UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

E DE COMPUTAÇÃO

DEPARTAMENTO DE COMUNICAÇÕES

Tese de Doutorado

Um Método de Compressão de Áudio Baseado na Codificação de Subbandas Wavelets

Guillermo Leopoldo Kemper Vásquez

Orientador: Prof. Dr. Yuzo Iano

Banca Examinadora:

Prof. Dr. Guillermo Fernandez Segovia (UCV-Valparaiso-Chile)

Prof. Dr. Fernando Toshinori Sakane (ITA-São José dos Campos)

Prof. Dr. Edson Moschim (FEEC/UNICAMP)

Prof. Dr. Dalton Soares Arantes (FEEC/UNICAMP)

Prof. Dr. João Batista Destro Filho (FEEC/UNICAMP)

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.

Campinas – SP – Brasil

Fevereiro de 2001

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Kemper Vásquez, Guillermo Leopoldo
K323m Um método de compressão de áudio baseado na codificação de subbandas wavelets / Guillermo Leopoldo Kemper Vásquez. -- Campinas, SP: [s.n.], 2001.

Orientador: Yuzo Iano.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Compressão de dados (Telecomunicações). 2. Processamento de sinais. 3. Filtros e filtração. 4. Sistemas multimídia. 5. Simulação (Computadores digitais). 6. MATLAB (Programa de computador). I. Iano, Yuzo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

RESUMO

No presente trabalho propõe-se um método de codificação de áudio, baseado basicamente na quantização e codificação de subbandas *wavelets*.

No início, realiza-se um detalhamento da Transformada de *Wavelets*, enfocando-se a mesma a partir dos conceitos da análise de multiresolução e sua relação com os processos de filtragem e decomposição em subbandas. Descreve-se também a chamada transformada de *Wavelet Packets*, na qual é baseado o formato de decomposição utilizado no processo de compressão.

O codificador é formado também por um modelo psico-acústico, um algoritmo de alocação de bits, uma etapa de quantização escalar, uma etapa de quantização vetorial e uma etapa da codificação de entropia.

O modelo psico-acústico e o algoritmo de alocação de bits são baseados parcialmente naqueles utilizados pelo codificador MPEG-1 nas *Layers* 1 e 2.

O codificador utiliza um método de mapeamento que é proposto neste trabalho para se colocar os resultados do modelo no domínio das subbandas wavelets a fim de que os mesmos se tornem válidos para se determinar os quantizadores a serem utilizados na codificação das amostras subbandas.

Por outro lado, propõe-se também um formato de quantização vetorial e de codificação de entropia adaptado ao comportamento do conjunto de amostras subbandas que formam a informação de áudio a ser armazenada ou transmitida.

Usando-se essas técnicas de compressão, o codificador apresentou um bom desempenho para taxas em torno de 128, 96 e 80 kbit/s para sinais de áudio monocanais. O método de avaliação utilizado foi do tipo subjetivo e é detalhado ao final deste trabalho junto com os resultados e as conclusões finais pertinentes.

ABSTRACT

This work presents a method for audio compression based on wavelets sub-band quantization and coding, and proposes a coder based on that method.

First, the work describes the Wavelet Transform with emphasis on the concepts of multi-resolution analyzes and their relationship with filtering processes and decomposition in sub-bands. A description of the Wavelet Packets Transform is also included since it is used as a decomposition format by the proposed coder.

Then, the paper describes the elements of the proposed coder. They include a psychoacoustic model, a bit allocation algorithm, an escalar quantization stage, a vector quantizer, and an entropy coder. The psycho-acoustic model and the bit allocation algorithm are partially based on those used by the MPEG-1 coder in Layers 1 and 2.

The proposed coder introduces a method of mapping to place the results of the psychoacoustic model MPEG-1 layer 2 in the domain of the wavelets. In that way, the results of the model will be valid to determine the resolution of the quantizers to be used in coding each sub-band.

The proposed coder also introduces a format of vector quantization and an entropy code that is adapted to the behavior of the sub-band samples. Those samples are the audio information to be stored or transmitted.

Using these compression techniques the coder presents a good performance for rates around of 128, 96, and 80 kbit/s for mono-channel audio signals.

Finally, the paper discusses the results and explains the subjective-type evaluation method used.

AGRADECIMENTOS

A Deus Todo-Poderoso por tudo.

A minha querida esposa **Anellí**, a quem dedico também este trabalho pela sua paciência, compreensão e carinho que nunca faltaram.

Muito especialmente à **Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)** e ao **Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)**, pelo apoio econômico e financeiro.

Ao **Prof. Dr. Yuzo Iano** pelo estímulo e pela consideração mostrados desde o início do curso de pós-graduação.

Aos **meus queridos pais Nelly e Sixtilio, a meus irmãos : Roberto, Eduardo, Patricia e Carolina** e a minha família em geral, pelo seu carinho e apoio mostrados em todo momento.

Aos **meus amigos da faculdade** que sempre estiveram prontos para me ajudar e me apoiar em tudo que foi necessário.

Aos **professores membros da banca examinadora** agradeço a presença e a participação.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CD	:	*Compact Disk*
DAB	:	*Digital Audio Broadcasting,*
db	:	*Daubechies (wavelets)*
dB	:	*Decibeis*
DCT	:	*Discrete Cosine Transform*
DFT	:	*Discrete Fourier Transform*
DM	:	*Delta Modulation*
DPCM	:	*Diferential Pulse Code Modulation*
DSP	:	*Digital Signal Processor*
DST	:	*Discrete Sine Transform*
DTWT	:	*Discrete Time Wavelets Transform*
DWT	:	*Discrete Wavelets Transform*
FFT	:	*Fast Fourier Transform*
FIR	:	*Finite Impulse Response*
HDTV	:	*High Definition Television*
IFFT	:	*Inverse Fast Fourier Transform*
ISDN	:	*Integrated Services Digital Network.*
ISO/ IEC	:	*International Standard Organization/ International Engineering Consortium*
ITU	:	*International Telecommunication Union*
MDCT	:	*Modified Discrete Cosine Transform*
MNR	:	*Masking to Noise Ratio*
MPEG	:	*Motion Picture Expert Group*
MP1	:	*MPEG1-Layer1*
MP2	:	*MPEG1-Layer2*
MP3	:	*MPEG1-Layer3*
NN	:	*Nearest Neighbor Quantizer*
PRQMF	:	*Perfect Reconstruction Quadrature Mirror Filters*
QMF	:	*Quadrature Mirror Filter*
ROM	:	*Read Only Memory*
SDG	:	*Subjective Difference Grade*

SMR : *Signal to Masking Ratio*

SNR : *Signal to Noise Ratio*

SQNR : *Signal to Quantization Noise Ratio*

TDAC : *Time Domain Aliasing Cancellation*

VQ : *Vectorial Quantization*

LISTA DOS PRINCIPAIS SÍMBOLOS

$\phi(t)$: Função de Escala.

$\psi(t)$: Função Wavelet.

Λ_v : Hiperplano

$\eta_{sw}(p)$: Energia do ruído permitido para a subbanda "p".

$a(k,n)$: Coordenadas ou coeficientes de expansão das funções $f_k(t)$ em termos de suas respectivas funções bases no subespaço de aproximação V_k.

$absthr(k)$: Limiar mínimo absoluto de potência da componente de freqüência"k" para que seja perceptível pelo ouvido humano.

$b(k,n)$: Coordenadas ou coeficientes de expansão das funções $f_k(t)$ em termos de suas respectivas funções bases no subespaço de detalhes W_k.

$B_{TR}(p)$: Número de bits do código de identificação da árvore de procura a ser utilizado para a subbanda "p".

B_x : Espaço (em numero de bits) disponível no *buffer* de saída.

$escal(p)$: Fator de escala da subbanda "p".

$escal_q(p)$: Fator de escala quantizado para a subbanda "p".

$E'_{sw}(p)$: Energia contida na subbanda *wavelet* "p".

f_m : Freqüência de amostragem

$f_k(t)$: Aproximação de uma função $f(t)$ no nível "k"

$g_d(n)$: Resposta impulsiva do filtro wavelet de decomposição passa-altas.

$g_r(n)$: Resposta impulsiva do filtro wavelet de reconstrução passa-altas.

$g_{Td}(n)$: Resposta impulsiva do filtro de decomposição passa-altas, utilizado no mapeamento da função de autocorrelação $R(n)$ no domínio das wavelets.

$h_d(n)$: Resposta impulsiva do filtro wavelet de decomposição passa-baixas.

$h_r(n)$: Resposta impulsiva do filtro wavelet de reconstrução passa-baixas.

$h_{Td}(n)$: Resposta impulsiva do filtro de decomposição passa-baixas, utilizado no mapeamento da função de autocorrelação $R(n)$ no domínio das wavelets.

H_v : *Half-space*

$J_B(n)$: Janela utilizada na segmentação do sinal de áudio em blocos de amostras.

k_d : Dimensão de um quantizador vetorial

$L(p)$: Número de amostras que forma a subbanda "p".

L_B : Tamanho em número de amostras do bloco de entrada.

L_{ov} : Número de amostras de superposição entre blocos de áudio.

$mthr(k)$: Densidade espectral de potência do ruído mínimo permitido para a componente "k".

n_{ba} : Representa o número de bits utilizados na codificação do vetor $n_{bs}(p)$.

n'_{ba} : Número de bits utilizado na codificação entrópica diferencial do vetor de alocação $n_{bs}(p)$.

"k" : Número de bits disponíveis para se codificar um bloco de áudio.

$n_{be}(n_{bs}(p))$: Número de bits utilizado na identificação (códigos de 4 bits) dos *codebooks* Huffman a serem utilizados na codificação entrópica da subbanda "p".

n_{bfs} : Representa o número de bits utilizados na codificação do fator de escala.

n'_{bfs} : Número de bits utilizado na codificação entrópica diferencial do fator de escala.

$n_{BL}(p)$: Tamanho em número de amostras dos blocos a ser quantizado na subbanda "p".

$n_{bs}(p)$: Número de bits por amostra alocado á subbanda "p".

$n_{et}(p)$: Número de etapas da árvore que foi utilizado no processo de busca.

$n_{ev}(p)$: Número de bits utilizado na codificação entrópica da subbanda "p"

$N_{MUL/AD}(p)$: Número de multiplicações ou somas envolvidas no cálculo de todos os vetores quantizadores para uma determinada subbanda "p".

$N_{nq}(p)$: Níveis de quantização escalar utilizados na codificação da subbanda "p".

$N_V(p)$: Número de bits a ser utilizado na identificação dos vetores quantizadores calculados para uma determinada subbanda "p".

p : Índice utilizado na identificação das 29 subbandas wavelets em que foi decomposto o sinal de áudio de entrada.

$R(n)$: Função de autocorrelação do ruído.

R_i : I-ésima célula que faz parte de um espaço euclidiano R^{k_d} particionado em células ou regiões.

R^{k_d} : Espaço Euclidiano de "k_d" dimensões.

$R_{sw}(p,n)$: n-ésima amostra da função de autocorrelação $R(n)$ mapeada na subbanda wavelet "p".

R_{tx} : Taxa de transmissão requerida do codificador de áudio proposto.

$SMR_{sw}(p)$: Valor SMR da subbanda wavelet "p".

$thr(k)$: Limiar absoluto de potência do ruído permitido para a componente de freqüência "k".

V_k : Subespaço de aproximação no nível "k"

W_k : Subespaço de Detalhes.

$x_{AJ}(n)$: Seqüência que forma o bloco de 1024 amostras resultante do processo de janelamento e superposição.

$\hat{x}_{AJ}(n)$: Amostras dos blocos reconstruídos da transformada.

$x_{sw}(p,n)$: Constitui a amostra "n" da subbanda "p".

$\hat{x}_{sw}(p,n)$: n-ésima amostra dequantizada da subbanda "p".

$x^q_{sw}(p,n)$: n-ésima amostra quantizada da subbanda "p".

y_i : I-ésimo vetor quantizador que faz parte do *codebook* de um quantizador vetorial.

CAPÍTULO 1

ASPECTOS INTRODUTÓRIOS

1.1 INTRODUÇÃO

Os estudos sobre novos processos de codificação e compressão de sinais têm tido considerável aumento nos últimos anos. Isso se deve principalmente ao fato de que existe um grande interesse por parte de cientistas e pesquisadores, em descobrir e implementar novas técnicas de compressão e codificação que permitam aumentar o desempenho dos diferentes tipos de codificadores tornando assim mais eficiente a utilização dos canais de transmissão.

Hoje em dia, embora existam meios como as fibras ópticas que permitam a transmissão de grande quantidade de informação por unidade de tempo, a crescente demanda de serviços e o elevado custo de espaço nos meios de transmissão fazem cada vez mais necessária a utilização de sistemas de compressão, capazes de adequar os diferentes tipos de sinais às limitações ou requisitos de capacidade de um canal de transmissão.

O desempenho dos sistemas de compressão de sinais é medido, geralmente, através dos parâmetros **qualidade/ fator de compressão** ou **qualidade/ taxa de bits**. Para o primeiro caso, se diz que um codificador é de bom desempenho quando, para um considerável fator de compressão, a qualidade do sinal resultante é pouco afetada pelos processos de codificação aos quais foi submetido. No segundo caso, quanto menor for a taxa de bits resultante e quanto menos perceptível for a distorção introduzida, se diz que tanto melhor é o desempenho do codificador.

De acordo com estes parâmetros e com o grau de complexidade dos diversos algoritmos de compressão utilizados, os diversos codificadores hoje existentes são classificados em codificadores de baixa, intermediária e alta complexidade.

Os codificadores de alto nível de complexidade, atingem altos fatores de compressão, sem afetar seriamente a qualidade do sinal resultante. Geralmente esses codificadores são utilizados em aplicações de armazenamento, ou de tempo não real.

Os codificadores de intermediária complexidade, por outro lado, são utilizados em aplicações de radiodifusão (*broadcasting*) onde se requer a codificação e a decodificação de sinais em tempo real mantendo o nível de qualidade requerido pelos sistemas de comunicações onde são utilizados.

Na atualidade, existe uma ampla gama de diferentes tipos de codificadores de áudio que são utilizados em diferentes aplicações.

Entre esses codificadores podemos encontrar aqueles que pertencem à família MPEG (*Motion Picture Expert Group*) e que, durante os últimos anos, têm sido aperfeiçoados regularmente através de suas distintas versões lançadas no mercado.

Os primeiros codificadores dessa família foram aqueles da linha MPEG-1, os quais de acordo ao nível de complexidade e desempenho, foram classificados em três camadas ou *layers* diferentes.

A **Layer-1** (MP1) por exemplo é bastante simples e é adequada para taxa de bits **acima de 128 kbit/ s** (por canal). Esse formato de codificação é utilizado no **Philips Digital Compact Cassette (DCC)** para uma taxa de bits de **192 kbit/ s** (por canal).

Por outro lado, a **Layer-2** (MP2) especifica um formato de compressão de intermediária complexidade e é adequada para taxa de bits de aproximadamente **128 kbit/ s** (por canal). É apropriada para aplicações **DAB** (*Digital Audio Broadcasting*), para armazenamento de vídeo e áudio sincronizado sobre CD-ROM, Vídeo CD, etc.

Finalmente, a **Layer-3** (MP3) especifica um formato de compressão bastante complexo e é adequado para taxas de bits de aproximadamente **64 kbit/ s** (por canal). Essa camada apresenta a melhor qualidade de áudio e é apropriada para transmissões sobre sistemas **ISDN** (*Integrated Services Digital Network*).

Posteriormente, o MPEG lançou o codificador MPEG-2 - Multicanal (5 canais de áudio de 20kHz mais um canal de baixas freqüências *superwoofer* de aproximadamente 120Hz) apropriado para filmes de cinema e sistemas HDTV (*High Definition Television*).

Outros codificadores importantes são aqueles pertencentes à família DOLBY e que, também, durante os últimos anos têm evoluído notavelmente através das versões **DOLBY AC-1, DOLBY AC-2 e DOLBY AC-3 (AC – Audio Coding)**. Essa última versão, por exemplo, foi escolhida pela **Grande Aliança em Televisão dos Estados Unidos**, para ser o codificador de áudio utilizado no sistema proposto para HDTV naquele país. Os codificadores DOLBY são concorrentes dos codificadores MPEG para distintos tipos de aplicações. No entanto, um aspecto

comum entre eles é a codificação do sinal de áudio no domínio da transformada MDCT (**Modified Discrete Cosine Transform**), que é uma versão modificada das transformadas DCT (*Discrete Cosine Transform)* convencionais. A modificação é realizada, geralmente, na fase das componentes co-senoidais de acordo com a teoria de Cancelamento de Superposição (*Aliasing)* no Domínio do Tempo (TDAC).

No codificador que é proposto através deste trabalho de tese (considerado de intermediária complexidade) utiliza-se no lugar da MDCT a chamada transformada de **Wavelets Packets**, a qual permite a decomposição do sinal de áudio em espaços ortogonais ou biortogonais (dependendo do caso) de diferentes tamanhos (largura de banda variável) chamados de Subbandas *Wavelet*s.

Para se obter a compressão desejada, é necessário uma quantização adicional, e este processo gera e introduz distorção no sinal decodificado. Portanto, o desempenho do codificador será melhor à medida que a distorção se torne mais imperceptível para consideráveis taxas de compressão.

A re-quantização da informação no formato de compressão proposto é aplicada às subbandas *wavelet*s resultantes da decomposição do sinal de entrada.

Cada subbanda tem seu respectivo mapeamento no domínio da freqüência, o qual permite identificar as componentes espectrais contidas em cada uma delas. Isso facilita e torna flexível a utilização de um formato de decomposição que se acomode ao esquema de bandas críticas do ouvido humano. Assim, as subbandas resultantes podem ser independentemente codificadas e quantizadas facilitando o processo de compressão (descorrelação da informação).

O número de níveis de quantização designado para cada subbanda é calculado a partir de um modelo psico-acústico que determina componentes de freqüência altamente audíveis, pouco audíveis ou inaudíveis.

Dessa forma, a resolução do quantizador designado para uma determinada componente será tanto maior quanto mais audível for a mesma. Isso logicamente permite um ganho de compressão, já que, em média, o número de bits por amostra necessário para se codificar toda a informação de áudio no domínio da transformada é muito menor do que aquele do sinal original.

O modelo psico-acústico usado neste trabalho é baseado naquele aplicado no sistema de compressão de áudio **MP2**. Esse modelo foi projetado para determinar o número de níveis de quantização necessário para codificar os coeficientes MDCT que formam o espectro do sinal de entrada.

Portanto, para que os resultados desse modelo sejam válidos no codificador proposto, é necessário um mapeamento adequado dos mesmos no domínio das subbandas *wavelet*s.

Dessa forma, neste trabalho se propõe um método de mapeamento que permitiu obter resultados muito satisfatórios para taxas em torno de 96 kbit/s (monocanal).

Após a conclusão do processo de quantização, cada subbanda é submetida a um processo de codificação de entropia com o objetivo de se remover qualquer redundância estatística introduzida no sinal de áudio.

É utilizada também a técnica de quantização vetorial para se codificar as subbandas *wavelet*s correspondentes às altas freqüências do espectro auditivo. Essa técnica se encontra continuamente em fase de desenvolvimento para aplicação em diversos tipos de codificadores de sinais. Segundo vários autores, constitui uma das ferramentas de compressão que serão mais utilizadas no futuro.

A quantização vetorial encontra sua aplicação em domínios onde se requer um valor SNRQ (relação sinal / ruído de quantização) relativamente não muito alto (como é o caso das subbandas correspondentes às altas freqüências do espectro auditivo). A vantagem disso está no fato de que este método de quantização permite alcançar ganhos de compressão significativos que justificam sua utilização em vários tipos de sistemas de codificação.

Finalmente, a avaliação do codificador é realizada utilizando-se métodos subjetivos, já que na atualidade não existe um método objetivo adequado que possa indicar de forma confiável o nível de qualidade do sinal decodificado. Os resultados da avaliação são apresentados em função da qualidade do sinal reconstruído, da *wavelet* utilizada e da taxa de bits resultante do processo de compressão.

Os algoritmos de codificação foram implementados em computador utilizando o **MATLAB 5.0**. O sinal original (diversos tipos de música e som) foi extraído de um **CD** (compact disk) convencional de música (freqüência de amostragem: **44.1kHz**; e **16 bits/ amostra**) através de fragmentos de 8 a 12 segundos de duração. Cada fragmento de música foi armazenado como um arquivo **WAV** do formato multimídia e em sistema de áudio monocanal. Com essas características, o sinal original de entrada no codificador apresentará uma taxa de **705.6 kbit/ s**.

1.2 OBJETIVOS DO TRABALHO

Os principais objetivos deste trabalho são os seguintes:

- Propor um codificador de áudio de intermediária complexidade baseado na quantização e codificação de subbandas *wavelet*s.
- Estudar o desempenho da transformada de w*avelet*s em processos de codificação e compressão de sinais de áudio.

- Implementar técnicas de codificação de subbandas *wavelet*s que permitam obter ganhos consideráveis de compressão.
- Adaptar técnicas de codificação convencionais para serem utilizadas no domínio das sub-bandas *wavelet*s.
- Comparar o codificador proposto com aqueles de intermediária complexidade que utilizam outros tipos de transformadas.

1.3 DESCRIÇÃO DO TRABALHO

O presente trabalho é dividido em 5 capítulos que vão desde a descrição dos aspectos teóricos até os resultados e as conclusões pertinentes.

Dessa forma, no **Capítulo 2,** inicia-se com uma introdução à teoria de *Wavelet*s enfocando a mesma através dos conceitos da análise de multiresolução. Serão também estudados os processos de filtragem utilizados na implementação da transformada, assim como o detalhamento da transformada de *Wavelets Packets* que será usada neste trabalho. Esses conhecimentos ajudarão posteriormente a entender a aplicação e utilização das *wavelet*s em codificadores de sinais de áudio.

No **Capítulo 3,** por outro lado, realiza-se uma descrição das principais técnicas de codificação e compressão que serão utilizadas no modelamento do codificador proposto. Aqui serão estudados os principais conceitos que descrevem o modelo psico-acústico assim como a técnica de quantização vetorial e o procedimento de aplicação da mesma no codificador proposto.

A descrição do codificador é realizada, finalmente, no **Capítulo 4**, onde se descreve o funcionamento e as características das diversas partes que conformam o mesmo. Igualmente se detalharão os critérios que foram levados em conta para a utilização das diferentes técnicas de codificação e de processamento de sinais.

No **Capítulo 5** são apresentados os resultados do trabalho, os quais foram obtidos através de testes subjetivos de áudio. Sobre esses resultados serão apresentadas também as conclusões e os comentários conforme o caso.

CAPÍTULO 2

TRANSFORMADA DE *WAVELET*S: ANÁLISE DE MULTIRESOLUÇÃO E *WAVELET PACKETS*

2.1 INTRODUÇÃO

Durante os últimos anos, uma grande quantidade de trabalhos que descrevem a Teoria de *Wavelet*s têm sido desenvolvidos e publicados com diversas aplicações nas diferentes áreas da eletrônica e das telecomunicações modernas. Naturalmente, nesse conjunto de publicações, não poderiam faltar trabalhos de pesquisa orientados à utilização e exploração dessa transformada em sistemas de processamento, compressão e codificação de sinais.

Hoje em dia pode-se ver que existe uma grande quantidade de codificadores propostos, utilizando distintos métodos de codificação, e que tem atingido resultados muito satisfatórios no que se refere ao desempenho das *wavelet*s em processos de codificação de imagem, áudio, vídeo e voz.

Os benefícios e as vantagens da utilização dessa transformada em sistemas de compressão poderão ser facilmente entendidos enfocando-se a mesma a partir dos conceitos da análise de multiresolução. Isso leva ao princípio de que a transformada de *wavelet*s permite decompor qualquer sinal em espaços de diferentes escalas e tamanhos (ortogonais ou biortogonais dependendo do caso) que podem ser localizados num determinado instante de tempo. Esses espaços armazenam distintas características e detalhes do sinal decomposto e são chamados

geralmente de **subbandas wavelets** (nome atribuído principalmente na literatura de processamento de sinais).

O processo é similar a se observar um mapa geográfico a partir de diversos níveis de escala, onde os detalhes perdidos entre uma escala e o nível seguinte de menor resolução constituem a informação disponível em cada espaço de detalhes (denomina-se espaço de detalhes devido ao fato de que o mesmo armazena uma parte das características do sinal decomposto).

Do ponto de vista de tratamento de sinais, pode-se dizer que uma sub-banda *wavelet* agrupa um conjunto de amostras cujo conteúdo espectral corresponde a uma das faixas de freqüências que formam o espectro do sinal original.

Sob esse mecanismo é baseado também o sistema de audição humano, o qual funciona como um banco de filtros que decompõe o sinal de entrada em bandas de largura variável, sendo que as bandas correspondentes às altas freqüências são de maior largura do que aquelas que correspondem às baixas freqüências (maior resolução do ouvido para componentes de baixa freqüência).

O ouvido humano apresenta por outro lado uma sensibilidade de percepção variável para cada uma das bandas resultantes deste processo, as quais recebem também o nome de **Bandas Críticas do Sistema de Audição Humano**.

A sensibilidade variável constitui, dessa forma, um benefício para o processo de compressão. De fato, por esse motivo, cada subbanda *wavelet* similar a uma banda crítica pode ser independentemente quantizada, codificada e transmitida.

A maior vantagem no entanto encontra-se no fato de que as subbandas de alta potência podem mascarar subbandas vizinhas de menor nível de energia, as quais podem se tornar quase inaudíveis, e permitir maior nível de distorção ou menor número de bits por amostra no processo de quantização (menor número de níveis de quantização). Este fenômeno é chamado de Efeito de Mascaramento (*masking*), o qual será estudado com mais detalhe no Capítulo 3.

A implementação da transformada, por outro lado, é realizada através de consecutivos processos de filtragem que obedecem a uma árvore de decomposição. Para esse caso, o formato da árvore deve permitir a geração de subbandas com características similares às bandas críticas do ouvido humano.

O formato convencional da transformada de *wavelet*s, no entanto, não permite obter subbandas adequadas para esse objetivo, sendo que em seu lugar é utilizada a chamada **Transformada de Wavelet Packets**, a qual apresenta um formato de árvore que flexivelmente pode se adaptar ao esquema das bandas críticas.

Por outro lado, os filtros usados no processo de decomposição e reconstrução devem satisfazer também as condições de ortogonalidade e da análise em multiresolução, em função do tipo *wavelet* utilizada.

Em sistemas de compressão de áudio, geralmente são utilizados filtros *wavelet*s com certo nível de seletividade, a fim de se evitar uma alta distorção por efeitos de superposição de subbandas adjacentes. Os filtros deverão ser também mais seletivos na medida em que se exija um maior fator de compressão do codificador. Porém, a utilização de filtros muito seletivos poderá comprometer o tempo de processamento dos algoritmos de decomposição e reconstrução, já que o número de coeficientes que conforma a resposta impulsiva dos filtros (FIR) é também incrementado com a seletividade exigida no domínio da freqüência.

No presente capítulo serão estudados, portanto, os conceitos básicos da teoria da análise de multiresolução e as características dos filtros de decomposição e reconstrução que poderão ser utilizados na implementação da transformada. Se descrevem também os princípios básicos da transformada de *wavelet packets* que dá origem as subbandas utilizadas no codificador proposto.

É importante mencionar que grande parte do equacionamento matemático que será apresentado neste capítulo foi desenvolvido a partir das referências [1], [2] e [3].

2.2 ANÁLISE DE MULTIRESOLUÇÃO

Uma análise de multiresolução em $L^2(R)$ consiste de uma seqüência de sucessivas aproximações ou subespaços $.....V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2}.....$ que satisfazem as seguintes condições :

1. $$\overline{\bigcup_{k \in Z} V_k} = L^2(R) \tag{2.1}$$

2. $$\bigcap_{k \in Z} V_k = \{0\} \tag{2.2}$$

Essa última expressão afirma que a função nula é a única função que pode ser representada em qualquer escala. Na realidade, é claro que qualquer função constante pode ser representada em qualquer escala, no entanto a única função constante em $L^2(R)$, é a

função nula [3].

3. Se $f(t) \in V_k$ então $f(2t) \in V_{k-1}$, e vice-versa. A função $f(t)$ constitui uma função integrável no espaço $L^2(R)$. Por outro lado a condição 3 implica que a dilatação por um fator de 2 de qualquer função, permite gerar uma nova função localizada no subespaço seguinte de menor resolução ou aproximação. Assim mesmo, dilatando por um fator igual a $1/2$, pode-se gerar uma nova função localizada, neste caso, no espaço seguinte que é de maior resolução (ou aproximação).

4. Existe uma função (chamada **função de escala**) $\phi(t)$ tal que $\{\phi(t-n) : n \in Z\}$ constitui uma base para o subespaço V_0.

   Essa condição requer assim a existência de uma função de escala e um conjunto $\{\phi(t-n) : n \in Z\}$ que sejam linearmente independentes.

   As bases constituem um método natural de se obter a representação de um espaço de funções. Diz-se que um conjunto $B = \{e_j ; j \in Z\}$ constitui uma base de um espaço de funções F, se para cada $f \in F$ existe uma seqüência de números complexos $(\alpha_j)_{j \in Z}$, tal que:

$$f = \sum_{j \in Z} \alpha_j e_j \tag{2.3}$$

Esta igualdade significa a existência de convergência das somas parciais da série segundo a norma do espaço F:

$$\lim_{n \to \infty} \left\| f - \sum_{j=-n}^{n} \alpha_j e_j \right\| = 0 \tag{2.4}$$

Assim, pode-se observar que a partir da equação (2.3) é possível se reconstruir a função f através da seqüência de representação (α_j).

Num espaço de funções pode-se também impor hipóteses adicionais para uma base a fim de garantir unicidade da representação [3]. Um caso particular de grande importância ocorre

quando o espaço de funções possui um produto interno e é completo na norma desse produto interno. Esses espaços são denominados comumente de **Espaços de Hilbert** e uma coleção de funções $\{\varphi_n ; n \in Z\}$ num espaço de Hilbert separável H é um conjunto ortonormal completo se as três condições abaixo são satisfeitas [3]:

- **Ortogonalidade**: $\langle \varphi_m, \varphi_n \rangle = 0$ se $n \neq m$;
- **Normalização**: $\|\varphi_n\| = 1$ para cada $n \in Z$;
- **Completeza**: Para todo $f \in H$ e todo $\varepsilon > 0$ se tem:

$$\left\| f - \sum_{k=-N}^{N} \langle f, \varphi_k \rangle \varphi_k \right\| < \varepsilon \tag{2.5}$$

De acordo com esses conceitos pode-se dizer, portanto, que qualquer função $f_0(t) \in V_0$ pode ser expressa como:

$$f_0(t) = \sum_{n=-\infty}^{\infty} a(0,n)\phi(t-n) \tag{2.6}$$

Onde $a(0,n)$ constitui uma seqüência de escalares para $n = 0, \pm 1, \pm 2, \ldots\ldots$.

2.2.1 FUNÇÃO DE ESCALA E SUB-ESPAÇOS

Uma função de escala satisfaz as seguintes condições:

1. Integral igual a "1":

$$\int_{-\infty}^{\infty} \phi(t)\, dt = 1 \tag{2.7}$$

2. Energia unitária:

$$\|\phi(t)\|^2 = \int_{-\infty}^{\infty} |\phi(t)|^2 = 1 \tag{2.8}$$

3. O conjunto $\phi(t)$ e suas translações inteiras formam uma base ortonormal:

$$\langle \phi(t), \phi(t-n) \rangle = \delta(n) \tag{2.9}$$

substituindo, t por $2^{-k}t$ em (2.9), obtemos:

$$\langle \phi(2^{-k}t), \phi(2^{-k}t-n) \rangle = 2^k \delta(n) \tag{2.10}$$

o qual indica que o conjunto:

$$\{\phi(2^{-k}t-n) : n \in Z\} \tag{2.11}$$

é também um conjunto ortogonal (para um valor dado de k) localizado no subespaço V_k .

Os subespaços $.....V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2}.....$ satisfazem, por outro lado:

$$f(t) \in V_k \Leftrightarrow f(2t) \in V_{k-1} \tag{2.12}$$

e levando-se em conta que :

$$V_0 \subset V_{-1} \tag{2.13}$$

pode-se concluir que se $\phi(t)$ está localizada no espaço V_0 , então deve estar também localizada no espaço V_{-1} . Por esse motivo, deve ser possível expressar $\phi(t)$ como uma combinação linear das bases localizadas em V_{-1} : $\{\phi(2t-n) : n \in Z\}$, o qual dá origem assim à equação de dilatação expressa como:

$$\phi(t) = \sum_{n=-\infty}^{\infty} c(n)\phi(2t-n) \tag{2.14}$$

onde $c(n)$, $n = 0, \pm 1, \pm 2...$, constitui uma seqüência de coeficientes escalares.

O nome "equação de dilatação" deve-se ao fato de que $\phi(t)$ pode ser expressa em termos de sua própria dilatação e translação.

Substituindo por outro lado t por $2^{-k}t$ em (2.14) , obtém-se:

$$\phi(2^{-k}t) = \sum_{n=-\infty}^{\infty} c(n)\phi(2^{-(k-1)}t - n) \tag{2.15}$$

o qual implica que as bases para V_k estão contidas em V_{k-1}:

$$V_k \subset V_{k-1} \tag{2.16}$$

2.2.2 IMPLICAÇÕES DA EQUAÇÃO DE DILATAÇÃO E DA ORTOGONALIDADE

A primeira condição imposta pela equação de dilatação (2.14) é a **condição dos coeficientes**. Essa condição é determinada a partir da integração de ambos os lados de (2.14) e cancelando logo em seguida os termos comuns [1]. Dessa forma obtém-se :

$$\sum_{n=-\infty}^{\infty} c(n) = 2 \tag{2.17}$$

Das equações (2.10) e (2.14) obtém-se também:

$$\langle \phi(t), \phi(t-\ell) \rangle = \int \phi(t)\phi(t-\ell)dt = \delta(\ell) \tag{2.18}$$

$$\langle \phi(t), \phi(t-\ell) \rangle = \frac{1}{2}\sum_{m=-\infty}^{\infty} c(m)c(m-2\ell) = \delta(\ell) \tag{2.19}$$

onde (2.18) constitui a chamada condição de ortogonalidade.

Define-se:

$$R(\tau) = \langle \phi(\tau), \phi(t-\tau) \rangle \tag{2.20}$$

como sendo a função de autocorrelação de $\phi(t)$ [1] . A equação (2.18) especifica, portanto, que $R(\tau)$ é zero para qualquer valor inteiro de "τ" exceto para $\tau = 0$.

Por outro lado, a equação (2.19) especifica um detalhe a respeito da autocorrelação de seqüência $c(n)$. Para se determinar isso definimos a seqüência de autocorrelação $c(n)$ como [1]:

$$r_c(\ell) = \sum_{n=-\infty}^{\infty} c(m)c(m-\ell) \tag{2.21}$$

logo, a partir de (2.19) obtém-se :

$$r_c(2\ell) = 2\delta(\ell) \tag{2.22}$$

Isso implica que a seqüência de autocorrelação de $c(n)$ é zero para deslocamentos pares diferentes de zero.

Da equação (2.14) tem-se também que:

$$\langle \phi(t), \phi(2t-m) \rangle = \sum_{n=-\infty}^{\infty} c(n) \langle \phi(2t-m), \phi(2t-n) \rangle \tag{2.23}$$

Usando-se a equação (2.10) e o lado direito da equação (2.23) obtém-se:

$$\langle \phi(t), \phi(2t-m) \rangle = \frac{c(m)}{2} \tag{2.24}$$

Definindo-se:

$$h(m) = \frac{c(m)}{2} \tag{2.25}$$

obtém-se

$$\langle \phi(t), \phi(2t-m) \rangle = h(m) \tag{2.26}$$

e de (2.17), obtém-se a condição:

$$\sum_{n=-\infty}^{\infty} h(n) = 1 \tag{2.27}$$

Por outro lado, tomando-se a transformada de *Fourier* de ambos os lados da equação (2.14) e utilizando-se a equação (2.25) obtém-se:

$$\Phi(w) = \sum_n h(n) e^{\frac{-j2\pi wn}{2}} \Phi(w/2) \tag{2.28}$$

onde $\Phi(w)$ é a transformada de *Fourier* de $\phi(t)$.

Agrupando-se termos podemos expressar (2.28) da seguinte forma:

$$\Phi(w) = H(w/2)\Phi(w/2) \tag{2.29}$$

onde :

$$H(w) = \sum_n h(n) e^{-jwn} \tag{2.30}$$

é a transformada de Fourier da seqüência $h(n)$ e constitui, logicamente, uma função periódica com período igual a 2π.

Da ortogonalidade de $\phi(t)$ e da periodicidade de e^{jwn} tem-se [2]:

$$\delta(n) = \int \phi(t)\phi(t-n)dt = \int |\Phi(w)|^2 e^{jwn} dw = \int_0^{2\pi} e^{jwn} \sum_{\ell \in Z} |\Phi(w+2\pi\ell)|^2 dw \tag{2.31}$$

o qual implica que:

$$\sum_{\ell \in Z} |\Phi(w+2\pi\ell)|^2 = 1 \tag{2.32}$$

Substituindo-se (2.29) em (2.32) e usando-se a periodicidade de $H(w)$ tem-se:

$$\sum_{\ell \in Z} \left| H\left(\frac{w+2\pi\ell}{2}\right) \Phi\left(\frac{w+2\pi\ell}{2}\right) \right|^2 = 1 \tag{2.33}$$

Decompondo-se o lado direito de (2.33) em duas somatórias (para valores pares e impares de "ℓ") obtém-se [2]:

$$\sum_{\ell\in Z}\left|H\left(\frac{w+4\pi\ell}{2}\right)\Phi\left(\frac{w+4\pi\ell}{2}\right)\right|^2+\sum_{\ell\in Z}\left|H\left(\frac{w+4\pi\ell+2\pi}{2}\right)\Phi\left(\frac{w+4\pi\ell+2\pi}{2}\right)\right|^2=1 \tag{2.34}$$

Fazendo-se $w'=w/2$ e substituindo-se em (2.34), tem-se:

$$\sum_{\ell\in Z}\left|H(w'+2\pi\ell)\Phi(w'+2\pi\ell)\right|^2+\sum_{\ell\in Z}\left|H(w'+2\pi\ell+\pi)\Phi(w'+2\pi\ell+\pi)\right|^2=1 \tag{2.35}$$

Logo, aplicando a condição (2.32) e levando-se em conta a periodicidade de $H(w)$, obtém-se finalmente:

$$\left|H(w')\right|^2+\left|H(w'+\pi)\right|^2=1 \tag{2.36}$$

Essa equação constitui a condição de ortogonalidade (2.22) expressa no domínio da freqüência [1]. Essa expressão é conhecida também na literatura como a **condição da potência complementaria**.

Define-se agora uma função $f(t)$ cujas projeções ortogonais sobre os espaços V_k, $k=0,\pm1,\pm2,....$ são dadas por $f_k(t)$, $k=0,\pm1,\pm2,....$. Logo, pode-se representar essas projeções em termos das funções bases de (2.11) utilizando-se a seguinte expressão:

$$f_k(t)=\sum_{n=-\infty}^{\infty}a(k,n)\phi(2^{-k}t-n) \tag{2.37}$$

As seqüências $a(k,n)$ são as coordenadas ou coeficientes de expansão das funções $f_k(t)$ em termos de suas respectivas funções bases. Refere-se também a $f_k(t)$ como sendo a aproximação de $f(t)$ no nível "k" [1].

O objetivo agora é estabelecer as relações entre os coeficientes de um nível de aproximação e o nível seguinte consecutivo. Para isso inicia-se com o caso particular das projeções $f_{-1}(t)$ e $f_0(t)$. Assim, da equação (2.37) tem-se que:

$$f_{-1}(t)=\sum_{n=-\infty}^{\infty}a(-1,n)\phi(2t-n) \tag{2.38}$$

$$f_0(t) = \sum_{n=-\infty}^{\infty} a(0,n)\phi(t-n) \tag{2.39}$$

Devido ao fato de que $f_0(t)$ é a projeção ortogonal de $f_{-1}(t)$ em V_0 , pode-se dizer que o erro ou função detalhe dado por:

$$g_0(t) \equiv f_{-1}(t) - f_0(t) \tag{2.40}$$

é ortogonal ao sub-espaço V_0 e particularmente a suas bases. Portanto:

$$\langle f_{-1}(t) - f_0(t), \phi(t-n)\rangle = 0 \tag{2.41}$$

para qualquer inteiro "n" .

Igualmente:

$$\langle f_0(t), \phi(t-n)\rangle = \langle f_{-1}(t), \phi(t-n)\rangle \tag{2.42}$$

Da equação (2.39) e da ortonormalidade de $\phi(t)$ com suas translações inteiras na equação (2.9), pode-se definir o lado esquerdo de (2.42) como sendo:

$$\langle f_0(t), \phi(t-n)\rangle = a(0,n) \tag{2.43}$$

Logo, da equação (2.38) tem-se que:

$$\langle f_{-1}(t), \phi(t-n)\rangle = \sum_{m=-\infty}^{\infty} a(-1,m)\langle \phi(t-n), \phi(2t-m)\rangle \tag{2.44}$$

e em combinação com (2.24) obtém-se:

$$\langle f_{-1}(t), \phi(t-n)\rangle = \sum_{m=-\infty}^{\infty} a(-1,m)\frac{c(m-2n)}{2} \tag{2.45}$$

Substituindo-se (2.43) e (2.45) em (2.42), tem-se :

$$a(0,n) = \sum_{m=-\infty}^{\infty} a(-1,m)\frac{c(m-2n)}{2} \tag{2.46}$$

A equação (2.46) é obtida para o caso particular de $f_{-1}(t)$ e $f_0(t)$.

Para um caso mais geral e utilizando-se o mesmo procedimento obtém-se :

$$a(k,n) = \sum_{m=-\infty}^{\infty} a(k-1,m)\frac{c(m-2n)}{2} \tag{2.47}$$

onde $a(k,n)$ são os coeficientes de expansão de $f_k(t)$ em termos das bases para V_k . A equação (2.47) especifica também que, com o conhecimento dos coeficientes $c(n)$ na equação (2.14), é possível obter-se os coeficientes $a(k,n)$ a partir das projeções no espaço seguinte de maior resolução ou aproximação.

Substituindo-se (2.25) em (2.47) pode-se chegar também à seguinte expressão:

$$a(k,n) = \sum_{m=-\infty}^{\infty} a(k-1,m)h(m-2n) \tag{2.48}$$

onde observa-se que $a(k,n)$ pode ser obtida também a partir da convolução das seqüências $a(k-1,n)$ e $h(n)$ seguido de um processo de sub-amostragem por um fator de dois.

O processo pode ser considerado como sendo uma filtragem passa-baixas já que, de acordo com (2.27) e (2.36), $H(w)$ satisfaz as seguintes condições:

$$H(0) = 1 \tag{2.49}$$

$$H(\pi) = 0 \tag{2.50}$$

Finalmente, a partir da equação (2.14) determina-se que:

$$c(n) = \int \phi(t)\phi(2t-n)dt = \langle \phi(t), \phi(2t-n) \rangle \tag{2.51}$$

Portanto:

$$h(n) = \frac{1}{2}\int \phi(t)\phi(2t-n)dt = \frac{1}{2}\langle \phi(t), \phi(2t-n)\rangle \tag{2.52}$$

De acordo com essa última expressão, pode-se concluir que a resposta impulsiva do filtro passa-baixas $h(n)$ pode ser determinada a partir da função de escala utilizada, a qual mostra a relação existente entre a filtragem e a análise de multiresolução.

2.3 MULTIRESOLUÇÃO E *WAVELET*S

Para se poder determinar a relação que existe entre uma análise de multiresolução e uma função *Wavelet* começa-se por definir:

$$g_k(t) = f_{k-1}(t) - f_k(t) \tag{2.53}$$

como sendo a **função detalhe** no nível de aproximação "k".

As funções $g_k(t)$ e $f_k(t)$ são ortogonais entre si devido a que pertencem a espaços de aproximação que são também ortogonais.

Da equação (2.53) tem-se que:

$$f_{k-1}(t) = f_k(t) + g_k(t) \tag{2.54}$$

o qual implica a existência de um espaço W_k que é o complemento de V_k em V_{k-1}.

Portanto pode-se dizer que:

$$V_{k-1} = V_k \oplus W_k \tag{2.55}$$

com :

$$V_k \perp W_k \tag{2.56}$$

onde o símbolo $\oplus$ representa a soma direta entre os espaços que formam a operação.

Por outro lado, temos que :

$$W_k \perp W_{k'} \Leftrightarrow k \neq k' \tag{2.57}$$

Os espaços W_k são denominados **espaços de detalhes** já que representam os detalhes perdidos ou ganhos entre dois níveis adjacentes de aproximação.

A partir de (2.55) e para um valor dado $k < K_0$ temos que:

$$V_k = V_{K_0} \oplus \bigoplus_{j=0}^{K_0 - k} W_{K_0 - j} \tag{2.58}$$

o qual do ponto de vista de tratamento de sinais especifica que os sinais cujo espectro se encontra localizado na banda de freqüências V_k são constituídos pela soma de sinais com banda de freqüências em V_{K_0} e de sinais com banda de freqüências em $W_{K_0}, W_{K_0-1} ... W_k$. Dessa forma, pode-se dizer que todos os subespaços envolvidos nessa soma são ortogonais [3].

A tarefa agora é encontrar uma maneira de se determinar as projeções de uma função $f(n)$ sobre os espaços de detalhes W_k. Para isso inicia-se pelo fato de que se existe uma função $f(t) \in W_0$ então $f(t) \in V_{-1}$ e $f(t) \perp V_0$. Portanto:

$$f(t) = \sum_n p(n) \phi(2t - n) \tag{2.59}$$

onde $p(n)$ constitui um seqüência de escalares para $n = 0, \pm 1, \pm 2, \ldots$.

Logo, tomando-se a transformada de *Fourier* em ambos os lados de (2.59) tem-se:

$$F(w) = \frac{1}{2} \sum_n p(n) e^{\frac{-jwn}{2}} \Phi(w/2) = P(w/2)\Phi(w/2) \tag{2.60}$$

onde :

$$P(w) = \frac{1}{2} \sum_n p(n) e^{-jwn} \tag{2.61}$$

constitui logicamente uma função periódica com período igual a 2π.

A condição (2.56) implica que $f(t) \perp \phi(t)$. Portanto:

$$\left\langle f(t), \phi(t-n) \right\rangle = 0 \tag{2.62}$$

No domínio da freqüência pode-se representar (2.62) como:

$$\int F(w)\Phi(w)e^{jwn}dw = 0 \tag{2.63}$$

ou

$$\int_0^{2\pi} e^{jwn} \sum_{\ell} F(w+2\pi\ell)\Phi\,(w+2\pi\ell)dw = 0 \tag{2.64}$$

Desde que :

$$\sum_{\ell} F(w+2\pi\ell)\Phi(w+2\pi\ell) = 0 \tag{2.65}$$

utilizando-se (2.29) e (2.60), e fazendo-se a mesma análise da seqüência (2.33)-(2.36) obtém-se:

$$P(w)\overline{H(w)} + P(w+\pi)\overline{H(w+\pi)} = 0 \tag{2.66}$$

Como uma conseqüência de (2.36), $H(w)$ e $\overline{H(w+\pi)}$ não podem ser ambos zero para um dado valor de "w". Portanto, $P(w)$ e $\overline{H(w+\pi)}$ devem ser linearmente dependentes; o que implica a existência de uma função "λ" com período igual a 2π que satisfaz as seguintes equações [2]:

$$P(w) = \lambda\,(w)\overline{m_0(w+\pi)} \tag{2.67}$$

$$\lambda\,(w) + \lambda\,(w+\pi) = 0 \tag{2.68}$$

Por outro lado, é conveniente introduzir uma função "v" com período igual a 2π que tenha a forma [2]:

$$\lambda\,(w) = e^{-jw}v(2w) \tag{2.69}$$

a fim de satisfazer (2.68).

Introduzindo-se (2.69) em (2.67) obtém-se:

$$P(w) = e^{-jw} v(2w) \overline{H(w+\pi)} \tag{2.70}$$

Logo, introduzindo-se (2.70) em (2.60) obtém-se:

$$F(w) = v(w) \underbrace{e^{\frac{-jw}{2}} H\left(\frac{w+2\pi}{2}\right) \Phi(w/2)}_{\Psi(w)} \tag{2.71}$$

$$F(w) = v(w)\Psi(w) \tag{2.72}$$

Desde que "v" tenha período igual a 2π pode-se expandir (2.72) em uma forma de série de *Fourier*:

$$F(w) = v(w)\Psi(w) = \sum_{n \in Z} v(n) e^{-jwn} \Psi(w) \tag{2.73}$$

Aplicando-se a transformada inversa de *Fourier* em ambos os lados de (2.73) obtém-se:

$$f(t) = \sum_{n \in Z} v(n) \psi(t-n) \tag{2.74}$$

onde a função $\psi(t)$ é finalmente a denominada **função wavelet .**

Assim, desde que $f(t) \in W_0$, justifica-se então que o conjunto $\{\psi(t-n) : n \in Z\}$ constitui uma base ortogonal para o espaço de funções W_0 [2].

A transforma de Fourier de $\psi(t)$ é dada por:

$$\Psi(w) = e^{\frac{-jw}{2}} \overline{H\left(\frac{w+2\pi}{2}\right)} \Phi(w/2) \tag{2.75}$$

Logo, introduzindo-se (2.30) em (2.75) e aplicando-se a transformada inversa de *Fourier*, tem-se:

$$\psi(t) = 2\sum_n (-1)^{n+1}\overline{h(-n-1)}\phi(2t-n) \tag{2.76}$$

Fazendo-se:

$$g(n) = (-1)^{n+1}\overline{h(-n-1)} \tag{2.77}$$

obtém-se :

$$\psi(t) = 2\sum_n g(n)\phi(2t-n) \tag{2.78}$$

onde observa-se que se $h(n)$ é um filtro **passa-baixas**, então $g(n)$ constitui um filtro do tipo **passa-altas**. Assim, também, pode-se dizer que a expressão (2.78) constitui uma equação de dilatação para $\phi(t)$.

A função *wavelet* $\psi(t)$ deverá satisfazer, por outro lado, as seguintes condições [1]:

$$\int_{-\infty}^{\infty}\psi(t)dt = 0 \tag{2.79}$$

$$\int_{-\infty}^{\infty}\left|\psi(t)\right|^2 dt = 1 \tag{2.80}$$

$$\left\langle\psi(t),\psi(t-n)\right\rangle = \delta(n) \tag{2.81}$$

$$\left\langle\psi(t),\phi(t-n)\right\rangle = 0 \tag{2.82}$$

A condição (2.79) deriva de (2.75) já que :

$$\Psi(0) = 0 \tag{2.83}$$

Além disso, tem-se :

$$H(\pi) = 0 \tag{2.84}$$

A condição (2.80) impõe energia unitária para a função $\psi(t)$, enquanto que condição (2.81) se justifica uma vez que $\{\psi(t-n) : n \in Z\}$ constitui uma base ortogonal para o espaço de funções

W_0 . Finalmente, a condição (2.82) deriva do fato de que $\psi(t) \in W_0$, $\phi(t) \in V_0$ e, portanto, $W_0 \perp V_0$ [1].

2.3.1 CONSEQÜÊNCIAS DA ORTOGONALIDADE DE UMA FUNÇÃO *WAVELET*

A partir das condições impostas na seção anterior, podem-se derivar várias propriedades que vão permitir entender melhor as características da transformada e sua implementação.

Inicia-se integrando ambos os lados de (2.78) e usa-se o fato de que a integral de $\psi(t)$ é igual a zero. Obtém-se:

$$\sum_{n=-\infty}^{\infty} g(n) = 0 \tag{2.85}$$

Dessa forma, multiplicando-se ambos os lados de (2.78) por $\psi(t-m)$ e utilizando-se as condições (2.7) e (2.81) obtém-se:

$$2\sum_{n=-\infty}^{\infty} g(n)g(n-2\ell) = \delta(\ell) \tag{2.86}$$

A equação (2.86) significa que a autocorrelação da seqüência $g(n)$ é zero para translações pares, exceto para o caso onde o deslocamento é igual a zero.

A condição (2.82) implica, por outro lado, que:

$$2\sum_{n=-\infty}^{\infty} g(n)h(n-2\ell) = 0 \tag{2.87}$$

A equação (2.87) especifica que a correlação cruzada entre $g(n)$ e $h(n)$ é zero para deslocamentos ou translações pares. As condições (2.19), (2.25), (2.86) e (2.87) implicam também que o conjunto de seqüências formadas pelos deslocamentos pares de $h(n)$ e $g(n)$ constituem um conjunto ortogonal.

Definindo-se :

$$G(w) = \sum_{n=-\infty}^{\infty} g(n) e^{-jwn} \tag{2.88}$$

pode-se dizer que no domínio da freqüência (2.86) eqüivale a:

$$\left|G(w)\right|^2 + \left|G(w+\pi)\right|^2 = 1 \tag{2.89}$$

Da equação (2.87) obtém-se:

$$H(w)G^*(w) + H(w+\pi)G^*(w+\pi) = 0 \tag{2.90}$$

Igualmente podem se derivar as expressões:

$$\left|H(w)\right|^2 + \left|G(w)\right|^2 = 1 \tag{2.91}$$

$$H^*(w)H(w+\pi) + G^*(w)G(w+\pi) = 0 \tag{2.92}$$

as quais podem ser escritas em forma de equação matricial :

$$\begin{bmatrix} H^*(w) & G^*(w) \\ H^*(w+\pi) & G^*(w+\pi) \end{bmatrix} \begin{bmatrix} H(w) & H(w+\pi) \\ G(w) & G(w+\pi) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.93}$$

sendo que (2.93) é conhecida também como a **condição paraunitária** [1]**.**

Tomando-se a transformada de Fourier inversa de (2.91) e (2.92), respectivamente, obtém-se por outro lado [1]:

$$\sum_{n=-\infty}^{\infty} h(n)h(n+\ell) + \sum_{n=-\infty}^{\infty} g(n)g(n+\ell) = \delta(\ell) \tag{2.94}$$

$$\sum_{n=-\infty}^{\infty} (-1)^n h(n)h(n+\ell) + \sum_{n=-\infty}^{\infty} (-1)^n g(n)g(n+\ell) = 0 \tag{2.95}$$

Pode-se observar em (2.94) e (2.95) que o deslocamento em π radianos no domínio da freqüência, equivale a multiplicar a correspondente seqüência no domínio do tempo por $(-1)^n$. Somando-se as equações (2.94) e (2.95) pode-se obter finalmente a condição:

$$\sum_{n=-\infty}^{\infty} h(2n)h(2n+\ell) + \sum_{n=-\infty}^{\infty} g(2n)g(2n+\ell) = \delta(\ell) \tag{2.96}$$

2.4 PROCESSOS DE FILTRAGEM: DECOMPOSIÇÃO E RECONSTRUÇÃO

Em seções anteriores foram detalhadas as condições que devem satisfazer os filtros *wavelet*s utilizados na decomposição de uma função em subespaços W_k . Determinou-se assim que os filtros utilizados nesse processo apresentam características passa-baixas e passa-altas, as quais satisfazem as condições de dilatação e ortogonalidade impostas pela teoria da análise de multiresolução.

Nesta parte do trabalho, por outro lado, serão detalhadas as características que descrevem o processo de reconstrução de um sinal, a partir dos sub-espaços de detalhes W_k. Será portanto realizado todo um equacionamento matemático a fim de se determinar os procedimentos seguidos para a recuperação da função ou do sinal original.

Enfoca-se primeiramente o caso onde $f_{-1}(t)$ é decomposta e logo reconstruída a partir das funções $f_0(t)$ e $g_0(t)$. Para isso inicia-se lembrando-se do fato de que os filtros de decomposição $H(w)$ e $G(w)$ satisfazem as condições de potência complementar (2.36) e (2.89), respectivamente. Isso implica que ambos os filtros apresentam um comportamento similar aos chamados filtros de quadratura QMF (*Quadrature Mirror Filter*), os quais são amplamente utilizados em diversas aplicações de processamento digital de sinais.

A resposta em freqüência que caracteriza esse tipo de filtros é mostrada na Fig. 2.1, onde se aprecia que a condição (2.36) é totalmente satisfeita por $H(w)$.

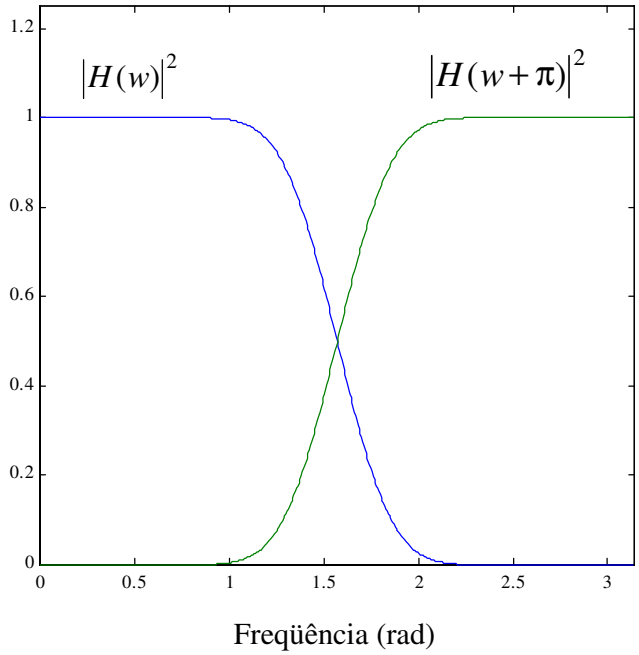


Fig. 2.1. Resposta em Freqüência de $|H(w)|^2$ e $|H(w+\pi)|^2$

Uma das propriedades mais importantes dos filtros QMF, no entanto, é a recuperação perfeita do sinal original a partir da utilização de filtros **PRQMF** *(Perfect Reconstruction Quadrature Mirror Filters*), os quais para esse objetivo deverão satisfazer certas condições .

Assume-se daqui para frente que os filtros utilizados são do tipo FIR de N coeficientes, os quais serão os únicos utilizados neste trabalho. De acordo com isso, a expressão (2.77) que define $g(n)$ pode ser escrita como:

$$g(n) = (-1)^{n+1} h(N-n-1) \tag{2.97}$$

Os filtros de reconstrução, por outro lado, são obtidos a partir das expressões determinadas em [5]. Nesse trabalho se encontrou que, para uma perfeita reconstrução de um sinal decomposto por filtros QMF, as respostas impulsivas dos filtros de reconstrução $\overline{h}(n)$ e $\overline{g}(n)$ deverão satisfazer as seguintes condições:

$$\overline{h}(n) = h(N-n-1) \tag{2.98}$$

$$\overline{g}(n) = (-1)^n h(n) \tag{2.99}$$

Observa-se que as expressões de ambos os filtros são dadas em função do filtro $h(n)$ com certas variações na fase e acrescentando-se os fatores multiplicativos $(-1)^{n+1}$ e $(-1)^n$. Isso permite assegurar que as condições da análise de multiresolução sejam também satisfeitas por $\overline{h}(n)$ e $\overline{g}(n)$.

Retornando à função $f_{-1}(t)$ decomposta em $f_0(t)$ e $g_0(t)$, tem-se que [1]:

$$f_{-1}(t) = \sum_{n=-\infty}^{\infty} a(0,n)\phi(t-n) + \sum_{n=-\infty}^{\infty} b(0,n)\psi(t-n) \tag{2.100}$$

onde $a(0,n)$ e $b(0,n)$ constituem uma seqüência de valores escalares.

Note-se que a primeira somatória de (2.100) é a função $f_0(t)$ da expressão (2.39), enquanto que a segunda constitui a função $g_0(t)$ definida como:

$$g_0(t) = \sum_{n=-\infty}^{\infty} b(0,n)\psi(t-n) \tag{2.101}$$

Pode-se observar também que a função $f_0(t)$ localizada no espaço V_0 pode ser expressa em função de $\phi(t)$ e suas translações inteiras. Da mesma forma, a função $g_0(t)$ localizada no subespaço W_0 pode ser expressa em função de $\psi(t)$ e também de suas translações inteiras.

O subespaço W_0 constitui a diferença entre os subespaços V_{-1} e V_0. Portanto, da mesma forma como foi obtida uma expressão para relacionar $a(0,n)$ e $a(-1,n)$, será possível também encontrar uma expressão que relacione $a(-1,n)$ e $b(0,n)$ [1].

Assim, tomando-se o produto interno em ambos os lados de (2.101) com respeito a $\psi(t-n)$ e utilizando-se o fato de que $\psi(t) \perp \phi(t)$ obtém-se:

$$\langle f_{-1}(t), \psi(t-n) \rangle = \langle g_0(t), \psi(t-n) \rangle \tag{2.102}$$

Da ortonormalidade de $\psi(t)$ e suas translações inteiras pode-se dizer que o lado direito de (2.102) é dado por:

$$\langle g_0(t), \psi(t-n) \rangle = b(0,n) \tag{2.103}$$

Da equação (2.78) tem-se que :

$$\langle \psi(t), \phi(2t-m) \rangle = 2 \sum_{n=-\infty}^{\infty} g(n) \langle \phi(2t-m), \phi(2t-n) \rangle \tag{2.104}$$

Usando-se a equação (2.10) no lado direito de (2.104) obtém-se:

$$\langle \psi(t), \phi(2t-m) \rangle = g(m) \tag{2.105}$$

Logo, da equação (2.38) tem-se que:

$$\langle f_{-1}(t), \psi(t-n) \rangle = \sum_{m=-\infty}^{\infty} a(-1,m) \langle \psi(t-n), \phi(2t-m) \rangle \tag{2.106}$$

Combinando-se (2.106) com (2.105) tem-se:

$$\langle f_{-1}(t), \psi(t-n) \rangle = \sum_{m=-\infty}^{\infty} a(-1,m)\, g(m-2n) \tag{2.107}$$

Substituindo-se as equações (2.103) e (2.107) em (2.102) chega-se a:

$$b(0,n) = \sum_{m=-\infty}^{\infty} a(-1,m)\, g(m-2n) \tag{2.108}$$

Para um caso mais geral tem-se finalmente que:

$$b(k,n) = \sum_{m=-\infty}^{\infty} a(k-1,m)\, g(m-2n) \tag{2.109}$$

onde pode-se observar a similaridade com o resultado obtido para a equação (2.48).

Conclui-se, portanto, o seguinte: a partir de um filtro passa-baixas dado por $h(n)$ e um filtro passa altas dado por $g(n)$, pode-se obter as coordenadas da projeção de qualquer função $f_{k-1}(t)$ (localizada no espaço V_{k-1}) nos subespaços V_k e W_k, respectivamente. A transformada de *Wavelet*s é baseada assim na expressão (2.58) onde uma determinada função localizada num

determinado espaço de aproximação V_k pode ser decomposta em subespaços de detalhes $W_{K_0}, W_{K_0-1} \ldots W_k$ mais um espaço de aproximação V_{K_0}. Estes subespaços constituem subbandas que armazenam distintas características da função ou sinal original, o qual é justificável uma vez que as mesmas são obtidas a partir de consecutivos processos de filtragem e subamostragens de acordo com o que especificam as expressões (2.48) e (2.109).

Para facilitar a representação desse processo, é utilizada geralmente uma árvore de decomposição ou reconstrução, dependendo do caso.

Da equação (2.100) e das equações de dilatação para $\phi(t)$ e $\psi(t)$ tem-se também que [1]:

$$f_{-1}(t) = 2\sum_{m=-\infty}^{\infty}\sum_{\ell=-\infty}^{\infty} a(0,\ell)h(m)\phi(2t-2\ell-m) + 2\sum_{m=-\infty}^{\infty}\sum_{\ell=-\infty}^{\infty} b(0,\ell)g(m)\psi(2t-2\ell-m) \tag{2.110}$$

Substituindo $n = 2\ell + m$ na equação (2.110) tem-se:

$$f_{-1}(t) = 2\sum_{n=-\infty}^{\infty}\sum_{\ell=-\infty}^{\infty} a(0,\ell)h(n-2\ell)\phi(2t-n) + 2\sum_{n=-\infty}^{\infty}\sum_{\ell=-\infty}^{\infty} b(0,\ell)g(n-2\ell)\psi(2t-n) \tag{2.111}$$

Logo, a partir de (2.38) e (2.111), obtém-se:

$$a(-1,n) = 2\sum_{\ell=-\infty}^{\infty} a(0,\ell)h(n-2\ell) + 2\sum_{\ell=-\infty}^{\infty} b(0,\ell)g(n-2\ell) \tag{2.112}$$

De acordo com as definições de $\overline{h}(n)$ e $\overline{g}(n)$ dadas em (2.98) e (2.99), respectivamente, pode-se dizer que:

$$a(-1,n) = 2\sum_{\ell=-\infty}^{\infty} a(0,\ell)\overline{h}(2\ell-n) + 2\sum_{\ell=-\infty}^{\infty} b(0,\ell)\overline{g}(2\ell-n) \tag{2.113}$$

A partir de (2.113) definem-se:

$$\overline{a}(0,n) \equiv 2\sum_{\ell=-\infty}^{\infty} a(0,\ell)\overline{h}(2\ell - n) \tag{2.114}$$

e:

$$\overline{b}(0,n) \equiv 2\sum_{\ell=-\infty}^{\infty} b(0,\ell)\overline{g}(2\ell - n) \tag{2.115}$$

onde $\overline{a}(0,n)$ e $\overline{b}(0,n)$ constituem as coordenadas reconstruídas de $f_{-1}(t)$ nos subespaços V_0 e W_0, respectivamente.

Considerando-se o lado direito de (2.114) e expandindo-se a somatória que conforma o mesmo, obtém-se:

$$\overline{a}(0,n) = ... + 2a(0,0)h(n) + 2a(0,1)h(2-n) + 2a(0,2)h(4-n).... \tag{2.116}$$

a qual pode ser reescrita como:

$$\overline{a}(0,n) = ... + 2a(0,0)h(n) + 0h(1-n) + 2a(0,1)h(2-n) + 0h(3-n) + 2a(0,2)h(4-n).... \tag{2.117}$$

A expressão (2.117) pode ser interpretada como uma seqüência de zeros inserida nas amostras adjacentes a $a(0,n)$ (**superamostragem** ou *upsampling*). O resultado do processo é em seguida convoluído com o filtro $\overline{h}(n)$ e multiplicado escalarmente por um fator de 2.

Esse procedimento (superamostragem mais filtragem) denomina-se também de interpolação de $a(0,n)$ a partir do filtro passa-baixas $2\overline{H}$ [1].

Igualmente, pode-se dizer que a seqüência $\overline{b}(0,n)$ é obtida interpolando-se $b(0,n)$ com o filtro passa-altas $2\overline{G}$.

Dessa forma, a seqüência de coeficientes usada para expandir $f_{-1}(t)$ em termos das bases para V_{-1}, pode ser obtida a partir das interpolações escalonadas de $\overline{a}(0,n)$ e $\overline{b}(0,n)$ geradas a partir dos filtros $\overline{H}$ e $\overline{G}$, respectivamente [1].

Para um caso mais geral, as equações (2.113), (2.114) e (2.115) podem ser generalizadas para qualquer inteiro "k":

$$a(k,n) \equiv \overline{a}(k+1,n) + \overline{b}(k+1,n) \tag{2.118}$$

onde:

$$\overline{a}(k+1,n) \equiv 2\sum_{\ell=-\infty}^{\infty} a(k+1,\ell)\overline{h}(2\ell - n) \tag{2.119}$$

$$\overline{b}(k+1,n) \equiv 2\sum_{\ell=-\infty}^{\infty} a(k+1,\ell)\overline{g}(2\ell - n) \tag{2.120}$$

Fazendo-se $\ell = \frac{m}{2}$ em (2.119) e (2.120) obtém-se :

$$\overline{a}(k+1,n) \equiv 2\sum_{m=-\infty}^{\infty} a(k+1,\frac{m}{2})\overline{h}(m - n) \tag{2.121}$$

$$\overline{b}(k+1,n) \equiv 2\sum_{\ell=-\infty}^{\infty} a(k+1,\frac{m}{2})\overline{g}(m - n) \tag{2.122}$$

onde :

$$a(k+1,\frac{m}{2}) = \begin{cases} 0 & \text{para } m \text{ impar} \\ a(k+1,\frac{m}{2}) & \text{para } m \text{ par} \end{cases} \tag{2.123}$$

A condição (2.123) constitui o processo de *upsampling* ou inserção de zeros que é aplicado a seqüência $a(k+1,\frac{m}{2})$ na etapa de reconstrução. Portanto, pode-se dizer que o processo de reconstrução tanto na parte passa-baixas quanto na parte passa-altas, é conformado por uma etapa de *upsampling* (por um fator de dois) seguida de uma etapa de filtragem com $\overline{H}$ e $\overline{G}$, respectivamente.

Por outro lado, muitos autores costumam associar os filtros de decomposição passa-baixas e passa-altas com $\overline{h}(n)$ e $\overline{g}(n)$. Esse fato será assumido daqui para frente neste trabalho. Isso implica em :

$$h_d(n) = \overline{h}(n) \tag{2.124}$$

$$g_d(n) = \overline{g}(n) \tag{2.125}$$

onde $h_d(n)$ e $g_d(n)$ constituem as respostas impulsivas que identificarão daqui para frente os filtros *wavelet*s utilizados no processo de decomposição.

Igualmente, os filtros de reconstrução identificados agora como $h_r(n)$ e $g_r(n)$ são definidos como:

$$h_r(n) = h(n) \tag{2.126}$$

$$g_r(n) = g(n) \tag{2.127}$$

De acordo com todo o equacionamento anterior e com as condições impostas para uma exata recuperação da função ou sinal original, os processos de decomposição e reconstrução para dois níveis consecutivos ($k-1$, k) são implementados da forma mostrada na Fig. 2.2.

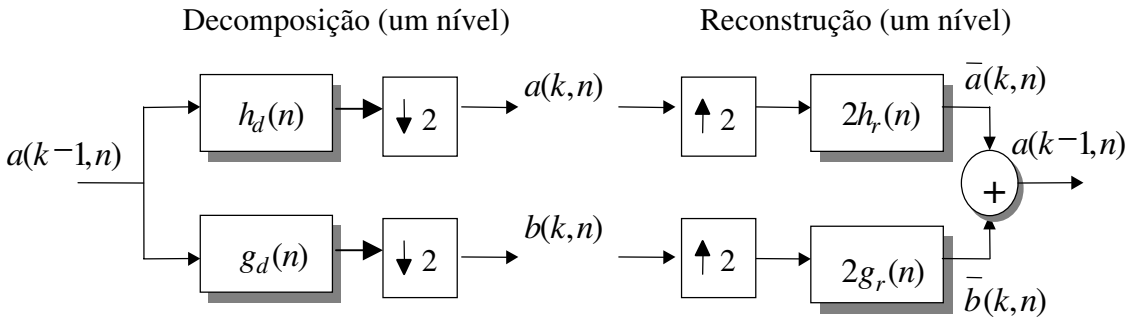


Fig. 2.2. Decomposição e Reconstrução (um nível).

2.4.1 TRANSFORMADA DISCRETA E SINAIS DISCRETOS

Geralmente, na prática, os processos de decomposição e reconstrução que conformam a transformada são implementados para serem aplicados a sinais discretos no domínio do tempo (como é o caso deste trabalho). Isso implica, portanto, na utilização da chamada Transformada

Discreta de *Wavelet*s (DWT) cujas características e relacionamento com a análise de multiresolução serão descritos nesta parte do trabalho.

Para começar, define-se primeiramente o que se denomina transformada contínua de *wavelet*s de uma função $f(t)$. Essa é dada por :

$$W(a,b) = \int_{-\infty}^{\infty} f(t) \underbrace{\frac{1}{\sqrt{|a|}} \psi^* \left(\frac{t-b}{a} \right)}_{\psi_{a,b}(t)} dt \tag{2.128}$$

onde a e b constituem números reais.

A transformada inversa, por outro lado, é definida como:

$$f(t) = \frac{1}{C} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{|a|^2} W(a,b) \psi_{a,b}(t) \, da \, db \tag{2.129}$$

onde :

$$C = \int_{-\infty}^{\infty} \frac{|\psi(w)|^2}{|w|} dw \tag{2.130}$$

tal que $0 < C < \infty$.

A expressão (2.130) é conhecida como **condição de admissibilidade**, a qual deverá ser satisfeita para a existência da transformada. Observe-se também, que desde que "a" e "b" constituam números reais, a dilatação ("a") e a translação ("b") da função *wavelet* são realizadas de forma contínua.

Para o caso discreto, (2.129) pode ser representada como:

$$f(t) = \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} d(k,\ell) 2^{-k/2} \psi(2^{-k} t - \ell) \tag{2.131}$$

onde $a = 2^k$, $k \in Z$ e $b = 2^k \ell$, $\ell \in Z$. Para esse caso a dilatação e a translação da *wavelet* é realizada em fatores e em múltiplos de 2^k, respectivamente. Portanto, desde que "k" constitua um valor inteiro, pode-se dizer que ambas as operações são feitas de forma discreta.

Para se poder diferençar entre DWT aplicada a sinais contínuos e a DWT aplicada a sinais discretos, chama-se a esta última de Transforma Discreta de *Wavelet*s em Tempo Discreto (DTWT)

[1]. Assim, para se determinar as características da DTWT e seu relacionamento com a análise de multiresolução, define-se primeiramente uma função contínua dada por :

$$\bar{f}(t) = \sum_{n=-\infty}^{\infty} f(n)\phi(t-n) \tag{2.132}$$

onde $f(n)$ constitui uma seqüência discreta no tempo, enquanto que $\phi(t)$ é a função de escala definida anteriormente para uma análise de multiresolução ortogonal.

Seja $\bar{f}_k(t)$ a projeção de $\bar{f}(t)$ no subespaço V_k. Portanto, definindo:

$$p(k,n) = 2^{-k}\left\langle \bar{f}(t), \phi(2^{-k}t - n)\right\rangle \tag{2.133}$$

tem-se:

$$\bar{f}_k(t) = \sum_{n=-\infty}^{\infty} p(k,n)\phi(2^{-k}t - n) \tag{2.134}$$

Particularmente:

$$f(n) = p(0,n) \tag{2.135}$$

implicando em que $\bar{f}(t) \in V_0$.

Dado que $.....V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2}.....$ pode-se dizer que:

$$\bar{f}(t) \in V_k \tag{2.136}$$

para $k < 0$.

Seja $\bar{g}_k(t)$ a ser a função de detalhes no nível "k", e $q(k,n)$ a serem os coeficientes de expansão em termos da *wavelet* dilatada e transladada (nessa parte $q(k,n)$ faz o papel de $b(k,n)$). Portanto, a seqüência $g(k,n)$ constitui a saída do filtro $\overline{G}$ e $p(k,n)$ é a seqüência de saída de $\overline{H}$. Além disso, $p(k,n)$, neste caso, cumpre o papel de $a(k,n)$.

Da equação (2.136) tem-se:

$$g_k(t) = 0 \tag{2.137}$$

para $k < 0$, o qual implica que:

$$q(k,n) = 0 \tag{2.138}$$

Dessa forma:

$$\bar{f}(t) = \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} q(k,\ell)\psi(2^{-k}t - \ell) = \sum_{k=1}^{\infty} \sum_{\ell=-\infty}^{\infty} q(k,\ell)\psi(2^{-k}t - \ell) \tag{2.139}$$

Observe-se que a DTWT é essencialmente constituída pelas seqüências $q(k,n)$ para $k > 0$. Isso implica em que a DTWT de $f(n)$ é a DWT de $\bar{f}(t)$.

Para interpretar a DTWT do ponto de vista de uma análise de multiresolução, inicia-se levando $\overline{p}(k,n)$ e $\overline{q}(k,n)$ a serem as saídas dos filtros $2H$ e $2G$, respectivamente. Substituindo-se $k = 0$ em (2.118), (2.119) e (2.120) , e fazendo-se uma apropriada troca de variáveis obtêm-se:

$$\overline{p}(1,n) \equiv 2\sum_{\ell=-\infty}^{\infty} p(1,\ell)h(n-2\ell) \tag{2.140}$$

$$\overline{q}(1,n) \equiv 2\sum_{\ell=-\infty}^{\infty} q(1,\ell)h(n-2\ell) \tag{2.141}$$

e:

$$f(n) = \overline{p}(0,n) = \overline{p}(1,n) + \overline{q}(1,n) \tag{2.142}$$

As seqüências $\overline{p}(1,n)$ e $\overline{q}(1,n)$ são ortogonais entre si desde que $h(n-2\ell)$ e $g(n-2\ell)$ sejam também ortogonais, de acordo com (2.87).

Suponha-se agora que uma função discreta $f(n)$ é decomposta e reconstruída a partir de K_0 níveis. Isso é mostrado na Fig. 2.3 (para este caso $K_0 = 3$) . Note-se que as seqüências de saída do processo de decomposição são conformadas por $q(1,n), q(2,n), ..., q(K_0,n)$, e $p(K_0,n)$. Assim, também observe-se que a seqüência $\overline{p}(K_0,n)$ poderá constituir o resultado do processo de reconstrução se as seqüências $q(1,n), q(2,n), ..., q(K_0,n)$ são forçadas a valores zero (Fig. 2.4). Essa seqüência constitui também a aproximação de mais baixa resolução do sinal no nível K_0.

Similarmente na Fig. 2.5 mostra-se que $\overline{q}(m,n)$ pode constituir a saída ou resultado final do processo de reconstrução se $\overline{p}(K_0,n)$ e as outras $\overline{q}(k,n)$ $(1 \le k \le K_0; k \ne m)$ são forçadas a valores iguais a zero. Diferentemente de $\overline{p}(K_0,n)$, a seqüência $\overline{q}(m,n)$ armazena detalhes ou características (não confundir com aproximação) da função $f(n)$ no nível "m".

Portanto, de acordo com as Figs. 2.4 e 2.5 pode-se dizer que o processo de reconstrução é uma operação linear. Isso implica em:

$$f(n) = \sum_{k=1}^{K_0} \overline{q}(k,n) + \overline{p}(K_0,n) \tag{2.143}$$

Levando $K_0 \to \infty$, tem-se:

$$f(n) = \sum_{k=1}^{\infty} \overline{q}(k,n) = \sum_{k=-\infty}^{\infty} \overline{q}(k,n) \tag{2.144}$$

onde:

$$\overline{q}(k,n) = 0 \tag{2.145}$$

para $k \le 0$.

A expressão (2.142), por outro lado, pode ser expressa de uma maneira mais geral da seguinte forma:

$$\overline{p}(k,n) = \overline{p}(k+1,n) + \overline{q}(k+1,n) \tag{2.146}$$

Em processos de compressão assumem-se as seqüências $q(k,n)$ e $p(K_0,n)$ como sendo as amostras das subbandas *wavelet*s resultantes da decomposição do sinal discreto original.

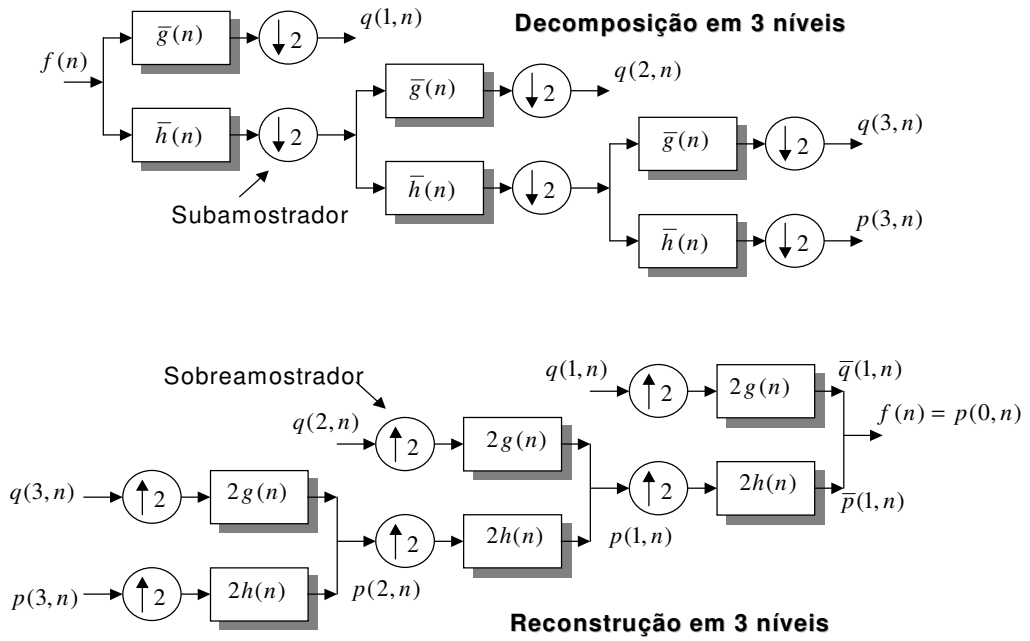


Fig. 2.3. Decomposição e reconstrução em 3 níveis.

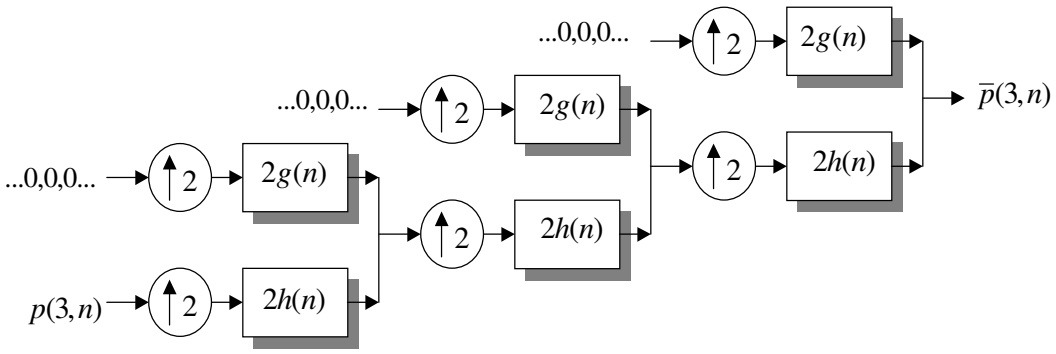


Fig. 2.4. Reconstrução da função de aproximação $\overline{p}(3,n)$.

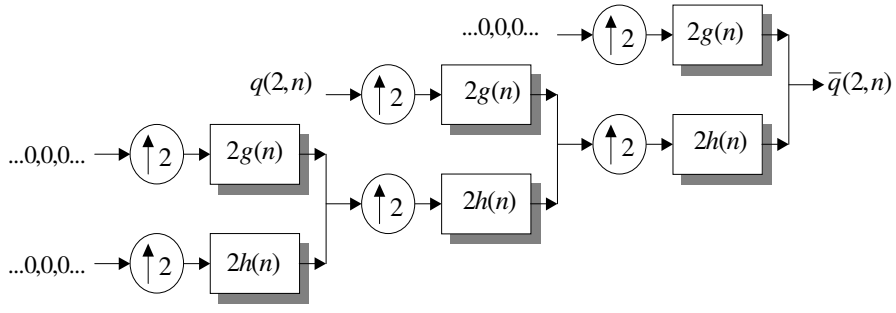


Fig. 2.5. Reconstrução da função de detalhes para $\overline{q}(2,n)$.

2.5 FAMÍLIAS DE *WAVELET*S

Existem diferentes tipos de famílias de *wavelet*s cujas características variam de acordo com vários critérios. Os principais são [4]:

- **O suporte de ψ e ϕ** : está relacionado com a rápida convergência para zero dessas funções no infinito, quando o tempo ou a freqüência tendem também ao infinito. Essa característica quantifica o desempenho das *wavelet*s no que se refere à localização tanto no tempo quanto na freqüência.

  As *wavelet*s utilizadas neste trabalho (Daubechies) existem ou são suportadas unicamente num intervalo de tempo $t1 \le t \le t2$, motivo pelo qual são chamadas geralmente de **wavelets compactamente suportadas no domínio do tempo**.

  Dessa forma, à medida que esse intervalo de suporte é incrementado, vai aumentando também o desempenho da *wavelet* no que se refere à localização de uma determinada subbanda no domínio da freqüência (resulta na geração de filtros mais seletivos).
- **Simetria** : é usada para evitar defasagens em processamento de imagens. *Wavelet*s simétricas dão origem a filtros de fase linear, os quais são necessários para esse tipo de aplicações.
- **O número de momentos nulos (vanishing moments) para ψ e ϕ** : esse resultado é utilizado para efeitos de determinação da seletividade dos filtros derivados dessas funções.

  Em processos de compressão, o número de momentos nulos de $H(w)$ na freqüência π fornece uma referência do grau de seletividade do filtro utilizado. Conforme mencionado anteriormente, os filtros com alto número de momentos nulos são atrativos para compressão de sinais de áudio. No entanto, isso implica em que o número de coeficientes dos filtros de decomposição e reconstrução também é incrementado. Esse fato pode comprometer o tempo de processamento do codificador para aplicações em tempo real.
- **Regularidade**: é utilizada para adquirir características finas como a uniformidade do sinal ou da imagem reconstruída. A regularidade, por exemplo, de um processo de decomposição é assegurada desde que $\psi(t)$ tenha um considerável número de momentos nulos. Portanto, pode-se demonstrar que se a condição [11]:

$$\sum_n (-1)^n n^j h(n) = 0 \qquad j = 0, 1, 2, \ldots p-1 \tag{2.147}$$

é satisfeita, então a *wavelet* $\psi(t)$ pertinente deverá conter "p" *momentos nulos*. Isso implica em que $H(w)$ deverá apresentar um zero de ordem "p" em $w = \pi$.

Assim, quanto mais regular for uma *wavelet*, maior será a seletividade dos filtros derivados da mesma. Esse fato para efeitos de compressão de sinais de áudio é mais conveniente.

A título de exemplo, na Fig. 2.6. apresenta-se a variação dessa seletividade em função do número de momentos nulos .

Note-se que à medida que "p" aumenta, a resposta em freqüência do filtro torna-se mais seletiva. Isso beneficia o processo de compressão devido ao fato de que a distorção introduzida pela quantização de uma subbanda não atinge considerávelmente as subbandas vizinhas.

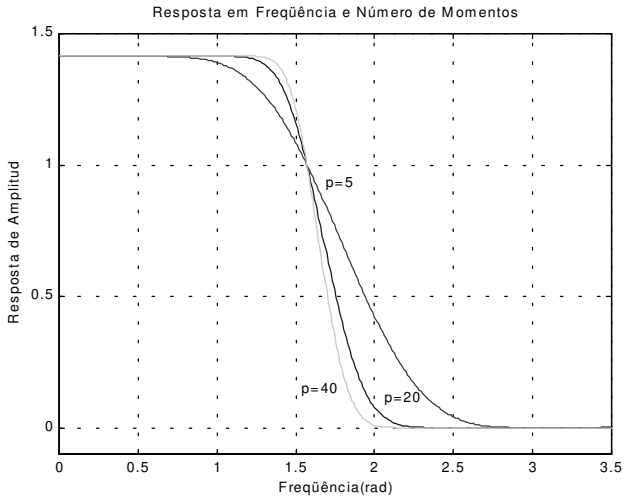


Fig. 2.6. Resposta em Freqüência de $H(w)$ em função do número de momentos nulos.

Entre as famílias de *wavelet*s mais utilizadas tem-se: a **Morlet**, **Chapéu Mexicano**, **Meyer**, **Haar**, **Daubechies** (utilizada neste trabalho) **Coiflets** e as chamadas **Wavelets Spline Biortogonais** utilizadas principalmente em processamento de imagens dado que os filtros em questão apresentam fase linear.

Neste trabalho utiliza-se exclusivamente *Wavelet*s de Daubechies com diferentes números de momentos nulos devido a que as mesmas encontram-se amplamente disponíveis no MATLAB 5.0. Portanto, daqui para frente será utilizada a nomenclatura dbN_{vm} para se identificar uma Daubechies (db) de N_{vm} momentos nulos.

Esse valor está também diretamente relacionado com o número de coeficientes N que conformam a resposta impulsiva dos filtros. Assim, tem-se que:

$$N = 2N_{vm} \tag{2.148}$$

Em processos de compressão de áudio são utilizados comumente filtros com um número de momentos nulos na faixa 10 (db25) a 60 (db60). Esses últimos são recomendáveis para taxas menores do que 80kbit/s.

Nas Figs. (2.7), (2.8) e (2.9) apresentam-se as formas das *wavelet*s db4, db10 e db20, assim como as funções de escala e os filtros associados. Note-se que à medida que o número de momentos aumenta, a *wavelet* pertinente torna-se mais regular e apresenta também maior suporte no domínio temporal (ou vice-versa).

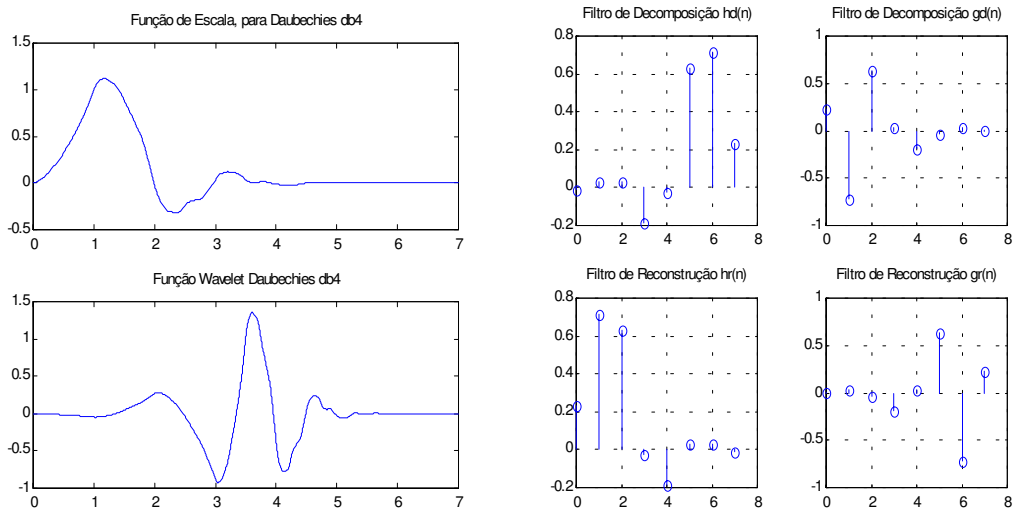


Fig. 2.7. Função Wavelet db4, Função de Escala e Filtros associados.

Como exemplo, seja a função :

$$f(n) = A_1 \cos(w_1 n) + A_2 \cos(w_2 n) \tag{2.149}$$

a qual, para efeitos de implementação, se assumirá que está localizada no espaço de aproximação V_0, sendo que isso é justificável dado que no mundo real não existem espaços ou subespaços negativos.

Com $A_1 = 8.75$, $w_1 = 0.1425$ (o que corresponde a um tom de 1KHz, para uma freqüência de amostragem de 44.1 KHz), $A_2 = 6.75$ e $w_2 = 1.8641$ (o que corresponde a um tom de 13KHz, para uma freqüência de amostragem de 44.1 KHz) , tem-se a função $f(n)$ cuja forma é mostrada

na Fig. 2.10 (desenhada em forma contínua) . Decompondo-se essa função através da árvore mostrada na Fig. 2.3 e utilizando-se a Daubechies db4, obtém-se as funções $q(1,n)$, $q(2,n)$, $q(3,n)$ e $p(3,n)$, as quais são mostradas na Fig. 2.10 e constituem as projeções de $f(n)$ nos subespaços W_1, W_2 , W_3 , e V_3, respectivamente.

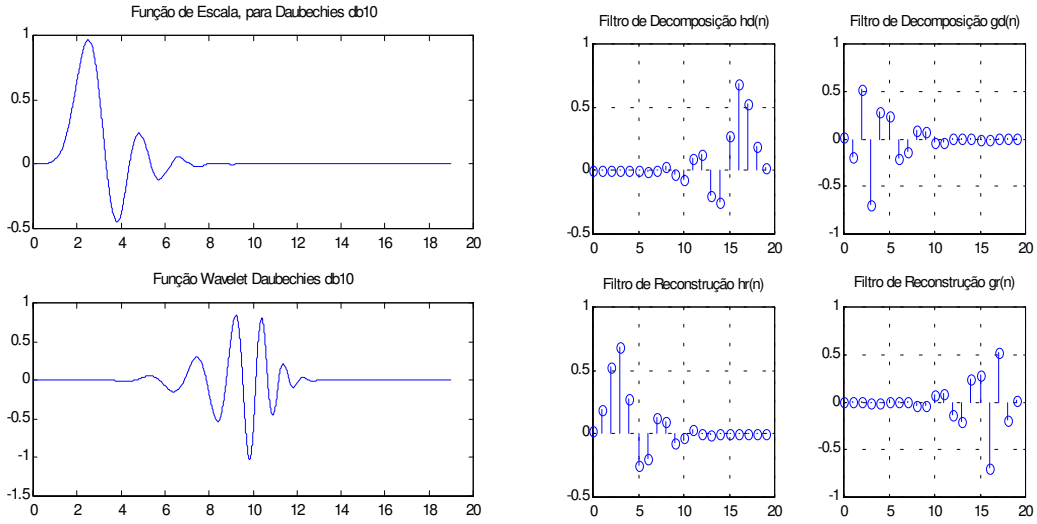


Fig. 2.8. Função wavelet db10, função de escala e filtros associados.

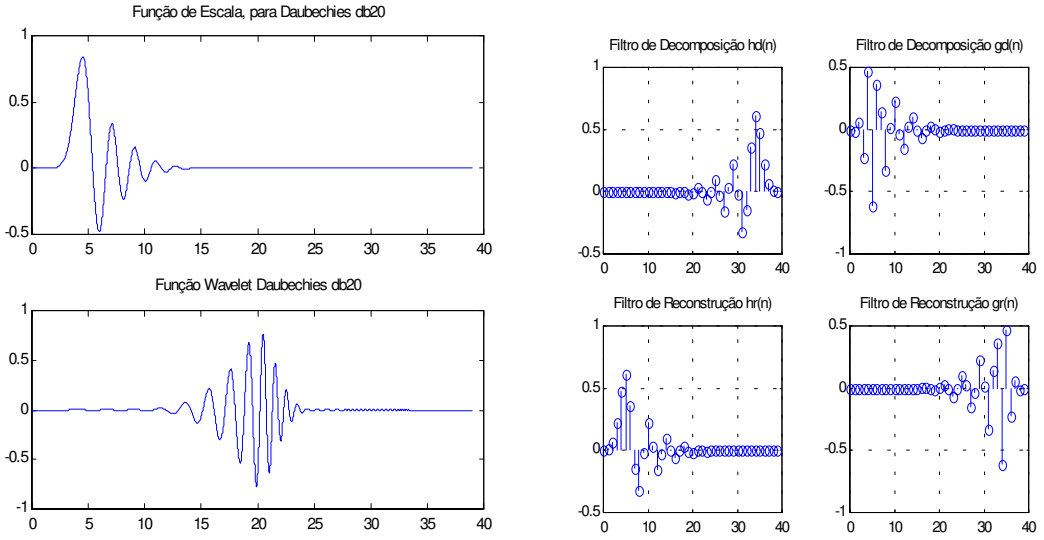


Fig. 2.9. Função wavelet db20, função de escala e filtros associados.

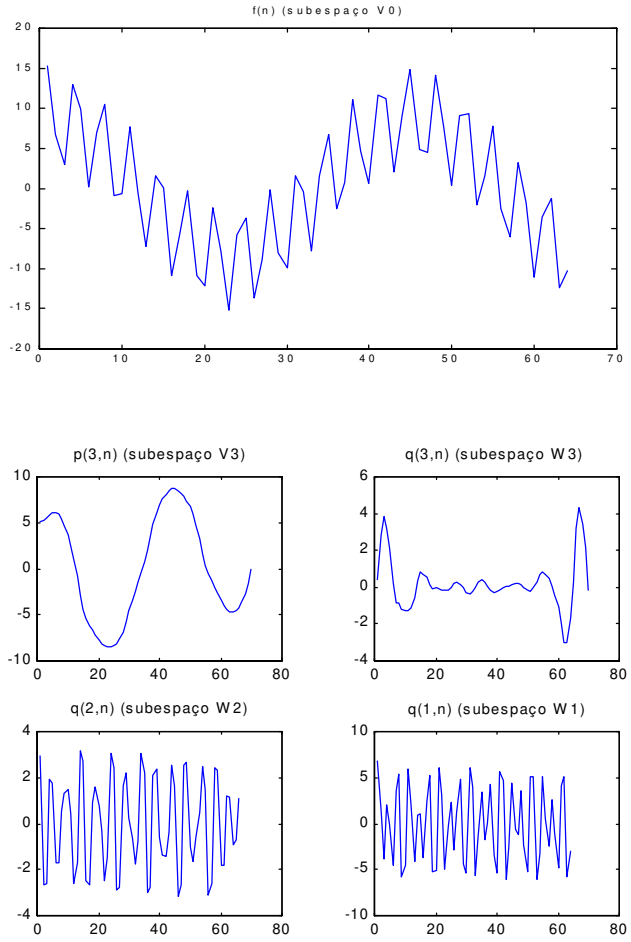


Fig. 2.10. Função Discreta $f(n)$ e suas projeções. nos subespaços V_3 , W_3 , W_2 e W_1.

Note-se que a função resultante sobre o subespaço V_3 constitui uma aproximação da forma da função original (subespaço de aproximação) enquanto que as funções nos outros subespaços constituem detalhes localizados em diferentes bandas de freqüências (subespaço de detalhes), o que é razoável.

2.6 *WAVELET PACKETS*

Em seções anteriores foi mostrado que o processo de decomposição em subbandas *wavelet*s é realizado iterativamente decompondo-se a seqüência de saída dos filtros passa-baixas até se chegar

a um nível de decomposição desejado. Dessa forma, os resultados eram conformados pelas projeções da função de entrada nos diferentes espaços de detalhes localizados em cada nível de decomposição.

Com o passar do tempo apareceu o desejo de se gerar formatos secundários de decomposição que impliquem não só na divisão da parte passa-baixas, senão também na decomposição da seqüência de saída dos filtros passa-altas.

Isso levou ao conceito da chamada Transformada de *Wavelet Packets*, a qual apresenta uma árvore de decomposição bastante flexível que facilmente pode-se adaptar a qualquer tipo de aplicação.

As características e princípios dessa transformada serão descritas nesta parte do trabalho sendo que o equacionamento matemático apresentado será baseado em [1].

Inicia-se considerando-se o processo de decomposição de dois níveis que é mostrado na Fig. 2.11.

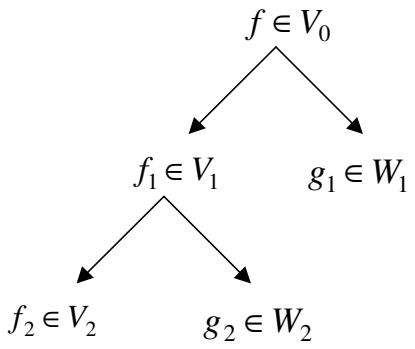


Fig. 2.11. Árvore de decomposição de dois níveis.

Logo, observa-se que as funções resultantes desse processo $f_2(t)$, $g_2(t)$ e $g_1(t)$ podem ser expressas como:

$$g_1(t) = \sum_{n=-\infty}^{\infty} b(1,n)\psi(t/2-n) \tag{2.150}$$

$$g_2(t) = \sum_{n=-\infty}^{\infty} b(2,n)\psi(t/4-n) \tag{2.151}$$

$$f_2(t) = \sum_{n=-\infty}^{\infty} a(2,n)\phi(t/4-n) \tag{2.152}$$

e portanto:

$$f(t) = f_0(t) = f_2(t) + g_2(t) + g_1(t) \tag{2.153}$$

Define-se:

$$p_{1,1,n}(t) = \frac{1}{2}\psi(t/2-n) \tag{2.154}$$

$$p_{2,0,n}(t) = \frac{1}{4}\phi(t/4-n) \tag{2.155}$$

$$p_{2,1,n}(t) = \frac{1}{4}\psi(t/4-n) \tag{2.156}$$

onde as funções $p_{1,1,n}(t)$, $p_{2,0,n}(t)$ e $p_{2,1,n}(t)$ conformam um conjunto ortogonal. Assim, as equações (2.153), (2.154), (2.155) e (2.156) indicam também que essas funções formam uma base para V_0. A tarefa agora será colocar as expressões anteriores em termos de $\phi(t)$. Para isso, inicia-se com a equação de dilatação definida como:

$$\phi(t) = 2\sum_{n=-\infty}^{\infty} h(n)\phi(2t-n) \tag{2.157}$$

Substituindo-se t por $2t$ na equação (2.157) obtém-se:

$$\phi(2t) = 2\sum_{n=-\infty}^{\infty} h(n)\phi(4t-n) \tag{2.158}$$

Colocando-se a equação (2.158) na equação (2.157) tem-se:

$$\phi(t) = 4\sum_{n}\sum_{m} h(n)h(m)\phi(4t-2n-m) \tag{2.159}$$

Substituindo-se t por $t/4$ na equação (2.159) obtém-se:

$$p_{2,0,0}(t) = \frac{1}{4}\phi(t/4) = \sum_n \sum_m h(n)h(m)\phi(t-2n-m) \tag{2.160}$$

Por outro lado, a função de escala para uma *wavelet* é definida como:

$$\psi(t) = 2\sum_{n=-\infty}^{\infty} g(n)\phi(2t-n) \tag{2.161}$$

Utilizando-se a equação (2.158) em (2.161) tem-se:

$$\psi(t) = 4\sum_n \sum_m g(n)h(m)\phi(4t-2n-m) \tag{2.162}$$

Logo, tem-se que:

$$p_{2,1,0}(t) = \frac{1}{4}\psi(t/4) = \sum_n \sum_m g(n)h(m)\phi(t-2n-m) \tag{2.163}$$

Além disso, tem-se também:

$$p_{1,1,0}(t) = \frac{1}{4}\psi(t/4) = \sum_{n=-\infty}^{\infty} g(n)\phi(t-n) \tag{2.164}$$

Para examinar $p_{2,0,0}(t)$ e $p_{2,1,0}(t)$ fazemos $\ell = 2n+m$ nas equações (2.160) e (2.163) :

$$p_{2,0,0}(t) = \sum_n \sum_\ell h(n)h(\ell-2n)\phi(t-\ell) = \sum_\ell A(\ell)\phi(t-\ell) \tag{2.165}$$

$$p_{2,1,0}(t) = \sum_n \sum_\ell g(n)h(\ell-2n)\phi(t-\ell) = \sum_\ell B(\ell)\phi(t-\ell) \tag{2.166}$$

onde:

$$A(\ell) = \sum_n h(n)h(\ell - 2n) \tag{2.167}$$

$$B(\ell) = \sum_n g(n)h(\ell - 2n) \tag{2.168}$$

Observe-se que a seqüência $A(\ell)$ é obtida superamostrando-se primeiro $h(n)$ por um fator de 2 e convoluenndo-se o resultado novamente com $h(n)$. Igualmente, a seqüência $B(\ell)$ é obtida superamostrando-se $h(n)$ por um fator de 2 e convoluendo-se em seguida o resultado com $g(n)$ [1].

O mesmo procedimento pode ser seguido para se colocar as outras funções em termos de $\phi(t)$. Para isso, definem-se:

$$C(\ell) = \sum_n h(n)g(\ell - 2n) \tag{2.169}$$

$$D(\ell) = \sum_n g(n)g(\ell - 2n) \tag{2.170}$$

e:

$$p_{2,2,0}(t) = \sum_\ell C(\ell)\phi(t - \ell) \tag{2.171}$$

$$p_{2,3,0}(t) = \sum_\ell D(\ell)\phi(t - \ell) \tag{2.172}$$

$$p_{2,2,n}(t) = p_{2,2,0}(t - 4n) \tag{2.173}$$

$$p_{2,3,n}(t) = p_{2,3,0}(t - 4n) \tag{2.174}$$

Dessa forma, $p_{2,i,n}(t), i = 0, 1, 2, 3,$ formam uma base ortonormal para V_0. Observe-se também que $p_{1,1,n}(t)$, $p_{2,0,n}(t)$ e $p_{2,1,n}(t)$ constituem bases para W_1, V_2 e W_2, respectivamente. Assim, projetando-se $f_0(t)$ sobre esses subespaços geram-se duas funções dadas por:

$$g_{20} = \sum_n c(2,n)\, p_{2,2,n}(t) \tag{2.175}$$

$$g_{21} = \sum_n d(2,n)\, p_{2,3,n}(t) \tag{2.176}$$

onde as seqüências $c(2,n)$ e $d(2,n)$ são obtidas a partir da dizimação de $b(1,n)$ com a utilização dos filtros $\overline{H}$ e $\overline{G}$, respectivamente. Esse processo é realizado da mesma maneira para $a(2,n)$ e $b(2,n)$ que são obtidas de $a(1,n)$.

Isso demonstra que a partir da decomposição de W_1 pode-se gerar iterativamente diversos subespaços e formatos de decomposição de acordo com o tipo de aplicação na qual será utilizada a árvore. As funções bases resultantes desse processo são as chamadas **Wavelet Packets**.

Na Fig. 2.12 mostra-se um formato de decomposição *wavelet packets* de três níveis. Denomina-se a esse tipo de árvore de **formato de decomposição balanceado**, já que todos os subespaços de todos os níveis são decompostos, exceto os subespaços resultantes.

Alternativamente, pode-se ter outros formatos de decomposição como aquele que é mostrado na Fig. 2.13. Nesse caso, note-se que a árvore mostrada é do tipo **não balanceada**. Tais árvores são utilizadas em aplicações onde deseja-se obter subbandas de largura variável como é o caso do esquema de bandas críticas do ouvido humano.

No codificador proposto será utilizado esse tipo de árvore, cujas características serão detalhadas no Capítulo 4.

Finalmente, como exemplo, mostra-se na Fig. 2.14 a forma das funções *wavelet packets* Daubechies db2, que correspondem aos subespaços resultantes do esquema de decomposição da Fig. 2.12.

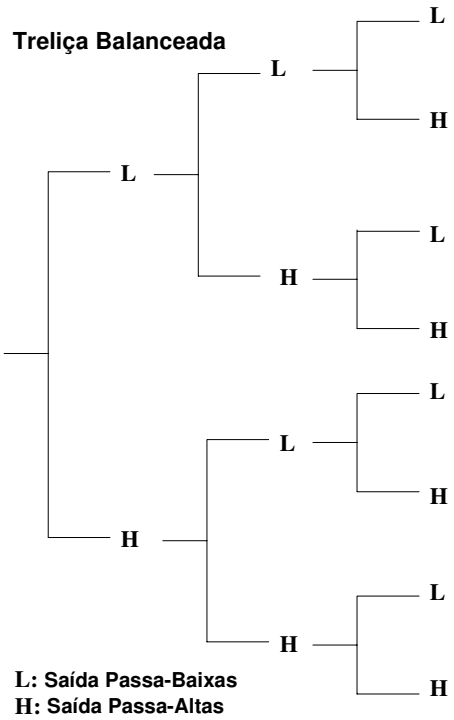


Fig. 2.12. Árvore de decomposição balanceada de 3 níveis.

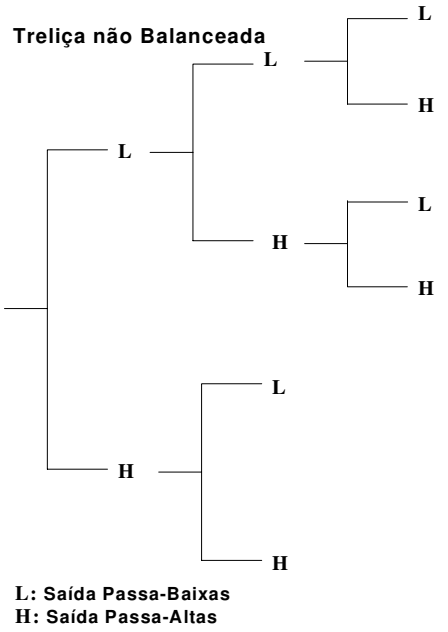


Fig. 2.13. Árvore de decomposição não balanceada.

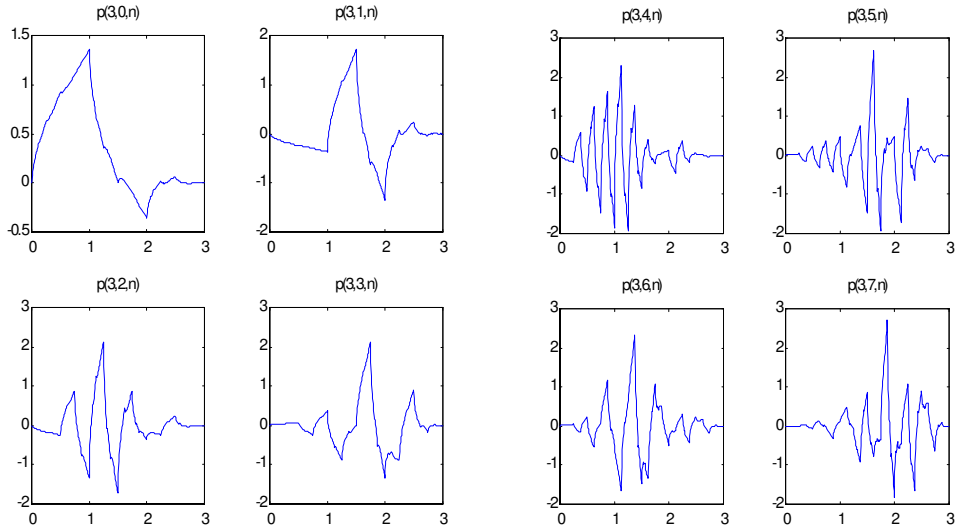


Fig. 2.14. ***Wavelets Packets*** **db2 para um formato de decomposição de 3 níveis.**

2.7 COMENTÁRIOS FINAIS

O enfoque da transformada de *wavelet*s através da teoria da análise de multiresolução tem possibilitado entender o efeito dessa transformada sobre os sinais que serão submetidos aos processos de compressão. Para o caso de áudio, esse efeito se traduz na decomposição de sinais em subbandas que serão quantizadas pelo codificador de acordo com o nível de intensidade que é percebido pelo ouvido humano.

O formato de decomposição que será utilizado para esse caso será apresentado no Capítulo 4 e corresponde a uma árvore *wavelet packets* que, conforme mencionado anteriormente, adapta-se flexivelmente às bandas críticas do nosso sistema de audição.

CAPÍTULO 3

TÉCNICAS DE COMPRESSÃO: MODELO PSICO-ACÚSTICO E QUANTIZAÇÃO VETORIAL

3.1 INTRODUÇÃO

A utilização de técnicas e ferramentas de compressão cada vez mais eficientes tem sido uma das principais preocupações a serem levadas em conta no desenvolvimento de novos codificadores. Esse fato tem possibilitado uma rápida evolução de algoritmos e técnicas propostas que hoje em dia fazem parte dos principais codificadores de sinais de som.

As primeiras técnicas de compressão de áudio digital apareceram aproximadamente no começo dos anos 80 e foram baseadas na codificação do sinal no domínio do tempo. Tais técnicas não atingiram os resultados esperados devido à pouca estacionaridade que apresentam esse tipo de sinais no domínio temporal. Isso impedia que técnicas de compressão como a DPCM (*Diferential Pulse Code Modulation*) ou a DM (*Delta Modulation*) apresentassem um melhor desempenho no que se refere ao fator de qualidade por taxa de bits.

Com o passar dos anos apareceu a idéia de se codificar os sinais no domínio da freqüência utilizando-se para esse caso os chamados modelos psico-acústicos do sistema de audição humano. Esses modelos exploram o fenômeno de mascaramento do ouvido e, portanto, permitem identificar aproximadamente as componentes de freqüência audíveis ou pouco audíveis que fazem parte do espectro auditivo. A partir dessa discriminação é possível a determinação da informação que deverá ser posteriormente quantizada, codificada e transmitida (audível), obtendo-se

logicamente um ganho considerável de compressão. A desvantagem da utilização desse formato de codificação encontra-se, no entanto, no fato de que aumenta-se muito a complexidade dos codificadores.

Hoje em dia, com o avanço da tecnologia de circuitos integrados de alta velocidade e com o desenvolvimento de algoritmos cada vez mais eficientes, é possível implementar esse tipo de codificadores para aplicações em tempo real (codificadores Dolby AC-2, Dolby AC-3, MPEG-1, MPEG-2, etc), os quais já estão sendo utilizados nos diferentes sistemas de telecomunicações para aplicações em radiodifusão (*broadcasting*), armazenamento e transmissões via internet.

O diagrama de blocos do esquema de codificação no domínio da freqüência ou da transformada é mostrado na Fig.3.1. Note-se que o mesmo é formado por etapas que cumprem diferentes funções : segmentação e janelamento da seqüência de amostras de entrada, Transformada (DCT, DFT, DST, Wavelets, etc), Modelo Psico-Acústico, Quantização (Escalar, Vetorial, etc) e Codificação de Dados (Ex. : codificação de entropia).

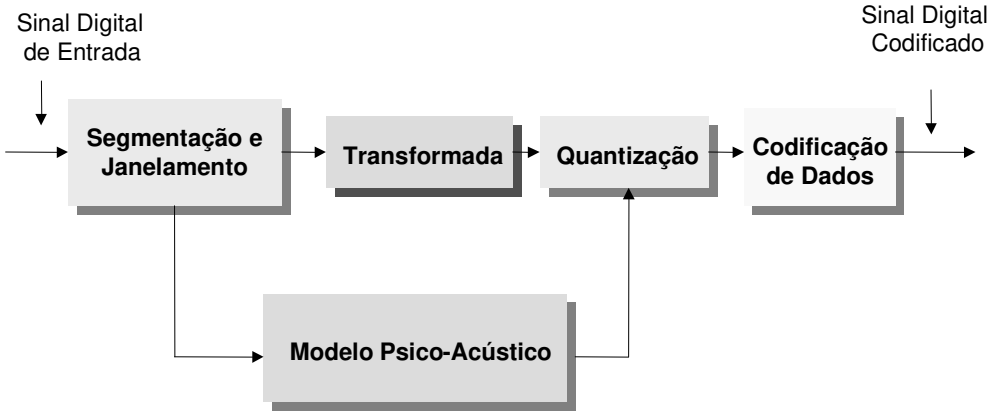


Fig. 3.1 Esquema de codificação de áudio no domínio da transformada.

Sob esse formato de codificação é baseado também o codificador de áudio proposto neste trabalho. As etapas de codificação que formam o mesmo serão descritas, no entanto, com maior detalhe no Capítulo 4.

No presente capítulo serão detalhadas as características que descrevem o algoritmo do modelo psico-acústico MP2, que será utilizado parcialmente neste trabalho. Assim, serão detalhados também os aspectos teóricos que descrevem a técnica de quantização vetorial e os procedimentos utilizados para se gerar os "*codebooks*" pertinentes.

A quantização vetorial constitui, para muitos autores, um dos formatos de codificação que será mais utilizado no futuro devido a que, em muitos casos, ela atinge altas taxas de compressão.

Atualmente, têm sido publicados diversos métodos de utilização da quantização vetorial de acordo com os diferentes tipos de aplicações. Neste capítulo, propõe-se um método de construção de quantizadores vetoriais adaptado aos requerimentos do codificador proposto.

Os aspectos introdutórios ao modelo psicoacústico e à quantização vetorial a serem detalhados neste capítulo, tem por outro lado, o objetivo de colocar os conhecimentos básicos que permitam entender posteriormente a forma como os mesmos serão aplicados no codificador de áudio proposto.

3.2 FATORES PSICO-ACÚSTICOS

Os fatores Psico-Acústicos definem características e limitações da audição humana que são exploradas por sistemas de compressão de sinais de alta qualidade.

O primeiro fator a ser levado em conta é a **sensibilidade variável do ouvido frente às componentes de freqüência** que formam o espectro auditivo. Essa característica poderá ser entendida melhor através de um exemplo que é descrito a seguir.

Considere-se um oscilador que possa gerar um sinal exatamente senoidal (tom) para ser reproduzido através de um amplificador e uma caixa de som com nível constante de volume. Observa-se que à medida que se varia apenas a freqüência do oscilador, a percepção do ouvido para os distintos tons gerados é diferente, sendo que algumas freqüências são percebidas com maior intensidade do que outras. O resultado desse teste resulta em uma família de curvas. Tais curvas são apresentadas na Fig. 3.2 [6]. Cada curva representa a variação do nível de volume do amplificador (em dB) a fim de que todas as componentes do espectro sejam percebidas com igual intensidade. As curvas geradas para esse caso foram de intensidade 0 (limiar de silêncio), 10, 20, 30, 40, 50, 60, 70, 80, 90 100, 110 e 120 dB.

O **maior nível de percepção** apresenta-se na faixa de **1 a 5 kHz**.

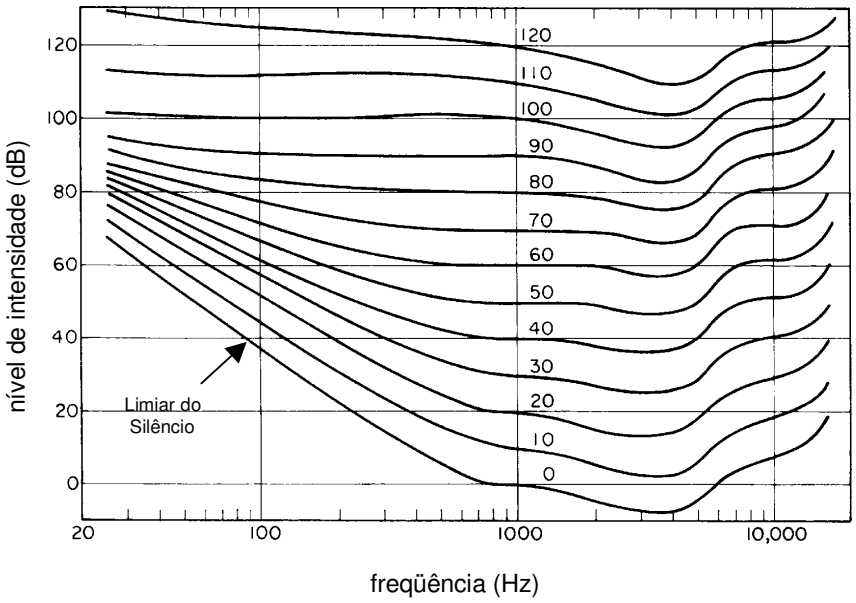


Fig. 3.2. Percepção do ouvido humano em função da Freqüência [6].

Outro fator ou característica importante é o chamado **fenômeno de mascaramento** (*masking*), o qual se apresenta quando componentes de alta potência mascaram ou tornam quase inaudíveis componentes vizinhas de menor nível de potência reproduzidas simultaneamente. Essa característica constitui um beneficio direto para os processos de compressão, já que componentes ou grupo de componentes que são fortemente mascaradas são codificados com poucos bits.

O mascaramento pode ser devido a um tom isolado de alta potência (mascaramento tonal) ou devido a um grupo de componentes vizinhas com nível regular de potência (mascaramento não tonal).

Os modelos psico-acústicos são, portanto, baseados em algoritmos de identificação de componentes tonais e não tonais, assim como também em procedimentos de cálculo dos níveis de mascaramento.

Para se poder entender melhor esse fenômeno utiliza-se um exemplo baseado na Fig. 3.3 [6] e que é descrito a seguir.

Considere-se dois tons de áudio que são reproduzidos simultaneamente: um tom primário de 1200 Hz com intensidade de 80dB acima do limiar de silêncio e um tom secundário com freqüência e intensidade variáveis. Observa-se que, para uma dada freqüência do tom secundário, a intensidade do mesmo deverá ser maior do que certo limiar, a fim de que não seja mascarado pelo tom primário reproduzido simultaneamente. A variação desse limiar em função da freqüência é o que mostra a curva da Fig. 3.3. Para valores de intensidade abaixo da curva de mascaramento, o tom secundário é

inaudível. A curva é chamada geralmente de função de espalhamento ou *spreading funtion*. Os modelos psico-acústicos calculam essa função para todas as componentes de freqüência em cada instante de codificação.

Considerando-se um terceiro tom de alta intensidade a ser reproduzido simultaneamente com os outros dois tons, então o nível de mascaramento do tom secundário será concebido como sendo a soma dos efeitos produzidos pelo primeiro e pelo terceiro tom.

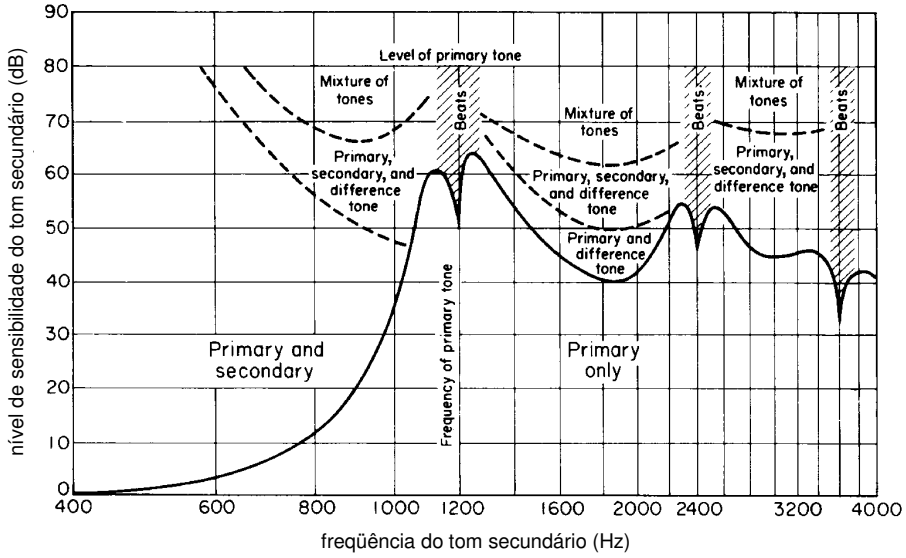


Fig. 3.3. Percepção do ouvido humano frente a dois tons reproduzidos simultaneamente [6].

Finalmente, outra característica muito importante é a divisão do espectro auditivo (por parte do ouvido) em bandas de largura variável, as quais são chamadas de bandas críticas do ouvido humano.

Essa forma de divisão constitui a máxima resolução que o ouvido apresenta para detectar componentes ou sons que fazem parte do espectro. Dessa forma, pode-se comprovar também que a sensibilidade do ouvido é variável para cada banda crítica, sendo que, geralmente, assume-se que as componentes de freqüência que formam cada uma delas são percebidas com igual intensidade.

Os modelos psico-acústicos geralmente fazem os cálculos de mascaramento levando-se em conta as bandas críticas a fim de também diminuir a carga computacional do codificador.

Existem vários formatos e esquemas de bandas críticas. Um deles é aquele mostrado na Tabela 3.1 e que é considerado nos sistemas MP2 [7] para uma freqüência de amostragem de 44.1KHz.

Observa-se que as bandas correspondentes às altas freqüências são de maior largura do que aquelas que correspondem às baixas freqüências (Fig. 3.4). Isso implica, portanto, em uma maior resolução do ouvido para as componentes que formam os chamados sons graves e médios.

Finalmente, pelo fato de existir uma relação não linear entre o esquema de bandas críticas e a escala de freqüências, utiliza-se muitas vezes a escala logarítmica Bark, a qual, segundo mostra a Tabela 3.1, apresenta uma relação quase linear com o espectro das bandas críticas.

Existem várias formas de conversão da escala de freqüências para a escala Bark. Uma delas é a proposta por Schroeder [8] e é dada por:

$$z / Bark = 7\text{arsinh}\left(\frac{f.}{650}\right) \tag{3.1}$$

onde f representa o valor da freqüência pertinente.

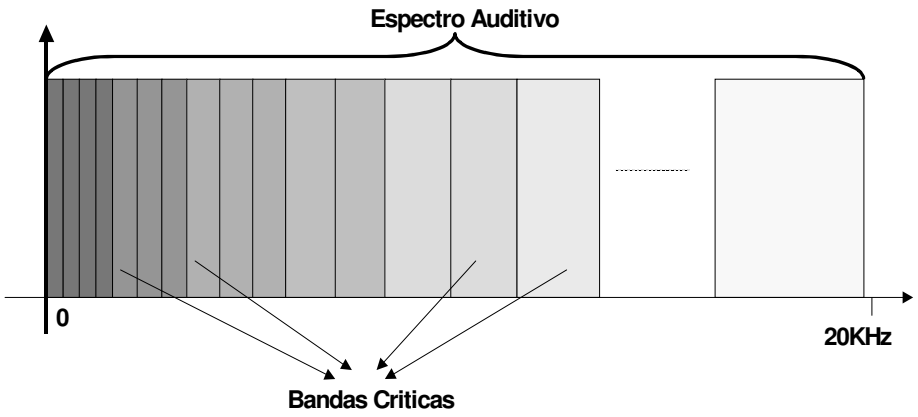


Fig. 3.4. Formato de divisão do espectro auditivo em bandas críticas.

3.3 MODELO PSICO- ACÚSTICO

Conforme mencionado anteriormente, uma das etapas que faz parte do codificador de áudio proposto neste trabalho é formada pelo chamado modelo psico-acústico do ouvido humano.

A tarefa fundamental desse processo é a de determinar o nível de audibilidade das subbandas wavelets que resultarão da decomposição do sinal de áudio de entrada.

TABELA 3.1 BANDAS CRÍTICAS DO OUVIDO HUMANO SEGUNDO O PADRÃO MP2.

Número da Banda	Freqüência Limite Superior da Banda (Hz)	Escala Bark
0	43.006	0.425
1	86.133	0.850
2	129.199	1.273
3	215.332	2.112
4	301.465	2.934
5	430.664	4.124
6	559.863	5.249
7	689.063	6.301
8	818.262	7.274
9	947.461	8.169
10	1119.727	9.244
11	1291.992	10.195
12	1507.324	11.232
13	1722.656	12.125
14	1981.055	13.042
15	2325.586	14.062
16	2756.250	15.100
17	3273.047	16.11
18	3875.977	17.079
19	4478.906	17.904
20	5340.234	18.922
21	6373.828	19.963
22	7579.688	20.971
23	9302.344	22.074
24	11369.531	22.984
25	15503.906	24.013
26	19982.813	24.573

A partir dessa informação, o codificador poderá determinar os níveis de quantização adequados para cada subbanda, a fim de introduzir a mínima distorção possível no sinal de áudio decodifcado. Quanto melhor for o modelo utilizado, melhor será também o desempenho do codificador no que se refere a qualidade por fator de compressão.

A construção de um modelo psico-acústico é relativamente complexa, por isso neste trabalho utiliza-se parcialmente os resultados do Modelo Psico-Acústico MP2 que apresenta um bom desempenho em processos de codificação de subbanda. Esse modelo foi projetado, no entanto, para ser utilizado em codificadores baseados na quantização de coeficientes MDCT (como

é o caso dos codificadores da linha MPEG). Dessa forma, foi necessário fazer um mapeamento de seus resultados, a fim de que os mesmos sejam válidos no domínio das subbandas wavelets.

Neste trabalho, propõe-se no Capítulo 4 um método de mapeamento freqüência-subbandas wavelets, do nível de ruído permitido para cada componente de freqüência. Esses valores de ruído constituem um dos resultados parciais do modelo MP2 cujos procedimentos de cálculo serão descritos a seguir.

3.3.1 MODELO PSICO-ACÚSTICO MPEG1 *LAYER*-2 (MP2)

O modelo Psico-acústico MP2 caracteriza-se por não fazer uma distinção dicotômica entre componentes tonais e não tonais. Em lugar disso, o espectro é transformado para o domínio de uma partição (57 partições ou subbandas especificadas na Tabela A.1 do Apêndice A [7]), onde a fração das componentes tonais e não tonais serão determinadas.

A partição suporta uma relação quase linear com o espaço das bandas críticas. Assim, ela provê uma resolução de quase uma linha espectral (ou aproximadamente 1/3 de uma banda crítica) na divisão do espectro auditivo [9].

O sinal de entrada no modelo psico-acústico deverá estar segmentado em blocos de 1024 amostras, sendo que os cálculos de mascaramento serão realizados uma vez para cada bloco a ser codificado.

O espectro auditivo (Fig.3.5) é determinado utilizando-se uma janela convencional de *Von Hann* e aplicando-se logo em seguida a transformada rápida de Fourier **FFT**.

Portanto, definindo-se $x(n)$ como sendo uma amostra de áudio que forma o bloco de entrada, então pode-se dizer que o espectro de áudio (segundo padrão MP2) é dado por:

$$X(k) = \frac{1}{1024} \sum_{n=0}^{1023} x_J(n) e^{-j\frac{2\pi nk}{1024}} \tag{3.2}$$

onde:

$$x_J(n) = x(n) w_J(n) \tag{3.3}$$

$$w_J(n) = \underbrace{\frac{1}{2} - \frac{1}{2}\cos\left(\frac{2\pi(n-0.5)}{1024}\right)}_{\text{janela de von Hann}} \tag{3.4}$$

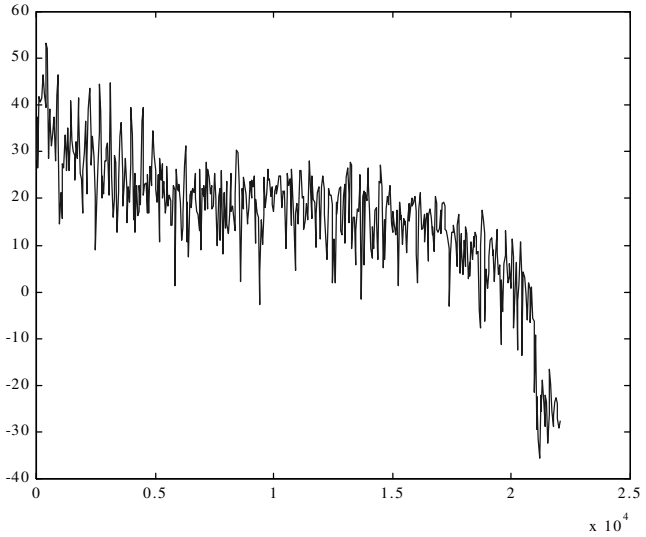


Fig. 3.5. Espectro de áudio na faixa 0Hz-22050Hz .

As amostras espectrais $X(k)$ constituem valores complexos que podem ser expressos na forma polar:

$$X(k) = \left|X(k)\right| e^{j\theta(k)} \tag{3.5}$$

A tonalidade do espectro é determinada medindo-se a função de imprevisibilidade $c(k)$ no domínio do tempo. Esse processo é feito primeiramente extrapolando-se linearmente a fase e a amplitude de dois espectros prévios consecutivos e calculando-se logo em seguida a diferença entre os mesmos [9]. Posteriormente, o resultado dessa operação é passado através de um adequado processo de normalização.

Dessa forma, tem-se que:

$$c(k) = \frac{[A_1(k)]^2 + [A_2(k)]^2}{r_d(w)} \tag{3.6}$$

onde :

$$A_1(k) = |X(k)|\cos(\theta(k)) - |\hat{X}(k)|\cos(\hat{\theta}(k)) \tag{3.7}$$

$$A_2(k) = |X(k)|\sin(\theta(k)) - |\hat{X}(k)|\sin(\hat{\theta}(k)) \tag{3.8}$$

$$r_d(k) = |X(k)| + |\hat{X}(k)| \tag{3.9}$$

$$\hat{X}(k) = 2|X_{t-1}(k)| - |X_{t-2}(k)| \tag{3.10}$$

$$\hat{\theta}(k) = 2\theta_{t-1}(k) - \theta_{t-2}(k) \tag{3.11}$$

Quando são detectadas componentes tonais deve-se esperar que a imprevisibilidade $c(k)$ seja quase igual a zero. Por outro lado, valores entre zero e um são esperados quando são detectadas componentes não tonais (Fig. 3.6).

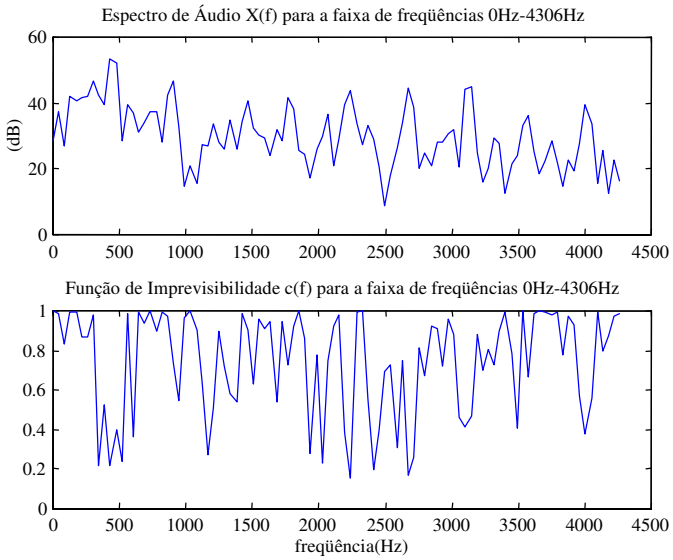


Fig. 3.6. Espectro de áudio e função de imprevisibilidade.

Os espectros da energia e da imprevisibilidade ponderada são calculados e mapeados dentro do domínio das 57 partições mencionadas anteriormente e especificadas em [7]. Os resultados disso constituem o espectro particionado da energia $E_p(b)$ e o espectro particionado da imprevisibilidade $c_p(b)$. Essas duas funções podem ser expressas, portanto, como (Fig. 3.8) :

$$E_p(b) = \sum_{k=k_i(b)}^{k_f(b)} |X(k)|^2 \tag{3.12}$$

$$c_p(b) = \sum_{k=k_i(b)}^{k_f(b)} |X(k)|^2 c(k) \tag{3.13}$$

onde $k_i(b)$ e $k_f(b)$ constituem os índices da primeira e da última linha espectral que formam a partição "b" , $b = 1, 2,...52$.

Os subíndices "p" indicarão, de agora em diante, que a respectiva variável encontra-se no domínio da partição.

Ambos os espectros $E_p(b)$ e $c_p(b)$ são prontamente convoluídos com uma função de espalhamento e em seguida re-normalizados para se corrigir o ganho introduzido por essa operação.

A função de espalhamento calcula os efeitos de mascaramento de uma partição com respeito às outras em que foi dividido o espectro auditivo. Essa função é calculada da seguinte forma:

Define-se primeiramente:

$$tmpx = 1.05(j - i) \tag{3.14}$$

onde "i" constitui o valor em escala Bark da componente ou partição que está sendo espalhada como componente mascaradora; "j" constitui também outro valor em escala Bark, neste caso, da componente ou partição que será mascarada por "i" .

Define-se também uma variável temporal "x" como sendo:

$$x = 8\min\left((tmpx\text{-}0.5)^2 - 2(tmpx\text{-}0.5), 0\right) \tag{3.15}$$

onde a função $\min(u, v)$ retorna o menor valor entre "u" e "v".

Finalmente, define-se:

$$tmpy = 15.811389 + 7.5(tmpx + 0.474)\text{-}17.5\left(1.0 + (tmpx + 0.474)^2\right)^{0.5} \tag{3.16}$$

Assim, a função de espalhamento $sprdng(i,j)$ (Fig.3.8) fica definida sob as seguintes condições :

- Se $tmpy < -100$ então

$$sprdng(i,j) = 0 \tag{3.17}$$

- caso contrário

$$sprdng(i,j) = 10^{\frac{(x+tmpy)}{10}} \tag{3.18}$$

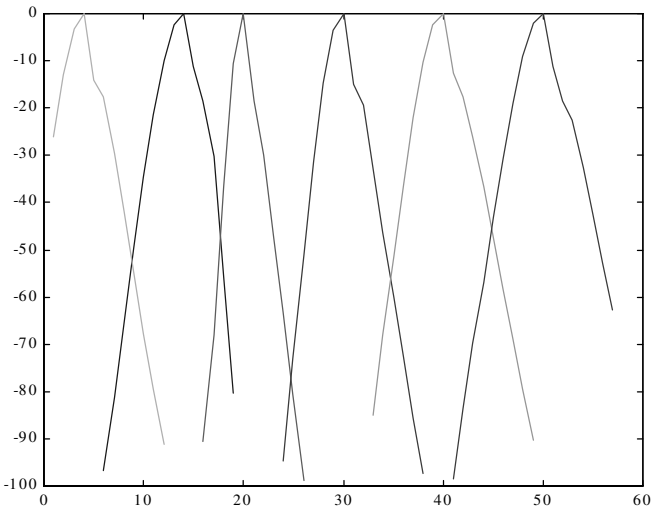


Fig. 3.7. Funções de espalhamento.

As convoluções de $E_p(b)$ e $c_p(b)$ com $sprdng(i,j)$ podem ser expressas através das seguintes expressões (Fig. 3.8) :

$$Ec_p(b) = \sum_{v=1}^{52} E_p(v)\, sprdng\big(bval(v), bval(b)\big) \tag{3.19}$$

$$ct_p(b) = \sum_{v=1}^{52} c_p(v)\, sprdng\big(bval(v), bval(b)\big) \tag{3.20}$$

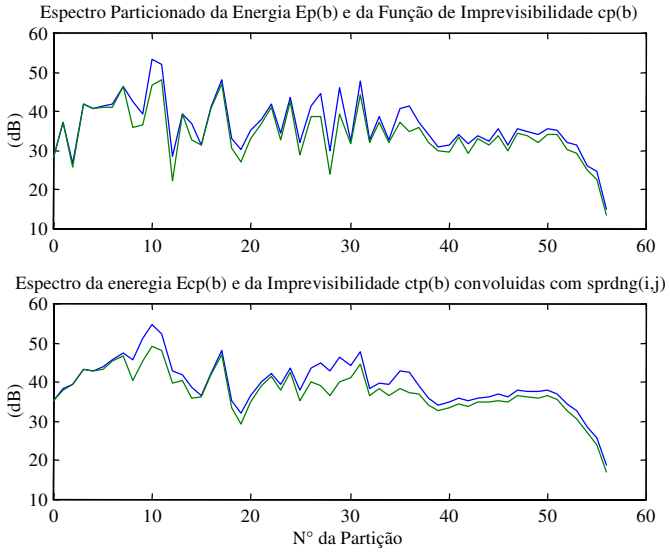


Fig. 3.8. Espectros particionados e convoluídos da energia (curva superior) e da função da imprevisibilidade (curva inferior).

onde $bval(b)$ constitui o valor em escala Bark correspondente à partição "b" (esses valores são especificados na Tabela A.1 do Apêndice A).

Logo $Ec_p(b)$ é re-normalizada a partir da seguinte expressão:

$$En_p(b) = Ec_p(b)\, rnorm(b) \tag{3.21}$$

onde:

$$rnorm(b) = \frac{1}{\sum_{v=1}^{52} sprdng\big(bval(v), bval(b)\big)} \tag{3.22}$$

A re-normalização da imprevisibilidade convoluída $ct_p(b)$ é calculada, por outro lado, como sendo a razão desta com o espectro convoluído da energia $Ec_p(b)$ (Fig. 3.9). Assim, tem-se que:

$$cn_p(b) = \frac{ct_p(b)}{Ec_p(b)} \tag{3.23}$$

A medição da tonalidade dessa razão é obtida posteriormente em forma logarítmica através da seguinte expressão (Fig. 3.9) :

$$t_p(b) = -0.299 - 0.43\ln\left(cn_p(b)\right) \tag{3.24}$$

onde os valores $t_p(b)$ são restritos ao intervalo $0 < t_p(b) < 1$.

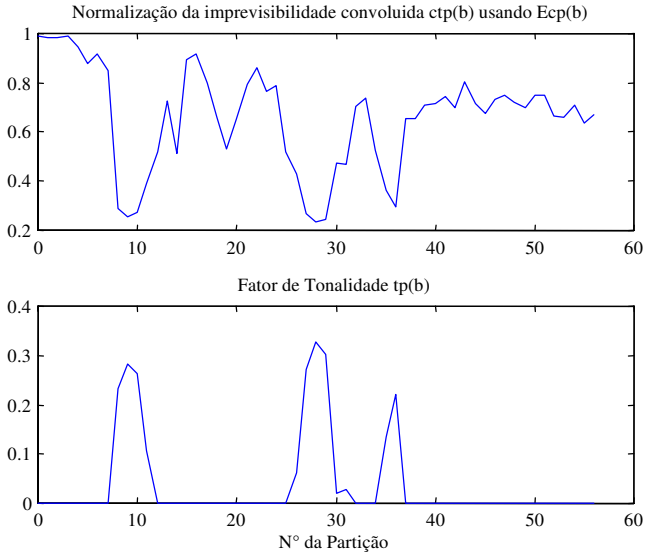


Fig. 3.9. Curvas da função de imprevisibilidade convoluída e normalizada e do fator de tonalidade.

O nível de mascaramento que será calculado para cada partição é dado pelo limiar de energia do ruído permitido $N_p(b)$. Isso é justificável uma vez que o ruído introduzido posteriormente pelo processo de quantização de amostras de áudio será inaudível à medida que este for sendo mascarado.

Os valores $N_p(b)$ (Fig. 3.10) para cada partição são calculados multiplicando-se o espectro de energia $En_p(b)$ por um fator de atenuação $fa_p(b)$ (Fig. 3.10). O logaritmo desse fator constitui a chamada relação sinal-ruído $SNR_p(b)$, a qual é determinada interpolando-se as funções $TMN(b)$

(especificada na Tabela A.1 do Apêndice A) e $NMT(b) = 5dB$ mediante o fator de tonalidade $t_p(b)$. Esse processo pode ser expresso como:

$$SNR_p(b) = \max\left(\min val(b), t_p(b)TMN(b) + \left((1 - t_p(b))NMT(b)\right)\right) \tag{3.25}$$

$$fa_p(b) = 10^{\frac{-SNR_p(b)}{10}} \tag{3.26}$$

$$N_p(b) = En_p(b)\, fa_p(b) \tag{3.27}$$

onde a função $\max(u, v)$ retorna o maior valor entre "u" e "v". Note-se também que a $SNR_p(b)$ é limitada a um valor máximo dado por $min\,val(b)$, o qual é especificado para cada partição na Tabela A.1 do Apêndice A.

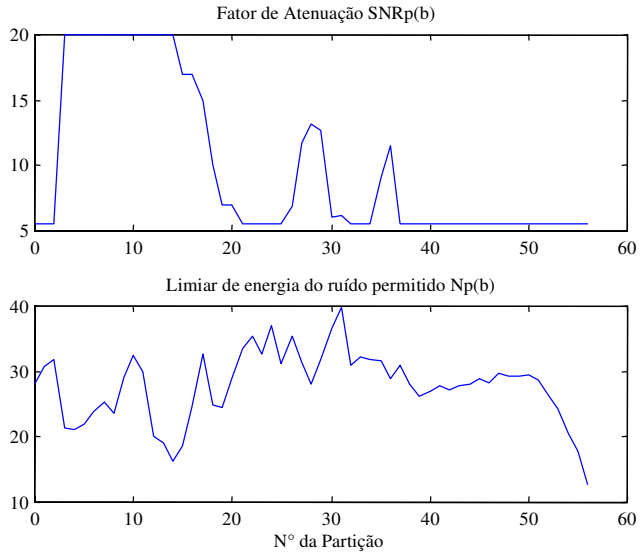


Fig. 3.10. Fator de atenuação e limiar de energia do ruído permitido.

Para se espalhar $N_p(b)$ nas linhas espectrais que formam o espectro de áudio utiliza-se:

$$N_f(k) = \frac{N_p(b)}{k_f(b) - k_i(b) + 1} \tag{3.28}$$

onde o valor obtido $N_f(k)$ é designado a todas as componentes de freqüência que formam a correspondente partição "b" (Fig. 3.11).

Finalmente, o limiar absoluto do ruído, permitido para cada componente (Fig. 3.11) "k" que forma o espectro auditivo, é calculado através da seguinte expressão:

$$thr(k) = \max\left(N_f(k), absthr(k)\right) \tag{3.29}$$

onde os valores $absthr(k)$ (especificados em dB na Tabela A.2 do Apêndice A) constituem os limiares mínimos absolutos de potência (limiar absoluto de sensibilidade) a partir dos quais o ouvido humano pode perceber as diversas componentes que formam o espectro auditivo (Fig. 3.12).

A operação (3.29) se justifica uma vez que abaixo do limiar $absthr(k)$, a componente correspondente torna-se inaudível e, portanto, o ruído mínimo permitido fica também limitado aos limiares de audibilidade que apresenta cada componente.

O modelo Psico-Acústico MP2 convencional envolve a execução de outras operações após o cálculo de $thr(k)$. No entanto, no codificador proposto neste trabalho será utilizada unicamente esta parte do modelo, já que os níveis de mascaramento de cada subbanda wavelets serão calculados a partir de $thr(k)$ mediante um método de mapeamento que será proposto e descrito no Capítulo 4.

3.4 QUANTIZAÇÃO VETORIAL

A quantização consiste em aproximar uma amostra (quantização escalar) ou conjunto de amostras (quantização vetorial) a um valor ou conjunto de valores predeterminados. Dessa forma, quanto maior for o número de valores disponíveis para se realizar essa aproximação, maior será também a resolução do quantizador correspondente. A diferença entre os valores originais e os valores aproximados constitui o que se denomina comumente de ruído de quantização.

A quantização constitui um processo irreversível, já que os valores originais das amostras não poderão ser jamais recuperados no decodificador. No entanto, esse processo constitui umas das partes mais importantes dos sistemas de codificação devido a que nele se obtém, geralmente, os maiores ganhos de compressão.

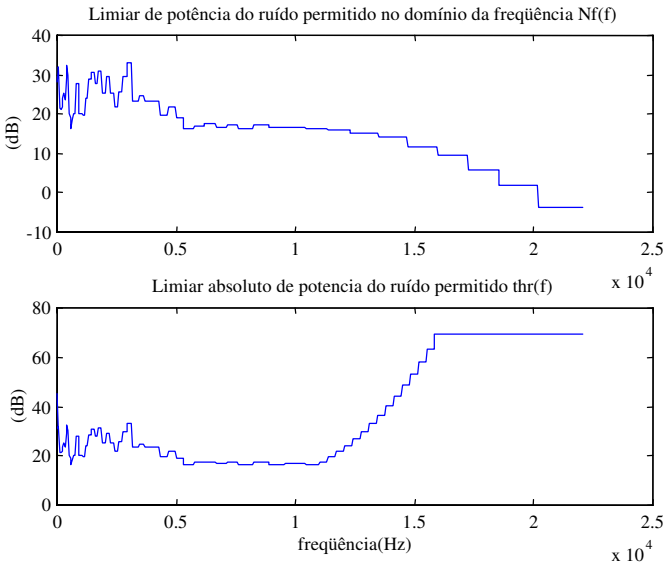


Fig. 3.11. Curvas do limiar de potência do ruído permitido e do limiar absoluto resultante.

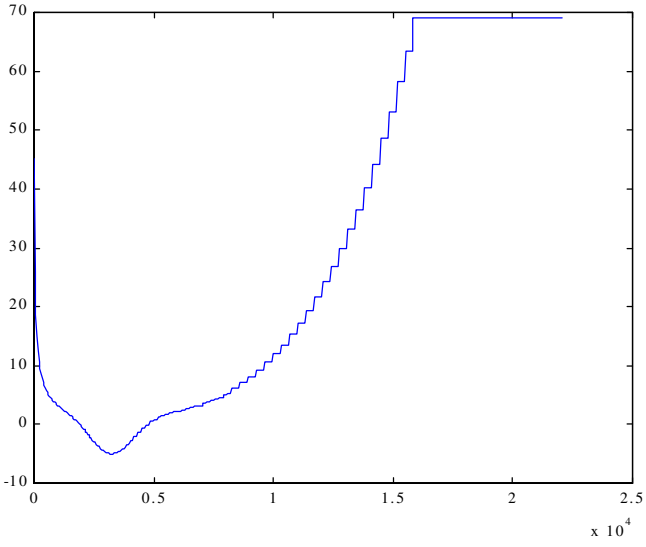


Fig. 3.12. Limiar absoluto de sensibilidade do ouvido humano em função da freqüência.

No codificador proposto neste trabalho, a resolução dos quantizadores, a serem utilizados em cada subbanda wavelet, é calculada a partir dos resultados do modelo psico-acústico. Isso permitirá, portanto, manter inaudível o ruído introduzido para certas taxas de compressão exigidas do codificador.

Existem dois formatos de quantização que serão utilizados na forma adaptativa no codificador proposto. Eles são : a **quantização escalar** e a **quantização vetorial** (VQ).

A primeira é a mais simples e conhecida, e é aplicada para se aproximar, de cada vez, valores de amostras individuais.

A segunda será descrita nesta parte do trabalho, e é concebida como uma generalização da quantização escalar quando essa é aplicada a um conjunto ordenado de amostras denominado comumente de "vetor". Isso levou a uma mudança no conceito de quantização, já que esse processo tinha sido aplicado somente a amostras individuais (vetores unidimensionais de amostras: conformados por uma única amostra) que formavam o conjunto de amostras do sinal a ser quantizado. Assim, levou também a novas idéias, conceitos e critérios a respeito do processo de quantização que é aplicado a sinais processados em blocos finitos de amostras (vetores) [10].

A VQ pode ser considerada também como um sistema de reconhecimento de padrões, onde o sinal de entrada (padrão) é casado a um dos padrões predeterminados que se encontram armazenados no codificador e no decodificador, para somente se transmitir o índice de identificação do padrão casado (compressão). O conjunto de padrões armazenados é denominado de **"codebook**" [10] .

Cada padrão predeterminado é chamado de **vetor quantizador**, e o número de elementos que forma o mesmo determina a dimensão do quantizador vetorial que está sendo utilizado. Porém, a resolução ou tamanho do quantizador é determinado unicamente pelo número de vetores quantizadores que conformam o "*codebook*".

Portanto, pode-se definir um quantizador vetorial "Q" de dimensão "k_d" e tamanho "N" como sendo o mapeamento de um vetor (no espaço Euclidiano R^{k_d}), dentro de um conjunto finito "C" de "N" pontos de reprodução denominados "*codewords*". Isso implica em que:

$$Q : R^{k_d} \rightarrow C \tag{3.30}$$

onde $C = (y_1, y_2, ..., y_N)$ e $y_i \in R^{k_d}$ para cada $i \in J \equiv \{1,2,...,N\}$.

O conjunto "C" é denominado "*codebook*" de tamanho "N", o qual significa que dispõe de "N" vetores quantizadores distintos definidos em R^{k_d} .

A resolução ou numero de bits por amostra resultante de um quantizador vetorial, é determinado da seguinte forma:

$$r = \frac{\log_2 N}{k_d} \tag{3.31}$$

a qual especifica o número de bits necessários para se representar cada componente do vetor quantizador utilizado na representação do sinal de entrada. Essa medida indica também a precisão que é obtida com um quantizador vetorial quando o "*codebook*" é bem construído.

Um "*codebook*" é normalmente implementado como uma tabela de valores nas memórias que acompanham os processadores digitais de sinais.

O número de bits utilizados, na representação das componentes que formam os vetores quantizadores, não afeta a taxa do quantizador vetorial, mas sim, o espaço reservado na memória para a implementação do "*codebook*".

Associado com cada quantizador vetorial de "N" pontos existe uma partição de R^{k_d} dentro de "N" regiões ou célula R_i onde $i \in J$. A i-ésima célula é definida por:

$$R_i = \left\{ \mathrm{x} \in R^{k_d} : Q(\mathrm{x}) = \mathrm{y_i} \right\} \tag{3.32}$$

a qual é chamada algumas vezes de imagem inversa ou pré-imagem de "$\mathrm{y_i}$" sob o mapeamento "Q" e é denotado mais exatamente por $R_i = Q^{-1}(\mathrm{y}_i)$.

Da definição de células, pode-se concluir que:

$$\bigcup_i R_i = R^{k_d} \quad \text{e} \quad R_i \bigcap R_j = 0 \quad \text{para } i \neq j \tag{3.33}$$

tal que as células formam uma partição de R^{k_d} .

Uma célula que é ilimitada é chamada de *"œverload cell"* e a coleção de "*œverload cells"* é chamada de *"œverload region"*.

Por outro lado, uma célula limitada é definida como sendo aquela que apresenta um volume finito em k_d -dimensões, e é chamada de "*granular cell*" . A coleção de "*granular cells*" conforma uma *"granular region"* [10].

Uma propriedade importante de um conjunto em R^{k_d} é a convexidade. Diz-se que um conjunto é definido como sendo convexo quando uma linha que une dois pontos localizados

dentro do conjunto forma parte ou encontra-se localizada totalmente dentro do mesmo[10]. Essa idéia bastante familiar permanece também aplicável em R^{k_d}. Portanto, um conjunto $S \in R^{k_d}$ é convexo se dado que $a \in S$ e $b \in S$ implica em que $\alpha a + (1-\alpha)b \in S$ para todo $0 < \alpha < 1$.

Um quantizador vetorial, por outro lado, é chamado de regular se [10] :

a) Cada célula R_i constitui um conjunto convexo, e

b) Para cada i, $y_i \in R_i$.

Um quantizador regular cujas células são limitadas por segmentos de superfícies hiperplanas é chamado de **quantizador vetorial politopal**, devido a que cada região da partição é um politope regular que consiste da interseção de um número finito de *"half spaces"* da forma $\left\{x \in R^{k_d} : u_v . x + \beta_v \geq 0\right\}$.

Por outro lado, um quantizador vetorial é considerado limitado se é definido sobre um domínio limitado $B \subset R^{k_d}$, onde cada vetor de entrada "**x**" pode ser localizado ou definido.

O volume do conjunto B denotado por $V(B)$ é finito e é dado por:

$$V(B) = \int_B dx \tag{3.34}$$

O processo de quantização vetorial pode ser decomposto em duas operações bem definidas: a **codificacão** e a **decodificação.**

O codificador "E" faz o mapeamento de R^{k_d} dentro de um conjunto de índices "J", enquanto que o decodificador "D" mapeia o conjunto de índices "J" dentro de um conjunto de reprodução "C" . Portanto, tem-se que:

$$E : R^{k_d} \rightarrow J \text{ e } D : J \rightarrow R^{k_d} \tag{3.35}$$

O processo total da quantização vetorial pode ser concebido também como sendo as duas operações em cascata expressas da seguinte maneira:

$$Q(x) = D \,.\, E(x) = D\big(E(x)\big) \tag{3.36}$$

Esse processo é mostrado na Fig. 3.13.

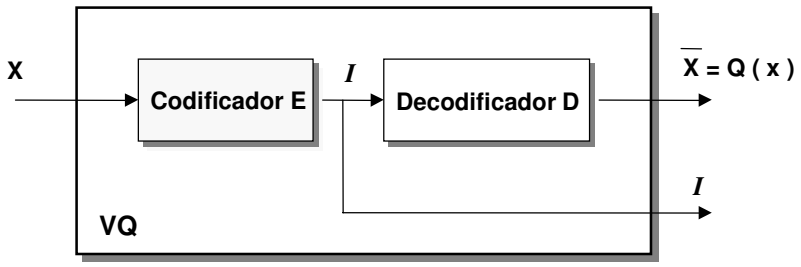


Fig. 3.13. Quantizador vetorial como um sistema em cascata Entre codificador e decodificador.

No contexto de um sistema de comunicações digitais, o codificador de um quantizador vetorial desempenha a tarefa de selecionar (implícita ou explicitamente) um apropriado vetor quantizador "$\mathbf{y}_i$" para aproximar ou em algum sentido para descrever ou representar um vetor de entrada "$\mathbf{x}$" [10].

O índice "*i*" de um vetor quantizador selecionado é transmitido como uma palavra binária para o receptor, onde o decodificador realiza um procedimento de "*look-table*" para se gerar a reprodução "$\mathbf{y}_i$" que constitui a aproximação quantizada do vetor original de entrada. Se uma seqüência de vetores de entrada é quantizada e transmitida, então o numero de bits utilizados para representar cada vetor (R) é dado por :

$$R = k_d r \tag{3.37}$$

onde "r" é a resolução definida por (3.31), enquanto que "k_d" constitui a dimensão ou número de componentes dos vetores quantizadores.

Sendo "f_v" a taxa dos vetores de entrada a serem codificados por segundo (*vetor rate*) então a taxa de bits (*bit-rate*) "R_s", em bits por segundo, é dada por [10]:

$$R_s = Rf_v \tag{3.38}$$

É conveniente também ver a operação de um quantizador vetorial em forma geométrica usando-se a nossa intuição para o caso de duas ou três dimensões. Assim, a partir desse ponto de vista, pode-se dizer que um quantizador de duas dimensões designa qualquer ponto de entrada no plano para um ponto particular de um conjunto de "N" pontos localizados também no mesmo plano. A Fig. 3.14 mostra esse exemplo onde se visualiza um quantizador de duas dimensões (2

componentes por cada vetor quantizador) que é aproximadamente politopal não regular desde que as células apresentem lados que não sejam segmentos de hiperplanos, além de serem células não convexas. Os pontos dentro de cada célula representam os vetores quantizadores num espaço de duas dimensões (o plano) e a região contendo cada vetor quantizador constitui uma partição ou célula.

Na Fig. 3.15 mostra-se, por outro lado, a título de exemplo um quantizador vetorial regular cujas células são limitadas nesse caso por polígonos politopes em duas dimensões.

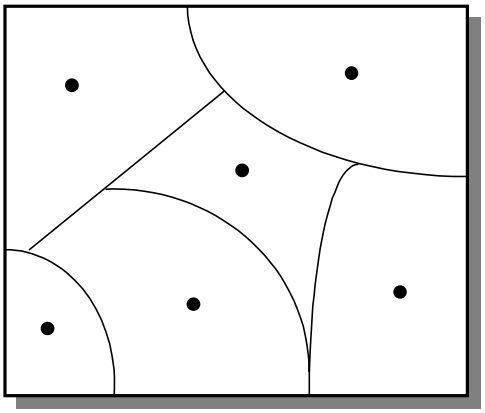


Fig. 3.14. Quantizador não regular.

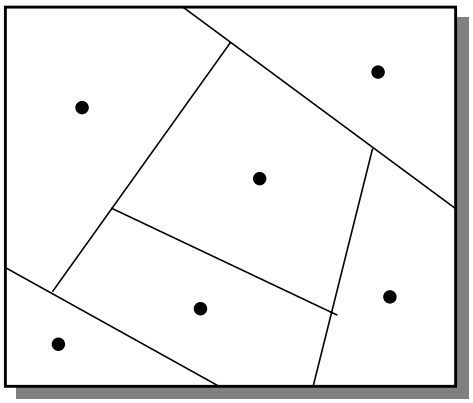


Fig. 3.15. Quantizador regular.

3.4.1 CARACTERÍSTICAS E PROPRIEDADES ESTRUTURAIS

Uma decomposição estrutural de um quantizador vetorial é particularmente importante desde que permita encontrar algoritmos computacionais mais eficientes. Isso é também importante

para efeitos de se fazer um estudo analítico do quantizador, assim como realizar a sua otimização [10]. Basicamente, existem duas estruturas que serão detalhadas a seguir.

- **ESTRUTURA PRIMÁRIA**

Como mencionado anteriormente, a tarefa do codificador na quantização vetorial é a de examinar cada vetor de entrada "**x**" e identificar em qual célula do espaço R^{k_d} pode ser localizado. O codificador vetorial identifica unicamente o índice "i" dessa região e o decodificador gera o vetor quantizador "$\mathbf{y}_i$" que representa essa região.

Para se modelar a operação do codificador, define-se uma função de seleção $S_i(\mathrm{x})$ como sendo o indicador ou "*membership function*" para a célula R_i da partição. Assim, tem-se que:

$$S_i(\mathrm{x}) = \begin{cases} 1 & \text{se } \mathrm{x} \in R_i \\ 0 & \text{outra forma} \end{cases} \tag{3.39}$$

A operação do quantizador vetorial pode então ser representada como:

$$Q(\mathrm{x}) = \sum_{i=1}^{N} \mathrm{y_i} S_i(\mathrm{x}) \tag{3.40}$$

Note-se que para qualquer valor do vetor de entrada, existe um único termo da somatória diferente de zero.

Na Fig. 3.16 mostra-se a chamada estrutura primária de um quantizador vetorial. Cada multiplicador, indicado por círculos, simplesmente multiplica um vetor quantizador por "um" ou "zero" a fim de produzir na sua saída um vetor quantizador ou um vetor zero, respectivamente. A somatória de fato é unicamente simbólica desde que apenas uma das suas entradas é diferente de zero.

Cada caixa S é uma operação não linear sem memória que examina todas as componentes do vetor de entrada simultaneamente, a fim de avaliar se a entrada se encontra ou não em uma determinada célula. A tarefa de se calcular o resultado binário de uma caixa S pode ser, de fato uma operação complexa, dependendo das características geométricas das células definidas no espaço R^{k_d}.

É conveniente abreviar $S_i(\mathrm{x})$ para simplesmente S_i, e considerar isso como sendo o valor binário resultante da i-ésima caixa seletora.

Da mesma forma que no caso escalar, o codificador e o decodificador podem ser separadamente modelados em uma estrutura primária correspondente. Dessa forma, leva-se "A" a ser o **gerador de índices** definido como sendo o mapeamento $A : B_N \rightarrow J$ e sua inversa como sendo o **decodificador de índices** definido como $A^{-1} : J \rightarrow B_N$ onde B_N denota um conjunto binário da forma $b = (b_1, b_2, ..., b_N)$ com $b \in \{0,1\}$, o qual é restringido a possuir exatamente um elemento diferente de zero. Portanto, $A(b) = i$ se $b_i = 1$ e $b_j = 0$ para todo $j \neq i$. Assim, a operação do codificador E pode ser escrita como:

$$E(\mathrm{x}) = A\big(S_1(\mathrm{x}), S_2(\mathrm{x}), ..., S_N(\mathrm{x})\big) \tag{3.41}$$

e a correspondente operação do decodificador pode ser expressa como:

$$D(i) = \sum_{l=1}^{N} \mathrm{y_i} A^{-1}(i)_l \tag{3.42}$$

O diagrama estrutural correspondente ao codificador e ao decodificador são mostrados na Fig. 3.17. No contexto de um sistema de comunicações, o índice "i" pode ser concebido como sendo o identificador de um símbolo de canal cuja natureza física pode ser de muitas formas. O gerador de índices e o decodificador de índices correspondem, respectivamente, ao codificador e ao decodificador de endereços, utilizados nos circuitos digitais de memórias.

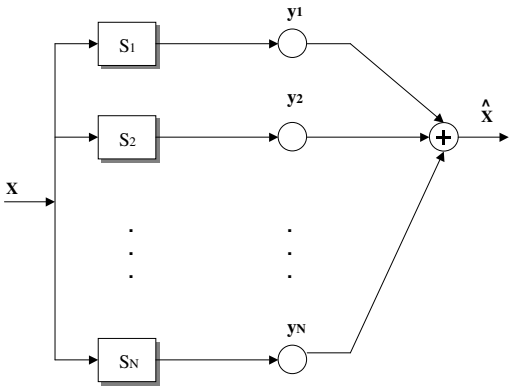


Fig. 3.16. Estrutura primária de decomposição de um quantizador vetorial.

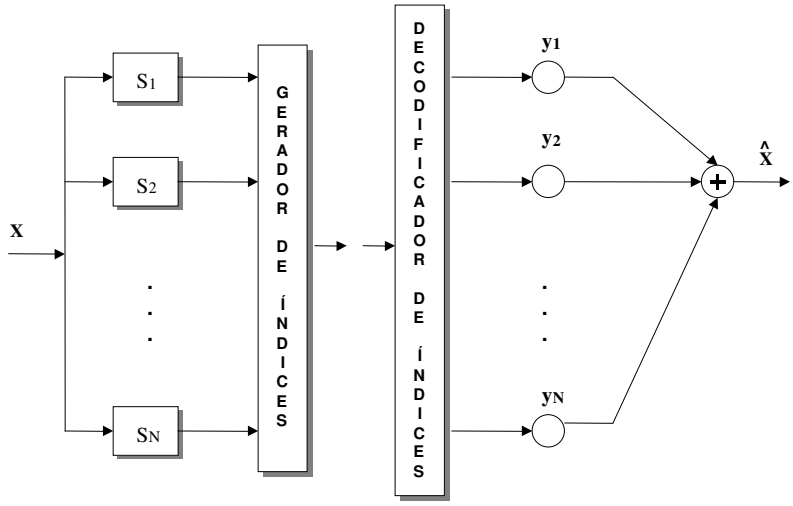


Fig. 3.17. Estrutura primária de um codificador e um decodifcador vetorial.

- **ESTRUTURA SECUNDÁRIA**

Para a definição dessa estrutura, focaliza-se a atenção sobre uma classe de quantizadores muito utilizada e extremadamente importante: a dos **quantizadores politopais** .

Tem-se definido um quantizador regular como sendo politopal quando cada célula que forma a sua partição constitui um politope convexo. Isso significa que os lados das células definidos em $k_d - 1$ dimensões consistem de segmentos hiperplanos.

Para explicar mais claramente isso, define-se um "*half-spaœ*" como sendo:

$$H_V = \left\{ \mathrm{x} \in R^{k_d} : \mathrm{u}_v.\mathrm{x} + \beta_v \geq 0 \right\} \tag{3.43}$$

onde o produto entre os dois vetores denota o produto escalar ou o produto interno definido como:

$$\mathrm{u}.\mathrm{x} = \mathrm{u}^t \mathrm{x} = \sum_{i=1}^{k_d} \mathrm{u}_i \mathrm{x}_i \tag{3.44}$$

O *half-spaœ* H_v é caracterizado pelo vetor parâmetro $\mathrm{u}_v \in R^{k_d}$ e o valor escalar β_v . Leva-se:

$$\Lambda_V = \left\{ \mathrm{x} \in R^{k_d} : \mathrm{u}_v.\mathrm{x} + \beta_v = 0 \right\} \tag{3.45}$$

a ser o hiperplano que limita H_v . Assim, qualquer região politopal pode ser representada pela interseção de *half planes,* onde os hiperplanos associados contém os lados ou limites do politope.

Particularmente, pode-se definir as células de um quantizador vetorial politopal da seguinte forma:

$$R_i = \bigcap_{v=1}^{L_i} H_V \tag{3.46}$$

onde L_i constitui o número de lados de $k_d - 1$ dimensões que formam uma célula R_i .

Leva-se $u(v)$ a denotar uma função degrau, onde $u(v) = 1$ para $v \geq 0$ e $u(v) = 0$ para $v < 0$. Assim, a função de indicação $T_v(\mathrm{x}) = 1_{H_v(x)}$ para o *half spaœ* H_v é definida por:

$$T_v(\mathrm{x}) = u\left(\mathrm{u}_v.\mathrm{x} + \beta_v\right) \tag{3.47}$$

Logo, obtém-se:

$$S_i(\mathrm{x}) = \prod_{v=1}^{L_i} T_v(\mathrm{x}) \tag{3.48}$$

A equação (3.48) é uma decomposição explícita da função de seleção em valores binários resultantes da função de decisão de hiperplanos. Note-se que é possível substituir o produto de (3.48) por uma operação lógica AND; a idéia é que S_i seja igual a "1" quando todos os valores T_v são "1".

Na Fig. 3.18 mostra-se a decomposição da função de seleção dentro de um produto de funções de seleção de hiperplanos. Note-se também que essa função, literalmente, responderá à pergunta: "Encontra-se "**x**" no lado positivo do hiperplano Λ_v?"

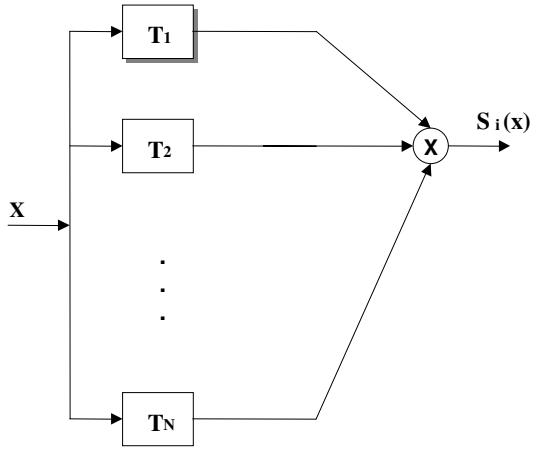


Fig. 3.18. Estrutura secundária de um seletor de função.

3.4.2 QUANTIZADORES *NEAREST NEIGHBOR*

Uma classe importante de quantizadores vetoriais são os chamados quantizadores **Voroni** ou **Nearest Neighbor** (NN). Esses quantizadores tem a característica de que a partição é totalmente determinada pelo *codebook* e pelo nível de distorção introduzido pelo quantizador. De fato, assume-se que o termo "**quantização vetorial**" é um sinônimo do conceito de quantizador vetorial *nearest neighbor* [10]**.**

Uma vantagem desse quantizador enraíza-se no fato de que o processo de codificação não requer nenhum armazenamento particular para se descrever a geometria das células.

Para se poder entender isso, inicia-se definindo-se uma medida da distorção $d(\mathrm{x},\mathrm{y})$ entre a entrada e a saída do quantizador. Por exemplo, poderia ser a energia do erro entre dois vetores (**x** e **y**) definida como:

$$d(\mathrm{x},\mathrm{y}) = \| \mathrm{x} - \mathrm{y} \|^2 = \sum_{i=1}^{k_d} (\mathrm{x}_i - \mathrm{y}_i)^2 \tag{3.49}$$

A equação (3.49) constitui uma das medidas mais comuns em sistemas de codificação de formas de onda. A partir disso, um quantizador vetorial *Voroni* ou *nearest neighbor* (NN) pode ser concebido como uma partição de células, definida como:

$$R_i = \{\mathrm{x} : d(\mathrm{x},\mathrm{y}_i) \le d(\mathrm{x},\mathrm{y}_j) \text{ para todo } j \in J\} \tag{3.50}$$

Em outras palavras, em um codificador NN cada célula R_i consiste de todos os pontos "**x**" que apresentam menor distorção quando são reproduzidos através de um vetor $\mathbf{y_i}$. Caso o vetor de entrada "**x**" esteja localizado no limite de duas células, deve-se designar ao mesmo a célula com o menor índice de identificação.

O algoritmo básico que descreve um codificador NN é apresentado através de 5 passos que são detalhados a seguir [10]:

Passo 1: Fazer $d = d_0, \quad j = 1, \text{ e } \quad i = 1.$

Passo 2: Calcular $D_j = d(\mathrm{x}, \mathrm{y_j}).$

Passo 3: Se $D_j < d,$ fazer $D_j \rightarrow d.$ Fazer $j \rightarrow i$.

Passo 4: Se $j < N,$ fazer $j + 1 \rightarrow j$ e retornar para o passo 2.

Passo 5: Parar. O resultado é o índice *i*.

O valor resultante de "i" constitui a saída do codificador $C(\mathrm{x})$, enquanto que o valor final de "d" é a distorção resultante entre "**x**" e "$\mathbf{y_i}$". O valor inicial "d_0" deve ser maior que qualquer valor de distorção esperado. Normalmente, esse valor é levado a ser o maior número positivo que pode suportar o processador correspondente.

A operação do codificador NN poderá também ser descrita por:

$$E(\mathrm{x}) = c(\mathrm{x}, C) \tag{3.51}$$

onde a operação funcional descrita por $c(.,.)$ é independente das especificações do quantizador e depende unicamente da medida do nível de distorção. Esse importante conceito é fundamental na implementação de virtualmente todos os quantizadores hoje em dia.

Na Fig. 3.19 mostra-se a forma de um quantizador NN. O *codebook* para esse caso constitui uma memória tipo somente de leitura (ROM-*Read Only Memory*) que é acessada pelo codificador que implementa a função $c(\mathrm{x}, \mathrm{C})$. Geralmente, qualquer distorção calculável pode ser utilizada em um codificador NN.

Uma característica adicional desse tipo de codificadores está no fato de que os parâmetros que definem a função de decisão de hiperplanos podem ser convenientemente especificados a partir dos valores dos vetores quantizadores "$\mathbf{y_i}$".

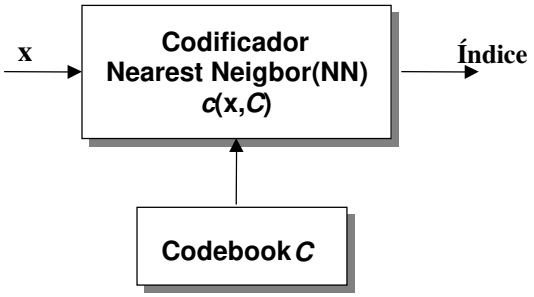


Fig. 3.19. Codificador Nearest Neighbor com codebook ROM.

Dessa forma, aplicando-se (3.49) e (3.50) as células podem ser expressas como:

$$R_i = \bigcap_{\substack{j=1 \\ j \neq i}}^{N} H_{ij} \tag{3.52}$$

Onde:

$$H_{ij} = \left\{ \mathrm{x} : \left\| \mathrm{x} - \mathrm{y_i} \right\|^2 \leq \left\| \mathrm{x} - \mathrm{y_j} \right\|^2 \right\} \tag{3.53}$$

Observação : A expressão $\|\mathrm{v}\|$ retorna a norma do vetor "**v**".

Expandindo-se os termos quadráticos e simplificando-se obtem-se:

$$H_{ij} = \left\{ \mathrm{x} : \mathrm{u_{ij}} \cdot \mathrm{x} + \beta_{ij} \geq 0 \right\} \tag{3.54}$$

onde:

$$\mathrm{u_{ij}} = 2(\mathrm{y_i} - \mathrm{y_j}) \tag{3.55}$$

e:

$$\beta_{ij} = \left\| \mathrm{y_j} \right\|^2 - \left\| \mathrm{y_i} \right\|^2 \tag{3.56}$$

Dessa forma, observa-se que as células NN de um quantizador Voroni são formadas pela intersecção de *half-spaces* tal que cada quantizador é politopal e seus *half-spaces* são explicitamente determinados a partir dos vetores quantizadores "$\mathbf{y}_i$" que formam seu *codebook*. Essa conveniente propriedade é uma conseqüência direta do fato de que a distorção medida pela energia do erro é uma função quadrática das componentes dos vetores quantizadores.

A partir dessas propriedades pode-se fazer uma simples e importante variação do algoritmo NN, o qual implica no cálculo do produto escalar antes de se determinar os termos quadráticos. Isso pode ser expresso como:

$$\min_i{}^{-1} \left\| \mathrm{x} - \mathrm{y_i} \right\|^2 = \max_i{}^{-1} \left[\mathrm{x}^t \mathrm{y_i} + \alpha_i \right] \tag{3.57}$$

onde os valores:

$$\alpha_i = \frac{-\left\| \mathrm{y_i} \right\|^2}{2} \tag{3.58}$$

podem ser computados e armazenados junto com o *codebook*.

O algoritmo NN modificado, portanto, é constituído também de 5 passos que são detalhados a seguir:

Passo 1: Fazer $f = f_0, \quad j = 1, \quad \text{e} \quad i = 1.$

Passo 2: Calcular $F_j = \mathrm{x}^t \mathrm{y_j} + \alpha_j$.

Passo 3: Se $F_j > f$, fazer $F_j \rightarrow f$. Fazer $j \rightarrow i$.

Passo 4: Se $j < N$, fazer $j + 1 \rightarrow j$ e retornar para o passo 2.

Passo 5: Parar. O resultado é o índice *i*.

O valor resultante de "i" é a saída do codificador $C(\mathrm{x})$, enquanto que o valor final de "f" determina a distorção entre "$\mathbf{x}$" e "$\mathbf{y}_i$", de acordo com $d(\mathrm{x}, \mathrm{y_i}) = \left\| x \right\|^2 - 2f$.

O valor inicial f_0 deve ser menor do que qualquer valor esperado de distorção, e usualmente é levado a zero.

A título de exemplo, considere-se o quantizador bidimensional mostrado na Fig. 3.20. Nessa figura, os seis vetores quantizadores são indicados com pontos pretos e os limites de decisão hiperplanos (segmentos lineares) são indicados com linhas em negrito para se separar as regiões

vizinhas (*neighbor regions*). As linhas que não são apresentadas em negrito, constituem as extensões dos segmentos hiperplanos para ajudar a visualizar a eficiente operação de codificação. Cada limite de decisão é um segmento de hiperplano e é indicado com as letras A,B,C..., enquanto que os dois *half-spaces* separados por cada hiperplano são indicados com "+" e "-".

Para se determinar eficientemente a célula onde pode ser localizado o vetor de entrada, descreve-se a seguir uma estrutura tipo árvore, baseada em uma série de testes para decisões de hiperplano.

Suponha que o passo inicial do algoritmo de procura é calcular a decisão binária do hiperplano "**A**". Se a entrada "**x**" é localizada no lado direito de "**A**" (indicado com "+"), então os vetores quantizadores 1 e 5 são imediatamente eliminados como candidatos a serem o *nearest neighbor* (vizinho mais próximo) uma vez que eles estão inteiramente contidos no lado esquerdo de "**A**"**.** Similarmente, os vetores quantizadores 2 e 3 seriam eliminados como candidatos se a entrada "**x**" estivesse localizada no lado esquerdo de "**A**" (-). Dependendo do resultado do primeiro teste, escolhe-se um de dois novos testes para continuar a procura. Assim, se a entrada é localizada no lado + de A, então testa-se sobre qual lado do hiperplano "**C**" pode estar localizada. Se a entrada estiver no lado negativo (-) de "**A**", então testa-se os dois lados do hiperplano "**B**". Cada teste elimina um ou mais vetores candidatos. Esse procedimento corresponde a um algoritmo de procura com estrutura de árvore, tal como mostra a Fig. 3.21. Nessa figura, cada nó é indicado com uma letra correspondente a uma decisão binária para os hiperplanos indicados com segmentos na Fig. 3.20. A árvore é não balanceada uma vez que os diferentes caminhos através da árvore não apresentam o mesmo número de nós. Cada nó da árvore onde um teste de hiperplano acontece é indicado para identificar o correspondente hiperplano da Fig. 3.20. Os nós terminais da árvore identificam os vetores quantizadores finais.

Esse algoritmo diminui notavelmente a complexidade computacional do processo de procura. Para casos onde o tamanho do *codebook* é maior do que 6, é eliminado um maior número de candidatos em cada teste de decisão. A máxima profundidade da árvore pode ser muito menor do que o tamanho N do *codebook*.

É importante mencionar que os quantizadores vetoriais que serão utilizados no codificador proposto serão construídos em formato de árvore binária para dar maior eficiência computacional ao processo de compressão.

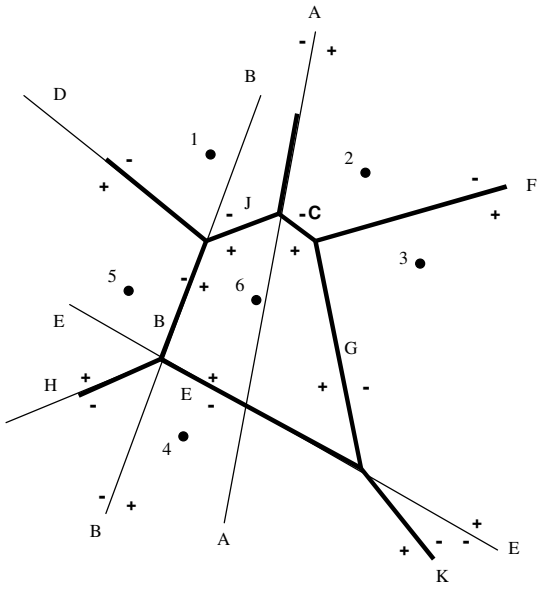


Fig. 3.20. Quantizador bidimensional.

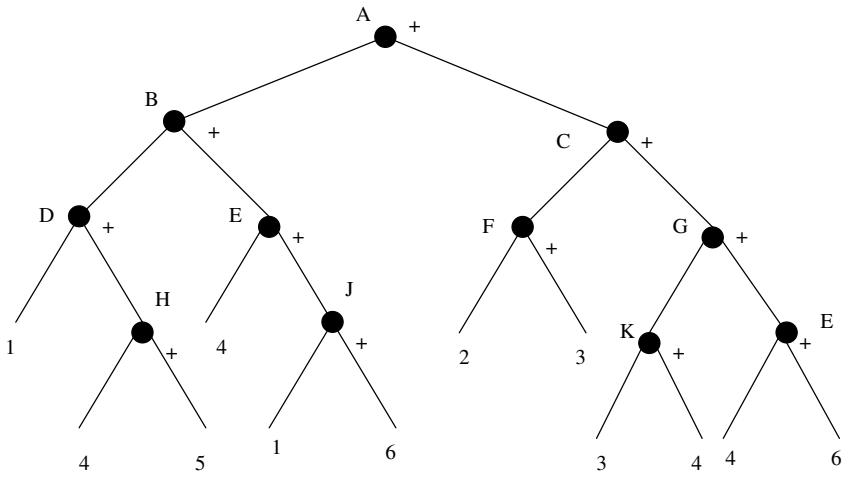


Fig. 3.21. Estrutura de árvore para um eficiente algoritmo de procura NN.

3.4.3 CONDIÇÕES PARA OTIMIZAÇÃO

O principal objetivo no desenvolvimento de quantizadores vetoriais é encontrar um *codebook* (que especifique ao decodificador) e uma partição ou regra de codificação (que especifique ao

codificador) que maximizem o desempenho total do quantizador durante todos os instantes de tempo em que são quantizados os vetores que formam o sinal de entrada.

Esse desempenho poderá ser determinado através da média estatística de uma adequada medida de distorção, que pode ser expressa como:

$$D = Ed(\mathrm{X}, Q(\mathrm{X})) = \int d(\mathrm{x}, Q(\mathrm{x}))f_{\mathrm{X}}(\mathrm{x})d\mathrm{x} \tag{3.59}$$

onde $f_{\mathrm{X}}(\mathrm{x})$ é a função de densidade de probabilidade do vetor "**X**", enquanto que a integração constitui uma integral múltipla sobre todo o espaço de "k_d" dimensões.

Quando o sinal de entrada apresenta uma distribuição discreta, a distorção pode ser expressa como:

$$D = Ed(\mathrm{X}, Q(\mathrm{X})) = \sum_i d(\mathrm{x}_\mathrm{i}, Q(\mathrm{x}_\mathrm{i}))p_{\mathrm{X}}(\mathrm{x}_\mathrm{i}) \tag{3.60}$$

onde $\{\mathrm{x}_\mathrm{i}\}$ são os valores de "**X**" que tem probabilidade diferente de zero.

Para estabelecer as condições de otimização, assume-se que o *codebook* de tamanho N é dado, que o vetor aleatório "**X**" é estatisticamente especificado e que uma medida de distorção tenha sido selecionada. Deseja-se então determinar as condições necessárias para que um quantizador seja ótimo no sentido de que minimize a distorção média do processo.

Basicamente são estabelecidas três condições de otimização: a condição do vizinho mais próximo ou *nearest neighbor*, a condição do centróide e a condição de probabilidade zero para pontos localizados no limite entre duas ou mais células.

- **CONDIÇÃO *NEAREST NEIGHBOR* (NN)**

Para um dado conjunto de possíveis níveis de saída, especificados no *codebook* $C = (\mathrm{y}_1, \mathrm{y}_2, ..., \mathrm{y}_\mathrm{N})$, a partição ótima de células deve satisfazer:

$$R_i \subset \{\mathrm{x} : d(\mathrm{x}, \mathrm{y}_i) \le d(\mathrm{x}, \mathrm{y}_j);\ \text{para todo } j\} \tag{3.61}$$

o que implica:

$$Q(\mathrm{x}) = \mathrm{y}_i \text{ só se } d(\mathrm{x}, \mathrm{y}_i) \le d(\mathrm{x}, \mathrm{y}_j) \text{ para todo } j \tag{3.62}$$

Dessa forma, para um determinado decodificador, existe um codificador que atinge a mínima distorção mediante o mapeamento *nearest neighbor:*

$$d\left(\mathrm{x}, Q(\mathrm{x})\right) = \min_{\mathrm{y}_i \in C} d(\mathrm{x}, \mathrm{y}_i) \tag{3.63}$$

Posteriormente, considera-se o caso de otimização de um *codebook* definido a partir de uma partição que é especificada no codificador. Isso leva à **condição do centróide**, $cent(R)$ o qual é definido como sendo o vetor "**y**" (se é que existe) que minimiza a distorção média entre um ponto "**X**" (em R^{k_d}) e "**y**". Esse cálculo é feito a partir da distribuição de probabilidade do vetor de entrada "**X**".

Dessa forma, tem-se que:

$$\mathrm{y}^* = \mathrm{cent}(R) \text{ se } E\left[d\left(\mathrm{X}, \mathrm{y}^*\right) \middle| \mathrm{X} \in R\right] \leq E\left[d(\mathrm{X}, \mathrm{y}) \middle| \mathrm{X} \in R\right] \tag{3.64}$$

para todo $\mathrm{y} \in R^{k_d}$.

É conveniente também escrever isso como um mínimo inverso através da seguinte expressão:

$$\mathrm{cent}(R) = \min_{\mathrm{y}}{}^{-1} E\left[d\left(\mathrm{X}, \mathrm{y}\right) \middle| \mathrm{X} \in \mathrm{R}\right] \tag{3.65}$$

Facilmente pode-se observar que o centróide de R pode ser expresso como:

$$\mathrm{cent}(R) = E\left(\mathrm{X} \middle| \mathrm{X} \in R\right) \tag{3.66}$$

- **CONDIÇÃO DO CENTRÓIDE**

Para uma partição especificada $\{R_i; i = 1,..., N\}$, os vetores quantizadores ótimos devem satisfazer:

$$\mathrm{y}_i = \mathrm{cent}(R_i) \tag{3.67}$$

Desde que a partição é fixa, o seletor de funções $S_i(\mathrm{x}) = 1_{\mathrm{R}_i(\mathrm{x})}$ que é bem definido no decodificador, pode ser expresso em termos de um vetor binário:

$$\mathrm{S} = (S_1(\mathrm{X}), S_2(\mathrm{X}), \ldots S_N(\mathrm{X})) \tag{3.68}$$

A condição de otimização está no fato de que o erro de estimação **X**-**Y** deve ser ortogonal a cada uma das variáveis de observação, S_i. Portanto:

$$E(\mathrm{X}S_i) = \sum_{j=1}^{N} \mathrm{y}_j E(S_j S_i) \tag{3.69}$$

Note que $E(S_j S_i)$ é zero para $j \neq i$ e é igual a $ES_i^{\,2} = P_i$ quando $j = i$.

Também, o valor esperado $E(\mathrm{X}S_i)$ pode ser expresso como:

$$E(\mathrm{X}S_i) = E\left[\, \mathrm{X} \middle| S_i = 1 \middle| \,\right] P_i \tag{3.70}$$

Dessa forma, utilizando-se (3.69) e (3.70), pode-se chegar a:

$$\mathrm{y}_i = \frac{E(\mathrm{X}S_i)}{ES_i^{\,2}} = E\left[\mathrm{X} \middle| S_i = 1\right] = \mathrm{cent}(R_i) \tag{3.71}$$

Esse resultado assume que cada partição tem probabilidade diferente de zero $P_i \neq 0$. No caso em que $P_i = 0$, apresenta-se o problema *"empty cell"*, o qual é resolvido aplicando-se a condição de otimização que é detalhada a seguir.

- **CONDIÇÃO DE PROBABILIDADE ZERO PARA PONTOS NO LIMITE DAS CÉLULAS**

Essa condição constitui outro aspecto importante que deve ser levado em conta na otimização de quantizadores vetoriais e pode ser expressa como:

$$P\left(\bigcup_{j=1}^{N} B_j \right) = 0 \tag{3.72}$$

Isso implica em que todos os pontos B_j localizados nos limites entre as células adjacentes devem ocorrer com probabilidade zero.

Uma forma alternativa de dar essa condição é exigir que a coleção de pontos eqüidistantes de pelo menos dois centróides deve ter probabilidade de ocorrência igual a zero. Isso implica em que:

$$P(\mathrm{x} : d(\mathrm{x}, \mathrm{y}_i) = d(\mathrm{x}, \mathrm{y}_j) \text{ para algum } i \neq j) = 0 \tag{3.73}$$

Essa condição poderá ser satisfeita desenvolvendo-se o quantizador com seus correspondentes centróides alocados em lugares estratégicos a fim de se eliminar qualquer probabilidade de ocorrência dos pontos limites.

Outra forma muito utilizada é a de ajuntar os dois centróides comprometidos numa única região e particionando (para compensar) logo em seguida aquele centróide em cuja célula são alocados a maior quantidade de vetores de entrada "**x**".

3.4.4 DESENVOLVIMENTO DOS QUANTIZADORES PARA O CODIFICADOR DE ÁUDIO PROPOSTO

Na literatura sobre quantização vetorial, existe uma grande quantidade de métodos propostos que, baseados em diferentes critérios, permitem a geração de quantizadores vetoriais adequados para distintos tipos de aplicações.

No caso do codificador de áudio proposto, a quantização vetorial será utilizada adaptativamente para quantizar as subbandas wavelets resultantes do processo de decomposição. O termo adaptativamente se refere ao fato de que o tamanho do "*codebbok*" utilizado numa determinada subbanda dependerá do nível de audibilidade dessa em um determinado instante de tempo. Assim, por exemplo, utilizando-se uma árvore de procura binária na forma que mostra a Fig. 3.21, o número de etapas utilizadas na procura vai depender do ruído permitido pela correspondente subbanda, já que à medida que essa se torna mais audível, a procura deverá ser formada por um maior número de etapas a fim de se dar uma maior resolução ao quantizador e vice-versa.

Outro fator que foi levado em conta é a dimensão dos vetores quantizadores a serem utilizados em cada *"codebook"*. Logicamente, à medida que essa dimensão for ficando menor, a resolução do quantizador irá ficando maior, mas também, por outro lado, será aumentada a tarefa computacional do codificador. Portanto, para se poder chegar a um ponto de equilíbrio em ambas as condições, os quantizadores foram dimensionados de acordo com as subbandas onde serão

aplicados e em função da capacidade do ouvido para detetar o ruído de quantização em cada uma delas. Essa capacidade foi avaliada através de uma série de testes de escuta aplicada a vários tipos de sinais de som.

Após a realização dessas provas e de se levar em conta os critérios mencionados anteriormente, foi desenvolvido um procedimento de geração de codebooks que deu resultados bastante satisfatórios no codificador de áudio proposto. Esse procedimento é detalhado através de 3 etapas, as quais são descritas a seguir

1. **GERAÇÃO DAS SEQÜÊNCIAS FONTES DE AMOSTRAS DE ÁUDIO:**

   Para a construção dos quantizadores vetoriais a serem utilizados na quantização das subbandas wavelets, foi necessário primeiramente dispor de uma fonte geradora de seqüências de amostras que formam as subbandas wavelets resultantes da decomposição de um sinal de áudio. Isso foi feito submetendo-se vários tipos de sinais de som ao processo de decomposição utilizado no codificador proposto e, em seguida, armazenando-se as seqüências que formam as subbandas correspondentes à faixa de **7kHz à 22kHz**. Nessa faixa de freqüências comprovou-se do ponto de vista perceptual, que o desempenho do ouvido no que se refere à detecção da distorção introduzida diminui consideravelmente, sendo que, por esse motivo, é justificável a utilização da quantização vetorial.

   Conforme será visto no Capítulo 4, a partir do processo de decomposição são obtidas 3 subbandas wavelets para a faixa de **7kHz a 11kHz** e 4 subbandas para a faixa de **11kHz a 22kHz**. No primeiro caso, cada subbanda é formada por seqüências de 64 amostras sendo que o número total de seqüências armazenadas para o cálculo dos vetores quantizadores foi de 40.000. Para o Segundo caso, o tamanho das seqüências que formam cada subbanda é de 128 amostras, sendo que o número total de seqüências armazenadas para esse caso foi de 52.000.

   É importante mencionar que, antes do armazenamento, os valores das amostras que formam cada seqüência fonte são escalonados para o intervalo $[-1, 1]$, utilizando-se para tanto, um fator de escala que constitui o maior valor absoluto das amostras que formam a correspondente seqüência ou subbanda. Isso é feito devido ao fato de que o codificador fará a quantização escalar ou vetorial utilizando um fator de escala que será codificado e transmitido separadamente.

2. **DIMENSIONAMENTO E CLASSIFICAÇÃO DAS SEQÜÊNCIAS FONTES:**

   Conforme mencionado anteriormente, o dimensionamento dos vetores quantizadores a

serem utilizados no codificador dependerá do grau de distorção, do ponto de vista de percepção que poderá suportar uma determinada subbanda. Dessa forma, após realizar vários testes de escuta, chega-se a resultados muito satisfatórios utilizando-se vetores quantizadores de **2 componentes** para a faixa de 7kHz a 11kHz e vetores quantizadores de **4 componentes** para a faixa de 11kHz a 22kHz.

Por outro lado, para se obter uma boa qualidade de áudio e aumentar posteriormente a eficiência dos algoritmos de procura, decidiu-se implementar quantizadores vetoriais separadamente em função do sinal das componentes que formam os seus correspondentes vetores quantizadores. Logicamente, isso requererá maior quantidade de memória, mas isso é compensado pelos resultados alcançados.

Para a implementação de quantizadores vetoriais com as características mencionadas, foi necessário particionar as seqüências geradas na etapa 1 em vetores de amostras com 2 e 4 componentes (dependendo do caso). Os vetores foram classificados e armazenados separadamente de acordo com o sinal das componentes.

A classificação por sinal implica, para o caso de 2 componentes, na geração de 4 tipos de quantizadores vetoriais correspondentes a 4 tipos de vetores quantizadores : (positivo [+], positivo[+]), (negativo[-], positivo[+]), (positivo[+], negativo[-]) e (negativo[-], negativo[-]). O valor zero em todos os casos será considerado positivo.

Dessa forma, seguindo-se o mesmo procedimento para o caso de 4 componentes torna-se fácil comprovar que seriam 16 tipos de quantizadores vetoriais correspondentes a 16 tipos de vetores quantizadores.

3. ALGORITMOS DE CÁLCULO DOS VETORES QUANTIZADORES:

Para o cálculo dos vetores quantizadores que formarão os *codebooks* dos diferentes quantizadores vetoriais utiliza-se: **o algoritmo NN modificado** para distribuir os vetores fonte nas correspondentes partições e o chamado "**Lloyd Algorithm**" para otimizar as partições em função da mínima energia do erro.

O "*Lloyd Algorithm*" é descrito a seguir:

Passo 1: Começa-se com um *codebook* inicial C_1 e se faz o correspondente índice $m = 1$.

Passo 2: Dado um *codebook* $C_m = \{y_i\}$ particiona-se, utilizando-se a condição NN, a seqüência de vetores fonte de entrada de acordo com a região à qual pertencem:

$$R_i = \{\mathrm{x} \in \mathrm{T} : d(\mathrm{x}, \mathrm{y}_i) \leq d(\mathrm{x}, \mathrm{y}_j); \text{para } j \neq i\} \tag{3.74}$$

Dessa forma, "**x**" constitui um vetor da seqüências fonte " T ", " y_i " constitui o vetor quantizador que melhor se casa com "**x**" enquanto que " y_j " representa qualquer outro vetor quantizador que forma o *codebook* C_m.

Para medir o melhor casamento de "**x**" com algum dos vetores quantizadores " y_i " utiliza-se :

$$i = \text{iden max}(r_1, r_2 ... r_{N-1}, r_N) \tag{3.75}$$

Onde:

$$r_i = \mathrm{x}^{\mathrm{T}} \mathrm{y}_i + \alpha_i \tag{3.76}$$

Os valores α_i são definidos de acordo com a (3.58). Por outro lado, a função $\text{iden max}(r_1, r_2 ... r_{N-1}, r_N)$ retorna o índice do máximo valor r_i que para esse caso corresponderia à célula ou vetor quantizador com o qual "**x**" foi casado.

Caso aconteça uma situação na qual $r_i = r_j$, então se retornará o índice de menor valor.

Passo 3: Uma vez particionada a seqüência fonte de entrada em função das células que formam o *codebook* C_m, realiza-se em seguida o cálculo dos novos centróides ou vetores quantizadores através da seguinte expressão:

$$\mathrm{y}_i = \frac{1}{N_i} \sum_{n=1}^{N_i} \mathrm{x}_{i,n} \tag{3.77}$$

Onde " $\mathrm{x}_{i,n}$ " constitui o n-ésimo vetor fonte localizado na região ou célula " i ".

Portanto a somatória (3.77) implica em uma soma vetorial dos N_i vetores que foram localizados nessa região. Os novos vetores quantizadores formam agora o

codebook C_m, enquanto que o inicial passa a constituir o *codebook* C_{m-1} .

Passo 4: Calcula-se a distorção total D_m produzida pela quantização da seqüência fonte a partir dos vetores quantizadores calculados no passo 3. Essa distorção é determinada através da seguinte expressão:

$$D_m = \sum_{i=1}^{N} Eq_i \tag{3.78}$$

onde :

$$Eq_i = \sum_{n=1}^{N_i} \left(x_{i,n} - y_i\right)^2 \tag{3.79}$$

constitui energia do erro obtido em cada célula ou partição.

Passo 5: Compara-se a distorção produzida pelo *codebook* C_{m-1} com aquela obtida no passo 4. Logo, se:

$$\frac{D_{m-1} - D_m}{D_m} < 0.001 \tag{3.80}$$

então, retorna-se ao passo 2. Caso contrário, o algoritmo termina.

Para dar uma maior eficiência computacional ao algoritmo de procura, decidiu-se construir os quantizadores vetoriais em formato de árvore binária, o qual foi realizado utilizando-se o procedimento que é descrito a seguir:

Passo 1: Define-se N_e como sendo o número de etapas da árvore a ser construída e se faz o contador de etapas $i_e = 1$.

Passo 2: Calcula-se o centróide inicial de toda a seqüência de entrada utilizando-se a expressão (3.77). Esse centróide constituirá o único vetor quantizador que

formará o "*codebook*" inicial.

Passo 3: Dado um *codebook* $C_m = \{y_i\}$, de tamanho N, particiona-se o mesmo utilizando-se:

$$\begin{aligned} \hat{y}_{2i+1} &= y_i \\ \hat{y}_{2i} &= 1.2 y_i \end{aligned} \tag{3.81}$$

onde o *codebook* resultante $\hat{C}_m = \{\hat{y}_i\}$ estará constituído de $2N$ vetores quantizadores.

Passo 4: Otimiza-se $\hat{C}_m = \{\hat{y}_i\}$, utilizando-se o "*Lloyd Algorithm*" descrito anteriormente. O *codebook* resultante dessa etapa é em seguida armazenado para formar posteriormente parte da árvore.

Passo 5: Faz-se $i_e = i_e + 1$

Passo 6: Se $i_e < N_e$, então retorna-se ao passo 3 colocando-se como *codebook* de entrada aquele otimizado no passo 4, caso contrário, o algoritmo termina.

Utilizando-se esse procedimento foram implementadas 20 árvores balanceadas correspondentes a 20 tipos de seqüências fontes.

Para as subbandas wavelets correspondentes à faixa de **7kHz a 11kHz** foram geradas 4 tipos de árvores de 7 etapas cada uma, enquanto que para as subbandas correspondentes à faixa de **11kHz a 22kHz** foram construídas 16 árvores de 5 etapas cada uma.

Para o primeiro caso, um maior número de etapas se justifica uma vez que, nesse caso, requer-se uma maior resolução dos quantizadores vetoriais a fim de se manter o ruído introduzido imperceptível para o ouvido humano.

Dessa forma, de acordo com o nível de resolução requerido para uma determinada subbanda num determinado instante de tempo, o codificador decidirá o número de etapas de procura que serão utilizados para se determinar o vetor quantizador adequado.

Por outro lado, o número de bits utilizados para se identificar o correspondente vetor quantizador dependerá da etapa onde o mesmo esteja localizado. Espera-se assim um maior número de bits para se identificar quantizadores localizados nas etapas superiores das

árvores.

Todos esses parâmetros assim como também os requerimentos de memória e de carga computacional, serão detalhados no Capítulo 4.

3.5 COMENTÁRIOS FINAIS

O desempenho de codificadores de áudio baseados em modelos psico-acústicos tem sido muito satisfatório para altas e intermediárias taxas de compressão. No entanto, como se mencionou anteriormente, o desenvolvimento desse tipo de modelos é relativamente complexo, sendo que por esse motivo decidiu-se utilizar parcialmente o modelo MP2 que apresenta um bom desempenho para processos de codificação de subbanda. Neste capítulo foi detalhado, portanto, a parte do modelo que será utilizada no codificador de áudio proposto.

Por outro lado, foi realizado também um detalhamento da teoria de quantização vetorial, assim como os procedimentos que foram seguidos para a implementação dos quantizadores a serem utilizados no codificador.

A interação entre todas essas etapas de codificação será descrita no Capítulo 4 onde será detalhado o funcionamento do processo de compressão.

CAPÍTULO 4

DESCRIÇÃO DO CODIFICADOR DE ÁUDIO PROPOSTO

4.1 Introdução

No presente capítulo descreve-se finalmente o codificador de áudio que é proposto neste trabalho de tese. O processo de compressão é baseado na quantização das amostras que formam as sub-bandas *wavelets* assim como na codificação de entropia que utiliza os chamados códigos de Huffman.

O funcionamento geral do codificador pode ser resumido da seguinte forma: o sinal original de entrada é segmentado em blocos de 1024 amostras com uma superposição de 48 amostras nos blocos de áudio adjacentes. Em seguida, cada bloco é decomposto em subbandas *wavelets* através de uma árvore de decomposição *wavelet packets*. Para cada subbanda resultante calcula-se um fator de escala (máximo valor absoluto dos valores das amostras que formam cada subbanda), o qual é logo quantizado logaritmicamente através de 64 níveis de quantização (6bits por fator de escala) e em seguida codificado e transmitido separadamente.

Em paralelo a todo esse processo executa-se parcialmente o Modelo Psico-Acústico MP2, o qual foi descrito no Capítulo 3. O resultado parcial do modelo constitui o nível de ruído permitido para cada componente de freqüência que forma o espectro do bloco de entrada. Essa função de ruído é mapeada em seguida ao domínio das subbandas *wavelets*, para finalmente obter-se a chamada relação sinal-mascaramento (SMR) que indica aproximadamente o nível de audibilidade de cada subbanda.

Os valores SMR alimentam em seguida o algoritmo de alocação de bits a fim de se determinar o número de níveis de quantização para cada subbanda. Esse processo se encerra quando o número total de bits utilizados, na codificação do atual bloco de áudio, ultrapassa um certo limiar calculado a partir da taxa de bits requerida para o sinal codificado.

A partir dos resultados do algoritmo de alocação de bits, o codificador poderá decidir o tipo de quantização (vetorial ou escalar) que será aplicado a cada subbanda *wavelet*. Após esse processo utiliza-se a codificação de entropia para se codificar unicamente as subbandas que foram quantizadas escalarmente.

Na saída do codificador utiliza-se um buffer de controle de taxa, dado que os códigos utilizados na codificação de entropia são de comprimento variável.

Na Fig. 4.1 apresenta-se o diagrama de blocos do codificador proposto. A descrição das etapas que formam o mesmo, assim como a interação entre elas serão detalhadas a seguir nas seções que fazem parte deste capítulo.

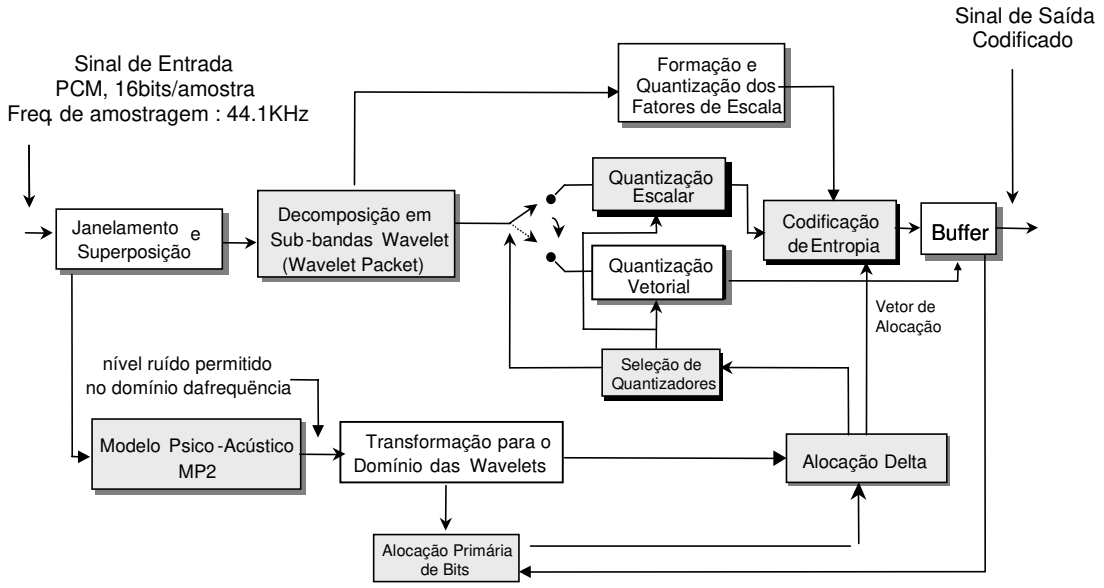


Fig. 4.1. Diagrama de Blocos do Codificador de Áudio Proposto.

4.2 SEGMENTAÇÃO E JANELAMENTO

O sinal de entrada do codificador é inicialmente segmentado em blocos de 1024 amostras superpostos em 48 amostras nos blocos de áudio adjacentes. O tamanho do bloco não pode ser aumentado excessivamente devido ao fato de que a qualidade do som fica empobrecida como conseqüência da não estacionariedade dos sinais de áudio. Por outro lado, blocos de tamanho muito pequeno podem comprometer o tempo de processamento do codificador, principalmente para aplicações em tempo real.

Um aumento no tamanho dos blocos implica também em uma diminuição da resolução temporal do sistema de compressão. Nessa situação, a distorção produzida pela quantização de transitórios é aumentada consideravelmente através da introdução de um ruído perceptível denominado ruído de pré-eco. O nome de pré-eco deve-se ao fato de que o transitório espalha o ruído de quantização sobre amostras passadas (20ms) e sobre amostras futuras (250ms) de menor nível de energia. Isso se deve ao fato de que o ouvido humano apresenta um tempo de reação frente a sons de alta intensidade que acontecem em curtos intervalos de tempo.

Por outro lado, a superposição entre blocos de áudio adjacentes permite a diminuição do efeito de bloco que é traduzido na percepção de um ruído "click" no sinal de áudio reconstruído. Isso é devido a que uma segmentação sem superposição implica em se considerar uma correlação zero entre blocos de áudio adjacentes. Esse fato não é valido em música já que a formação da melodia é baseada na correlação existente entre as notas musicais.

À medida que o número de amostras superpostas aumenta considera-se portanto uma maior correlação entre blocos de áudio, porém, esse incremento pode resultar num excessivo aumento do número de amostras a se codificar, o que afetaria o desempenho do processo de compressão.

Para facilitar a decodificação das amostras que participam na superposição de blocos, foi utilizada uma janela que é igual a raiz quadrada da janela convencional de **Hanning** [11]. Esse tipo de janela permitiu a obtenção de bons resultados no que se refere à tarefa computacional para se recuperar um bloco de áudio.

Na Fig. 4.2 apresenta-se o processo de segmentação e janelamento. Por outro lado, o formato da janela aplicada a todo um bloco de áudio, é mostrado a Fig. 4.3. Note-se que somente as 48 amostras que ficam nas bordas são passadas pela janela modificada de **Hanning**.

A expressão que define a janela da Fig. 4.3 é dada por:

$$J_B(n) = \begin{cases} \sqrt{\frac{1}{2} - \frac{1}{2}\cos\left(\frac{2\pi n}{95}\right)} & 0 \le n \le 47 \\ 1 & 48 \le n \le 975 \\ \sqrt{\frac{1}{2} - \frac{1}{2}\cos\left(\frac{2\pi(n-976+48)}{95}\right)} & 976 \le n \le 1023 \end{cases} \tag{4.1}$$

Definindo-se $x^t(n)$ como sendo uma amostra de um novo bloco de áudio de 976 amostras $(0 \le n \le 975)$que ingressa no codificador (as outras 48 são tomadas do bloco anterior $x^{t-1}(n)$), então pode-se expressar o janelamento como:

$$x_{AJ}(n) = \begin{cases} x^{t-1}(n+928)J_B(n) & 0 \le n \le 47 \\ x^t(n-48)J_B(n) & 48 \le n \le 1023 \end{cases} \tag{4.2}$$

onde $x_{AJ}(n)$ representa a seqüência que forma o bloco de 1024 amostras resultante do processo de janelamento e superposição.

Esse bloco de amostras será, portanto, posteriormente submetido à árvore de decomposição *wavelet packets*.

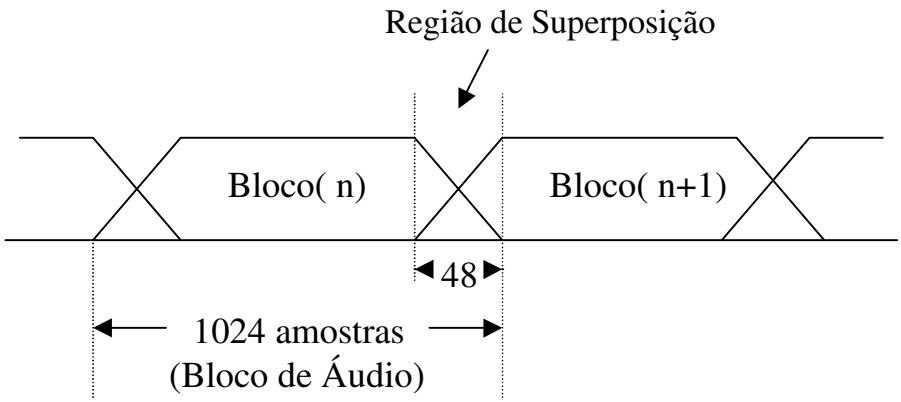


Fig. 4.2. Janelamento e Superposição de Blocos de Áudio.

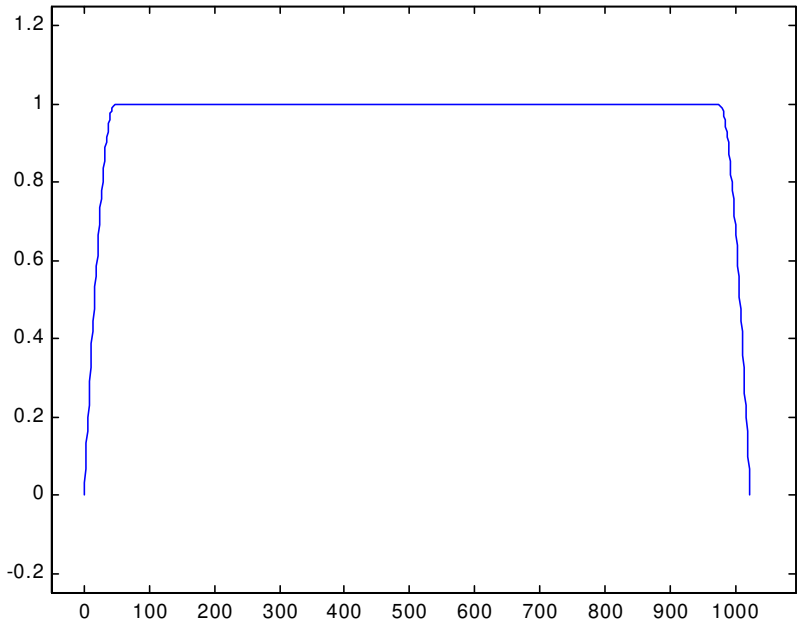


Fig. 4.3. Janela uitilizada em blocos de 1024 amostras.

Por outro lado, o processo de dejanelamento no decodificador pode ser expresso como:

$$\hat{x}^t{}_D(n) = \hat{x}_{AJ}(n) J_B(n) \quad , \quad 0 \le n \le 1023 \tag{4.3}$$

Logo:

$$\hat{x}^t(n) = \begin{cases} \hat{x}^t{}_D(n) + \hat{x}_D^{t-1}(n+976) & 0 \le n \le 47 \\ \hat{x}^t{}_D(n) & 0 \le n \le 975 \end{cases} \tag{4.4}$$

onde $\hat{x}_{AJ}(n)$ e $\hat{x}^t(n)$ constituem, respectivamente, as amostras dos blocos reconstruídos da transformada e do dejanelamento.

Note-se que em cada decodificação somente são recuperadas 976 amostras de um bloco de áudio. Isso se deve ao fato de que as últimas 48 amostras deverão ser recuperadas junto com as primeiras 48 do bloco seguinte consecutivo.

4.3 Decomposição em Sub-bandas *Wavelets*

Após o processo de janelamento, cada bloco de áudio é submetido a uma árvore de decomposição *wavelets packets* que especifica os contínuos processos de filtragens e subamostragens que formam a transformada. O formato de decomposição utilizado foi proposto por Deppen Sinha e Ahmed H. Tewfik em [11], e permite a obtenção de subbandas que se aproximam do esquema das bandas críticas do ouvido humano.

A forma da árvore de decomposição é mostrada na Fig. 4.4. Note-se que para cada uma das 29 subbandas *wavelets* resultantes, são especificadas as faixas de freqüências correspondentes.

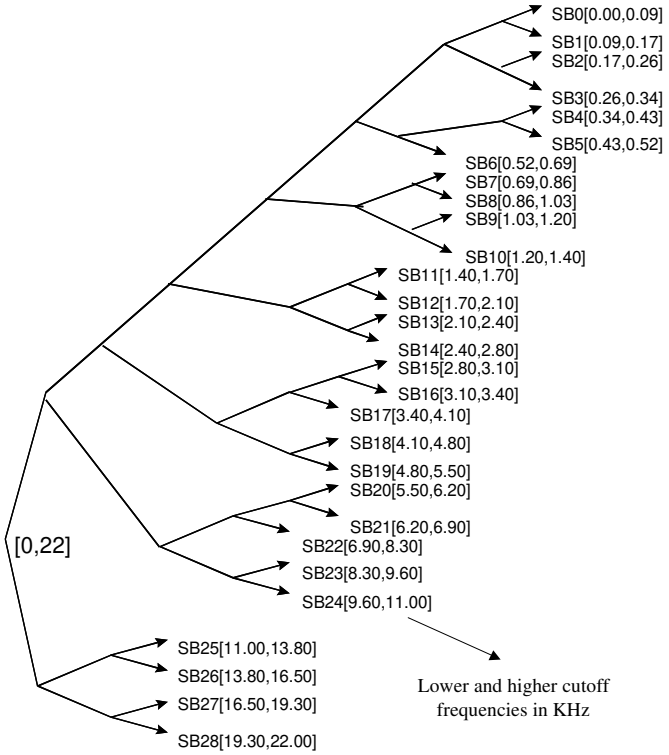


Fig. 4.4. Árvore de decomposição wavelets packets utilizada no codificador de áudio proposto.

Cada par de ramos na árvore representa o formato de decomposição mostrado na Fig. 4.5. Diferentemente do esquema apresentado na Fig. 2.2, o fator 2 nesse caso encontra-se multiplicando as respostas impulsivas dos filtros de decomposição. Esse fato permitiu a obtenção de melhores resultados no que se refere à qualidade do sinal decodificado.

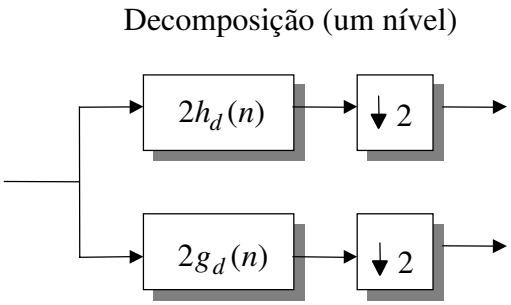


Fig. 4.5. Esquema de decomposição de um nível utilizado no Codificador de Áudio Proposto.

O correspondente par de reconstrução que é utilizado na árvore de recuperação do sinal original é mostrado na Fig. 4.6. Logicamente, esse esquema é utilizado no decodificador.

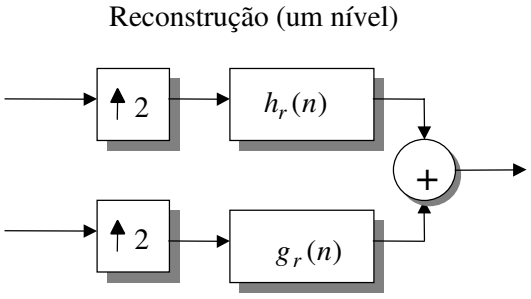


Fig. 4.6. Esquema de reconstrução de um nível utilizado no Decodificador .

As respostas impulsivas dos filtros *wavelets* utilizados deverão satisfazer as condições de ortogonalidade e de perfeita reconstrução que foram detalhadas no Capítulo 2. O tamanho dos filtros foi dimensionado de acordo com a seletividade requerida para uma determinada taxa de bits do sinal codificado.

Neste trabalho, utilizam-se filtros Daubechies na faixa de db10 a db40 para taxas de 128 a 80 kbit/s, respectivamente.

De acordo com a teoria de processamento digital de sinais, sabe-se por outro lado, que a convolução de um bloco de L amostras com um filtro de N coeficientes resulta em uma

seqüência de $L+N-1$ amostras. Se esse processo é seguido de uma etapa de sub-amostragem por um fator de 2, então o número de amostras resultantes será igual a :

$$floor\left(\frac{L+N-1}{2}\right) \tag{4.5}$$

onde a função *floor* retorna o menor número inteiro mais próximo do argumento.

A título de exemplo, considere-se um bloco de $L=1024$ amostras que ingressa num único nível de decomposição (Fig. 4.5) com filtros de tamanho $N=80$. De acordo com (4.5) obtém-se na saída de cada subamostrador um número de amostras igual a 552, sendo que o número total de amostras dessa única etapa de decomposição é igual a 1104.

Observa-se assim que o número de amostras de saída é maior do que o número de amostras do bloco de entrada. Portanto, à medida que se continua o processo nos níveis seguintes de filtragem da árvore, tem-se um incremento considerável do número total de amostras resultantes do processo de decomposição.

Isso constitui um problema devido ao fato de que o número total de amostras das subbandas em conjunto será maior do que o do bloco de entrada, o qual comprometerá, em todos os aspectos, o desempenho do mesmo.

Para solucionar esse problema, cada bloco de amostras de tamanho L que ingressa em cada par de decomposição da árvore, é primeiramente estendido periodicamente com um número de amostras igual a $N-1$. O bloco de amostras resultante da filtragem e da subamostragem será portanto periódico com período igual a $\frac{L}{2}$. Considerando-se somente um período dessa seqüência na saída de cada subamostrador (parte passa altas e parte passa baixas), obtém-se um número total de amostras resultantes igual a aquele do bloco de entrada.

Na Fig. 4.7 apresenta-se a título de exemplo a extensão periódica de um bloco de tamanho $L=16$ que será posteriormente decomposto em duas sub-bandas (passa-altas e passa-baixas) mediante o esquema mostrado na Fig. 4.5 e utilizando filtros *Daubechies* db3.

As seqüências de amostras resultantes dos processos de filtragem e da subamostragem são mostradas na Fig. 4.8. Note-se que o período das seqüências é igual a 8 amostras e isso constitui a metade das amostras do bloco de entrada no par de decomposição. Na Fig. 4.8 indica-se portanto o período que será selecionado como sendo do bloco resultante.

No processo de recuperação do sinal original será igualmente aplicada a convolução circular em cada par de filtragem que forma a árvore de reconstrução.

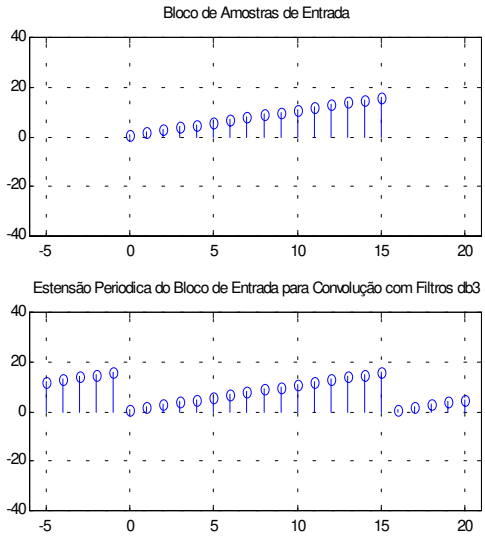


Fig. 4.7. Extensão Periódica de um Bloco de Amostras.

Para um bloco de entrada de 1024 amostras que é considerado no codificador proposto e que é decomposto através da árvore da Fig.4.4, o número de amostras para cada subbanda resultante é mostrado na Tabela 4.1. Nessa tabela, as subbandas foram classificadas em grupos segundo o tamanho das mesmas. Assim, por exemplo o conjunto de subbandas na faixa SB0...SB5 pertencem ao mesmo já que são formadas pelo mesmo número de amostras.

Observe-se finalmente que, através da aplicação da convolução circular em cada par de decomposição da árvore, o número total de amostras das subbandas resultantes em conjunto é o mesmo que o daquele do bloco de entrada.

Por outro lado, a implementação da árvore pode ser também realizada através do uso da FFT (*Fast Fourier Transform*). Para esse caso, os pares de decomposição e reconstrução são implementados da forma mostrada nas Figs. 4.9 e 4.10 [1]. Isso permitirá, em muitos casos, dar maior eficiência computacional aos algoritmos de decomposição e reconstrução implementados em circuitos DSP.

Para decidir sobre qual método utilizar, deve-se determinar o número de operações matemáticas envolvidas em cada caso. Isso constitui uma medida variável uma vez que depende do tamanho do bloco a ser decomposto e do tamanho dos filtros a serem utilizados. Geralmente, o método da FFT apresenta melhor desempenho computacional do que aquele da convolução direta no domínio do tempo.

Para representar a operação de decomposição de um bloco de áudio através da árvore mostrada na Fig. 4.4, utiliza-se a seguinte expressão:

$$x_{sw}(p,n) = Wt_{p,n}\left(x_{AJ}(m), h_d(m), g_d(m)\right) \tag{4.6}$$

onde $x_{sw}(p,n)$ constitui a amostra "n" da subbanda "p", enquanto que $x_{AJ}(m)$ constitui á seqüência ou bloco resultante do processo de janelamento definido em (4.2).

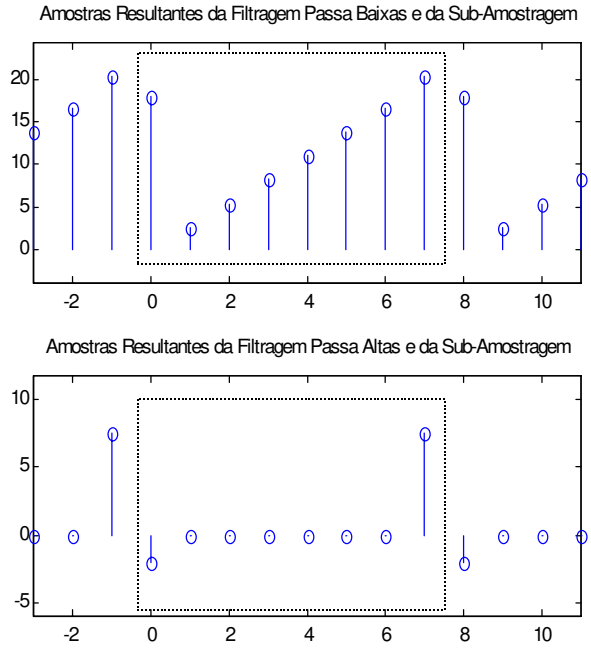


Fig. 4.8. Blocos de amostras resultantes de um par de decomposição passa-baixas e passa-altas para um bloco de entrada de 16 amostras e utilizando filtros db3.

TABELA 4.1. NÚMERO DE AMOSTRAS POR SUBBANDA, PARA UM BLOCO DE ENTRADA DE 1024 AMOSTRAS

SubBanda	N° Amostras/ Subbanda	Total
SB0...SB5	4	24
SB6...SB10	8	40
SB11...SB16	16	96
SB17...SB21	32	160
SB22...SB24	64	192
SB25...SB28	128	512
	TOTAL	1024

Para o caso do codificador proposto tem-se $0 \le p \le 28$ e $0 \le n \le L(p)-1$. O valor $L(p)$ especifica assim, o número de amostras que forma a subbanda "p", a qual é dada na segunda coluna da Tabela 4.1.

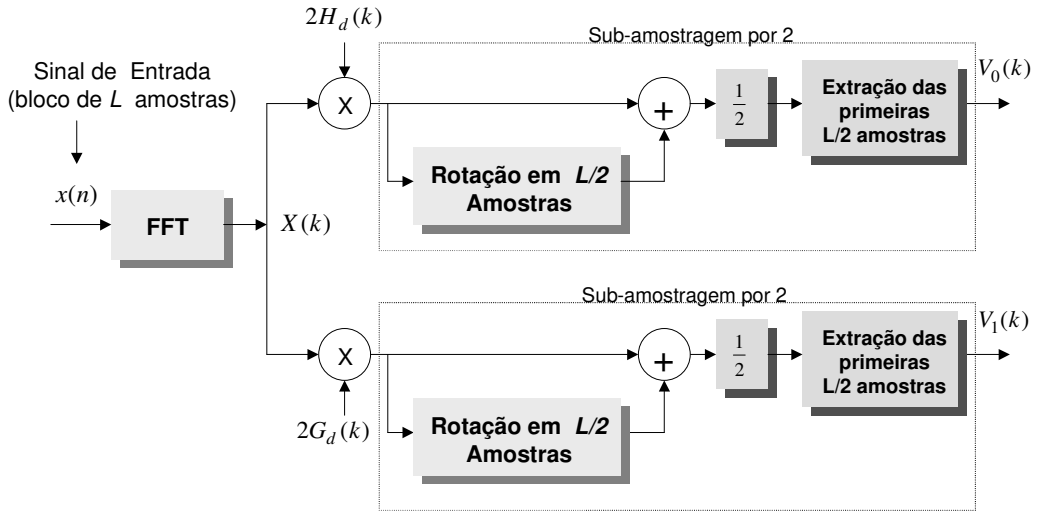


Fig. 4.9. Par de decomposição passa-baixas e passa-altas, utilizando a FFT.

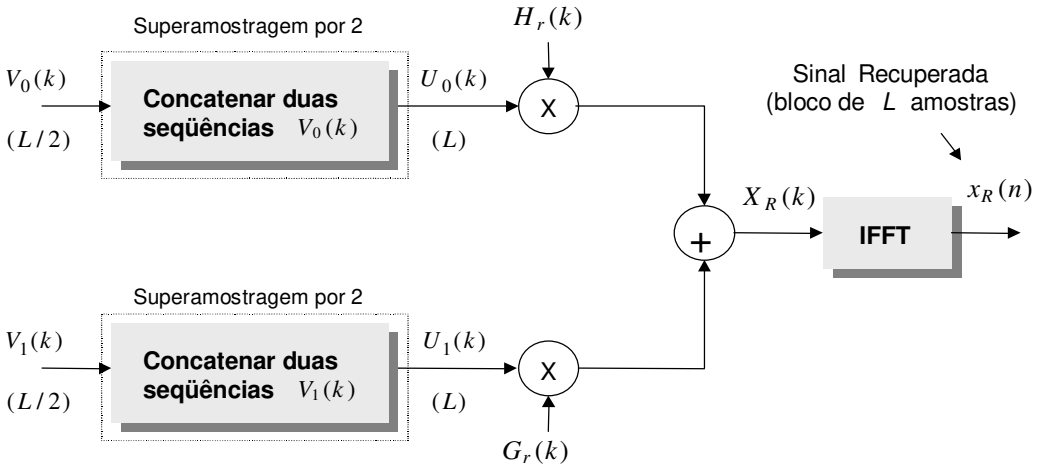


Fig. 4.10. Par de reconstrução passa-baixas e passa-altas utilizando a FFT.

4.4 MODELO PSICO-ACÚSTICO

Em paralelo ao processo de decomposição é executado parcialmente o modelo psicoacústico **MP2** descrito na norma ISO/IEC 11172-3 [7]. Esse modelo recebe como entrada o bloco de amostras $x_{AJ}(n)$ a partir do qual será calculado o limiar de ruído permitido para cada componente do espectro. O procedimento foi descrito no Capítulo 3 e tem como resultado o vetor $thr(k)$.

Nessa parte do trabalho descreve-se o processo utilizado para mapear $thr(k)$ no domínio das subbandas wavelets resultantes da árvore da Fig. 4.4.

O objetivo desse mapeamento é determinar o valor SMR (relação Sinal/Mascaramento) para cada subbanda em que foi decomposto o sinal de entrada. Quanto maior for a relação SMR, maior será o nível de audibilidade da correspondente subbanda, já que isso implica que é percebida com um alto nível de intensidade (potência) ou que é fracamente mascarada pelas componentes vizinhas.

O mapeamento inicia-se particionando $thr(k)$ nas 29 faixas (subbandas) de freqüências (especificadas na Fig. 4.4) que correspondem às 29 subbandas *wavelets*.

Em seguida determina-se o nível mínimo de ruído localizado dentro de cada subbanda e se designa esse valor às outras componentes que formam essa faixa. Dessa forma, todas as componentes de uma mesma subbanda suportarão o mesmo nível de ruído (pior caso).

O resultado final desse procedimento é a densidade espectral de potência do ruído mínimo identificado por $mthr(k)$, onde $k = 0,...512$.

Em seguida, constrói-se um vetor ou seqüência $S(k)$ definido como:

$$S(k) = \begin{cases} mthr(k) & k = 0,...,512 \\ mthr(1024-k) & k = 513,...,1023 \end{cases} \tag{4.7}$$

Logo, calcula-se a IDFT de $S(k)$ (a partir do padrão MP2) :

$$R(n) = \sum_{k=0}^{1023} S(k) e^{\frac{j2\pi nk}{1024}} \tag{4.8}$$

obtendo-se assim a seqüência simétrica $R(n)$ que se chama **função de autocorrelação do ruído**.

Para colocar $R(n)$ no domínio das subbandas *wavelets* decompõe-se a mesma, através da árvore de decomposição mostrada Fig. 4.4. Essa operação, pode ser expressa como:

$$R_{sw}(p,n) = Wt_{p,n}\left(R(m), h_{Td}(m), g_{Td}(m)\right) \tag{4.9}$$

onde:

$$H_T(k) = 2\left|H_d(k)\right|^2 \tag{4.10}$$

$$h_{Td}(n) = \frac{1}{N}\sum_{k=0}^{N-1} H_T(k) e^{\frac{j2\pi nk}{N}} \tag{4.11}$$

$$G_T(k) = 2\left|G_d(k)\right|^2 \tag{4.12}$$

$$g_{Td}(n) = \frac{1}{N}\sum_{k=0}^{N-1} G_T(k) e^{\frac{j2\pi nk}{N}} \tag{4.13}$$

As seqüências $H_d(k)$ e $G_d(k)$ (respostas em freqüência discreta determinadas a partir de (2.30) e (2.88) respectivamente) constituem respectivamente os espectros dos filtros de decomposição passa-baixas e passa-altas utilizados na árvore da Fig. 4.4.

A utilização de filtros da forma $H_T(k) = 2\left|H_d(k)\right|^2$ e $G_T(k) = 2\left|G_d(k)\right|^2$ deve-se ao fato de que se está decompondo um sinal que é Densidade Espectral de Potência do Ruído, expressa no domínio temporal mediante a função de autocorrelação.

Os pares de decomposição utilizados nesse caso são do mesmo esquema que o daquele usado na decomposição do bloco de aúdio de entrada.

Após a decomposição, calcula-se a energia do ruído em cada subbanda *wavelet* utilizando-se a seguinte expressão:

$$\eta_{sw}(p) = \left\|R_{sw}(p,n)\right\| \tag{4.14}$$

a qual indica que a energia do ruído é determinada a partir da norma das amostras que formam uma determinada subbanda "p" da função de autocorrelação decomposta $R_{sw}(p,n)$.

Por outro lado, a energia de cada subbanda *wavelet* resultante do processo de decomposição é calculada da forma mostrada a seguir.

Calcula-se primeiramente a transformada discreta de Fourier (mediante a FFT) de $x_{AJ}(n)$ utilizando-se a seguinte expressão (note-se que esta operação já foi calculada no começo do Modelo Psico-Acústico MP2 descrito no capítulo 3) :

$$X_{AJ}(k) = \frac{1}{1024}\sum_{n=0}^{1023} x_{AJ}(n) e^{-j\frac{2\pi nk}{1024}} \tag{4.15}$$

Em seguida, determina-se a energia de cada componente de freqüência "k" e depois calcula-se a sua transformada inversa de Fourier.

Dessa forma, tem-se:

$$Ex(n) = \sum_{k=0}^{1023} \left| X_{AJ}(k) \right|^2 e^{j\frac{2\pi nk}{1024}} \tag{4.16}$$

Logo faz-se a decomposição de $Ex(n)$ em subbandas *wavelets*, utilizando-se a árvore da Fig. 4.4. Isso pode ser expresso como:

$$Ex_{sw}(p,n) = Wt_{p,n}\left(Ex(m), h_{Td}(m), g_{Td}(m)\right) \tag{4.17}$$

Calcula-se agora a energia contida em cada subbanda *wavelet*, determinando-se a norma das amostras que formam cada uma delas. Isso pode ser expresso como:

$$E'_{sw}(p) = \left\| Ex_{sw}(p,n) \right\| \tag{4.18}$$

Observe-se que nos dois processos de decomposição (ruído e energia), o tamanho do bloco de entrada é de 1024 amostras, portanto os valores $L(p)$ são os mesmos para ambos os casos.

Finalmente, a $SMR_{sw}(p)$ (sempre em valores dB) para cada subbanda é calculada da seguinte forma:

$$SMR_{sw}(p) = 10\log\left(\frac{E'_{sw}(p)}{\eta_{sw}(p)}\right) + C(p) \tag{4.19}$$

onde $C(p)$ é uma constante de compensação que é mapeada da seguinte maneira:

$$\begin{aligned} C(p) &= 0 \qquad p \le 21 \\ C(p) &= -3 \qquad 22 \le p \le 24 \\ C(p) &= -8 \qquad 25 \le p \le 26 \\ C(p) &= -10 \qquad 27 \le p \le 28 \end{aligned} \tag{4.20}$$

Esses valores foram calibrados através de testes de escuta a fim de se obter resultados bastantes satisfatórios nas diferentes taxas de bits requeridas.

Por outro lado, note-se que a maioria das operações pode ser facilmente implementada através da FFT e da IFFT. Esse fato diminui ainda mais a complexidade computacional do codificador.

Os valores SMR_{sw} calculados pelo modelo constituem os parâmetros de entrada do algoritmo de alocação de bits. Baseado nesses valores, o algoritmo determinará a resolução dos quantizadores a serem utilizados na quantização de cada subbanda *wavelet.*

4.5 FATOR DE ESCALA

O fator de escala constitui o máximo valor absoluto das amostras contidas em cada sub-banda *wavelet*. Através desse fator é possível normalizar cada subbanda para valores entre **-1** e **1** com o objetivo de facilitar o processo de quantização. Cada fator de escala é codificado e transmitido separadamente ao decodificador onde será utilizado para escalonar as amostras segundo seus correspondentes valores originais. A utilização do fator de escala se justifica devido ao fato de que o sinal de áudio apresenta uma faixa dinâmica de variação bastante grande, em torno de 96dB.

A codificação do fator de escala é realizada através de um processo de quantização logarítmica cuja curva é mostrada na Fig. 4.11.

A curva foi gerada através de um processo de calibração subjetiva (teste de escuta) baseado na avaliação da distorção mínima no sinal recuperado. A equação da curva é dada por:

$$escalcod(n) = 2^{0.00055(512n - 32768)} \qquad 0 \le n \le 63 \tag{4.21}$$

No processo de codificação de cada fator de escala é utilizado também um valor de normalização $max escal(\ p\ )$ que constitui o máximo valor possível que pode atingir uma amostra dentro de qualquer subbanda.

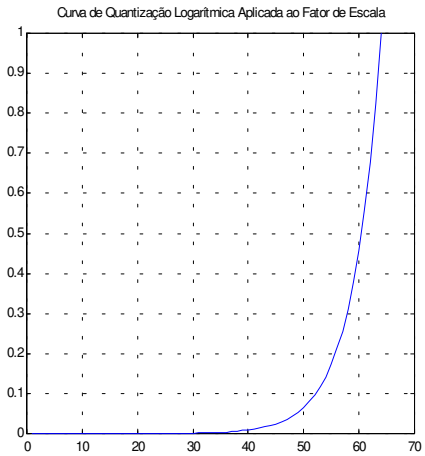


Fig. 4.11. Curva de Quantização Logarítmica Aplicada ao Fator de Escala.

A normalização pode ser expressa como :

$$esca\ln orm(p) = \frac{escal(p)}{max\, escal(p)} \tag{4.22}$$

onde o valor $escal(p)$ constitui o fator de escala da subbanda "p". Por outro lado note-se que os valores resultantes $esclanorm(p)$ são escalonados para a faixa de 0 a 1.

Cada valor $esclanorm(p)$ é quantizado logaritmicamente com 64 níveis de quantização (ou 6 bits por amostra). A quantização pode ser expressa pela seguinte equação:

$$indescal(p) = IQ_{log}\left(esca\ln orm(p)\right) \tag{4.23}$$

onde $IQ_{\log}(x)$ retorna o índice do maior valor mais próximo a "**x**" na curva de quantização. Assim, $0 \le indescal(p) \le 63$.

Os valores da curva de quantização são também conhecidos pelo decodificador e, portanto, é suficiente a transmissão do índice $indescal(p)$ para se realizar a decodificação pertinente ao fator de escala. A curva tem 64 valores disponíveis, o qual significa que o índice poderá ser codificado com 6 bits/amostra.

A decodificação do fator de escala pode ser expressa como:

$$escal_q(p) = escalcod(indescal(p)).max\,escal(p) \tag{4.24}$$

onde $escal_q(p)$ constitui o fator de escala quantizado para a subbanda "p". Esse valor é também calculado no codificador e utilizado para normalizar as amostras antes do processo de quantização.

Por outro lado, após monitorar os fatores de escala de uma longa seqüência de blocos codificados, observou-se que os mesmos apresentam uma alta correlação no domínio temporal, sendo que, para aproveitar essa correlação e ganhar compressão, optou-se por transmitir os índices $indescal(p)$ na forma diferencial em relação ao bloco de áudio anterior.

Isso permitiu a obtenção de diferenças muito pequenas, próximas de zero. Aproveitou-se tal fato codificando-se as diferenças através da codificação de entropia.

O ganho de compressão atingido com esse formato de codificação foi de quase 1:3 e será descrito com mais detalhe na seção seguinte.

4.6 ALOCAÇÃO DE BITS, QUANTIZAÇÃO ESCALAR E CODIFICAÇÃO DE ENTROPIA

O algoritmo de alocação de bits determina a partir dos valores $SMR_{sw}(p)$ o número de níveis de quantização (uniforme, *mid-tread*) que serão utilizados na codificação de cada subbanda.

O processo de alocação é realizado bit a bit e é similar a aquele utilizado nas *layers* 1 e 2 do sistema MPEG-1 [2]. A alocação é feita iterativamente até que o número de bits disponíveis para se codificar um bloco de áudio (N_{BB}) seja ultrapassado.

O valor N_{BB} é calculado a partir da seguinte expressão:

$$N_{BB} = \frac{R_{tx}(L_B - L_{ov})}{f_m} + B_x \tag{4.25}$$

onde R_{tx} é a taxa de transmissão requerida, L_B é o tamanho em número de amostras do bloco de entrada (1024 amostras), L_{ov} é o número de amostras de superposição (48 amostras), f_m é a freqüência de amostragem utilizada (para esse caso : 44.1 KHz), enquanto que B_x constitui o espaço (em numero de bits) disponível no *buffer* de saída, cuja função é a de manter constante a taxa de bits requerida.

Para tornar o processo de alocação mais eficiente divide-se o mesmo em duas etapas, sendo que a primeira denomina-se **alocação primária de bits**, enquanto que a segunda é chamada de **alocação delta de bits**. Os algoritmos utilizados em ambas as etapas, assim como o processo de codificação de entropia serão descritos a seguir nas diversas subseções.

4.6.1 ALOCAÇÃO PRIMÁRIA DE BITS

A alocação primária de bits é realizada antes da codificação de entropia. Nessa etapa será alocada a maior quantidade de bits segundo os valores SMR_{sw} calculados para cada subbanda *wavelet*.

A alocação é baseada na minimização do parâmetro MNR (*Masking to Noise ratio*), definido como:

$$MNR_{sw}(p) = SNR_{sw}(p) - SMR_{sw}(p) \tag{4.26}$$

onde $SNR_{sw}(p)$ constitui a relação sinal-ruído da subbanda *"p"* (*signal to noise ratio*). Essa relação é incrementada em 6dB (caso a quantização seja escalar) quando se acrescenta ao processo de quantização um bit de alocação.

No início do algoritmo, a SNR_{sw} de cada subbanda é levada ao valor zero. Assim, tem-se também disponível o vetor $n_{bs}(p)$ (que se inicia em zero) o qual vai contabilizando o número de bits/amostra que vai sendo alocado a cada subbanda " p ". Em seguida, determina-se a subbanda com menor valor MNR_{sw} e logo acrescenta-se a esta um bit ($n_{bs}(p) = n_{bs}(p) + 1$) e 6dB a sua correspondente SNR_{sw} ($SNR_{sw}(p) = SNR_{sw}(p) + 6$). É importante indicar que para a primeira alocação do algoritmo, o número de bits acrescentados a $n_{bs}(p)$ é igual a 2 e, portanto, são somados 12dB à correspondente SNR_{sw}. Isso, no entanto, é aplicado às subbandas na faixa de SB0...SB24, as quais aceitam um número mínimo de 2bits de alocação. Para as subbandas SB25, SB26, Sb27 e SB28 aceita-se como alocação mínima um bit por amostra utilizando-se para esse caso a quantização vetorial.

Posteriormente, calcula-se novamente os valores MNR_{sw} para cada subbanda e repete-se o mesmo procedimento até que o número de bits alocados N_{ba} ultrapasse o valor N_{BB}. Dessa forma, N_{ba} deve ser calculado em cada iteração utilizando-se a seguinte expressão:

$$N_{ba} = n_{ba} + n_{bfs} + \sum_{p=0}^{28} n_{be}\left(n_{bs}(p)\right) + \sum_{p=0}^{28} n_{bs}(p)L(p) \tag{4.27}$$

onde $n_{ba} = 4 \times 29 = 116$, e representa o número de bits utilizados na codificação do vetor $n_{bs}(p)$. Por outro lado, $n_{bfs} = 6 \times 29 = 174$ e representa o número de bits utilizados na codificação do fator de escala sem usar ainda a codificação entrópica das diferenças com respeito ao quadro (*frame*) anterior.

Os valores $n_{be}\left(n_{bs}(p)\right)$ constituem o número de bits utilizado na identificação (códigos de 4 bits) dos *codebooks* Huffman a serem utilizados na codificação entrópica da subbanda "p".

A codificação de entropia é aplicada somente se a correspondente subbanda é quantizada escalarmente. Isso vai depender do número de bits que foi alocado à respectiva subbanda. Os valores $n_{be}\left(n_{bs}(p)\right)$ portanto são condicionados ao mapeamento mostrado na Tabela 4.2.

Observe-se que as subbandas na faixa SB22-SB24 são quantizadas vetorialmente quando o número de bits alocado às mesmas é maior do que 4. A Tabela 4.2 mostra, portanto, as condições que devem ser satisfeitas para a aplicação da codificação de entropia.

É importante se mencionar também que quando o número de bits alocados a qualquer subbanda é igual a 2, aplica-se a quantização escalar junto com a codificação de entropia. No entanto, para esse caso considera-se $n_{be}\left(n_{bs}(p)\right) = 0$ devido a que tem-se disponível um único *codebook* Huffman que não precisa de identificação.

TABELA 4.2 MAPEAMENTO DO VALORES $n_{be}\left(n_{bs}(p)\right)$

Condição	$n_{be}\left(n_{bs}(p)\right)$
$3 \le n_{bs}(p) \le 7, \quad 11 \le p \le 21$	**4**
$n_{bs}(p) = 3 \quad 22 \le p \le 24$	**4**
Outra forma	**0**

Logicamente, o resultado final do algoritmo de alocação é o vetor de $n_{bs}(p)$, o qual é composto por 29 códigos binários de 4 bits que são transmitidos ao decodificador através dos

campos de informação. Cada código pode especificar desde 0 até 15 bits/amostra. O número de níveis de quantização $N_{nq}(p)$ utilizado na sub-banda " p " é finalmente dado por:

$$N_{nq}(p) = 2^{n_{bs}(p)} - 1 \tag{4.28}$$

A equação (4.28) especifica um número ímpar de níveis de quantização que asseguram a mesma resolução (em níveis de quantização) para a parte positiva e a parte negativa de cada subbanda.

Foi observado também que os valores $n_{bs}(p)$ resultantes para cada bloco de áudio são altamente correlacionados no domínio temporal. Por esse motivo optou-se também por codificar as diferenças desse vetor, com respeito ao bloco de áudio anterior, utilizando-se a codificação de entropia. Esse processo permitiu um ganho de compressão adicional em torno de 1:3 e será descrito com maior detalhe na seção 4.6.3.

Após a alocação primária de bits, as subbandas que satisfazem as condições da Tabela 4.2, são quantizadas escalarmente mediante o procedimento que será descrito na seção seguinte.

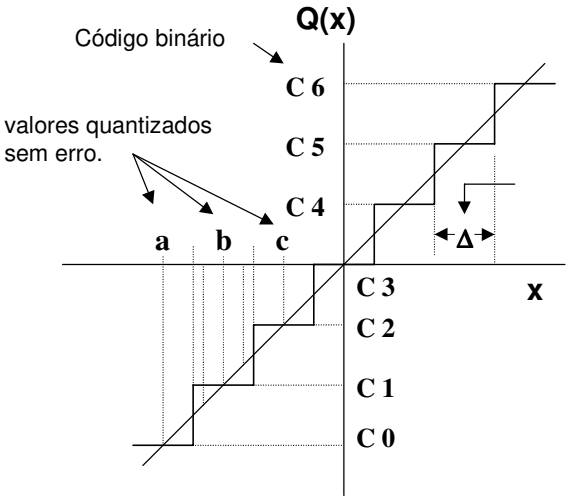
4.6.2 QUANTIZAÇÃO ESCALAR

A quantização escalar de uma amostra $x_{sw}(p,n)$ que forma uma determinada subbanda $"p"$ pode ser expressa da seguinte forma:

$$x_{sw}^{q}(p,n) = round\left(\left(2^{n_{bs}(p)-1} - 1\right) \left[\frac{x_{sw}(p,n)}{escal_q(p)} \right] \right) \tag{4.29}$$

onde a função *round* retorna o valor inteiro mais próximo, enquanto que $x_{sw}^{q}(p,n)$ constitui o valor da amostra quantizada.

De acordo com (4.28) pode-se observar que o processo de quantização é do tipo *mid-tread*, uniforme e simétrico. Essa função de quantização é mostrada na Fig. 4.12., onde para esse caso está-se considerando $N_{nq} = 7$ e $n_{bs}(p) = 3$ (3 bits por amostra).



Formato de quantização uniforme, mid-tread e simétrico.
Neste caso: 7 níveis de quantização e 3 bits por amostra.

Fig. 4.12. Formato de quantização escalar aplicado às subbandas wavelets.

Por outro lado, o processo de dequantização de amostras no decodificador pode ser expresso como:

$$\hat{x}_{sw}(p,n) = escal_q(p)\left[\frac{x^q_{sw}(p,n)}{\left(2^{n_{bs}(p)-1}-1\right)}\right] \tag{4.30}$$

onde $\hat{x}_{sw}(p,n)$ constitui a n-ésima amostra dequantizada da subbanda "p".

4.6.3 CODIFICAÇÃO DE ENTROPIA

O processo de codificação de entropia é projetado como sendo a codificação das amostras quantizadas escalarmente utilizando-se famílias de códigos de Huffman que minimizem a quantidade de bits a ser utilizada na representação da informação de áudio.

Esse processo explora a redundância estatística da informação a se transmitir, designando mais bits a valores de amostras que têm menor probabilidade de ocorrência e menos bits a aqueles que têm maior probabilidade de ocorrência.

O processo de geração de códigos de Huffman é descrito a seguir. Para esse caso está-se considerando 6 possíveis mensagens ou valores de amostras para serem codificados, onde cada um deles constitui um nó de entrada na árvore da Fig. 4.13.

O primeiro passo de construção do código consiste em se escolher as mensagens com a mais baixa probabilidade de ocorrência. Nesse caso, serão as mensagens a_4 e a_6. Depois somam-se as probabilidades deles para se gerar o nó a_7 designando o valor de "1" a um ramo e o valor de "0" ao outro . A ordem de designação afeta a codificação, mas não a taxa de bits a se transmitir. Considere-se a partir de agora as duas mensagens a_4 e a_6 como uma só mensagem a_7 com probabilidade $P_7 = \frac{1}{16}$.

O segundo passo consiste em se escolher as duas mensagens com menor probabilidade de ocorrência a partir das mensagens a_1, a_2, a_3, a_5 e a_7. Nesse caso, serão as mensagens a_3 e a_7. Então somam-se as probabilidades deles para formar um novo nó a_8 com probabilidade $P_8 = \frac{5}{32}$. Logo se associa um "1" a um ramo e um "0" ao outro. O processo continua até se obter um nó com probabilidade igual a um.

Finalmente, a codificação é feita seguindo-se os ramos como se estivesse voltando, isto é, partindo do nó com probabilidade 1 até se chegar à mensagem desejada. Assim, por exemplo, a mensagem a_5 será codificada através do código : 101.

A comparação da taxa média de bits (no caso do exemplo anterior) obtida mediante a codificação com palavras de código uniforme (quantização escalar uniforme) e com a codificação de *Huffman* é mostrada a seguir:

- Codificação com palavras de código uniforme : **3 bit/ mensagem**
- Codificação de Huffman :

$$\frac{5}{8}.1 + \frac{3}{32}.3 + \frac{3}{32}.3 + \frac{1}{32}.4 + \frac{1}{8}.3 + \frac{1}{32}.4 = \frac{29}{16} = 1.813 \frac{\text{bits}}{\text{mensagem}}$$

Observe-se que para esse caso a taxa de compressão foi de aproximadamente 40%. No entanto, se a distribuição de probabilidade mantém seu valor médio em torno de zero e se torna menos uniforme, então taxas maiores de compressão poderão ser alcançadas. Isso acontece quando as amostras de uma determinada subbanda são quantizadas com poucos bits, permitindo assim a obtenção de uma grande quantidade de valores iguais a zero.

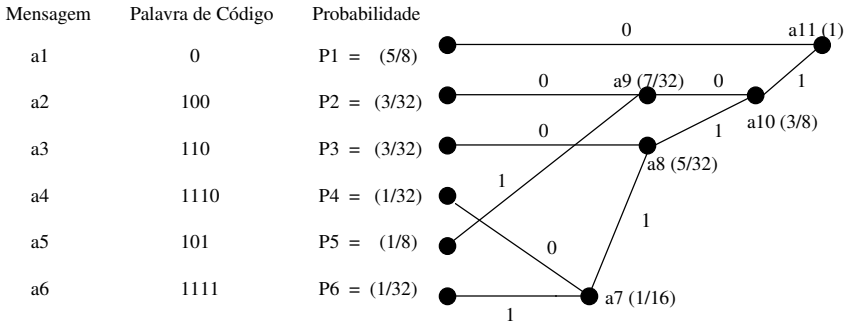


Fig. 4.13. Geração de códigos de Huffman.

Por outro lado, quando o número de bits alocados a uma subbanda é incrementado, o histograma correspondente tende a ser mais uniforme e esse fato reduz o desempenho da codificação de entropia. A aplicação desse formato de codificação estará portanto sujeito a certas condições, que são mapeadas de acordo com o que mostra a Tabela 4.3.

TABELA 4.3. CONDIÇÕES PARA A APLICAÇÃO DA CODIFICAÇÃO DE ENTROPIA, DA QUANTIZAÇÃO ESCALAR E DA QUANTIZAÇÃO VETORIAL.

Condição	Codificação de Entropia	Quantização Escalar	Quantização Vetorial
$n_{bs}(p) = 2, \quad 0 \le p \le 24$	sim	sim	não
$3 \le n_{bs}(p) \le 15, \quad 0 \le p \le 10$	não	sim	não
$3 \le n_{bs}(p) \le 7, \quad 11 \le p \le 21$	sim	sim	não
$n_{bs}(p) \ge 8, \quad 11 \le p \le 21$	não	sim	não
$n_{bs}(p) = 3, \quad 22 \le p \le 24$	sim	sim	não
$n_{bs}(p) \ge 4, \quad 22 \le p \le 24$	não	não	sim
$n_{bs}(p) \ge 1, \quad 25 \le p \le 28$	não	não	sim

Para a geração dos códigos utilizados no codificador proposto foi necessário o desenvolvimento de um algoritmo baseado no procedimento mostrado na Fig. 4.13.

As distribuições de probabilidade utilizadas para esse caso foram escolhidas de acordo com a forma dos histogramas apresentados por subbandas quantizadas com níveis distintos de resolução. Encontrou-se assim que as funções de densidade de probabilidade Gaussiana, Laplaciana e Triangular, são as que melhor se aproximam da maioria dos histogramas monitorados.

Essas funções são definidas por expressões matemáticas $p_x(x)$ que são mostradas na Tabela 4.4.

TABELA 4.4. DISTRIBUIÇÕES DE PROBABILIDADE UTILIZADAS NA CONSTRUÇÃO DOS CÓDIGOS DE HUFFMAN

Nome	**Notação**	$p_x(x)$
Uniforme	U $\left(\sigma_x^2 = \Delta^2/12\right)$	$\frac{1}{\Delta}; \quad x \in \left(-\frac{\Delta}{2}, \frac{\Delta}{2}\right)$ 0; outra forma
Gaussiana ou Normal	G $N\left(0, \sigma_x^2\right)$	$\frac{1}{\sqrt{2\pi\sigma_x^2}} e^{\left[\frac{-x^2}{2\sigma_x^2}\right]}$
Laplaciana	L $L\left(0, \sigma_x\right)$	$\frac{1}{\sqrt{2}\sigma_x} e^{\left[\frac{-\sqrt{2}\lvert x\rvert}{\sigma_x}\right]}$
Triangular	T $T(a,b)$	$b - a\lvert x\rvert \quad x \in \left(-\frac{b}{a}, \frac{b}{a}\right)$

Para a construção dos códigos considera-se a variável "x" como sendo discreta, já que a codificação de entropia será aplicada a amostras cujos valores são também discretos.

Por outro lado, cada família de códigos de Huffman foi construída a partir de uma determinada função de probabilidade e de acordo com um valor de alocação $n_{bs}(p)$. Essa distribuição é mostrada na Tabela 4.5.

TABELA 4.5 DISTRIBUIÇÕES DE PROBABILIDADE UTILIZADAS SEGUNDO A ALOCAÇÃO DE BITS

Alocação de Bits	Distribuição Gaussiana	Distribuição Laplaciana	Distribuição Triangular
$n_{bs}(p) = 2$	não	não	$T(0.25,0.5)$
$n_{bs}(p) = 3$	$N(0,1.0)$ $N(0,1.5)$ $N(0,2.0)$	não	$T(0.013,0.16)$
$n_{bs}(p) = 4$	$N(0,1.0)$ $N(0,1.5)$ $N(0,2.5)$ $N(0,4.0)$	$L(0,1.0)$ $L(0,2.0)$ $L(0,3.0)$ $L(0,4.0)$	$T(0.018,0.13)$ $T(0.016,0.12)$ $T(0.008,0.09)$ $T(0.005,0.08)$
$n_{bs}(p) = 5$	$N(0,1.0)$ $N(0,2.0)$ $N(0,3.0)$ $N(0,3.5)$	$L(0,2.0)$ $L(0,2.5)$ $L(0,4.0)$ $L(0,5.0)$	$T(0.0042,0.062)$ $T(0.0019,0.047)$
$n_{bs}(p) = 6$	$N(0,1.0)$ $N(0,2.0)$ $N(0,3.0)$ $N(0,3.5)$	$L(0,2.0)$ $L(0,3.0)$ $L(0,4.0)$ $L(0,5.0)$	$T(0.0010,0.031)$ $T(0.0009,0.030)$ $T(0.0006,0.026)$
$n_{bs}(p) = 7$	$N(0,1.0)$ $N(0,2.0)$ $N(0,3.0)$ $N(0,3.5)$	$L(0,2.0)$ $L(0,3.0)$ $L(0,4.0)$ $L(0,5.0)$	$T(0.0002,0.016)$

Cada família de códigos de Huffmam para uma determinada alocação é identificada através de um código de 4 bits, o qual é também transmitido ao decodificador.

A título de exemplo, apresenta-se nas Figuras 4.14, 4.15, e 4.16, a forma das distribuições utilizadas para uma alocação de 4 bits por amostra.

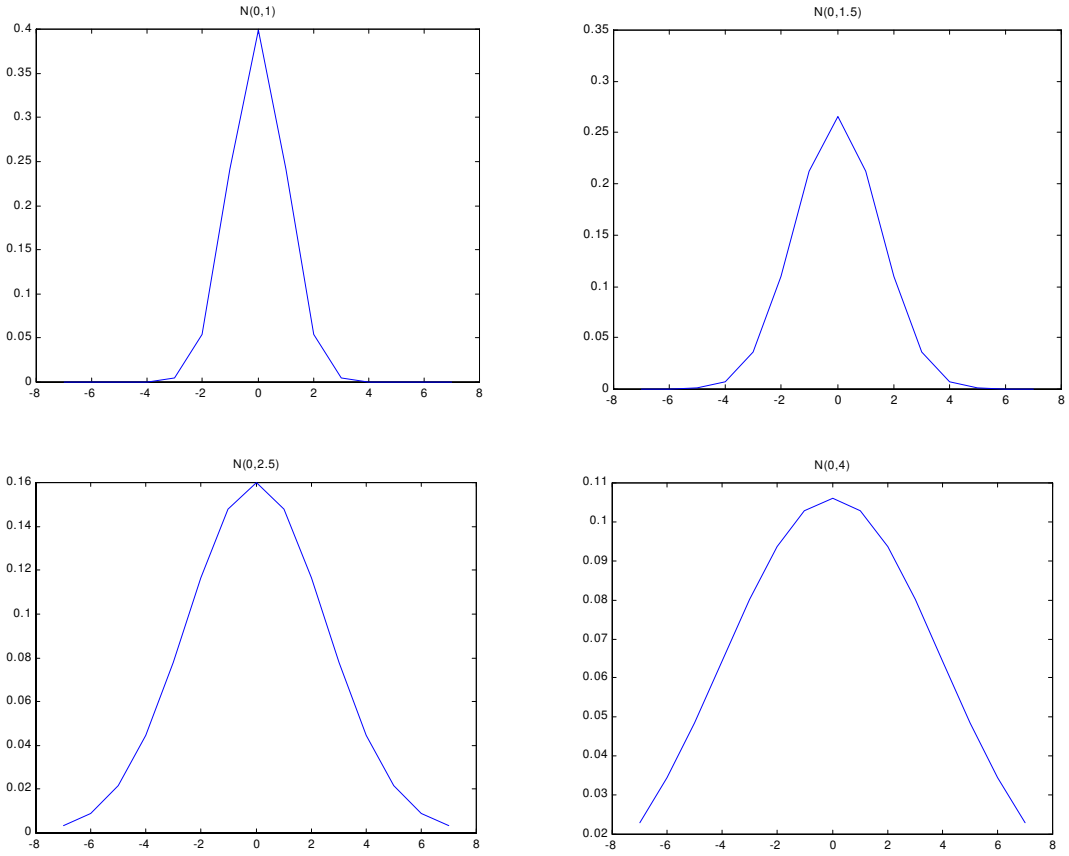


Fig. 4.14. Distribuições Gaussianas utilizadas na construção de códigos de Huffmam para uma alocação de 4 bits por amostra.

Para se determinar a família de códigos que minimizará o número de bits que serão utilizados na codificação de uma determinada subbanda, fator de escala ou nos valores de alocação $n_{bs}(p)$, é utilizado o seguinte procedimento:

1. Calcula-se o histograma H_i das amostras a serem codificadas
2. Executa-se a seguinte equação matricial:

$$E_D = M_H H_i^T \tag{4.31}$$

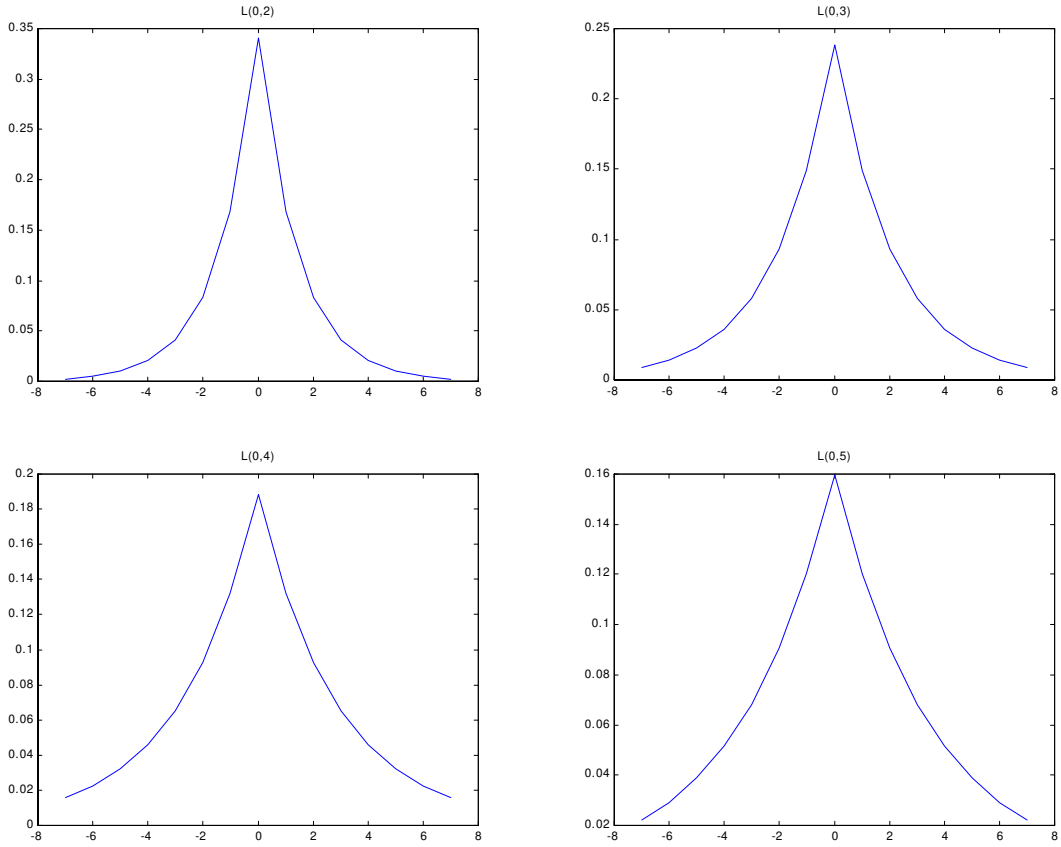


Fig. 4.15. Distribuições Laplacianas utilizadas na construção de códigos de Huffmam para uma alocação de 4 bits por amostra.

onde:

$$M_H = \begin{bmatrix} D_1\left(-\left(2^{n_{bs}(p)-1}-1\right)\right) & D_1\left(-\left(2^{n_{bs}(p)-1}-2\right)\right) & \ldots & D_1(0) & \ldots & D_1\left(2^{n_{bs}(p)-1}-2\right) & D_1\left(2^{n_{bs}(p)-1}-1\right) \\ D_2\left(-\left(2^{n_{bs}(p)-1}-1\right)\right) & D_2\left(-\left(2^{n_{bs}(p)-1}-2\right)\right) & \ldots & D_2(0) & \ldots & D_2\left(2^{n_{bs}(p)-1}-2\right) & D_2\left(2^{n_{bs}(p)-1}-1\right) \\ : & : & : & : & : & : & : \\ D_m\left(-\left(2^{n_{bs}(p)-1}-1\right)\right) & D_m\left(-\left(2^{n_{bs}(p)-1}-2\right)\right) & \ldots & D_m(0) & \ldots & D_m\left(2^{n_{bs}(p)-1}-2\right) & D_m\left(2^{n_{bs}(p)-1}-1\right) \end{bmatrix} \tag{4.32}$$

$$H_i^T = \begin{bmatrix} N_a\left[-\left(2^{n_{bs}(p)\text{-}1}-1\right)\right] \\ N_a\left[-\left(2^{n_{bs}(p)\text{-}1}-1\right)+1\right] \\ \vdots \\ \vdots \\ 0 \\ \vdots \\ \vdots \\ N_a\left[\left(2^{n_{bs}(p)\text{-}1}-1\right)-1\right] \\ N_a\left[\left(2^{n_{bs}(p)\text{-}1}-1\right)\right] \end{bmatrix} \tag{4.33}$$

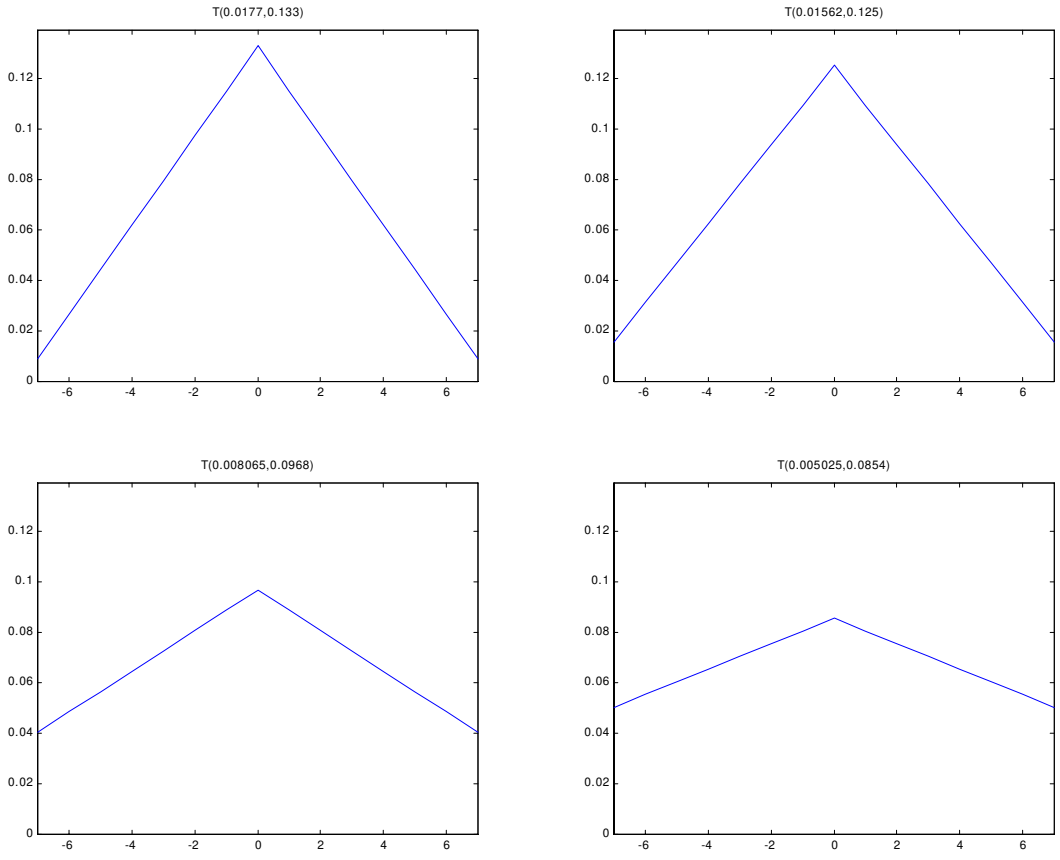


Fig. 4.16. Distribuições Triangulares utilizadas na construção de códigos de Huffmam para uma alocação de 4 bits por amostra.

Dessa forma, a função $D_j(x)$ retorna o número de bits que forma código de Huffman designado ao valor "x". Especifica também que o código mencionado ou o conjunto de códigos

em questão foram gerados utilizando-se a distribuição "j". Esse índice é portanto o identificador de uma família de códigos utilizada no processo de codificação de entropia.

Por outro lado, a função $N_a(y)$ retorna o número de amostras com valor "y" que fazem parte da seqüência a ser codificada. Essa função logicamente constitui a principal ferramenta para o cálculo do histograma.

3. Determina-se o índice da componente de menor valor do vetor resultante $E_D(d_1, d_2, ...d_j)$. Isso pode ser expresso como:

$$i = iden\,min\left[E_D(d_1, d_2, ...d_j)\right] \tag{4.34}$$

onde, "i" constitui o índice de identificação da família de códigos Huffmam que minimizará a quantidade de bits utilizados na codificação da seqüência de entrada.

- **CODIFICAÇÃO DO FATOR DE ESCALA:**

  Como mencionado anteriormente, os índices de identificação do fator de escala $indescal(p)$ são codificados entropicamente e na forma diferencial com respeito ao bloco de áudio anterior. Essas diferenças podem ser expressas como:

$$indescal_D(t, p) = indescal(t, p) - indescal(t-1, p) \tag{4.35}$$

  Em (4.35), os índices "t" e "$t-1$" referem-se, respectivamente, ao bloco atual e ao bloco anterior de áudio codificado.

  As diferenças $indescal_D(t, p)$ são codificadas entropicamente através do conjunto de códigos Huffman construídos para $n_{bs}(p) = 6$. Um histograma dessas diferenças é mostrado na Fig. 4.17. Note-se que a forma do mesmo justifica a utilização da codificação de entropia.

- **CODIFICAÇÃO DO VETOR DE ALOCAÇÃO $n_{bs}(p)$:**

  O vetor de alocação é também transmitido na forma diferencial e codificado entropicamente. As diferenças para esse caso são dadas por:

$$n_D(t, p) = n_{bs}(t, p) - n_{bs}(t-1, p) \tag{4.36}$$

Para esse caso o conjunto de códigos Huffman utilizados foi aquele construído para $n_{bs}(p)=5$. O histograma das diferenças é mostrado na Fig. 4.18, onde se pode observar a alta correlação temporal do vetor $n_{bs}(p)$.

Fig. 4.17. Histograma dos valores diferenças $indescal_D(t,p)$

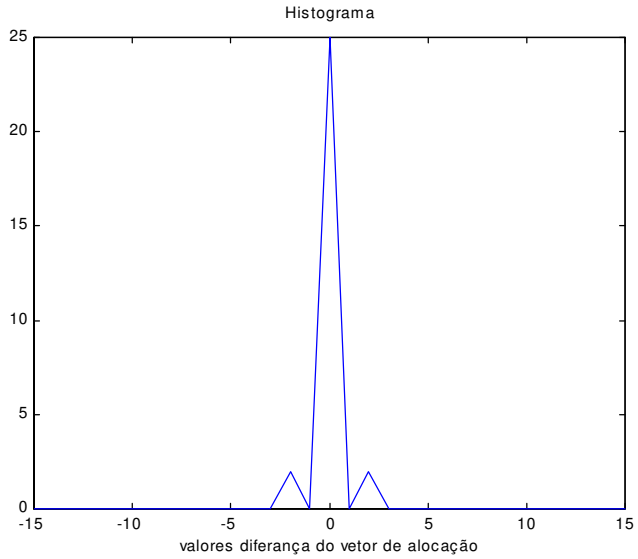


Fig. 4.18. Histograma dos valores diferenciais $n_D(t,p)$**.**

Em todos os casos (subbandas, fator de escala e vetor de alocação) o procedimento de codificação é realizado via histogramas.

4.6.4 ALOCAÇÃO DELTA DE BITS

Após o processo de codificação de entropia, calcula-se novamente o número de bits utilizado na codificação do bloco atual de áudio. Para esse caso utiliza-se a seguinte expressão:

$$N'_{ba} = n'_{ba} + n'_{bfs} + \sum_{p=0}^{28} n_{be}\left(n_{bs}(p)\right) + \sum_{p=0}^{28} n_{ev}(p) \tag{4.37}$$

Em (4.37), n'_{ba} constitui o número de bits utilizado na codificação entrópica diferencial do vetor de alocação $n_{bs}(p)$; n'_{bfs} é o número de bits utilizado na codificação entrópica diferencial do fator de escala, enquanto que $n_{ev}(p)$ especifica o número de bits utilizado na codificação entrópica da subbanda "p" ou na identificação dos vetores utilizados, caso a codificação seja via quantização vetorial (ver equação (4.39)).

A alocação delta de bits constitui uma pequena alocação adicional que tem por objetivo melhorar a SNR_{sw} daquelas subbandas que ainda estejam com certos níveis de distorção. Isso pode ser conseguido graças ao fato de que a codificação de entropia e a quantização vetorial permitem diminuir consideravelmente o número de bits utilizados na representaçao da informação de áudio via alocação primária. Portanto, no início da alocação delta de bits se tem:

$$N'_{ba} < N_{BB} \tag{4.38}$$

Dessa forma, a partir dos valores de SNR_{sw} alcançados até agora, para cada subbanda é executado novamente o algoritmo de alocação primária. Dessa vez, no entanto para cada iteração calculam-se os códigos de Huffman adequados para a codificação da subbanda "p", cujo valor de alocação será incrementado. O codificador poderá atualizar o valor N'_{ba} através de (4.37).

Note-se que para cada iteração devem ser atualizados os valores n'_{ba} , $n_{be}\left(n_{bs}(p)\right)$ e $n_{ev}(p)$. O algoritmo termina quando finalmente $N'_{ba} > N_{BB}$.

Os resultados de alocação conseguidos para cada subbanda através do algoritmo primário e do algoritmo de alocação delta são mostrados na Fig. 4.19. Observe-se que graças à codificação de entropia e à quantização vetorial, consegue-se incrementar os níveis de alocação e, portanto, melhorar o desempenho do codificador no que se refere a qualidade por taxa de bits.

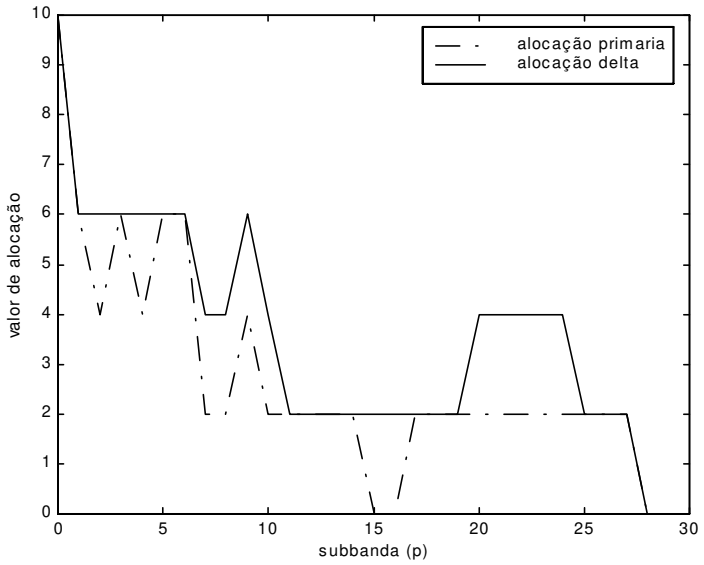


Fig. 4.19. Valores resultantes do vetor $n_{bs}(p)$ após a alocação primária e da alocação delta de bits.

4.7 QUANTIZAÇÃO VETORIAL

No capítulo 3 foram detalhados os principais aspectos que descrevem a teoria de quantização vetorial bem como os procedimentos utilizados na construção dos vetores quantizadores. Foi também dito que essa técnica de codificação resulta ser muito apropriada para a quantização de subbandas na faixa de 7KHz a 22 KHz, onde o ouvido humano apresenta menor sensibilidade para a detecção dos níveis de ruído introduzidos.

As subbandas, portanto, que podem ser submetidas a esse formato de quantização, são : a SB22, a SB23, a SB24, a SB25, a SB26, a SB27 e a SB28. Nas três primeiras será utilizado um esquema de quantização de maior resolução pois suportam menores níveis de ruído.

O formato de quantização utilizado é baseado primeiramente na segmentação em blocos (ou vetores de amostras) da seqüência (ou subbanda) a ser codificada. Logo, dependendo do sinal das amostras que formam um determinado bloco utiliza-se aquela árvore de quantização cujos vetores quantizadores são do mesmo tamanho e do mesmo sinal daquele que está sendo quantizado. Como foi visto no Capítulo 3, as árvores foram construídas em formato binário para tornar mais eficiente o algoritmo de procura dos vetores quantizadores.

Após o procedimento em que é identificada a árvore de procura binária, prossegue-se com a determinação do vetor quantizador que melhor se casa com o bloco de entrada. Para isso em cada par de decisão utiliza-se o algoritmo NN modificado que foi descrito no Capítulo 3. O procedimento é mostrado na Fig. 4.20.

Cada árvore é formada por várias etapas de procura, sendo que a precisão na quantização de um determinado bloco aumenta à medida que se utiliza um maior número de etapas (Fig. 4.21). Essa precisão é determinada assim pelo número de bits alocado à subbanda correspondente.

Como foi visto no Capítulo 3, foram construídas 16 árvores de sinal para vetores de 4 amostras e 4 árvores para vetores de 2 amostras. Essas últimas são utilizadas na quantização das subbandas SB22, SB23 e SB24 devido a que necessitam de maior resolução. O outro grupo de árvores é utilizado nas subbandas SB25, SB26, SB27 e SB28.

Para se determinar o número de bits a ser utilizado na identificação dos vetores quantizadores calculados para uma determinada subbanda $"p"$ utiliza-se a seguinte expressão:

$$N_V(p) = \begin{cases} \dfrac{L(p)}{n_{BL}(p)}.[B_{TR}(p) + n_{et}(p)] & \begin{matrix} 4 \le n_{bs}(p) \le 7, & 22 \le p \le 24 \\ 3 \le n_{bs}(p) \le 5, & 25 \le p \le 28 \end{matrix} \\ 320 & n_{bs}(p) \ge 8 \quad 22 \le p \le 24 \\ 256 & n_{bs}(p) \ge 6 \quad 25 \le p \le 28 \\ 96 & n_{bs}(p) = 2 \quad 25 \le p \le 28 \\ 64 & n_{bs}(p) = 1 \quad 25 \le p \le 28 \end{cases} \tag{4.39}$$

onde $n_{BL}(p)$ constitui o tamanho em número de amostras dos blocos a ser quantizado na subbanda $"p"$; $B_{TR}(p)$ constitui por outro lado, o número de bits do código de identificação da árvore de procura a ser utilizado, enquanto que $n_{et}(p)$ constitui o número de etapas da árvore que foi utilizado no processo de busca. Esse valor depende linearmente do número de bits alocado à subbanda corrrespondente.

Para o caso das subbandas SB22, SB23 e SB24 $(22 \le p \le 24)$ tem-se: $n_{BL}(p) = 2$, $B_{TR}(p) = 2$ (4 árvores de sinal disponíveis para esse caso) e $n_{et}(p) = n_{bs}(p) + 1$ para $n_{bs}(p) \ge 4$.

Assim, para SB25, SB26, SB27 e SB28 $(25 \le p \le 28)$, tem-se: $n_{BL}(p) = 4$, $B_{TR}(p) = 4$ (16 árvores de sinal disponíveis para esse caso) e $n_{et}(p) = n_{bs}(p) - 1$ para $n_{bs}(p) \ge 3$. Para o caso

em que $n_{bs}(p)=1$ e $n_{bs}(p)=2$, foi construída uma árvore de 2 etapas através do mesmo procedimento usado nos casos anteriores (vetores de 4 componentes), mas sem respeitar o sinal das componentes. Portanto, cada vez que se especifica um ou dois bits de alocação para SB25, SB26, SB27 ou SB28 procede-se a dividir a subbanda correspondente em blocos de 4 amostras e realiza-se, para cada bloco, um procedimento de procura binária de duas etapas para o caso de um bit de alocação e de três etapas para o caso de dois bits de alocação. Com esse procedimento se determinará o vetor quantizador que melhor se casa com o bloco de entrada.

Pode-se conferir por outro lado que o número total de bits utilizados para identificar os vetores quantizadores para ambos os casos é, respectivamente, de 64 (um bit de alocação, 50% de compressão em comparação à quantização escalar) e de 96 bits (62.5% de compressão em comparação à quantização escalar).

Nas Figs. 4.22 e 4.23 mostram-se o desempenho em número de bits da quantização vetorial em comparação com a quantização escalar para diversos valores de alocação nos dois grupos de subbandas. Observe-se que à medida que a alocação é incrementada, o desempenho da quantização vetorial também melhora no que se refere a maior ganho de compressão.

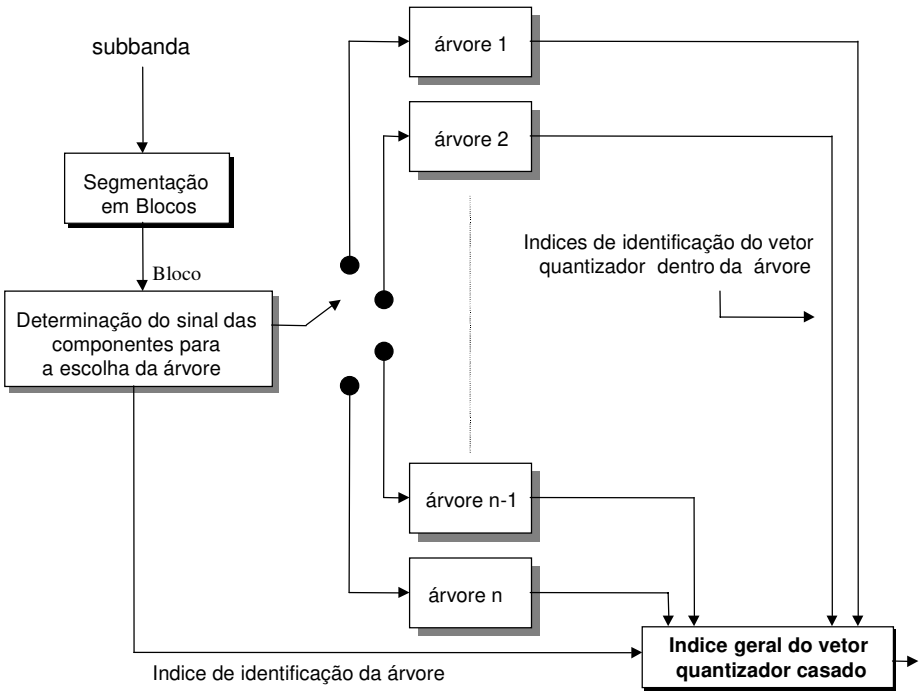


Fig. 4.20. Diagrama de blocos do processo de quantização vetorial.

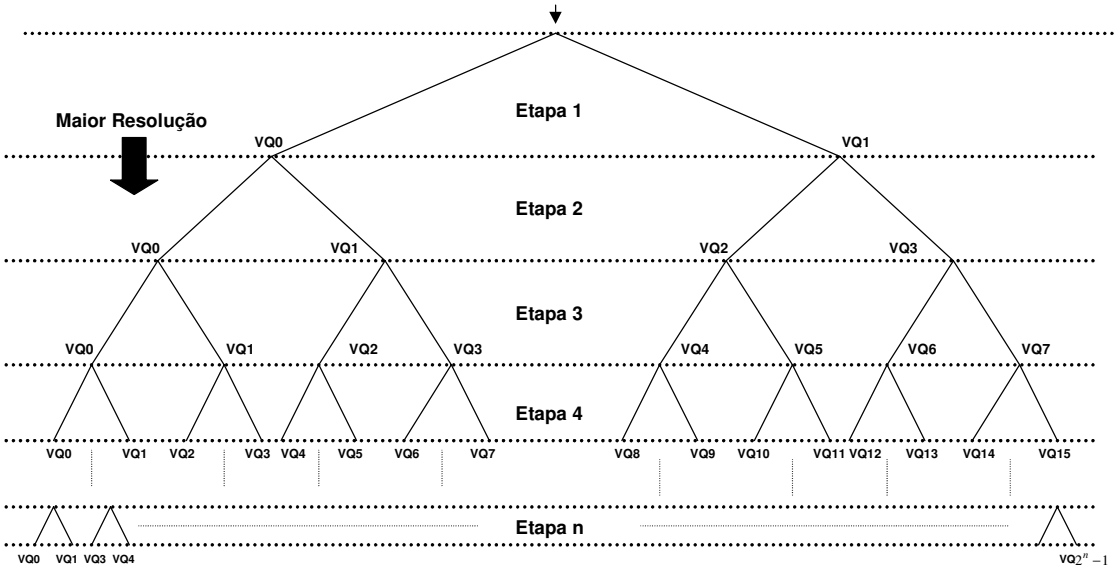


Fig. 4.21. Árvore de procura binária utilizada no quantizador vetorial.

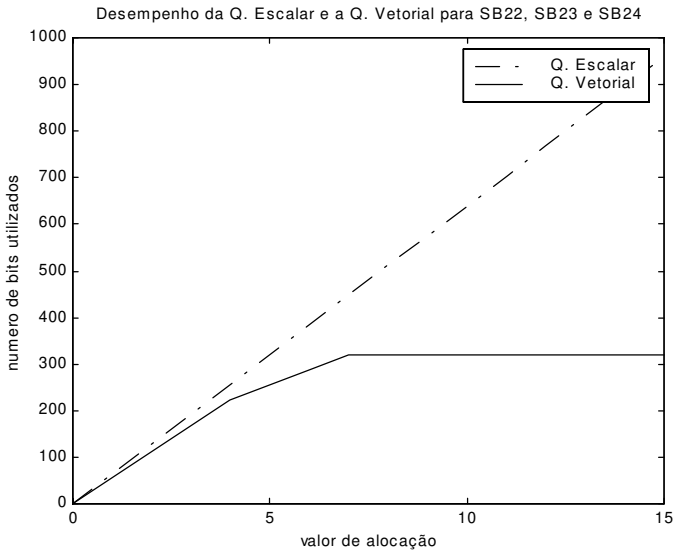


Fig. 4.22. Desempenho da Quantização Escalar e da Quantização Vetorial para diversos valores de alocação em SB22, SB23 e SB24.

Note-se também que a partir de um certo número de bits alocado, a precisão do quantizador vetorial torna-se constante, já que os valores *SNR* conseguidos a partir desse ponto permitem uma

distorção quase totalmente inaudível. Portanto, um incremento dessa precisão não modificará significativamente a percepção de maior ou menor distorção.

A título de exemplo, apresenta-se na Fig. 4.24 a variação da relação sinal-ruído para diversos valores de alocação especificados para a subbanda SB24. Geralmente, para esse grupo de subbandas, o incremento da SNR para cada bit alocado encontra-se na faixa de 2 a 3dB, enquanto que para o grupo SB25, SB26, SB27 e SB28 encontra-se na faixa de 1 a 2 dB. Aparentemente, poder-se-ia concluir que o processo de quantização vetorial introduziria uma alta distorção na subbanda a ser quantizada. No entanto, na maioria dos casos, pelo fato de que as subbandas em questão agrupam componentes de alta freqüência, a distorção muitas vezes permanece inaudível.

Como mencionado no Capítulo3, cada vetor quantizador que faz parte de uma árvore tem seus valores escalonados na faixa de zero a um. Portanto, cada subbanda de entrada do quantizador é primeiramente normalizada através de seu fator de escala correspondente.

Por outro lado, através do formato de procura binária, o número de multiplicações ou somas envolvidas no cálculo de todos os vetores quantizadores para uma determinada subbanda é dado por :

$$N_{MUL/AD}(p) = 2.L(p).n_{et}(p) \tag{4.40}$$

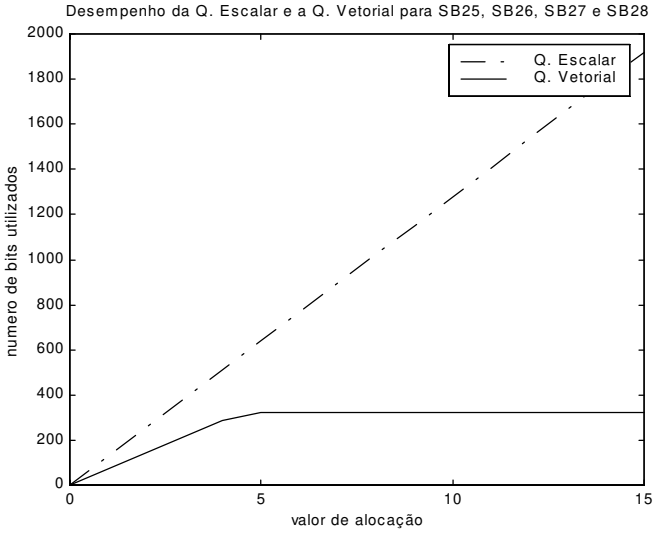


Fig. 4.23. Desempenho da Quantização Escalar e da Quantização Vetorial para diversos valores de alocação em SB25, SB26, SB27 e SB28.

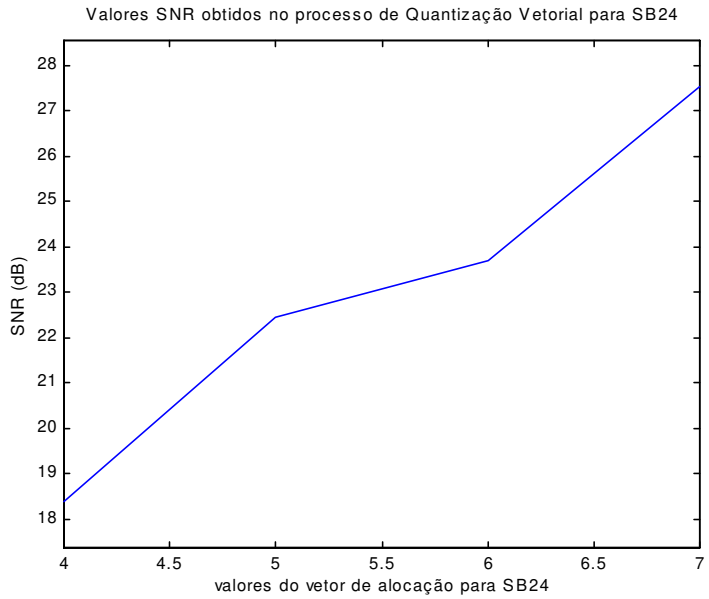


Fig. 4.24. SNR em função dos valores de alocação para a subbanda SB24.

Dessa forma, assumindo-se que todas as subbandas correspondentes são quantizadas vetorialmente e com a máxima resolução possível, o número de multiplicações reais envolvidas em todo o processo é de 7168, o qual constitui também a quantidade de somas reais. Essa tarefa computacional é menor do que uma FFT de 512 amostras. Portanto, a implementação desse formato de quantização é possível para aplicações em tempo real.

Assim, assumindo-se a utilização de um DSP (*Digital Signal Processor*) com precisão de 24 bits, o espaço de memória utilizado pelas árvores será de aproximadamente 5 kbytes.

4.8 QUADRO (*FRAME*) DE SINCRONIZAÇÃO

Cada bloco de áudio codificado é transmitido em pacotes de bits denominados *frames* (quadros) de sincronização. O formato do *frame* de sincronização que foi projetado para o codificador proposto é mostrado na Fig. 4.25. Note-se que toda a informação necessária para uma adequada decodificação é colocada no começo do *frame* para assim configurar os algoritmos correspondentes . O decodificador poderá portanto recuperar a partir das subbandas quantizadas e codificadas a informação de áudio que será posteriormente reproduzida no receptor.

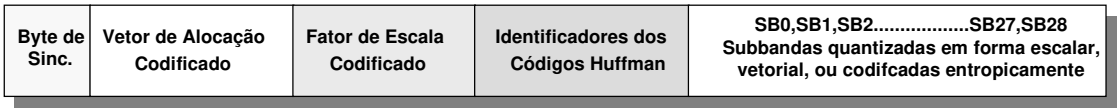


Fig. 4.25. Frame de sincronização utilizado no codificador proposto.

4.9 DECODIFICADOR

O processo de decodificação é mais simples do que aquele utilizado no codificador. Isso pode ser observado na Fig. 4.26, onde as etapas principais são formadas pelo decodificador de entropia (formado por uma tabela de todas as famílias de códigos de Huffman utilizadas no codificador), pelo dequantizador de subbandas a nível escalar ou vetorial, e pelo processo de reconstrução *wavelet packets* formado por uma árvore, cujos pares de filtragem de recuperação apresentam o formato mostrado na Fig. 4.6.

Na etapa final do decodificador tem-se o processo de dejanelamento encarregado da recuperação do sinal de áudio a partir da seqüência de blocos superpostos.

Para o acesso aleatório a uma seqüência de *frames* é necessário a atualização dos valores que são codificados em forma diferencial. Por esse motivo, o codificador transmite os valores reais dos fatores de escala e do vetor de alocação a cada 20 *frames* de áudio o que eqüivale a um tempo máximo de aceso de 0.4 segundos.

4.10 COMENTÁRIOS FINAIS

Neste capítulo foram descritas as diferentes etapas que formam o codificador de áudio proposto através deste trabalho de tese. É importante mencionar que todos os algoritmos foram desenvolvidos em MATLAB 5.0 fazendo uso de suas ferramentas de processamento de sinais e de processamento com wavelets.

No Apêndice B apresenta-se o programa principal formado pelas distintas etapas de codificação, instruções e procedimentos desenvolvidos para esse trabalho de tese.

Vários aspectos muito específicos, referentes aos detalhes de programação não foram abordados porque o objetivo principal esteve centrado em apresentar o detalhamento das técnicas

utilizadas para se atingir os níveis de compressão próprios de um codificador de intermediária complexidade.

Os resultados, as conclusões e os comentários finais serão apresentados no próximo capítulo onde será descrito também o método de avaliação utilizado no codificador proposto.

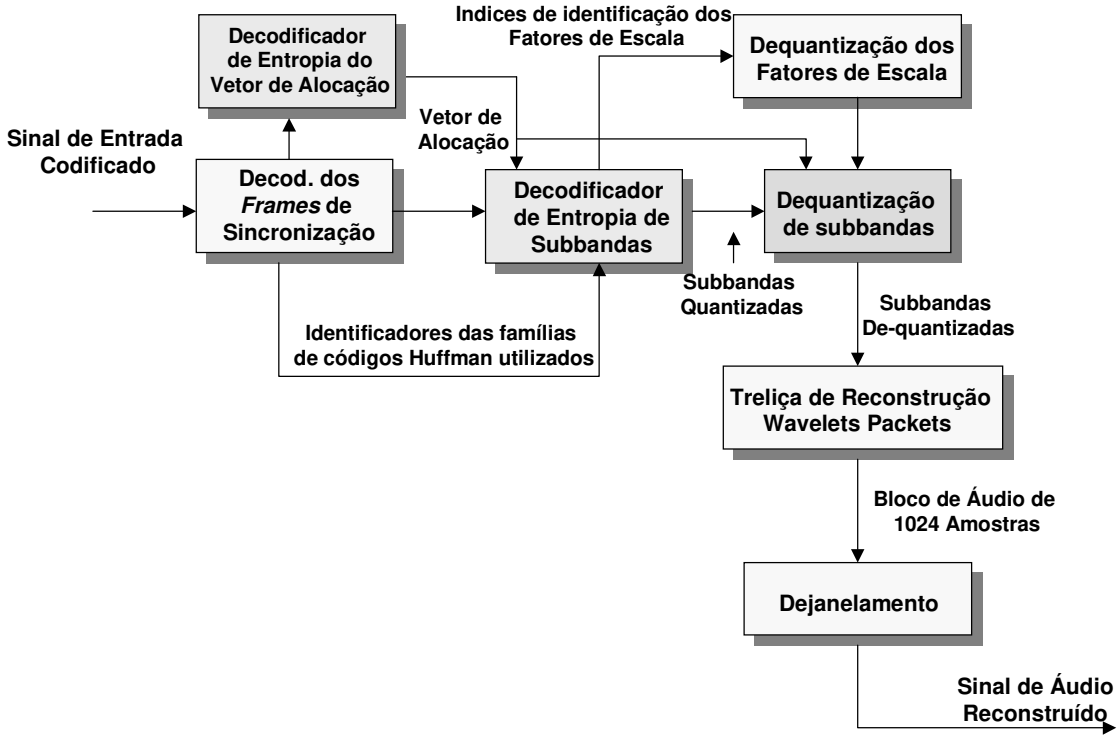


Fig. 4.26. Diagrama de blocos do decodificador de áudio proposto.

CAPÍTULO 5

TESTES, RESULTADOS E CONCLUSÕES FINAIS

5.1 MÉTODO DE AVALIAÇÃO

O método de avaliação aplicado ao codificador foi do tipo subjetivo e é baseado parcialmente na recomendação **ITU-R BS. 1116.** O princípio básico desse particular teste pode ser brevemente descrito da seguinte forma: o ouvinte escuta três fontes de áudio : **A**, **B** e **C**. O sinal de referência original encontra-se sempre disponível na fonte **A**. Nas fontes **B** e **C** encontram-se disponíveis, aleatoriamente, os sinais tanto de referência quanto de teste.

O ouvinte deve avaliar de acordo com uma escala contínua de graus de distorção, a distorção de **B** comparado com **A**, e a de **C** também comparado com **A**. Uma das fontes **B** ou **C**, deve ser indiscernível da fonte **A**; enquanto que a outra deve revelar algum tipo de distorção. Qualquer diferença percebida entre a referência e as outras fontes deve ser interpretada como uma distorção. Normalmente, um único atributo é utilizado na avaliação : **Qualidade Básica do Áudio**. Isso é definido como um atributo global que inclui quaisquer e todas as diferenças detectadas entre a referência e a outra fonte.

A escala de distorção é contínua e sujeita a cinco níveis de referência especificados na Tabela 5.1 (ITU-R BS.562).

TABELA 5.1 ESCALA DE DISTORÇÃO PARA A AVALIAÇÃO SUBJETIVA DE ÁUDIO (ITU-R BS.562)

Distorção	Grau
Imperceptível	**5.0**
Perceptível, com distorção aceitável	**4.0**
Perceptível, com distorção regular	**3.0**
Perceptível, com alta distorção	**2.0**
Perceptível, com altíssima distorção (qualidade muito pobre de áudio)	**1.0**

A análise do resultado da avaliação subjetiva está geralmente baseada no grau de diferença subjetiva (**SDG**) definido como:

$$\mathrm{SDG} = \mathrm{Grau}_{\text{sinal sob teste}} - \mathrm{Grau}_{\text{sinal de referência}} \tag{5.1}$$

Os valores SDG encontram-se na faixa de 0 a -4, onde 0 corresponde a uma distorção imperceptível enquanto que -4 a uma distorção julgada como muito alta em comparação com a qualidade do sinal original.

Os valores finais SDG obtidos para cada tipo de som ou música, são calculados tomando-se a média dos valores atribuídos por cada ouvinte (a cada tipo de som) que participa no processo de avaliação.

5.2 SINAIS DE TESTE

Para a avaliação do codificador proposto, os sinais originais foram extraídos em formato monocanal de um CD (*Compact Disk*) convencional de áudio e armazenados em seguida no computador através de arquivos do formato multimidia WAV.

Os parâmetros técnicos que definem portanto esse processo de digitalização (**PCM : Pulse Code Modulation**) podem ser especificados da seguinte forma:

- Freqüência de amostragem : **44100 Hz.**
- Bits por amostra : **16**

- Número de Canais de áudio : **1**
- Taxa de Bits : **705600 bit/ Seg**

A duração de cada fragmento de música armazenado foi de **8** a **12 segundos**.

Na execução do processo de compressão um determinado arquivo de música é lido seqüencialmente pelo codificador a fim de extrair e formar os blocos de áudio que serão codificados e armazenados em formato de frame de sincronização. O codificador deverá receber também como parâmetros de funcionamento: a **taxa de bits requerida**, o **tipo de wavelet** as ser utilizado e o número de amostras que formam a **superposição** de blocos de áudio adjacentes.

Após o processo de codificação, prossegue-se com a decodificação do arquivo comprimido e com a geração do arquivo WAV que contém o sinal reconstruído.

Posteriormente, usando-se as ferramentas de multimídia, serão reproduzidos os arquivos WAV tanto do sinal original quanto do sinal decodificado a fim de se realizar os testes e as comparações correspondentes.

Os tipos de sinais que foram armazenados e testados correspondem a vários tipos de música com diferentes tipos de instrumentos, vozes e sons em geral. As características desses sinais serão mostradas mais adiante nas tabelas de resultados.

5.3 EQUIPAMENTO E SOFTWARE UTILIZADO

Nos processos de digitalização, implementação de algoritmos , armazenamento, testes e reprodução dos sinais de áudio foram utilizadas as seguintes ferramentas de hardware e software:

1. HARDWARE :

- Computador PENTIUM II – 300 MHz, 192 Mbytes de RAM e HD's de 4.2 e 17 Gbytes.
- Unidade CD ROM Creative 24x
- Unidade CD Writer HP Plus 8200
- ZIP Driver IOMEGA 100 Mbytes
- Placa Multimídia AWE64 ISA : digitalização de 8 e 16 bits em modos estéreos e mono, taxas de amostragem programadas de 5 KHz a 44.1KHz, filtragem dinâmica para gravação e reprodução de áudio digital, volume principal a 28 níveis em 2dB/etapa, controle de agudos/graves com 15 níveis de –14 dB a 14 dB em 2dB/etapa, amplificador de força

estéreo com saída estéreo de 6 watts e 4Ω por canal.

- Sistema Integrado de Som SONY FH-G88AV : amplificador estéreo/surround com potência de saída RMS de 80 watts e 6Ω por canal, entrada de sinal com tomada RCA sensibilidade de 450mV e 47 kΩ de impedância. Duas caixas acústicas com sistema Bass Reflex de 3 vias magneticamente blindadas (**woofer**, **tweeter** e **super tweeter**) e com impedância nominal de 6Ω.

2. SOFTWARE :

- Windows 95 : Sistema Operacional
- MATLAB 5.0 : Plataforma para a implementação dos algoritmos de codificação.
- Aplicativos Multimídia Creative : Creative CD, Creative WAV, Creative Mixer, etc.
- Winamap 2.65 : Reprodutor de arquivos WAV, MP1, MP2, MP3, etc.
- MusicMatch Jukebox 5.10 : Reprodutor e conversor de arquivos WAV-MP3 e MP3-WAV.

5.4 IMPLEMENTAÇÃO DOS TESTES SUBJETIVOS

A implementação dos testes subjetivos foi realizada da seguinte forma:

O computador primeiramente foi conectado com o aparelho de som SONY ligando a saída *speaker* da placa multimídia com a entrada de sinal estéreo RCA que é disponível nesse último. Assim, qualquer som reproduzido pelo computador será também reproduzido através das caixas acústicas do aparelho de som.

Utilizando as ferramentas de software **Creative WAV** e **Winamp** foram listadas as seqüências de músicas e sons que serão testados pelo ouvintes.

Baseados no formato de avaliação descrito na seção 5.1 foram implementados 4 tipos de teste que são detalhados a seguir:

- **Teste 1** : Nesse teste o sinal reconstruído é comparado diretamente com o sinal original, as quais se encontravam disponíveis aleatoriamente nas fontes **B** e **C**. Para esse caso, os valores SDG são calculados de acordo com (5.1), sendo que o desempenho do codificador será melhor à medida que os mesmos sejam menos negativos e mais próximos de zero.

Observação : Se os valores SDG resultam positivos no Teste 1 para algum tipo de música, então os mesmos serão levados a valores 0 e isso indicará que o ouvinte não percebeu diferença alguma entre ambos os sinais.

- **Teste 2** : Nesse teste, o sinal reconstruído é comparado com um sinal decodificado (com o mesmo fator de compressão) a partir de outro formato de codificação, sendo que ambos estarão disponíveis aleatoriamente nas fontes **B** e **C**. Para esse caso, a comparação foi realizada com os codificadores MPEG-1 *Layer* 2 (MP2) e MPEG-1 *Layer* 3 (MP3) para diferentes taxas de bits. Os valores SDG para esse tipo de teste são calculados da seguinte forma:

$$\text{SDG} = \text{Grau}_{\text{sinal sob teste}} - \text{Grau}_{\text{sinal de outro formato de codifcação}} \tag{5.2}$$

Assim, o desempenho do codificador será melhor, à medida que os valores SDG sejam mais positivos.

- **Teste 3 :** Nesse teste são comparados dois sinais reconstruídos a partir do decodificador proposto, os quais foram decodificados a partir de sinais com taxas de bits distintas. Dessa forma, são realizadas duas avaliações: 1-Uma comparando sinais para 128 kbit/s e 96kbit/s e; 2- A outra comparando sinais para 96 kbit/s e 80 kbit/s. Em cada caso os sinais estarão disponíveis aleatoriamente nas fontes **B** e **C**. Os valores SDG para esse tipo de teste são calculados da seguinte forma:

$$\text{SDG} = \text{Grau}_{\text{maior taxa de tansmissão}} - \text{Grau}_{\text{menor taxa de tansmissão}} \tag{5.3}$$

- **Teste 4 :** Nesse teste são comparados dois sinais reconstruídos a partir do decodificador proposto, os quais apresentam o mesmo fator de compressão mas utilizam diferentes tipos de *wavelet* nos processos de decomposição e reconstrução . Esse sinais encontram-se disponíveis aleatoriamente nas fontes **B** e **C**. Os valores SDG para esse caso são calculados da seguinte forma:

$$\text{SDG} = \text{Grau}_{\text{wavelet mais seletiva}} - \text{Grau}_{\text{wavelet menos seletiva}} \tag{5.4}$$

Em cada tipo de teste o ouvinte escutará a seguinte seqüência auditiva:

1. Sinal auditiva de voz **: "ORIGINAL"** (5 segundos)
2. Fragmento de áudio : música ou som original, **Fonte A**, 8 a 12 segundos
3. Sinal auditivo de voz **: "MÚSICA UM"** (5 segundos)
4. Fragmento de áudio : **Fonte B**, 8 a 12 segundos.
5. Sinal auditivo de voz **: "MÚSICA DOIS"** (5 segundos)
6. Fragmento de áudio : **Fonte C**, 8 a 12 segundos.
7. Sinal auditivo de voz **: "AVALIAR"** (10 segundos)

Cada ouvinte recebe algumas fichas onde se encontra especificada graficamente para cada tipo de música, a escala de distorção apresentada na Tabela 5.1 (Fig. 5.1). Assim, cada pessoa julgará a qualidade básica de áudio da fonte **B** e da fonte **C** em relação à fonte **A** (sinal original). O ouvinte indicará com uma marca na linha de escala mostrada na Fig. 5.1, o nível de distorção que é percebido nas comparações correspondentes.

Por outro lado, para a realização dos testes foi reunido um grupo de 15 a 20 pessoas em sua maioria estudantes de ciências e engenharia elétrica e de computação com idade entre 20 a 35 anos.

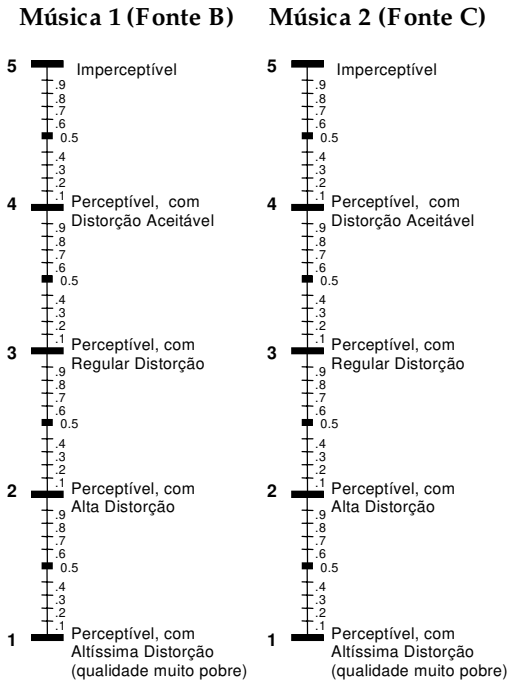


Fig 5.1. Escala de distorção em formato de linha utilizado nas fichas de avaliação entregues a cada ouvinte.

5.5 RESULTADOS

TABELA 5.2 RESULTADOS DO TESTE 1

TESTE 1 : VALORES SDG RESULTANTES DA COMPARAÇÃO COM O SINAL ORIGINAL				
N°	**TIPO DE SOM**	**WAVELET : DB40 TAXA : 80 KBIT/ S, COMPRESSÃO : 1:8.6**	**WAVELET : DB25 TAXA : 96 KBIT/ S, COMPRESSÃO : 1:7.2**	**WAVELET : DB15 TAXA : 128 KBIT/ S, COMPRESSÃO : 1:5.4**
1	Flauta Transversal	-0.45	-0.33	-0.29
2	Música Clássica (Orquestras Sinfônicas)	-0.42	-0.29	-0.01
3	Música Disco	-0.58	-0.13	-0.06
4	Voz Feminina	-0.16	-0.18	-0.05
5	Violão	-0.74	-0.29	-0.03
6	Piano	-0.39	-0.09	-0.13
7	Voz Masculina + Orquestra	-0.48	-0.16	-0.09
8	Voz Feminina + Orquestra	-0.05	-0.16	-0.10
9	Clarineta	-0.75	-0.61	-0.30
10	Piano + Orquestra	-0.56	-0.24	-0.29
11	Música Rock	-0.49	-0.13	-0.03
12	Tenor + Orquestra	-0.28	-0.30	-0.31
13	Soprano + Orquestra	-0.25	-0.20	-0.17
14	Violino (Vivaldi)	-0.38	-0.06	-0.12

Comentários TESTE 1 :

- De acordo com a Tabela 5.2 observa-se um melhor desempenho do codificador à medida que se aumenta a taxa de bits do sinal codificado. Isso logicamente, é razoável já que quanto maior for a taxa de bits menor será o fator de compressão.

- Observa-se também que os sinais que são puramente de voz não são muito afetados pelo fator de compressão. Isso se deve ao fato de que o codificador proposto tende a cortar as freqüências altas à medida que se exige do mesmo um maior fator de compressão. Portanto, as freqüências baixas e médias onde a voz concentra a maior quantidade de energia não sofrem uma forte deterioração.

- Pode-se notar, por outro lado, que em muitos casos o valor SDG obtido para uma taxa de bits baixa é maior do que aquele obtido para taxas maiores. Isso acontece em sons de tipo "voz de

alta intensidade mais orquestra", onde a voz que é pouco deteriorada pelo processo de compressão tende a enmascarar a distorção introduzida em sons de outros instrumentos.

TABELA 5.3 RESULTADOS DO TESTE 2

TESTE 2 : VALORES SDG RESULTANTES DA COMPARAÇÃO COM OUTROS CODIFICADORES				
N°	**TIPO DE SOM**	**CODIFICADOR PROPOSTO (DB40) X MP2 TAXA : 80 KBIT/ S COMPRESSÃO : 1:8.6**	**CODIFICADOR PROPOSTO (DB25) X MP2 TAXA : 96 KBIT/ S COMPRESSÃO : 1:7.2**	**CODIFCADOR PROPOSTO (DB15) X MP3 TAXA: 128 KBIT/ S COMPRESSÃO : 1:5.4**
1	Flauta Transversal	0.31	-0.24	0.18
2	Música Clássica (Orquestras Sinfônicas)	0.54	0.33	0.16
3	Música Disco	-0.31	0.19	0.08
4	Voz Feminina	0.56	0.53	-0.02
5	Violão	-0.06	-0.10	0.05
6	Piano	0.11	0.28	0.09
7	Voz Masculina + Orquestra	0.28	0.16	0.09
8	Voz Feminina + Orquestra	0.31	0.40	0.04
9	Clarineta	0.23	-0.13	-0.43
10	Piano + Orquestra	-0.05	-0.13	0.12
11	Música Rock	-0.09	0.13	0.06
12	Tenor + Orquestra	0.30	-0.18	0.06
13	Soprano + Orquestra	0.25	-0.09	0.06
14	Violino (Vivaldi)	0.24	0.31	0.07

Comentários TESTE 2 :

- Os valores SDG resultantes foram obtidos a partir da equação (5.2). Dessa forma, valores positivos indicam um melhor desempenho do codificador proposto nos diferentes tipos de sons.

- No caso de 80 kbit/s observa-se que o codificador MP2 apresenta um melhor desempenho para sons de música tipo disco e *rock*, os quais caracterizam-se por apresentar altos níveis de energia nas freqüências altas ou chamados também de sons agudos. Isso indica portanto, que para esses tipos de sons, o MP2 distribui melhor os níveis de quantização a serem utilizados na codificação da informação de áudio e portanto, consegue manter vivas as freqüências altas mesmo em baixas taxas de bits. No caso de sons que concentram a maior quantidade de

energia em freqüências baixas e médias observa-se que o codificador proposto apresenta melhor desempenho.

- No caso de 96kbit/s observa-se que em geral, o desempenho do codificador proposto é melhor do que o MP2, já que os poucos valores negativos resultantes do processo de avaliação são muito próximos de zero e portanto pode-se considerar que o desempenho de ambos os codificadores para esses tipos de sons é quase igual.

- Para a taxa de 128kbit/s o codificador proposto foi comparado com o conhecido MP3. Para esse caso, observa-se que o desempenho de ambos os codificadores em geral é praticamente o mesmo.

TABELA 5.4 RESULTADOS DO TESTE 3

TESTE 3 : VALORES SDG RESULTANTES DA COMPARAÇAO DE DIFERENTES TAXAS DE BITS			
Nº	**TIPO DE SOM**	**128 KBIT/ S (DB15) x 96 KBIT/ S (DB25)**	**96 KBIT/ S (DB25) x 80 KBIT/ S (DB40)**
1	Flauta Transversal	0.00	0.30
2	Música Clássica (Orquestras Sinfônicas)	0.00	-0.25
3	Música Disco	0.05	0.93
4	Voz Feminina	0.08	-0.08
5	Violão	0.20	-0.35
6	Piano	-0.20	0.55
7	Voz Masculina + Orquestra	0.38	0.08
8	Voz Feminina + Orquestra	-0.15	-0.15
9	Clarineta	0.50	-0.35
10	Piano + Orquestra	-0.05	0.68
11	Música *Rock*	0.05	1.10
12	Tenor + Orquestra	0.00	0.10
13	Soprano + Orquestra	0.00	0.36
14	Violino (Vivaldi)	-0.23	0.25

Comentários TESTE 3 :

- Os valores SDG resultantes foram obtidos para esse caso a partir da equação (5.3). Portanto, valores positivos indicam um melhor desempenho do codificador proposto para uma maior taxa de bits.

- Para o caso de 128 e 96 kbit/s observa-se que praticamente os sinais resultantes apresentam o mesmo nível de qualidade. Isso se deve também ao fato de que estamos utilizando *wavelets* com maior número de momentos com menores taxas de bits a fim de compensar a distorção por superposição de subbandas adjacentes.

- No caso de 96 e 80 kbit/s o desempenho do codificador é bem diferençável para ambos os níveis de compressão. Assim, pode-se observar que existe um melhor desempenho do codificador para a taxa de 96 kbit/s, o que é esperado e logicamente razoável. Por outro lado, observe-se também que mesmo utilizando uma *wavelet* de maior número de momentos para uma menor taxa de bits, a distorção por compressão se faz consideravelmente perceptível. Os valores SDG negativos obtidos nesse caso, correspondem a sons de freqüências baixas e médias, os quais como se mencionou anteriormente não são afetados seriamente pelo processo de compressão e em muitos casos o ouvinte pode escutar igual ou melhor do que aquele de maior taxa de bits.

TABELA 5.5 RESULTADOS DO TESTE 4

TESTE 4 : VALORES SDG RESULTANTES PARA UMA MESMA TAXA DE BITS E WAVELETS DE DIFERENTES NÚMEROS DE MOMENTOS			
Nº	**TIPO DE SOM**	**80 kbit/ s (db40) x 80 kbit/ s (db10)**	**96 kbit/ s (db25) x 96 kbit/ s (db10)**
1	Flauta Transversal	0.02	0.00
2	Música Clássica (Orquestras Sinfônicas)	0.10	0.20
3	Música Disco	0.00	0.00
4	Voz Feminina	0.00	0.00
5	Violão	0.10	0.00
6	Piano	0.26	0.20
7	Voz Masculina + Orquestra	0.00	0.10
8	Voz Feminina + Orquestra	0.00	0.00
9	Clarineta	0.00	0.00
10	Piano + Orquestra	0.00	0.10
11	Música *Rock*	0.00	0.15
12	Tenor + Orquestra	0.00	0.00
13	Soprano + Orquestra	0.00	0.10
14	Violino (Vivaldi)	0.20	-0.15

Comentários TESTE 4 :

- Os resultados do Teste 4 foram obtidos a partir da equação (5.4). Dessa forma, valores positivos indicam um melhor desempenho do codificador quando o mesmo é implementado com *wavelets* com maior número de momentos que geram filtros mais seletivos.

- Os valores SDG resultantes apresentados na Tabela 5.5 são de muita importância nos critérios de avaliação do codificador. Primeiro, porque esses resultados indicam que a utilização de *wavelets* muito menos seletivas no codificador proposto não deteriora significativamente a qualidade do sinal resultante em comparação com a qualidade obtida com *wavelets* que geram filtros mais seletivos. Isso se deve principalmente, ao bom desempenho do esquema de mapeamento do modelo psico-acústico MP2 utilizado no codificador proposto. É importante também mencionar que para se chegar a esse esquema foram testados outros formatos de mapeamento cujos resultados apresentavam uma forte dependência com a seletividade das *wavelets* utilizadas no processo de decomposição e reconstrução do sinal de áudio.

- Outra conclusão importante encontra-se no fato de que os resultados da Tabela 5.5 indicam que o codificador pode tolerar a utilização de *wavelets* que gerem filtros de pouca seletividade (menor número de coeficientes). Esse fato constitui uma vantagem do ponto de vista de eficiência computacional nos processo de decomposição e reconstrução.

5.6 CONCLUSÕES FINAIS

Sobre a utilização da transformada de Wavelets no codificador proposto

- Os resultados obtidos partir da utilização da transformada de *wavelets* no codificador proposto foram muito satisfatórios para taxas de bits correspondentes aos formatos de compressão de intermediária complexidade.

- A transformada apresentou um bom desempenho no que se refere à distorção introduzida pelo efeito de processamento em blocos de amostras (ruído "click"). Esse bom desempenho se traduz no fato de que somente foi necessária a utilização de um grupo muito reduzido de amostras de superposição para se poder eliminar esse tipo de ruído (48 amostras para um bloco de 1024 amostras : 4.68% de superposição). Normalmente, em codificadores que utilizam a chamada transformada modificada do co-seno MDCT utiliza-se uma superposição de 50% ou mais para se poder evitar uma forte distorção por efeitos de processamento em blocos.

- No presente trabalho foi utilizada também uma árvore de decomposição fixa que se aproxima do esquema de bandas críticas do ouvido humano. Essa aproximação permitiu evitar a utilização de formatos de decomposição que gerem esquemas de subbandas cuja resolução espectral seja maior do que aquilo que o ouvido pode detectar ou perceber. Portanto, o esquema de decomposição utilizado permitiu aliviar a tarefa computacional do codificador já que a aplicação dos algoritmo de compressão foi realizada a nível de subbandas e não a nível de amostras individuais de áudio.

- Com respeito ao método de implementação da árvore foi apresentado o formato de convolução direta no domínio temporal e no da transformada rápida de Fourier FFT. Esse último formato constitui logicamente o método mais atrativo para efeitos de implementação em plataformas DSP. No entanto, neste trabalho não se esteve muito preocupado em procurar ou estudar um método de implementação eficiente, já que existiu uma maior preocupação no planejamento e desenvolvimento do processo de compressão em si. Portanto, posteriormente, pode-se realizar trabalhos de pesquisa orientados ao desenvolvimento de métodos de implementação de árvore *wavelet packets* que sejam computacionalmente mais eficientes para serem utilizados diretamente nas aplicações em tempo real.

- Por outro lado, a seletividade dos filtros *wavelets* utilizados foi modificada de acordo com a taxa de bits requerida no codificador. Dessa forma, foram utilizados filtros mais seletivos para taxas menores de bits ou, em todo caso, para maior fator de compressão. No entanto, de acordo com a qualidade dos sinais reconstruídos observou-se uma pequena dependência do codificador com respeito à seletividade dos filtros *wavelets* com que foram implementadas as árvores de decomposição e reconstrução.

- No codificador proposto foram utilizados filtros FIR obtidos a partir das *wavelets* de Daubechies. Existem outros tipos de *wavelets* que podem ser também utilizados, e que podem ser especificados nas configurações iniciais do codificador. É importante se mencionar, no entanto, que em codificação de sinais de áudio, os filtros não necessitam ser de fase linear e, portanto, o tipo de filtros a ser utilizado pode ser de resposta impulsiva simétrica (*wavelets* biortogonais) ou não simétrica.

- Além dos testes apresentados foram realizadas outras avaliações isoladas com *wavelets* de tipo biortogonal, cujos resultados não representaram mudança significativa na qualidade do sinal reconstruído.

Sobre o Modelo Psico-acústico

- A utilização parcial do modelo psico-acústico MP2 e o formato de mapeamento utilizado no codificador proposto permitiram a obtenção de resultados muito satisfatórios para taxas de bits em torno de 128, 96 e 80 kbit/s.

- O esquema de mapeamento baseado na decomposição da energia do sinal de áudio e da função de autocorrelação do ruído apresentou um melhor desempenho com respeito aos outros esquemas de mapeamento que foram também testados. Neste trabalho foram utilizadas as técnicas de codificação de entropia, quantização escalar e quantização vetorial para se codificar as amostras de áudio que conformam cada subbanda *wavelet*. Esses formatos de codificação foram aplicados em função dos resultados do modelo psico-acústico e do algoritmo de alocação de bits. Portanto, a partir do mesmo esquema de mapeamento apresentado neste trabalho podem-se utilizar e testar outras técnicas de compressão de dados que possam melhorar ainda mais o desempenho do codificador proposto. Isso é possível porque o codificador é formado por blocos de codificação relativamente independentes.

Sobre o algoritmo de alocação de bits e a codificação de entropia

- A utilização de um algoritmo de alocação primária e um algoritmo de alocação delta de bits permitiu dar uma maior eficiência computacional ao codificador. Ambos os processos são baseados na minimização do parâmetro *MNR* (*masking to noise ratio*) calculado a partir da diferença entre a relação sinal-ruído *SNR* (*signal to noise ratio*) e o parâmetro resultante do modelo psico-acústico *SMR* (*signal to masking ratio*). Esse formato de alocação é também utilizado nos codificadores da linha MPEG e foi adaptado ao esquema de subbandas do codificador proposto obtendo-se, conforme visto anteriormente, resultados muito satisfatórios.

- Na literatura de compressão de sinais podem ser encontrados também outros métodos de alocação que no futuro podem ser testados e avaliados.

- Por outro lado, a aplicação da codificação de entropia em função dos histogramas resultantes para cada subbanda, permitiu obter ganhos consideráveis de compressão, principalmente em subbandas nas quais não pode ser utilizada a quantização vetorial dado que nelas o ouvido humano detectaria fortemente a distorção ou ruído introduzido. Portanto, devido ao fato de que a codificação de entropia não introduz distorção no sinal reconstruído, as subbandas correspondentes aos chamados sons médios (1kHz – 6kHz) foram somente quantizadas escalarmente e codificados entropicamente fazendo-se uso de famílias de códigos de Huffman geradas a partir de diferentes tipos de distribuições. A validade desse critério foi comprovada através dos testes realizados e dos resultados obtidos.

- Posteriormente, poder-se-á também explorar outros formatos de codificação de entropia que talvez possam melhorar o desempenho do codificador.

Sobre a quantização vetorial

- A técnica de quantização vetorial utilizada no codificador proposto foi destinada a obter ganhos de compressão consideráveis em subbandas onde o ouvido apresenta pouca sensibilidade na detecção de distorção. Os resultados obtidos foram ótimos, mesmo para baixas taxas de bits onde o procedimento de procura nas árvores de vetores quantizadores é de menor resolução, o que teoricamente introduziria altos níveis de distorção.

- O dimensionamento dos vetores quantizadores utilizados em cada faixa e a resolução do algoritmo de procura baseado nos resultados do modelo psico-acústico, constituíram também critérios muito adequados para a obtenção de uma boa qualidade de áudio em diferentes taxas de bits.

Sobre os testes, os resultados e os trabalhos futuros

- O método de avaliação subjetiva aplicado ao codificador proposto permitiu a obtenção de resultados que, de certa forma, se aproximam muito da qualidade verdadeira do sinal reconstruído, em comparação com o sinal original ou aos outros formatos de codificação.

- Existem também na literatura diversos métodos de avaliação do tipo objetivo que são utilizados para se testar codificadores de distintos tipos de sinais. Os resultados desses métodos no entanto, muitas vezes não são correlacionados com aqueles obtidos mediante testes subjetivos, sendo que na atualidade não existe um método objetivo que possa indicar confiavelmente a verdadeira qualidade do sinal reconstruído. Isso logicamente justifica a utilização dos testes subjetivos para a avaliação do codificador proposto.

- Através de futuros projetos de pesquisa, o formato de codificação proposto neste trabalho poderá ser estendido para aplicações sobre sinais estéreo e implementado em linguagem de programação C++ para em seguida ser instalado sobre plataformas Windows 95, 98 ou 2000.

- Da mesma forma podem-se realizar novos testes com novos formatos de quantização vetorial e de codificação de entropia, a fim de melhorar o desempenho do codificador e se buscar como meta atingir taxas de 64kbits/s ainda com excelente qualidade de áudio.

APÊNDICE A

TABELA A.1 VALORES DA PARTIÇÃO ESPECTRAL UTILIZADOS NO MODELO PSICO-ACÚSTICO MP 2 [7]

Índice	Linha Espectral Inferior	Linha Espectral Superior	Bval	Minval	TMN
1	1	1	0.00	0.0	24.5
2	2	2	0.43	0.0	24.5
3	3	3	0.86	0.0	24.5
4	4	4	1.29	20.0	24.5
5	5	5	1.72	20.0	24.5
6	6	6	2.15	20.0	24.5
7	7	7	2.58	20.0	24.5
8	8	8	3.01	20.0	24.5
9	9	9	3.45	20.0	24.5
10	10	10	3.88	20.0	24.5
11	11	11	4.28	20.0	24.5
12	12	12	4.67	20.0	24.5
13	13	13	5.06	20.0	24.5
14	14	14	5.42	20.0	24.5
15	15	15	5.77	20.0	24.5
16	16	16	6.11	17.0	24.5
17	17	19	6.73	17.0	24.5
18	20	22	7.61	15.0	24.5
19	23	25	8.44	10.0	24.5
20	26	28	9.21	7.0	24.5
21	29	31	9.88	7.0	24.5
22	32	34	10.51	4.4	25.0
23	35	37	11.11	4.5	25.6
24	38	40	11.65	4.5	26.2
25	41	44	12.24	4.5	26.7
26	45	48	12.85	4.5	27.4
27	49	52	13.41	4.5	27.9
28	53	56	13.94	4.5	28.4
29	57	60	14.42	4.5	28.9
30	61	64	14.86	4.5	29.4
31	65	69	15.32	4.5	29.8
32	70	74	15.79	4.5	30.3
33	75	80	16.26	4.5	30.8
34	81	86	16.73	4.5	31.2
35	87	93	17.19	4.5	31.7
36	94	100	17.62	4.5	32.1
37	101	108	18.05	4.5	32.5

38	**109**	**116**	**18.45**	**4.5**	**32.9**
39	**117**	**124**	**18.83**	**4.5**	**33.3**
40	**125**	**134**	**19.21**	**4.5**	**33.7**
41	**135**	**144**	**19.60**	**4.5**	**34.1**
42	**145**	**155**	**20.00**	**4.5**	**34.5**
43	**156**	**166**	**20.38**	**4.5**	**34.9**
44	**167**	**177**	**20.74**	**4.5**	**35.2**
45	**178**	**192**	**21.12**	**4.5**	**35.6**
46	**193**	**207**	**21.48**	**4.5**	**36.0**
47	**208**	**222**	**21.84**	**4.5**	**36.3**
48	**223**	**243**	**22.20**	**4.5**	**36.7**
49	**244**	**264**	**22.56**	**4.5**	**37.1**
50	**265**	**286**	**22.91**	**4.5**	**37.4**
51	**287**	**314**	**23.26**	**4.5**	**37.8**
52	**315**	**342**	**23.60**	**4.5**	**38.1**
53	**343**	**371**	**23.95**	**4.5**	**38.4**
54	**372**	**401**	**24.30**	**4.5**	**38.8**
55	**402**	**431**	**24.65**	**4.5**	**39.1**
56	**432**	**469**	**25.00**	**4.5**	**39.5**
57	**470**	**513**	**25.33**	**3.5**	**39.8**

TABELA A.2 LIMIAR ABSOLUTO DE PERCEPÇÃO DO OUVIDO HUMANO VÁLIDO PARA UM FREQÜÊNCIA DE AMOSTRAGEM DE 44.1KHZ [7]

Linha Espectral Inferior	Linha Espectral Superior	Limiar Absoluto de Percepção do Ouvido Humano absthr (dB)
1	**1**	**45.05**
2	**2**	**25.87**
3	**3**	**18.70**
4	**4**	**14.85**
5	**5**	**12.41**
6	**6**	**10.72**
7	**7**	**9.47**
8	**8**	**8.50**
9	**9**	**7.73**
10	**10**	**7.10**
11	**11**	**6.56**
12	**12**	**6.11**
13	**13**	**5.72**
14	**14**	**5.37**
15	**15**	**5.07**
16	**16**	**4.79**
17	**17**	**4.55**
18	**18**	**4.32**

19	**19**	**4.11**
20	**20**	**3.92**
21	**21**	**3.74**
22	**22**	**3.57**
23	**23**	**3.40**
24	**24**	**3.25**
25	**25**	**3.10**
26	**26**	**2.95**
27	**27**	**2.81**
28	**28**	**2.67**
29	**29**	**2.53**
30	**30**	**2.39**
31	**31**	**2.25**
32	**32**	**2.11**
33	**33**	**1.97**
34	**34**	**1.83**
35	**35**	**1.68**
36	**36**	**1.53**
37	**37**	**1.38**
38	**38**	**1.23**
39	**39**	**1.07**
40	**40**	**0.90**
41	**41**	**0.74**
42	**42**	**0.56**
43	**43**	**0.39**
44	**44**	**0.21**
45	**45**	**0.02**
46	**46**	**-0.17**
47	**47**	**-0.36**
48	**48**	**-0.56**
49	**50**	**-0.96**
51	**52**	**-1.37**
53	**54**	**-1.79**
55	**56**	**-2.21**
57	**58**	**-2.63**
59	**60**	**-3.03**
61	**62**	**-3.41**
63	**64**	**-3.77**
65	**66**	**-4.09**
67	**68**	**-4.37**
69	**70**	**-4.60**
71	**72**	**-4.78**
73	**74**	**-4.91**
75	**76**	**-4.97**
77	**78**	**-4.98**
79	**80**	**-4.92**
81	**82**	**-4.81**
83	**84**	**-4.65**
85	**86**	**-4.43**
87	**88**	**-4.17**
89	**90**	**-3.87**
91	**92**	**-3.54**
93	**94**	**-3.19**

95	**96**	**-2.82**
97	**100**	**-2.06**
101	**104**	**-1.33**
105	**108**	**-0.64**
109	**112**	**-0.04**
113	**116**	**0.47**
117	**120**	**0.89**
121	**124**	**1.23**
125	**128**	**1.51**
129	**132**	**1.74**
133	**136**	**1.93**
137	**140**	**2.11**
141	**144**	**2.28**
145	**148**	**2.45**
149	**152**	**2.63**
153	**156**	**2.82**
157	**160**	**3.03**
161	**164**	**3.25**
165	**168**	**3.49**
169	**172**	**3.74**
173	**176**	**4.02**
177	**180**	**4.32**
181	**184**	**4.64**
185	**188**	**4.98**
189	**192**	**5.35**
193	**200**	**6.15**
201	**208**	**7.07**
209	**216**	**8.10**
217	**224**	**9.25**
225	**232**	**10.54**
233	**240**	**11.97**
241	**248**	**13.56**
249	**256**	**15.30**
257	**264**	**17.23**
265	**272**	**19.33**
273	**280**	**21.64**
281	**288**	**24.15**
289	**296**	**26.88**
297	**304**	**29.84**
305	**312**	**33.04**
313	**320**	**36.51**
321	**328**	**40.24**
329	**336**	**44.26**
337	**344**	**48.58**
345	**352**	**53.21**
353	**360**	**58.17**
361	**368**	**63.48**
369	**376**	**69.13**
377	**384**	**69.13**
385	**392**	**69.13**
393	**400**	**69.13**
401	**408**	**69.13**
409	**416**	**69.13**

417	**424**	**69.13**
425	**432**	**69.13**
433	**440**	**69.13**
441	**448**	**69.13**
449	**456**	**69.13**
457	**464**	**69.13**

APÊNDICE B

```
%     CODIFICADOR/DECODIFICADOR DE SINAIS DE ÁUDIO DE ALTA QUALIDADE
%       BASEADO NA CODIFICAÇÃO DE SUBBANDAS  WAVELETS (MATLAB 5.0)
%     TAXAS : 128, 112 E 80 Kbit/s, para sinais de áudio monocanais.
%                 TESE DE DOUTORADO (Programa Principal)
%       "UM MÉTODO DE COMPRESSÃO DE ÁUDIO BASEADO NA CODIFICAÇÃO DE
%                            SUBBANDAS WAVELETS"
%                         Projeto FAPESP : 97/05390-7
%         Responsável: Eng. M.Sc. Guillermo Leopoldo Kemper Vásquez
%                      Orientador : Prof. Dr. Yuzo Iano
%                   DECOM/FEEC/UNICAMP - Fevereiro, 2001

   clear;
%  Especificações Gerais: Tipo de wavelet, taxa requerida, freqüência
%  de amostragem (fm), tamanho do bloco de amostras a ser
%  codificado(tamframe), superposição (overlapp=48 amostras), modelo
%  psico-acústico parcial utilizado (modelo).

   Wavelet='db20';
   taxa=80*1024;
   fm=44100;
   tamframe=1024;
   overlapp=48;
   modelo='mpeg';

%  Especificação dos diretorios dos arquivos fonte e destino.
   dirorigem='c:\wav\';    %diretorio de origem
   dirdestino='c:\wav80\'; %diretorio de destino
   armusica='laura';  %arquivo origem
   ardestino='laura'; %arquivo destino
   [wv,fuente,destino,nompsico,modpsico,mattonal]=paraini3(wavelet,tax
   a,modelo,armusica,ardestino,dirorigem,dirdestino);

%  comentário :  A função paraini3 gera o nome completo do arquivo
%  destino em função dos parametros especificados.

   NN=tamframe;
   tj=overlapp;

%  Cálculo dos filtros Wavelets.
   [lod,lhd,lor,lhr]=wfilters(wv);
   [lod,lhd,lor,lhr,fld,fhd,w1,mlop]=calfiltroswav(lod,lhd,lor,lhr);

%  comentário :  A função wfilters é uma ferramenta do matlab que
%  determina os filtros de decomposição e reconstrução a partir de uma
%  wavelet dada. Por outro lado, a função calfiltroswav é uma
%  ferramenta desenvolvida para modificar os filtros de acordo com o
%  funcionamento do codificador proposto. Essa função também determina
%  os filtros de decomposição utilizados no modelo psico-acústico.
```

```
%  Cálculo da Janela de superposição a ser utilizada.
   tipjan=1;
   jhx=[];
   jhpx=[];
   [jhx,jhpx]=gerjant(tj,tipjan);
   M=length(jhx);
   M1=M/2;
   jh01=jhx(1:M1);
   jh02=jhx(M1+1:M);
   jh11=jhpx(1:M1/2);
   jh12=jhpx((M1/2)+1:M1);
   bi=[];
   bre=[];

%  Comentário : A função gerjant determina os valores da janela de
%  superposição a ser utilizada no codificador (janelamento) e no
%  decodificador (dejanelamento).

%  Função de geração dos parâmetros de controle para o arquivo fonte.
   [fid,fad,FF1,N,NS,RS,TAM]=pararch(fuente,NN,tj);

%  Geração dos parâmetros de controle das treliças de decomposição e
%  reconstrução.
   nban=29;  %número de subbandas.
   [Nt,slop]=calcnt1(NN,nban,mlop); %L(p)=Nt(p)

%  Cálculo dos parâmetros de funcionamento dos algoritmos de  alocação
%  de bits.
   Frameseg1=fm/(NN-tj);
   bitframe1=taxa/frameseg1; % bits por bloco de áudio.
   naloca1=((nban+18)*4);
   namost1=bitframe1-naloca1-7;
   resti=0;
   alocan=[zeros(1,nban)];


%  Definição dos Parâmetros de Codificação do Fator de Escala
   fol=fopen('c:\Guille\datos\fescal.dat','r');
   sc3=[];
   sc3=rot90(fread(fol,'float'));
   fclose(fol);
   nbs1=15;
   nbs=6;
   sctipo='ubit6';
   limmax1=(2^nbs1)-1;

%  Cálculo e Definição dos Parametros do Modelo Psico-acústico
   p=1;
   for i=1:NN
      hps1(i)=sqrt(0.5-(0.5*cos(((2*pi)*(i-0.5))/NN)));
      if rem(i,2)==0
         hps2(p)=sqrt(0.5-(0.5*cos(((2*pi)*(p-0.5))/(NN/2))));
         p=p+1;
      end;
   end;
```

```
npar=57;
[limpar,wlow,whig,minval,TMN,sprmat,rnorm,cfai,cfas,lowpar,higpar,b
val]=mpamodel(npar);
comp=[];

rw1=[zeros(1,513)];
rw2=[zeros(1,513)];
fw1=[zeros(1,513)];
fw2=[zeros(1,513)];

% Definição das matrizes utilizadas na codificação de entropia
faw=fopen('c:\Guille\datos\MHUF3.dat','r');
MH3=fread(faw,'uint');
n1=length(MH3)/((2^3)-1);
fclose(faw);
MH3=reshape(MH3,n1,7);

faw=fopen('c:\Guille\datos\MHUF4.dat','r');
MH4=fread(faw,'uint');
n1=length(MH4)/((2^4)-1);
fclose(faw);
MH4=reshape(MH4,n1,15);

faw=fopen('c:\Guille\datos\MHUF5.dat','r');
MH5=fread(faw,'uint');
n1=length(MH5)/((2^5)-1);
fclose(faw);
MH5=reshape(MH5,n1,31);

faw=fopen('c:\Guille\datos\MHUF6.dat','r');
MH6=fread(faw,'uint');
n1=length(MH6)/((2^6)-1);
fclose(faw);
MH6=reshape(MH6,n1,63);

faw=fopen('c:\Guille\datos\MHUF7.dat','r');
MH7=fread(faw,'uint');
n1=length(MH7)/((2^7)-1);
fclose(faw);
MH7=reshape(MH7,n1,127);

% Disponibilização dos codebooks utilizados no processo de
% quantização vetorial.
SA1=[];
SA2=[];
SA3=[];
SA4=[];
SA5=[];
SA6=[];
SA7=[];
GA1=[];
GA2=[];
GA3=[];
GA4=[];
GA5=[];
GA6=[];
GA7=[];
```

```
SB1=[];
SB2=[];
SB3=[];
SB4=[];
SB5=[];
SB6=[];
SB7=[];
GB1=[];
GB2=[];
GB3=[];
GB4=[];
GB5=[];
GB6=[];
GB7=[];


% Codebook de vetores quantizadores para as Subbandas SB22...SB24
fcsh=fopen('c:\Guille\datos\F24-2ex.dat','r');
[SA4,GA4]=forcod4(fcsh,2,32,512,4);
[SA5,GA5]=forcod4(fcsh,2,64,512,4);
[SA6,GA6]=forcod4(fcsh,2,128,512,4);
[SA7,GA7]=forcod4(fcsh,2,256,512,4);
fclose(fcsh);

% Codebook de vetores quantizadores para as Subbandas SB25...SB28
fcsh=fopen('c:\Guille\datos\F28-1.dat','r');
[SB1,GB1]=forcod11(fcsh,4,2);
fclose(fcsh);
fcsh=fopen('c:\Guille\datos\F28-1.dat','r');
[SB2,GB2]=forcod11(fcsh,4,3);
fclose(fcsh);
fcsh=fopen('c:\Guille\datos\F28-4.dat','r');
[SB3,GB3]=forcod4(fcsh,4,4,64,16);
[SB4,GB4]=forcod4(fcsh,4,8,64,16);
[SB5,GB5]=forcod4(fcsh,4,16,64,16);
fclose(fcsh);
banele=23;

dt=0;
aculec=0;

disp('codificando......');

% ALGORITMO DE CODIFICAÇÃO

i=0;
final=0;
while final==0,

% Processo de Leitura do Bloco de Amostras Originais
   FX1=[];
   FB=[];
   i=i+1;
   if rem(i-1,20)==0
      km=1;
   else
```

```
      km=10;
   end;
   if i==1
      FX1=rot90(fread(fid,NN,'int16'));
      FB=FX1;
      FB1=[];
      FB1=FB(NN-tj+1:NN);
      Aculec=aculec+NN;
   else
      if aculec+N<=TAM
         RF=N;
         aculec=aculec+RF;
      else
         RF=TAM-aculec;
         aculec=aculec+RF;
         final=1;
      end;
      FX1=rot90(fread(fid,RF,'int16'));
      FB=[FB1 FX1 zeros(1,NN-tj-RF)];
      FB1=[];
      FB1=FB(NN-tj+1:NN);
   end;
```

% **Função de janelamento e superposição de blocos.**

```
[X]=jantrantes(FB,tj,jh01,jh02,i,NN,final);
```

% **Processo de Decomposição em 29 Subbandas *Wavelets*.**

```
[F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18,F19
,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29]=daudiowav(X,w1,lod,lhd);
```

% **Algoritmo de Cálculo e Quantização do Fator de Escala.**

```
for kk=1:nban
    BANR=[];
[BANR]=idenban1(F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,
F16,F17,F18,F19,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29,kk);
       Scal(kk)=max(abs(BANR));
       if scal(kk)==0
          scalneto(kk)=1;
       else
          scalneto(kk)=scal(kk);
       end;
       scalmax1(kk)=scal(kk)/slop(kk);;
       if scalmax1(kk)>1
          disp('alto scal');
          pause;
       end;
       hy=1;
       while (scalmax1(kk)>sc3(hy) & hy<64),
             hy=hy+1;
       end;
       scalmax(kk)=sc3(hy)*slop(kk);
       scalmax1(kk)=hy-1;
end;
```

```
% Escalonação das amostras a partir do fator de escala decodificado.

[F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18,F19
,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29]=escalacion(F1,F2,F3,F4,F5
,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18,F19,F20,F21,F22,F2
3,F24,F25,F26,F27,F28,F29,scalneto,scalmax);

namostot=namost1-(nbs*nban)+resti;

% Modelo Psico-Acústico MPEG-Layer 2 (parcial).
[tthi,MBB,limbanwav,rw1,fw1,rw2,fw2]=mpmodelm10wav(X,hps1,rw1,fw1,r
w2,fw2,wlow,whig,sprmat,rnorm,TMN,minval,limpar,cfai,cfas,nban,lowp
ar,higpar,bval,Nt,comp);


% Função de Mapeamento do Modelo no Domínio das Subbandas Wavelets.
[SMR]=dominwav2pdb10(tthi,MBB,w1,fld,fhd,nban,NN,Nt,F1,F2,F3,F4,F5,
F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18,F19,F20,F21,F22,F23
,F24,F25,F26,F27,F28,F29,slop,limbanwav,taxa);


% Alocação Primária de Bits.
[alocapri,SNRban,acuban]=alocaprimarioteswav4(SMR,Nt,namostot,nban,
dt,banele,taxa);
idvet=[zeros(1,nban)];

% Alocação Delta de Bits.
[aloca,resti,acuban,namosti,alocan,idvet,FQ1,FQ2,FQ3,FQ4,FQ5,FQ6,FQ
7,FQ8,FQ9,FQ10,FQ11,FQ12,FQ13,FQ14,FQ15,FQ16,FQ17,FQ18,FQ19,FQ20,FQ
21,FQ22,FQ23,FQ24,FQ25,FQ26,FQ27,FQ28,FQ29]=alocahuffmanteswav4tes(
SMR,Nt,scalmax,namostot,nban,MH3,MH4,MH5,MH6,MH7,F1,F2,F3,F4,F5,F6,
F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18,F19,F20,F21,F22,F23,F2
4,F25,F26,F27,F28,F29,km,alocan,SNRban,alocapri,banele,dt,idvet,tax
a);

Fwrite(fad,aloca,'ubit4');
Fwrite(fad,idvet,'ubit4');

% Quantização Vetorial.
   for kk=banele:nban
         SH=[];
         G=[];
         IT=[];
         if idvet(kk)==1 & aloca(kk)>0
           if kk>=26 | (aloca(kk)==1)
              tolve=4;
           else
              tolve=2;
           end;
           if kk<=25

[SH,G]=idenvecA(aloca(kk),SA1,SA2,SA3,SA4,SA5,SA6,SA7,GA1,GA2,GA3,G
A4,GA5,GA6,GA7);
              if aloca(kk)>=4 & aloca(kk)<=7
                 prc=aloca(kk)+3;
              else
```

```
                prc=10;
             end;
          else

[SH,G]=idenvecB(aloca(kk),SB1,SB2,SB3,SB4,SB5,GB1,GB2,GB3,GB4,GB5);
             if aloca(kk)<=2
                prc=aloca(kk)+1;
             elseif aloca(kk)>=3 & aloca(kk)<=5
                prc=aloca(kk)+3;
             else
                prc=8;
             end;
          end;
          [preci]=precivec(prc);
          BW=[];
          IT=[];

[BW]=idenban1(F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F1
6,F17,F18,F19,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29,kk);

[V,IT,scalmax1(kk)]=matchvec42(BW,tolve,Nt(kk),SH,G,slop(kk),sc3,al
oca(kk),scalmax1(kk),scalmax(kk));  %algoritmo de procura
          fwrite(fad,IT,preci);
       end;
    end;


% Codificação de Entropia dos Fatores de Escala.
 if km>1
[namostot]=scalcod(namost1,scalmax1,scalmax2,MH3,MH4,MH5,MH6,MH7,nb
an,nbs);
      resti=(namost1-namostot)+resti;
 end;
   scalmax2=scalmax1;
resti

% Frame de sincronização: armazenamento de amostras codificadas
(quantização escalar).

     for kk=1:nban
        fwrite(fad,scalmax1(kk),sctipo);
        if aloca(kk)>0 & idvet(kk)==0
              BANR=[];
              RR=[];
              switch kk
                   case 1,
                   RR=FQ1;
                   case 2,
                   RR=FQ2;
                   case 3,
                   RR=FQ3;
                   case 4,
                   RR=FQ4;
                   case 5,
                   RR=FQ5;
                   case 6,
                   RR=FQ6;
```

```
                    case 7,
                    RR=FQ7;
                    case 8,
                    RR=FQ8;
                    case 9,
                    RR=FQ9;
                    case 10,
                    RR=FQ10;
                    case 11,
                    RR=FQ11;
                    case 12,
                    RR=FQ12;
                    case 13,
                    RR=FQ13;
                    case 14,
                    RR=FQ14;
                    case 15,
                    RR=FQ15;
                    case 16,
                    RR=FQ16;
                    case 17,
                    RR=FQ17;
                    case 18,
                    RR=FQ18;
                    case 19,
                    RR=FQ19;
                    case 20,
                    RR=FQ20;
                    case 21,
                    RR=FQ21;
                    case 22,
                    RR=FQ22;
                    case 23,
                    RR=FQ23;
                    case 24,
                    RR=FQ24;
                    case 25,
                    RR=FQ25;
                    case 26,
                    RR=FQ26;
                    case 27,
                    RR=FQ27;
                    case 28,
                    RR=FQ28;
                    case 29,
                    RR=FQ29;
                    end;
                atipo=amotipo(aloca(kk));
                fwrite(fad,RR,atipo);
              end;
            end;

end;

NS=i;
```

```
   fclose(fid);
   fclose(fad);


%  DECODIFICAÇÃO

%  Parâmetros de controle do arquivo codificado.

   fad=fopen('c:\Guille\wav\coringa.dat','r');
   fud=fopen(destino,'w');
   fwrite(fud,FF1,'uint8');

   aloca=[];
   bp=[];
   scalmax=[];
   disp('decodificando..............');


%  ALGORITMO DE DECODIFICAÇÃO
   acu=0;
   dta=0;
   dt=0;
   for i=1:NS,


     for  idt=0:dt

        aloca=rot90(fread(fad,nban,'ubit4'));
        idvet=rot90(fread(fad,nban,'ubit4'));

         IT=[];
         lk=0;

%        Leitura dos índices dos vetores quantizadores utilizados.
         for kk=banele:nban
             if idvet(kk)==1 & aloca(kk)>0
                prc=aloca(kk);
                if kk>=26 | (aloca(kk)==1)
                   tolve=4;
                   if aloca(kk)<=2
                      prc=aloca(kk)+1;
                   elseif aloca(kk)>=3 & aloca(kk)<=5
                      prc=aloca(kk)+3;
                   else
                      prc=8;
                   end;
                else
                   tolve=2;
                   if aloca(kk)>=4 & aloca(kk)<=7
                      prc=aloca(kk)+3;
                   else
                      prc=10;
                   end;
                end;
                [preci]=precivec(prc);
                ras=Nt(kk)/tolve;
```

```
                lk=1;
                IK=[];
                IK=rot90(fread(fad,ras,preci));
                IT=[IT IK];
             end;
          end;

        if lk==1
           IT=IT+1;
        end;
      p=0;

%   Algoritmo de Dequantização.

    for kk=1:nban,
        RR=[];
        BB=[];
        R1=[];
        R2=[];

%       Decodificação e dequantização do fator de escala.
        Scal=rot90(fread(fad,1,sctipo));
        scalmax=sc3(scal+1)*slop(kk);%dequantização do fator de escala

%       Processo de Dequantização Vetorial.
         if aloca(kk)>0
            if idvet(kk)==0
               atipo=amotipo(aloca(kk));
               RR=rot90(fread(fad,Nt(kk),atipo));
               [BB]=dequantiz(RR,aloca(kk),scalmax);
            else
               if kk>=26 | (aloca(kk)==1)
                  tolve=4;
               else
                  tolve=2;
               end;
               ras=Nt(kk)/tolve;
               IL=[];
               r1=p+1;
               r2=r1+ras-1;
               IL=IT(r1:r2);
               if kk<=25

   [SH,G]=idenvecA(aloca(kk),SA1,SA2,SA3,SA4,SA5,SA6,SA7,GA1,GA2,GA3,G
   A4,GA5,GA6,GA7);
               else

   [SH,G]=idenvecB(aloca(kk),SB1,SB2,SB3,SB4,SB5,GB1,GB2,GB3,GB4,GB5);
               end;
               BR=[];
               for n=1:ras
                   BR=[BR sevelin(SH,IL(n))];
               end;
               BB=scalmax*BR;
               p=r2;
            end;
        else
```

```
            BB=[zeros(1,Nt(kk))];
        end;

%       Decodificação e dequantização escalar de amostras de áudio.

         switch kk
         case 1,
            F1=BB;
         case 2,
            F2=BB;
         case 3,
            F3=BB;
         case 4,
            F4=BB;
         case 5,
            F5=BB;
         case 6,
            F6=BB;
         case 7,
            F7=BB;
         case 8,
            F8=BB;
         case 9,
            F9=BB;
         case 10,
            F10=BB;
         case 11,
            F11=BB;
         case 12,
            F12=BB;
         case 13,
            F13=BB;
         case 14,
            F14=BB;
         case 15,
            F15=BB;
         case 16,
            F16=BB;
         case 17,
            F17=BB;
         case 18,
            F18=BB;
         case 19,
            F19=BB;
         case 20,
            F20=BB;
         case 21,
            F21=BB;
         case 22,
            F22=BB;
         case 23,
            F23=BB;
         case 24,
            F24=BB;
         case 25,
            F25=BB;
         case 26,
```

```
          F26=BB;
        case 27,
          F27=BB;
        case 28,
          F28=BB;
        case 29,
          F29=BB;
        end;

      end;

       S=[];
```

% **Função da Treliça de Reconstrução.**

```
[S]=raudiowav(F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F1
6,F17,F18,F19,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29,w1,lor,lhr);
```

% **Dejanelamento e recuperação de amostras a partir de blocos superpostos.**

```
    if i==1
       FR1=[];
       FX=extrac(S,1,NN-tj);
       FR1=extrac(S,NN-tj+1,NN)*diag(jh02);
       dta=0;
    end;
    if i>1 & i<NS
       if dt==0
          FX=extrac(S,1,tj)*diag(jh01);
          FX=FX+FR1;
          FX=[FX extrac(S,tj+1,NN-tj)];
          FR1=[];
          FR1=extrac(S,NN-tj+1,NN)*diag(jh02);
       end;
       if dt==1 & idt==0
          FX=[];
          FX=extrac(S,1,tj)*diag(jh01);
          FX=FX+FR1;
          FX=[FX extrac(S,tj+1,(NN/2)-(tj/2))];
          FR1=[];
          FR1=extrac(S,(NN/2)-(tj/2)+1,NN/2)*diag(jh12);
       end;
       if dt==1 & idt==1
          FX=[];
          FX=extrac(S,1,(tj/2))*diag(jh11);
          FX=FX+FR1;
          FX=[FX extrac(S,(tj/2)+1,(NN/2)-tj)];
          FR1=[];
          FR1=extrac(S,(NN/2)-tj+1,NN/2)*diag(jh02);
       end;
       dta=dt;
    end;
    if i==NS & dta==0
       FX=[];
       FX=extrac(S,1,tj)*diag(jh01);
       FX=FX+FR1;
       FX=[FX extrac(S,tj+1,tj+RF)];
```

```
      end;
      if i==NS & dta==1
         FX=[];
         FX=extrac(S,1,tj)*diag(jh01);
         FX=FX+FR1;
         FX=[FX extrac(S,tj+1,tj+RF+(tj/2))];
      end;

      acu=acu+length(FX);
      fwrite(fud,FX,'int16');
   end;

      i;
      acu;
      TAM;
end;

fclose(fad);
fclose(fud);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Raghuveer M. Rao and Ajit S. Bopardikar, *Wavelet Transform : Introduction to Theory and Applications.* Addison Wesley Longman, Inc, 1998.

[2] Ingrid Daubechies, *Ten Lectures on Wavelets* , Philadelphia, PA: Springer-Verlag, 1992.

[3] Jonas Gomes, Luis Velho, Siome Goldenstein, *Wavelets: Teoria, Software e Aplicações*, 21° Colóquio Brasileiro de Matemática, IMPA, Julho 21-25, 1997.

[4] Michel Misti, Yves Misti, Georges Oppenheim and Jean Michel Poggi, *Wavelet Toolbox For Use with MATLAB*, The Math Works, Inc., March 1996.

[5] P. P. Vaidyanathan, "Quadrature Mirror Filter Banks, M-Band Extensions and Perfect-Reconstruction Techniques", *IEEE ASSP Magazine*, July 1987.

[6] A. V. Oppenheim, Aplications of Digital Signal Processing", Prentice Hall, 1987.

[7] ISO/IEC 11172-3:1993, "*Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1,5 Mbit/s - Part 3 : Audio*", 1993.

[8] ITU-R Task Gropup 10/4, Preliminary Draft Recomendation, "*Method For Objective Measurements Of Perceived Audio Quality*", Doc. 10-4/19-E, 19 March, 1998.

[9] Seymour Shlien, "Guide to MPEG-1 Audio Standard", *IEEE Trans. Broadcasting*, vol 40, no. 4, 1998.

[10] Allen Gersho and Robert M. Gray, "*Vector Quantization and Signal Compression*" , Kluwer Academic Publishers, 1992.

[11] Deppen Sinha and Ahmed H. Tewfik, "Low Bit Rate Transparent Audio Compression using Adapted Wavelets", *IEEE Trans. Signal Processing*, vol. 41, no. 12, 1993.

[12] Pramila Srinivassan and Leah H. Jamieson, "High-Quality Audio Compression Using na Adaptive Wavelet Packet Decomposição and Psychoacoustic Modeling", *IEEE Trans. Signal Processing*, vol 46, no. 4, 1998.

[13] Martin Vetterli, and Cormac Herley, "Wavelets and Filter Banks : Theory and Design", *IEEE Trans. Signal Processing*, vol. 40, no. 9. 1992.

[14] N.S. Jayant and Peter Noll, *Digital Coding of Waveforms, Principles and Applications to Speech and Video*, Prentice-Hall, 1984.

[15] Allen Gersho, "On the Structure of Vector Quantizers", *IEEE Trans. on Information Theory*, Vol. IT-28, N° 2, March 1982.

[16] Yoseph Linde, Andrés Buzo and Robert M. Gray, "An Algorithm for Vector Quantizer Design", *IEEE Trans. on Communications*, Vol. COM-28, N° 1, January 1980.

[17] Andres Buzo, Augustine H. Gray Jr., and Robert M. Gray and John Markel, "Speech Coding Based Upon Vector Quantization", *IEEE Trans. on Acoustic and Signal Processing*, Vol. ASSP-28, N° 5, October 1980.

[18] Guillermo L. Kemper Vásquez, *Aspectos Relevantes para a Definição de um Sistema de Televisão HDTV*, Campinas : FEEC, UNICAMP, Outubro 1996, Mestrado(Comunicações)-Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas.

[19] B. P. Lathi, *An Introduction to Random Signals and Communication Theory*, International Textbook Company, 1968.

[20] John Watkinson, Compression in Video and Audio, Focal Press, 1995.

LISTA DE PUBLICAÇÕES

Publicações realizadas durante o curso de mestrado

[1] A. M. Do Nascimento, Yuzo Iano y Guillermo Kemper "Conversión de Frecuencias de Imagenes Digitalizadas para Aplicaciones en Simulación de SDVPI", *revista "Información Tecnológica"* , Vol 8, N° 4. La Serena, Chile, 1997.

[2] G. L. Kemper Vásquez, Antônio Moraes y Yuzo Iano "Sistemas de Transporte e Multiplexage para Señales de Televisión de Alta Definición (HDTV)" *Congreso de Ingeniería Eléctrica y Electrónica Universidad de Las Villas*, Santa Clara, Cuba Junho, 1997.

Publicações realizadas durante o curso de doutorado

[3] Guillermo L. Kemper Vásquez, Antônio Moraes e Yuzo Iano "Camada de Transporte e Multiplexagem para HDTV-ATSC (U.S.A)" *Convênio CPqD-TELEBRAS-UNICAMP,* Publicação FEEC 27/98, RT 16, Campinas S.P., Brasil, Dezembro 1997.

[4] Guillermo L. Kemper Vásquez, Antônio Moraes e Yuzo Iano, "Camada de Transporte e Multiplexagem para HDTV-DVB (EUROPA)", *Convênio CPqD-TELEBRAS-UNICAMP*, Publicação FEEC 19/98, RT 16, Campinas S.P., Brasil, Dezembro 1997.

[5] Guillermo L. Kemper Vásquez, Edgard Da Silva e Yuzo Iano "Compressão de Áudio : Aspectos Relevantes dos Sistemas DOLBY AC-3 e MPEG" *Convênio Fundação CPqD-UNICAMP* Publicação FEEC 008, RT 08, Campinas S.P., Brasil, Junho 1998.

[6] Guillermo L. Kemper Vásquez, Edgard Da Silva e Yuzo Iano "Considerações Teóricas para a Avaliação Objetiva de Sinais de Áudio (Formato ITU-R Task Group 10/4)" *Convênio CPqD-TELEBRAS – UNICAMP* , Publicação FEEC 005/99, RT 06, Campinas S.P., Brasil, Março 1999.

[7] Guillermo L. Kemper Vásquez y Yuzo Iano, "Compresión de Audio utilizando Transformadas de Wavelets e Codificación de Huffman" *Primer Congreso Internacional en Administración de Telecomunicaciones "Inteligencia 99"* Instituto Tecnológico de Estudios Superiores de Monterrey (ITESM), Mexico D.F. 27 - 29, Outubro 1999.

[8] A. Cardoso, A. M. do Nascimento, Yuzo Iano y G. L. Kemper Vásquez, "Desarrollo de Programas em Lenguaje C en el Ambiente de Programación Visual Khoros" *revista "Información Tecnológica"* , Vol 11, N° 2. La Serena, Chile, 2000, ISSN 0716-8756.

[9] Guillermo L. Kemper Vásquez, Edgard da Silva e Yuzo Iano, "Compressão de Sinais de Áudio para HDTV : Sistema Dolby AC-3 (U.S.A)", *revista "Telecomunicações (INATEL)"* Vol. 03, N° 1, pp 65-73, Fevereiro 2000, ISSN 1516-2338.

[10] Edgard Da Silva, Guillermo L. Kemper Vásquez e Yuzo Iano, "Compressão de Sinais de Áudio para HDTV : Sistema MPEG-2 (Europa)", *revista "Telecomunicações (INATEL)"* Vol. 03, N° 1, pp 55-64, Fevereiro 2000, ISSN 1516-2338.

[11] Guillermo. L. Kemper Vásquez y Yuzo Iano, "Un Tutorial sobre Principios y Aplicaciones de Procesos de Compresión de Señales de Audio y Video", *VII Congresso Internacional de Ingenieria Electrónica, Eléctrica y de Sistemas: "Intercon 2000 (IEEE)"* Universidad Peruana de Ciencias Aplicadas (UPC), Lima – Peru, 15 - 18, Agosto 2000.