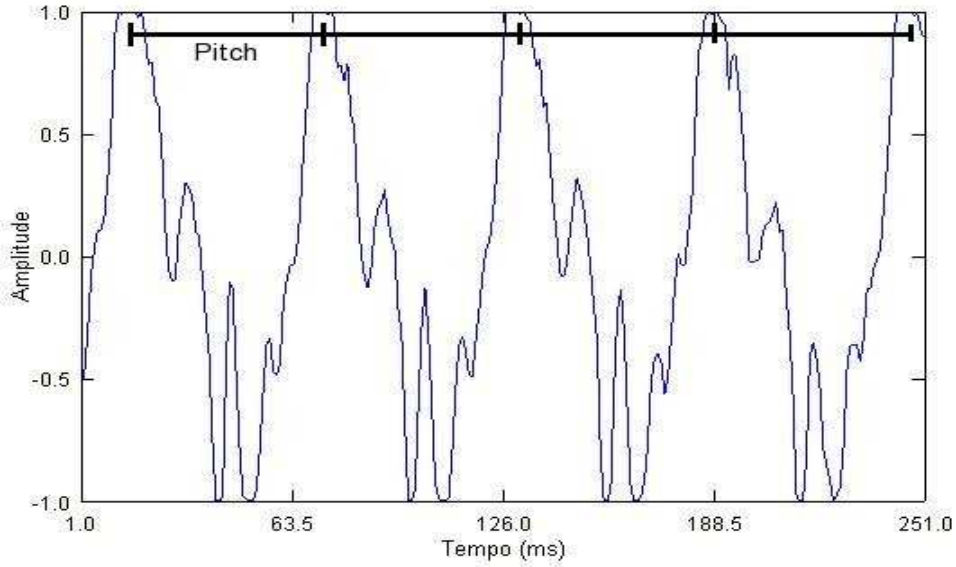


Figura 9 – Período de *pitch*.

Fonte: Elaborado pelo autor

O trato vocal tem suas ressonâncias naturais, e essas ressonâncias mudam quando a forma do trato vocal é modificada. Essas ressonâncias geram picos em um periodograma e a frequência onde ocorrem esses picos são chamadas de frequência do formante, ou formante.

3.3 Modelo *source-filter* genérico

É um modelo usado para representar o processo de produção da fala, que padroniza o trato vocal com um filtro variante no tempo. O objetivo é produzir sinteticamente a fala, de forma que se assemelhe o máximo possível ao original (GOLDBERG; RIEK, 2000) (LEVINSON, 2005).

A fonte de energia para este tipo de filtro é a excitação do sinal. O modo em que a excitação do sinal é obtida varia entre os tipos de modelos *source-filter* (GOLDBERG; RIEK, 2000)(LEVINSON, 2005).

A excitação do sinal é periódica quando o sinal é vozeado. Essa periodicidade resulta dos pulsos criados pela rápida abertura e fechamento das

cordas vocais. Quando o sinal é não-vozeado, o modelo amplamente usado para representá-lo é o ruído branco, porque é o mesmo sinal que resulta quando o ar passa por uma constrição. Para sinais mistos, a excitação também deve ser mista (LEVINSON, 2005).

Um diagrama genérico de um modelo *source-filter* está ilustrado na Figura 10, onde o "*Pitch*" é o período de *pitch*, que produz a excitação vozeada. De acordo com as informações temporalmente variáveis do vozeamento, é definido se será usada uma excitação vozeada ou não-vozeada. A excitação é usada no trato vocal, onde as informações deste modelam o espectro da excitação. As informações do trato vocal podem ser obtidas através de predição linear ou magnitudes de Fourier (GOLDBERG; RIEK, 2000).

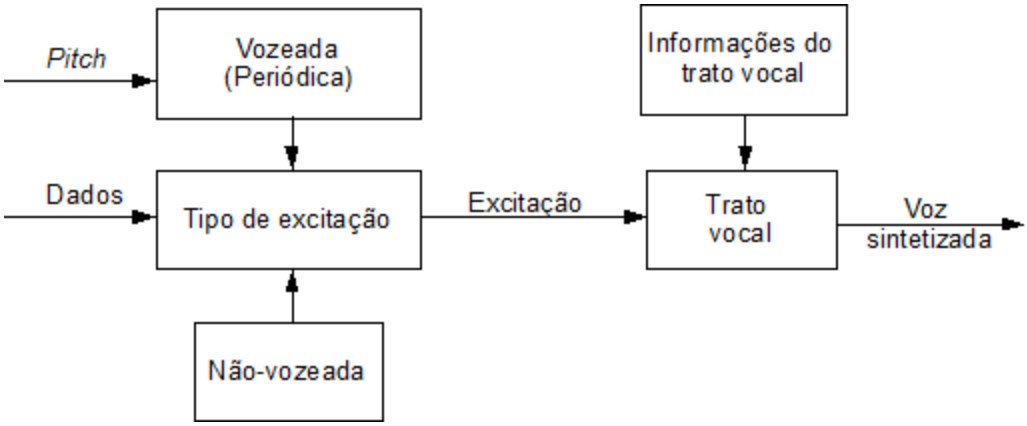


Figura 10 – Diagrama de funcionamento de um modelo *source-filter*
Fonte: Elaborado pelo autor

Este capítulo apresentou o funcionamento do processo de produção da fala, a classificação dos tipos de sinal produzido pela fala humana e as características presentes nestes sinais usadas no modelo *source-filter,* que visa produzir sinteticamente a fala humana. Com base nestas informações, o próximo capítulo inicia apresentando uma implementação do modelo *source-filter* usando predição linear.

4 MODELO DE PREDIÇÃO LINEAR

A modelagem matemática da propagação das ondas sonoras pelo trato vocal, permite um maior entendimento de como o trato vocal modela a frequência da excitação do sinal. Para auxiliar no entendimento da produção da fala, o trato vocal é considerado como um tubo uniforme de tamanho e formato fixo, sem perdas devido a condições térmicas ou variações de pressão (GOLDBERG; RIEK, 2000).

A Figura 11 ilustra o funcionamento de um tubo uniforme de tamanho e formato fixo, onde A_{tubo} é a área do tubo, U_g é o volume da velocidade de ar que passa pela glote e U_m é o volume da velocidade ao ar que passa pela boca (GOLDBERG; RIEK, 2000).

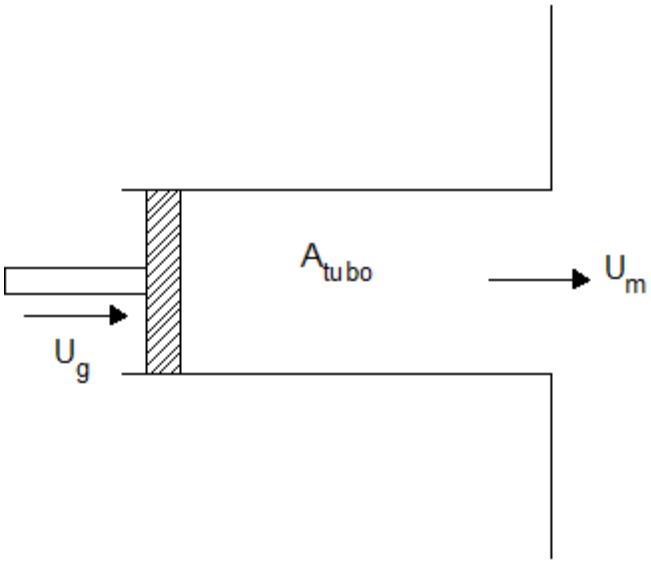


Figura 11 – Diagrama de funcionamento de um tubo uniforme
Fonte: Elaborado pelo autor

Através desse modelo simplificado, é possível calcular as frequências formantes produzidas pelo tubo uniforme. Considerando o tamanho médio do trato vocal humano como 17,5 cm e a velocidade do som como 35.000 cm/s, as frequências formantes geradas estão ilustradas na Figura 12. Neste caso, a primeira frequência formante situa-se em 500 Hz e o restante a cada 1.000 Hz (GOLDBERG; RIEK, 2000).

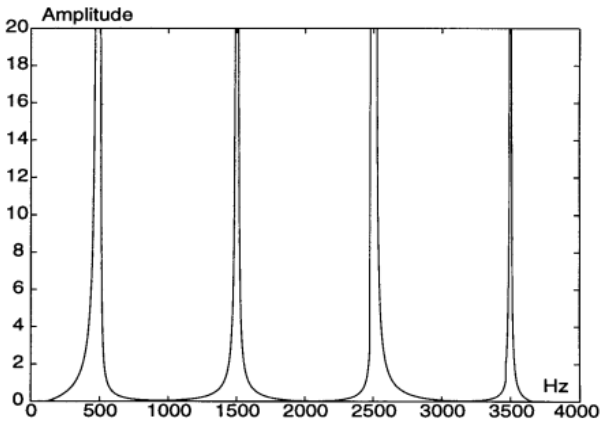


Figura 12 – Frequências formantes geradas por um tubo uniforme

Fonte: (GOLDBERG; RIEK, 2000, p. 54)

Para adaptar o modelo do tubo uniforme para trato vocal humano, a área do tubo deve variar de acordo com o tempo e essas variações criam vários sons diferentes com a mesma fonte de excitação. Então, ao considerar o trato vocal humano como a concatenação de vários tubos uniformes, o modelo será mais próximo do real. O modelo está descrito na Figura 13 (GOLDBERG; RIEK, 2000).

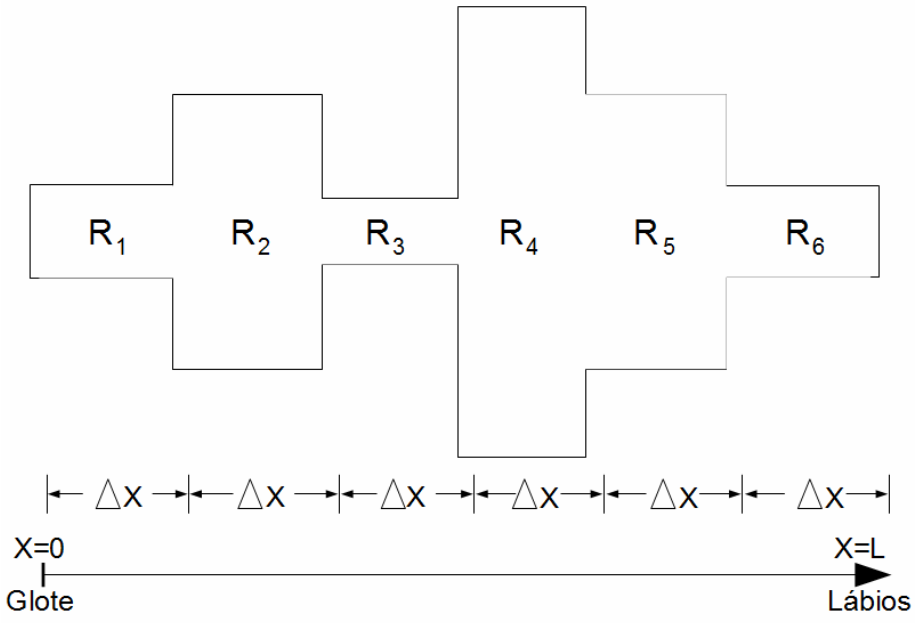


Figura 13 – Modelo de múltiplos tubos uniformes concatenados

Fonte: Elaborado pelo autor

A excitação do modelo pode ocorrer na glote, quando o valor de “X” é igual a zero, ou em uma constrição da passagem de ar. Essa excitação é propagada pelos

tubos uniformes concatenados, onde parte da energia é refletida e parte é propagada (GOLDBERG; RIEK, 2000).

A partir destas características do sinal produzido pela fala humana e sua modelagem matemática de predição linear, a seguir serão abordadas a codificação através do uso de *codecs* de áudio LPC e CELP.

4.1 Predição linear LPC

A predição linear LPC usa a combinação ponderada linear dos valores das amostras anteriores de um sinal para predizer o valor atual da amostra. O filtro de síntese usado está descrito conforme a equação 1 (GOLDBERG; RIEK, 2000).

$$H(z) = \frac{1}{A(z)}, \quad (1)$$

O filtro inverso “A(Z)” é definido pela equação 2.

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k}, \quad (2)$$

Executando a transformada Z inversa da equação 2, resulta em:

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + e(n), \quad (3)$$

O valor de ”s’(n)” é o valor previsto baseado nos valores anteriores do sinal “s(n)” e os valores de “a_k“ são os coeficientes de predição linear, conforme equação 4.

$$s'(n) = \sum_{k=1}^{p} a_k s(n-k)\,, \qquad (4)$$

O erro entre o valor previsto e o valor atual é representado pela equação 5:

$$e(n) = s(n) - s'(n)\,, \qquad (5)$$

onde “e” é o erro entre o valor previsto e o valor atual.

Para estimar o valor dos coeficientes de predição linear, é usada a minimização do erro quadrático entre a amostra atual e a sua aproximação predita, conforme a equação 6:

$$E = \sum_{n} e^2(n)\ , \qquad (6)$$

onde “E*”* é a energia do erro predição (erro quadrático).

Para minimizar a energia do resíduo de predição, os valores dos coeficientes de predição a_k são estimados usando os métodos específicos para esse fim. Os métodos usados em *codecs* são os métodos de autocorrelação ou o método da covariância (GOLDBERG; RIEK, 2000).

4.1.1 Método da autocorrelação

No método da autocorrelação, os valores do segmento de voz fora dos limites especificados são considerados como zero. As equações para os coeficientes a_k são expressas em forma matricial, gerando a seguinte matriz:

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(p-1) \\ r(1) & r(2) & \cdots & r(p-2) \\ \vdots & \vdots & \ddots & \\ r(p-1) & r(p-2) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(p) \end{bmatrix}$$

Os valores de “r(l)” correspondem a autocorrelação de *lag* “l” calculada através da equação 7:

$$r(l) = \sum_{m=0}^{N-1-l} s(m)s(m+l) \ , \qquad (7)$$

onde “N” é a largura do segmento do sinal de voz.

A estrutura da matriz é *Toeplitz* sendo possível resolver este sistema de equações de forma mais eficiente usando o método recursivo de *Levinson-Durbin* (SPANIAS, 1994).

4.1.2 Método da covariância

No método da covariância, a faixa usada para calcular a soma existente na equação 6 é limitada à faixa os índices presentes segmento do sinal de voz, resultando na matriz:

$$\begin{bmatrix} c(1,1) & c(1,2) & \cdots & c(1,p) \\ c(2,1) & c(2,2) & \cdots & c(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ c(p,1) & c(p,2) & \cdots & c(p,p) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} c(1,0) \\ c(2,0) \\ \vdots \\ c(p,0) \end{bmatrix}$$

A covariância é definida pela equação 8.

$$c(i,k) = \sum_{m=0}^{N-1} s(m-i)s(m-k) \, , \qquad (8)$$

A matriz resultante não é uma matriz *Toeplitz*, resultando que o sistema de equações não pode ser resolvido usando o método recursivo de *Levinson-Durbin*. Porém, a matriz resultante é simétrica, sendo possível usar a Decomposição ou Fatoração de *Cholesky* para resolver esse sistema de equações (GOLDBERG; RIEK, 2000).

4.2 Quantização dos coeficientes

Os coeficientes de predição linear são sensíveis a erros de quantização causados pela discretização do seu valor. Devido à necessidade da quantização para a transmissão dos coeficientes, os mesmos são transformados em uma representação equivalente, onde erros de quantização são minimizados (GOLDBERG; RIEK, 2000) (HUANG; ACERO; HON, 2001).

A transformação para LSF (*Line Spectral Frequencies*), ou LSP (*Line Spectral Pairs*), é usada para fazer a codificação de um conjunto de parâmetros de predição linear. Os coeficientes LSF são raízes dos polinômios “P(z)” e “Q(z)”, definidos como:

$$\begin{aligned} P(z) &= A(z) + z^{-(p+1)} A\left(z^{-1}\right) \\ Q(z) &= A(z) - z^{-(p+1)} A\left(z^{-1}\right), \end{aligned} \qquad (9)$$

onde “p” é a ordem da análise da predição linear (GOLDBERG; RIEK, 2000).

As raízes p, ou zeros, de “P(z)” e “Q(z)” ficam em um círculo unitário. São pares de números complexos conjugados, onde uma raiz estará em +1 e uma em -1. Seus ângulos no plano Z representam a frequência, e os pares ou grupos de três,

representam as frequências responsáveis pelos formantes no espectro da predição linear (GOLDBERG; RIEK, 2000).

Os coeficientes de predição linear podem ser calculados a partir dos LSF através da equação 10.

$$A(z) = \frac{1}{2}\left[P(z) + Q(z)\right] \qquad (10)$$

4.3 Codificadores LPC

Codificadores LPC usam o modelo de predição linear com a quantidade de coeficientes entre oito e 14, onde o valor dez é o mais comumente encontrado. Quando a quantidade de coeficientes for maior que 12, o detalhamento do modelo das formantes de sinais vozeados aumenta. Entretanto, requer mais *bits* para a sua quantização (GOLDBERG; RIEK, 2000).

O diagrama de bloco de um codificador está ilustrado na Figura 14. É realizada a amostragem no sinal de voz de entrada e o segmento de dados resultante é dividido em partes (*frames*). Para cada *frame*, é realizada uma análise LP para representar o envelope espectral. O período de *pitch* é estimado por um outro algoritmo. A classificação entre vozeado e não-vozeado pode usar informações da estrutura harmônica, ou a sua ausência, no espectro da estimação do *pitch*, ou ainda pode ser baseada em parâmetros simples, como a energia do *frame* e a quantidade de vezes em que o sinal passa pelo eixo das abscissas (*zero-crossings*). Sinais vozeados tendem a ter maior energia e uma quantidade menor de *zero-crossings* do que um sinal não-vozeado (GOLDBERG; RIEK, 2000).

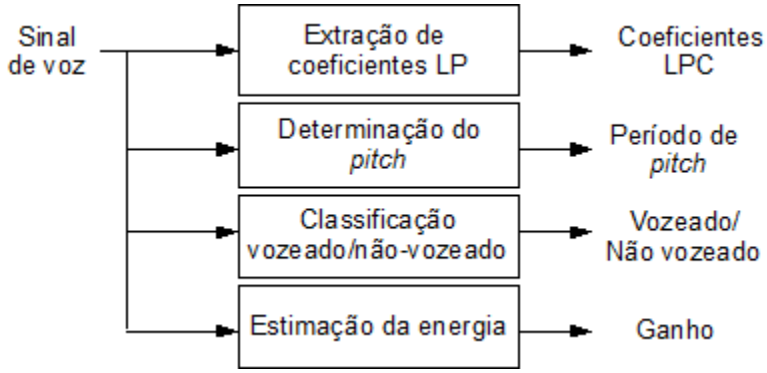


Figura 14 – Diagrama de funcionamento de um codificador LPC
Fonte: Elaborado pelo autor

O decodificador LPC é um modelo *source-filter* de produção da fala, ilustrado na Figura 15. O gerador de pulsos produz um *waveform* periódico no intervalo do período de *pitch* e o gerador de ruído gera um ruído branco. A informação do vozeamento controla a excitação do sinal, indicando que será periódica para sinais vozeados, ou randômica para sinais não vozeados. O sinal de excitação é então modelado na frequência pelo filtro LPC e multiplicado pelo ganho para produzir a energia ou a amplitude correta do sinal de voz sintetizado (GOLDBERG; RIEK, 2000).

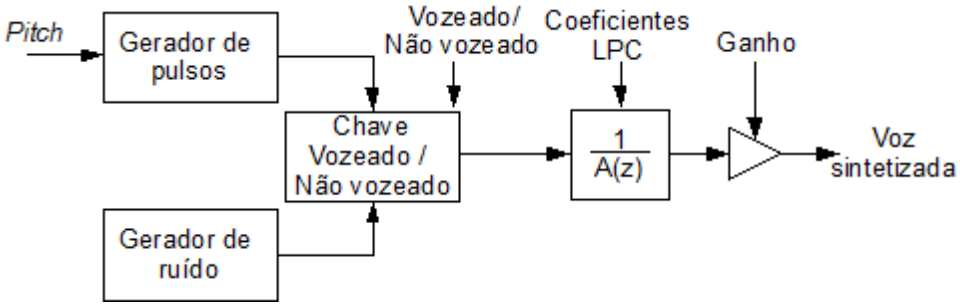


Figura 15 – Diagrama de funcionamento de um decodificador LPC
Fonte: Elaborado pelo autor

Um exemplo de implementação de um codificador LPC é o *Federal Standard* 1015 LPC10e, operando a 2,4 *kbit/s*. O Departamento de Defesa dos Estados Unidos da América adotou um *vocoder* LPC a 2,4 *kbit/s*, que após algumas modificações, tornou-se o *Federal Standard* 1015, LPC10e. A taxa de amostragem é

8 kHz, o tamanho do *frame* é divido em 22,5 ms e compacta cada *frame* em 54 *bits*, alcançando a taxa de 2,4 *kbit/s* (GOLDBERG; RIEK, 2000).

A faixa de frequências de *pitch* aceitáveis é limitada de 50 até 400 Hz. Para estimar o valor do período de *pitch* foi usada a função de *Average Magnitude Difference Function* (AMDF), calculada como:

$$AMDF(m) = \sum_{n=0}^{N-1} \left| s(n) - s(n+m) \right|, \tag{11}$$

para as amostras, “s(n)”, em uma janela de tamanho “N”.

O AMDF é similar ao método de autocorrelação usado na estimação de *pitch*, porém é computacionalmente mais simples, buscando reduzir a quantidade de multiplicações (GOLDBERG; RIEK, 2000).

A análise LPC é de décima ordem e aplica o método de covariância para a estimação dos parâmetros, onde a fatoração de *Cholesky* é usada para resolver o sistema de equações (GOLDBERG; RIEK, 2000).

Todos os dez parâmetros são codificados para segmentos vozeados. Para segmentos não-vozeados, apenas os quatro primeiros parâmetros são usados. Os quatro primeiros coeficientes são quantizados usando cinco *bits*. Os coeficientes restantes são usados apenas em sinais vozeados. Os coeficientes de cinco a oito são quantizados usando quatro *bits*. O nono coeficiente usa três *bits* e o décimo usa dois *bits*. Quando o sinal é não-vozeado, os coeficientes de cinco a dez são usados no controle de erros do canal. S*ão* usados seis *bits* para quantizar o valor de *pitch* (GOLDBERG; RIEK, 2000).

Este *codec* possui uma alta compactação do sinal de voz. Porém, afeta a qualidade do *codec*, tornando-a artificial ou “robótica” durante a sua sintetização (GOLDBERG; RIEK, 2000).

4.4 *Code Excited Linear Prediction* – CELP

A predição linear com excitação por código (CELP - *Code-Excited Linear Prediction*), proposta por *Atal* e *Shroeder* em 1980, é um tipo de codificador híbrido que usa análise por síntese baseada em predição linear (LPAS - *Linear Prediction Analysis-by-Synthesis*) (SPANIAS, 1994) (GOLDBERG; RIEK, 2000).

A análise por síntese (AbS - *Analysis-by-Synthesis*) é usada para a estimação dos parâmetros utilizados na codificação do sinal em malha fechada. Assim, o codificador possui uma versão do decodificador, que é usado para auxiliar na determinação ou ajuste dos parâmetros durante a codificação (SPANIAS, 1994) (GOLDBERG; RIEK, 2000). O seu uso em *codecs* de sistemas embarcados para comunicação *VoIP* aumenta a necessidade de processamento da plataforma de *hardware*, devido a característica principal deste tipo de *codec*, que é a codificação do sinal em malha fechada.

Ao aplicar o filtro inverso de predição linear em um segmento de sinal de voz é gerado um sinal residual. Usar este sinal residual como excitação do filtro torna o sinal de saída idêntico ao de entrada. Se o sinal de excitação usado possuir valores similares ao sinal residual, então o sinal se saída terá alta qualidade. O CELP usa quantização vetorial para transmitir a informação residual do filtro de predição linear inversa (SPANIAS, 1994) (GOLDBERG; RIEK, 2000).

Na sua formulação inicial, as excitações do sinal eram lidas de dicionários, onde cada uma dessas excitações era sintetizada pelo decodificador presente no codificador (GOLDBERG; RIEK, 2000).

O decodificador usado para sintetizar está descrito na Figura 16. Primeiramente, é aplicado o ganho calculado pelo codificador na excitação do sinal. Após, aplica-se o filtro *long-term prediction* (LTP), onde é feita a predição do residual baseada no *pitch*. Neste ponto, é realizada uma busca em malha fechada no dicionário adaptativo pelos parâmetros do filtro LTP que produzem o menor erro (GOLDBERG; RIEK, 2000).

Os coeficientes LPC são usados para aplicar o filtro de *short-term prediction* (STP). O dicionário usado no filtro STP é chamado de dicionário fixo ou estocástico. É preenchido com sequências aleatórias com uma distribuição gaussiana e uma variância unitária (GOLDBERG; RIEK, 2000).

O sinal sintetizado é subtraído do sinal original e é aplicado o filtro perceptual W(z). O filtro perceptual foi desenvolvido baseado no fato de que o aparelho auditivo humano suporta melhor o ruído nas zonas de frequência onde a energia do sinal é maior. Porém quando o ruído ocorre nas zonas de frequência de baixa energia, este ruído é melhor percebido. Este filtro é representado pela equação:

$$W(z) = \frac{A(z)}{A(z\gamma)}, \tag{12}$$

onde A(z) representa o filtro inverso de síntese LPC e γ é um parâmetro com valores entre 0 e 1, usado para controlar o grau de mascaramento auditivo. São tipicamente utilizados valores de γ entre 0,75 e 0,95 (VALIN, 2006) (GOLDBERG; RIEK, 2000).

A sequência com menor energia total é a que provê a melhor excitação do filtro e seu índice é armazenado e transmitido ao decodificador (GOLDBERG; RIEK, 2000).

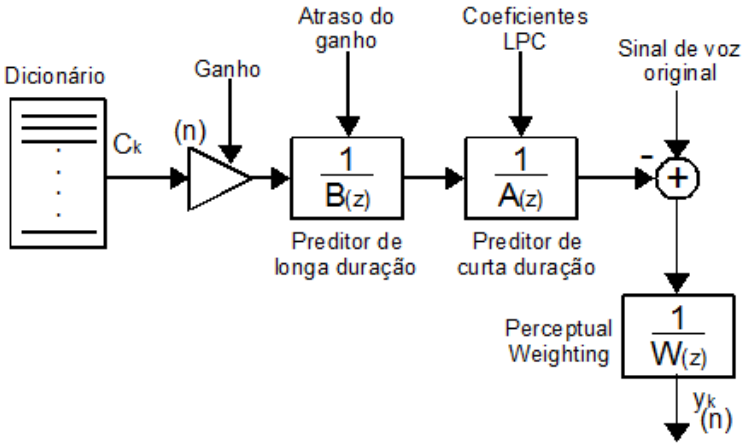


Figura 16 – Diagrama de funcionamento do CELP

Fonte: Elaborado pelo autor

A formulação original do CELP possui o problema de ser necessário muito processamento para sintetizar todas as entradas do dicionário pelo decodificador. Este problema foi atenuado através de um rearranjo dos blocos do codificador e uso de dicionários computacionalmente mais eficientes. Estes ajustes reduziram a necessidade de processamento para níveis praticáveis (GOLDBERG; RIEK, 2000).

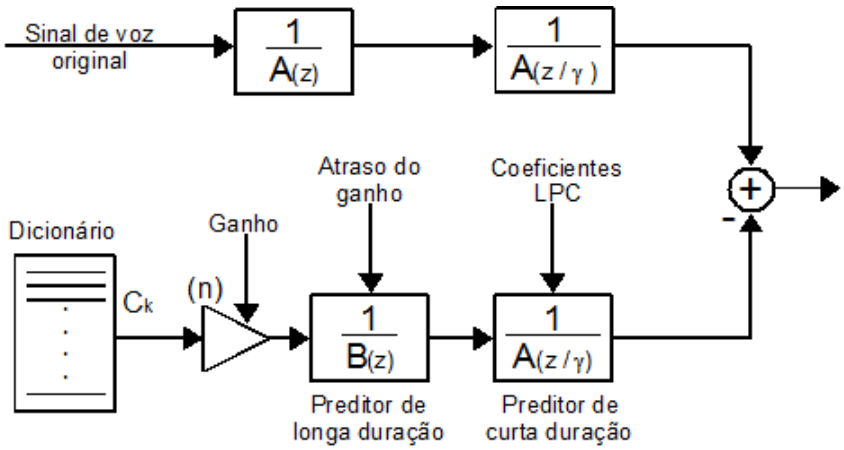


Figura 17 – Diagrama de funcionamento do CELP

Fonte: Elaborado pelo autor

Movendo o filtro perceptual, conforme ilustrado na Figura 17, reduz a necessidade de processamento, pois o sinal de voz de entrada é ponderado perceptualmente apenas uma vez. No esquema original, este processo era feito para todas as entradas do dicionário (GOLDBERG; RIEK, 2000).

Este capítulo encerra a apresentação de definições relativas à *codecs* de voz, modelos de predição linear e suas implementações. O capítulo seguinte, iniciará a apresentação de definições e características referentes à teoria de *wavelets* relevantes para este trabalho.

5 *WAVELETS*

A transformada *wavelet* foi desenvolvida no início da década de 1980 por *Morlet*, sendo usada primeiramente no estudo de dados sísmicos. Desde então, o uso da transformada *wavelet* foi expandido a outras áreas de aplicação, como compressão de imagens, retirada de ruídos, reconhecimento de padrões entre outras aplicações (WALKER, 1999).

Os *wavelets* possuem a ideia fundamental de analisar o sinal de acordo com a escala, decompondo um sinal em diferentes níveis de resolução. *Wavelets* são funções que satisfazem certas exigências matemáticas e são usadas em representação de dados ou funções (GRAPS, 1995) (DAUBECHIES, 1995).

A ideia de superposição de funções para representação de dados ou funções existe desde que Fourier publicou "*Théorie Analytique de la Chaleur*" (1822), e descobriu que poderia usar a superposição de senos e cossenos para a representação de outras funções (GRAPS, 1995).

O diferencial do uso de *wavelets* está no uso de escala para o processamento dos dados, podendo assim ser usadas escalas ou resoluções diferentes para a análise dos dados. O uso de *wavelets* apresenta resultados superiores à análise de Fourier quando o sinal possui picos abruptos (GRAPS, 1995).

No processo de análise de uma função usando *wavelets* usa-se uma função mãe (*mother wavelet*). Desta função-mãe é gerada as funções-filhas, alterando a amplitude, largura e translação desta função-mãe (GRAPS, 1995).

Usando a técnica de Análise de Multiresolução, é possível analisar um sinal em níveis de detalhamento sucessivos. Através da decomposição sucessiva do sinal em aproximação e detalhes, é possível obter vários níveis de detalhamento para análise. Nos sinais de aproximação, estão as informações de mais baixo

detalhamento, que contém informações de baixa frequência do sinal. No sinais de detalhe, estão as informações de alta frequência, onde os detalhes são preservados. Ao juntar as informações de aproximação, o sinal original é reconstruído sem perdas. É possível realizar uma compressão com perda do sinal ao usar um número menor de coeficientes de detalhe na reconstrução do sinal (GRAPS, 1995) (WALKER, 1999) (STOLLNITZ; DEROSE; SALESIN, 1995).

Na Figura 18 são apresentados alguns exemplos de famílias de *wavelets* (tipos de *wavelets*).

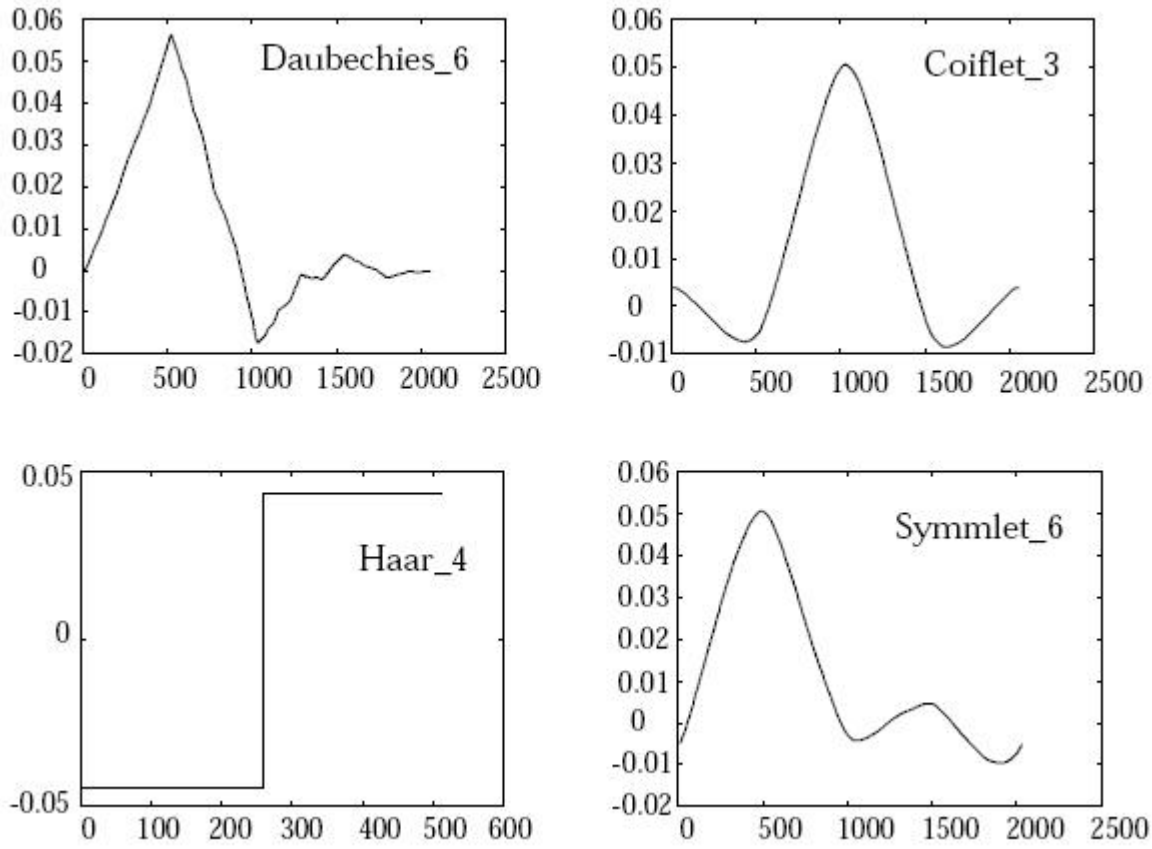


Figura 18 – Exemplos de famílias de wavelets.
Fonte: (GRAPS, 1995, p. 8)

Para melhor compreensão do funcionamento de *wavelets*, será apresentado com mais detalhes o funcionamento da transformada discreta *Haar* para sinais unidimensionais, da família de *wavelets Haar*, considerado o tipo mais simples de *wavelets* (WALKER, 1999).

5.1 Transformada Discreta *Haar*

A transformada discreta *Haar* é uma operação matemática usada em sinais diádicos discretos, geralmente representados por uma sequência de números inteiros ou reais na forma:

$$f = (f_1, f_2, f_3, \cdots, f_n), \tag{13}$$

onde “f(n)” é um número inteiro positivo representando o tamanho de “f”.

Pode ser usada para a compactação de sinais ou a remoção de ruído. A sua aplicação em um sinal é feita em diferentes níveis. No primeiro nível, a transformada discreta *wavelet* decompõe o sinal original em dois sub-sinais, a_1 e d_1, cada um com a metade do tamanho do sinal original. O sub-sinal a_1 é a aproximação e o d_1 o detalhe, conforme ilustrado na Figura 19.

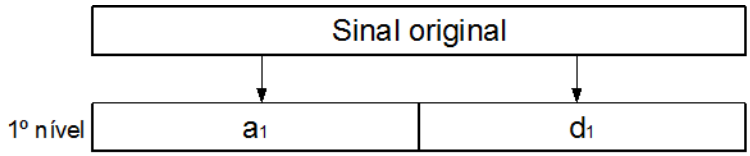


Figura 19 – Transformada discreta *Haar* nível um.
Fonte: Elaborado pelo autor

Para calcular os valores do sub-sinal a_1 é usada a seguinte fórmula:

$$a_m = \frac{f_{2m-1} + f_{2m}}{\sqrt{2}}, \tag{14}$$

onde m = (1,2,3,...,N/2).

Para calcular os valores do sub-sinal d_1 é usada a seguinte fórmula:

$$d_m = \frac{f_{2m-1} - f_{2m}}{\sqrt{2}}, \tag{15}$$

onde m = (1,2,3,...,N/2).

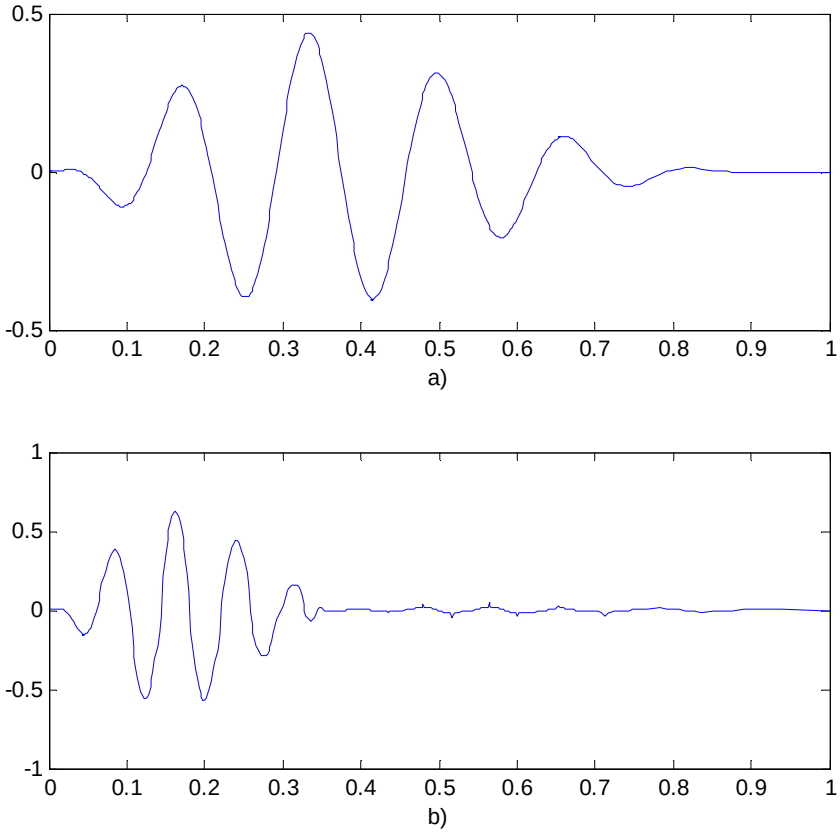


Figura 20 – a) Sinal original b) Sinal após transformada discreta *Haar*.
Fonte: Elaborado pelo autor, baseado em (WALKER, 1999, pág. 13)

A Figura 20 ilustra o sinal antes e depois da aplicação da transformada discreta *Haar*. Após a aplicação da transformada, a informação ficou concentrada no sub-sinal a_1 e os coeficientes no sinal d_1 (WALKER, 1999).

É possível reconstruir um sinal sem perdas se forem usados todos os seus coeficientes d_1. Removendo alguns desses coeficientes d_1 é possível fazer desde a retirada de ruídos do sinal até uma compressão com perda (STOLLNITZ; DEROSE; SALESIN, 1995).

O próximo nível da transformada discreta *Haar* é obtido ao aplicar a

transformada no sinal resultante a_1, resultando nos sub-sinais a_2 e d_2. Os próximos níveis são obtidos usando o sinal da aproximação como o sinal onde será aplicada a transformada. O último nível da transformada é alcançado quando a quantidade de elementos dos sinais resultantes for ímpar. A Figura 21 ilustra o processo de aplicação da transformada em múltiplos níveis.

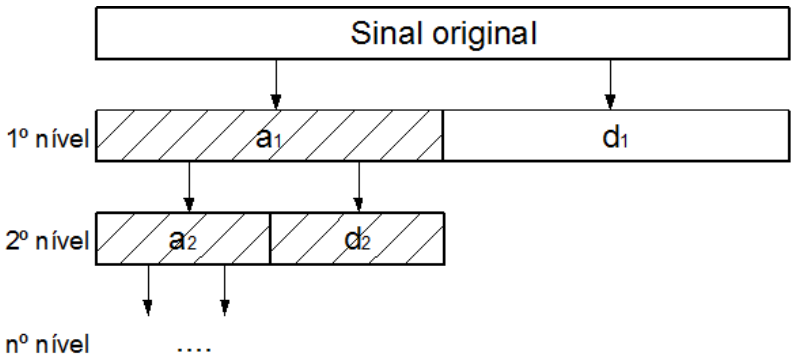


Figura 21 – Transformada discreta *Haar* múltiplos níveis

Fonte: Elaborado pelo autor

5.2 *Wavelets Haar*

Wavelets Haar é o tipo mais simples de *wavelets* e servirá como base na explicação da teoria de *wavelets*. Os *wavelets Haar* são definidos conforme equação 16.

$$\begin{aligned} W_1^1 &= \left(\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0, \cdots, 0\right) \\ W_2^1 &= \left(0, 0, \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0 \cdots, 0\right), \\ &\vdots \\ W_{N/2}^1 &= \left(0, 0, \cdots, 0, \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right) \end{aligned} \tag{16}$$

De acordo a equação acima, o número “um” superior no W, representa o nível do *wavelet* e o número “um” inferior representa posição de translação do *wavelet*. A translação é realizada de duas em duas amostras.

A partir dos *wavelets Haar*, é possível obter o sub-sinal de detalhe, bastando calcular o produto escalar entre o sinal e o *wavelet*.

Para obter o sub-sinal de aproximação, é necessário calcular o produto escalar entre o sinal e os sinais de escala, definidos conforme equação 17.

$$\begin{aligned} V_1^1 &= \left(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0,\cdots,0\right) \\ V_2^1 &= \left(0,0,\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0\cdots,0\right), \\ &\vdots \\ V_{N/2}^1 &= \left(0,0,\cdots,0,\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right) \end{aligned} \tag{17}$$

5.3 *Wavelets Daubechies*

Wavelets Daubechies são semelhantes aos *wavelets Haar*, com diferenças na quantidade variável dos valores dos sinais de escala e coeficientes *wavelet* e na forma de definição dos valores (WALKER, 1999).

Os números de escala (α) e coeficientes *wavelet* (β) usados da transformada *Daubechies* com quatro coeficientes (*Daub4*) estão descritos abaixo:

$$\begin{aligned} &\alpha_1 = \frac{1+\sqrt{3}}{4\sqrt{2}} \quad \alpha_2 = \frac{3+\sqrt{3}}{4\sqrt{2}} \quad \alpha_3 = \frac{3-\sqrt{3}}{4\sqrt{2}} \quad \alpha_4 = \frac{1-\sqrt{3}}{4\sqrt{2}} \\ &\beta_1 = \frac{1-\sqrt{3}}{4\sqrt{2}} \quad \beta_2 = \frac{\sqrt{3}-3}{4\sqrt{2}} \quad \beta_3 = \frac{3+\sqrt{3}}{4\sqrt{2}} \quad \beta_4 = \frac{-1-\sqrt{3}}{4\sqrt{2}}, \end{aligned} \tag{18}$$

Para obter o sub-sinal de aproximação, o método é similar ao que é feito nos *wavelets Haar*. É necessário calcular o produto escalar entre o sinal a transformar e os sinais de escala *Daubechies*, conforme a equação 19.

$$
\begin{aligned}
V_1^1 &= (\alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0 \cdots, 0) \\
V_2^1 &= (0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0 \cdots, 0) \\
V_3^1 &= (0, 0, 0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0 \cdots, 0) \\
&\vdots \\
V_{N/2-1}^1 &= (0, 0, \cdots, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4) \\
V_{N/2}^1 &= (\alpha_3, \alpha_4, 0, 0, \cdots, 0, \alpha_1, \alpha_2)
\end{aligned}
\qquad (19)
$$

O sub-sinal de detalhe é obtido também de forma semelhante ao *wavelet Haar*, mudando somente a quantidade de coeficientes. Basta calcular o produto escalar entre o *wavelet Daubechies*, descritos na equação 20.

$$
\begin{aligned}
W_1^1 &= (\beta_1, \beta_2, \beta_3, \beta_4, 0, 0 \cdots, 0) \\
W_2^1 &= (0, 0, \beta_1, \beta_2, \beta_3, \beta_4, 0, 0 \cdots, 0) \\
W_3^1 &= (0, 0, 0, 0, \beta_1, \beta_2, \beta_3, \beta_4, 0, 0 \cdots, 0) \\
&\vdots \\
W_{N/2-1}^1 &= (0, 0, \cdots, 0, \beta_1, \beta_2, \beta_3, \beta_4) \\
W_{N/2}^1 &= (\beta_3, \beta_4, 0, 0, \cdots, 0, \beta_1, \beta_2)
\end{aligned}
\qquad (20)
$$

5.4 *Wavelets Coiflets*

Outro tipo de *wavelets* são as *CoifI wavelets*, que foram desenvolvidas com o objetivo de manter uma maior semelhança entre os valores da aproximação e o sinal original. Seguindo uma sugestão de *Coifman*, essas *wavelets* foram desenvolvidas inicialmente por Ingrid *Daubechies*, que chamou de *coiflets* essa nova família de *wavelets* (WALKER, 1999).

Uma característica que torna atraente o uso de *Coiflets* é que quando o sinal consiste em amostragem de um sinal analógico, como por exemplo, um sinal de áudio, a transformada *Coif6* produz uma maior semelhança entre os valores dos sub-sinais de aproximação e os valores originais do que qualquer transformada *Daubechies* (WALKER, 1999).

Todos os *coiflets* são definidos de maneira semelhante. Como exemplo o *Coif6*, onde os números de escalamento estão abaixo:

$$\alpha_1 = \frac{1-\sqrt{7}}{16\sqrt{2}}, \qquad \alpha_2 = \frac{5+\sqrt{7}}{16\sqrt{2}}, \qquad \alpha_3 = \frac{14+2\sqrt{7}}{16\sqrt{2}},$$
$$\alpha_4 = \frac{14-2\sqrt{7}}{16\sqrt{2}}, \qquad \alpha_5 = \frac{1-\sqrt{7}}{16\sqrt{2}}, \qquad \alpha_6 = \frac{-3+\sqrt{7}}{16\sqrt{2}}. \tag{21}$$
,

Com esses números, o sinal de escalamento para o primeiro nível é definido de acordo com a equação 22.

$$\begin{aligned} V_1^1 &= (\alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0 \cdots, 0, \alpha_1, \alpha_2) \\ V_2^1 &= (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0 \cdots, 0) \\ V_3^1 &= (0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0 \cdots, 0), \\ &\vdots \\ V_{N/2}^1 &= (\alpha_5, \alpha_6, 0, 0, \cdots, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4) \end{aligned} \tag{22}$$

A partir dos números de escalamento, é possível obter os coeficientes *wavelet*, como pode ser visto abaixo.

$$\beta_1 = \alpha_6, \ \beta_2 = -\alpha_5, \ \beta_3 = \alpha_4, \ \beta_4 = -\alpha_3, \ \beta_5 = \alpha_2, \ \beta_1 = -\alpha_1 \tag{23}$$

Com esses números, podemos determinar o primeiro nível de *wavelets Coif6*, de acordo com a equação 24.

$$\begin{aligned} W_1^1 &= (\beta_3, \beta_4, \beta_5, \beta_6, 0, 0 \cdots, 0, \beta_1, \beta_2) \\ W_2^1 &= (\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, 0, 0 \cdots, 0) \\ W_3^1 &= (0, 0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, 0, 0 \cdots, 0), \\ &\vdots \\ W_{N/2}^1 &= (\beta_5, \beta_6, 0, 0, \cdots, 0, \beta_1, \beta_2, \beta_3, \beta_4) \end{aligned} \tag{24}$$

5.5 Conservação de energia e frequência do sinal

A energia do sinal, calculada pela equação 25 é conservada depois de obtidos os sub-sinais de detalhe e aproximação, onde a energia é dividida em sinal de aproximação e detalhe (WALKER, 1999).

$$\varepsilon_f = f_1^2 + f_2^2 + \cdots + f_N^2, \quad (25)$$

Para calcular a energia dos sub-sinais de aproximação e detalhe, basta aplicar a equação 25 nos valores dos sub-sinais. A compactação da energia ocorre quando grande parte da energia se concentra no sub-sinal de aproximação. Quanto maior o nível do sinal de aproximação, mais a concentração energia tende a diminuir. Essa diminuição da concentração da energia pode ser explicada pelo Principio da Incerteza de *Heisenberg*, onde é impossível acumular uma quantidade fixa de energia em um pequeno intervalo de tempo arbitrário (WALKER, 1999).

Após a aplicação da transformada *wavelet*, a informação permanece inalterada no processo, pois a energia do sinal original é igual à soma da energia do sub-sinal a_1 com o sub-sinal d_1 (WALKER, 1999).

Gerar o sub-sinal de escala no primeiro nível de um sinal, é equivalente a aplicar um filtro passa-baixo no domínio da frequência, permitindo que as frequências mais baixas passem pelo filtro. Aplicando os sinais de *wavelet* no primeiro nível em um sinal, é equivalente a aplicar um filtro passa-alta. Nos níveis seguintes, a aplicação do sinal de escala continua agindo como um filtro passa-baixo, porém, a aplicação sinais de *wavelet* atua como um filtro de banda (*band-pass filter*), onde faixas de frequência com valores diferentes de zero passam pelo filtro (WALKER, 1999).

5.6 *Wavelet packet transform*

A aplicação *Wavelet Packet Transform* (WPT) é similar à aplicação de uma transformada *wavelet*. Considerando apenas o primeiro nível de aplicação, a transformada *wavelet* e a *Wavelet Packet Transform* são idênticas. Contudo, nos níveis seguintes, a aplicação do WPT é diferente, pois os sinais de escala e *wavelets* podem ser aplicados também no sub-sinal de detalhe. O processo é visto no diagrama da Figura 22.

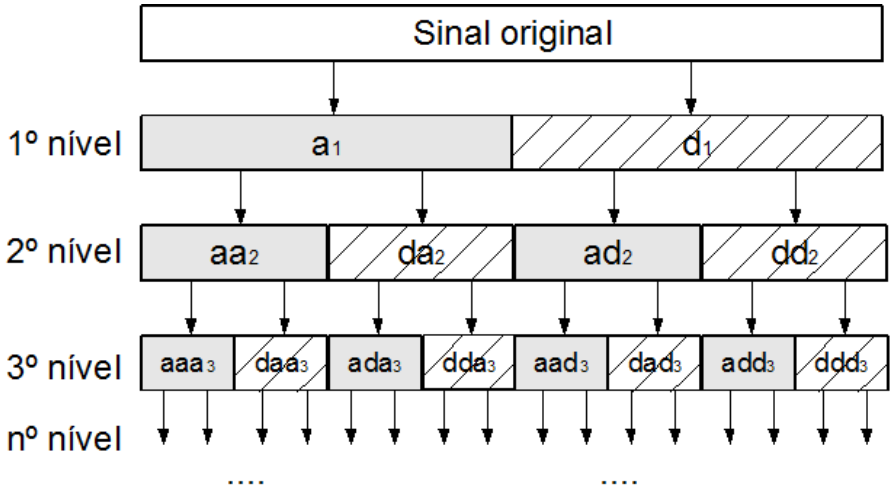


Figura 22 – Digrama do funcionamento de *Wavelets Packet Transform*

Fonte: Elaborado pelo autor

Como visto anteriormente, a aplicação dos sinais de escalamento e *wavelet* tem efeito na frequência do sinal, separando as baixas frequências na aproximação e as altas frequências no detalhe. Usando *Wavelet Packet Transform*, é possível separar as frequências nos sub-sinais de detalhe e obter uma melhor compactação da energia do sinal (AKANSU; MEDLEY, 1999) (WALKER, 1999).

A possibilidade de aplicar ou não os sinais de escalamento e *wavelet* no sub-sinal de detalhe permite múltiplas representações para o sinal transformado. É possível usar funções para definir o nível ideal para *Wavelet Packet Transform*, chamadas de *best-basis.* Para aplicar o algoritmo de *best-basis,* é necessário calcular todos os níveis até um nível limite de forma iterativa. A função usada como critério para escolha pode ser simples, como quantidade de valores dos sub-sinais que são menores que o valor de corte (*threshold*), ou mais complexas como calcular a entropia de *Shannon* (AKANSU; MEDLEY, 1999).

Este capítulo apresentou definições e características da teoria de *Wavelets* que auxiliaram na construção dos *codecs* desenvolvidos neste trabalho. O próximo capítulo mostra os métodos usados na medição da qualidade de *codecs*.

6 MEDIÇÃO DA QUALIDADE DE CODECS

A medição da qualidade de um *codec* de áudio está relacionada com a percepção de sons da fala pelo ouvido humano. Qualidade e entendimento da fala, condições como reconhecimento do individuo, ruído e perda de informações.

Para a medição da qualidade, os métodos podem ser divididos como objetivos e subjetivos.

6.1 Métodos Subjetivos

A avaliação da qualidade do *codec* é realizada através de testes usando pessoas que avaliam a qualidade do sinal decodificado. Apesar de depender das opiniões de ouvintes, gerando a necessidade de realizar os testes com um grande número de indivíduos, os testes subjetivos são mais precisos quanto à indicação da qualidade (GOLDBERG; RIEK, 2000).

Os testes subjetivos são recomendados como avaliação para qualquer tipo de *codec*, em especial para *codecs* paramétricos e híbridos. Em codificadores do tipo *waveform* é de certa forma trivial calcular a qualidade, bastando comparar o sinal original com o decodificado. Porém, em codificadores paramétricos, o resultado o sinal é somente perceptualmente semelhante ao sinal original, sendo necessário que um ouvinte avalie a qualidade do sinal (GOLDBERG; RIEK, 2000).

6.1.1 Mean opinion score

O *Mean Opinion Score* (MOS) é um método comumente usado para avaliar a qualidade de um sinal. Baseia-se na opinião de pessoas que avaliam a qualidade do sinal gerado pelo codificador, em uma escala de um a cinco, onde um é o pior

resultado e cinco é o melhor. Os valores correspondentes à escala estão descritos a seguir:

5 = Excelente
4 = Bom
3 = Regular
2 = Ruim
1 = Inaceitável

Os testes podem ser de conversação ou de escuta. Nos testes de conversação, é possível avaliar outros fatores que contribuem para a qualidade da conversação, como o tempo de atraso, eco, etc. Neste teste, os indivíduos conversam ente si separados por cabines a prova de som, de no mínimo 20 m3 (GOLDBERG; RIEK, 2000) (ITU-T P.800, 1996).

Durante a conversação, é recomendado inserir artificialmente ruídos ambientes e de automóveis. Os indivíduos envolvidos no teste não devem ter conhecimentos em processamento de voz e nem ter participado de um teste similar nos últimos seis meses (GOLDBERG; RIEK, 2000) (ITU-T P.800, 1996).

Nos testes de escuta, os ouvintes escutam as gravações, uma de cada vez, e avaliam a qualidade, de acordo com a mesma escala usada em testes de conversação. São usados geralmente grupos de 15 a 20 pessoas, podendo chegar a mais de 40. Os resultados do teste tendem a ficar mais confiáveis quanto mais pessoas participam. Os ouvintes também não devem ter conhecimentos em processamento de voz e nem ter participado de um teste similar nos últimos seis meses (GOLDBERG; RIEK, 2000) (ITU-T P.800, 1996).

Para realizar a gravação usada nos testes de escuta, deve ser usada uma sala quieta, de dimensões entre 30 m3 e 120 m3, com ruído abaixo de 30 DB. O microfone deve ser posicionado a uma distância de 140 mm a 200 mm. As frases usadas no teste devem ser curtas, com duração variando de dois a três segundos e seu conteúdo deve ser de conhecimento comum, podendo ser retiradas da obras literárias (GOLDBERG; RIEK, 2000) (ITU-T P.800, 1996).

Quanto à escolha de locutores, recomenda-se usar quatro locutores diferentes, sendo dois homens e duas mulheres. Devido à natureza subjetiva do teste, os resultados podem apresentar uma variância significativa por contar com opiniões de pessoas. Mudando de grupo de ouvintes ou usando o mesmo grupo em outro dia pode apresentar um resultado diferente. Os resultados da avaliação com valores entre quatro e cinco são considerados de alta qualidade, sendo que a maioria dos sistemas de codificação possui valores entre três e quatro (GOLDBERG; RIEK, 2000) (ITU-T P.800, 1996).

6.1.2 Diagnostic Rhyme Test

Para testar a inteligibilidade das palavras é usado o *Diagnostic Rhyme Test* (DRT), onde o ouvinte deve diferenciar duas palavras que são diferentes apenas pelo som de uma consoante. O ouvinte escolhe entre as palavras escritas a sua frente à palavra que ele ouviu. Exemplos de palavras podem ser:

cola/bola
venho/tenho
vaca/faca

Para resultados confiáveis é necessário um grande número de palavras, locutores de ambos os sexos e vários ouvintes. O valor do resultado varia de zero a 100%, dependendo da porcentagem de respostas corretas (GOLDBERG; RIEK, 2000).

6.1.3 Comparação em pares

A comparação em pares (*Pair-Wise Comparison*), também conhecida como comparação A/B, avalia sinais de voz processados por dois codificadores diferentes.

Os sinais processadores pelo codificador são apresentados ao ouvinte e o mesmo seleciona qual tem a melhor qualidade. Esse tipo de avaliação é fácil de organizar e razoavelmente confiável (GOLDBERG; RIEK, 2000).

6.1.4 Diagnostic Acceptability Measure

O *Diagnostic Acceptability Measure* (DAM) é um método usado para avaliar a qualidade do sinal de fala codificado, onde ouvintes treinados são testados e re-testados com entradas de controladas e avaliam as mesmas com critérios de aceitabilidade geral e de escalas individuais de qualidade (GOLDBERG; RIEK, 2000).

6.2 Métodos Objetivos

Nos métodos objetivos, são realizados cálculos para avaliar a qualidade do sinal, ou apenas comparando os sinais originais e decodificados, ou modelando matematicamente a percepção da fala humana.

6.2.1 Relação sinal-ruído

A relação sinal-ruído, ou *signal-to-noise ratio* (SNR) em inglês, calcula a energia do sinal original e compara com a energia do ruído. O ruído é erro entre o sinal original e o sinal decodificado. Para calcular o SNR de um sinal em dB, é usada a equação 26 onde “s(n)” representa o sinal original e “s'(n)” é o sinal decodificado (GOLDBERG; RIEK, 2000).

$$SNR = 10\log_{10}\frac{\sum_{n=0}^{N=1} s^2(n)}{\sum_{n=0}^{N=1}\left(s(n)-s'(n)\right)^2} \qquad (26)$$

O SNR privilegia os valores de maior amplitude, porém, os valores de menor amplitude tendem a ser perceptualmente importantes, e pequenos erros, relacionados a valores de baixa amplitude, podem resultar em uma degradação visível do sinal (GOLDBERG; RIEK, 2000).

O SNR segmental calcula o SNR como a média de um número de segmentos pequenos, conforme a equação 27. Como o erro é calculado relativamente à energia do sinal de pequenos segmentos, valores de pequena amplitude contribuem igualmente para média geral (GOLDBERG; RIEK, 2000).

$$SNR = \frac{10}{M}\sum_{m=0}^{M-1}\log_{10}\left(\frac{\sum_{l=0}^{L=1} s^2(mL+l)}{\sum_{l=0}^{L=1}\left(s(mL+l)-s^{'}(mL+l)\right)^2}\right) \qquad (27)$$

6.2.2 Perceptual Speech Quality Measure

O método PSQM (*Perceptual Speech Quality Measure*) e seu sucessor, PSQM+, foram desenvolvidos pela ITU (*International Telecommunications Union*) em 1996, originando a recomendação P.861 (ITU-T P.861, 1998) (RIDDEL, 2007).

O objetivo deste método é reproduzir a percepção acústica humana, levando em consideração o tempo, frequência e a amplitude dos sinais de fala (HARDY, 2003) e estabelecer uma relação dos valores obtidos com o PSQM com os do teste subjetivo MOS. Porém, essa relação entre o MOS e PSQM não foi alcançada (RIDDEL, 2007).

6.2.3 Perceptual Analisys Measurement System

O método *Perceptual Analysis / Measurement System* (PAMS) foi criado pela *British Telecom* para uso interno e não originou recomendações. Seu conceito é semelhante ao PSQM, buscando reproduzir a percepção acústica humana através

de testes de escuta. A pontuação desse método é muito similar ao MOS e pesquisas mostraram que a média da pontuação do PAMS diverge em meio ponto se comparado a pontuação do teste MOS (RIDDEL, 2007).

6.2.4 Perceptual Evaluation of Speech Quality

O método *Perceptual Evaluation of Speech Quality* (PESQ) combina os métodos PSQM e PAMS, gerando a recomendação P.862 da ITU. Assim como no PSQM e PAMS, são usados testes de escuta, porém com uma melhor precisão dos resultados e é possível comparar esses resultados ao do teste subjetivo MOS (RIDDEL, 2007) (ITU-T P.862, 2001).

Este capítulo apresentou métodos usados na medição da qualidade de codecs, possibilitando comparar qualitativamente os *codecs* analisados neste trabalho. Para comparar a necessidade de processamento de *codecs*, no próximo capítulo serão abordados os métodos usados na medição de desempenho de programas.

7 MÉTODOS DE ANÁLISE DE DESEMPENHO

O método usado para a análise do desempenho deve ser escolhido de acordo com os critérios de tempo para fazer a análise, exatidão necessária, custo, estágio de desenvolvimento do objeto de avaliação. Os métodos para fazer a análise do desempenho podem ser divididos em Modelagem analítica (*Analytical modeling*), Simulação (*Simulation*) e Medição (*Measurement*) (JAIN, 1991).

Os resultados obtidos através da modelagem analítica possuem menor precisão, porém são obtidos em menos tempo. O custo também é menor na modelagem analítica, pois se refere apenas ao tempo dispensado pelo profissional na análise. Outro método é a simulação, onde são adicionados detalhes ao modelo a fim de obter resultados mais próximos à realidade. Tanto na simulação quanto na modelagem analítica é possível obter dados de desempenho antes de possuir um protótipo funcional, porque na medição só é possível depois de possuir um protótipo funcional, e a precisão dos resultados irá depender dos parâmetros, se eles representam à situação de uso real. (JAIN, 1991)

Um tipo de simulação usada para analisar o desempenho de programas é a simulação orientada a traço (*trace-driven simulation*), onde informações de *trace* são usadas como entrada do simulador. Um traço é o registro ordenado temporalmente dos eventos ocorridos em um sistema real.

A simulação orientada a traços é comumente usada para avaliar o desempenho de processadores, sejam *pipelines* escalares, superescalares e outros (HINES; BORRIELLO*,* 1997) (KIM; YI; HA, 2005) (*BECKER,* 2007).

Outro tipo de uso para simulação orientada a traço é a análise de programas, como, por exemplo, algoritmos de paginação, análise da memória *cache*, algoritmos de prevenção de *deadlock*, entre outros (JAIN, 1991).

Além de simuladores*,* é possível fazer a análise do desempenho usando técnicas de medição. Ferramentas como *Lauterbach* LA-7742 e *Kiel Arm Realview*

ICE/TRACE 2 que executam o programa no processador real e obtém dados das instruções executadas através do uso de um cabo *JTAG* (*Joint Test Action Group*).

Existem também programas como o *gprof,* usado para a análise de desempenho de outros programas de forma sintética, através da geração de informações de *profiling.* Desta forma, é possível encontrar as partes onde se deve melhorar o desempenho do programa analisado.

Para realizar a comparação entre programas, existe a possibilidade de fazer somatórios da quantidade de operações executadas em linguagem de alto nível. Consiste em contabilizar a quantidade de somas, subtrações, multiplicações e divisões. Assim, é possível comparar o desempenho de forma superficial, pois as operações da linguagem em alto nível são compiladas para o *assembly* do processador e o compilador pode fazer otimizações durante a compilação. Outra desvantagem deste método é que não serão consideradas operações como *load/store,* desvios e interrupções (JAIN, 1991).

Para que a execução do *codec* no simulador seja mais próxima à execução em um processador real, o funcionamento do processador e o tempo de execução das instruções devem ser modelados no simulador.

Para a análise de desempenho, usar os dados de traço como base provê mais informações do que as instruções extraídas de um programa descompilado. Isso porque nas instruções de um programa descompilado contém pseudo-instruções, laços e desvios, não sendo possível mensurar o desempenho somente pela sua análise. Já nas instruções obtidas através de traço, são consideradas apenas as instruções executadas pelo processador, fornecendo maior exatidão sobre instruções executadas pelo programa.

Este capítulo encerra a revisão dos assuntos necessários para a compreensão dos termos presentes na metodologia deste trabalho. No capítulo seguinte, é apresentada a metodologia e os resultados obtidos em dissertações e artigos correlatos, com a finalidade de apontar as diferenças e contribuições presentes neste trabalho.

8 TRABALHOS CORRELATOS

Este capítulo apresenta artigos e dissertações que usaram a teoria de *wavelets* na codificação de áudio com a finalidade de comparar seus objetivos, metodologias e resultados com os presentes neste trabalho.

Na dissertação de mestrado de título *Comparison of CELP Speech Coder with a Wavelet Method*, o autor compara o uso de CELP e *wavelets* para a compactação/descompactação de sinais de áudio. A comparação é realizada considerando a qualidade do sinal após a compactação/descompactação. O *codec* CELP segue a implementação padrão definida pelo *Federal Standard* 1016. O *codec wavelet* foi implementado usando tamanho de *frame* de 1024 amostras, obtendo os coeficientes de *wavelet packet transform* com cinco níveis, gerando 32 sub-bandas. As sub-bandas foram reorganizadas de modo a favorecer a ocorrência de sequências de zeros. O tipo de *wavelet* usado foi *Daubechies* com 12 coeficientes. Cada sub-banda foi classificada como vozeado, não-vozeado, ruído e transiente, e, de acordo com a classificação, foi aplicado um corte. Os coeficientes foram compactados usando *run-lenght encoding* (RLE) (SALOMON, 2004). Ambos os métodos foram implementados usando MATLAB. O método usado para medir a qualidade após o processo foi o MOS (NAGASWAMY, 2005).

Quanto à quantidade de sinais analisados, foram usados sinais de voz de homens e mulheres. Também foram analisados sinais de voz com ruídos para comparar o resultado da retirada de ruído por ambos os métodos de compactação (NAGASWAMY, 2005).

Segundo Nagaswamy (2005), os melhores resultados, para sinais sem ruído, foram obtidos usando CELP e para sinais com ruído, usando *wavelets*.

O artigo *Celp-Wavelet Scalable Wideband Speech Coder* apresenta os resultados e a implementação de um *codec* para *wideband*, com as taxa de transmissão variando de oito a 32 *Kbits/s*. O *codec* G.729 é usado como base desta implementação. O *codec* G.729 usa uma variação do CELP, o *Conjugate-Structure*

Algebraic-Code-Excited Linear-Prediction (CS-ACELP), onde a estrutura dos dicionários é diferente se comparada ao CELP (DE MEULENEIRE, 2006).

O codificador apresentado no artigo opera com sinais de entrada com taxa de amostragem de 16 Khz. O sinal é divido em blocos de 160 amostras ou fatias de tempo de dez ms. Cada bloco é convertido para oito Khz e usado como entrada no *codec* G.729 para obter os parâmetros principais. Após o sinal é convertido para 16 Khz usando o algoritmo *Time Domain Band Width Extension* (TDBWE). A diferença entre o sinal original e o sinal obtido pela aplicação do algoritmo TDBWE é transformada usando *wavelet packet decomposition* (DE MEULENEIRE, 2006).

O tipo de *wavelet* usado foi *Vaidyanathan* com 24 coeficientes. Para a quantização dos coeficientes *wavelet*, foi usado *Set Partioning In Hierarchical Tree* (SPIHT) (SALOMON, 2004), onde os *bits* dos coeficientes são quantizados do mais significativo para o menos significativo. Os coeficientes com maiores amplitudes são quantizados primeiro e os com menores amplitudes por último (DE MEULENEIRE, 2006).

A validação do *codec* foi feita usando a comparação A/B ou comparação em pares. Primeiramente foi comparado um *codec* inteiramente feito usando *wavelets*, chamado de WP, com outro usando a combinação do G.729 com *wavelets*, chamado de CELP+WP. O *codec* CELP+WP foi escolhido como o melhor por 90 % dos ouvintes. O próximo passo foi comparar o *codec* apresentado no artigo com o que utiliza a combinação do G.729, *wavelets* e TDBWE, chamado de CELP+TDBWE+WP, com o CELP+WP. O *codec* CELP+TDBWE+WP foi escolhido por 80 % dos ouvintes (DE MEULENEIRE, 2006).

A próxima validação foi comparar o *codec* CELP+TDBWE+WP com o *codec* G.722 em taxas de transmissão de 40 *kbit/s* e 50 *kbit/s*. Na codificação de voz, o *codec* CELP+TDBWE+WP se mostrou similar ao G.722 operando a 40 *kbit/s*, pois a opinião dos ouvintes ficou divida em 50 %. Já na codificação de música, o G.722 foi escolhido por 70 % dos ouvintes. O *codec* G.722 operando a 56 *kbit/s* obteve aceitação superior ao CELP+TDBWE+WP nos sinais de voz e música (DE MEULENEIRE, 2006).

O artigo *A Computationally Efficient Wavelet Transform Celp Coder* sugere a modificação o funcionamento do *codec* CELP na maneira como o residual é obtido do dicionário adaptativo. Ao invés de usar o residual para fazer uma busca em malha fechada no dicionário fixo, o residual é transformado usando *wavelet* da família *Daubechies*, com quatro níveis de transformação. Ao invés de transmitir o índice do dicionário fixo, são transmitidos apenas os valores de maior amplitude, para reconstrução no decodificador (OOI, VISWANATHAN, 1994).

Essa modificação diminuiu a qualidade de sons não-vozeados. Então, foi adicionado um classificador do *frame* como vozeado ou não-vozeado. Se o sinal for vozeado, é realizada a transformada *wavelet*, senão, usa o dicionário fixo para codificação do sinal (OOI, VISWANATHAN, 1994).

A avaliação da qualidade do sinal foi executada usando comparação A/B e 61% dos ouvintes preferiram o codificador que usa *wavelet*. A avaliação das necessidades de processamento foi obtida através da análise da quantidade de adições e multiplicações, onde o *codec* CELP obteve 8,3 MIPS e o CELP *wavelet* obteve 1,2 MIPS (OOI, VISWANATHAN, 1994).

Para auxiliar na identificação dos *codecs*, os mesmos foram identificados pelo primeiro autor.

Tabela 1 – Comparação entre *codecs* que usaram a transformada discreta *wavelet* para a codificação de áudio.

Codec	*Narrowband*	*Wavelet*	Qualidade	Desempenho
NAGASWAMY	Sim	*Daubechies*	MOS	Não
DE MEULENEIRE	Não	*Vaidyanathan*	A/B	Não
OOI, VISWANATHAN	Sim	*Daubechies*	A/B	MIPS

Fonte: Elaborado pelo autor

Em relação ao *codec NAGASWAMY*, não foi analisada a necessidade de processamento. Os resultados apresentados pela utilização do codec *NAGASWAMY* são importantes porque atingiram níveis similares em relação à qualidade e

compactação se comparado ao CELP. Porém, o *codec NAGASWAMY* possui o funcionamento mais complexo que os *codecs* desenvolvidos neste trabalho, influenciando no seu desempenho.

O *codec* DE MEULENEIRE possui um objetivo diferente dos *codecs* desenvolvidos neste trabalho, sendo de uso para ligações *wideband.* A importância deste artigo vem dos bons resultados quanto à qualidade de compactação. Entretanto, também não apresenta dados referentes à necessidade de desempenho.

O *codec* OOI é um exemplo de *codec* para *narrowband* que otimiza o algoritmo CELP usando a transformada discreta *wavelet.* Este artigo é pertinente por mostrar em seus resultados que a necessidade de processamento diminuiu devido ao uso da transformada discreta *wavelet*. Contudo, esta melhoria pode ser usada apenas em sinais vozeados, sendo que em sinais não vozeados, foi utilizado o algoritmo CELP original. Por isso, não foi possível realizar uma implementação similar neste trabalho, porque o codificador do *Speex* não foi capaz de fazer a codificação do sinal em tempo real no sistema embarcado do projeto VoIPWIFI.

Este trabalho possui um grande diferencial no método de análise de desempenho do *codec*, onde foi usado um método com maior precisão para medir a necessidade de processamento dos *codecs* em uma arquitetura específica. Outra contribuição deste trabalho está no uso de um método subjetivo e um método objetivo para a análise da qualidade dos *codecs.* Os outros autores usaram apenas métodos subjetivos na análise da qualidade. Por fim, outro ponto de destaque está na menor complexidade dos *codecs* implementados neste trabalho, se comparado aos implementados nos trabalhos correlatos.

Mais detalhes da metodologia usada neste trabalho para o desenvolvimento de dois *codecs wavelet,* a análise do desempenho, qualidade e taxa de compactação entre os *codecs wavelet* e o *codec* CELP estão descritas no capítulo a seguir.

9 METODOLOGIA

Neste capítulo inicia a descrição da metodologia usada neste trabalho, onde foram desenvolvidos dois *codecs wavelet* para uso específico em ligações *VoIP* em um dispositivo embarcado, assim como metodologia usada para realizar a análise do desempenho, qualidade e taxa de compactação entre os codecs *wavelet* e o *codec* CELP.

9.1 Estudo do codec *Speex*

Inicialmente, foi realizado um estudo sobre o funcionamento do *codec* de compressão de voz *Speex*. Este *codec* usa o CELP como base para a compactação do sinal e possui vários parâmetros de configuração, que dentre os principais está permitir várias frequências de amostragem, vários tipos de taxa de compactação e quantidade de canais (VALIN, 2006). Neste estudo, foi analisado somente o funcionamento e desempenho do *codec* em modo de banda restrita (*narrowband),* onde a codificação é de um canal, com amostragem de oito KHz e nível de qualidade cinco.

Na análise inicial foram identificadas as funções responsáveis pela implementação do CELP. Após identificar os arquivos com o código fonte responsáveis pela compactação, este código foi analisado buscando relação com o algoritmo teórico descrito na bibliografia do funcionamento do CELP. Apesar do código não possuir uma documentação que facilite o entendimento, foi possível relacionar o funcionamento teórico do CELP com a sua implementação em um *codec*.

A partir dessa análise, o *Speex* foi modificado nas funções onde é feita a compactação e descompactação dos sinais em *narrowband*, substituindo o algoritmo CELP por uma implementação usando somente a teoria de *wavelets* e métodos de compactação de dados para compactar os coeficientes.

O modo de utilização do *Speex* foi mantido para que o *codec* seja testado em *softphones* que possuem suporte ao *Speex*, como *Ekiga* e o *Linphone*. A seguir, maiores detalhes da implementação do *codec wavelet*.

9.2 Desenvolvimento do codec *Wavelet*

Para a compactação do áudio usando a teoria de *wavelets*, inicialmente foi feita uma busca por *codecs* em funcionamento, encontrando somente implementações na ferramenta MATLAB, que usa sinais de áudio de forma estática, através da leitura um arquivo de som previamente gravado.

Como atualmente não existe *codec* de áudio com o código fonte aberto que usa a teoria de *wavelets*, foram desenvolvidos dois *codecs* para serem analisados neste trabalho.

Tendo em vista que a aplicação final dos *codecs* desenvolvidos neste trabalho é realizar ligações *VoIP* em um dispositivo embarcado, a necessidade de processamento para a execução dos *codecs* deve ser minimizada para ser capaz de codificar e decodificar o sinal em tempo real, sem prejudicar a qualidade da ligação.

O uso de CELP, através do *codec Speex*, não conseguiu realizar a codificação do sinal em tempo real no dispositivo embarcado usado do projeto VoIPWIFI, sendo assim, descartada a utilização da teoria de *wavelets* em conjunto com o CELP nos *codecs*, como realizada nos trabalhos de *De Meuleneire* (2006) e *Ooi e Viswanathan* (1994).

Devido a estas limitações da capacidade de processamento, os *codecs* desenvolvidos neste trabalho usaram somente a teoria de *wavelets* para a compactação do sinal.

Tendo como base a descrição de um compressor de áudio descrito em (ZANARTU, 2005), foi elaborado um modelo de *codec* usando a ferramenta

MATLAB. Este *codec* usava *Wavelet Packet Transform* com *Daubechies* de 20 coeficientes. Um modelo psicoacústico era aplicado no sinal para definir a quantidade de *bits* necessários para a quantização, limitada em, no máximo, oito *bits*.

Como o objetivo do *codec* desenvolvido neste trabalho é usar apenas *wavelets*, o modelo psicoacústico não foi aplicado porque o mesmo executa a transformada discreta de *Fourier*. A quantidade de *bits* usada na quantização foi fixada em oito, pois era a quantidade máxima de *bits* da definição original do *codec*. A compactação foi feita usando corte (*thresholding)* dos valores próximos à zero. Com isso, aumentou a quantidade de zeros consecutivos, beneficiando a aplicação da técnica de compactação *Run-Lenght Encoding (RLE).*

Neste modelo, foi informado como entrada um arquivo de áudio e na saída foi gerado um arquivo com os dados compactados, a fim de analisar a taxa de compactação. Apesar do aumento da quantidade de zeros consecutivos, as taxas de compactação não foram maiores que 60 %, com a qualidade do sinal prejudicada em partes das frases onde sons não-vozeados são pronunciados.

A razão da baixa qualidade deveu-se em parte ao modo como foi feita a quantização do sinal, usando quantização uniforme, e, em parte, à definição de um nível de corte mais alto para aumentar a compactação.

O uso de quantização uniforme gerou erros na reconstrução devido ao aumento da amplitude do sinal, que ocorre ao aplicar a transformada *wavelet*. Quando a amplitude máxima aumenta e a quantidade de intervalos permanece inalterada, o intervalo entre os níveis de quantização uniforme também aumenta.

Caso fosse usado um nível de corte mais baixo, os sinais sonoros não-vozeados teriam sido mantidos, com o custo de uma perda da taxa de compressão.

Devido à necessidade de melhora na taxa de compactação e na qualidade do sinal, o passo seguinte foi usar a ferramenta MATLAB para verificar a compactação da energia do sinal usando diversas famílias de *wavelets* e suas várias

quantidades de coeficientes. Nesta etapa foram realizados testes com *wavelets Haar*, *Daubechies* e C*oiflet.* A quantidade de coeficientes dos *wavelets Daubechies* variou de 4 a 20 e quantidade de coeficientes dos *wavelets* C*oiflet* variou de 6 a 30. Deste modo foram testados todos os valores de coeficientes pré-definidos para os tipos de *wavelets Daubechies* e C*oiflets* do *Wavelet Toolbox*, que fazem parte do MATLAB.

Com o auxílio do GUIDE, ferramenta usada na criação de interfaces gráficas para o MATLAB, uma interface gráfica foi desenvolvida para dividir um sinal de áudio carregado de um arquivo em pequenos pedaços e aplicar diferentes tipos de transformada discreta *wavelet*, variando a quantidade de coeficientes. A interface gráfica da aplicação está ilustrada na Figura 23.

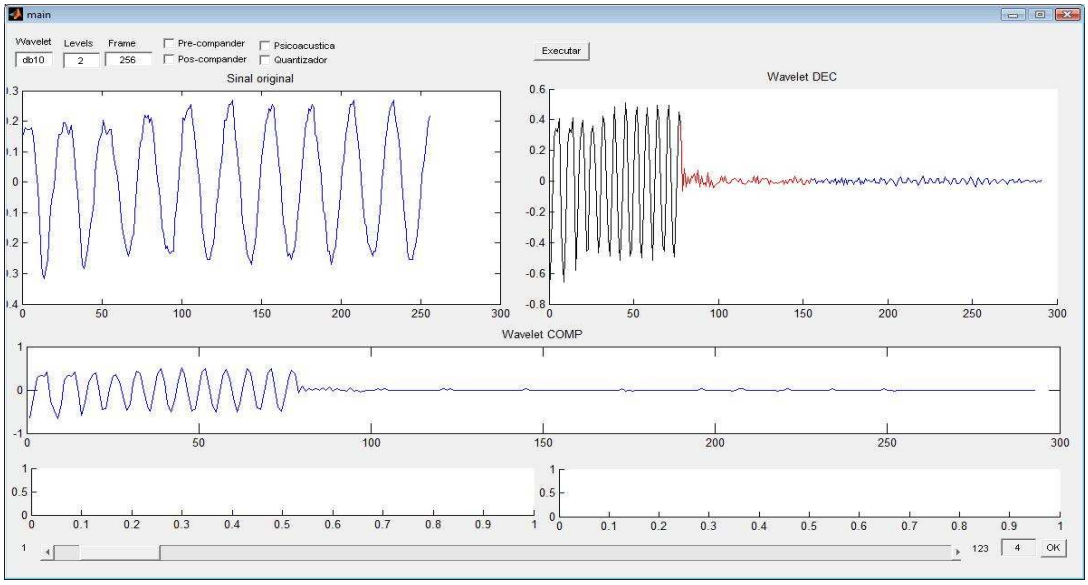


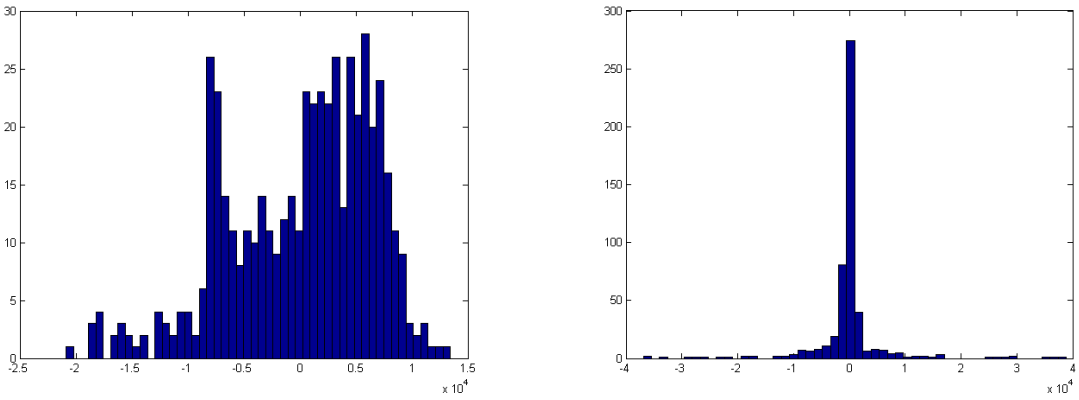
Figura 23 – Tela da aplicação de análise de sinais de áudio usando transformada discreta *wavelet*.

Fonte: Elaborado pelo autor

Através do uso dessa aplicação foi possível definir o tipo de *wavelet* e a quantidade de coeficientes que apresentam o maior acúmulo de energia dos sub-sinais de aproximação. Percebeu-se que as *wavelets* das famílias *Daubechies* e *Coiflet* com quantidade de coeficientes superiores a 10 apresentavam os melhores resultados, obtendo sinais com menor variação nos sub-sinais de detalhe.

O próximo passo foi desenvolver as transformadas discretas *Haar*, *Daubechies* e *Coiflet*. As *wavelets* da família *Daubechies* foram implementadas em 4, 6 e 20 coeficientes e da família *Coiflet* com 6 e 30 coeficientes. Os algoritmos de transformação e a sua transformada inversa foram escritos na linguagem Java e testados comparando os resultados com o MATLAB e da biblioteca de métodos para uso científico *JSci* (JSCI, 2008).

Após concluir a etapa da escrita das transformadas discretas *wavelet*, iniciou-se o desenvolvimento do *codec* usando *Wavelet Packet Transform*. O resultado da sua aplicação fez que os valores do sinal concentrem-se em valores próximos a zero, conforme representado no histograma de um sinal de exemplo, presente na Figura 24.



a) Sinal Original b) Sinal Transformado

Figura 24 – Histograma de sinal de voz original (a) e o histograma do sinal resultante (b) da aplicação da *Wavelet Packet Transform*.

Fonte: Elaborado pelo autor

Porém, verificou-se que, após a aplicação da transformada os valores de amplitude mínima e máxima do sinal tendem a aumentar, necessitando de mais *bits* para a representação de seus valores. Para evitar transmitir o valor do sinal transformado, foi usado um dicionário com valores dispostos em uma matriz de oito linhas (três *bits*) e 16 colunas (quatro *bits*). Os valores de cada linha eram comparados com o sinal a ser transmitido. Após a comparação, o índice da linha com a menor diferença era transmitido e no lugar dos valores era transmitido o índice das colunas usando quatro *bits* para cada valor, conforme ilustrado no

diagrama da Figura 25. Esta abordagem é similar a uma quantização vetorial (GOLDBERG; RIEK, 2000), com a diferença que além do índice da linha, era transmitido também o índice das colunas.

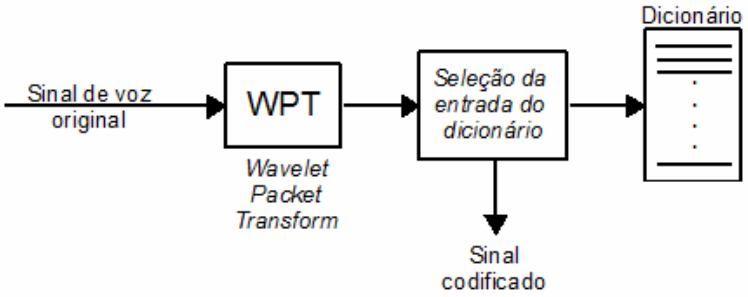


Figura 25 – Diagrama do *codec wavelet* com dicionário 8x16 posições
Fonte: Elaborado pelo autor

Nesta versão, ocorreram problemas com os sinais de maior amplitude, que não foram representados corretamente nos intervalos onde os valores eram menores. Quando os intervalos entre os valores eram maiores, perdia-se precisão nos valores mais próximos à zero. Em ambos os casos, nas partes onde os sinais possuem maior amplitude, a qualidade do sinal reconstruído era prejudicada.

Outra forma de compactação do sinal obtido pela aplicação da transformada discreta *wavelet* foi realizar uma variação da quantização uniforme dos sinais. O funcionamento desta forma de compactação está ilustrado conforme a Figura 26, onde o intervalo da quantização foi usado como parâmetro inicial e as amostras do sinal são representadas pelo nível da quantização uniforme. Após as amostras do sinal resultante foram compactadas aplicando o algoritmo de codificação de *Huffman*. (SALOMON, 2004).

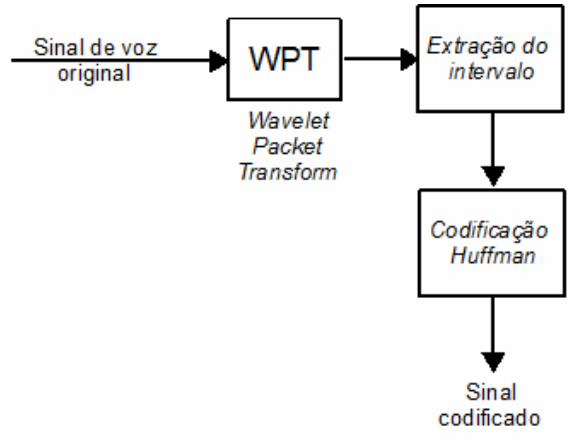


Figura 26 – Diagrama do *codec wavelet* com quantização uniforme e codificação *Huffman.*

Fonte: Elaborado pelo autor

Os resultados também não foram satisfatórios, pois, além do aumento da complexidade e quantidade de operações realizadas do *codec*, a taxa de compactação obtida foi abaixo dos 70 % e, em algumas partes dos sinais, ocorreu inflação do sinal.

Para simplificar o processo de quantização dos dados do sinal transformado, foram usados dois vetores fixos de 32 posições. Um vetor é usado nos sinais de aproximação e o outro, nos sinais de detalhe. Ambos estão representados graficamente conforme a Figura 27. O intervalo entre os valores da matriz não é uniforme, beneficiando sinais de baixa amplitude. Os sinais com maior amplitude são representados com um erro perceptualmente aceitável.

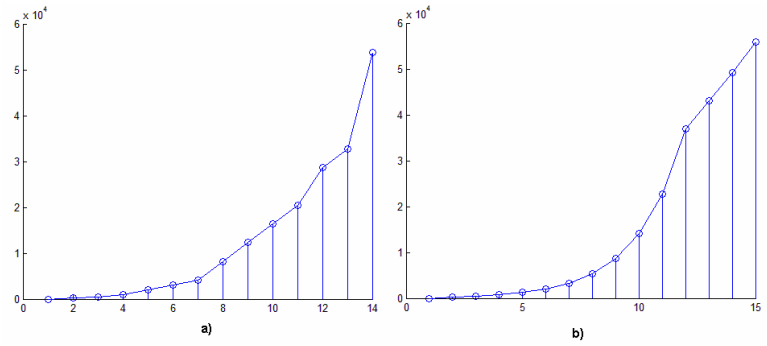


Figura 27 – a) Coeficientes usados na quantização dos sinais de aproximação. b) Coeficientes usados na quantização dos sinais de detalhe.

Fonte: Elaborado pelo autor

A aplicação da transformada foi modificada, buscando obter maior compactação de energia na aproximação e menor amplitude nos valores do sub-sinal de detalhe.

Para os testes comparativos de desempenho e qualidade, foram desenvolvidas duas versões do modelo de *codec*, com a intenção de comparar os ganhos em qualidade e desempenho de cada versão do *codec*. A primeira versão, chamada de *Speex-Wavelet Coiflet30,* usa *wavelets Coiflet* com 30 coeficientes porque a sua transformada obteve o melhor acúmulo de energia no sub-sinal de aproximação nos testes realizados anteriormente. Contudo, a aplicação desta transformada requer a execução de 30 operações de multiplicação de ponto flutuante, o que torna esta operação computacionalmente custosa.

A segunda versão do *codec*, chamada de *Speex-Wavelet Haar,* usa a transformada *Haar* porque sua aplicação é uma operação de menor necessidade de processamento. No entanto, o uso da mesma apresentou acúmulo menor de energia se comparado à transformada usando *wavelets Coiflet* com 30 coeficientes.

O *codec Speex-Wavelet Coiflet30* inicialmente transforma o sinal original nos sub-sinais a_1 e d_1. Após é aplicada a *Wavelet Packet Transform* somente no sub-sinal a_1 até o quarto nível de aplicação da transformada, resultando em 16 elementos por sub-sinal. O funcionamento da aplicação da transformada no *codec Speex-Wavelet Coiflet30* está ilustrado na Figura 28.

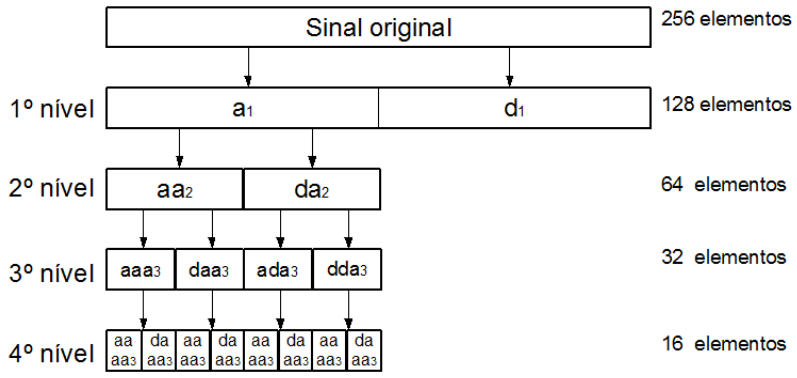


Figura 28 – Aplicação da transformada *wavelet* no *codec Speex-Wavelet Coiflet30*.

Fonte: Elaborado pelo autor

O *codec Speex-Wavelet Haar* tem seu funcionamento um pouco diferente. Como é possível aplicar a transformada *Haar* para obter o nível desejado de forma não iterativa, é aplicado a *Wavelet Packet Transform* no sub-sinal a_1 até o nível máximo. No sub-sinal d_1 a transformação é aplicada até o nível dois. O funcionamento do *codec Speex-Wavelet Haar* está ilustrado na Figura 29.

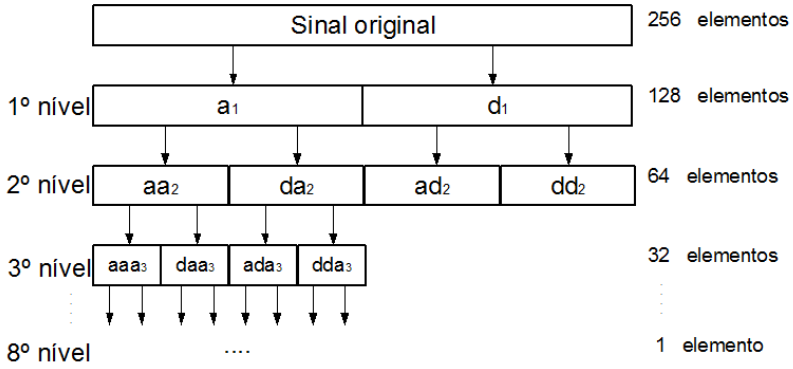


Figura 29 – Aplicação da transformada *wavelet* no *codec Speex-Wavelet Haar*.

Fonte: Elaborado pelo autor

Na aplicação de *wavelets Haar* no *codec Speex-Wavelet Haar*, foi reduzida a execução de operações com ponto flutuante, multiplicações e divisões.

No primeiro nível da transformada, os valores dos sinais são apenas somados e não divididos pela raiz quadrada de dois. No segundo nível da transformada, os valores dos sinais previamente somados são divididos por dois. Essa divisão por dois pode ser realizada de maneira rápida deslocando um *bit* para a esquerda.

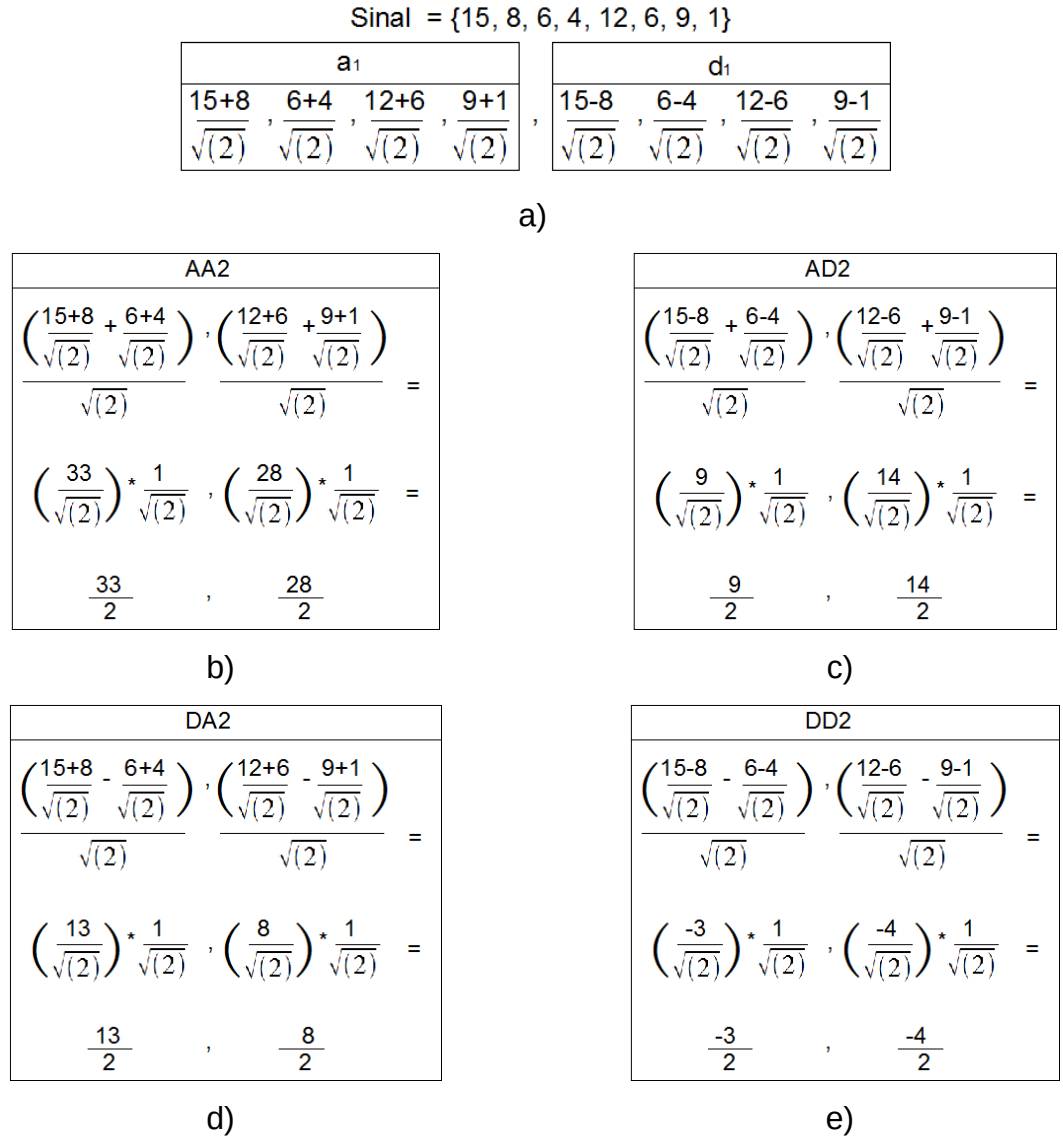


Figura 30 – Funcionamento da modificação da transformada discreta *Haar*.
Fonte: Elaborado pelo autor

Esta operação é equivalente ao modo de aplicação normal da transformada discreta *Haar*, como pode ser visto na Figura 30. O sinal original é transformado em a_1 e d_1 (a). Os valores dos sinais do segundo nível de aplicação da transformada, composto pela aproximação de a_1 (b), detalhe de a_1 (c), aproximação de d_1 (d) e detalhe de d_1 (e), são obtidos somando ou subtraindo os seus elementos e dividindo-os por dois.

Os valores dos sinais de aproximação do nível oito foram obtidos de forma similar. Para obter os valores referentes à aplicação da transformada no nível oito, foram realizadas somente as somas e subtrações, de acordo com o funcionamento

da transformada discreta *Haar.* Somente no oitavo nível de aplicação da transformada o resultado das somas e subtrações foi divido por 16.

As vantagens de executar a transformada discreta *Haar* desta forma são a diminuição de operações usando valores de ponto flutuante e a diminuição da ocorrência de erros por arredondamentos sucessivos.

9.3 Análise da qualidade

A qualidade do *codec* foi obtida através do método objetivo PESQ e do método subjetivo MOS.

Para a realização do teste MOS, foram usadas quatro frases de duração entre dois e três segundos, retiradas de livros, revistas e jornais. Quanto aos locutores, foram usados quatro locutores no total, sendo duas mulheres adultas e dois homens, um deles adulto e o outro um jovem de 12 anos. Na seleção dos locutores, buscou-se diversificar quanto à idade dos participantes, o que possui influência no trato vocal, em consequência das características fisiológicas.

Os testes foram aplicados em 15 ouvintes usando sinais sem ruído, onde foram avaliados os *codecs Speex, Speex*-Wavelet *Coiflet30* e *Speex-Wavelet Haar*. Todos os *codecs* usaram sinais de entrada com taxa de amostragem de oito Khz e quantização de 16 *bits.*

Para a execução do PESQ foi usado o *software* fornecido pela ITU-T, usado para fins de teste do modelo psicoacústico. A execução consiste em fornecer como parâmetro o sinal original e o sinal após a aplicação do *codec* de áudio. Os dados obtidos pelo MOS foram usados para verificar a validade dos resultados obtidos pelo PESQ.

Os resultados dos testes realizados estão descritos a seguir, no capítulo 10.

9.4 Análise do desempenho

A análise e comparação do desempenho entre os *codecs* foram obtidas usando os dados de *trace* de execução dos *codecs*.

Para a extração de dados de *trace* e obtenção da quantidade de instruções, ferramentas como *Lauterbach* LA-7742 e *Kiel Arm Realview ICE/TRACE 2* não puderam ser usadas pela necessidade de um *hardware* para a extração das informações. O processador onde foi executado o *codec* não possui *Embedded Trace MacroCell*, que impossibilita usar essas ferramentas.

Visando uma maior exatidão nos resultados, foram usados simuladores de microprocessadores para analisar e comparar o desempenho de *codecs* de áudio.

O processador a ser simulado é o *Cirrus Logic* EP9302, da família de processadores ARM920T. Esses processadores executam suas instruções em um *pipeline* de cinco estágios. Os estágios são:

- Busca das instruções na memória
- Decodificação das instruções e leitura de registradores
- Execução das instruções
- Acesso à memória
- Escrita nos registradores

O processador EP9302 possui também um coprocessador aritmético para realizar operações de ponto flutuante, MMU (*Memory Management Unit*) e memória *cache* de instruções e de dados, ambas com tamanho de 16 KB (ARM920T, 2001).

A quantidade de ciclos necessária para a execução das instruções é variável e depende do tipo de instrução e dos dados usados nas instruções (ARM, 2005).

A lista dos tipos de instrução e do tempo em ciclos de execução está descrita na Tabela 2.

Tabela 2 – Quantidade de ciclos necessária para execução de instruções

Tipo de instrução	Ciclos	Observações
Data Op[1]	1	Padrão
	2	Operações com *shift* em registradores
STR	1	
LDR	1	Padrão
	2	Não carregando o PC (*Program Counter*) e as próximas instruções usam o dado carregado
	3	Se o dado carregado for um *byte*, *halfword* ou *unaligned word* e for usado nas próximas instruções
	5	Se o PC for o registrador padrão
LDM	2	Carregando um registrador, contanto que não seja o PC.
	n	Carregando ”n” registradores, não sendo o PC.
	n+4	Carregando ”n” registradores.
STM	2	Gravando um registrador
	n	Gravando ”n” registradores
SWP	2	
	3	*Byte* carregado é usado na próxima instrução.
B,BL,BLX	3	
SWI	3	
LDC,STC	b+n	Onde ”b” é o tempo de espera para o uso do coprocessador e ”n” é a quantidade de registradores.
MCR	b+1	
MRC	b+1	
MRC	b+2	Quando a próxima usa o dado transferido.
MUL, MLA	2+m	O valor de ”m” depende da instrução e do conteúdo dos operandos. Visto com mais detalhes a seguir.

[1] As instruções referentes à Data OP são: AND, EOR, SUB, RSB, ADD, ADC, SBC, RSC, TST, TEQ, CMP, CMN, ORR, MOV, BIC e MVN (ARM, 2005) (KNAGGS; WELSH, 2004).

SMULL,UMULL, SMLAL,UMLAL	3+m	

Fonte: (ARM, 2005)

O valor de “m” nas operações de multiplicação depende da instrução e do conteúdo dos operandos. Se a instrução for MUL, MLA, SMULL e SMLAL, o valor de ”m” será:

1: Se os *bits* [31:8] do operando multiplicador forem todos zero ou um.
2: Se os *bits* [31:16] do operando multiplicador forem todos zero ou um.
3: Se os *bits* [31:24] do operando multiplicador forem todos zero ou um.
4: Se nenhuma das condições acima for atendida.

Nas operações de UMULL, UMLAL, o valor de “m” é:

1: Se os *bits* [31:8] do operando multiplicador forem zero.
2: Se os *bits* [31:16] do operando multiplicador forem zero.
3: Se os *bits* [31:24] do operando multiplicador forem zero.
4: Se nenhuma das condições acima for atendida.

Existem vários simuladores de microprocessadores que podem ser usados, entre eles o *Simics (SIMICS, 2009)*, *SimpleScalar (*BURGER; AUSTIN*, 1997)*, *VirtualBox (VIRTUALBOX, 2009)*, *SoftGun (SOFTGUN, 2009)*, *Armulator (ARMULATOR, 2009)*, *Bochs (BOCHS, 2009)*, *QEmu (QEMU, 2009)*, entre outros. Porém, somente o *Simics* e o *SimpleScalar* fazem o *trace* das instruções e somente os simuladores *Simics*, *SoftGun*, *Armulator e QEmu* suportam o processador ARM.

O *Simics (SIMICS, 2009)* possui suporte a arquitetura ARM e gera informações de *trace*, porém é uma ferramenta de licença comercial, e por isto, buscou-se uma alternativa *open-source* antes de considerar a compra deste software.

O simulador *SimpleScalar (*BURGER; AUSTIN*, 1997)* é *open-source*, possui informações detalhadas da simulação, como informações da *cache* de instruções e

de dados, acesso a memória, TLB, entre outras, bem como a capacidade de gerar arquivos de *trace*. Porém, não tem suporte ao processador ARM e é uma tarefa de alta complexidade implementar suporte para essa arquitetura de processadores.

Dentre os simuladores citados anteriormente, somente o *QEmu (QEMU, 2009)*, em uma versão usada no emulador do *hardware* do projeto *open-source Android* (ANDROID, 2009) possui suporte para a execução de instruções do processador ARM e geração de arquivos de *trace*. Na geração dos dados de *trace*, é realizada uma estimativa da quantidade de ciclos necessária para a execução das instruções baseada nas definições do processador ARM920T. Por isto, o *QEMU* foi usado para gerar um arquivo *trace* para a análise e comparação do desempenho dos *codecs.*

O emulador do *Android*, baseado no *QEMU*, foi usado como base na modificação do *QEMU* na versão 0.10.5. Não foi possível utilizar diretamente o emulador porque o sistema operacional do *Android* executa os programas usando a biblioteca *Bionic*, uma versão do *Glibc* mantida pelo projeto *Android* com licença *BSD*. Apesar de a biblioteca *Bionic* ser baseada no *Glibc*, não é compatível com o mesmo. Os *codecs Speex, Speex-Wavelet Haar e Speex-Wavelet Coiflet30* usam a *Glibc* e, portanto, são incompatíveis.

Inicialmente, foi copiado a parte do emulador *Android* que executa o *trace* das instruções para a versão 0.10.5 do *QEmu.* Esta tarefa foi realizada tendo o cuidado de copiar somente das modificações necessárias, pois, além do suporte a geração de *trace*, o projeto *Android* modificou o *QEmu* retirando arquivos não utilizados pela plataforma e adicionado outras funcionalidades como, por exemplo, o suporte a *skins* e a plataforma de *hardware* do *Android.*

Através do uso de ferramentas gráficas de comparação de arquivos, visualização da evolução das versões do emulador na interface *web* do *Git* na página do projeto *Android* e depuração do *emulador Android*, foi possível modificar o *QEmu* para gerar arquivo de *trace*.

O *QEmu* pode ser dividido em dois modos de operação. O modo *system,*

onde o sistema operacional inteiro é executado no simulador e o modo *user,* onde apenas as instruções do programa são traduzidas do processador simulado para o processador real. O projeto *Android* modificou o *QEmu* apenas no modo de operação *system*, porém para obter o processamento apenas de um programa, tornou-se necessário programar a geração de *trace* para o modo *user* do *QEmu*.

Após modificar o *QEmu* para geração do *trace* em modo *user*, os *codecs Speex, Speex-Wavelet Haar e Speex-Wavelet Coiflet3* foram compilados usando um *cross-compiler* para gerar os programas com *assembly* ARM. No teste realizado para obter a quantidade de processamento dos *codecs*, os mesmos leram as informações de um arquivo de áudio de três segundos para realizar a compactação e descompactação.

Ao executar o *Speex-Wavelet Coiflet30* no processador emulado ARM, o valor do *SNR* mostrou-se elevado, o que não acontecia durante a execução do mesmo em um processador da família x86. Comparando as execuções do mesmo *codec* nas duas famílias de processadores, ARM e x86, observou-se que, devido aos coeficientes *wavelet* e escala do *Coiflet* com 30 coeficientes, ocorria um erro de arredondamento sucessivo devido à multiplicação entre números de ponto flutuante e o sinal a ser transformado.

Após análise do arquivo *trace* gerado pela execução do *codec*, descobriu-se que parte do problema ocorreu devido ao compilador, que não usava instruções do coprocessador aritmético para calcular operações de ponto flutuante. Mesmo após a troca do compilador e usando instruções do coprocessador aritmético, ainda ocorria o erro devido ao arredondamento sucessivo. O motivo da persistência do erro foi devido aos dados de ponto flutuante correspondente aos coeficientes, que estavam armazenados em variáveis com o tamanho de 16 *bits*, levando o compilador a usar instruções especificas para variáveis de 16 *bits*. Para corrigir o problema, as variáveis foram modificadas para usar precisão de 32 *bits*.

Os dados usados na análise comparativa de desempenho dos *codecs* foram extraídos dos arquivos de *trace* gerados pelo *QEmu*, obtendo a quantidade de ciclos estimada para cada tipo de instrução. Esses dados estão descritos a seguir, no capítulo 10.

10 RESULTADOS OBTIDOS

Este capítulo contém os resultados obtidos da análise da taxa de compactação, resultados dos métodos de medição da qualidade MOS e PESQ, e o resumo da extração dos dados do traço para a análise de desempenho de cada *codec*. A partir destes dados, é possível realizar uma análise comparativa entre os *codecs*.

Quanto à compactação do sinal, o *codec Speex* operou a 15 *kbps*, onde cada parte do sinal de 160 valores foi compactada em 300 *bits*. O *Speex* foi configurado em modo *narrowband,* sem usar *VBR* e com qualidade cinco.

Ambos os *codecs Speex-Wavelet Haar* e *Coiflet30* operam a uma taxa de transmissão de 40 *kbps* devido à quantização de cinco *bits* por amostra. Cada parte do sinal é composta por 256 amostras, então para cada parte o tamanho máximo é de 1280 *bits*.

Os resultados dos testes de qualidade do sinal apontam o *Speex* como o melhor, seguido do *Speex-Wavelet Coiflet30* e *Speex-Wavelet Haar*, conforme exibido no gráfico da Figura 31. Neste gráfico, a coluna escura é o resultado do teste MOS e a clara é o resultado do PESQ. No primeiro grupo estão os resultados do *Speex-Wavelet Coiflet30,* seguido do Speex e por último o *Speex-Wavelet Haar.*

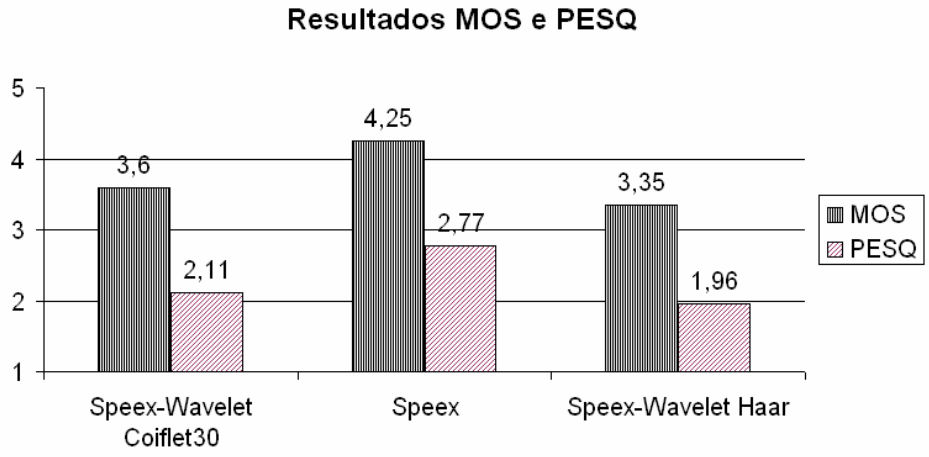


Figura 31 – Gráfico comparativo entre *codecs* nas escalas *MOS* e *PESQ*.
Fonte: Elaborado pelo autor.

O *Speex* pontuou em 4,25 na escala MOS e em 2,77 em PESQ. O *Speex-Wavelet Coiflet30* pontuou em 3,6 na escala MOS e em 2,11 em PESQ. O *Speex-Wavelet Haar* pontuou em 3,37 na escala MOS e em 1,96 em PESQ.

O resultado da extração de dados do traço de instruções está disposto na Tabela 3. Nesta tabela, a quantidade de ciclos necessária para a execução de cada *codec* foi agrupada de acordo com o tipo de operação da instrução.

Tabela 3 – Quantidade de ciclos por tipo de instrução e codec.

Tipo de operação	Codificador			Decodificador		
	Speex	*Speex-Wavelet Coiflet30*	*Speex-Wavelet Haar*	*Speex*	*Speex-Wavelet Coiflet30*	*Speex-Wavelet Haar*
Data OP	71.722.136	59.327.057	14.635.414	7.926.843	58.652.375	19.704.380
Desvios	66.511.341	17.690.763	6.898.782	6.867.411	13.466.433	6.159.372
Multiplicações	351.369	260.354	257.399	37.702	129.870	192.292
Coprocessador	118.810.425	15.130.098	2.087.713	10.253.232	15.135.735	2.859.893
Load múltiplo	861.142	1.122.571	581.108	539.409	1.494.328	1.104.079
Load	11.567.587	39.815.464	10.820.980	716.623	33.018.004	9.538.156
Store múltiplo	431.630	914.521	332.049	365.555	1.101.628	713.938
Store	3.023.884	5.137.617	2.586.512	642.911	4.951.397	1.849.108
Interrupções	105	129	129	126	135	135
Total ciclos	273.279.619	139.398.574	38.200.086	27.349.812	127.949.905	42.121.353

Fonte: Elaborado pelo autor

Considerando que o codificador e decodificador podem ser executados simultaneamente pelo *softphone* durante uma ligação *VoIP*, a quantidade de ciclos necessária para a execução do codificador e decodificador foi somada para realizar comparações entre os *codecs*. A quantidade de ciclos por *codec* está ilustrada na Figura 32.

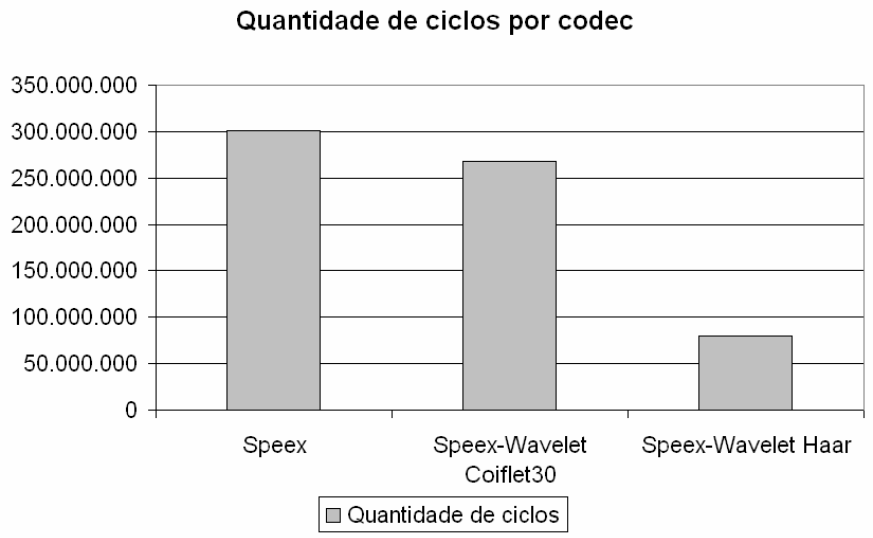


Figura 32 – Quantidade de ciclos por *codec*.

Fonte: Elaborado pelo autor.

Quanto à necessidade de processamento, o uso da transformada discreta *wavelet* obteve resultados positivos. Devido a sua simplicidade, o *codec Speex-Wavelet Haar* executou 73 % menos instruções que o *Speex,* enquanto o *codec Speex-Wavelet Coiflet30* executou 11 % menos instruções que o *Speex.*

11 CONCLUSÕES

O uso da teoria de *wavelets* obteve resultados positivos quanto à necessidade de processamento do codificador. O codificador mais simples, o *Speex-Wavelet Haar* executou 73 % menos instruções que o *Speex* durante a codificação e decodificação. Já o *codec Speex-Wavelet Coiflet30* executou 11 % menos que o *Speex.*

Um dos principais motivos para esta diferença de desempenho entre os codificadores desenvolvidos neste trabalho e o *Speex* deve-se ao uso de análise por síntese no algoritmo CELP, usado pelo *Speex.* No processo de codificação do algoritmo CELP é realizada uma busca pela melhor excitação nos dicionários adaptativo e estocástico em malha fechada, enquanto os codificadores desenvolvidos neste trabalho operam em malha aberta, tornando-se mais eficientes computacionalmente.

A diferença de desempenho foi menor no *codec Speex-Wavelet Coiflet30* em grande parte devido à quantidade de coeficientes usada pela transformada discreta *wavelet,* aumentando a necessidade de processamento tanto no codificador quanto no decodificador. No processo de codificação foi necessário executar 30 multiplicações de ponto flutuante com 32 *bits* de precisão para obter cada elemento do sub-sinal. Na decodificação, foi necessário executar a mesma quantidade de operações de ponto flutuante para obter cada elemento do sinal reconstruído.

Analisando o desempenho dos decodificadores, a quantidade de processamento do *Speex-Wavelet Haar* aumentou em 29 % se comparado ao *Speex*. Isso porque no decodificador do *Speex* a necessidade de processamento é menor, pois usa apenas os parâmetros recebidos do codificador na sintetização do sinal. Os decodificadores baseados em *wavelets* fazem à transformada inversa do sinal, uma operação que necessita tanto processamento quanto à transformada normal.

No caso específico do *Speex-Wavelet Haar*, a quantidade de instruções executadas no decodificador foi maior que no codificador porque na transformada normal o processo foi otimizado e na transformada inversa não.

Apesar da utilização de *wavelet Coiflet* com 30 coeficientes ser mais indicada para sinais de origem analógica, tanto o *Speex-Wavelet Haar* quanto o *Speex-Wavelet Coiflet30* obtiveram uma avaliação de qualidade semelhante. Porém, ambas as versões obtiveram resultados inferiores quanto à qualidade do sinal se comparado ao *Speex*.

Entretanto, apesar de possuir resultados inferiores ao *Speex* quanto à qualidade, os *codecs Speex-Wavelet* foram avaliados entre regular e bom, indicando um nível de qualidade aceitável.

Em relação à compactação do sinal, o *Speex* operou a 15 *kbps,* enquanto os *codecs Speex-Wavelet* operaram a taxa de 1280 *bits* por *frame*, ou 40 *kbps.*

A taxa de compactação e a qualidade obtida pelo *Speex* devem-se à evolução do algoritmo CELP, iniciado em 1980, contando várias contribuições para o aumento da qualidade, diminuição da necessidade de processamento e melhora da taxa de compactação ao longo deste tempo. Contudo, a necessidade de processamento do CELP representa um obstáculo para o uso em sistemas embarcados *VoIP*, sendo necessário usar um processador poderoso para realizar a codificação de voz.

O *codec Speex-Wavelet Haar* representa uma alternativa para realizar compactação de áudio em dispositivos de processamento limitado, contando com uma qualidade aceitável.

Contudo, os *codecs* desenvolvidos neste trabalho estão apenas em sua primeira versão, podendo ser aperfeiçoados futuramente para atingir melhores resultados em compactação, qualidade e desempenho, como será visto a seguir.

11.1 Trabalhos futuros

Visando aumentar a taxa de compactação obtida, pode ser usado o algoritmo para compressão de dados *SPIHT* (*Set Partioning In Hierarchical Tree*), utilizado principalmente em compressão de imagens. Também pode ser usada a transformada BWT (*Burrows–Wheeler transform*) (SALOMON, 2004), que tende a aumentar a ocorrência de valores repetidos, melhorando o desempenho de RLE (*Run-lenght encoding*).

Para melhorar a qualidade do sinal obtida pelo *codec*, é possível fazer uma filtragem de ruídos após a descompactação. É possível também, dependendo do resultado das técnicas de compressão, aumentar a quantidade de *bits* usada na quantização para aumentar a qualidade do sinal. Pode-se ainda implementar outros tipos de *wavelets,* buscando melhorar a compactação de energia e consequentemente, a taxa de compactação e qualidade.

Porém, todas essas modificações irão afetar o desempenho, sendo necessários ajustes para otimizar o mesmo. O ajuste que foi feito na execução da transformada discreta *Haar* poderá ser feito de forma similar para outros tipos de transformadas de maior complexidade, evitando que operações de radiciação e multiplicação sejam usadas com frequência no cálculo.

Uma mudança maior no funcionamento dos *codecs* existentes seria enviar o sinal em partes, objetivando menor tráfego de dados na rede. Inicialmente, os valores referentes à aproximação de uma parte do sinal são enviados. Depois em um próximo pacote são enviados os sinais de detalhe necessários para reconstruir esta parte do sinal. Na próxima parte enviada, verifica-se se o sinal de aproximação anterior pode ser usado para codificar a parte atual. Em caso positivo, são enviados apenas os sinais de detalhe, se não, são enviados os sinais de aproximação e detalhe.

Esta implementação possui desvantagens, por ser prejudicada com perdas de pacotes, no caso de o pacote referente à aproximação ser perdido. E também irá necessitar maior processamento, a fim de verificar quais os sinais do detalhe serão enviados, onde o teste seria feito em malha fechada.

12 REFERÊNCIAS

ANDROID. *Projeto Android*. Disponível em <http://www.android.com/>. Acesso em 20 mai. 2009.

AKANSU, A. N.; MEDLEY M. J. *Wavelet, subband, and block transforms in communications and multimedia*. Massachusetts: Springer, 1999

ARM: *Architecture Reference Manual*. ARM DDI 0100I. ARM Limited. [S.l.:s:n], 2005.

ARM920T (Rev 1) *Technical Reference Manual*. ARM DDI 0151C. ARM Limited. [S.l.:s:n], 2001.

ARMULATOR. *ARMulator*. Disponível em <http://www.uclinux.org/pub/uClinux/utilities/armulator/>. Acesso em 20 mai. 2009.

BECKER, D. et. al. *Automatic Trace-Based Performance Analysis of Metacomputing Applications*. In: International Parallel and Distributed Processing Symposium, 21., California:[s:n], 2007.

BARR, Michael. *Programming Embedded Systems in C and C++*. Sebastopol, California: O'Reilly & Associates, 1999.

BOCHS. *Bochs*. Disponível em <http://bochs.sourceforge.net/>. Acesso em 20 mai. 2009.

BURGER, D.C.; AUSTIN, T. M. *The SimpleScalar Tool Set, Version 2.0*. Madison: University of Wisconsin, 1997.

DAUBECHIES, Ingrid. *Wavelets and other phase space localization methods.* In: International Congress of Mathematicians. Basel: Birkhäuser Verlag, 1995.

DE MEULENEIRE, M. et al. *A Celp-Wavelet Scalable Wideband Speech Coder.* In:

Acoustics, Speech and Signal Processing. Toulouse:[s.n.], 2006.

DOHERTY, J.; ANDERSON, N. *Internet Phone Services Simplified (VoIP)*. 1 ed., Indianapolis: Cisco Press, 2006.

FLAC. *Desenvolvido pela equipe do projeto FLAC*. Disponível em <http://flac.sourceforge.net/>. Acesso em: 18 set. 2009.

GRAPS, Amara. *An Introduction to Wavelets*. In: IEEE Computational Science and Engineering, v. 2, n.2, [S.l:s.n], 1995. p. 55-61.

GOLDBERG, R.; RIEK, L., *A Practical Handbook of Speech Coders*. Nova Iorque: CRC Press, 2000.

HARDY, William C.; *VoIP Service Quality: Measuring and Evaluating Packet-Switched Voice*. [S.l.]:McGraw-Hill, 2003.

HINES, K.; BORRIELLO, G. *Optimizing communication in embedded system cosimulation*, In: Proceedings of the Fifth International Workshop on Hardware/Software Codesign, 5., [S.l.:s.n.],1997. p.121-125.

HUANG, X.D; ACERO, A.; HON H.-W. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Nova Jersey:Prentice Hall, 2001.

INTERNATIONAL TELECOMMUNICATION UNION. ITU-T Recommendation P.800: Methods for subjective determination of transmission quality. [S.l. : s.n.], 1996.

_____. ITU-T Recommendation P.861: Objective quality measurement of telephone-band (300-3400 hz) speech codecs. [S.l. : s.n.], 1998.

_____. ITU-T Recommendation P.862: Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. [S.l. : s.n.], 1998.

JAIN, R. K. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Chichester, Inglaterra: John Wiley & Sons, Inc. 1991

JSCI. A science API for Java. Disponível em <http://jsci.sourceforge.net/ >. Acesso em 30/11/2008.

LEVINSON, S. E. *Mathematical Models for Speech Technology*. Chichester, Inglaterra: John Wiley & Sons Ltd, 2005.

KIM, D.; YI, Y.; HA, S. *Trace-driven hw/sw cosimulation using virtual synchronization technique*. In: Proceedings of the 42nd annual Design Automation Conference, Nova Iorque:[s.n.], 2005.

KNAGGS, P.; WELSH, S. ARM: *Assembly Languague Programing. Bournemouth University*. School of Design, Engineering & Computing. Inglaterra:[s.n.], 2004

MERTINS, Alfred. *Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications*. Chichester, Inglaterra: John Wiley & Sons Ltd, 1999

NAGASWAMY, Sriram. *Comparison of CELP Speech Coder with a Wavelet Method*. Dissertação de mestrado. University of Kentucky, 2005. Disponível em <https://archive.uky.edu/handle/10225/283> Acesso em 20 out. 2007.

OOI, J.; VISWANATHAN, Vishu. A computationally efficient wavelet transform celp coder. In: *Acoustics, Speech, and Signal Processing,* 3., Adelaide, Australia:[s.n.], 1994

QEMU. *QEMU*. Disponível em < http://www.qemu.org/>. Acesso em 30 mai. 2009.

RABINER, L. R.; SCHAFER, R. W. *Digital processing of speech signals*. Nova Jersey, E.U.A.: Prentice Hall, 1978

RIDDEL, Jeff; *PacketCable Implementation*. Indianapolis: Cisco Press, 2007

SALOMON, David. *Data Compression The complete reference*. 3rd Edition, Nova Iorque: Springer, 2004

SIMICS. *Simcs*. Disponível em < http://www.virtutech.com/>. Acesso em 25 jul. 2009.

SOFTGUN. *SoftGun*. Disponível em <http://softgun.sourceforge.net/>. Acesso em 20 mai. 2009.

SPANIAS, A. S. *Speech Coding: A Tutorial Review*. In: PROCEEDINGS OF THE IEEE, 10., Arizona,Tempe:[s.n.] 1994. p. 1541-1582. Disponível em <http://www.eas.asu.edu/ ~spanias/papers/spanias94tutorial.pdf> Acesso em 30 nov. 2007.

STARK, H. G. *Wavelets and Signal Processing - An Application-Based Introduction.* Berlin: Springer, 2005

STOLLNITZ, E. J.; DEROSE, T. D.; SALESIN, D. H. *Wavelets for computer graphics: A primer*. In: IEEE COMPUTER GRAPHICS AND APPLICATIONS, 3., Washington, Seattle:[s.n.], 1995. p. 76-84

WALKER, James S. *A Primer on Wavelets and Their Scientific Applications*. Boca Raton, Flórida: CRC Press, 1999.

VALIN, Jean-Marc. *The Speex Codec Manual (version 1.2 Beta 2)*. [S.l.:s.n], 2007 Disponível em <http://www.speex.org> Acesso em 05 ago. 2007.

VALIN, Jean-Marc. *Speex: A Free Codec For Free Speech.* [S.l.:s.n], 2006. Disponível em <http://www.speex.org> Acesso em 05 ago. 2007.

VIRTUALBOX. *VirtualBox*. Disponível em < http://www.virtualbox.org/>. Acesso em 20 mai. 2009.

VOIPWIFI. Comunicação segura de voz sobre ip em redes wireless. Disponível em <http://www.inf.unisc.br/gpsem/gpsem/doku.php?id=voipwifi> Acesso em 20 jul. 2006.

ZANARTU, Matias. *Audio Compression using Wavelet Techniques*. Project Report. Purdue University. West Lafayette, Indiana, E.U.A.:[s:n], 2005. Disponível em <http://web.ics.purdue.edu/~mzanartu/Documents/Wavelets%20Project.pdf>. Acesso em 10 mar. 2008.