CODIFICAÇÃO DE IMAGEM UTILIZANDO TURBO QUANTIZAÇÃO CODIFICADA POR TRELIÇAS

Marcelo Santos de Souza

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Eduardo Antônio Barros da Silva, Ph.D.

Prof. Gelson Vieira Mendonça, Ph.D.

Prof. Abraham Alcaim, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2005

SOUZA, MARCELO

Codificação de Imagem Utilizando Turbo Quantização Codificada por Treliças [Rio de Janeiro] 2005

xiii,107 pp 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1.Codificação de imagem

2.Transformada wavelet

3.Quantização por treliça

4.Códigos turbo

5.Controle da Taxa de Bits

6.JPEG-2000

I.COPPE/UFRJ II.Título (série)

Agradecimentos

- Aos meus pais por todo o apoio durante estes anos.
- Aos meus colegas de trabalho que me auxiliaram e me deram espaço para que eu pudesse realizar este projeto.
- A todos que estiveram do meu lado durante esta jornada, principalmente ao meu orientador Eduardo que acreditou no trabalho e ao José Fernando Leite.
- A Deus, que ilumina o meu caminho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CODIFICAÇÃO DE IMAGEM UTILIZANDO TURBO QUANTIZAÇÃO CODIFICADA POR TRELIÇAS

Marcelo Santos de Souza

Abril/2005

Orientador: Eduardo Antônio Barros da Silva

Programa: Engenharia Elétrica

Este trabalho é um codificador em sub-bandas para imagens monocromáticas, utilizando a técnica de quantização TTCQ *(Turbo Trellis Coded Quantization)*. O codificador utiliza diferentes algoritmos para compressão das sub-bandas. No caso de sub-bandas uniformes, a banda de baixa freqüência é transformada utilizando a DCT, e é modelada junto com as outras sub-bandas como distribuições de gaussianas generalizadas. No caso de sub-bandas não uniformes, utilizamos a transformada *wavelet* implementada por um banco de filtros em árvore, onde todas as sub-bandas são modeladas também como gaussianas generalizadas. Um método de otimização de alocação de bits é utilizado depois da modelagem, para determinar os parâmetros do turbo quantizador por treliça do codificador como, por exemplo, o passo de quantização para cada fonte e o número de palavras códigos. O tamanho do bloco de quantização é fixo para todas as fontes. Codificação aritmética é utilizada para codificar os símbolos da treliça, onde o contexto é modificado de acordo com o estado da treliça. Os resultados obtidos propõem que otimizações sejam realizadas para um melhor desempenho comparado à técnicas de compressão de imagem existentes.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

TURBO TRELLIS CODED QUANTIZATION BASED IMAGE CODING

Marcelo Santos de Souza

April/2005

Advisor: Eduardo Antônio Barros da Silva

Department: Electrical Engineering

This work presents a monochromatic image subband coder, using TTCQ (Turbo Trellis Coded Quantization) as the quantization process. The coder uses different subband decomposition. When using the uniform linear decomposition, the lower frequency subband is DCT transformed and then modeled, along with the other subbands as generalized gaussian distributions. In the case of non-uniform subband decompositions, a wavelet transform, implemented as a tree-structure filter bank, is used and all subbands are modeled as generalized gaussian distributions. An optimal bit allocation algorithm is implemented after modeling the subbands, to find the parameters of the trellis quantizer for each source, such as step size and number os codewords. Arithmetic coding is used to entropy-code the trellis symbols, where the context is changed upon the trellis state. The obtained results suggest that optimizations in the quantizer should be done for better achievements compared to other known image coders.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o avanço da ciência, e conseqüentemente da tecnologia, passamos a exigir um melhor desempenho e eficiência do uso de recursos. Do espectro de freqüência utilizado em transmissões de rádio-freqüência até o disco rígido, a tecnologia demanda por formas eficientes de comunicação e/ou armazenamento de dados na forma digital. Uma das principais ferramentas para tornar essa eficiência viável é a codificação de dados. Esta ferramenta é composta por três partes principais: a transformação, a quantização e a codificação. Na transformação, os dados originais são transformados em coeficientes que possuem características específicas, menos redundantes. A quantização corresponde ao processo de mapear os coeficientes em um número finito de símbolos, e a codificação é responsável pela representação de cada símbolo gerado a partir dos coeficientes, usando bits. Logo notou-se que técnicas eram necessárias para se chegar ao limite da eficiência de cada processo. Denominamos este processo como codificação de fonte. Na decodificação, recebemos os coeficientes e tentamos descobrir quais dados de entrada os geraram.

Em um sistema de comunicação, o codificador não fica junto do receptor. Então, depois de codificados, os coeficientes precisam ser transmitidos para o decodificador. Chamamos de canal o meio em que eles são transmitidos. Este canal insere ruído, o que conseqüentemente gera erros na decodificação. Para isso foram desenvolvidas técnicas para proteção dos dados na transmissão. Este processo é a codificação de canal. A codificação de canal transforma os coeficientes de entrada em símbolos que representarão uma forma de onda no transmissor. A decodificação consiste em receber a forma de onda, mapear no símbolo correspondente e descobrir

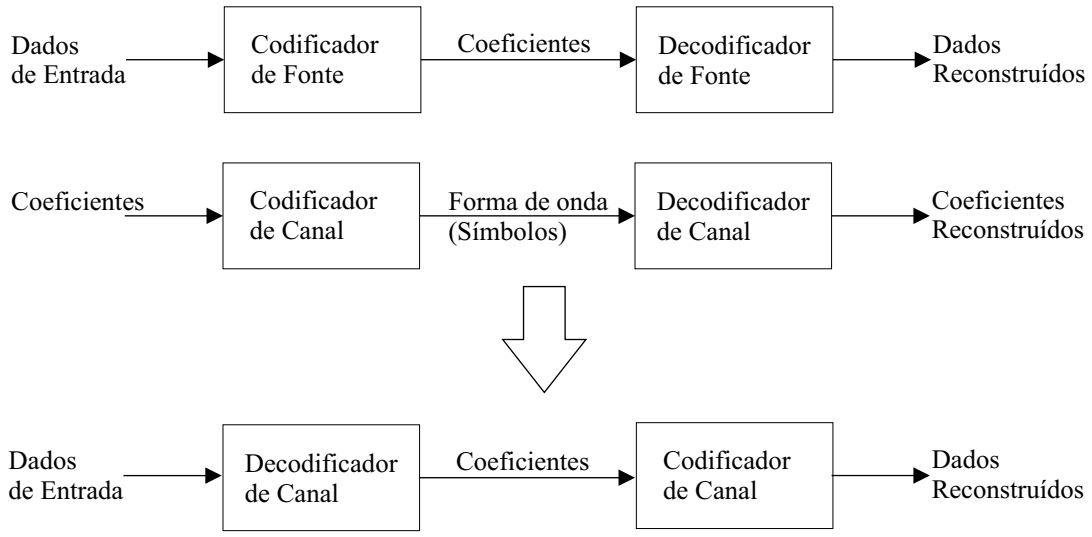


Figura 1.1: Codificação e decodificação de fonte utilizando técnicas de codificação e decodificação de canal.

quais coeficientes geraram este símbolo.

Normalmente, a codificação de fonte consiste em explorar as características da fonte e comprimir o sinal para uma certa taxa de bits e uma distorção determinada. A codificação de canal insere uma proteção para que o máximo de erros de transmissão possam ser corrigidos, e a informação transmitida possa ser recuperada [1].

Pela descrição anterior dos processos de codificação e decodificação podemos verificar a dualidade entre a teoria de codificação de fonte e a teoria de codificação de canal [3], onde implementamos da seguinte forma (figura 1.1): levamos o processo de decodificação de canal para a codificação de fonte, e o processo de codificação de canal para a decodificação de fonte. Neste novo codificador, os dados de entrada são mapeados em símbolos e depois transformados em coeficientes. O decodificador transforma os coeficientes recebidos em símbolos que serão mapeados em formas de onda, que representa os dados de entrada reconstruídos.

As codificações de canal estudadas foram a TCM *(Trellis Coded Modulation)* e a TTCM *(Turbo Trellis Coded Modulation)*. A partir da introdução da TCM, alguns estudos foram feitos na área de modulação vetorial [1]. Formas eficientes de gerar códigos, como os códigos Turbo, foram introduzidas [4]. A decodificação Turbo permitiu um aumento ainda maior na eficiência da transmissão. Em seguida,

os códigos Turbo foram agregados a TCM, que deu origem a TTCM. Esta técnica de modulação se mostrou bastante eficiente, como podemos verificar em [5].

A codificação de fonte que explora a dualidade entre a teoria de codificação de canal e a teoria taxa x distorção, é a TCQ *(Trellis Coded Quantization)*, e suas variações. Esta técnica apresentou ganhos em relação às quantizações uniformes e não-uniformes baseadas em níveis Lloyd-Max [6] [1]. Em [2], foi explorada a dualidade da TTCM através da implementação da TTCQ *(Turbo Trellis Coded Quantization)*. Verificou-se que com pequenas modificações do algoritmo, a TTCQ apresentou ganhos em relação à TCQ.

Em relação à compressão de imagens, várias técnicas surgiram nos últimos anos, com o advento da imagem e do vídeo digitais [1] [7] [8]. Como a representação de uma imagem possui duas dimensões, a quantidade de informação é bem maior do que na voz, por exemplo. Além disso, a voz humana é particular, pois é reproduzida pelo órgão fonador humano. Então, nem sempre as técnicas utilizadas em processamento de voz podem ser aplicadas diretamente à imagem. Em compressão de imagens, uma das técnicas de transformação mais conhecida é a transformação da imagem original em componentes de freqüência. Essa transformação diminui a distorção final do sistema e permite a exploração da redundância entre amostras próximas. Outra técnica é a divisão da imagem em sub-bandas na freqüência através de banco de filtros e a exploração de forma diferente de cada faixa de freqüência [9]. Dentro dessa família de técnicas de divisão em bandas está a transformada *wavelet* [1] [10]. Podemos descrever a transformada *wavelet* no domínio discreto, como uma implementação de banco de filtros estruturado em árvore. Uma importante aplicação da transformada *wavelet*, por ser uma técnica multi-resolução, é a transmissão progressiva de uma imagem comprimida, onde as sub-bandas de baixa freqüência são transmitidas primeiro. Esta técnica foi uma das propostas para o JPEG-2000 [7].

A contribuição do trabalho apresentado é a implementação da TTCQ de [2] em compressão de imagens. Estudou-se o codificador desenvolvido em [11], onde somente substituímos o bloco do quantizador. A imagem é dividida em sub-bandas utilizando transformada *wavelets*. Depois as sub-bandas são modeladas para determinarem os parâmetros do quantizador TTCQ. A modelagem das sub-bandas é

feita para uma distribuição de fontes Gaussianas Generalizadas (GGD), de acordo com estudo em [12]. Na saída do quantizador, aplica-se uma codificação aritmética adaptativa [1].

1.1 Organização da tese

A estrutura da tese apresentada tenta seguir a seqüencia das técnicas utilizadas no codificador de imagem proposto. No capítulo 2, temos referências aos algoritmos de codificação de fonte, passando pela quantização. Ainda neste capítulo, apresentamos a transformada *wavelet* e como implementá-la através da associação de banco de filtros. No capítulo 3, introduzimos a modulação codificada por treliças (TCM) e a posterior implementação de um quantizador codificado por treliças (TCQ). Neste capítulo temos também a replicação dos resultados de um codificador de imagens utilizando TCQ. No capítulo 4, estudamos os códigos Turbo e o algoritmo de decodificação Turbo, e apresentamos um turbo quantizador codificado por treliças. No último capítulo, 5, introduzimos a contribuição do trabalho, que é um codificador de imagens baseados em turbo quantização codificada por treliças, junto com os resultados obtidos e comparativos com resultados da literatura. A conclusão segue com o fechamento do trabalho e uma proposta de próximos estudos.

Capítulo 2

Técnicas de Compressão

Técnicas ou algoritmos de compressão se referem, na verdade, a dois algoritmos: o de compressão que recebe uma entrada e gera uma representação com um número menor de bits; e o de reconstrução que, a partir da representação, reconstrói a entrada do sistema [1] [13]. Como mencionado anteriormente, podemos separar três processos diferentes na compressão. A transformação dos dados de entrada em componentes com características específicas, menos redundantes. A quantização das componentes em representações limitadas e a codificação que determina a quantidade de bits de cada representação. Essas representações são realizadas através de códigos. Esses códigos pertencem a um conjunto de palavras (palavras códigos), formadas por letras que compõe um determinado alfabeto de elementos finitos [14]. Em uma rápida analogia, poderíamos chamar a Língua Portuguesa de um código, onde as palavras são compostas por combinações das 26 possíveis letras do alfabeto português. Existem vários modos de gerar esses códigos. Baseado no objetivo ao qual a compressão de dados se propõe, um sistema sem perdas gera uma saída que é idêntica à entrada, e um sistema com perdas gera uma saída que é ligeiramente diferente da entrada [1]. O desempenho dos codificadores é medido em termos de taxa de compressão, e no caso de algoritmos com perdas, também em distorção. O desenvolvimento de técnicas de compressão de dados, passa basicamente por duas fases: modelamento e codificação. No modelamento tenta-se extrair o máximo de informação da fonte em termos de redundância intrínseca e modelar essa redundância por modelos estatísticos, como o modelo de Markov. Com isso, aumentamos a eficiência da codificação, que tenta achar códigos que consigam representar o mod-

elo gerado [1] [13].

2.1 Codificação Aritmética

Dentro do ambiente de codificação de fonte, existem dois tipos de códigos: com comprimento fixo e variável. Quando a taxa é o limitante, os códigos de comprimento variável são mais eficientes, pois levam em consideração a probabilidade de cada palavra código acontecer dentro de uma mensagem. Esse tipo de código também pode ser chamado de código de entropia. Por definição [15], sabemos que a entropia de uma fonte determina a taxa mínima (bits/símbolo) em que ela poderá ser codificada.

$$H = -K \sum_{i=1}^{n} p_i \log p_i \tag{2.1}$$

Dado um conjunto de eventos independentes $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_n$ com probabilidades $p_i = P(\mathcal{A}_i)$, a entropia dessa fonte é calculada pela equação 2.1 [1]. Por convenção fazemos $K = 1$ para $\sum_{i=1}^{n} p_i = 1$, onde n é o número de eventos da variável aleatória $\mathcal{A}$.

Um característica importante que devemos saber é o tamanho de cada palavra do código ou seja a quantidade de bits que forma uma palavra.

$$l = \sum_{i=1}^{n} P(\mathcal{A}_i) n(\mathcal{A}_i) \tag{2.2}$$

Podemos medir o tamanho médio de um código com palavras pertencentes ao alfabeto $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_n$ através de uma média dos tamanhos de cada palavra, ponderada pela probabilidade de ocorrência do código, através da equação 2.2. Onde $n(\mathcal{A}_i)$ é o tamanho de cada palavra código associada à letra $\mathcal{A}_i$ do alfabeto.

Contudo, a medida do tamanho médio do código não é suficiente para dizermos que ele é um bom código. Outra condição necessária é que o código seja unicamente decodificável, ou seja, não pode haver ambiguidade na decodificação. Uma palavra código somente pode ser decodificada de uma maneira [1].

Dois códigos que atendem às especificações acima são o código Huffman e o código aritmético. O algoritmo de Huffman gera um código prefixo que é ótimo. O tamanho médio de um código de Huffman $\mathcal{S}$ está entre $H(\mathcal{S}) \leqslant \bar{l} < H(\mathcal{S}) + \frac{1}{m}$, onde m é a quantidade de letras agrupadas em blocos [1] [13]. Podemos ver que quando

aumentamos o tamanho do bloco, chegamos mais próximos do limite de Shannon. Mas a complexidade do dicionário de palavras códigos aumenta exponencialmente com o aumento do tamanho da seqüência a ser codificada [1]. Com a codificação aritmética, temos o tamanho médio do código $\mathcal{S}$ limitado em $H(\mathcal{S}) \leqslant \bar{l} < H(\mathcal{S}) + \frac{2}{m}$ [1] [13]. Podemos perceber que a para um mesmo tamanho de bloco de letras, a codificação de Huffman se aproxima mais do limite inferior, mas a grande vantagem do código aritmético é que não é necessário gerar todas as palavras para todos os comprimentos como o código de Huffman faz. A codificação aritmética é muito eficiente para alfabetos pequenos, como fontes binárias, e para alfabetos onde as probabilidades de cada letra são muito desbalanceadas.

O processo de codificação aritmética consiste em gerar identificadores únicos de cada conjunto de símbolos (palavras pertencentes ao alfabeto $\mathcal{A} = \{a_1, a_2, ..., a_n\}$), de modo a distinguir uma seqüência da outra [1]. Uma possibilidade encontrada foi utilizar os números reais no intervalo $[0, 1)$, com isso temos um número de identificadores distintos infinito. Consideremos uma variável aleatória $\mathcal{X}$

$$\mathcal{X}(a_i) = i \quad a_i \in \mathcal{A}$$

com função de densidade de probabilidade definida por

$$P(\mathcal{X} = i) = P(a_i)$$

Utilizando a função de distribuição acumulada da variável aleatória associada à fonte

$$F_{\mathcal{X}}(i) = \sum_{k=1}^{i} P(\mathcal{X} = k) \tag{2.3}$$

conseguimos mapear as palavras em um intervalo desejado [16].

Considerando o identificador como o ponto médio de um intervalo, o processo para gerar o identificador é reduzir o intervalo que o próprio identificador reside, de acordo com a função de distribuição acumulada (equação 2.3), ou seja, de acordo com a probabilidade de ocorrência daquele evento. Quanto mais provável o símbolo for, maior é o intervalo. O símbolo é representado por um intervalo único, que não se sobrepõe a nenhum outro intervalo. A cada iteração, o intervalo é dividido pela mesma função de distribuição acumulada. Isso pode ser ruim devido à precisão numérica computacional, onde mais e mais resolução numérica se torna necessária.

O final da seqüência é marcado por um código de final de seqüência (EOS), caso contrário o decodificador não teria como saber onde terminar [1] [13].

Seja uma fonte de entrada $\mathbf{x} = \{x_1, x_2, ..., x_n\}$. O identificador de x_1 seria representado pelo ponto médio do intervalo $[F(x_1) - p(x_1), F(x_1)) = [l_1, u_1)$. A seqüência x_1, x_2 é representada por $[l_2, u_2)$ onde

$$l_2 = l_1 + \frac{1}{u_1 - l_1}[F(x_2) - p(x_2)] \tag{2.4}$$

e

$$u_2 = l_1 + \frac{1}{u_1 - l_1}F(x_2) \tag{2.5}$$

e o identificador $T_X(\mathbf{x}) = \frac{u_n + l_n}{2}$. As equações 2.4 e 2.5 podem ser facilmente generalizadas para $[l_n, u_n)$ dado $[l_{n-1}, u_{n-1})$.

Vamos exemplificar para facilitar o entendimento. Considerando um alfabeto $\mathcal{A} = \{a_1, a_2, EOS\}$ com probabilidades $p(a_1) = 0.5$, $p(a_2) = 0.4$ e $p(EOS) = 0.1$, supomos que a entrada x seja a seqüência de letras $\{a_2, a_1, a_2, EOS\}$. Iniciando com o intervalo [0,1), fazemos as sucessivas divisões do intervalo depois de codificarmos um símbolo novo. A seqüência de intervalos neste caso é

$$[0, 1) \rightarrow [0.5, 0.9) \rightarrow [0.5, 0.7) \rightarrow [0.6, 0.68) \rightarrow [0.672, 0.680)$$

e o identificador da seqüência é 0.676.

O decodificador recebe o identificador da seqüência e tenta achar qual letra do alfabeto representa o intervalo no qual o identificador reside. O intervalo inicial é [0,1). Para o correto funcionamento, o decodificador precisa das mesmas tabelas de probabilidade de símbolos do codificador, caso contrário os intervalos não serão os mesmos. No caso de codificação adaptativa, as tabelas são atualizadas a cada decodificação do símbolo. No nosso exemplo, o primeiro símbolo decodificado seria a_2, pois 0.626 está no intervalo [0.5,0.9). Atualizamos agora o valor do intervalo e tentamos encontrar recursivamente onde o identificador se encaixa. Segue a ilustração do exemplo de codificação e decodificação aritmética descrito, na figura 2.1.

Para contornar o problema da resolução numérica, criaram-se técnicas de expansão dos limites dos intervalos e codificação incremental [1]. A técnica de expansão dos limites funciona da seguinte forma: depois que o intervalo for sub-divido, o novo intervalo é escalado para ficar entre $[0, 1)$ com isso aumentamos a faixa dinâmica

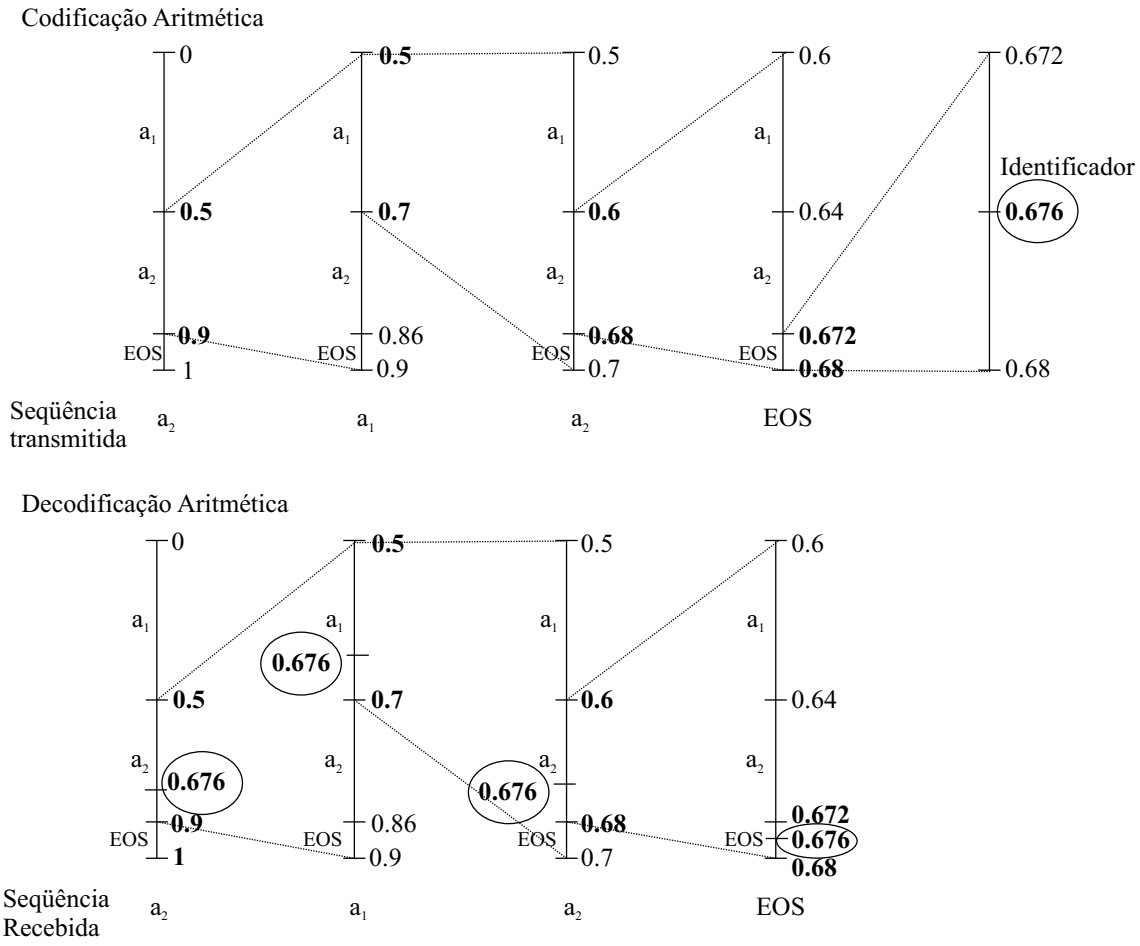


Figura 2.1: Ilustração do exemplo de codificação aritmética.

dos valores possíveis para o próximo intervalo. Quando não ocorre mudança do bit mais significativo, que representa o intervalo, ele é enviado progressivamente, o que chamamos de codificação incremental.

Uma modificação do algoritmo original, e que torna o codificador adaptativo, é a mudança do contexto durante a codificação de uma fonte. A mudança de contexto na verdade muda a função de densidade de probabilidade, para que a sub-divisão acompanhe as características estatísticas da variável da entrada, e possa ser mais eficiente [13]. No trabalho proposto, temos somente dois contextos.

2.2 Quantização

Nos esquemas de compressão com perdas, temos duas variáveis importantes: a taxa e a distorção. Como vimos anteriormente, a taxa é limitada pela entropia. A distorção é a medida que caracteriza o quanto o sinal de entrada é diferente da saída reconstruída. Neste tipo de processo, por aceitarmos uma perda de informação controlada do sinal de entrada, conseguimos taxas maiores de compressão em relação às técnicas sem perdas [1]. Os limites e a relação entre essas variáveis são estudados na teoria taxa-distorção [1]. Desejamos codificar uma fonte com o mínimo de distorção para uma mínima taxa. Os algoritmos de compressão com perdas na verdade inserem distorção no processo, levando em consideração, ou não, as características da entrada. Para isso, temos que ter medidas de desempenho como o

$$PSNR(dB) = 10 \log_{10} \frac{x_{pico}^2}{\sigma_d^2} \tag{2.6}$$

onde x_{pico} é o valor máximo do sinal de entrada e σ_d^2 é a medida de erro de quantização. Para imagens, a medida do PSNR, equação 2.6, é uma boa caracterização do processo de compressão, onde medimos a razão entre o valor máximo da fonte e a quantidade de erro (ruído) inserido. A idéia mais simples e geral para comprimir uma fonte com perdas, é a quantização [1].

Generalizado, a quantização é a técnica de representar um conjunto de valores por um outro conjunto menor de valores. Com isso temos perda de informação da fonte. Uma técnica de quantização é determinada pelo codificador e decodificador. Considerando uma escala de valores contínua, o codificador determina os intervalos desta escala que mapearão os níveis de reconstrução. Cada intervalo é representado

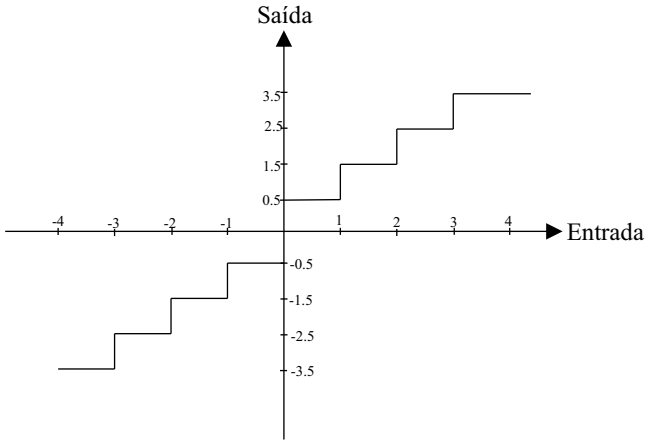


Figura 2.2: Mapeamento entrada-saída de um quantizador.

por uma palavra código. O decodificador mapeia as palavras códigos para os níveis de reconstrução definidos pelo codificador [1].

A figura 2.2 representa o mapeamento entrada-saída de um quantizador para melhor visualização. No caso, qualquer valor entre 0 e 1, seria mapeado para o nível 0.5, que poderia ser representado em binário por 001 (8 níveis de quantização - 3 bits).

Existem várias técnicas de mapeamento. Um meio de medir a qualidade de uma técnica é através do cálculo do msqe *(mean squared quantization error)*, também conhecido como erro de quantização ou ruído de quantização. Com essa medida podemos estabelecer critérios para definição dos níveis de reconstrução ou a taxa, que nos remete ao problema de taxa-distorção [3] [1] [17].

Em termos precisos, considere uma variável aleatória $\mathcal{X}$ com uma *pdf* (função densidade de probabilidade) $f_{\mathcal{X}}(x)$ que modele o sinal de entrada. Desejamos utilizar M intervalos no quantizador, gerando $M+1$ valores de bordas de decisão $\{b_i\}_{i=0}^{M}$ e M níveis de reconstrução $\{y_i\}_{i=1}^{M}$. Mesmo que o processo de quantização seja discreto, modelamos para distribuições contínuas pois simplificam o projeto. O erro médio quadrático de quantização, que mede a distorção do quantizador é dado por

$$\sigma_q^2 = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_{\mathcal{X}}(x) dx \tag{2.7}$$

O método mais simples é escolher o passo de quantização, que determinará a granularidade do mapeamento, para uma taxa fixa, ou seja, com tamanho de palavras

código fixa. Para este tipo de quantizador a taxa é determinada por

$$R = \lceil \log_2 M \rceil \tag{2.8}$$

onde $\lceil x \rceil$ é o menor número inteiro maior ou igual a x. Nos problemas onde utilizamos códigos com tamanho variável, temos que levar em consideração a probabilidade de ocorrência de cada intervalo, logo as bordas de decisão influenciam na taxa como vemos abaixo

$$R = \sum_{i=1}^{M} l_i \int_{b_{i-1}}^{b_i} f_{\mathcal{X}}(x)dx \tag{2.9}$$

Com isso dito, podemos reformular o problema de quantização da seguinte maneira [1] [3] [17]:

- Dada uma restrição de distorção

$$\sigma_q^2 \leqslant D^* \tag{2.10}$$

  ache as bordas de decisão, níveis de reconstrução e os códigos binários que minimizem a taxa da equação 2.9 satisfazendo 2.10

- Dada uma restrição de taxa

$$R \leqslant R^* \tag{2.11}$$

  ache as bordas de decisão, níveis de reconstrução e os códigos binários que minimizem a distorção da equação 2.7 satisfazendo 2.11

Existem várias formas de achar uma solução aproximada para esse problema, com diversas complexidades envolvidas. A forma mais simples é a quantização uniforme. Neste tipo de quantização, os níveis de quantização possuem passos regulares e iguais. No caso da distribuição uniforme, podemos apenas dividir a faixa dinâmica do sinal pelo número de níveis. No caso da entrada possuir uma distribuição não uniforme, podemos utilizar a sua *pdf* para projetar os intervalos. Nosso objetivo é minimizar a distorção achando o passo de quantização para um dado número de níveis M [1] [17].

No caso das técnicas mencionadas acima, modelamos a fonte segundo uma dada distribuição e consideramos que as fontes são estacionárias. A adaptação do quantizador à entrada é um fator muito importante e tende a diminuir o erro de reconstrução. Determina-se, a cada conjunto de amostras (blocos), a distribuição

correspondente, e modifica-se o quantizador. Com isso existe a necessidade da transmissão de uma informação adicional para efeito de sincronização. A escolha do número de amostras que formam um bloco é uma troca. Para blocos pequenos, o tamanho dessa informação adicional cresce. Para blocos grandes, temos a perda da fidelidade do modelo de distribuição. Este tipo de adaptação, onde olhamos as características da entrada para modificar o quantizador, é chamado de *forward adaptive quantization* [1] e será utilizado no trabalho.

Uma forma para achar o melhor quantizador não-uniforme, tendo o modelo de probabilidade da fonte, é achar os níveis de reconstrução e as bordas dos intervalos que minimizem a equação 2.7 [1]. Resolvendo a derivada e igualando a zero temos os níveis de reconstrução

$$y_j = \frac{\int_{b_{j-1}}^{b_j} x f_{\mathcal{X}}(x) dx}{\int_{b_{j-1}}^{b_j} f_{\mathcal{X}}(x) dx} \tag{2.12}$$

O centróide é calculado para cada intervalo. Tirando a derivada em relação a b_j, e igualando a zero, temos o ponto médio entre dois intervalos subseqüentes

$$b_j = \frac{y_{j+1} + y_j}{2} \tag{2.13}$$

Um algoritmo para resolver estas equações é devido a Lloyd Max, que pode ser achado em [1]. A otimização do passo de quantização e dos níveis de reconstrução aumentam a eficiência do quantizador. No nosso trabalho, utilizaremos passos regulares, com otimização dos níveis de reconstrução.

Quando a taxa é uma medida de desempenho, podemos trabalhar com os quantizadores por entropia, onde os códigos que representam os níveis de reconstrução possuem tamanho variável. Existem duas formas básicas de gerar os níveis. A primeira, e mais complexa, é ditar uma restrição de entropia para a saída do quantizador, e com isso recalcular os níveis de reconstrução [1]. A segunda, e mais simples de implementar, é manter o projeto do codificador como anteriormente (ex. Lloyd Max) e aplicar um codificador entropia nos códigos gerados [1].

Até agora, comentamos somente dos quantizadores escalares. Mesmo quando consideramos blocos de amostras para determinar os parâmetros do quantizador, continuávamos quantizando amostra a amostra. No caso de quantizadores vetoriais, um conjunto de amostras serve de entrada para o quantizador. Esta entrada será quantizada por um conjunto de seqüências representativas daquele tipo de fonte.

Essas seqüências representativas são vetores-códigos que pertencem a um dicionário de vetores. O vetor de entrada é então comparado com as palavras (vetores) do dicionário e o índice da palavra mais próxima, em termos de erro médio quadrático, é transmitido. Em termos precisos, considerando um dicionário $\mathcal{C}$ com K vetores-código $\{Y_i\}$ com $i = 0, 1, ..., K$, e um vetor de entrada $X = \{x_1, x_2..., x_L\}$, dizemos que o índice j é transmitido quando

$$\|X - Y_j\|^2 \leqslant \|X - Y_i\|^2 \tag{2.14}$$

para todo $Y_i \in \mathcal{C}$, sendo que

$$\|X\|^2 = \sum_{i=0}^{L} x_i^2 \tag{2.15}$$

O algoritmo ótimo para achar os parâmetros do quantizador, no caso da quantização vetorial, é o LBG (Linde-Buzo-Gray) ou também chamado de algoritmo generalizado de Lloyd. Este algoritmo forma a base para muitos algoritmos de quantização vetorial [1].

Outro algoritmo, que pode ser considerado uma quantização vetorial, é utilizado na TCQ (*trellis coded quantization*). No capítulo 3 explicaremos melhor essa técnica que é o ponto de início do trabalho. Na verdade a TCQ processa amostra a amostra. No entanto, como a decisão pela palavra código que representa a seqüência de entrada é tomada depois de um conjunto de amostras, esta técnica cabe nesta classificação de quantizadores. Esse atraso na decisão é para diminuir a distorção média [1] [3].

2.3 Codificação por transformadas

A transformação da entrada, como um dos processos da codificação de fonte, é realizada por diversas técnicas que mudam o espaço de representação da entrada. Como o espaço é representado por bases, quando aplicamos a transformada mudamos as bases de representação, gerando um coeficiente para cada elemento transformado. A codificação por transformada é a exploração de forma diferente desses coeficientes, de acordo com suas características individuais. O objetivo principal da transformada em um algoritmo de codificação é compactar a energia da fonte em um número menor de coeficientes, para que esses coeficientes sejam quantizados e codificados, resultando na menor distorção total possível [1] [9].

As transformadas que conseguem uma maior compactação de informação, medida pela energia da fonte (no caso de variáveis aleatórias com média zero, a energia da fonte é igual a variância σ^2), são aquelas que melhor descorrelacionam a entrada, ou seja, tornam a correlação entre amostras de componentes diferentes nula [9]. A transformada ótima que atende a este critério é a Karhunen-Loéve, como veremos em seguida. Seguimos com a caracterização de algumas transformadas importantes para o trabalho.

A transformada direta v_n de um vetor de entrada u_n pode ser representada por

$$v_n = \sum_{i=0}^{N-1} u_i a_{n,i} \tag{2.16}$$

em representação matricial

$$\mathbf{v} = \mathbf{A}\mathbf{u} \tag{2.17}$$

onde $\mathbf{A}$ é uma matriz unitária, isto é, uma matriz cuja inversa é igual à matriz complexa conjugada transposta, $\mathbf{A}^{-1} = \mathbf{A}^{*T}$ [9]. A transformada inversa pode ser vista da seguinte forma

$$u_n = \sum_{i=0}^{N-1} v_i b_{n,i} \tag{2.18}$$

em representação matricial

$$\mathbf{u} = \mathbf{B}\mathbf{v} \tag{2.19}$$

onde $\mathbf{B} = \mathbf{A}^{*T}$. Para o caso de transformada de imagem, ou seja, bidimensional, utilizamos uma transformada separável na imagem [9], onde aplicamos a transformada primeiro nas colunas e depois nas linhas, ou vice-versa, conforme

$$\mathbf{V} = \mathbf{A}\mathbf{U}\mathbf{A}^T \tag{2.20}$$

e a transformada inversa

$$\mathbf{U} = \mathbf{B}\mathbf{V}\mathbf{B}^T \tag{2.21}$$

Por estarmos sempre mapeando um espaço com bases ortonormais, a matriz de transformada inversa é simplesmente a transposta da matriz complexo conjugada direta $\mathbf{B} = \mathbf{A}^{-1} = \mathbf{A}^{*T}$, e com isso mantemos o princípio de conservação de energia [9]. A equação 2.20 pode ser reescrita como

$$\mathbf{V}^T = \mathbf{A}[\mathbf{A}\mathbf{U}]^T \tag{2.22}$$

A freqüência associada a cada componente está relacionada à posição na seqüência final. O tamanho do bloco para aplicar a transformada deve ser escolhido por meios práticos. Um bloco muito pequeno pode significar um custo computacional muito grande enquanto um bloco muito grande pode esconder variações estatísticas da fonte.

A transformada de Karhunen-Loéve é ótima no sentido que ela maximiza a concentração de energia em poucos coeficientes, descorrelacionando a entrada, e com isso maximizando o ganho de codificação (razão entre o média aritmética das variâncias das componentes da transformada, e a média geométrica das mesmas [9] [1]). O problema da transformada ótima é que ela precisa conhecer a saída da fonte, e o receptor não tem esta informação, sendo necessário o envio de informações adicionais (a matriz de auto-correlação ou a própria transformada). Para fontes não-estacionárias isto representaria um aumento muito grande em informações adicionais.

Com isso surgiram as transformadas que independem da fonte, como a DFT (*Discrete Fourier Transform*), DCT *(Discrete Cossine Transform)*, DST *(Discrete Sine Transform)* e Haddamard [9]. Cada transformada possui uma característica diferente. A DCT é a que mais se aproxima da transformada ótima, para fontes modeladas como seqüências de Markov, com alto coeficiente de correlação, que é o caso de imagens onde os pixels próximos possuem uma correlação alta.

A matriz de transformada da DCT, $\mathbf{C}$ pode ser obtida a partir de funções bases de cossenos.

$$[\mathbf{C}]_{ij} = \begin{cases} \sqrt{\frac{1}{N}} \cos \frac{(2j+1)i\pi}{2N} & \text{i=0, j=0,1,...,N-1} \\ \sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N} & \text{i=1,2,...,N-1, j=0,1,...,N-1} \end{cases} \tag{2.23}$$

Pela equação 2.23 podemos comprovar que a posição da componente da transformada, indicará qual sua freqüência correspondente.

Da mesma forma que um vetor pode ser representado por série de funções ortogonais, a imagem pode ser representada por um conjunto de imagens base [9]. Na figura 2.3 temos as imagens bases para uma DCT 4x4. A freqüência cresce de cima para baixo e da esquerda para direita. Outro fato importante de lembrar para implementações, é que a DCT é muito próxima da DFT. Na verdade, a DCT pode ser obtida a partir do espelhamento da seqüência original e uma posterior

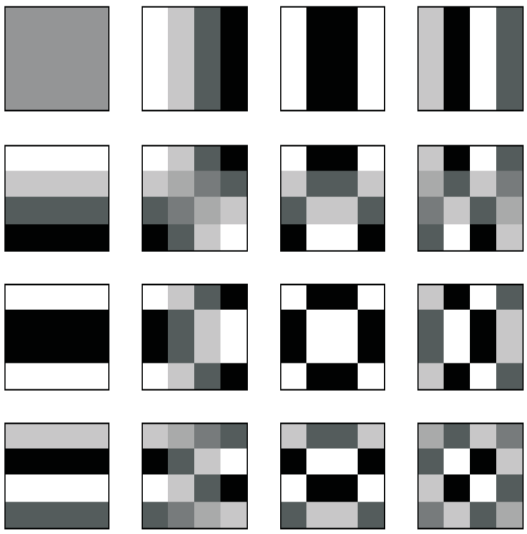


Figura 2.3: Imagens bases de uma DCT 4x4.

transformada "$2N$-*point*" DFT [9]. Com isso, podemos aproveitar as implementações computacionais otimizadas para a DFT [18].

Devido a essa facilidade computacional e ao alto desempenho para codificação de imagem, a DCT foi adotada pelos principais padrões internacionais de compressão de imagem e de vídeo como o MPEG [8].

No codificador proposto utilizamos a DCT como um dos métodos para transformar a imagem de mais baixa freqüência depois de uma análise em sub-banda.

2.4 Codificação em Sub-bandas utilizando Transformadas *Wavelet*

O princípio básico da codificação em sub-bandas é dividir o sinal de entrada em bandas de freqüência para que cada banda possa ser codificada de forma diferente como, por exemplo, utilizando um modelo estatístico diferente, atendendo aos requisitos de cada codificador. Para esta decomposição são utilizados bancos de filtros digitais [19] [1] [18], que são conjuntos de filtros passa-baixa e passa-alta com entrada e saída em comum. O processo de decomposição é chamado de análise e possui filtros de análise e decimadores. O processo de reconstrução do sinal em sub-banda é a síntese e possui interpoladores e filtros de síntese.

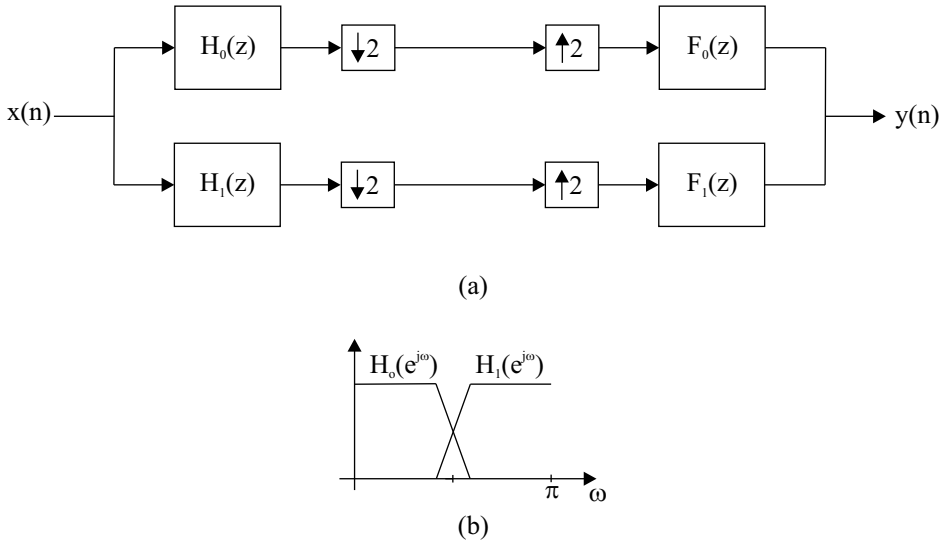


Figura 2.4: Banco de filtros QMF (a) estrutura de implementação (b) resposta na frequência típica dos filtros de análise.

O projeto de um banco de filtros deve ser realizado a partir de certas propriedades desejadas, tais como reconstrução perfeita, ou ausência de distorção de amplitude ou de fase [19]. Um banco de filtros que é bastante popular e implementado em várias aplicações atuais é o QMF (*Quadrature Mirror Filter*) devido à baixa complexidade de implementação e ordem dos filtros reduzida para um bom desempenho [20]. A partir dele podemos gerar estruturas mais complexas através de implementações em árvore como veremos em seguida.

Um banco de filtro QMF com reconstrução perfeita (PR) é aquele que é livre de todo tipo de distorção [19]. Esta afirmação é equivalente dizer que a função de transferência resultante do banco de filtros $T(z) = cz^{-n_0}$, ou seja, a versão reconstruída $y(n)$, é somente um escalonamento c e um atraso de n_0 amostras da entrada $x(n)$. Uma forma generalizada para projetar um banco de filtros PR é [18] [19].

$$\begin{aligned} T(z) &= cz^{-n_0} = H_0(-z)H_1(z) - H_0(z)H_1(-z) \\ F_0(z) &= -\frac{z^{-2n_0-1}}{c}H_1(-z) \\ H_1(z) &= \frac{z^{-2n_0-1}}{c}H_0(-z) \end{aligned} \tag{2.24}$$

Para o caso de querermos mais de 2 sub-bandas resultantes do banco de filtro podemos trabalhar com os banco de filtros M-ários. Mas a complexidade envolvida no projeto desses filtros aumenta muito com o número de sub-bandas. Então

implementou-se formas eficientes em árvores hierárquicas, usando bancos de filtros de dois canais, onde o filtro resultante possui as mesmas características do filtro que gerou a árvore, por exemplo, reconstrução perfeita [19] [18].

Pode-se implementar uma decomposição hierárquica uniforme, onde todos os ramos da árvore passam pela mesma decimação, gerando sub-bandas de tamanho iguais, ou então pode-se desbalancear a árvore e utilizar uma decomposição hierárquica em oitavas. Podemos ver na figura 2.5, as diferentes representações hierárquicas em árvore. Os sinais $x_k(n)$ são o resultado da filtragem e decimação no ramo k. Na decomposição uniforme de S estágios, geramos 2^S decomposições. Na decomposição em oitavas de S níveis geramos $S + 1$ decomposições. Com o banco de filtros resultante de uma representação hierárquica em oitavas, verificamos que cada faixa de freqüência possuirá uma banda diferente e uma amostragem diferente, por isso chamamos esta técnica de multi-taxas e multi-resolução. Em codificadores de imagem, estamos interessados em dar ênfase nas bandas baixas de freqüência [1]. As bandas de baixas freqüências de uma imagem possuem muita energia. As bandas de altas freqüências são responsáveis pelo contorno, normalmente possuem baixa energia [9]. Por isso, conforme o nível de decomposição vai aumentando, diminuímos a resolução no tempo (taxa de amostragem) das bandas de baixas freqüência.

Pode-se provar que a envoltória da resposta ao impulso do filtro resultante após a decomposição hierárquica tem o mesmo padrão para todos os nós, sendo que a mudança a cada ramo no banco de filtro é a expansão ou contração desta função única $\psi(t)$ [18]. Um sistema equivalente ao banco de filtros ilustrado anteriormente, no mundo contínuo, pode ser representado na figura 2.6 [18] [19]. Para representar a mesma saída, primeiro filtramos o sinal contínuo com filtros com resposta ao impulso idênticas às envoltórias do esquema discreto. Neste esquema contínuo os decimadores equivalem à taxa de amostragem do sistema dividida pelo fator de decimação equivalente de cada ramo. Verificamos que ao adicionar um ramo para cima, para mais baixa freqüência, dobramos a largura da resposta ao impulso do filtro e diminuímos a taxa de amostragem pela metade em relação ao ramo anterior. Se adicionamos um ramo abaixo, diminuímos a largura da resposta ao impulso pela metade e dobramos a taxa de amostragem em relação ao anterior. A transformada *wavelet* do sinal de entrada consiste na saída deste sistema quando adicionamos

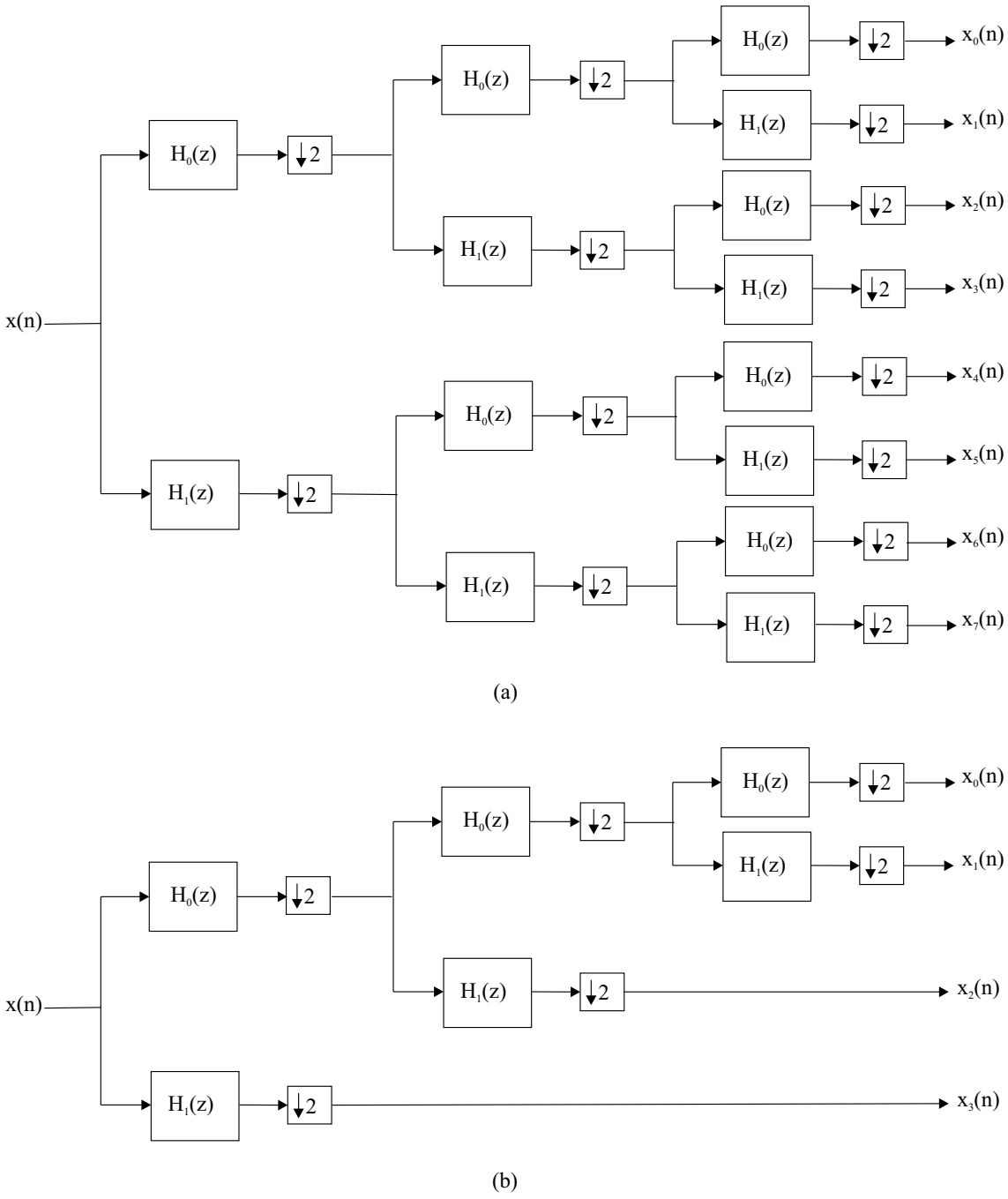


Figura 2.5: Representações hierárquicas de banco de filtros em árvore para decomposições em 3 níveis (a) decomposição uniforme (b) decomposição em oitavas.

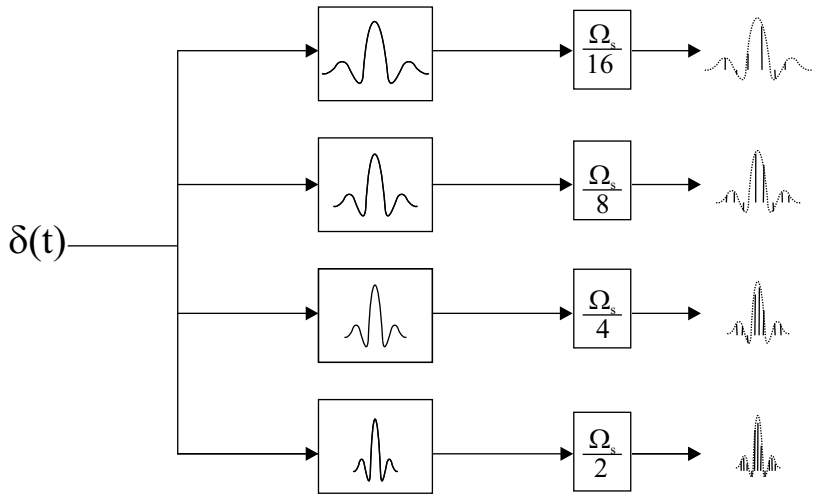


Figura 2.6: Representação do banco de filtros no mundo contínuo. As respostas ao impulso dos filtros possuem a mesma envoltória.

ramos infinitos tanto para baixas freqüências quanto para altas freqüências [18]. A função mãe $\psi(t)$ que determina esta envoltória da resposta ao impulso é chamada de *wavelet* ou *wavelet* de análise.

Em termos matemáticos, considerando a freqüência de amostragem do sistema $\Omega_s = 2\pi$, temos que a transformada *wavelet* de um sinal $x(t)$ é

$$c_{m,n} = \int_{-\infty}^{\infty} 2^{-\frac{m}{2}} \psi(2^{-m}t - n)x(t)dt \tag{2.25}$$

De modo similar podemos considerar uma *wavelet* de síntese $\overline{\psi}(t)$, e para reconstruir o sinal contínuo temos

$$x(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_{m,n} 2^{-\frac{m}{2}} \overline{\psi}(2^{-m}t - n) \tag{2.26}$$

Para utilizar estas formulações no mundo discreto, que é o nosso caso, temos que limitar o número de canais que utilizaremos para reconstruir o sinal. Considerando uma decomposição de S estágios que geram $S + 1$ canais, precisamos de funções escalonadoras que levem em consideração esta restrição. Da mesma forma teremos funções escalonadoras de análise, $\phi(t)$ e de síntese $\overline{\psi}(t)$. Então na prática, para uma decomposição de $(S + 1)$ canais, a transformada *wavelet* é representada pela função 2.25 e pelas duas funções abaixo

$$x(t) = \sum_{n=-\infty}^{\infty} x_{S,n} 2^{-\frac{S}{2}} \overline{\phi}(2^{-S}t - n) + \sum_{m=0}^{S-1} \sum_{n=-\infty}^{\infty} c_{m,n} 2^{-\frac{m}{2}} \overline{\psi}(2^{-m}t - n) \tag{2.27}$$

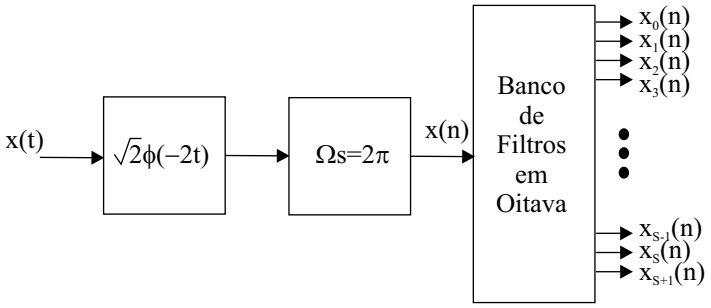


Figura 2.7: Transformada *wavelet* de um sinal contínuo no tempo de modo prático.

onde

$$x_{S,n} = \int_{-\infty}^{\infty} 2^{-\frac{S}{2}} \phi(2^{-S}t - n)x(t)dt \tag{2.28}$$

É interessante observar que a transformada *wavelet* é definida somente para os sinais contínuos no tempo. Para que exista a equivalência para o mundo discreto (decomposição em oitavas por banco de filtros), pela equação 2.28 temos que

$$x(n) = \int_{-\infty}^{\infty} \sqrt{2}\phi(2t - n)x(t)dt \tag{2.29}$$

que seria equivalente ao esquema da figura 2.7.

A relação entre os coeficientes dos filtros que compõem o banco de filtros e as *wavelets* pode ser encontrada em [18]. Quando temos $\phi(t) = \overline{\phi}(t)$ e $\psi(t) = \overline{\psi}(t)$ dizemos que a transformada *wavelet* é ortogonal. Do contrário ela é somente bi-ortogonal. Outro fato é que o banco de filtros para implementação da transformada *wavelet* discreta precisa ser PR.

Utilizamos a estrutura de banco de filtros com decomposições em oitavas para calcular a transformada de um sinal discreto no tempo. Somente alguns bancos de filtros QMF podem ser utilizados, caso atendam as restrições de regularidade, que determina a quantidade de derivadas contínuas de uma *wavelet*[18].

A transformada *wavelet* pode ser aplicada em imagens, ou seja, transformada bi-dimensional. Para isso, implementamos uma decomposição separável [9]. Isto significa que podemos trabalhar com transformadas uni-direcionais, aplicando a transformada primeiro nas linhas das imagens e depois nas colunas [18] [19]. Cada conjunto de transformadas na linha e depois na coluna corresponde a um nível de decomposição da imagem. Na figura 2.8 podemos verificar como a imagem seria decomposta para uma transformada bi-direcional separável. Na figura 2.9, vemos o

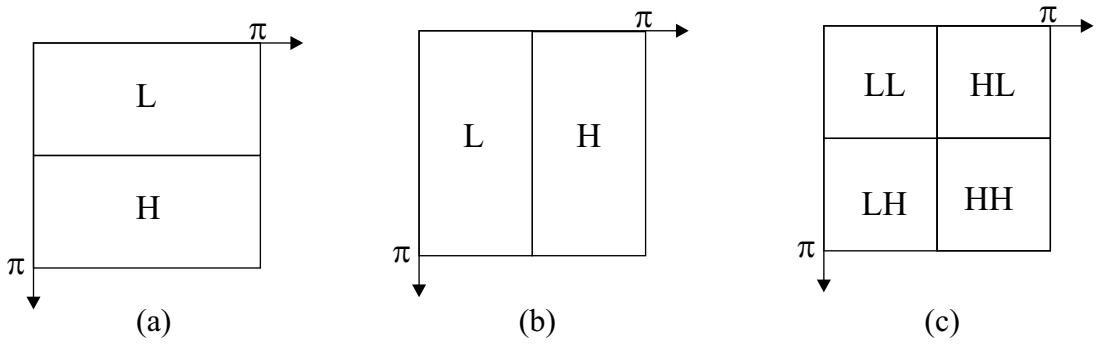


Figura 2.8: Um nível de decomposição separável de uma imagem. Primeiro aplicamos a transformada nas linhas (a), depois nas colunas (b), e o resultado final (c).

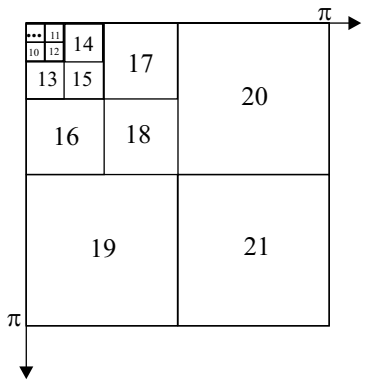


Figura 2.9: Decomposição não-uniforme em 22 bandas por transformada *wavelet*.

resultado para uma transformada *wavelet* com 7 níveis de decomposição.

Capítulo 3

Codificação com Quantização Codificada por Treliças

O principio das técnicas ou algoritmos de modulação, é mapear um conjunto de bits de entrada em símbolos, que representam uma forma de onda no transmissor. As técnicas de modulação se diferem em relação à eficiência desse mapeamento, de modo que o receptor consiga reconstruir mensagem com um mínimo de perda, ou seja, com uma probabilidade de erros pequena. No desenvolvimento de técnicas de modulação inserimos redundância de maneira controlada, através de códigos corretores de erro, e depois mapeamos o resultado em símbolos de acordo com a transmissão desejada [21] [14]. Neste sentido, um bom codificador de canal gera seqüências de símbolos afastadas o suficiente no espaço, de modo que o decodificador consiga distinguir, dentre um conjunto de possíveis seqüências, a mais próxima das seqüências de símbolos recebidas, mesmo com adição de ruído, minimizando a probabilidade de erros de decisão. Como vimos no capítulo anterior, um bom quantizador diminui a distorção média entre a seqüência original e a quantizada. Veremos em seguida como a dualidade entre a teoria de codificação de canal e a teoria taxa-distorção pode ser explorada, onde bons codificadores de canal podem ser utilizados como quantizadores.

3.1 Modulação TCM

Nesta seção sobre modulação, introduziremos a TCM (*Trellis Coded Modulation*) que é uma das motivações do trabalho. Tanto a TCQ (*Trellis Coded Quantization*) de [6], quanto o nosso codificador TTCQ (*Turbo Trellis Coded Quantization*) partem da TCM de [22].

Novamente, a modulação em sistemas digitais consiste em mapear os bits de entrada em símbolos para transmissão. Esses símbolos serão responsáveis pela caracterização da forma de onda a ser gerada no transmissor [21]. O desempenho de cada técnica de modulação é medido em eficiência espectral e probabilidade de erro na recepção quando os símbolos são transmitidos em canal ruidoso. Vamos definir essas duas métricas.

A eficiência espectral é definida como o número de bits por segundo transmitido em 1Hz de banda. Considerando as modulações multi-níveis, M-árias, quanto maior a ordem, maior a eficiência. Pela ordem da modulação, 2^m, sabemos que m bits são mapeados em 1 símbolo. O erro de transmissão é calculado através da probabilidade que um canal ruidoso tem de degradar o sinal transmitido de tal forma, que os símbolos não são identificados corretamente no receptor. A decisão pelo símbolo recebido, pode ser feita através de receptores de máxima verossimilhança, que calculam a distância Euclidiana dos símbolos do sinal recebido com cada posição no mapa de possíveis símbolos do sistema (também chamado de constelação). A menor distância é o critério de escolha [14] [21].

Para aumentar a robustez das técnicas de modulação em relação à presença de ruído no canal de transmissão, os símbolos podem ser codificados. Diferente da codificação de fonte, onde exploramos a redundância da entrada para comprimir o sinal, na codificação de canal, inserimos redundância através de bits de paridade de modo que o símbolo esteja mais protegido. Somente aumentar a eficiência da modulação não é uma boa prática, pois com isso aumentamos o número de símbolos na constelação e diminuímos a distância entre eles, para uma mesma potência de transmissão [21], ou seja, aumentamos a probabilidade de erro no receptor.

As técnicas de codificação de símbolos são usualmente aplicadas de forma independente da modulação. Antes de ser modulado, o sinal passa por um codificador de canal e depois pelo processo de modulação. Alguns dos códigos corretores

de erro mais utilizados são os códigos convolucionais e por blocos, que aumentam o ganho de codificação adicionando bits de paridade [14]. Esses bits de paridade são utilizados na correção de erros. A quantidade de redundância inserida em um sistema é medida pela taxa do codificador r, razão entre a quantidade de bits de informação da entrada do codificador e a quantidade de bits transmitida. Podemos dizer então que o aumento na taxa a ser transmitida é de $1/r$. Como normalmente temos uma restrição de banda de transmissão, precisamos de formas eficientes para gerar redundância sem aumentar a taxa transmitida. A TCM apresenta uma forma de aumentar a eficiência da modulação, inserindo redundância através de uma codificação convolucional, sem aumentar a taxa de informação transmitida, atacando os problemas de codificação e de modulação em conjunto [22].

O algoritmo de implementação da TCM, para uma modulação 2^m-ária, consiste resumidamente em:

1. Adicionar um bit de paridade a cada m bits de informação;
2. Expandir a constelação do sinal de 2^m para 2^{m+1} ;
3. Utilizar os $(m+1)$ bits codificados para selecionar os símbolos da constelação.

A expansão da constelação é um dos grandes avanços de Ungerboeck, onde R bits de informação são mapeados em uma constelação com 2^{R+1} símbolos. A constelação principal, ou mãe, se pensarmos em um modelo de árvore, é particionada de forma hierárquica, de modo que as constelações filhas possuam uma distância entre os símbolos pertencentes a elas maior, diminuindo a probabilidade de erro no receptor. No final das decomposições, cada constelação filha terá um código associado que está diretamente relacionado ao caminho na árvore de particionamento. Temos um exemplo deste particionamento na figura 3.1. Podemos ver que à medida que descemos em um nível na árvore, aumentamos a distância no espaço entre os símbolos dentro de uma sub-constelação. O código binário correspondente de cada símbolo pode ser dado pelo caminho na árvore.

Descrevemos agora o codificador de um sistema de modulação codificada por treliças (TCM), também ilustrado na figura 3.2. Para uma entrada $u_0, u_1, ..., u_{R-1}$ com R bits de informação, que será mapeada em uma constelação 2^{R+1}-ária, n bits são codificados por um codificador convolucional com taxa de $n/(n+1)$. Os $(n+1)$

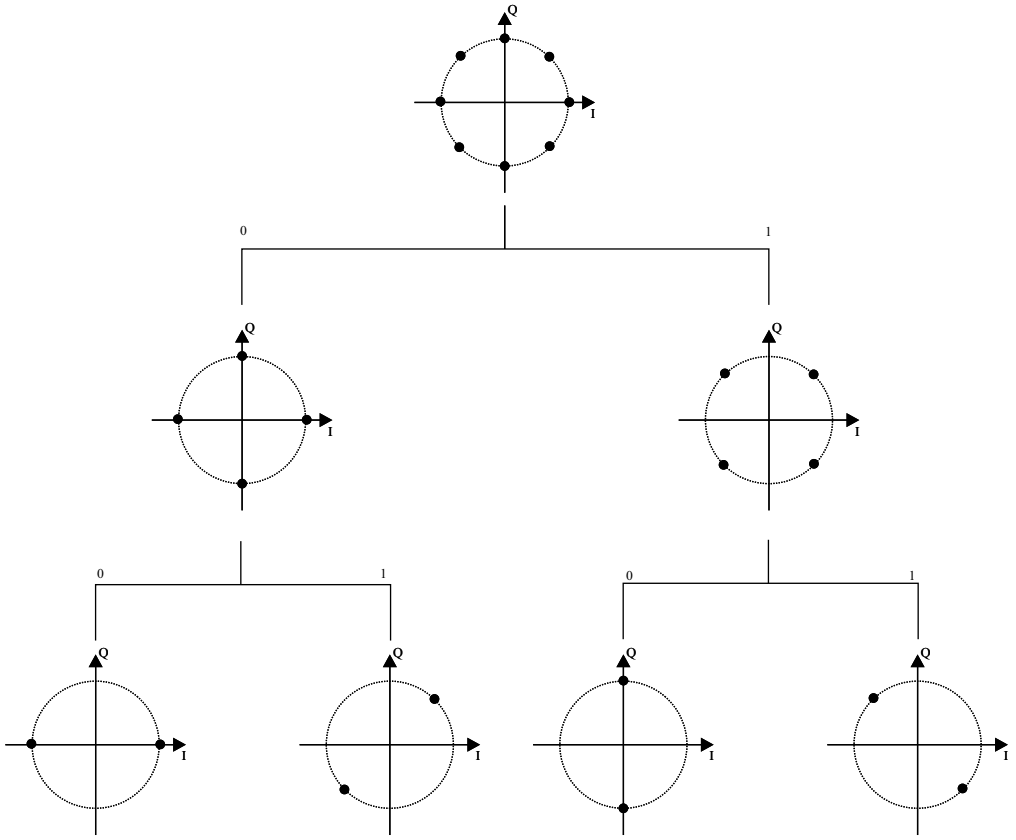


Figura 3.1: Exemplo de particionamento de uma constelação 8-PSK.

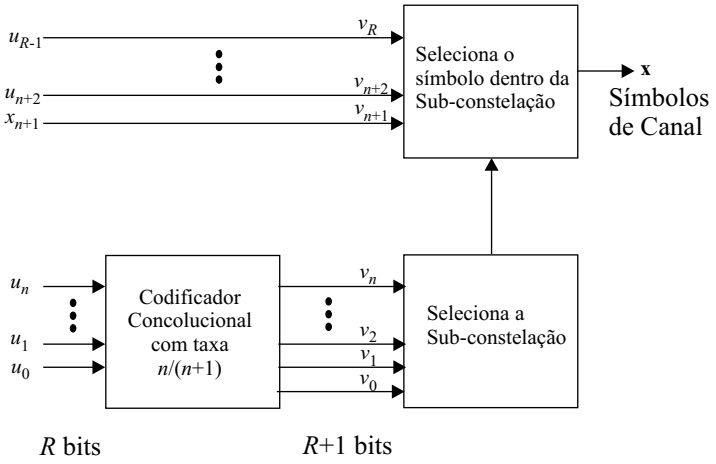


Figura 3.2: Estrutura do codificador na modulação codificada por treliças (TCM) [1].

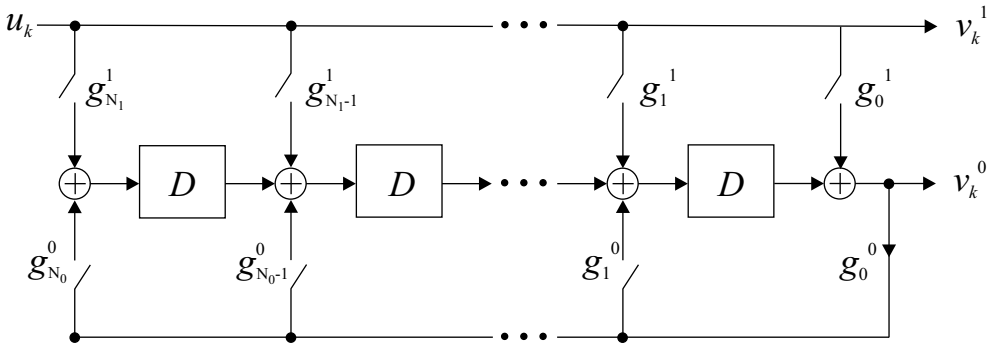


Figura 3.3: Codificador convolucional com taxa 1/2.

bits resultantes são utilizados para selecionar uma das 2^{n+1} partições da constelação 2^{R+1}-ária no nível $(n-1)$ da árvore de partição. Os $(R-n)$ bits remanescentes serão responsáveis por selecionar o símbolo dentro da partição determinada pelos $(n+1)$ bits do codificador.

Em uma modulação normal, não temos memória das escolhas passadas, ou seja, a cada transmissão podemos escolher qualquer símbolo da constelação. Na TCM, inserimos memória através dos registradores do codificador convolucional. A taxa do codificador convolucional de Ungerboeck é 1/2 [22], ou seja, para cada bit de entrada adicionamos 1 de paridade. Pelas nossas definições, isto significa que $n = 1$. Ilustramos um codificador convolucional com estas características na figura 3.3. Pela estrutura desenhada, podemos especificar um codificador convolucional pela posição das chaves que ligam os somadores da figura 3.3. Desta forma, representamos estas chaves por polinômios geradores binários através do operador D (atraso) [14], da seguinte forma

$$\begin{aligned} g^0(D) &= g^0_{N_0} D^{N_0} + g^0_{N_0-1} D^{N_0-1} + ... + g^0_1 D + g^0_0 \\ g^1(D) &= g^1_{N_0} D^{N_0} + g^1_{N_0-1} D^{N_0-1} + ... + g^1_1 D + g^1_0 \end{aligned} \tag{3.1}$$

sendo que $g^0(D)$ corresponde a parte recursiva. Como estamos representando somente dois estados das chaves da figura 3.3, ligado ou desligado, os coeficientes do polinômio pertencem a $\mathcal{G} = \{0, 1\}$ [14]. Abaixo estão dois exemplos de polinômios geradores

$$\begin{aligned} g^0(D) &= D^5 + D^3 + 1 \\ g^1(D) &= D^2 \end{aligned} \tag{3.2}$$

O número de estados do código convolucional é igual a $2^{\max\{N_0, N_1\}}$. Outra repre-

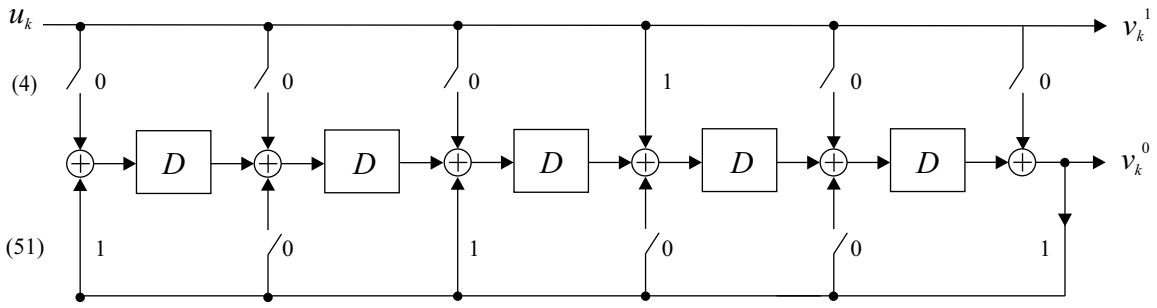


Figura 3.4: Construção de um codificador convolucional (51,4).

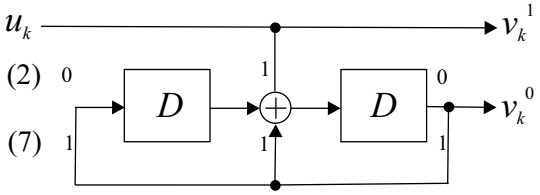


Figura 3.5: Construção de um codificador convolucional (7,2).

sentação do código convolucional pode ser em octal. Nesta forma, agrupamos os coeficientes binários dos polinômios geradores da esquerda para direita, de três em três, e convertemos para decimal. No exemplo, os coeficientes de g^0 em binário são 101001 e g^1, 000100. Agrupando de três em três temos o par (101 001,000 100). Passando para decimal, cada grupo de três bits, temos (5 1, 0 4), que podemos escrever como (51,4). Na figura 3.4 desenhamos uma estrutura recursiva e sistemática do código convolucional (51,4). Sistemática significa que a mensagem original é uma parte do código. Um código muito utilizado é o (7,2) cuja estrutura é ilustrada na figura 3.5 [2]. Pelo que podemos verificar, este codificador possui 4 estados. Pela construção do codificador convolucional, a partir de um estado, temos somente algumas possibilidades para mudança de estado, o que significa que limitamos a escolha do próximo símbolo na saída. O nome treliça vem da estrutura na qual podemos representar as mudanças de estado baseadas em uma entrada e resultando em uma saída. A treliça é ilustrada na figura 3.6 para o exemplo de codificador (7,2) discutido.

Para exemplificar melhor, consideremos o caso do ASK *Amplitude Shift Keying*. Representamos um vetor de entrada $\mathbf{u} = [u_0 u_1 ... u_{N-1}]^T$, com R bits por símbolo, ou seja, $u_k = (u_k^{R-1}, ..., u_k^1, u_k^0)$, para $k = 0, 1, ..., N-1$. Após a saída do

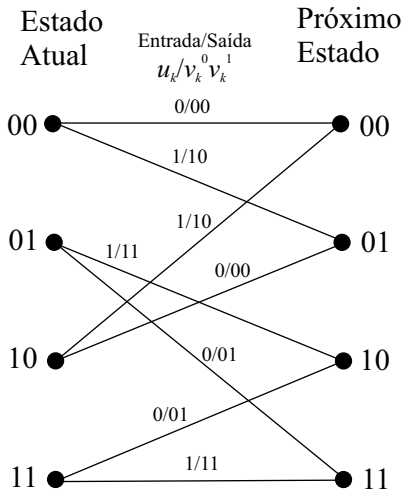


Figura 3.6: Estrutura da treliça para um código convolucional (7,2).

código convolucional, temos uma saída intermediária $\mathbf{v} = [v_0 v_1 ... v_{N-1}]^T$, com $R+1$ bits por símbolo, onde $v_k \in \mathcal{V}$ para $k = 0, 1, ..., N-1$, sendo $\mathcal{V} = \{0, 1, ..., M-1\}$ com $M = 2^{R+1}$. Este vetor intermediário será utilizado pelo mapeador, ilustrado na figura 3.2, para gerar a saída $\mathbf{x} = [x_0 x_1 ... x_{N-1}]^T$, contendo os símbolos de canal, onde $x_k \in \mathcal{R}$ para $k = 0, 1, ..., N-1$, sendo $\mathcal{R} = \{r_0, r_1, ..., r_{M-1}\}$ um conjunto de M números reais. Nesta representação, $\mathcal{R}$ é a constelação de símbolos do modulador resultante. Então a saída de símbolos do canal é sempre um elemento contido em $\mathcal{R}$, $x_k = r_{v_k}$.

A constelação $\mathcal{R}$ é particionada em 4 subconjuntos:

$$\mathcal{D}_i = \{r_{4j+i} : j = 0, 1, ..., M/4 - 1\}, i = 0, 1, 2, 3 \tag{3.3}$$

de modo a maximizar a distância mínima entre os símbolos das sub-constelações. Consideremos a união desses subconjuntos da seguinte forma

$$\mathcal{A}_0 = \mathcal{D}_0 \cup \mathcal{D}_2 \quad \text{e} \quad \mathcal{A}_1 = \mathcal{D}_1 \cup \mathcal{D}_3$$

O grupo formado pelos bits $v_k^1 v_k^0$, dos $R+1$ bits do vetor intermediário $\mathbf{v}$, seleciona a sub-constelação $\mathcal{D}_i$, e o grupo de bits restante $v_k^R v_k^{R-1} ... v_k^2$ seleciona o símbolo correspondente na sub-constelação. Utilizando o codificador convolucional (7,2) do exemplo anterior, podemos redesenhar a treliça, renomeando a saída da treliça para os devidos subconjuntos (figura 3.7). Percebemos que a partir de um estado, só é permitido escolher estados de uma das uniões de sub-constelações, $\mathcal{A}_0$ ou $\mathcal{A}_1$, o

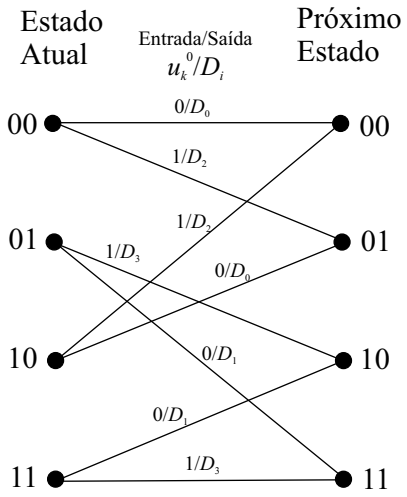


Figura 3.7: Estrutura da treliça de Ungerboeck (TCM) para um código (7,2) , com as transições de estado e a sub-constelação selecionada de acordo com o bit de entrada do codificador convolucional.

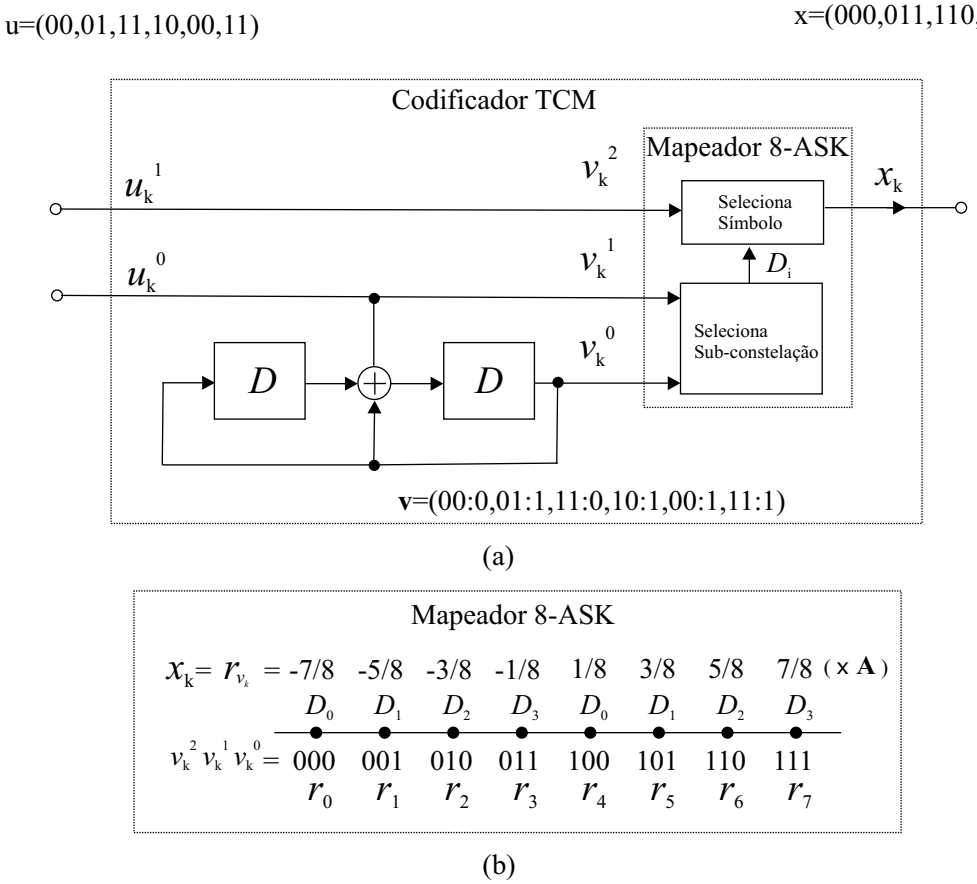


Figura 3.8: Exemplo de uma codificação TCM [2] em (a) e o mapeamento 8-ASK em (b).

que maximiza a distância entre os elementos destes dois grupos. A figura 3.8(a) ilustra o codificador TCM descrito, e mostra o processo de codificação para uma entrada $\mathbf{u}$ =(00,01,11,10,00,11). Depois do codificador convolucional temos o vetor intermediário $\mathbf{v}$ =(00:0,01:1,11:0,10:1,00:1,11:1) que é mapeado no vetor de símbolos $\mathbf{x} = (r_0, r_3, r_6, r_5, r_1, r_7)$, através do mapeamento ASK da figura 3.8(b). A variável A é um coeficiente para ajuste de potência. No caso da modulação ASK, vemos que somente o bit u_k^0 é codificado.

No receptor, pela utilização de codificadores convolucionais, podemos utilizar o algoritmo de Viterbi [14], que procura dentre todas as seqüências possíveis a mais próxima da seqüência recebida. Consideremos um sinal recebido $\mathbf{y} = [y_0 y_1 ... y_{N-1}]^T$ corrompido por um ruído durante a transmissão. Por conhecer a máquina de estados que gerou a seqüência, o algoritmo de Viterbi pode implementar os seguintes passos:

- para cada símbolo recebido, calcula a distância para os símbolos da constelação, estimando uma seqüência $\widehat{\mathbf{x}}$;
- Como ele sabe a treliça que gerou os símbolos, mantém um histórico de cada caminho possível e uma métrica (proporcional à distância Euclidiana) associada a aquele caminho;
- toma a decisão por um caminho (seqüência de símbolos) que resulte em uma menor métrica entre os símbolos recebidos e os símbolos decodificados (decodificador de máxima verossimilhança);
- retorna o vetor $\widehat{\mathbf{u}}$ correspondente.

Em outros termos, o algoritmo de Viterbi encontra a seqüência de símbolos de canal $\widehat{x}$ que minimiza

$$d_E(\mathbf{y}, \widehat{\mathbf{x}}) = \sqrt{\sum_{k=0}^{N-1} (y_k - \widehat{x}_k)^2} \tag{3.4}$$

O decodificador retorna o vetor $\widehat{\mathbf{u}}$ de acordo com o caminho percorrido na treliça para formar a seqüência de símbolos $\widehat{\mathbf{x}}$. O processo de decodificação é ilustrado na figura 3.9.

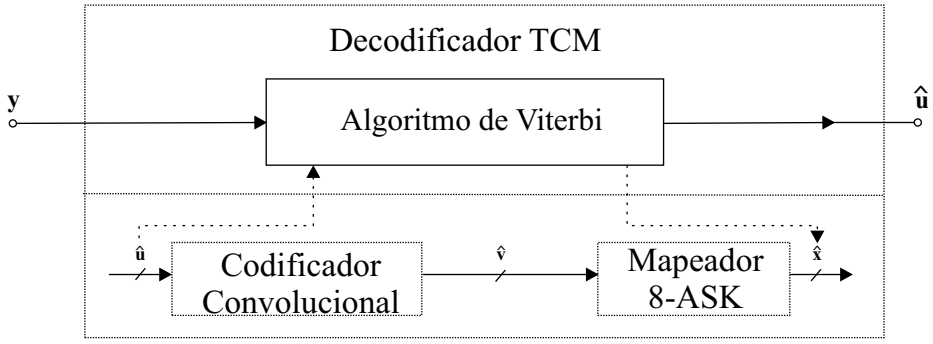


Figura 3.9: O decodificador TCM aplica o algoritmo de Viterbi para encontrar a seqüência de símbolos que está mais próxima, na métrica euclidiana, do sinal recebido corrompido pelo ruído.

3.2 TCQ

Em [3], a dualidade entre a teoria taxa-distorção e a teoria de codificação de canal é interpretada como um problema de preenchimento de cobertura e empacotamento de esferas, respectivamente. As conclusões encontradas das aproximações de cobertura e empacotamento são que, a teoria taxa-distorção procura minimizar a distância máxima entre um conjunto de palavras códigos, e a teoria de codificação de canal procura maximizar a distância mínima de um conjunto de palavras códigos. Marcellin em [6] propõe então um conjunto de códigos que, baseado na modulação de Ungerboeck, resolve de uma maneira mais eficiente o problema da esfera.

Como vimos no capítulo anterior, o algoritmo de Viterbi é utilizado para encontrar uma seqüência de símbolos de canal que mais se aproxime da seqüência transmitida e corrompida por ruído. Os diferentes caminhos na treliça do codificador convolucional definem todas as seqüências possíveis, como palavras códigos na codificação de fonte. Então podemos utilizar o algoritmo de Viterbi como codificador de fonte e os possíveis caminhos a se percorrer na treliça formariam o código de fonte [3]. Implementado como um processo de quantização, dada uma seqüência gerada por uma fonte, o algoritmo de Viterbi encontra uma seqüência de símbolos, que corresponde aos níveis de reconstrução do quantizador, que minimiza a distância (erro médio quadrático) entre a seqüência original e a seqüência quantizada. Para descrever o quantizador codificado por treliças, utilizamos a figura 3.10. O codificador convolucional utilizado neste exemplo é o (7,2) com 4 estados, figura 3.5,

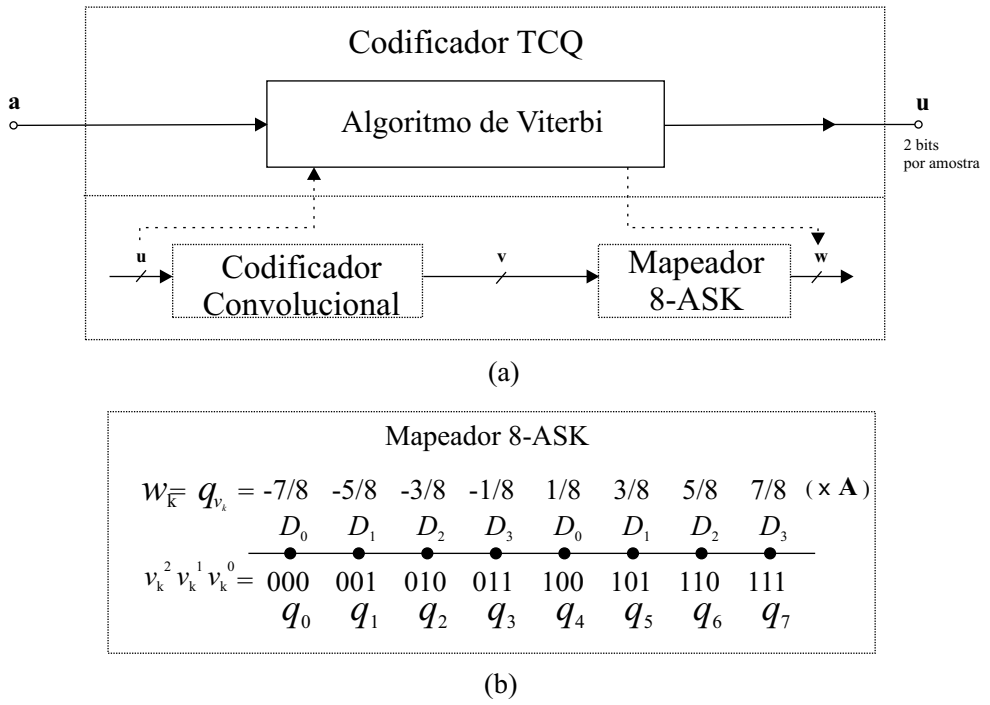


Figura 3.10: Estrutura do quantizador codificado por treliça (a) e o mapeador dos níveis de quantização (b)

com a mesma treliça da figura 3.7. A taxa do quantizador é $R = 2$ bits. Neste caso, o algoritmo de Viterbi tenta encontrar, dentro dos subconjuntos, o valor mais próximo do valor emitido pela fonte, com as restrições de mudança de estado impostas pela treliça do código convolucional. Considerando uma entrada no quantizador $\mathbf{a} = [a_0 a_1 ... a_{N-1}]^T$, onde $a_k \in \mathcal{A}$ com $\mathcal{A} \subset \mathbb{R}$, geramos um vetor de saída $\mathbf{u}$, com 2 bits por símbolo. Na verdade o algoritmo de Viterbi procura um vetor $\mathbf{w} = [w_0 w_1 ... w_{N-1}]^T$ de símbolos de reconstrução, onde $w_k \in \mathcal{Q}$ para $k = 0, 1, ..., N-1$, sendo $\mathcal{Q} = \{q_0, q_1, ..., q_{M-1}\}$ um conjunto de $M = 2^{R+1}$ números reais (níveis de reconstrução), que minimize $d_E(\mathbf{a}, \mathbf{w})$, e retorna o vetor $\mathbf{u}$ que produz a seqüência de símbolos $\mathbf{w}$.

Da mesma forma que acontece na TCM, o conjunto de níveis de reconstrução $\mathcal{Q}$, pode ser particionado em quatro subconjuntos

$$\mathcal{D}_i = \{q_{4j+i} : j = 0, 1, ..., M/4 - 1\}, i = 0, 1, 2, 3 \tag{3.5}$$

mantendo as propriedades de seleção de valores, onde, a cada amostra na entrada do quantizador só é possível escolher valores dentro do conjunto $\mathcal{A}_0 = D_0 \cup D_2$ ou no conjunto $\mathcal{A}_1 = D_1 \cup D_3$.

O processo de decodificação da TCQ é igual ao processo de codificação da TCM. O vetor quantizado **u** serve de entrada para o codificador convolucional que gerou a treliça, gerando o vetor intermediário **v**. Este vetor é mapeado pelo mapeador do quantizador em um vetor **w**, o qual corresponde a um símbolo de reconstrução dado pela relação $w_k = q_{v_k}$.

Não enviamos os mesmos símbolos de reconstrução do codificador para o decodificador TCQ, já que o decodificador possui a mesma máquina de estados que gerou a seqüência codificada. Considerando $\mathcal{B}_k$ o conjunto de entradas no decodificador que são mapeadas para um símbolo de reconstrução q_k, pode-se provar que a escolha de $\widehat{q}_k$ que minimiza $\sum_{u_k \in \mathcal{B}_k}(u_k - \widehat{q}_k)^2$ é o centróide

$$\widehat{q}_k = \frac{\sum_{u_k \in \mathcal{B}_k} u_k}{\|\mathcal{B}_k\|} \tag{3.6}$$

onde $\|\mathcal{B}_k\|$ é o número de elementos do conjunto B_k. Em outra palavras, a decisão por um símbolo de reconstrução é tomada baseada na distância entre a seqüência de entrada no decodificador e as seqüências possíveis da máquina de estado. Depois da decisão, tiramos a média do grupo de seqüências de entrada que foram mapeadas para cada símbolo, calculando assim uma distância média em um hiperespaço para cada grupo, chamado de centróide. Ou seja, fazemos aproximações sucessivas dos centróides a cada entrada no decodificador. Na prática, os centróides são calculados para seqüências de treinamento específicas e armazenados no decodificador. Existem outras formas de otimização dos centróides conforme apresentado por Marcellin e implementado por [2]

3.3 Codificador de imagem com ACTCQ

Depois de estudarmos as técnicas envolvidas em um processo de codificação usando TCQ, apresentamos um codificador de imagem. O ponto de partida para a contribuição do trabalho foi desenvolver o codificador e decodificador ACTCQ-SBC (*Arithmetic Coded Trellis Coded Quantization - SubBand Coding*), e reproduzir os resultados apresentados em [23]. Na figura 3.11 podemos verificar os blocos que compõem o codificador e decodificador de Joshi. Veremos a implementação de cada bloco em seguida.

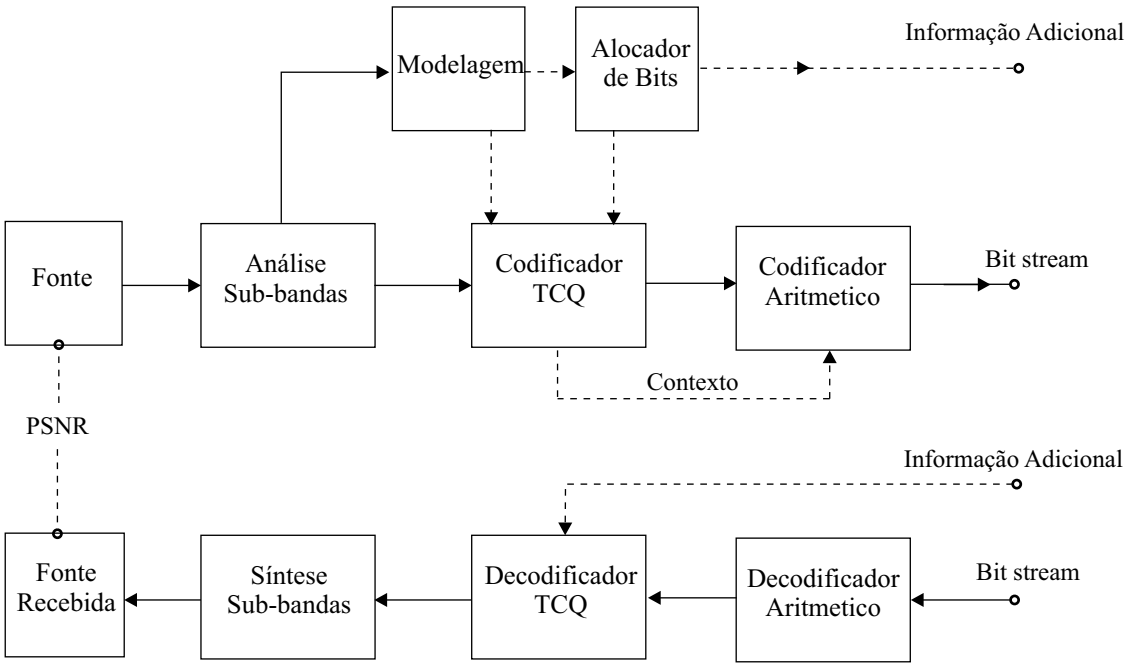


Figura 3.11: Diagrama em blocos do codificador de imagem em sub-bandas baseado em ACTCQ.

No codificador de imagem em sub-bandas, cada sub-banda é quantizada de acordo com sua distribuição de probabilidade. Depois da transformação, as faixas de mais alta freqüência (HFS) apresentam muito baixa correlação inter e intra bandas e podem ser consideradas fontes sem memória [23]. As decomposições utilizadas no codificador estudado foram duas: uma decomposição uniforme em 16 sub-bandas e uma decomposição não uniforme em 22 bandas. No caso da decomposição uniforme, utilizamos o banco de filtros 32D de Johnston, cujos coeficientes podem ser encontrados em [20]. Para a decomposição não uniforme utilizamos os filtros bi-ortogonais *spline* 9-7, cujos coeficientes podem ser encontrados em [1] [24].

Após a decomposição uniforme em sub-bandas, a banda de mais baixa freqüência (LFS) possui propriedades estatísticas semelhantes a da imagem original. Então esta sub-banda não deve ser modelada de forma direta. Foi proposto que a LFS fosse transformada para descorrelacionar os coeficientes e compactar a energia da LFS em poucos coeficientes. As transformadas utilizadas foram a DCT 4x4 e uma transformada wavelet de 2 níveis. As figuras 3.12 e 3.13 ilustram as decomposições a freqüência de uma imagem, seguidas de uma transformada da LFS.

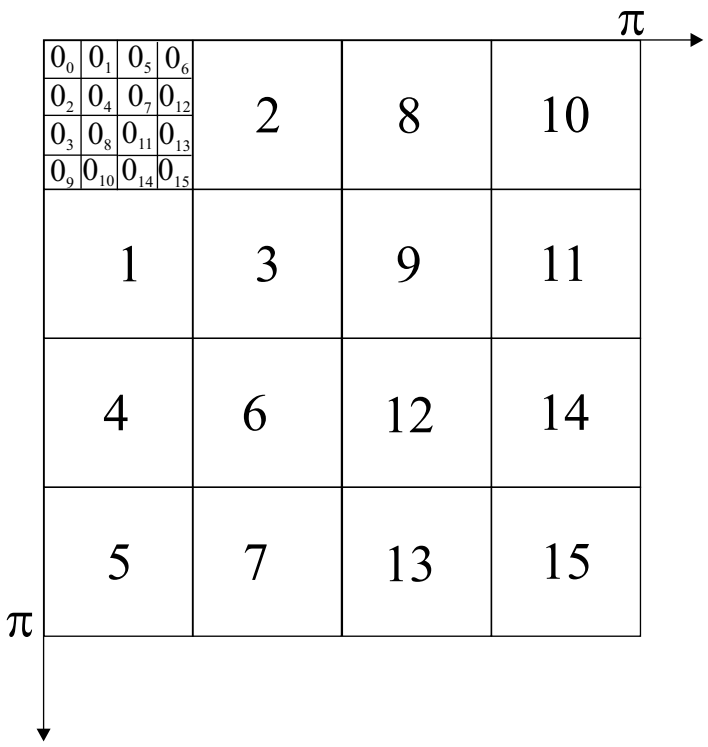


Figura 3.12: Decomposição uniforme em 16 bandas utilizando banco de filtros QMF e a posterior DCT da LFS.

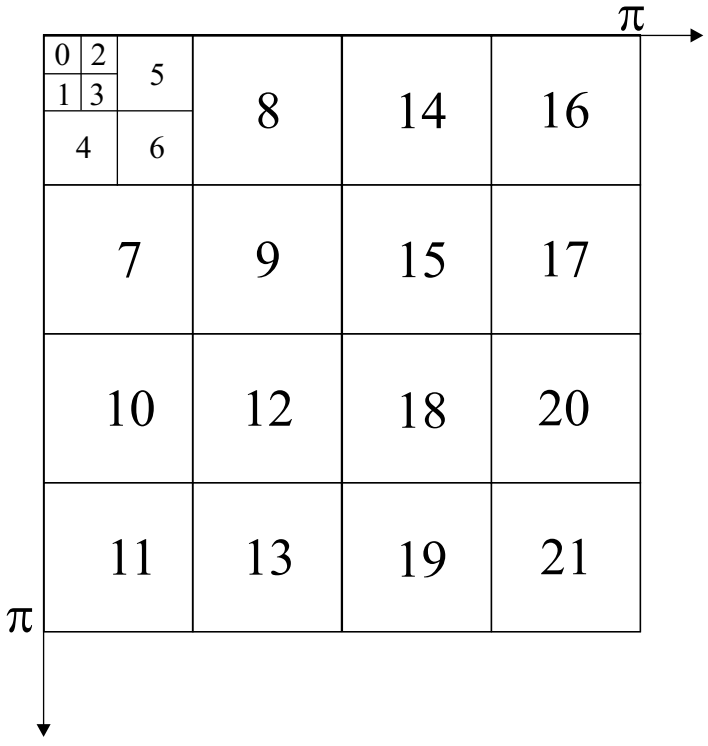


Figura 3.13: Decomposição uniforme em 16 bandas utilizando banco de filtros QMF e a posterior transformada *wavelet* da LFS.

Baseados no histograma das HFS, estudos chegaram a conclusão que podemos modelar a sua distribuição como uma família de gaussianas generalizadas (GGD) [12]. No caso da LFS, quando a DCT é utilizada, pode-se mostrar que o agrupamento dos mesmos índices (componentes com a mesma freqüência) dos diferentes blocos formam fontes com distribuição GGD. Tomando como exemplo uma imagem 512x512, no caso de decomposição uniforme, teremos 15 fontes da HFS contendo 16384 amostras e mais 16 fontes com 1024 amostras provindas do agrupamento dos coeficientes da DCT da LFS. No caso da decomposição da LFS em 7 bandas, quando aplicamos uma transformada *wavelet*, teremos fontes com quantidade de amostras diferentes, como percebemos na figura 3.13.

A função de densidade de probabilidade associada a GGD de uma variável aleatória $\mathcal{X}$ é dada por

$$f_{\mathcal{X}}(x) = \left[\frac{\nu\eta(\nu,\sigma)}{2\Gamma(1/\nu)}\right] e^{-[\eta(\nu,\sigma)|x|]^{\nu}} \tag{3.7}$$

onde

$$\eta(\nu,\sigma) = \sigma^{-1}\left[\frac{\Gamma(3/\nu)}{\Gamma(1/\nu)}\right]^{1/2}. \tag{3.8}$$

Chamamos ν de parâmetro de forma da GGD, pois está diretamente ligado ao decaimento exponencial desta distribuição, e σ é o desvio padrão. A distribuição Laplaciana e Gaussiana são casos particulares de GGD para $\nu = 1.0$ e $\nu = 2.0$, respectivamente. Nos gráfico da figura 3.14 ilustramos alguns exemplos de distribuições de gaussianas generalizadas. Podemos ver que quanto menor o parâmetro ν, mais "pontuda"a curva é em relação à média, onde o decaimento exponencial é maior.

Pelas equações 3.7 e 3.8, podemos ver que a GGD é totalmente representada por ν e σ. Para modelar as sub-bandas, encontramos esses dois parâmetros a partir da estimação da média absoluta da fonte e sua energia (variância), através da equação

$$\widehat{\nu} = F^{-1}\left(\frac{\widehat{E}\|X\|}{\widehat{\sigma}}\right) \tag{3.9}$$

onde

$$F(\alpha) = \frac{\Gamma(2/\alpha)}{\sqrt{(\Gamma(1/\alpha)\Gamma(3/\alpha))}} \tag{3.10}$$

Este modelamento pode ser encontrado em [12]. Neste estudo, também encontrou-se

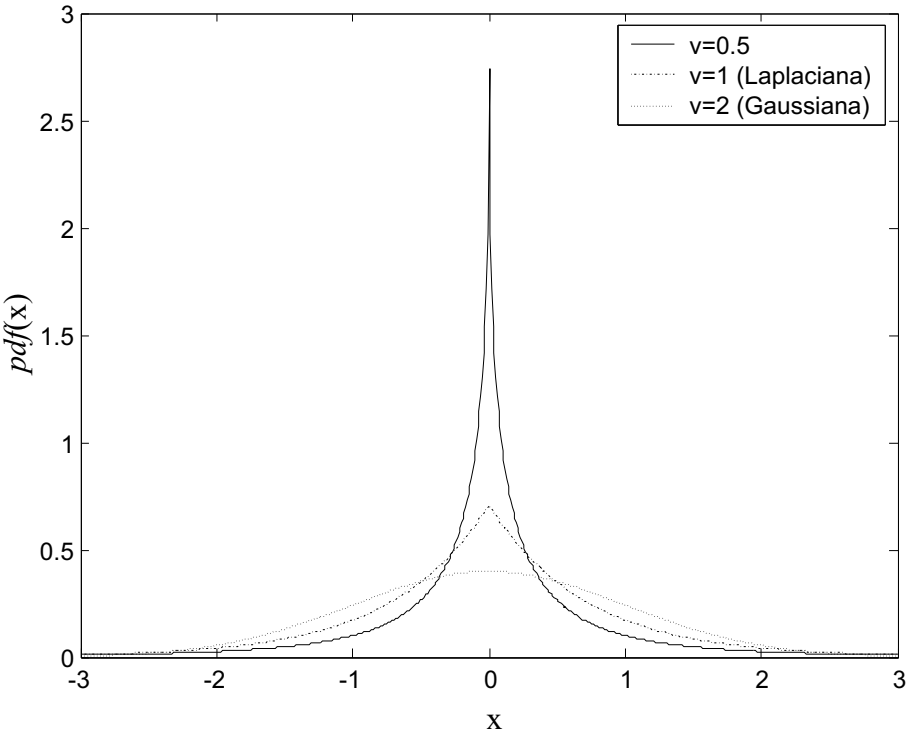


Figura 3.14: Exemplos de curvas com distribuição de gaussianas generalizadas para parâmetros de forma de onda $\nu = 0.5$, $\nu = 1$ (Laplaciana) e $\nu = 2$ (Gaussiana), para $\sigma = 1$ e média 0.

um conjunto de valores ν que conseguem representar as fontes na maioria dos casos, para compressão de imagens naturais. Este conjunto restrito está descrito abaixo

$$\nu \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0\}$$

Tendo esses valores limitados, podemos utilizar tabelas pré-computadas no codificador para escolher a distribuição mais próxima daquela da sub-banda. Existem métodos que estimam o parâmetro ν por verossimilhança, resolvendo as equações 3.9 e 3.10, mas não apresentam ganhos no processo de codificação [23].

Depois que cada sub-banda foi modelada segundo uma distribuição, vamos aplicar a quantização. Antes de quantizar, temos que determinar os parâmetros do quantizador: com quantas palavras códigos trabalharemos; qual o nível de reconstrução que será utilizado; qual a taxa utilizada para codificar cada sub-banda. Com isso ajustamos o quantizador para cada tipo de fonte.

O algoritmo de alocação de bits para cada sub-banda, a partir de uma taxa determinada, determinará a configuração do quantizador para atingir aquela taxa. Baseado em uma taxa total de codificação R bits por amostra, consideramos uma decomposição em M sub-bandas uniformes. Sendo que cada sub-banda i possui N_i amostras. O fator de decimação é dado por [18] [1]

$$f_i = \frac{N_i}{N}, \quad 0 \leq i \leq M-1. \tag{3.11}$$

Para um banco de filtros maximamente decimados, temos que

$$\sum_{i=0}^{M-1} f_i = 1. \tag{3.12}$$

Assumindo que o banco de filtro é ortogonal, pelo princípio de conservação de energia e pela linearidade [18], podemos dizer que

$$\sigma_q^2 = \sum_{i=0}^{M-1} f_i \sigma_{q_i}^2, \tag{3.13}$$

ou seja, a energia do ruído de quantização total é a soma dos ruídos de quantização por coeficiente de cada sub-banda individualmente, ponderadas pelo fator de decimação de cada sub-banda. Com isso, podemos montar o problema de alocação de bits através da minimização do ruído de quantização total [1] [23] [25]

$$\min \sum_{i=0}^{M-1} f_i \sigma_{q_i}^2 \tag{3.14}$$

com a restrição

$$\sum_{i=0}^{M-1} f_i R_i = R, \tag{3.15}$$

onde R_i é a taxa de codificação para cada sub-banda i. Uma forma de minimizar esta função é utilizar somente um conjunto de valores discretos para possíveis taxas de codificação. Então, podemos reescrever o problema de alocação de bits em função da distorção inserida por cada sub-banda i no sistema dada uma taxa de codificação R_i, conforme

$$\min \sum_{i=0}^{M-1} f_i D_i(R_i) \tag{3.16}$$

com a restrição

$$\textstyle\sum_{i=0}^{M-1} f_i R_i \leq R, \quad R_i \in B_i, \quad 0 \leq i \leq M-1, \tag{3.17}$$

onde M é o número total de sub-bandas e B_i é o conjunto de taxas permitidas para codificar as sub-bandas. Com isso montamos as curvas operacionais de taxa-distorção do quantizador com os valores $\{(D(R_i), R_i)\}$. Essas curvas podem ser montadas a partir de seqüências de treinamento ou pela quantização da sub-banda com as diferentes taxas. Um algoritmo rápido utilizando multiplicadores de Lagrange pode ser utilizado na minimização da equação 3.16 para encontrar o ponto ótimo de cada curva operacional [1] [25], quando quantizadas em conjunto. Este ponto indica que a quantidade de bits para alocar entre as sub-bandas acabou, conforme a figura 3.15. Reescrevendo, temos

$$\min \sum_{i=0}^{M-1} f_i (D_i(R_i) + \lambda R_i) \tag{3.18}$$

para $\lambda \geq 0$, com as mesmas restrições da equação 3.17.

A equação de minimização pode ser encontrada separadamente para cada sub-banda, pela presença do somatório na equação. Descrevemos agora a implementação do algoritmo de Westerink [25] para resolver este problema:

1. Alocar a menor taxa possível para cada sub-banda. Considere k_i, a taxa alocada para cada sub-banda i, $0 \leq i \leq M-1$;

2. Para cada sub-banda i, calcular a inclinação da derivada aproximada para o conjunto de pontos da curva taxa-distorção operacional

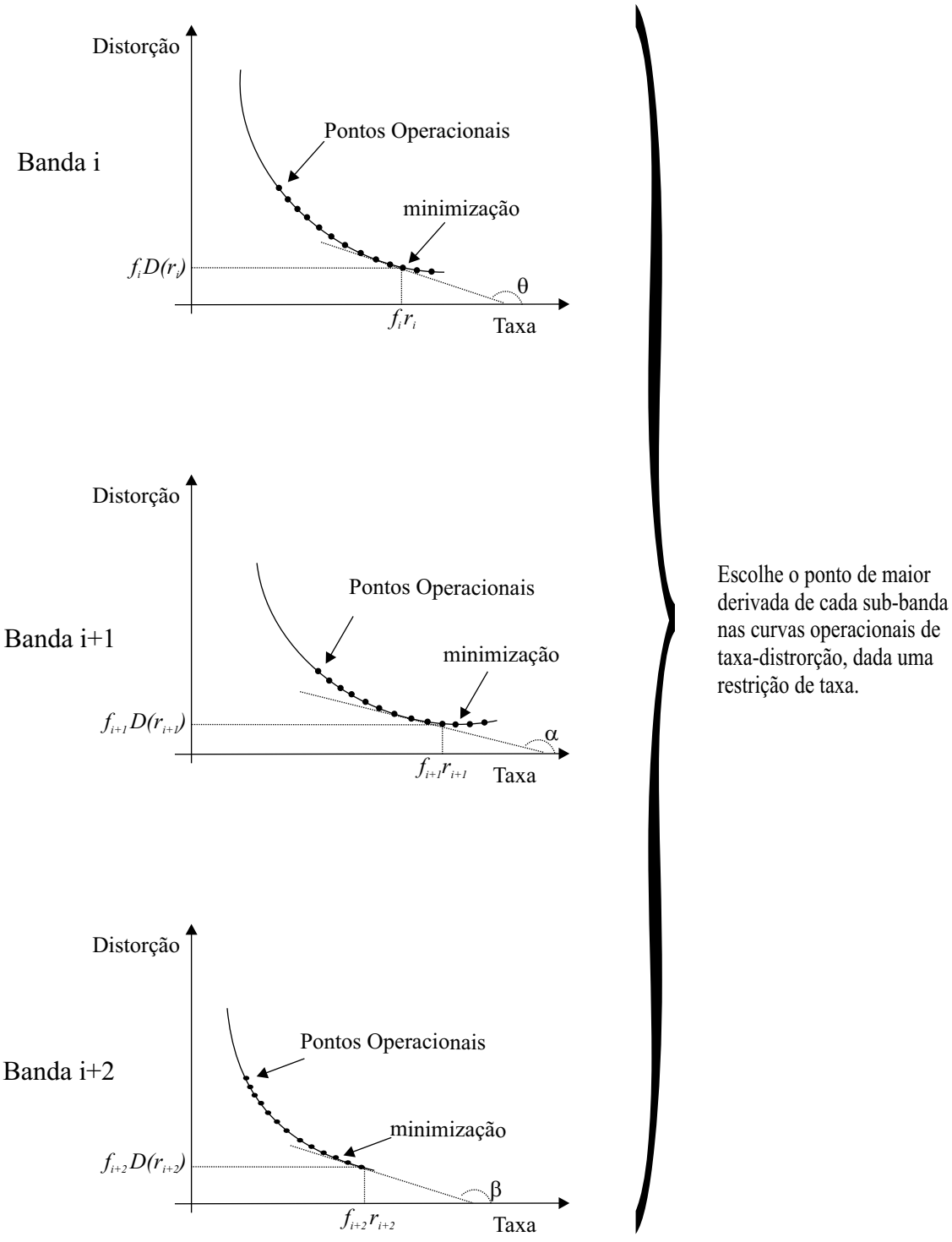


Figura 3.15: A minimização do problema taxa-distorção para alocação de bits das sub-bandas consiste em achar o ponto mais próximo nas curvas operacionais de cada sub-banda, que represente uma menor distorção total para uma determinada taxa.

$$s_i = \max \frac{r_i - k_i}{D(r_i) - D(k_i)}$$

onde o máximo é calculado para $\{r_i \in B_i, r_i > k_i\}$;

3. Achar o máximo s_i para $0 \leq i \leq M-1$. Caso a sub-banda j tenha sido a vencedora da busca, atualizar o valor k_j para valor de taxa onde o máximo foi obtido;

4. Calcular a distorção total com os novos valores de taxa. Caso a taxa esteja próxima o suficiente da taxa total desejada, parar o algoritmo. Caso contrário, continuar;

5. Repetir passos 2, 3 e 4 mas faça o passo 2 somente para as sub-bandas onde foi associada uma nova taxa.

Na implementação, geramos as curvas operacionais taxa-distorção para os valores permitidos ν que consideramos representar a maioria das fontes que serão quantizadas. A curva operacional foi feita para 100 pontos com taxas variando de 0 a 9 bits. Os pontos são separados quase uniformemente no eixo da taxa. Estas curvas são armazenadas no quantizador. Depois que as sub-bandas foram modeladas, o sistema aplica o algoritmo descrito de alocação de banda. Perceba que o passo 2 do algoritmo é calculado para diferentes curvas operacionais, pois cada sub-banda possui um modelo.

O quantizador utilizado neste codificador é o TCQ com algumas modificações. As sub-bandas possuem muitos coeficientes de baixa magnitude, o que pode ser verificado nos seus histogramas. Quando trabalhamos com taxas muito baixas, a probabilidade do nível de reconstrução "0"aumenta. Se olharmos novamente para a figura 3.7, percebemos que se o sistema estiver no estado (11) não seria possível retornar o valor "0". Com isso, deterioramos o desempenho da distorção, pois o nível de reconstrução "0"não é uma saída possível em toda transição da treliça que compõe a TCQ.

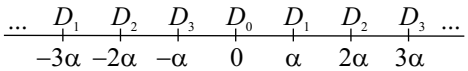


Figura 3.16: Níveis de reconstrução do TCQ.

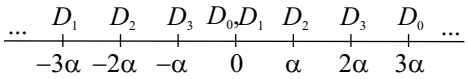


Figura 3.17: Níveis de reconstrução modificados do TCQ para melhorar desempenho em baixas taxas.

Podemos verificar a disposição dos níveis de reconstrução para este exemplo, figura 3.16. Este problema é facilmente contornado através de um deslocamento para a esquerda de todos os níveis positivos, conforme 3.17 [11]. A utilização do dicionário modificado não é fixa. A partir do parâmetro de gaussiana generalizada e da taxa, existe um modo ótimo de operação de dicionário para a TCQ, ou seja, com ou sem a sobreposição do nível "0". Antes de determinar qual o modo de operação de dicionário, calculamos a distorção que cada um insere na quantização. A partir deste resultado, escolhemos o que gera uma menor distorção. Essa escolha é feita através de uma tabela pré-computada, como veremos a seguir.

Com a taxa determinada, temos que parametrizar o quantizador de cada sub-banda. Na verdade, geramos uma tabela de quantizadores diferentes, variando todas as possibilidades de cada parâmetro. Cada linha desta tabela terá uma configuração associada. Esta tabela tem como parâmetros:

- a taxa (número de bits alocado para a sub-banda)
- o modelo de distribuição de probabilidade da entrada
- o passo de quantização α
- o número de níveis de reconstrução
- o modo de operação do dicionário (sem ou com sobreposição do nível de reconstrução "0"
- a distorção associada à configuração

Com as combinações desses parâmetros, temos todas as configurações possíveis, gerando os pontos da curva taxa-distorção. Deste maneira, quando o algoritmo de alocação de bit determina a taxa de quantização de cada sub-banda, é somente uma questão de "olhar"a tabela e configurar o quantizador de acordo com a linha

escolhida. No caso da TCQ, as distorções calculadas e armazenadas na tabela são geradas a partir de quantizador HOVA *(Hard Output Viterbi Algorithm)* [11].

Durante o processo de quantização, à medida em que os símbolos são gerados, aplicamos uma codificação de entropia para tornar o tamanho dos símbolos variável. A codificação aritmética baseada em contexto é utilizada, por isso o nome ACTCQ (*Arithmetic Coded Trelis Coded Quantization*). Como vimos anteriormente, a partir do estado da treliça somente podemos escolher entre dois grupos: se estamos em um estado par, somente podemos escolher entre os símbolos pertencentes a $\mathcal{A}_0$; se estamos em um estado ímpar, somente símbolos de $\mathcal{A}_1$ podem ser escolhidos. Utilizamos, então, a posição nos estados da treliça como contexto, gerando então duas tabelas de probabilidade: uma para o conjunto de símbolos de $\mathcal{A}_0$ e outra para $\mathcal{A}_1$. Deste modo, quando um símbolo for codificado, o sistema identifica o sub-grupo e aplica as sub-divisões de acordo com a densidade de probabilidade do grupo. Os modelos de probabilidade não são fixos, ou seja, a cada nova entrada, atualizamos a freqüência de cada símbolo, para refinar as sub-divisões dos intervalos e poder se adaptar melhor a fontes não estacionárias.

No receptor, já estão pré-computados os centróides para cada tipo de fonte de entrada no de-quantizador. Lembrar que a fonte é modelada pelos conjuntos de possíveis parâmetros de GGD listados anteriormente. O processo de decodificação é realizado. Para medirmos o desempenho, utilizamos o PSNR em dB, que para imagens monocromáticas é definido como [17]

$$PSNR(dB) = 10 \log \left(\frac{255^2}{MSE} \right) \text{dB} \tag{3.19}$$

Na codificação da luminância da imagem Lena, Joshi obteve os resultados da tabela 3.1. Os resultados para a imagem Barbara estão na tabela 3.2. Analisando as tabelas, podemos verificar que não houve diferença significativa em valores de relação sinal-ruído, para as diferentes tipos de decomposição (22 bandas não uniforme ou 16 bandas uniforme com a DCT 4x4 da LFS). As diferenças são maiores na visualização das imagens como veremos mais adiante.

No trabalho de Joshi, é apresentada uma comparação com outros métodos de quantização utilizados em codificadores baseados em decomposição em sub-bandas da imagem. Em relação ao UTQ-SBC (*Uniform Threshold Quantization*), a implementação do ACTCQ-SBC melhora o desempenho em aproximadamente 0.6 e 0.8

Tabela 3.1: Resultados da codificação da imagem Lena 512x512 utilizando o ACTCQ-SBC.

22 bandas 9-7 bi-ort		16 bandas 32D QMF	
Taxa (bpp)	PSNR (dB)	Taxa (bpp)	PSNR (dB)
0.238	33.03	0.245	33.01
0.351	34.65	0.359	34.76
0.467	35.94	0.475	36.12
0.583	36.93	0.588	37.15

Tabela 3.2: Resultados da codificação da imagem Barbara 512x512 utilizando o ACTCQ-SBC.

22 bandas 9-7 bi-ort		16 bandas 32D QMF	
Taxa (bpp)	PSNR (dB)	Taxa (bpp)	PSNR (dB)
0.248	27.07	0.239	27.71
0.359	28.71	0.345	29.46
0.462	29.99	0.449	30.91
0.568	31.30	0.553	32.21

Tabela 3.3: Resultados da codificação da imagem Lena 512x512 utilizando o ACTCQ-SBC comparados com o EZW.

EZW 13 bandas 9-7 bi-ort		ACTCQ-SBC 22 bandas 9-7 bi-ort		ACTCQ-SBC 16 bandas 32D QMF	
Taxa (bpp)	PSNR (dB)	Taxa (bpp)	PSNR (dB)	Taxa (bpp)	PSNR (dB)
0.250	33.18	0.238	33.03	0.245	33.01
0.375	34.65	0.351	34.65	0.359	34.76
0.500	36.50	0.467	35.94	0.475	36.12
0.625	37.08	0.583	36.93	0.588	37.15
0.750	37.64	0.695	37.71	0.697	37.99
0.875	38.58	0.812	38.44	0.817	38.79
1.000	39.56	0.931	39.15	0.935	39.50

dB [23]. O aumento do desempenho é creditado ao ganho granular na quantização de cada sub-banda, comparado ao UTQ-SBC. O custo deste ganho é a complexidade computacional e espaço ocupado na memória. Em relação ao quantizador ECTCQ (*Entropy Constrained Trelis Coded Quantization*) com dicionários não-uniformes e otimizados, o ACTCQ-SBC está entre 0.4 e 0.6 dB pior [23]. É visto que as menores diferenças estão para altas taxas de codificação. O ACTCQ-SBC foi comparado também com técnicas que exploram a dependência de energia inter sub-bandas e intra sub-bandas, como o EZW [26] e obteve resultados de PSNR superiores para uma mesma taxa. Pelo resultado da tabela 3.3, para a codificação da Lena 512x512, podemos ver que o ganho significativo do ACTCQ-SBC em relação ao EZW acontece para taxas mais altas, acima de 0.5 bpp. Por exemplo, a imagem codificada pelo EZW a 0.675 bpp possui uma PSNR de 37.08 dB, sendo que a mesma imagem codificada pelo ACTCQ-SBC, 16 bandas, a 0.588 bpp, já possui um PSNR com 1.07 dB de ganho. A perda de ganho do ACTCQ-SBC 22 bandas para a decomposição em 16 bandas, em baixas taxas, é porque a quantidade de bits alocadas para a banda de alta freqüência é menor em relação às outras técnicas e o filtro nesta faixa de freqüência é muito aberto, aumentando a distorção. Na figura 3.18 temos os

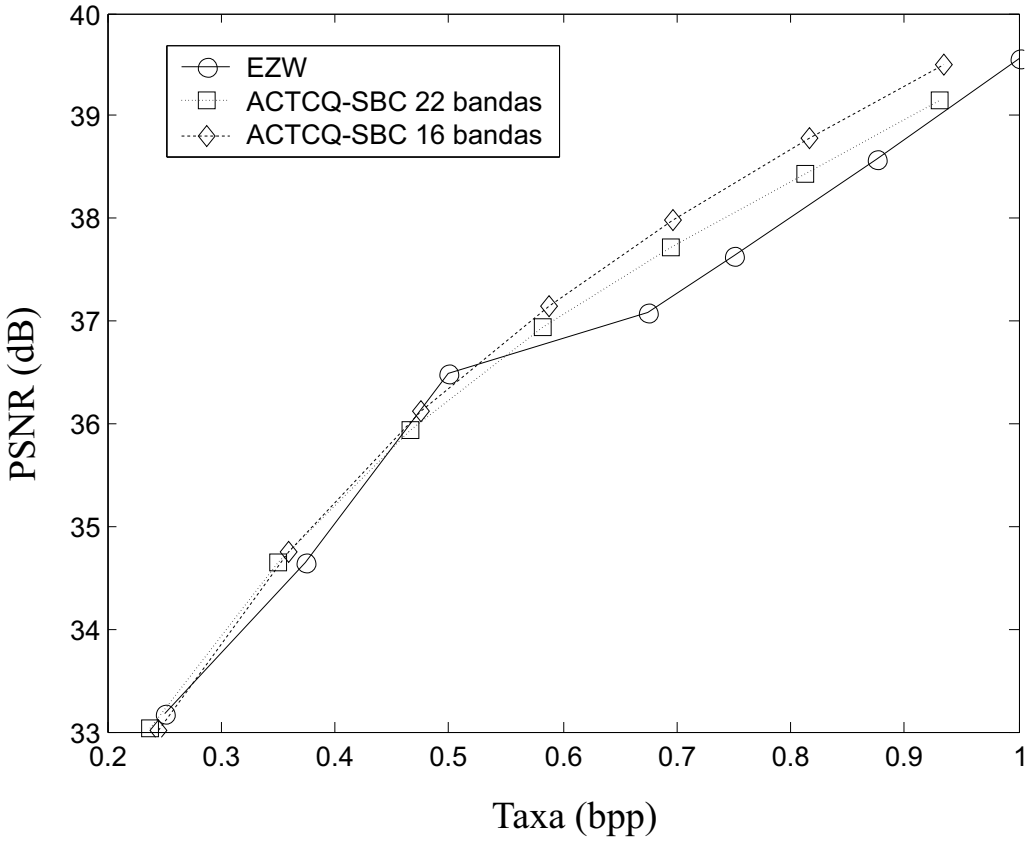


Figura 3.18: Comparação entre as técnicas de compressão EZW, ACTCQ-SBC com decomposição 16 bandas uniforme e ACTCQ-SBC com decomposição 22 bandas não uniformes, na codificação da imagem Lena 512x512.

resultados da tabela 3.3, para melhorar a visualização.

Uma das propostas de trabalhos futuros, implementada por Joshi durante o desenvolvimento do ACTCQ-SBC, é a consideração de sub-bandas como modelos estatísticos não estacionários. Com isso, ao invés de modelarmos a sub-banda como uma seqüência estacionária, codificaríamos ela em blocos. Como muitos desses blocos podem ser modelados pela mesma distribuição de probabilidade, agrupamos eles em classes. Cada classe terá uma quantização diferente. Todo o desenvolvimento desta técnica está descrito em [23] [27]. O ganho de codificação por classes é dependente da imagem a ser codificada. Para Lena, o ganho ficou entre 1dB, enquanto para Bárbara ficou em 1.3 a 1.9dB [23].

Vamos agora analisar os efeitos visuais dos quantizadores. Nas figuras 3.19 e 3.20, determinamos a taxa de 0.250 bpp para podemos analisar os tipos de artefatos inseridos na imagem, depois do processo de codificação e decodificação. Comparando com a imagem original, vemos que o EZW perde muita informação de alta freqüência, devido à árvore de zeros [1]. Então o rosto da Barbara fica muito fora de foco, assim como o fundo (chão). O mesmo acontece na perna dela, onde os detalhes da calça são perdidos. Subjetivamente, a utilização da decomposição em 22 sub-bandas, figura 3.19(c), gera uma imagem melhor que (d). Atribuímos esta diferença, ao fato da decomposição em 16 bandas, a baixa taxas, gerar uma granularidade das partes mais lisas da imagem, mesmo apresentando um PSNR maior. Conforme aumentamos a taxa, a solução de decomposição em 16 sub-bandas se iguala subjetivamente ao outro tipo de decomposição utilizado (22 bandas).

Para a imagem da codificação da Lena em sub-bandas, figura 3.20 temos os mesmos resultados. Para o EZW 3.20(b) perdemos muita informação de alta freqüência no chapéu, e damos um aspecto de "imagem lavada"no rosto e nos ombros dela. Já na configuração 3.20(d), temos uma imagem granulada em toda sua dimensão. Este ruído é uma característica da implementação da DCT. Existe também na figura 3.20(c) um ruído na borda do chapéu da Lena, mas em uma escala bem menor. Normalmente, quando passamos a utilizar a codificação em sub-banda, trocamos o efeito de blocos pelo efeito de *ringing*, que pode ser observado nas bordas nas imagens Lena codificadas.

(a) (b)

(c) (d)

Figura 3.19: Comparação dos codificadores para a imagem Bárbara 512x512 a 0.250bpp (a) original (b) EZW (c) ACTCQ-SBC 22 bandas (d) ACTCQ-SBC 16 bandas

Figura 3.20: Comparação dos codificadores para a imagem Lena 512x512 a 0.250bpp (a) original (b) EZW (c) ACTCQ-SBC 22 bandas (d) ACTCQ-SBC 16 bandas

Capítulo 4

Turbo Quantização codificada por Treliças

Em um sistema de comunicação, cada modelo de canal de transmissão, que insere ruídos na informação transmitida, possui uma capacidade, ou seja, uma taxa máxima de transmissão de informação de modo que consigamos receber o sinal transmitido com uma certa probabilidade de erros. Os limites teóricos para esta taxa máxima de transmissão de informação em um canal ruidoso foi estabelecido no Teorema de Capacidade de Canal [15]. Este teorema assegura que existem códigos capazes de atingir a capacidade do canal para blocos de código suficientemente grandes e escolhidos de forma aleatória. O teorema por sua vez não indica como construir tais códigos.

A procura por estes códigos sempre foi voltada para termos práticos e de implementação. Os códigos estruturados implementam de forma simples a codificação e decodificação [14], mas não apresentam curvas assintóticas de probabilidade de erro próximas da capacidade do canal. Os códigos turbo, apresentados por [4], se aproximam da capacidade gerando palavras códigos estruturadas e pseudo-aleatórias com um decodificador realizável, como veremos em seguida.

4.1 Codificação Turbo

O turbo codificador consiste na concatenação de dois codificadores convolucionais em paralelo. É um código sistemático, ou seja, a saída possui uma cópia da

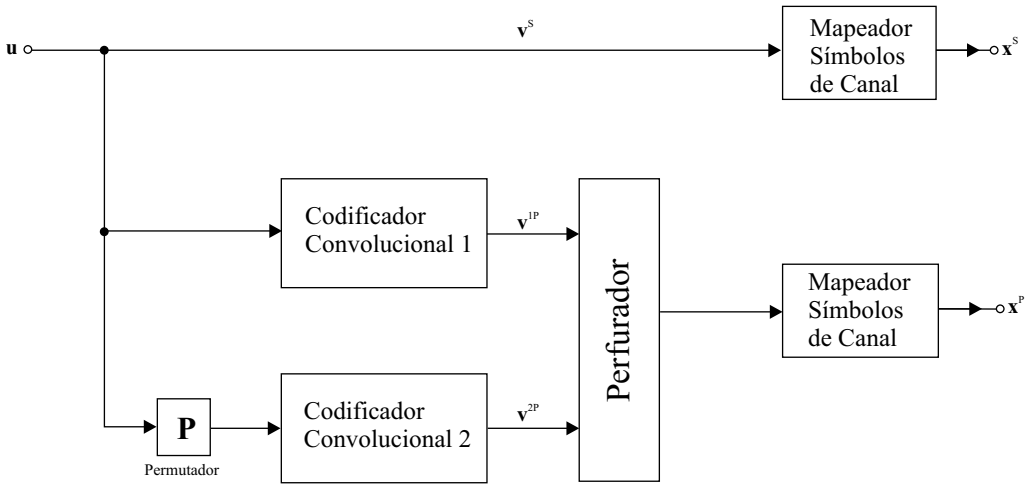


Figura 4.1: Estrutura de um codificador turbo [2].

entrada adicionada dos bits de paridade, que são produzidos por dois códificadores convolucionais recursivos e um permutador (*interleaver*). Pode-se utilizar perfuradores para conseguir taxas diferente da taxa básica 1/3. Pela figura 4.1 podemos verificar que um bit da entrada u_{k-1} produz três bits de saída intermediária composta de um bit sistemático e dois de paridade, $v_{k-1}^{S} v_{k-1}^{1P} v_{k-1}^{2P}$, que serão mapeados em três símbolos de canal $x_{k-1}^{S} x_{k-1}^{1P} x_{k-1}^{2P}$, no caso da taxa 1/3, com $k = 1, 2, ..., N$, onde N é a dimensão do vetor de entrada $\mathbf{u} = [u_0 u_1 ... u_{N-1}]^T$. O turbo quantizador, apesar de utilizar a estrutura de códigos convolucionais, pode ser considerado um código de bloco, onde N bits da informação são processados por iteração. Esta condição é imposta pelo permutador. O permutador **P** recebe os N bits de forma seqüencial e permuta, segundo um mapa de permutação, as posições de cada bit dentro do bloco antes de apresentar para o codificador convolucional 2. Com isso a saída deste codificador é diferente do primeiro codificador convolucional. Normalmente utilizamos N menor que 1000. Além da escolha dos códigos convolucionais, o tamanho do bloco e a escolha do permutador, influenciam na eficiência dos códigos turbo. Para facilidade de implementação escolhemos os dois codificadores convolucionais iguais, mas nada impede que eles sejam diferentes [2].

Utilizamos o perfurador para conseguirmos taxas diferentes como, por exemplo, a taxa 1/2, onde descartamos alternadamente os bits de paridade v_{k-1}^{1P} e v_{k-1}^{2P}.

4.2 Decodificação Turbo

Antes de falar do turbo decodificador, apresentamos de forma resumida um algoritmo de decodificação para códigos concatenados, operando de forma conjunta e iterativa, utilizando a mesma aproximação realizada em [2]. O BCJR-MAP, nomeado após Bahl, Cocke, Jalinek e Raviv [28], foi utilizado por Berrou na implementação do decodificador turbo [4].

Consideremos um sistema de comunicação onde um vetor $\mathbf{x} = [x_0 x_1 ... x_{N-1}]^T$ é transmitido e corrompido por ruído aditivo, resultando no vetor $\mathbf{y} = [y_0 y_1 ... y_{N-1}]^T$, que será decodificado para recuperar $\mathbf{u}$. Então, o princípio do algoritmo de decodificação BCJR-MAP é decidir, símbolo a símbolo, que o bit de mensagem $u_{k-1} = 1$ (considerando fonte binária) foi transmitido se

$$\Pr\{U_{k-1} = 1|\mathbf{y}\} > \Pr\{U_{k-1} = 0|\mathbf{y}\} \tag{4.1}$$

caso contrário, decide que $u_{k-1} = 0$ foi transmitido. Uma função que facilita o mapeamento dessa decisão é o logaritmo da razão de probabilidades *a posteriori*

$$L_{k-1}(1) = \ln\left[\frac{\Pr\{U_{k-1} = 1|\mathbf{y}\}}{\Pr\{U_{k-1} = 0|\mathbf{y}\}}\right]. \tag{4.2}$$

Com isso, temos que se a probabilidade condicional do bit $u_{k-1} = 1$ ter sido transmitido, dada a recepção de $\mathbf{y}$, for maior que a probabilidade condicional do bit $u_{k-1} = 0$ ter sido transmitido, dada a recepção de $\mathbf{y}$, $L_{k-1}(1)$ será maior que 0. Logo,

$$u_{k-1} = \begin{cases} 1, & L_{k-1}(1) > 0 \\ 0, & L_{k-1}(1) < 0 \end{cases} \tag{4.3}$$

para $k = 1, 2, ..., N$. Por utilizarmos códigos convolucionais, como vimos no capítulo anterior, a escolha do bit de entrada u_{k-1} determina um conjunto de transições de estado na treliça (s', s), sendo que $s', s \in \{0, 1, ..., N_e\}$ onde N_e é o número de estados da treliça. Então, uma forma equivalente de escrever a equação 4.2 é

$$L_{k-1}(1) = \ln\left[\frac{\sum_{(s',s)\in\mathcal{B}^1_{k-1}} \Pr\{S_{k-1} = s', S_k = s|\mathbf{y}\}}{\sum_{(s',s)\in\mathcal{B}^0_{k-1}} \Pr\{S_{k-1} = s', S_k = s|\mathbf{y}\}}\right]. \tag{4.4}$$

onde o conjunto $\mathcal{B}^1_{k-1}$ contém todas as transições de estado (s', s), correspondente a entrada $u_{k-1} = 1$, e $\mathcal{B}^0_{k-1}$ contém todas as transições de estado (s', s), correspondente a entrada $u_{k-1} = 0$. Considerando que os códigos convolucionais geram seqüências

de Markov, que o ruído de transmissão é aditivo e gaussiano, e que a modulação BPSK é utilizada, onde o bit 0 é mapeado no símbolo de canal +1 e o bit 1 é mapeado no símbolo de canal -1, pode-se manipular a equação 4.4 para

$$L_{k-1}(1) = L^a_{k-1} + \frac{2}{\sigma^2} y_{k-1} + \ln \left[\frac{\sum_{(s',s)\in\mathcal{B}^1_{k-1}} \alpha_{k-1}(s')\gamma^e_k(s',s)\beta_k(s)}{\sum_{(s',s)\in\mathcal{B}^0_{k-1}} \alpha_{k-1}(s')\gamma^e_k(s',s)\beta_k(s)} \right]. \tag{4.5}$$

onde o valor $2/\sigma^2$ (σ^2 é a energia do ruído branco gaussiano de média zero) é chamado de valor de canal, L_c. Em [2] encontramos o desenvolvimento para achar os valores dos coeficientes $\alpha_{k-1}(s')$, $\gamma^e_k(s',s)$ e $\beta_k(s)$. Pelas equações acima, percebemos que o algoritmo BCJR-MAP calcula o conjunto de coeficientes α_{k-1}, β_{k-1} e γ^e_{k-1} para cada símbolo recebido y_{k-1}, que entra na treliça do decodificador. Estes coeficientes são encontrados de forma recursiva através das equações abaixo:

$$\alpha_k(s) = \frac{\sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s)}{\sum_s \sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s)} \tag{4.6}$$

$$\beta_{k-1}(s) = \frac{\sum_s \gamma_k(s',s)\beta_k(s)}{\sum_s \sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s)} \tag{4.7}$$

$$\gamma^e_k(s',s) = e^{\frac{y^P_{k-1}x^P_{k-1}(s',s)}{\sigma^2}} \tag{4.8}$$

Em termos de implementação, consideramos que o estado inicial é 0, então $\alpha_0(s) = 0$ para $s \neq 0$. Se ao final da codificação de cada seqüência, inserirmos os *tail bits* [14], de modo que o codificador é forçado a retornar ao estado 0, teremos $\beta_N(s) = 1$ para $s = 0$ e $\beta_N(s) = 0$ para $s \neq 0$. Se não inserirmos os *tail bits*, teremos $\beta_N(s) = 1/N_e$, $\forall s \in 0, 1, ..., N_e - 1$, e N_e é o número de estados da treliça. [2]

Pela equação 4.5, chamamos de informação extrínseca o terceiro ramo, que representa toda informação nova a respeito da seqüência de símbolos transmitida $\mathbf{x}$ a partir da decodificação da seqüência recebida $\mathbf{y}$. Em representação vetorial, considerando $\mathbf{L} = [L_0 L_1 ... L_{N-1}]^T$, $\mathbf{L}^a = [L^a_0 L^a_1 ... L^a_{N-1}]^T$ e $\mathbf{L}^e = [L^e_0 L^e_1 ... L^e_{N-1}]^T$ temos

$$\mathbf{L} = \mathbf{L}^a + L_c \mathbf{y}^S + \mathbf{L}^e \tag{4.9}$$

O turbo decodificador utilizará dois algoritmos BCJR-MAP para decodificar o símbolo recebido conforme ilustrado na figura 4.2. A informação sistemática $\mathbf{y}^S$, e de paridade $\mathbf{y}^P$ geram estimativas da informação transmitida através de $\mathbf{L}_1$ e $\mathbf{P}^{-1}\mathbf{L}_2$. A partir dessas informações geramos a informação extrínseca, $\mathbf{L}^e_{12}$ e $\mathbf{L}^e_{21}$, que por sua

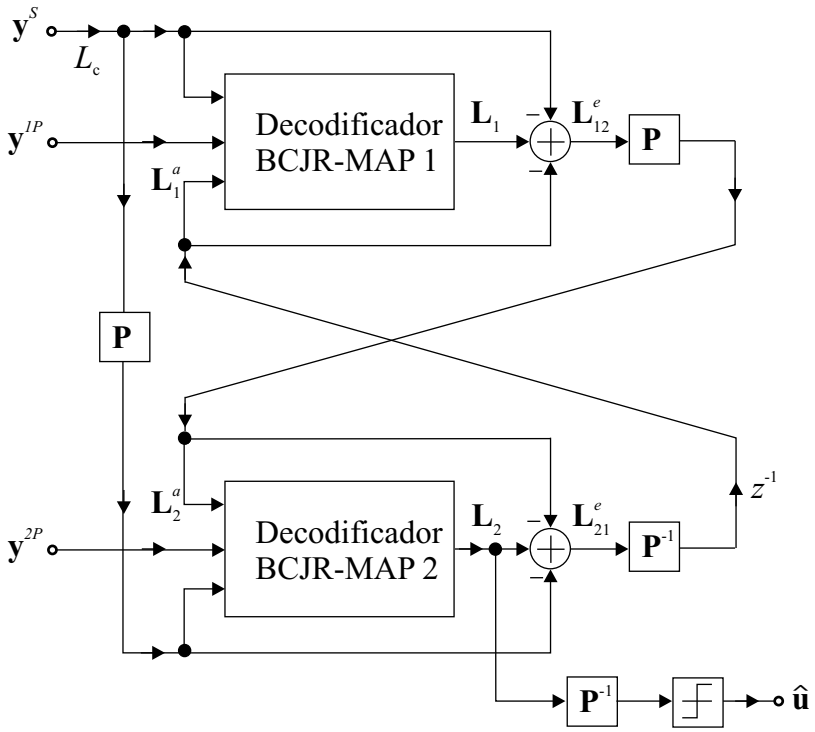


Figura 4.2: Esquema do turbo decodificador [2].

vez é realimentada nos decodificadores como informação *a priori*, $\mathbf{L}_1^a = \mathbf{P}^{-1}\mathbf{L}_{21}^e$ e $\mathbf{L}_2^a = \mathbf{P}\mathbf{L}_{12}^e$. Com essa realimentação, produzimos novas e melhores estimativas. Este processo descrito é feito de forma iterativa, onde o número de iterações N_{it}, varia normalmente entre 10 e 20 [2]. Representando o esquema da figura 4.2 em termos matemáticos, para cada iteração $i = 1, 2, ...N_{it}$

$$\mathbf{L}_{12}^{e(i)} = \mathbf{L}_1^{(i)} - \mathbf{P}^{-1}\mathbf{L}_{21}^{e(i-1)} - L_c\mathbf{y}^S \tag{4.10}$$

$$\mathbf{L}_{21}^{e(i)} = \mathbf{L}_2^{(i)} - \mathbf{P}\mathbf{L}_{12}^{e(i)} - L_c\mathbf{P}\mathbf{y}^S \tag{4.11}$$

sendo que $\mathbf{L}_{21}^{e(0)} = \mathbf{0}$ (estado inicial do sistema). Substituindo a equação 4.10 em 4.11 temos que

$$\mathbf{L}_{21}^{e(i)} = \Delta\mathbf{L}_{21}^{e(i)} + \mathbf{L}_{21}^{e(i-1)} \tag{4.12}$$

onde $\Delta\mathbf{L}_{21}^{e(i)} = \mathbf{L}_2^{e(i)} - \mathbf{P}\mathbf{L}_1^{e(i)}$. Essas equações mostram que a informação extrínseca gerada pelo decodificador 2 é a diferença entre as estimativas do decodificador 2 e do decodificador 1 na mesma iteração, somada à informação extrínseca da iteração anterior gerada pelo decodificador 2. Deste modo, quando a diferença $\Delta\mathbf{L}_{21}^{e(i)}$ for zero, temos a convergência do algoritmo na i-ésima iteração.

O mapeamento da informação transmitida estimada $\widehat{u}_{k-1}$ é realizada através da função $f(x)$

$$f(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases} \tag{4.13}$$

tal que, $\widehat{u}_{k-1} = f([\mathbf{P}^{-1}\mathbf{L}_2^{(N_{it})}]_{k-1})$.

Existem variações do algoritmo BCJR-MAP, como o BCJR-MaxLogMAP [2] que possuem desempenho idêntico ao algoritmo de Viterbi em termos de taxa de erro de bits. Antes de iniciarmos a discussão de desempenho, por definição, quando mencionamos *soft* queremos dizer que os valores são representados no mundo contínuo, como os valores reais, e *hard* significa valores discreto, como valores binários. O algoritmo de Viterbi, também chamado de HOVA (*Hard Output Viterbi Algorithm*), é um sistema SIHO (*Soft Input Hard Output*) com uma recursão para frente envolvendo decisões contínuas, sendo que a recursão termina com uma decisão de limiar depois que as estimativas já foram mapeadas para o mundo discreto (*hard decisions*), conforme descrito no capítulo anterior. Em sistemas concatenados, melhora-se o desempenho geral do decodificador com estimativas contínuas na saída *soft decisions* de cada bloco. Os algoritmos BCJR são do tipo SISO (*Soft Input Soft Output*). Na verdade o algoritmo de Viterbi pode ser modificado para ser SISO, ao qual denominamos de SOVA (*Soft Output Viterbi Algorithm*). O BCJR-MaxLogMap pode ser considerado uma implementação de SOVA. O algoritmo BCJR-MAP também é SISO e executa duas recursões na treliça, uma para frente e outra para trás. Esta recursão para trás insere uma complexidade maior no BCJR-MAP. Podemos dizer que o decodificador MAP minimiza a probabilidade de erro dos bits estimando probabilidades *a posteriori* de cada bit em uma seqüência de símbolos recebida. Embora mais complexo, o desempenho em taxa de erro do BCJR é sempre melhor ou igual ao algoritmo de Viterbi, por gerar estimativas melhores e contínuas nos estágios internos do algoritmo de decodificação [2] [1] [14] a [21].

Em termos de implementação, para códigos concatenados, podemos ter um modo serial e outro paralelo. No modo serial os decodificadores são ativados de forma seqüencial, onde o segundo decodificador utiliza a informação extrínseca do decodificador 1 como informação *a priori*, em um processo iterativo. No modo paralelo, os decodificadores são ativados juntos e processam simultaneamente as informações.

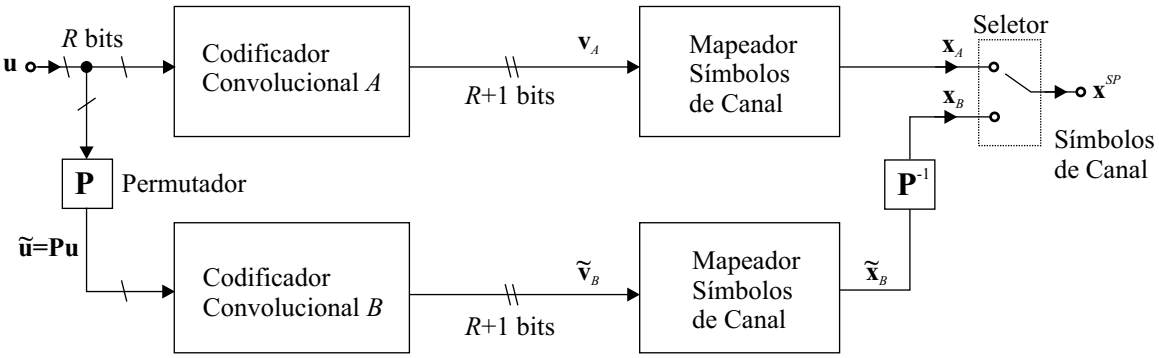


Figura 4.3: Esquema do codificador da turbo modulação codificada por treliças [2].

No nosso caso, onde utilizamos somente dois codificadores, o desempenho é similar [2].

4.3 TTCQ

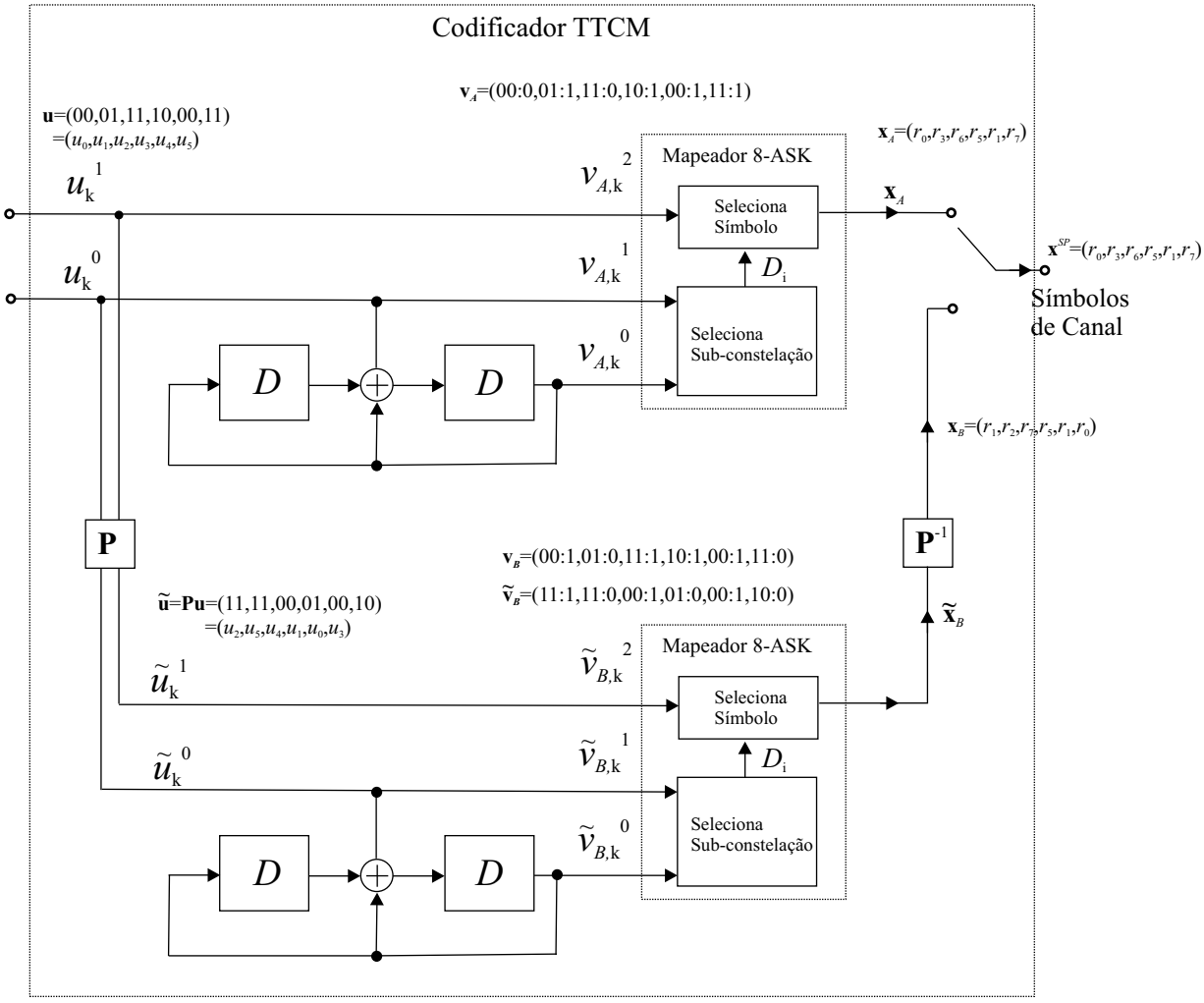
Para chegarmos na formulação da quantização codificada por treliças (TTCQ), vamos seguir os mesmos passos do capítulo anterior, na TCQ. Acabamos de verificar que os códigos turbo são implementados de maneira simples através de códigos convolucionais sistemáticos e recursivos, sendo que a utilização de um permutador pseudo-aleatório aumenta o desempenho da taxa de erros. A TCM apresentou ganhos de codificação significativos em relação à outras modulações multi-níveis, aumentando a eficiência espectral sem aumentar a banda de transmissão.

A turbo modulação codificada por treliças (TTCM), proposta por [5], tenta reunir as características da TCM e dos códigos turbo, substituindo os dois códigos convolucionais dos códigos turbo, por dois codificadores iguais aos utilizados na TCM, por exemplo, os codificadores da figura 3.5. O codificador TTCM está ilustrado na figura 4.3. Basicamente, ele é composto por dois códigos convolucionais sistemáticos e recursivos seguidos de mapeadores de símbolos de canal que serão transmitidos. Da mesma forma que na TCM, na TTCM, cada grupo de R bits na entrada produz uma saída intermediária com grupos de $R+1$ bits. Existem algumas diferenças na estrutura do codificador TTCM em relação a TCM. A permutação ocorre em grupos de bits, onde cada grupo possui R bits. A permutação não ocorre no nível de bits, somente entre os grupos. Cada um desses representará um símbolo

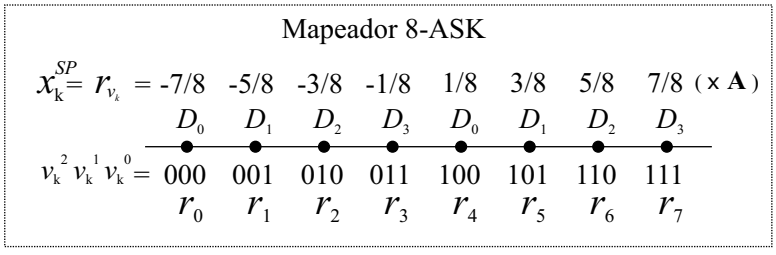
de canal. Existe uma maior complexidade de operação na perfuração da informação de paridade para ajustar o codificador da TTCM à eficiência espectral desejada. Os permutadores também são mais complexos, pois possuem restrições específicas em sua estrutura. Para visualizarmos a operação do codificador na TTCM, utilizamos a figura 4.4(a). Temos um modulador com eficiência espectral de 2 bits/s/Hz. O mapeador 8-ASK é utilizado novamente para o exemplo, e as sub-constelações estão representadas em 4.4(b). Os bits da saída de cada codificador convolucional, selecionam a sub-constelação e os bits de informação sistemática selecionam o símbolo dentro da sub-constelação. No caso da modulação ASK podemos rever que somente um dos bits é codificado. Um processo de permutação $\mathbf{P}$ é ilustrado na figura 3.8(c). Como queremos uma eficiência onde dois bits da entrada mapeiam 1 símbolo de canal, precisamos de um perfurador dos símbolos na saída, que neste caso será um selecionador, mantendo a mesma banda de transmissão na saída.

Para uma seqüência de entrada $\mathbf{u}$ =(00,01,11,10,00,11), temos dois vetores intermediários produzidos. A saída do codificador convolucional A é $\mathbf{v}_A$ = (00:0, 01:1,11:0,10:1,00:1,11:1), que é idêntica ao exemplo que vimos na TCM. A saída do codificador convolucional B é diferente, pois a entrada $\mathbf{u}$ sofre uma permutação antes de ser codificada, gerando $\widetilde{\mathbf{v}}_B$ =(11:1,11:0,00:1,01:0,00:1,10:1). Por meio de uma permutação inversa, os símbolos de $\widetilde{\mathbf{v}}_B$ são re-arrumados para formarem $\mathbf{v}_B$. A permutação utilizada mapeia as posições pares em pares e ímpares em ímpares, e os grupos são formados por 2 bits. Percebemos que a seqüência de bits sistemáticos dos vetores intermediários são iguais. Deste modo, a perfuração dos símbolos de paridade é realizada escolhendo de forma alternada as componentes da seqüência de símbolos de canal $\mathbf{x}_A$ e $\mathbf{x}_B$, garantindo que sempre teremos informações sistemáticas no símbolo transmitido, junto com as informações de paridade (por isso o sobrescrito no símbolo de canal $\mathbf{x}^{SP}$).

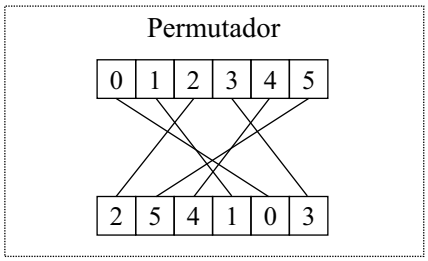
O decodificador da TTCM é semelhante ao da TCM, só que agora não podemos separar as informações sistemáticas das extrínsecas, pois o mesmo símbolo de canal transporta as duas informações simultaneamente. Com isso, temos uma nova informação no processo de decodificação chamada de mista, que correspondem a $\mathbf{L}_{12}^{es}$ e $\mathbf{L}_{21}^{es}$ na figura 4.5, onde o sobrescrito SP significa informação extrínseca e sistemática não separáveis. Neste caso, o algoritmo de BCJR é modificado [2].



(a)



(b)



(c)

Figura 4.4: Exmeplo do codificador da turbo modulação codificada por treliças [2] (a) com mapeador 8-ASK (b) e permutador conforme (c).

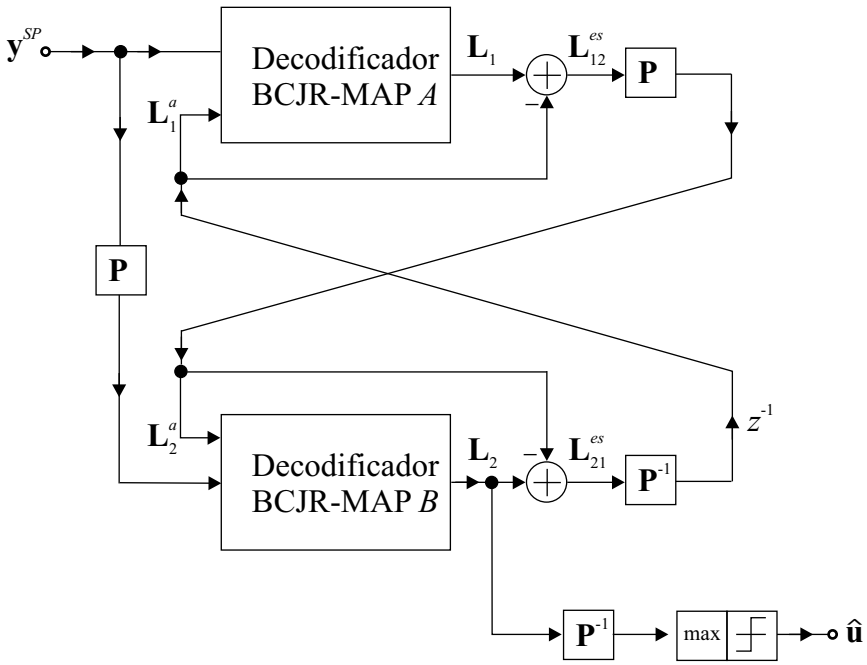


Figura 4.5: Esquema do decodificador da turbo modulação por treliças [2].

O BCJR-MAP primeiro encontra o símbolo $i \in \mathcal{U}^*$ onde $\mathcal{U} = \{0, 1, ..., 2^R - 1\}$ e $\mathcal{U}^* = \mathcal{U} - \{0\}$, para $k = 1, 2, ..., N$, a partir de

$$\frac{\Pr\{U_{k-1} = i|\mathbf{y}\}}{\Pr\{U_{k-1} = 0|\mathbf{y}\}} = \max\left\{\frac{\Pr\{U_{k-1} = j|\mathbf{y}\}}{\Pr\{U_{k-1} = 0|\mathbf{y}\}} : j \in \mathcal{U}^*\right\} \tag{4.14}$$

e decide se o símbolo $u_{k-1} = i$ foi transmitido se

$$\Pr\{U_{k-1} = i|\mathbf{y}\} > \Pr\{U_{k-1} = 0|\mathbf{y}\} \tag{4.15}$$

Do contrário, decide que $u_{k-1} = 0$ foi transmitido. Para facilitar a tomada de decisões, generalizamos a expressão do logaritmo da razão de probabilidades *a posteriori* para

$$L_{k-1}(i) = \ln\left[\frac{\Pr\{U_{k-1} = i|\mathbf{y}\}}{\Pr\{U_{k-1} = 0|\mathbf{y}\}}\right] \tag{4.16}$$

de modo que

$$u_{k-1} = \begin{cases} i, & L_{k-1} > 0 \\ 0, & L_{k-1} < 0, \end{cases} \tag{4.17}$$

tal que $i \in \mathcal{U}^*$ é o símbolo para o qual $L_{k-1}(i) = \max\{L_{k-1} : j \in \mathcal{U}^*\}$.

$$L_{k-1}(i) = L^a_{k-1}(i) + \ln\left[\frac{\sum_{(s',s)\in\mathcal{B}^i_{k-1}} \alpha_{k-1}(s')\gamma^{es}_k(s',s)\beta_k(s)}{\sum_{(s',s)\in\mathcal{B}^0_{k-1}} \alpha_{k-1}(s')\gamma^{es}_k(s',s)\beta_k(s)}\right]. \tag{4.18}$$

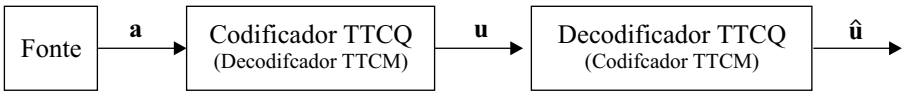


Figura 4.6: Esquema de codificação e decodificação da TTCQ.

Seguindo os mesmos passos utilizados para originar a equação 4.5, temos a equação 4.18, onde os valores de $\alpha_{k-1}(s')$ e $\beta_k(s)$ são encontrados de forma idêntica às equações 4.6 e 4.7. O valor de γ nesta caso é dado por

$$\gamma_k^{es}(s', s) = e^{\frac{x_{k-1}^{SP}(s',s)\left(y_{k-1}^{SP} - \frac{1}{2} y_{k-1}^{SP}(s',s)\right)}{\sigma^2}} \tag{4.19}$$

Vemos então que a estimativa $L_{k-1}(i)$ é a soma da informação *a priori* $L_{k-1}^a(i)$ e da informação extrínseca e sistemática não separáveis [2].

O processo de decodificação turbo pode ser implementado de forma serial ou de forma paralela. Na forma serial, um decodificador ativa o outro, ou seja, utiliza a estimativa como informação *a priori*. No caso da implementação paralela, os codificadores são ativados ao mesmo tempo, pela informação *a priori*, da iteração anterior. Exemplificamos um modo de operação serial na figura 4.2. A informação extrínseca e sistemática do decodificador A será a estimativa *a priori* do decodificado B. A cada iteração, a decisão do símbolo é dada pela equação 4.17.

Da mesma forma que utilizamos a modulação codificada por treliças no quantizador codificado por treliças [6], fazemos a utilização direta da turbo modulação codificada por treliças no turbo quantizador codificado por treliças [2]. Com isso temos a estrutura de codificação e decodificação do TTCQ na figura 4.6. É importante observar que para a TTCM, a utilização de blocos grandes na entrada do turbo quantizador gera resultados melhores com poucas iterações. No caso da TTCQ o tamanho do bloco na entrada do turbo codificador é um limitante, e o número de iterações necessários para a convergência dos algoritmos de estimativas é maior que na TCQ [2].

Capítulo 5

Codificação com Turbo Quantização Codificada por Treliças

Depois que a teoria dos códigos turbos foi apresentada, podemos pensar em aplicações para o turbo quantizador codificado por treliças. O codificador de Joshi [23] apresentou ganhos em relação às técnicas existentes. O desempenho da TCQ como quantizador do sistema, apresentou resultados muito positivos. Como a TTCQ apresenta ganhos de codificação em relação à TCQ [2], resolvemos aplicar a TTCQ no mesmo codificador proposto por Joshi. Desta forma, nos beneficiaríamos dos ganhos da TTCQ, para aumentar ainda mais o desempenho do codificador TCQ. Os resultados do novo codificador são apresentados no final do capítulo.

5.1 Codificador de imagem com ACTTCQ

O nome do codificador proposto neste trabalho é ACTTCQ-SBC (*Arithmetic Coded Turbo Trellis Coded Quantization-SubBand Coding*). O diagrama de blocos do codificador está ilustrado na figura 5.1. Se compararmos este sistema com o sistema de Joshi, figura 3.11, percebemos que a única diferença é a utilização do TTCQ como quantizador.

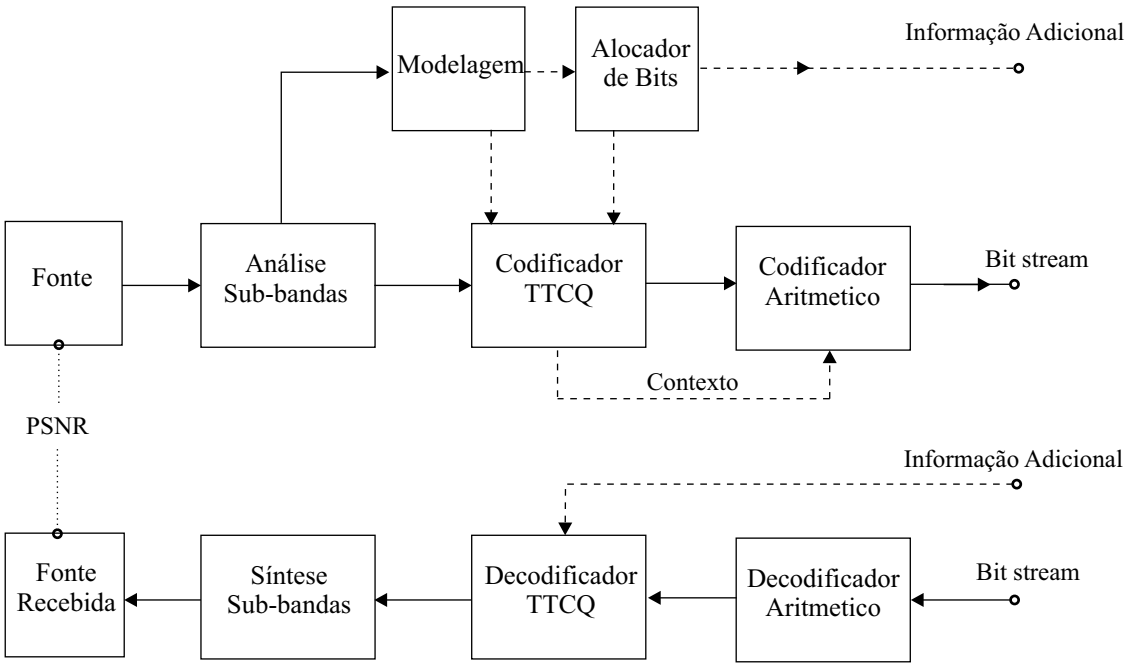


Figura 5.1: Codificador e decodificador de imagens ACTTCQ-SBC.

5.2 Codificação em sub-bandas e transformadas

A imagem de entrada passa pelo processo de análise em sub-bandas, para descorrelacionar as amostras. Implementamos dois tipos de decomposição em sub-bandas: uniforme e não uniforme. Na decomposição uniforme, utilizamos um banco de filtros QMF 32D da família de filtros de Johnston [20]. Este banco de filtros não é reconstrução perfeita, mas apresenta distorção de amplitude em torno de ± 0.025 dB na banda passante [20]. Após a decomposição em sub-bandas, a banda LFS é transformada pela DCT 4x4. Aplicamos a DCT porque a LFS apresenta uma correlação entre as amostras muito grande, mesmo depois da decomposição uniforme em sub-bandas. Na figura 5.2 ilustramos o resultado da análise em 16 sub-bandas uniformes com a transformada DCT 4x4 da LFS.

Também implementamos a decomposição em sub-bandas não uniformes, através da implementação do banco de filtros QMF em árvores hierárquicas desbalanceadas na LFS. O par de filtros 9-7 *spline* bi-ortogonal foi utilizado na decomposição [18] [24]. Podemos considerar que este processo é uma transformada *wavelet* bi-dimensional da imagem LFS. Utilizamos 2 níveis de decomposição, gerando 15 sub-bandas uniformes e 7 bandas não uniformes. Na figura 5.3 ilustramos o resultado

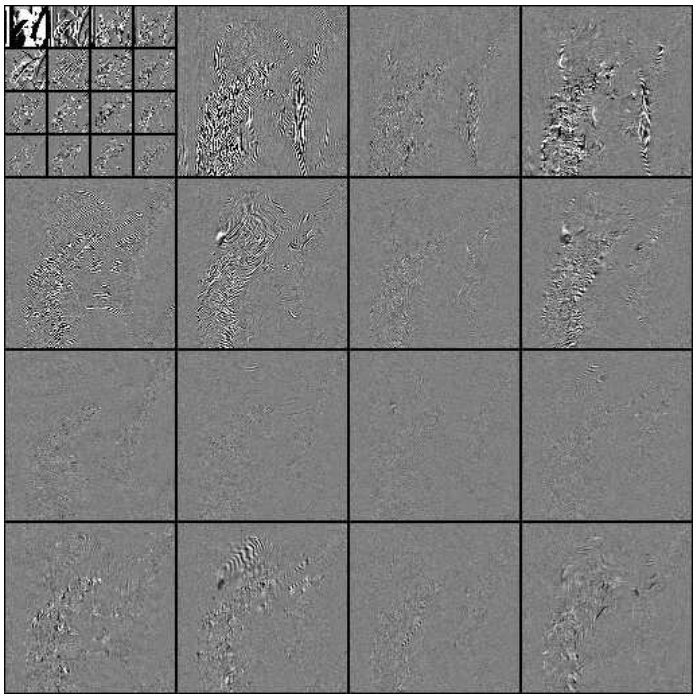


Figura 5.2: Imagem Lena 512x512 após um análise em 16 sub-bandas uniformes, e aplicando a DCT 4x4 na LFS.

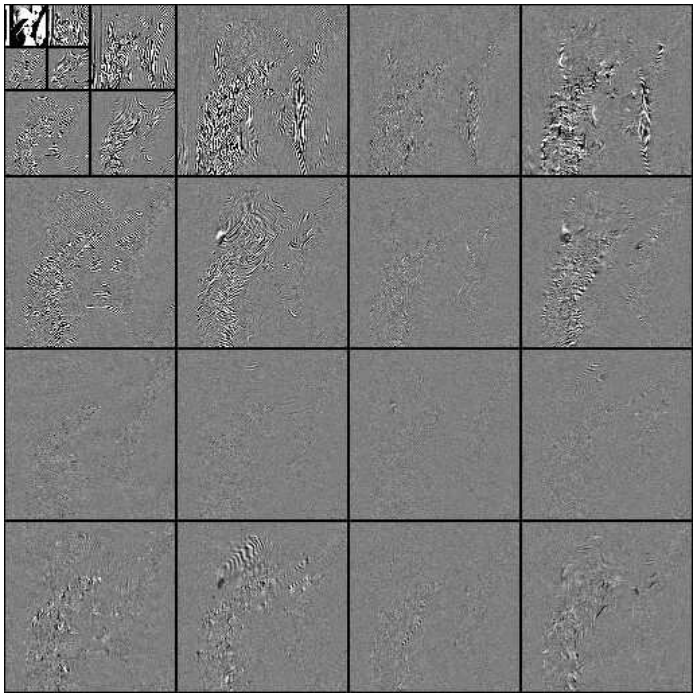


Figura 5.3: Imagem Lena 512x512 após um análise em 16 sub-bandas uniformes, e aplicando a transformada *wavelet* na LFS).

da análise em 22 sub-bandas não uniformes, da imagem Lena.

5.3 Modelagem

Seguindo o processo de codificação, cada sub-banda será modelada para uma distribuição estatística. Utilizamos uma família de gaussianas generalizadas para representar as sub-bandas [12]. A distribuição GGD é totalmente definida por dois parâmetros ν e σ. A partir da sub-banda, temos a sua variância σ^2 e calculamos o parâmetro ν através da estimação apresentada nas equações 3.9 e 3.10. Depois da estimação dos parâmetros, restringimos os valores possíveis de gaussianas generalizadas para

$$\nu \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0\} \ .$$

Este conjunto foi achado experimentalmente [23]. Na imagem Lena, durante as simulações, percebemos uma grande ocorrência de $\nu = 0.5$, mostrando que as sub-bandas possuem histogramas "pontudos". Este fato também pode ser um indicador de que após este processo, as sub-bandas estão bem intra-descorrelacionadas.

5.4 Quantização

Depois que as sub-bandas foram modeladas e aproximadas para o conjunto restrito de GGD, precisamos determinar os parâmetros do quantizador. Antes do processo de codificação, utilizamos seqüências de treinamento para configurar o quantizador. Na verdade, geramos tabelas com todas as configurações possíveis, de modo que, determinados a taxa e o parâmetro ν da entrada, temos a configuração do quantizador e uma distorção associada àquela sub-banda. Existem 3 parâmetros a serem definidos: o passo de quantização uniforme α, o modo de disposição dos níveis de reconstrução (vide abaixo) e o número de níveis de reconstrução.

Para definir estes parâmetros, utilizamos seqüências de treinamento com os modelos de distribuição GGD restritos. Uma forma de definir qual a melhor configuração do quantizador, é gerar todas as combinações de valores para os parâmetros mencionados acima:

- O modo de disposição dos níveis de reconstrução consiste em duas possibilidades, sem sobreposição do nível "0", ilustrado na figura 3.16, e com sobreposição do nível "0", figura 3.17;
- O passo de quantização α é inicializado com o valor 1/64 e a cada linha da tabela é somado 1/64;
- O número de níveis de reconstrução é inicializado com 8 níveis e é dobrado a cada linha da tabela;

Esses valores foram achados experimentalmente, para o tipo de entrada no quantizador, que são imagens monocromáticas com valores de 0 a 255. A partir desta tabela, inserimos mais 3 variáveis:

- a taxa
- o modelo de distribuição da entrada
- a distorção gerada para uma determinada configuração de quantizador

Para cada valor de ν, temos uma tabela com o modo, α, número de níveis, taxa e distorção. Só que ainda não variamos a taxa. Variando a taxa, calculamos a distorção gerada por cada linha da tabela (ou seja, cada quantizador). A distorção é medida pelo erro médio quadrático entre entrada e saída. O critério de parada para continuar preenchendo a tabela com novas entradas (linhas), é a diferença entre a distorção da configuração atual e antiga ser menor que 0.001 dB.

Com esta tabela pronta, montamos uma curva operacional taxa-distorção. Esta curva possui em média 256 pontos. O valor de taxa de bits na verdade é o número de bits alocado para quantizar a seqüência. Logo, o processo de configuração do quantizador é somente uma *look-up table*.

5.5 Alocação de bits

A definição da taxa é responsabilidade do bloco de alocação de bits. Este algoritmo define a quantidade de bits que será disponibilizada para quantizar cada sub-banda. Utilizamos o algoritmo de Westerink [25], que está descrito na seção 3.3, para fazer a alocação de bits das sub-bandas em conjunto levando em consideração

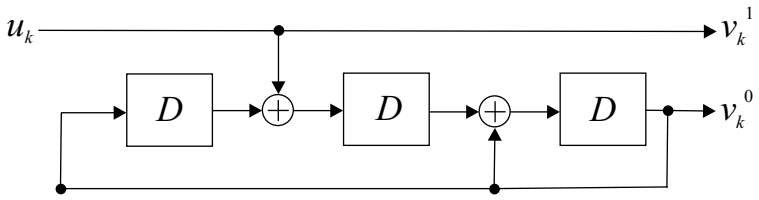


Figura 5.4: Estrutura recursiva do código convolucional (13,4) utilizado no ACTTCQ.

as curvas operacionais taxa-distorção de cada sub-banda. Após a alocação ótima, sabemos qual a taxa a ser utilizada em cada sub-banda. Com isso, procuramos na tabela de configurações que foi montada e achamos a configuração do quantizador para aquela taxa.

O processo de quantização TTCQ, descrito no capítulo 4, é aplicado na seqüência de entrada. O permutador utilizado é um *S-Random interleaver*. Em nossa implementação, os códigos convolucionais do turbo quantizador são iguais. A estrutura deste código é determinada pelo octal (13,4) e pode ser vista na figura 5.4. O mapeamento dos símbolos é um N-ASK, onde N é determinado pelo número de níveis do quantizador.

José Fernando em [2], fez um estudo do desempenho da TTCQ de acordo com o tamanho do bloco do turbo quantizador. Foi mostrado que, para taxas de 1 bit por amostra do bloco, para fontes gaussianas sem memória, média zero e variância unitária, o tamanho de bloco ótimo era em torno de 96 (existia uma pequena variação de acordo com a implementação do modo do decodificador [2]) e para fontes laplacianas sem memória, média zero e variância unitária, o tamanho do bloco era 64. Com isso, inserimos mais um parâmetro a ser configurado quando modelamos as sub-bandas. Nas figuras 5.5, 5.6 e 5.7 podemos ver os resultados do TTCQ em relação à dimensão do bloco no turbo quantizador.

No codificador implementado não foi realizada a otimização do tamanho de bloco para cada fonte GGD do conjunto restrito, listado anteriormente. Estimamos um valor de tamanho de bloco da seguinte maneira. Sabemos que Gaussiana e a Laplaciana são casos gerais da GGD, para $\nu = 2$ e $\nu = 1$, respectivamente. Como o tamanho de bloco ótimo diminui de uma distribuição para outra, mantivemos o mesmo padrão de diminuição para os outros valores de ν, sendo que o tamanho do

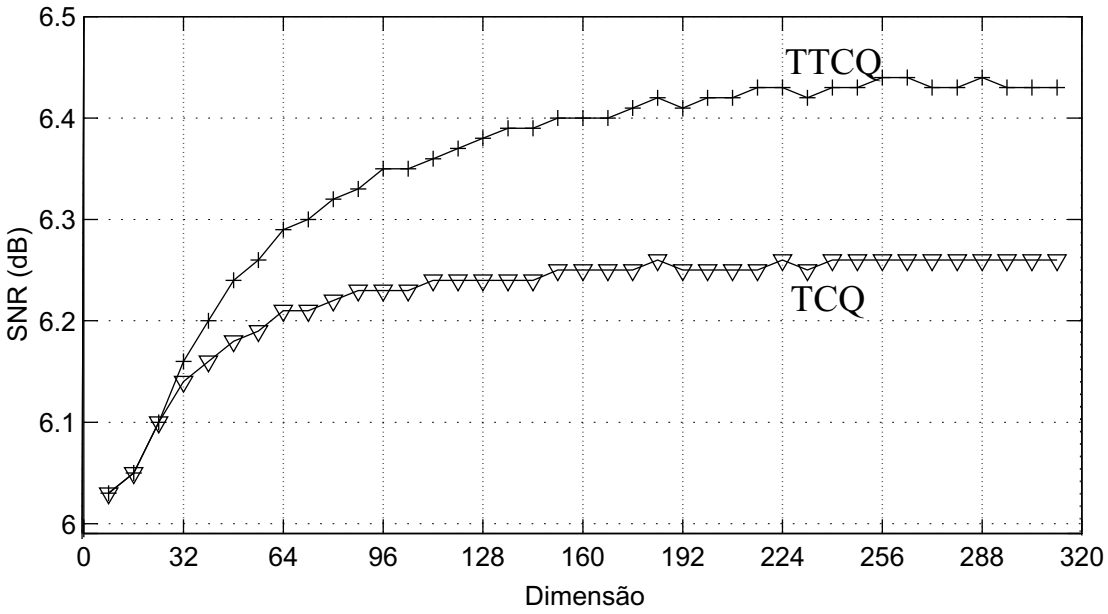


Figura 5.5: Comparação do desempenho em SNR dos quantizadores TCQ e TTCQ, para fonte uniforme, sem memória, de média zero e variância unitária.

bloco mínimo é 16. A otimização dos tamanhos de bloco ótimo, para cada taxa de bit por amostra, é uma proposta de trabalho futuro, e tende a aumentar o desempenho do quantizador.

Existem dois modos de gerarmos as tabelas que definem o quantizador. No primeiro modo, geramos a tabela calculando a distorção através de um algoritmo HOVA, no caso o algoritmo de Viterbi. O outro modo, que deve melhorar o desempenho do quantizador, é utilizar um algoritmo SISO, no caso o BCJR-MAP. O processo de gerar as tabelas do quantizador é muito demorado, e para a apresentação dos resultados, conseguimos simular somente as tabelas da codificação HOVA. Acreditamos que existirá um ganho quando as tabelas forem geradas com codificação baseada em *soft decisions*, devido aos resultados verificados também nas figuras 5.5, 5.6 e 5.7.

Durante a quantização, também utilizamos um codificador aritmético na saída dos níveis de reconstrução escolhidos para representar a seqüência de entrada. Da mesma forma que no codificador de imagens apresentado no capítulo 3, o codificador aritmético é adaptativo e muda a tabela de probabilidade de acordo com o nível de reconstrução escolhido. Esta tabela de probabilidade é responsável pela sub-divisão do intervalo no algoritmo de codificação aritmética, mostrado no

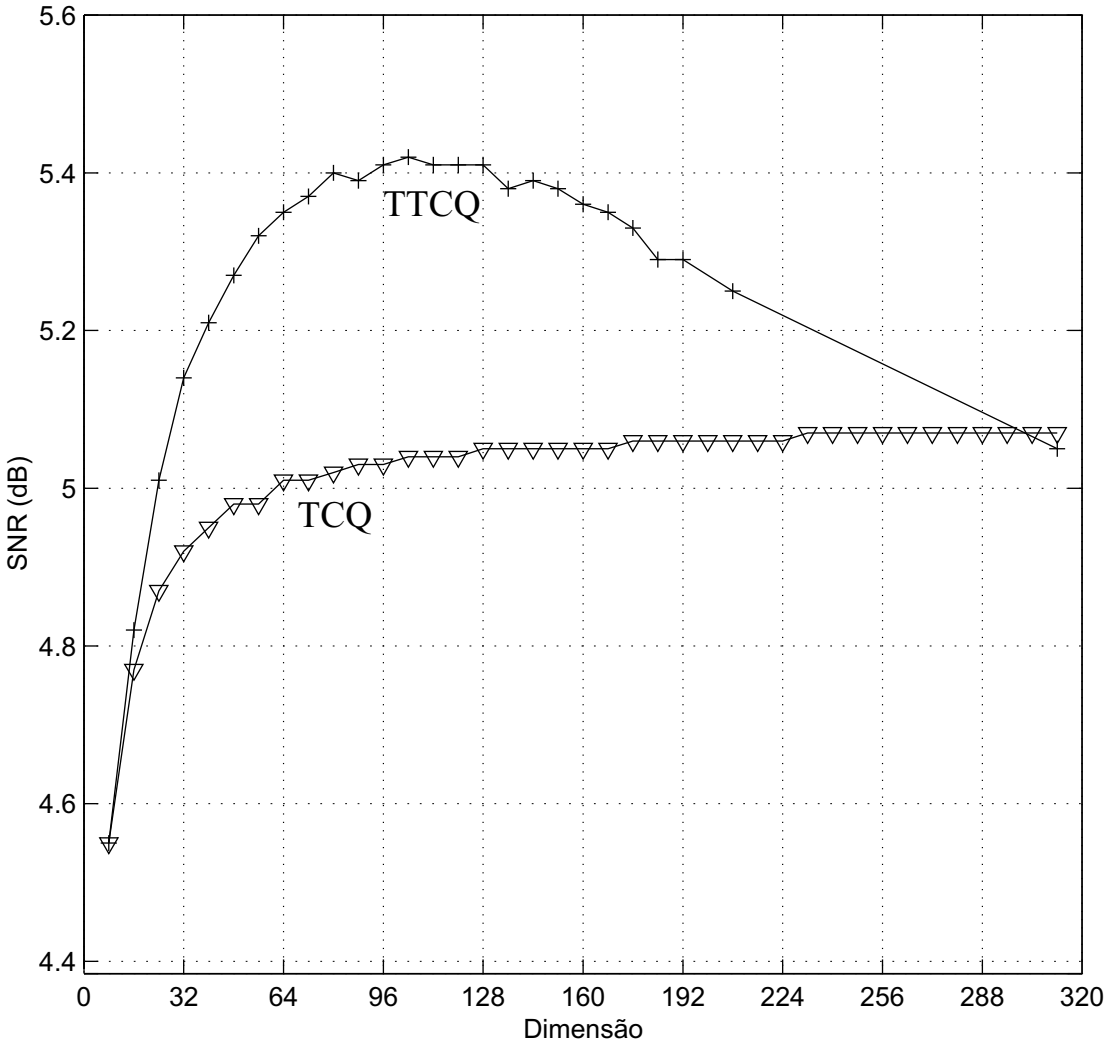


Figura 5.6: Comparação do desempenho em SNR dos quantizadores TCQ e TTCQ, para fonte gaussiana, sem memória, de média zero e variância unitária.

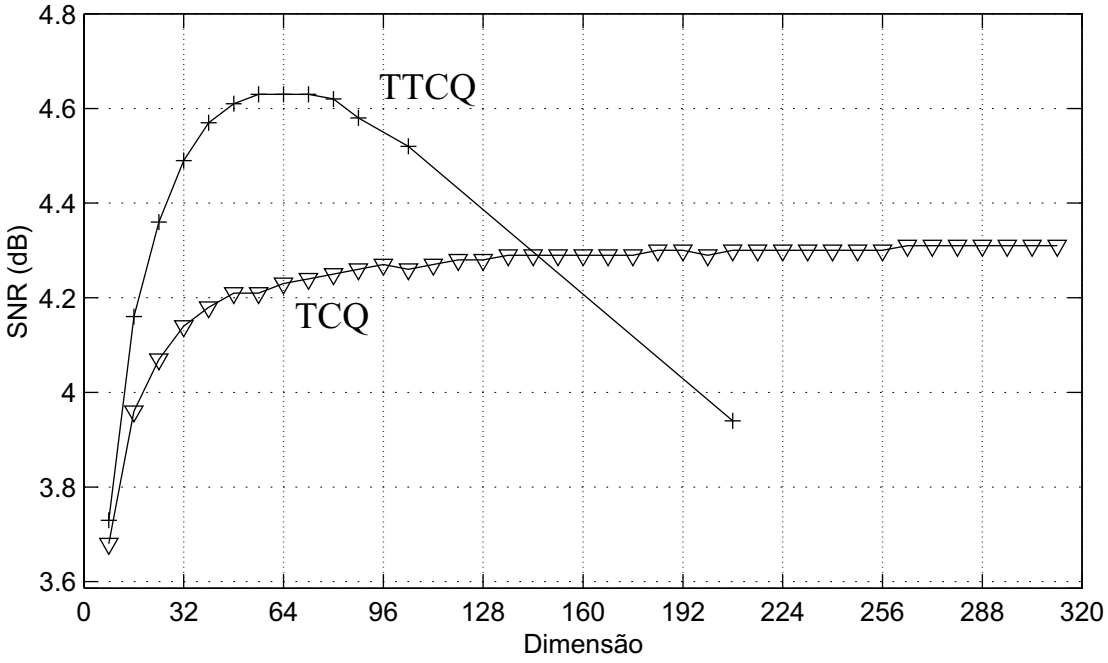


Figura 5.7: Comparação do desempenho em SNR dos quantizadores TCQ e TTCQ, para fonte laplaciana, sem memória, de média zero e variância unitária.

capítulo 2.

As informações que transmitimos são: o valor médio da imagem, o valor médio da sub-banda LFS, o parâmetro ν, o desvio padrão σ de cada sub-banda, e o identificador da codificação aritmética.

No decodificador, os centróides dos níveis de reconstrução são calculados através de seqüências de teste. Estas seqüências possuem 4096 amostras e são passadas em 7 rodadas. Os centróides são achados através do algoritmo *k-means*. Este algoritmo pode ser encontrado em [18] e foi explicado no capítulo 3. Ele acha os centróides através de aproximações sucessivas. Cada amostra representa um símbolo no decodificador. Cada símbolo é associado a um centróide inicial, de acordo com a proximidade. Depois que o símbolo é escolhido, o algoritmo faz uma média entre o valor das amostras mapeadas para aquele determinado símbolo e a quantidade de amostras mapeadas, achando o centróide do grupo.

Após a decodificação aritmética, o BCJR-MAP no modo paralelo [2] é aplicado para tentar recuperar as sub-bandas quantizadas. Lembramos novamente que a tabela de configuração do decodificador não é otimizada para o BCJR. Depois

Tabela 5.1: Resultados da codificação ACTTCQ-SBC da imagem Barbara 512x512 para 1 iteração.

22 bandas 9,7 bi-ort		16 bandas 32D QMF	
Taxa (bpp)	PSNR (dB)	Taxa (bpp)	PSNR (dB)
0.249	26.70	0.237	27.20
0.348	28.00	0.336	28.79
0.452	29.27	0.435	30.18
0.552	30.60	0.538	31.49

Tabela 5.2: Resultados da codificação ACTTCQ-SBC da imagem Lena 512x512 para 1 iteração.

22 bandas 9,7 bi-ort		16 bandas 32D QMF	
Taxa (bpp)	PSNR (dB)	Taxa (bpp)	PSNR (dB)
0.234	32.61	0.244	32.54
0.345	34.21	0.355	34.31
0.461	35.47	0.467	35.61
0.574	36.46	0.580	36.65

fazemos o processo de síntese das sub-bandas para encontrar a fonte reconstruída e podermos calcular o PSNR.

5.6 Resultados

Os resultados obtidos estão relacionados nas tabelas e gráficos a seguir.

A tabela 5.2 mostra o resultado da codificação da imagem Lena 512x512, com transformada *wavelet* em 22 bandas e DCt 4x4 em 16 bandas, utilizando o ACTTCQ-SBC com 1 iteração na decodifcação turbo. Juntando com as tabelas 5.3

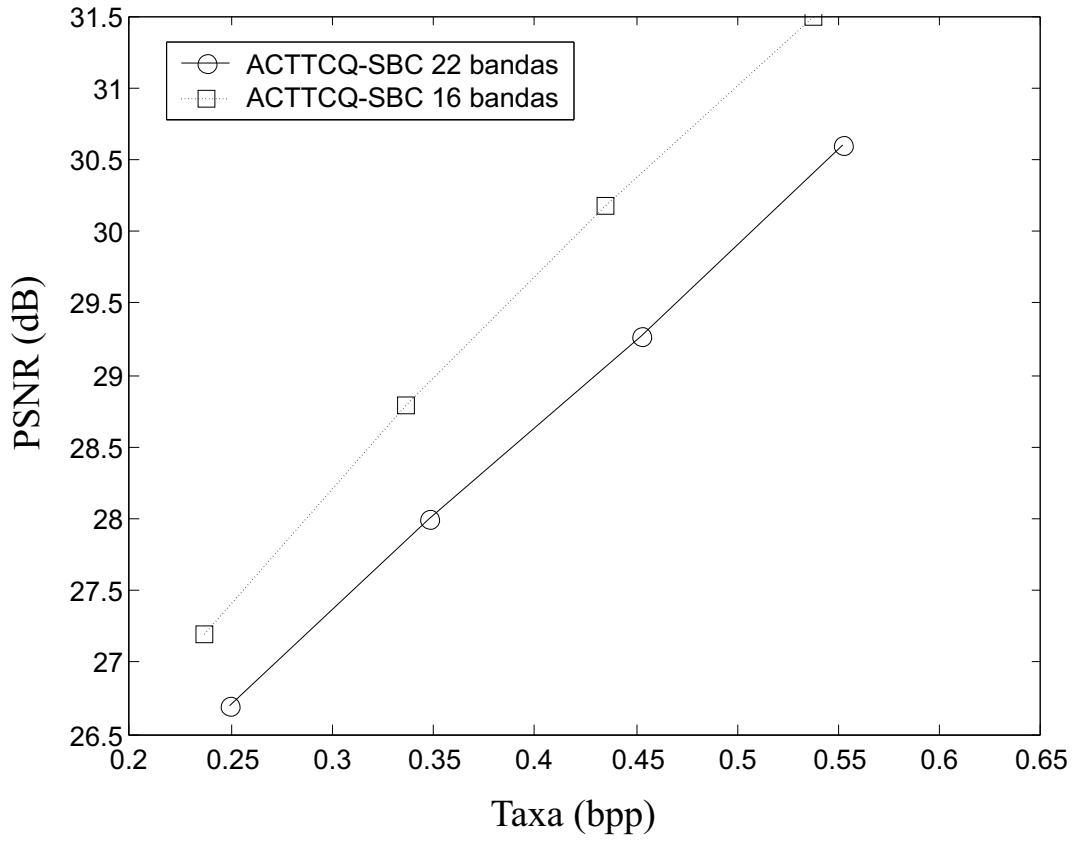


Figura 5.8: Resultados da codificação ACTTCQ-SBC da imagem Barbara 512x512 para 1 iteração.

Tabela 5.3: Resultados da codificação ACTTCQ-SBC da imagem Lena 512x512 para 10 iterações.

22 bandas 9,7 bi-ort		16 bandas 32D QMF	
Taxa (bpp)	PSNR (dB)	Taxa (bpp)	PSNR (dB)
0.240	32.73	0.248	32.67
0.354	34.35	0.363	34.45
0.470	35.61	0.478	35.77
0.588	36.62	0.595	36.82

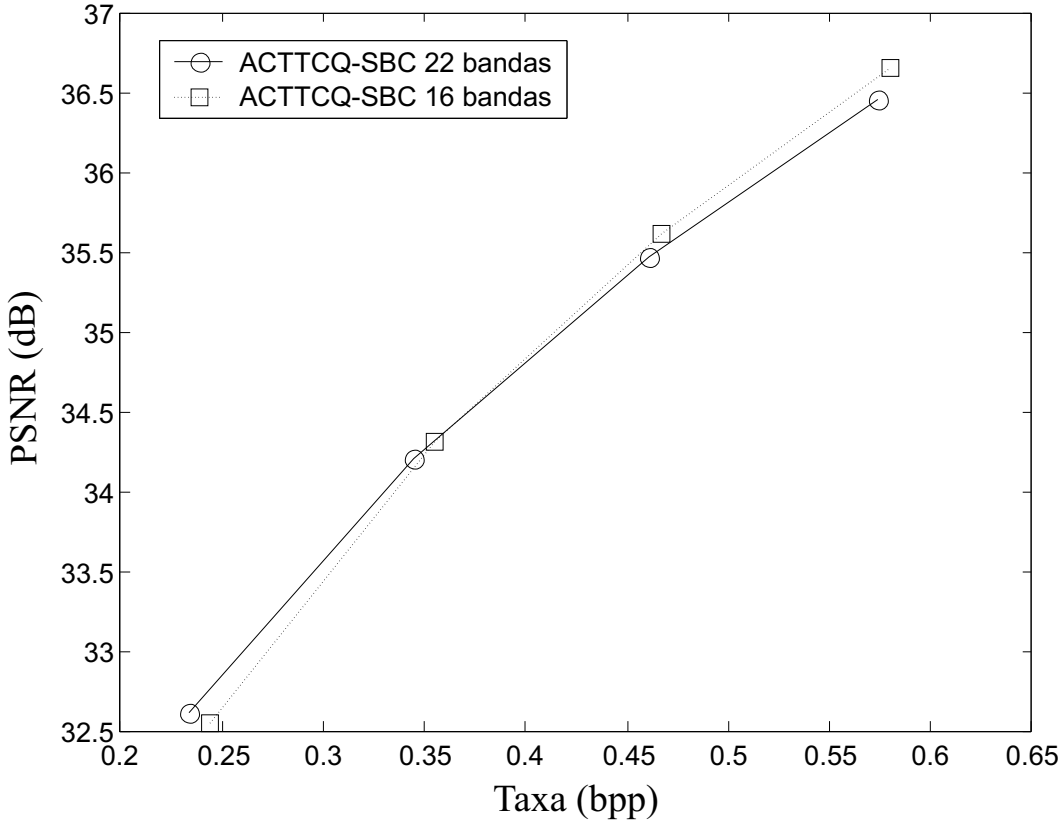
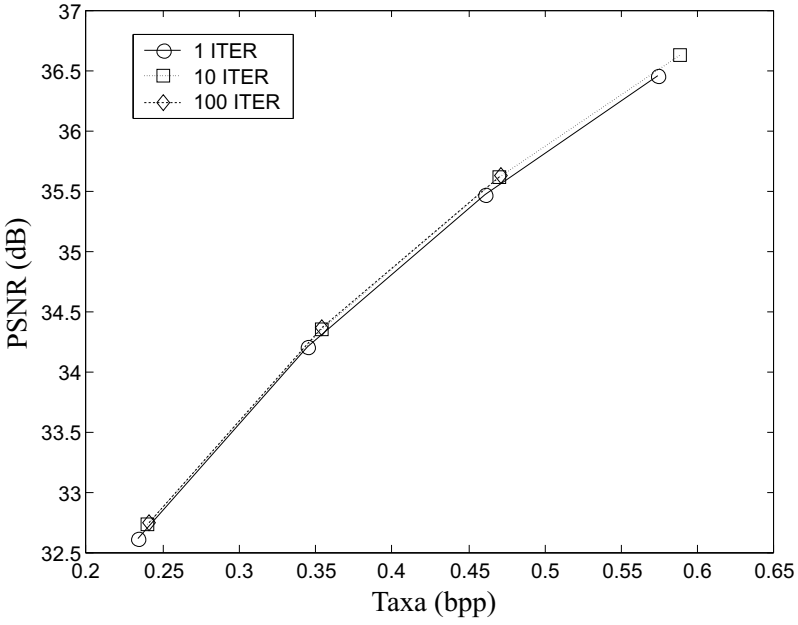


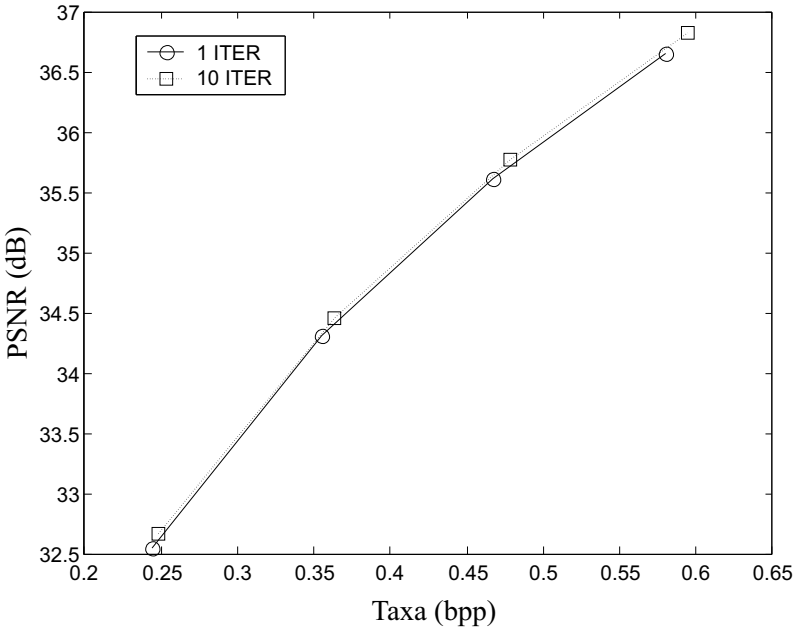
Figura 5.9: Resultados da codificação ACTTCQ-SBC da imagem Barbara 512x512 para 1 iteração.

Tabela 5.4: Resultados da codificação ACTTCQ-SBC da imagem Lena 512x512 para 100 iterações.

ACTTCQ-SBC 22 bandas	
Taxa (bpp)	PSNR (dB)
0.241	32.74
0.354	34.36
0.471	35.63



(a)



(b)

Figura 5.10: Comparação da codificação da Lena 512x512 para diferentes taxas, utilizando ACTTCQ-SBC, com diferentes números de iterações (a) decomposição em 22 sub-bandas (b) decomposição em 16 sub-bandas

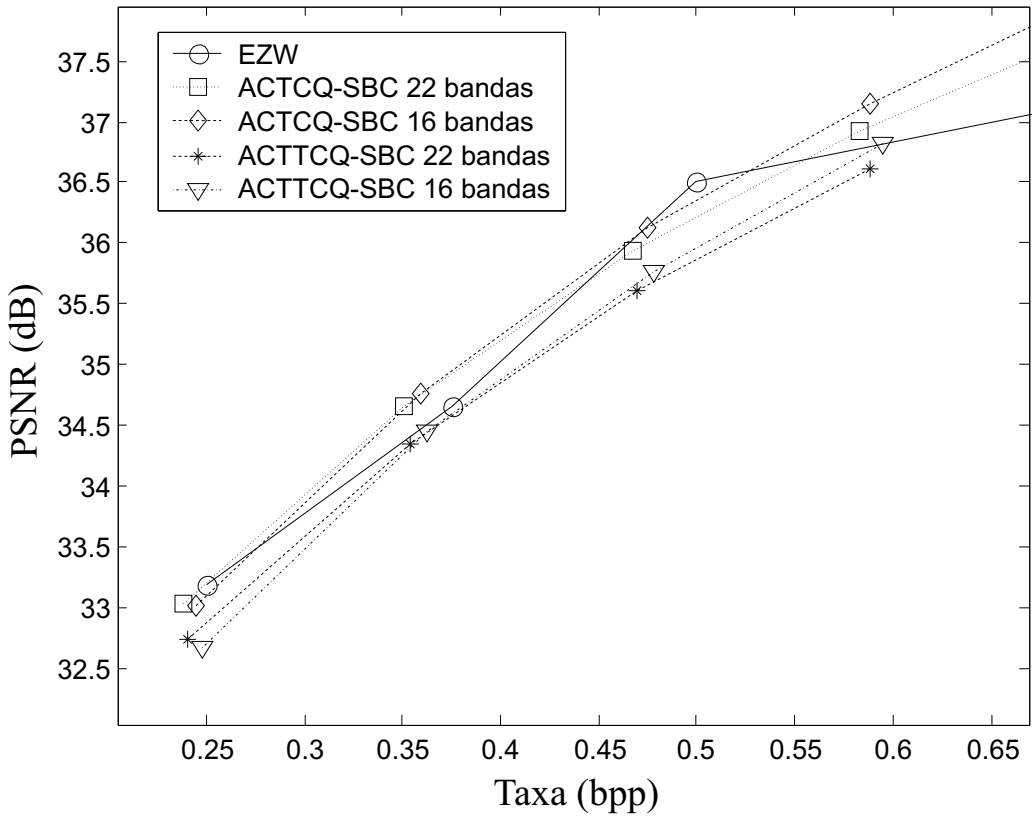


Figura 5.11: Comparação da codificação da Lena 512x512 para diferentes taxas, utilizando EZW, ACTCQ-SBC, ACTTCQ-SBC.

e 5.4 montamos o gráfico da figura 5.10. Nesta figura, vemos que não temos ganhos de codificação significativos para um número de iterações de 1 a 100.

Outra comparação que podemos fazer com os resultados das tabelas, é a diferença na decomposição em sub-bandas. Vemos que para decomposição uniforme da imagem Lena em 16 bandas supera, em termo de PSNR, a decomposição não uniforme em 22 sub-bandas, para taxas mais altas (figura 5.9). Este fato é inverso para taxas mais baixas. Visualmente, para taxas baixas, a imagem recuperada da decomposição uniforme possui um ruído gerado pela DCT da LFS, enquanto na decomposição não uniforme, não vemos tal artefato. No caso da imagem Barbara, figura 5.8 vemos claramente o ganho da decomposição em 16 bandas. Este fato ocorre pois a imagem Barbara possui mais componentes de alta freqüência.

Na figura 5.11, comparamos os codificadores EZW, ACTCQ-SBC e o ATTCQ-SBC. Vemos que o desempenho do ACTTCQ-SBC se aproxima muito do EZW, mas fica a 0.2dB de diferença do ACTCQ-SBC. Repetindo a análise de desempenho de

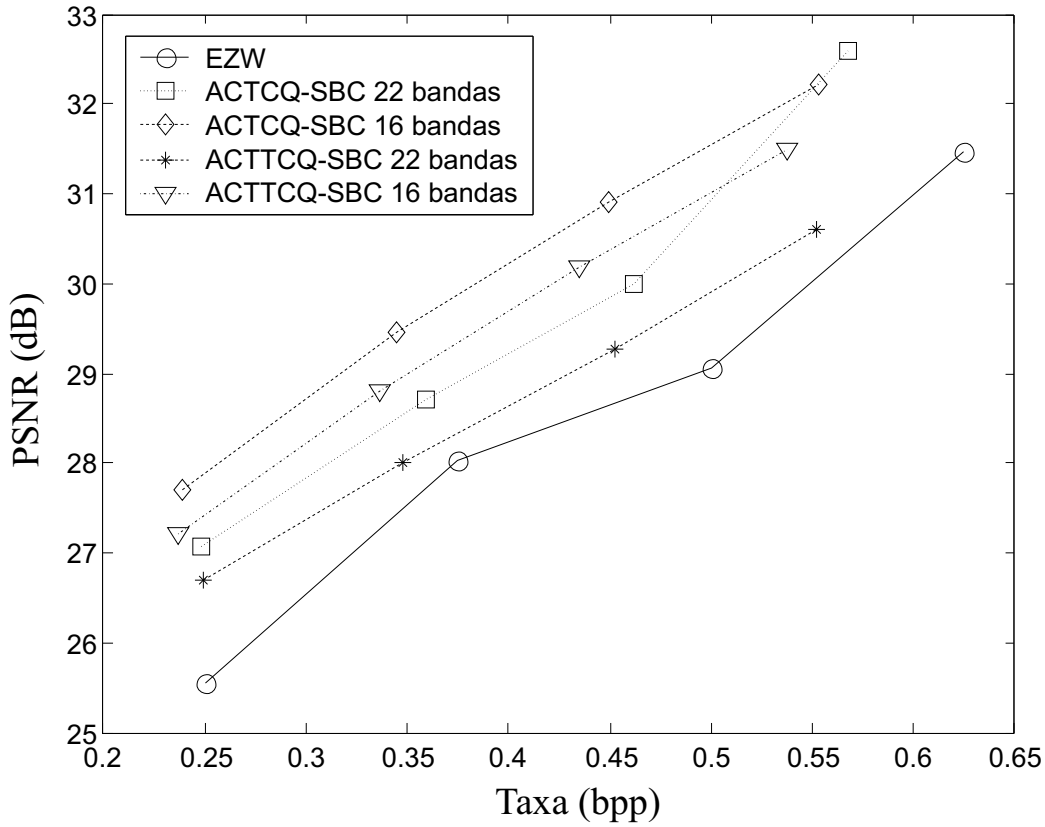


Figura 5.12: Comparação da codificação da Barbara 512x512 para diferentes taxas, utilizando EZW, ACTCQ-SBC, ACTTCQ-SBC.

codificadores para a imagem Bárbara, vemos na figura 5.12 que ambos codificadores ACTCQ-SBC e ACTTCQ-SBC superam o EZW, devido ao baixo desempenho deste último em imagens com alta freqüência espacial. O ACTTCQ-SBC com decomposição em 16 bandas utilizando QMF/DCT, supera o ACTCQ-SBC com decomposição em *wavelet*, mas fica 0.5dB abaixo do ACTCQ-SBC com decomposição em 16 bandas.

Em relação ao codificador que utiliza o SPHIT *set partitioning in hierarchical tree* [29], temos a seguinte comparação. Para a imagem Lena, o SPHIT apresenta PSNR 34.14dB e 37.25dB para as taxas de 0.250bpp e 0.500bpp, respectivamente. Comparando com os resultados do ACTTCQ-SBC percebemos um desempenho inferior em torno de 1.4dB em relação ao SPHIT, pelos resultados da tabela 5.2. Para a imagem Barbara, o SPHIT apresenta PSNR 24.58dB e 31.39dB para as taxas de 0.250bpp e 0.500bpp, respectivamente. Com isso, temos um desempenho inferior do ACTTCQ-SBC em torno de 0.2dB, pelos resultados da tabela 5.1. Esta proximidade de resultados na imagem Barbara é devido a grande quantidade de componentes de alta freqüência presentes nesta imagem, o que piora os resultados do SPHIT por ser baseados em árvores de zero [29].

Podemos justificar o desempenho do ACTTCQ-SBC devido a alguns fatores:

- A escolha não ótima do tamanho do bloco do turbo codificador;
- A utilização de uma tabela de configurações de parâmetros do quantizador não ótima, ou seja, estamos utilizando um algoritmo de *hard decisions* configurar um decodificador de *soft decisions*.

Contudo, é interessante observar que mesmo sem otimização, ficamos bem próximos dos resultados do ACTCQ-SBC. Seguem algumas imagens resultantes do codificador ACTTCQ-SBC para Lena e Bárbara.

Figura 5.13: Comparação da codificação da Lena 512x512 para diferentes taxas utilizando ACTTCQ-SBC. (a) 0.250bpp (b) 0.375bpp (c) 0.500bpp (d) 0.625bpp.

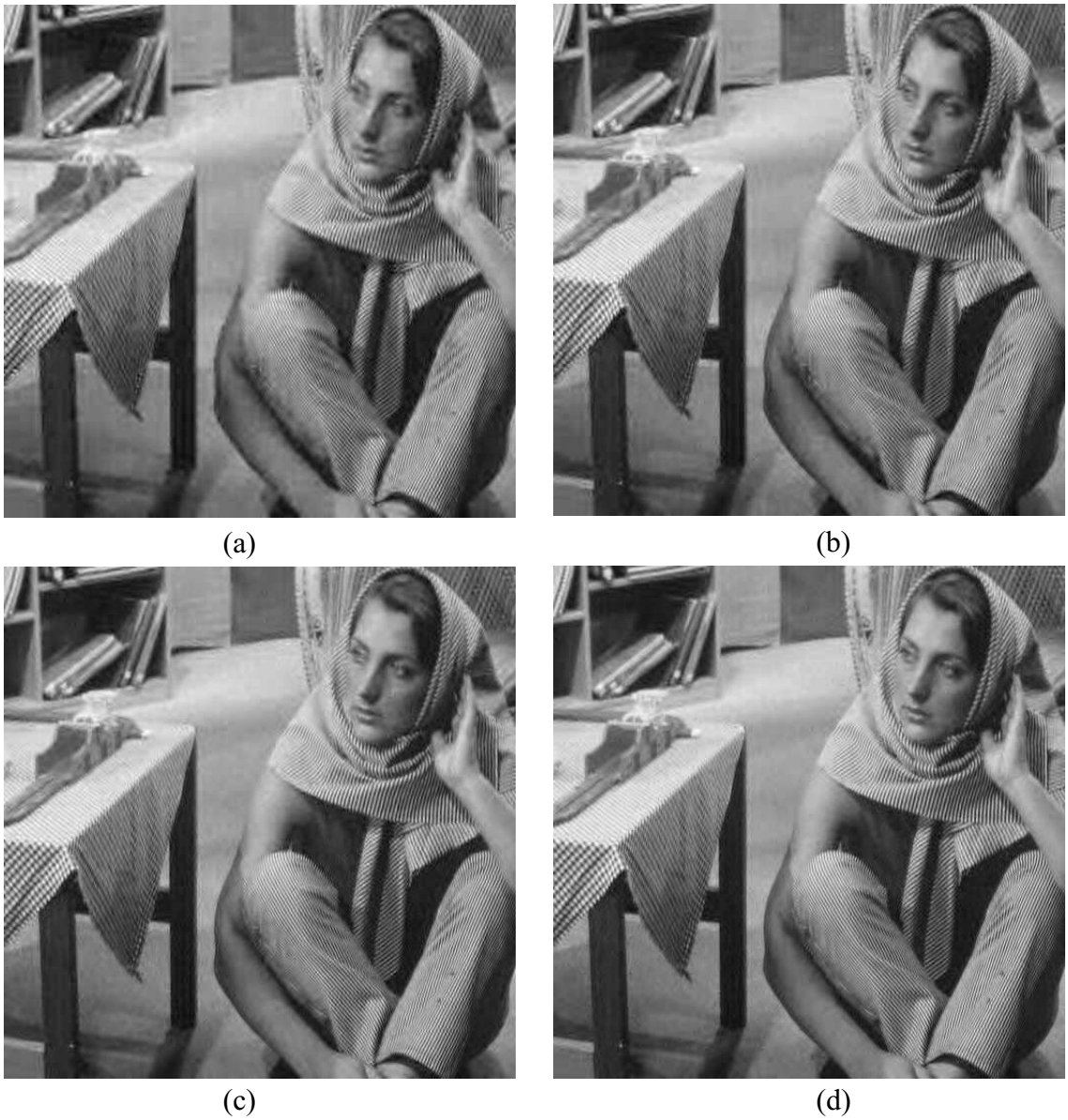


Figura 5.14: Comparação da codificação da Barbara 512x512 para diferentes taxas utilizando ACTTCQ-SBC. (a) 0.250bpp (b) 0.375bpp (c) 0.500bpp (d) 0.625bpp.

Capítulo 6

Conclusão

Um codificador de imagem, ACTTCQ-SBC (*Arithmetic Coded Turbo Trelis Quantization-SubBand Coding*), foi proposto no trabalho, utilizando técnicas de transformação da imagem em sub-bandas, como a transformada *wavelet*, e a turbo quantização codificada por treliças, com codificação aritmética dos códigos gerados. Um algoritmo de alocação de bit foi implementado baseado na modelagem estatística de cada sub-banda. As fontes foram modeladas como um conjunto restrito de distribuições de gaussianas generalizadas. O modelo da sub-banda define o passo de quantização, o número de níveis de reconstrução e o modo de operação dos níveis (com ou sem sobreposição do nível de reconstrução "0"), gerando uma tabela que é pré-computada no codificador e no decodificador. Definida a taxa de operação, o codificador configura o quantizador de acordo com a tabela pré-computada. A codificação aritmética adaptativa foi implementada através da mudança da tabela de probabilidades, de acordo com o símbolo a ser quantizado. Os centróides no receptor também eram pré-armazenados para o conjunto de modelos de sub-bandas. A tabela foi gerada a partir dos cálculos de distorção do ACTCQ-SBC utilizando um codificador HOVA. Os resultados mostram que o ACTTCQ-SBC não oferece ganhos de codificação com esta configuração, tendo um desempenho pior que o ACTCQ-SBC em 0.2dB, apesar da TTCQ ser superior a TCQ, como podemos ver em José Fernando [2] e em algumas figuras ilustradas durante o trabalho. A complexidade computacional da ACTTCQ-SBC aumentou muito em relação a ACTCQ-SBC. Outro fato que limita o desempenho é o tamanho do bloco no turbo quantizador para cada modelo de GGD. No estudo de [2] o tamanho do bloco ótimo do

turbo quantizador varia muito de acordo com o modelo da fonte. Otimizações nas curvas operacionais de taxa-distorção precisam ser feitas para que possamos esperar algum ganho no ACTTCQ-SBC. Essas novas curvas devem levar em consideração o tamanho de bloco ótimo para cada modelo estatístico das sub-bandas, e aplicarem o algoritmo SOVA.

6.1 Próximos Trabalhos

Pelos resultado obtidos, o primeiro trabalho é gerar novas tabelas para o quantizador TTCQ que levem em consideração o tamanho de bloco ótimo, de entrada do turbo quantizador, para cada tipo de fonte. Em conjunto com essa ponderação, devemos utilizar um sistema SISO para calcular as distorções do quantizador, com isso melhoramos a sensibilidade de nossos resultados. Aumentar o número de iterações, no turbo quantizador, também é uma outra proposta para verificarmos o ganho de codificação, mesmo que marginal. Durante as simulações, percebemos um grande índice de modelos de GGD que não estavam mapeados no conjunto restrito, definido no capítulo 3. Isto ocasionava uma perda de fidelidade na modelagem da sub-banda. Sugerimos que este conjunto de modelos seja aumentado. O algoritmo proposto somente se preocupa no modelo probabilístico intra-banda, ou seja, não existe nenhuma proposta de retirar a redundância existente inter-bandas. Como verificado nas figuras apresentadas durante o trabalho, e na literatura, existem áreas com grande dependência inter-bandas que podem ser exploradas, especialmente áreas de contorno dos objetos na imagem. Baseado nesta análise, uma última proposta de trabalho é aplicar a classificação de classes (blocos) dentro de uma sub-banda. Com isso conseguimos otimizar o resultado de quantização para cada classe, pois levamos em conta a característica não estacionária da sub-banda, conforme os estudos realizados em [27] [30] [31].

Referências Bibliográficas

[1] SAYOOD, K., *Data Compression*. 1 ed. New York, USA, John Wiley and Sons, Inc., 1991.

[2] OLIVEIRA, J. F. L. D., *Modulação Codificada e Compressão de Sinais*. Tese de doutorado, COPPE/URFJ, Dezembro 2003.

[3] COVER, T. M., THOMAS, J., *Elements of Information Theory*. New York, USA, John Wiley and Sons, Inc., 1991.

[4] BERROU, C., GLAVIEUX, A., HIATIMASHIMA, P., "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", *IEEE ICC 1993 - IEEE International Conference on Communication*, , 1993.

[5] ROBERTSON, P., WA-RZ, T., "Bandwidth-Efficient Turbo Trellis Coded Modulation Using Punctured Component Codes", *IEEE Journal on Selected Areas in Communication*, , 1998.

[6] MARCELLIN, M. W., "Trellis Coded Quantization: An Efficient Technique for Data Compression", *Ph.D. dissertation*, , 1987.

[7] MARCELLIN, M., GORMISH, M., BILGIN, A., *et al.*, "An Overview of JPEG-2000". In: *Data Compression Conference 2000*, pp. 523–541, March 2000.

[8] ISO/IEC-JTC1, *ISO/IEC 13818-2 - Generic Coding of Moving Pictures and Associated Audio Information: Video*. ISO, 1996/Rev.2000.

[9] JAIN, A. K., *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[10] MALLAT, S., "A Theory for Multiresolution Signal Decomposition: the Wavelet Represntation", *IEEE Trans. on Pattern Anal., and Machine Intell.*, v. 11, pp. 674–693, Junho 1989.

[11] JOSHI, R. L., CRUMP, V. J., FISCHER, T. R., "Image Subband Coding using Arithmetic Coded Trellis Coded Quantization", *IEEE Trans. on Circuits and Systems for Video Technology*, v. 5, n. 6, pp. 515–523, Dezembro 1995.

[12] JOSHI, R. L., FISCHER, T. R., "Comparisson of Generalized Gaussian and Laplacian Modeling in DCT Image Coding", *IEEE Signal Processing Letters*, v. 2, n. 5, pp. 81–82, Maio 1995.

[13] BELL, T. C., CLEARY, J. G., WITTEN, I. H., *Text Compression*. Prentice Hall, 1990.

[14] WICKER, S. B., *Error Control Systems for Digital Communication and Storage*. Prentice Hall, 1995.

[15] SHANNON, C. E., "A Mathematical Theory of Communication", *Bell System Technical Journal*, , 1948.

[16] BUSSAB, W. O., MORETTIN, P. A., *Estatística Básica*. 4 ed. Atual Editora, 1987.

[17] ORTEGA, A., RAMCHANDRAN, K., "Image and Video Compression", *IEEE Singal Processing Magazine*, pp. 23–50, Novembro 1998.

[18] DINIZ, P. S. R., SILVA, E. A. B. D., NETTO, S. L., *Digital Signal Processing - System Analysis and Design*. Cambridge University Press, 2002.

[19] VAIDYANATHAN, P. P., *Multirate Systems and Filter Banks*. Prentice Hall, 1995.

[20] JOHNSTON, J. D., "A Filter Family Designed for Use in Quadrature Mirror Filter Banks". In: *IEEE ICASSP - International Conference on Acoustics, Speech and Signal Processing*, pp. 291–294, Abril 1980.

[21] HAYKIN, J., *Digital Communication*. 2 ed. Prentice Hall, Outubro 1997.

[22] UNGERBOECK, G., "Channel Coding with Multilevel/Phase Signals", *IEEE Transactions on Information Theory*, v. IT-28, n. 1, pp. 55–67, January 1982.

[23] JOSHI, R. L., *Subband Image Coding using Classification and Trellis Coded Quantization*. Ph.D. dissertation, Washington State University, Agosto 1996.

[24] COHEN, A., DAUBECHIES, I., FEAUVEAU, J., "Biorthogonal bases of compactly supported wavelets", *Commun. Pure Appl. Math.*, v. 45, pp. 485–560, Junho 1992.

[25] WESTRINK, P. H., BIEMOND, J., BOEKEE, D. E., "An optimal bit allocation algorithm for sub-band coding". In: *Int. Conf. on Acust., Speech and Signal Proc.*, pp. 757–760, Abril 1988.

[26] SHAPIRO, J. M., "Embedded image coding using zerotrees of wavelet coefitients", *IEEE Trans. Signal Process.*, v. 41, pp. 3445–3462, Dezembro 1993.

[27] JOSHI, R. L., FISCHER, T. R., BAMBERGER, R. H., "Optimum Classification in subband coding of images". In: *IEEE International Conference Image Processing*, v. 2, pp. 883–887, Novembro 1994.

[28] BAHL, L. R., COCKE, J., JELINEK, F., "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", *IEEE Transaction on Information Theory*, v. IT-20, n. 2, pp. 248–287, Março 1974.

[29] SAID, A., PEARLMAN, W., "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", *IEEE Trans. Circuits and Systems for Video Technology*, v. 6, pp. 243–250, Junho 1996.

[30] JOSHI, R. L., JAFARKHANI, H., KASNER, J., *et al.*, "Comparison of Different Methods of Classification in Subband Coding of Images", *IEEE Trans. on Image Processing*, v. 6, pp. 1473–1486, Novembro 1997.

[31] YOO, Y., ORTEGA, A., YU, B., "Image Subband Coding Using Context-Based Classification and Adaptive Quantization", *IEEE Trans. on Image Processing*, v. 8, pp. 1702–1715, Dezembro 1999.