Codificação de Imagem e Vídeo com Casamento Parcial de Padrões em Transformadas de Bloco

Marcos Ricardo Alcântara Morais

Tese de Doutorado submetida à Coordenação dos Cursos de Pós-Gradução em Engenharia Elétrica da Universidade Federal da Paraíba - Campus II como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Elmar Uwe Kurt Melcher Dr.
Orientador

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

M827c
2004 Morais, Marcos Ricardo Alcântara.
Codificação de imagem e vídeo com casamento parcial de padrões em transformadas de bloco / Marcos Ricardo Alcântara Morais.- Campina Grande, 2004.
133f.

Tese (Doutorado em Engenharia Elétricas) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Referências.
Orientador : Prof. Dr. Elmar Uwe Kurt Melcher.

1. Codificação de vídeo. 2. Processamento Digital de Sinais. 3. Transformadas de Bloco. I. Título.

CDU – 621.397.45:004.383.3(043)

CODIFICAÇÃO DE IMAGEM E VÍDEO COM CASAMENTO PARCIAL DE PADRÕES EM TRANSFORMADAS DE BLOCO

MARCOS RICARDO DE ALCÂNTARA MORAIS

Tese Aprovada em 24.09.2004

ELMAR UWE KURT MELCHER, Dr., UFCG
Orientador

WEILER ALVES FINAMORE, Dr., CETUC-PUC-Rio
Componente da Banca

EDNA NATIVIDADE DA SILVA BARROS, Dr., UFPE
Componente da Banca

FRANCISCO MARCOS DE ASSIS, Dr., UFCG
Componente da Banca

FRANCISCO MADEIRO BERNARDINO JUNIOR, D.Sc., UNICAP
Componente da Banca (Ausência Justificada)

JOÃO MARQUES DE CARVALHO, Ph.D., UFCG
Componente da Banca

CAMPINA GRANDE – PB
SETEMBRO - 2004

Codificação de Imagem e Vídeo com Casamento Parcial de Padrões em Transformadas de Bloco

Marcos Ricardo Alcântara Morais

Tese de Doutorado apresentada em Outubro de 2004

Elmar Uwe Kurt Melcher Dr.

Orientador

Edna Natividade da Silva Barros, Dr. rer. nat., UFPE

Componente da Banca

Francisco Madeiro Bernardino Junior, Dr., UNICAP

Componente da Banca

Francisco Marcos de Assis, Ph.D., UFPB

Componente da Banca

João Marques de Carvalho, Ph.D., UFPB

Componente da Banca

Weiler Alves Finamore, Ph.D., PUC-RJ

Componente da Banca

Campina Grande, Paraíba, Brasil, Outubro de 2004

Dedicatória

Dedico este trabalho à minha esposa Flávia e
a meus filhos Gabriel, Lívia e Letícia.

Agradecimentos

Gostaria de expressar meus sinceros agradecimentos a algumas pessoas e instituições que contribuíram para a realização deste trabalho.

A Deus, por tudo.

À minha esposa Flávia, por seu amor e carinho. Sua força e paciência me guiam.

A meus filhos, que, mesmo sem a total compreensão, deram o estímulo necessário para enfrentar a jornada.

Aos meus pais, pelo apoio inestimável.

A todos os meus irmãos, por sempre me apoiarem e incentivarem.

Ao professor Elmar Melcher, pela orientação do trabalho e por sua amizade.

Aos professores e amigos Ângelo Perkusich, Antonio Marcus e todos os demais professores do DEE.

Aos funcionários da COPELE, Ângela e Pedro, pela presteza.

Resumo

Esta tese de doutorado objetiva o desenvolvimento e a avaliação de uma técnica eficiente para a codificação sem perda de coeficientes quantizados, oriundos da aplicação de transformadas de bloco, aplicados a compressão de imagem e vídeo. Para tanto, empregamos a técnicas de casamento parcial de padrões (CPP) como pre-processamento da etapa de codificação entrópica de codificadores de imagem e vídeo baseados em transformada. Para imagem utilizamos os blocos já codificados como dicionário de busca. No caso de vídeo, analisamos a compressão nas imagens de resíduo obtidos pelos processos de estimação e compensação de movimento, utilizando as imagens de resíduo anteriores na formação do dicionário. O algoritmo de casamento parcial de padrões baseia-se no fato de haver correlação residual entre os mapas de significância dos blocos de coeficientes transformados. Os blocos similares servem como referência para a codificação ao fornecer um mapa com probabilidades para a codificação do resíduo obtido. Para a implementação destes experimentos foram feitas modificações de um codificador MPEG-4 existente para inclusão destes algoritmos para a realização de experimentos para análise dos méritos da técnica e também dos resultados objetivos. Alguns fatores como os tipos de dicionários, os métodos de busca e as métricas a serem utilizadas na definição da distância a ser minimizada são discutidos. Apresentamos com isso algumas contribuições para o problema da codificação de coeficientes na compressão de imagem e vídeo.

Abstract

This doctorate dissertation aims at the development and evaluation of efficient techniques for lossless coding of quantized block transform coefficients, applied to image and video compression. For that, we use the partial pattern matching technique as a preprocessing stage in entropy coding for transform based image and video coders. For image coding we use the already coded blocks as a dictionary. For video we analyze the compression of residue images after the motion estimation and compensation steps, using the previous residue transformed elements in the dictionary formation. The partial pattern matching algorithm is based on the fact that there is residual correlation among the blocks of transformed coefficients. The similar blocks act as references during the coding by providing a probability map for the residue taken. For the implementation of those experiments we have made some modifications of an existing MPEG-4 coder to include those algorithms to make experiments for analysis of the technique merits and also of the objective results obtained. Some factors as dictionaries types, searching methods and the metrics to be used in the definition of distances to be minimized are discussed. With that we present some contributions to the problem of coefficient coding in image and video compression.

Sumário

Lista de Símbolos e Abreviaturas

KLT	transformada de Karhunen-Loève (*Karhunen-Loève transform*)
DWT	transformada *wavelet* discreta (*discrete wavelet transform*)
IDWT	transformada *wavelet* discreta inversa (*inverse discrete wavelet transform*)
DCT	transformada discreta de cosseno (*discrete cosine transform*)
LT	transformada com superposição (*lapped transform*)
LOT	transformada com superposição ortogonal (*lapped orthogonal transform*)
LBT	transformada com superposição biortogonal (*lapped biorthogonal transform*)
genLOT	transformada com superposição ortogonal generalizada (*generalized LOT*)
gLBT	transformada com superposição biortogonal generalizada (*generalized LBT*)
T	operação de transposição de uma matriz, ou ainda, operação de transformada direta
T^{-1}	operação de transformada inversa
SPIHT	particionamento de conjuntos em árvores hierárquicas (*set partitioning in hierarchical trees*)
EZW	árvore de zero embutida *wavelet* (*embedded zerotree wavelet*)
EZC	codificação embutida com árvores de zero (*embedded zerotree coding*)

CEB	codificador entrópico de blocos baseado em contextos (*context based entropy coding of block transform coefficients*)
CABAC	codificador aritmético binário adaptativo com contextos (*context adaptive binary arithmetic coder*)
ARL	codificação adaptativa de comprimento de seqüência (*adaptive runlength coding*)
RL	comprimento de seqüência (*runlength*)
VLC	código de comprimento variável (*variable length coding*)
RL+VLC	codificação com código de comprimento variável de um comprimento de seqüência (*runlength variable length coding*)
MPEG	*Motion Picture Experts Group*
JPEG	*Joint Photographics Experts Group*
JBIG	*Joint Bi-level Image Experts Group*
JBIG2	novo padrão para imagens binárias baseado no JBIG
H.26x	codificadores ITU-T
EOB	marcador de fim de bloco (*end of block*)
L_z	comprimento da seqüência de zeros no JPEG
L_a	número de bits para representar a amplitude no JPEG
A	amplitude de um coeficiente no JPEG
Q	fator de qualidade que multiplica a tabela de quantização no JPEG e MPEG
DjVu	formato de codificação de documentos com texto e figuras
CPP	casamento parcial de padrões
PPM	Predição através do casamento de padrões (*prediction by pattern matching*)
SPQ	SPIHT pós quantização
CBPB	codificador de blocos por planos de bits
TQC	transformada–quantizador–codificador
ECG	eletrocardiograma

SQ	quantização escalar (*scalar quantization*)
VQ	quantização vetorial (*vector quantization*)
MOS	escore médio de opinião (*mean opinion score*)
PQS	Escala de qualidade de imagens (*picture quality scale*)
SNR	relação sinal-ruído (*signal-to-noise ratio*)
PSNR	relação sinal-ruído de pico (*peak signal-to-noise ratio*)
MSE	erro médio quadrático (*mean square error*)
HVS	sistema visual humano (*human visual system*)
YCbCr	representação da imagem em luminância (Y) e crominâncias (Cb) e (Cr)
YUV	sinônimo para YCbCr
RGB	representação da imagem com as cores primárias (*red-green-blue*)
XOR	operação booleana de ou-exclusivo
PM&S	casamento de padrão e substituição (*pattern matching and substitution*)
SPM	casamento suave de padrões (*soft pattern matching*)
TSS	busca em três passos (*three step search*)
FSS	busca em quatro passos (*four step search*)
S	fonte aleatória de símbolos
$\mathscr{A}$	alfabeto de símbolos
s_i	i-ésimo símbolo
w_i	i-ésima palavra binária associada com s_i
l_i	comprimento da i-ésima palavra binária
$H(S)$	entropia da fonte S
p_i	probabilidade de ocorrência de s_i
$I(s_i)$	quantidade de informação para s_i
E	valor esperado
ISDN	redes digitais de serviços integrados (*integrated services digital networks*)

N	tamanho do bloco (número de coeficientes), ou número de símbolos de uma fonte aleatória
$d(x, w_i)$	distorção/distância entre os vetores x e w_i
R	taxa de codificação
bpp	bits por *pixel*
1-D	unidimensional
2-D	bidimensional
$x(n)$	sinal de entrada ou ainda, simplesmente, sinal original
$a_i(n)$	componente de aproximação (obtida por meio de DWT) no i-ésimo nível de resolução
$d_i(n)$	componente de detalhe (obtida por meio de DWT) no i-ésimo nível de resolução
$y(n)$	sinal de saída
$\mathscr{L}$	número de níveis de decomposição (número de níveis de resolução) da DWT
$h_0(n)$	filtro *wavelet* passa-baixa de análise
$h_1(n)$	filtro *wavelet* passa-alta de análise
$g_0(n)$	filtro *wavelet* passa-baixa de síntese
$g_1(n)$	filtro *wavelet* passa-alta de síntese
L	filtro passa-baixa
H	filtro passa-alta
HL_1	sub-banda correspondente à direção horizontal do 1º nível de resolução, também denotada por S_{11}
LH_1	sub-banda correspondente à direção vertical do 1º nível de resolução, também denotada por S_{12}
HH_1	sub-banda correspondente à direção diagonal do 1º nível de resolução, também denotada por S_{13}
HL_2	sub-banda correspondente à direção horizontal do 2º nível de resolução, também denotada por S_{21}

LH_2 sub-banda correspondente à direção vertical do $2^{\underline{o}}$ nível de resolução, também denotada por S_{22}

HH_2 sub-banda correspondente à direção diagonal do $2^{\underline{o}}$ nível de resolução, também denotada por S_{23}

HL_3 sub-banda correspondente à direção horizontal do $3^{\underline{o}}$ nível de resolução, também denotada por S_{31}

LH_3 sub-banda correspondente à direção vertical do $3^{\underline{o}}$ nível de resolução, também denotada por S_{32}

HH_3 sub-banda correspondente à direção diagonal do $3^{\underline{o}}$ nível de resolução, também denotada por S_{33}

LL_3 sub-banda correspondente à componente de aproximação (resultante de filtragens passa-baixa) do $3^{\underline{o}}$ nível de resolução, também denotada por S_{30}

S_{ij} sub-banda correspondente à j-ésima direção ($j = 1$ denota direção horizontal, $j = 2$ denota direção vertical, $j = 3$ denota direção diagonal e $j = 0$ denota direção LL) do i-ésimo nível de resolução (nível de decomposição)

$F(l,c)$ valor de *pixel* referente à l-ésima linha e c-ésima coluna da imagem original

$\hat{F}(l,c)$ valor de *pixel* referente à l-ésima linha e c-ésima coluna da imagem reconstruída

AR(1) fonte autoregressiva de $1^{\underline{a}}$ ordem

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Para reduzir custos de transmissão e armazenamento, a informação contida em sinais originais digitalizados pode ser de alguma forma comprimida, extraindo-se os elementos distintivos e eliminando-se os elementos menos importantes, a despeito de uma representação imperfeita.

Um objetivo fundamental da compressão de imagem e vídeo é reduzir o número de bits necessários para transmissão ou armazenamento mantendo uma fidelidade ou qualidade de imagem ou vídeo aceitável, de acordo com o seu destino. A compressão de imagem e vídeo desempenha um papel importante em aplicações tais como: sistemas multimídia, videoconferência, armazenamento de filmes em disco, televisão digital por assinatura, televisão de alta definição, envio e recepção de imagens e vídeo pela internet, sistemas de armazenamento de imagens médicas e de impressões digitais e transmissão de imagens de sensoriamento remoto obtidas por satélites. A compressão pode ser obtida através de diversas técnicas, destacando-se por sua ampla utilização a transformação dos dados através da projeção em funções de base, uma transformação linear, seguida pelo processo de quantização e pela codificação sem perda dos coeficientes transformados e quantizados.

As transformadas de bloco, como a Transformada Discreta Cossenoidal (DCT, *Discrete Cosine Transform*) e as Transformadas com Sobreposição (LT (*Lapped Transform*), que realizam o processamento atuando em pequenas regiões da imagem, têm encontrado

aplicação freqüente na compressão de imagem, porque apresentam reduzida complexidade computacional com eficiência na compressão [50, 119] [18, Capítulo 5].

Em um sistema de compressão de imagem ou vídeo baseado em transformada de bloco, os coeficientes obtidos pela transformação da imagem de entrada são quantizados e os valores obtidos passam por uma codificação sem perda também chamada codificação entrópica. Para obter-se os melhores resultados deve-se buscar a otimização de todos os passos do sistema. A transformada deve ser adequada ao sinal a ser codificado, concentrando as informações essenciais do sinal em poucos coeficientes significativos, de forma que, ao se desprezar ou quantizar de forma mais grosseira os demais coeficientes, a representação obtida ainda seja bastante fiel. A quantização deve ser projetada de forma a aproveitar a estatística dos grupos de coeficientes transformados. O papel da codificação entrópica é diminuir a redundância dos símbolos emitidos pelo quantizador.

Nesta tese investigamos uma técnica de codificação de vídeo baseado em casamento parcial de padrões. O principal objetivo é o desenvolvimento e a avaliação de técnicas eficientes para a codificação sem perda de coeficientes quantizados, oriundos da aplicação de transformadas de bloco, aplicados a compressão de imagem e vídeo. A técnica utilizada foi denominada de *casamento parcial de padrões*, ou CPP. Além desta técnica analisada com mais profundidade neste documento, outros estudos realizados até o momento resultaram na concepção de novas técnicas para a codificação de coeficientes quantizados em transformadas de bloco. A primeira, denominada SPQ (*SPIHT pós quantização*), baseia-se no reagrupamento em sub-bandas sugerido inicialmente por Xiong et.al. [126], que aplicou o algoritmo de codificação baseado em *wavelet* SPIHT [96] diretamente sobre um rearranjo dos coeficientes DCT. A nossa abordagem difere deste trabalho por não utilizar a quantização intrínseca do algoritmo SPIHT, aplicando a codificação aos elementos já quantizados. A separação entre quantização e codificação no SPQ permite a utilização de quantizadores otimizados, como o de Loyd-Max, podendo levar a ganhos na codificação. A segunda técnica, denotada por CBPB (*Codificação de Blocos por Planos de Bits*), trabalha com os mapas binários relativos a cada plano de bit de cada bloco individualmente. O CPP é o foco deste documento e é visto no texto principal. As outras técnicas, SPQ e

CBPB, são vistas nos apêndices.

Estas técnicas de codificação são avaliadas por meio de simulações envolvendo codificação de imagens e vídeo em um arcabouço de um codificador MPEG-4, modificado para acomodar estes algoritmos. A codificação de vídeo natural utilizada no padrão MPEG-4 é baseada na corrida de zeros, RL (*Run Length*) seguida por um código de comprimento variável, VLC (*Variable Length Coding*), com tabelas pré-estabelecidas, ou seja, sem adaptação. Denotaremos esta técnica como RL+VLC.

No presente trabalho, estudamos o uso do casamento parcial de padrões como pré-processamento da etapa de codificação entrópica de codificadores de imagem e vídeo baseados em transformada. No caso de vídeo, analisamos a compressão nas imagens naturais e nas imagens de resíduo, obtidas pelos processos de estimação e compensação de movimento.

Este trabalho contempla uma avaliação comparativa de desempenho dos algoritmos CPP, SPQ, CBPB e RL+VLC, separadamente. Esta avaliação é realizada primariamente no aspecto da taxa de bits obtida, haja visto que por se tratar de métodos de compressão sem perda, os demais aspectos, entre eles a qualidade dos sinais reconstruídos, são mantidos. Alguns outros aspectos são considerados, tais como a complexidade computacional envolvida nas fases de codificação e de decodificação, a capacidade de adaptação e a adequação a sinais de outras origens.

Os resultados obtidos são importantes e sugerem que, com algumas otimizações do método, seja possível obter uma redução ainda mais significativa do número de bits utilizados na codificação sem perda dos coeficientes transformados.

Em seguida veremos como o trabalho está organizado.

1.1 Organização do Trabalho

A presente tese é organizada em 8 capítulos e 2 apêndices.

No Capítulo 2 é apresentada uma visão geral de compressão de imagem, abordando conceitos fundamentais como codificação de fonte e abrangendo as técnicas mais convencionais de compressão de imagem. Por se tratar de um dos métodos mais amplamente uti-

lizados para codificação de imagens, o paradigma Transformada-Quantização-Codificação é descrito. Cada um desses elementos é abordado no capítulo. Analisamos alguns aspectos da utilização de transformadas lineares para codificação de sinais, com respeito ao ganho de codificação e à complexidade computacional. São apresentados alguns fundamentos das classes de transformadas mais usuais, que sejam, as transformadas de bloco, representadas pela DCT e pelas transformadas com superposição, e a transformada *wavelet* discreta (DWT, *discrete wavelet transform*), que opera sobre toda a imagem. Mais detalhes de um codificador baseado em *wavelet* são vistos no apêndice A. Em se tratando do processo de quantização, vemos rapidamente a quantização escalar e os princípios da quantização vetorial. Revisamos alguns aspectos da codificação entrópica, com as duas importantes classes de código, código de Huffman e código aritmético. Vemos padrão JPEG como exemplo de codificador de imagem baseado em DCT e codificação entrópica baseada em seqüência de zeros e código de comprimento variável. O capítulo é encerrado com uma breve avaliação de desempenho de codificadores de imagem, sendo abordadas metodologias de avaliação da distorção obtida quando realizamos codificação com perda.

O Capítulo 3 trata da codificação de vídeo. Apenas as técnicas usadas neste trabalho são tratadas. Iniciamos mostrando o processo básico da codificação de vídeo, como a exploração das redundâncias espacial e temporal. Vemos como exemplo de codificador de vídeo o padrão MPEG-4, que permite a codificação de elementos individuais da cena, através do conceito de objetos. No entanto, no presente trabalho nos atemos apenas à codificação de vídeo natural, onde há apenas um objeto de formato retangular, que é a própria seqüência de imagens.

Introduzimos no Capítulo4 a codificação baseada na separação entre coeficientes nulos e não nulos através de um mapa de significância. Realizamos um experimento de codificação utilizando um algoritmo simples para observarmos a proporção de bits gastos para codificar a significância, o sinal e a amplitude dos coeficientes. Neste algoritmo cada uma das partes (significância, sinal e amplitude) é codificada diretamente com um codificador aritmético baseado em contexto.

No Capítulo 5 é apresentada a técnica de casamento de padrões, que denotamos como

CPP, aplicada à codificação entrópica de coeficientes transformados e quantizados para compressão de imagem e vídeo. É realizada uma introdução com motivações para a escolha da técnica, através de testes que demonstram que há uma correlação residual entre os blocos transformados, principalmente após o passo de quantização. A técnica de casamento parcial de padrões baseia-se no aproveitamento desta correlação para obter os seus ganhos. Baseando-se nos resultados obtidos no capítulo anterior sobre a importância para a codificação da significância dos coeficientes transformados, finalizamos o capítulo com uma simplificação da técnica de CPP aplicada apenas aos mapas binários de significância.

Dedicamos o Capítulo 6 à implementação da técnica de casamento de padrões no arcabouço do codificador MPEG-4. Os detalhes da sua implementação em *software* e as estruturas de dados utilizadas são mostrados. O codificador aritmético foi incluído no MPEG-4 como parte do processo de adequação. O codificador utilizado foi o mesmo do padrão H.264/AVC/MPEG-4 parte 10, denominado CABAC.

O Capítulo 7 apresenta os resultados obtidos com a aplicação do CPP no codificador MPEG-4 para algumas seqüencias de vídeo utilizadas normalmente na literatura. Os resultados obtidos são comparados com a codificação entrópica originalmente encontrada no padrão MPEG-4. A implementação permite também que o casamento de padrões seja seletivamente ligada e desligada. Desta forma, foi também possível comparar os resultados com os obtidos apenas com a substituição do codificador original do MPEG-4 pelo CABAC do H.264.

No Capítulo 8 as contribuições científicas pertinentes são apresentadas juntamente com as conclusões do trabalho. Os possíveis trabalhos de continuação desta pesquisa são citados.

Temos dois apêndices neste trabalho que tratam de outros métodos de codificação de coeficientes transformados em um contexto de codificação de vídeo. Estes trabalhos servem como referencia de outras técnicas que podem ser utilizadas de separadamente ou em conjunto com a técnica de casamento parcial de padrões. Outro objetivo da inclusão neste documento é para comparação objetiva dos valores obtidos na codificação.

No ApêndiceA é apresentado o codificador SPQ, ou SPIHT pós-quantização. Trata-se da aplicação do renomado codificador SPIHT [96] à codificação dos coeficientes transfor-

mados de um codificador de vídeo MPEG-4. A principal diferença entre esta abordagem e a usual é que esta aplicação ocorre após a quantização já existente no MPEG-4. Desta forma é possível realizar apenas a substituição da etapa de codificação entrópica. O codificador SPIHT é originalmente aplicado sobre coeficientes *wavelet* de uma imagem. A codificação baseada em *zerotrees* (árvores de zeros) usada no SPIHT é mostrada. São vistos então detalhes deste codificador como também das alterações realizadas para que possa ser utilizado em um ambiente de codificação de vídeo baseado em transformadas de bloco. Concluímos este apêndice como resultados objetivos, comparando o número de bytes utilizados com esta codificação proposta e o codificador MPEG-4 original, para diversas seqüências normalmente encontradas na literatura.

No Apêndice B vemos uma formulação alternativa para a codificação entrópica dos coeficientes DCT quantizados na codificação de vídeo através do uso de planos de bits. A codificação é realizada sobre os blocos individuais de coeficientes DCT, sem reagrupamento. Um algoritmo de codificação de planos de bits, baseado no uso de codificação aritmética de contextos, é aplicado aos planos de bits destes coeficientes, em substituição ao processo de varredura e corrida de zeros, com código de comprimento variável encontrado na codificação de vídeo natural do padrão MPEG-4. Esta codificação foi inserida em um codificador MPEG-4. Observou-se uma redução no número de bytes utilizados na codificação da textura nas seqüências com texturas complexas, apesar de haver um aumento no número de bytes utilizados para codificar seqüências consideradas simples.

Capítulo 2

Codificação de imagem

A compressão ou codificação com perdas de uma imagem digital objetiva encontrar uma representação que utilize menos espaço de armazenamento e menos requisitos de transmissão do que a imagem original e que possa ser decodificada para uma imagem similar a original. Neste capítulo revisaremos alguns aspectos da compressão de imagens, com enfoque na codificação por transformada. Também discutiremos alguns aspectos das métricas utilizadas na medição da diferença existente entre a imagem original e imagem comprimida, chamada de distorção. Em particular observaremos com mais detalhes como é realizada a compressão no padrão JPEG.

2.1 Imagens digitais

Uma imagem pode ser entendida como qualquer função $f(x,y)$ que resulte em um vetor de valores reais que representam a cor da posição representada pelos números reais x e y. Uma imagem digital é uma aproximação discreta, tanto em posição quanto em tonalidade (crominância) e amplitude (luminância), da imagem base. Assim, uma imagem digital monocromática (apenas o valor de luminância por pixel) é uma matriz finita de inteiros de tamanho fixo. Podemos também dizer que uma imagem digital é um sinal bi-dimensional discreto.

Neste trabalho, quando da compressão de imagens paradas, utilizaremos normalmente imagens monocromáticas em tons de cinza, tipicamente com 512 por 512 pixels, cada um originalmente codificado com 8 bits. As técnicas de compressão de imagens monocromáticas podem, em geral, ser aplicadas a imagens coloridas, considerando-se mais bits por pixels ou realizando a compressão em vários planos. A separação em planos será vista com mais detalhes quando tratarmos do codificador de imagem JPEG.

2.2 Introdução à compressão de imagem

Nesta seção apresentamos um resumo de elementos comuns à compressão de imagem. Para um aprofundamento neste tema existem muitas referências na literatura [42, 83].

A compressão de imagem recai sobre duas características relativas ao conjunto de pixels de uma imagem: a redundância e a irrelevância [45]. A redundância está relacionada com as propriedades estatísticas das imagens, ao passo que a irrelevância está relacionada com as características do "observador" que utiliza a imagem. A redundância pode ser espacial (devido à correlação entre pixels vizinhos) ou espectral (devido a correlação entre planos de cor ou bandas espectrais). A irrelevância resulta da menor sensibilidade do sistema visual humano (*human visual system*, HVS) ou do algoritmo de processamento a alguns detalhes da imagem, incluindo o ruído introduzido no processo de aquisição da imagem e/ou introduzido pela quantização.

Uma técnica de compressão amplamente utilizada baseia-se no processo de decompor os dados da imagem de forma a remover a redundância e ordenar os dados em ordem de relevância. Esta decomposição é normalmente realizada pela transformação da imagem para outro domínio. Esta técnica denomina-se de codificação por transformada e é vista com detalhes na Seção 2.3.

2.3 Codificação por transformada

Um codificador de imagem ou de vídeo típico consiste de uma operação de transformação linear, seguida pela quantização dos coeficientes no domínio transformado e a compressão sem perda dos coeficientes quantizados usando um codificador entrópico. Os bits resultantes desta codificação podem ser armazenados ou transmitidos por um canal (que consideraremos sem perdas) para o decodificador, que reconstrói a aproximação da imagem ou da seqüência de imagens baseando-se na informação transmitida.

O papel da transformação é de aglutinar em poucos coeficientes a maior parte da informação referente ao sinal de entrada e procurar ordenar estes coeficientes segundo a sua relevância. Após a transformação, a informação, que está de certa forma mais organizada, é então sujeita à quantização que explora a relevância descartando ou retendo aproximadamente as partes consideradas mais relevantes. O passo de quantização promove a perda de qualidade do sinal ao passo que reduz a entropia da sua representação. Normalmente é na quantização que se define a qualidade e a taxa (número de bits) atingidos por um determinado método de compressão, fixados os demais componentes do codificador. A aplicação do quantizador aos coeficientes transformados produz uma distorção menor do que a aplicação aos componentes do sinal original, para a mesma entropia final [45]. Desta forma, pode-se dizer que a transformada efetivamente faz com que a distorção causada pelo passo de quantização seja menor.

O encadeamento de técnicas de transformação - quantização - codificação forma o paradigma TQC. Este paradigma forma uma base amplamente utilizada na compressão com perda de vários tipos de sinais, como áudio [78, 111], imagem [1, 11, 12, 80, 119], vídeo [39, 50], ECG [4], etc.

Os passos referentes ao paradigma TQC são mostrados na Figura 2.1.

Estes passos são altamente interdependentes. Por exemplo, a codificação dos símbolos pode ser feita de maneira muito mais eficiente através de técnicas de codificação entrópica, como o código de Huffman, se a transformação e quantização produzirem uma representação de baixa entropia.

A transformada não causa perdas nem realiza compressão, mas é o núcleo do sistema

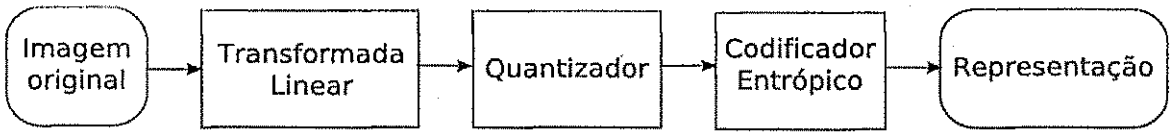


Figura 2.1: Paradigma TQC.

de compressão. Ela favorece a compressão na medida em que compacta a energia do sinal em poucos coeficientes, efetivamente concentrando a informação do sinal em poucos elementos. As transformadas freqüentemente utilizadas na compressão de imagens e vídeo serão vistas na seção seguinte.

Este trabalho é focado no passo de codificação sem perda dos coeficientes quantizados provenientes de transformada de bloco. Algumas técnicas encontradas na literatura e algumas contribuições originais serão vistas no Capítulo 7.

2.4 Transformadas lineares para compressão de imagens e vídeo

Um dos principais objetivos a serem buscados na escolha da transformada linear a ser utilizada em um sistema de compressão de imagem ou vídeo é a capacidade de aglutinação da energia do sinal em poucos coeficientes. Uma medida desta capacidade, denominada ganho de codificação, é sugerida por Jayant e Noll [45]. O ganho de codificação é definido através de um modelo simplificado de codificador onde apenas os primeiros coeficientes do vetor transformado são mantidos (os demais são zerados) e quantizados uniformemente, sendo calculada a sua entropia para uma classe de sinais específica, segundo o modelo auto-regressivo de ordem 1, AR(1). O uso do ganho de codificação como indicativo do poder de compressão de uma transformada fica comprometido se estas condições não forem obedecidas. Muito embora seja comum encontrarmos a modelagem de imagem como AR(1), este modelo é normalmente incapaz de prever a diversidade e a riqueza

de características encontradas nas imagens. Para sinais AR(1), a transformada ótima, no sentido de possuir o maior ganho de codificação possível, é a KLT (*Karhunen-Loève Transform*). A KLT tem a desvantagem de ser dependente do sinal a ser codificado, sendo necessário que seus coeficientes sejam enviados conjuntamente com o sinal codificado para que possa haver a decodificação. Como decorrência a KLT é normalmente substituída pelas transformadas independentes do sinal, como a DCT.

Outra característica desejada em uma transformada linear é a baixa complexidade computacional. Por isso, normalmente são utilizadas as transformadas separáveis. Uma transformada separável é aquela em que a transformação é realizada primeiramente nas linhas e em seguida nas colunas da imagem. Seja $\mathbf{X}$ a matriz contendo os pixels da imagem e $\mathbf{Y}$ a imagem transformada, e $\mathbf{H}_F$ e $\mathbf{H}_I$ as matrizes que representam a transformada unidimensional direta e inversa, respectivamente. A transformada separável direta pode ser escrita como

$$\mathbf{Y} = \mathbf{H}_F \mathbf{X} \mathbf{H}_F^T \tag{2.1}$$

e a sua inversão é

$$\mathbf{X} = \mathbf{H}_I \mathbf{Y} \mathbf{H}_I^T. \tag{2.2}$$

A transformada *wavelet* discreta é um exemplo de transformada separável. Embora possam existir transformadas *wavelet* não separáveis, estas são de difícil utilização para compressão de imagens. Entre as transformadas de bloco destacam-se a DCT, utilizada no padrão JPEG, e as transformadas com superposição (*Lapped Transform*) [17,18,64,67,108, 109]. Em seguida veremos uma descrição da DCT e das transformadas com superposição (*lapped transform*, LT), que são transformadas de bloco. A transformada *wavelet* discreta (*discrete wavelet transform*, DWT) [9,15,58,59,71,102] é muito utilizada na codificação de imagens com resultados entre os melhores da literatura [65,96,107], sendo inclusive utilizada no codificador padrão JPEG2000 [1,11,12]. No entanto, a DWT não é uma transformada de bloco, haja visto que normalmente emprega-se a transformada sobre toda a imagem.

2.4.1 Transformadas de bloco

As transformadas de bloco, como o nome indica, processam o sinal por partes. No caso de imagens, esta é dividida em pequenos blocos retangulares (em geral quadrados e de mesmo tamanho) e cada bloco é transformado individualmente. Isto é motivado pela redução de complexidade computacional. As transformadas de bloco dividem o plano espaço-freqüência da imagem em regiões uniformes, fornecendo resolução espacial fina e constante sobre toda a imagem, mas não revelam características globais. Veremos as principais características da DCT.

Transformada Discreta Cossenoidal

Para imagens, a transformada discreta de cosseno bi-dimensional (DCT-2D) é uma transformada de bloco muito utilizada que forma a base do padrão JPEG [80, 119]. Graças ao seu sucesso no JPEG, a DCT-2D também tem sido adotada por muitos padrões de codificação de vídeo, como nos padrões H.261, H.263 e MPEG [41, 50, 101]. Na Seção 2.8 veremos com mais detalhes o uso da DCT no padrão de codificação de imagem JPEG e na Seção 3.1, seu uso no padrão de codificação de vídeo MPEG-4. Descreveremos agora a base matemática da DCT e mostraremos como ela é aplicada para codificar uma imagem.

Para a aplicação da DCT bi-dimensional, particiona-se a imagem em blocos de tamanhos iguais. A DCT bi-dimensional é uma transformada de bloco linear ortonormal e separável. A aplicação da DCT promove a decomposição de um bloco da imagem em coeficientes que correspondem à projeção dos elementos deste bloco em uma base formada por cossenos de diferentes freqüências espaciais, nas direções vertical e horizontal.

Seja um bloco de imagem $M \times M$ denotado por f, com pixels individuais identificados como $f(i,j)$, com $0 \leq i,j \leq M$. Os coeficientes DCT $F(u,v)$, para a DCT II (tipo de DCT muito comum em codificação de imagem e vídeo), são computados realizando-se o produto interno de f com os blocos básicos para cada (u,v).

$$DCT(f)(u,v) = F(u,v) = \frac{C(u)C(v)}{4} \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} f(i,j) \cos \frac{(2i+1)u\pi}{2M} \cos \frac{(2j+1)v\pi}{2M}. \quad (2.3)$$

As constantes de normalização $C(x)$ são:

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & \text{para } x = 0 \\ 1 & \text{para } x \neq 0. \end{cases}$$

A transformada inversa da DCT (IDCT) de um bloco de coeficientes recuperando-se os pixels da imagem é dada por:

$$IDCT(F)(i,j) = f(i,j) = \frac{1}{4} \sum_{u=0}^{M-1} \sum_{v=0}^{M-1} C(u)C(v)F(u,v) \cos \frac{(2i+1)u\pi}{2M} \cos \frac{(2j+1)v\pi}{2M}. \quad (2.4)$$

O valor de cada coeficiente $F(u,v)$ corresponde a uma faixa de freqüência espacial presente no bloco de imagem. O coeficiente $F(0,0)$ corresponde ao valor médio e é referido como coeficiente DC. Os demais são chamados coeficientes AC. A maioria das imagens naturais é razoavelmente suave, assim a maioria da energia do bloco de imagem é capturada pelos coeficientes de baixa freqüência do bloco transformado. Além disso, o sistema visual humano é menos sensível às freqüências mais altas do que às baixas. Estes fatos podem ser usados para se obter uma alta compressão, mantendo-se apenas uma aproximação de alguns coeficientes através de quantização.

A popularidade do uso da DCT se deve principalmente a sua eficiência computacional. Há muitos algoritmos de cálculo rápido e eficiente da DCT. Estes baseiam-se no desdobramento das multiplicações e no aproveitamento de propriedades trigonométricas [89].

2.4.2 Transformadas com superposição

A DCT possui qualidades que a tornam adequada para a codificação de imagem e vídeo. No entanto, em imagens comprimidas com DCT a taxas baixas é comum o aparecimento do desagradável efeito de bloqueamento, em que as bordas dos blocos são visíveis. Para

resolver este problema, foi desenvolvido o conceito de transformadas com superposição (LT, *lapped transforms*). A idéia é estender as funções de base além das fronteiras do bloco, criando uma sobreposição, a fim de eliminar o efeito de bloqueamento. Um ponto muito importante é que o número de coeficientes da transformada é o mesmo da transformada sem superposição. A transformada ortogonal com superposição (LOT, *lapped orthogonal transform*) foi introduzida por Cassereau [10]. Porém foi Malvar [62,66,67] quem deu um tratamento elegante e um algoritmo rápido de cálculo, tornando a LOT mais interessante para utilizações práticas. Também foi Malvar que indicou a equivalência entre uma LT e um banco de filtros multitaxa [61], um tema muito estudado em processamento de sinais.

Para um aprofundamento do tema, o leitor pode dirigir-se a revisões encontradas na literatura [19,63,64]. Para a uso de compressão de imagens utilizando LTs, uma excelente introdução pode ser encontrada em Queiroz [18].

2.4.3 Transformada *Wavelet* Discreta

O processamento de sinais por sub-bandas desempenha um papel importante em diversos sistemas de codificação imagem, vídeo, além de áudio, voz e outros sinais como ECG.

No processamento de sinais em sub-bandas, um banco de filtros de análise é aplicado ao sinal de entrada, gerando um conjunto de sinais com faixa de freqüências mais estreita, cada um representando uma determinada sub-banda do espectro do sinal de entrada. Pode-se então realizar algum procedimento nas sub-bandas individuais e posteriormente reconstruir o sinal por meio da aplicação de filtros de síntese. No caso específico da codificação, esta também é realizada nas sub-bandas individuais.

A codificação por sub-bandas foi originalmente introduzida no contexto de codificação de voz por Crochiere *et al.* [14]. A extensão da filtragem de sub-bandas de 1-D para 2-D foi apresentada por Vetterli [116], sendo originalmente aplicada à codificação de imagens por Woods *et al.* [120, 124].

Mallat [58, 59] mostrou a equivalência direta entre a decomposição em sub-bandas realizada por um banco de filtros recursivo e a decomposição *wavelet*. Outro resultado desta

equivalência é um algoritmo rápido para cálculo da transformada *wavelet*. Deve-se observar que todo banco de filtros recursivo também pode ser substituído por um banco não recursivo equivalente [102]. Esta teoria possibilitou a integração de técnicas do tratamento de filtros oriundas de processamento de sinais com a teoria das aproximações. A decomposição *wavelet* tem sido amplamente utilizada em vários sistemas de codificação de imagens, permitindo a obtenção de resultados que estão entre os melhores da literatura [65,96,107].

Nas seções a seguir, são apresentados alguns fundamentos da transformada *wavelet* discreta, que opera com sinais discretos e produz coeficientes também discretos.

Transformada *Wavelet* Discreta

A transformada *wavelet* discreta [9, 15, 71, 102] pode ser descrita a partir da recursão de um banco de filtros, conforme mostra a Figura 2.2. Dada a facilidade de tratamento, utiliza-se mais freqüentemente a DWT diádica, ou seja, com apenas 2 filtros simétricos, um passa-baixas e outro passa-altas em cada etapa da recursão. Devido a divisão por 2 a cada fase da recursão, diz-se também tratar-se de uma tratamento por oitavas. Os filtros $h_0(n)$ e $h_1(n)$ correspondem aos filtros *wavelet* de análise, enquanto que $g_0(n)$ e $g_1(n)$ correspondem aos filtros *wavelet* de síntese. Um sinal de entrada $x(n)$ é convoluído com o filtro passa-baixa $h_0(n)$ e com o filtro passa-alta $h_1(n)$. O sinal resultante de cada convolução é então submetido a uma decimação (sub-amostragem) de ordem 2. Geram-se, assim, no processo de decomposição (ou análise), o sinal de aproximação $a_1(n)$ e o sinal de detalhe $d_1(n)$. Em outras palavras, $a_1(n)$ contém os coeficientes *wavelet* correspondentes à componente de aproximação do sinal, ao passo que $d_1(n)$ contém os coeficientes *wavelet* referentes à componente de detalhe do sinal. Observa-se ainda na Figura 2.2(a) a iteração do processo de filtragem, onde o próximo nível de decomposição (resolução mais grosseira) é obtido mediante a convolução do sinal $a_1(n)$ com um par idêntico de filtros $h_0(n)$ e $h_1(n)$, seguida da decimação de ordem 2. Desta forma, são gerados os sinais $a_2(n)$ e $d_2(n)$, em que o índice 2 diz respeito ao segundo nível de decomposição. Este processo pode ser repetido e promove uma decomposição *wavelet* multiresolucional (multinível) [9, 59, 117]. O número de filtragens realizadas determina o número de níveis de decomposição (ou

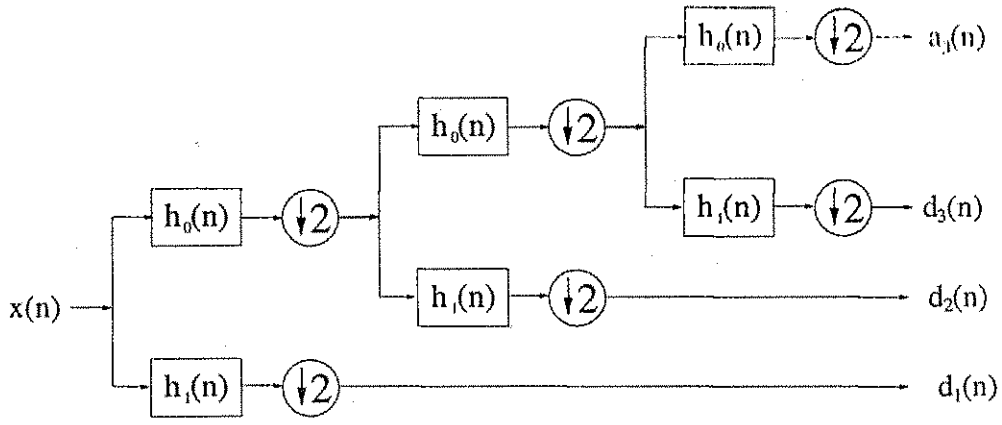
número de níveis de resolução), denotado por $\mathscr{N}$, da transformada *wavelet* discreta. Como resultado, são obtidos, ao final de $\mathscr{L}$ níveis de decomposição, um sinal de aproximação $a_{\mathscr{L}}(n)$, com resolução reduzida por um fator de $2^{\mathscr{L}}$ em relação ao sinal de entrada, e os sinais de detalhe $d_{\mathscr{L}}(n)$, $d_{\mathscr{L}-1}(n)$, ..., $d_1(n)$.

No processo de reconstrução (síntese), procede-se uma interpolação (super-amostragem) de ordem 2, seguida de uma convolução com os filtros passa-baixa $g_0(n)$ e passa-alta $g_1(n)$, como ilustra a Figura 2.2(b). Satisfeitas as condições de reconstrução perfeita, o sinal $x(n)$ é reconstruído de forma exata, ou seja, $y(n) = x(n)$. Na mesma figura observamos que os sinais $a_i(n)$ e $d_i(n)$ recuperados por um estágio de filtragem são submetidos, respectivamente, aos filtros *wavelet* de síntese $g_0(n)$ e $g_1(n)$ para permitirem a reconstrução de $a_{i-1}(n)$, que constitui a componente de aproximação no próximo nível de resolução mais alta. A operação de reconstrução do sinal a partir dos coeficientes *wavelet* constitui a transformada *wavelet* discreta inversa (IDWT, *inverse discrete wavelet transform*).

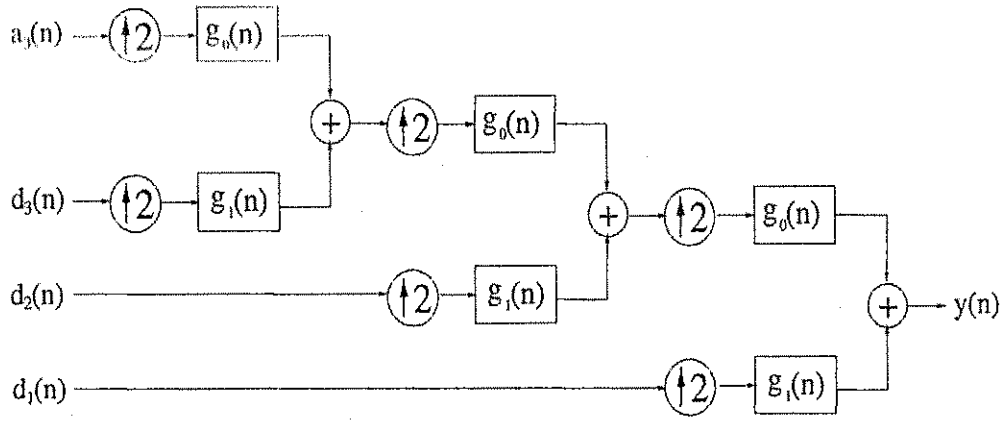
Codificação de Imagens Utilizando DWT

Para realizar a transformada *wavelet* discreta de imagens é necessário especificar como será realizada a extensão para 2-D da transformada definida para 1-D. Normalmente realiza-se a operação de forma separada sobre as linhas e as colunas da imagem. Isto corresponde a utilização de uma transformada cuja base seja separável.

A Figura 2.3 mostra uma decomposição *wavelet* de 3 escalas (3 níveis de resolução, decomposição em 3 níveis) de uma imagem 256 × 256. Os filtros passa-alta e passa-baixa são aplicados à todas as linhas e em seguida à todas colunas da imagem. Para uma imagem $N \times N$, o resultado de cada filtragem é decimado por um fator de 2, resultando $N/2$ coeficientes para cada filtro. O vetor de coeficientes de saída (linha ou coluna), com os elementos de índice de $1 \ldots N$ é formado agrupando-se o resultado da filtragem passa-baixa, denotada por L, nos elementos de índice$1 \ldots N/2$. Os coeficientes da filtragem passa alta, denotada como H, são armazenados nos elementos de índice $N/2+1 \ldots N$. Como resultado deste processo, obtemos um conjunto bi-dimensional de coeficientes com as mesmas dimensões da imagem formado por quatro subconjuntos denominados HL_i, LH_i,



(a)



(b)

Figura 2.2: Transformada *wavelet* discreta em 3 níveis: (a) operação de análise ou decomposição; (b) operação de síntese ou reconstrução.

HH_i, LL_i, que representam, as combinações formadas pela ordem em que os filtros são aplicados. Respectivamente, temos filtragem passa-alta nas linhas seguida de passa-baixa nas colunas formando a sub-banda denominada horizontal; filtragem passa-baixa nas linhas seguida de passa-alta nas colunas formando a sub-banda denominada vertical; filtragem passa-alta nas linhas e colunas formando a sub-banda denominada diagonal; filtragem passa-baixa nas linhas e colunas formando a sub-banda denominada de aproximação. O processo todo é repetido considerando como a imagem a ser transformada a sub-banda de aproximação LL_i, de dimensões $N/2 \times n/2$, que é substituído pelas quatro sub-bandas de índice $i+1$. O índice i indica o nível de decomposição atual.

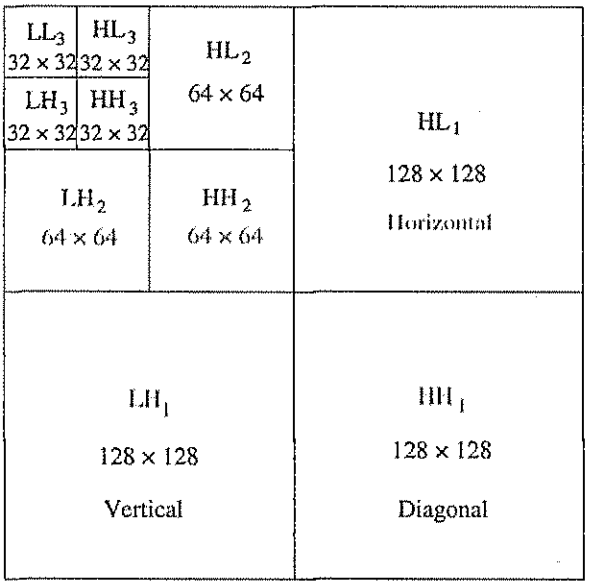


Figura 2.3: Decomposição *wavelet* de 3 níveis de uma imagem 256×256. A filtragem passa-baixa é denotada por L e a passa-alta por H. A ordem das letras indica a seqüência com que as operações são realizadas nas linhas e colunas.

A transformada *wavelet* discreta (DWT) apresenta-se como altamente adequada a ser utilizada em compressão de imagens devido à algumas de suas características [21, 125].

A DWT apresenta uma descrição multiresolucional de uma imagem, de forma próxima ao tratamento efetuado pelo sistema visual humano (HVS). O ganho de codificação obtido

pela DWT é elevado, ou seja, grande parte da energia se concentra em um pequeno número de coeficientes, permitindo uma representação compacta da energia da imagem. Além disso, as imagens reconstruídas por meio da DWT não apresentam a incômoda distorção sob a forma de bloqueamento da imagem, típica de técnicas de codificação envolvendo transformadas de bloco a elevadas taxas de compressão.

Diversos fatores afetam o desempenho dos sistemas de codificação de imagens que utilizam DWT, dentre os quais podem ser citados [16]: a escolha das bases (famílias ou filtros) *wavelet* [2, 3, 68, 114, 118], o tipo de extensão utilizada ao se aplicar a DWT [54, 102], a estratégia de quantização/alocação de bits adotada, a utilização de codificadores entrópicos e a concepção/aplicação de medidas de distorção sintonizadas com critérios de percepção visual.

2.5 Quantização

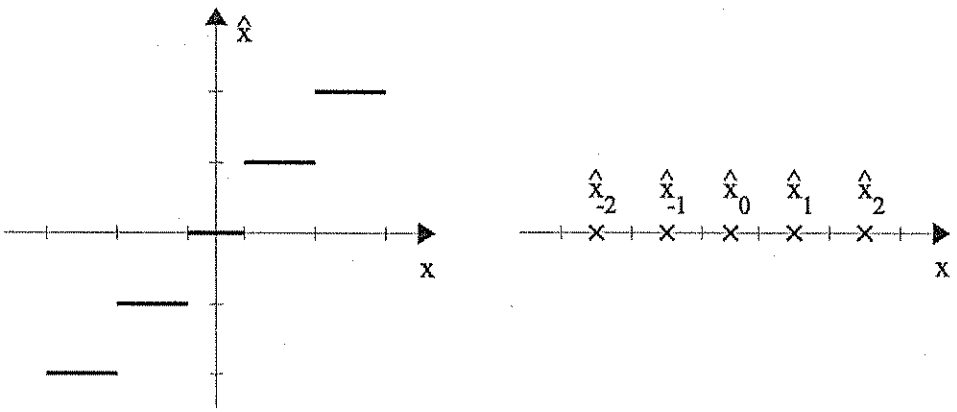
A transformação do sinal de entrada produz coeficientes que podem assumir valores reais. Estes valores devem ser aproximados e representados com precisão finita, ou seja, discreta em amplitude. O processo de mapear um sinal em uma aproximação discreta denomina-se quantização. Os valores quantizados são contáveis e podem ser associados a números inteiros, normalmente denominados índices. Cada número inteiro representa uma região de valores de entrada. A quantização é um processo não inversível, de forma que o sinal original não pode ser recuperado com perfeição. Durante o processo de reconstrução, associa-se um valor real ao índice em questão, buscando-se representar o mais adequadamente possível toda a região associada a este índice. Quando cada elemento é quantizado individualmente, dizemos tratar-se de uma quantização escalar. Quando a quantização é realizada em blocos formados por mais de um elemento, associado a cada bloco um índice, dizemos tratar-se da quantização vetorial.

A estratégia de quantização dos coeficientes transformados desempenha papel muito importante para o bom desempenho dos sistemas de codificação de imagens que utilizam estas transformadas. Tanto a quantização escalar (utilizada em diversos trabalhos com trans-

formadas *wavelet* [22,53,96,99,125] e DCT [119,126]) como a quantização vetorial [57] têm sido aplicadas. Em se tratando de quantização escalar, algumas abordagens podem ser mencionadas, dentre as quais os quantizadores Lloyd-Max e os quantizadores com zona morta (*dead zone*). Esta última utiliza um limiar abaixo do qual um grande número de coeficientes transformados são *zerados*, o que contribui para aumentar a eficiência de uma codificação entrópica subseqüente.

2.5.1 Quantização escalar

Podemos observar na Figura 2.4 o processo de quantização escalar como dois mapeamentos. O primeiro mapeamento, visto à esquerda na figura, associa partições do eixo x com um conjunto de inteiros. O segundo mapeamento, à direita, associa inteiros com um conjunto de valores de saída.



(a) quantizador (b) reconstrução

Figura 2.4: Quantização escalar e sua reconstrução.

O projeto de um quantizador envolve definir os pontos de corte na quantização e os valores de reconstrução, na desquantização. Faz-se necessário definir medidas da distorção para caracterizar os quantizadores. As medidas de distorção serão vistas mais adiante. O objetivo durante o projeto de um quantizador é minimizar a distorção, sobre todos os

valores de x. Para isso pode-se utilizar um modelo probabilístico dos valores dos sinais. A estratégia é ter poucos pontos nas posições em que a probabilidade do sinal seja pequena, ao mesmo tempo que os valores dos sinais mais prováveis possam ser representados mais fielmente. Para o caso específico de sinais provenientes de transformadas lineares, devido à probabilidade normalmente encontrada nos coeficientes, é comum utilizar-se um quantizador escalar com zona-morta. O quantizador com zona-morta, com região central com o dobro da distância entre os demais elementos, é particularmente muito eficiente para a quantização de imagem e vídeo [60]. Uma representação esquemática do quantizador com zona-morta pode ser vista na Figura 2.5.

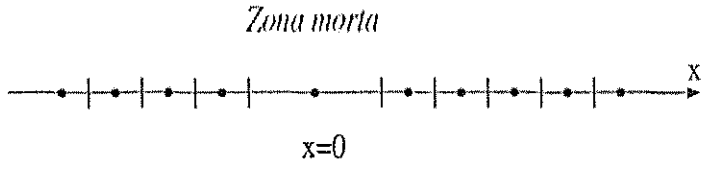


Figura 2.5: Representação do quantizador com zona-morta.

2.5.2 Quantização vetorial

Na quantização vetorial (VQ, *vector quantization*), uma imagem é segmentada em blocos de mesmas dimensões. Os blocos são representados por um número fixo de vetores denominados vetores-código. O conjunto de vetores-código é chamado dicionário. Este procedimento é análogo à quantização escalar, exceto que a quantização agora é realizada em vetores no lugar de escalares. O tamanho do dicionário afeta a taxa (número de bits necessários para codificar cada vetor) como também a distorção, fixada a dimensão (número de amostras quantizadas simultaneamente). Um dicionário maior aumenta a taxa e diminui a distorção média ao passo que um dicionário menor tem o efeito contrário.

Com a quantização vetorial, a codificação é mais intensiva computacionalmente do que a decodificação. A codificação requer a busca no dicionário de um vetor-código represen-

tativo para cada vetor de entrada, enquanto que a decodificação requer apenas um busca numa tabela. Normalmente, o mesmo dicionário é usado no codificador e no decodificador. A geração do dicionário é um processo que é computacionalmente exigente. Tal como um dicionário para codificação sem perda, um dicionário de VQ pode ser construído estaticamente, semi-adaptativamente ou adaptativamente. A quantização vetorial tem sido aplicado diretamente sobre blocos de pixels da imagem ou sobre o resultado da aplicação de transformadas. Algumas aplicações de VQ para a compressão de imagem e vídeo podem ser encontradas na literatura [2,3,13,48,49,81,97]

2.6 Codificação entrópica

Este passo envolve a codificação de uma seqüência de símbolos (produzida pela quantização das imagens decompostas) em um número pequeno de bits, de forma que a seqüência de entrada possa ser recuperada exatamente a partir dos bits codificados. Este processo é também conhecido como codificação sem perda.

Seja S uma fonte aleatória que obtém seus valores de um alfabeto finito de N símbolos $\mathscr{A} = \{s_i\}_{1 \le i \le N}$. O objetivo é minimizar a taxa média de bits necessários para armazenar os valores de S. Consideramos códigos que associam a cada símbolo s_i uma palavra binária w_i de comprimento l_i. Uma seqüência de valores produzidos pela fonte S é codificada agregando-se as palavras binárias correspondentes.

A codificação pode ser realizada através de um código de comprimento fixo. Neste caso, para um alfabeto com N símbolos, a taxa necessária, é de $\lceil \log_2 N \rceil$ bits por amostra. Se N é uma potência de 2, então esta é a menor taxa possível com palavras código de comprimento fixo. Se N não é uma potência de 2, a taxa pode ser melhorada agrupando-se blocos de símbolos. Porém, o comprimento médio do código pode ser reduzido com um código de comprimento variável. De forma geral, os códigos de comprimento variável codificam a informação produzida por uma fonte de forma mais eficiente ao associar palavras código mais curtas aos símbolos que ocorrem mais freqüentemente, minimizando a taxa total e por conseguinte a taxa média, que pode ser vista como o comprimento médio das palavras

código. Há um limite teórico para a melhor taxa que pode ser atingida fixada a fonte, denominada de entropia da fonte. Para uma fonte discreta estacionária e sem memória S de símbolos sobre um alfabeto de tamanho N, a entropia, $H(S)$, é definida como

$$H(S) = -\sum_{i=1}^{N} p_i \log_2 p_i \text{ bits/símbolo,} \tag{2.5}$$

em que p_i é a probabilidade de ocorrência do i-ésimo símbolo s_i. Este é o limite inferior para a taxa necessária para codificar a fonte. Para fontes com memória, a definição de entropia é estendida usando-se probabilidades conjuntas.

Podemos incluir o conceito de quantidade de informação ou auto-informação para cada símbolo s_i, $I(s_i)$, como sendo:

$$I(s_i) = -\log_2(p_i) \tag{2.6}$$

Neste contexto, podemos interpretar entropia $H(S) = \mathsf{E}(I(s_i))$ como a quantidade média de informação por símbolo da fonte, ou como a média da medida da incerteza a ser esclarecida.

Para uma fonte com memória, a incerteza sobre o M-ésimo símbolo é reduzida quando conhecemos os $(M-1)$ símbolos anteriores. A M-tupla com símbolos dependentes contém menos informação, ou resolve menos incerteza, do que uma com símbolos independentes. A entropia de uma fonte com memória é o limite

$$H(X) = \lim_{M \to \infty} H_M(X), \tag{2.7}$$

em que H_M é a entropia da fonte levando-se em conta os M símbolos anteriores.

Observamos que a entropia de uma fonte M-tupla com memória é sempre menor do que a entropia de uma fonte com o mesmo alfabeto e probabilidade de símbolos, porém sem memória:

$$H_M(X)_{\text{memória}} < H_M(X)_{\text{sem memória}}. \tag{2.8}$$

Também foi mostrado por Shannon [98], através do teorema da codificação sem ruído, que pode-se chegar arbitrariamente perto da entropia usando codificação de blocos com

código de comprimento variável. As técnicas de codificação usadas são conhecidas como técnicas de codificação entrópica. Veremos rapidamente duas das técnicas de codificação entrópica mais conhecidas: o código de Huffman e a codificação aritmética.

2.6.1 Código de Huffman

A codificação de Huffman [35] produz códigos de prefixo, ou seja, nenhuma palavra código é prefixo de outra. Desta forma, uma seqüência de palavras código pode ser unicamente mapeada em uma seqüência de símbolos. A construção de um código de Huffman é realizada através de uma estrutura de dados denominada de árvore de Huffman. A codificação pode ser realizada com uma simples busca em uma tabela. A decodificação pode ser realizada percorrendo a árvore de Huffman usando os bits do código como um mapa.

Pode-se mostrar que se todas as probabilidades dos símbolos são potências negativas de dois, a codificação de Huffman atinge o limite entrópico de forma exata [6]. Para probabilidades dos símbolos arbitrárias a codificação de Huffman pode ser ineficiente, já que o comprimento de cada palavra código só pode ser um inteiro. O comprimento ótimo para uma palavra código de probabilidade p_i é $-\log_2 p_i$, o mesmo que a sua auto-informação, que é um número real. Para este caso, pode-se mostrar que com a utilização da codificação de Huffman sobre blocos de símbolos, a entropia pode ser alcançada no limite $k \rightarrow \infty$, em que k é o tamanho dos blocos. No entanto, a complexidade da codificação cresce exponencialmente com k. Uma estratégia interessante, quando há uma assimetria acentuada entre as probabilidades dos símbolos, é realizar a codificação em dois passos. Inicialmente, utiliza-se um alfabeto de símbolos compostos que agrupam símbolos que ocorrem freqüentemente em um único símbolo composto. Isto é conhecido como codificação por comprimento de seqüência (*run-length*). Os símbolos compostos podem então ser codificados com o código de Huffman.

2.6.2 Código aritmético

Na codificação de Huffman há uma palavra código para cada símbolo ou bloco de símbolos. A codificação aritmética atribui diretamente um código completo para a seqüência inteira de símbolos. A idéia é particionar o intervalo $[0, 1)$ em subintervalos não sobrepostos, um para cada seqüência possível de símbolos, de forma que o comprimento do subintervalo correspondente à seqüência seja igual a sua probabilidade de ocorrência. Uma seqüência é codificada especificando-se o subintervalo correspondente. O particionamento do subintervalo é realizado iterativamente: usando uma ordem fixada dos símbolos no alfabeto, o intervalo atual é particionado em pedaços cujos comprimentos são proporcionais às probabilidades de ocorrência dos símbolos do alfabeto. Entre estes subintervalos recém criados, aquele correspondente ao próximo símbolo é escolhido. Este processo é repetido para a seqüência inteira de símbolos.

Uma seqüência de entrada é codificada especificando-se o subintervalo. Pode ser mostrado que se uma probabilidade de ocorrência da seqüência s é p_s, então o subintervalo s pode ser especificado usando $\lceil -\log_2 p_s \rceil$ bits. Assim, cada seqüência pode ser codificada no limite de um bit do seu comprimento ideal.

Da forma como foi apresentado aqui, a codificação aritmética é difícil de ser implementada, devido à precisão necessária para conduzir a aritmética do particionamento de subintervalo, que aumenta com o comprimento da seqüência a ser codificada. As implementações práticas usam diversas simplificações (tais como a normalização do tamanho do intervalo a cada passo e arredondamento para uma aritmética de precisão fixa [6,46,79,123]). Normalmente estas simplificações resultam uma pequena perda de compressão.

Modelos e contextos

A codificação aritmética utiliza um modelo de probabilidades que tem de ser fornecido. Este modelo fornece uma probabilidade para cada símbolo a ser codificado e deve refletir a estatística real da fonte. Existem três formas básicas de modelo: modelo estático, semi-adaptativo e adaptativo.

O modelo estático supõe um conhecimento prévio da função densidade de probabilidade

da fonte. Esta função é conhecida pelo codificador e pelo decodificador. A sua utilização é rara porque pode haver perda se a fonte não estiver correspondendo fielmente à função.

No modelo semi-adaptativo a codificação é realizada em dois passos. No primeiro passo a estatística da fonte é coletada e utilizada no segundo passo. O modelo obtido tem de ser transmitido em conjunto com os símbolos da fonte de forma eficiente.

No caso adaptativo, o modelo é construído continuamente com base nos símbolos já codificados. Isto permite que o modelo se adeqüe a qualquer fonte, sendo eficiente mesmo quando a fonte não é estacionária. O desconhecimento das probabilidades gerais dos símbolos leva a uma ligeira perda de codificação, ao passo que a adaptação a variações locais dessas probabilidades promove um ganho de codificação. Como resultado final a codificação aritmética adaptativa obtém aproximadamente os mesmos resultados da codificação semi-adaptativa, podendo superá-la em alguns casos específicos.

Quando não há um modelo probabilístico para a fonte, as probabilidades dos símbolos geralmente são substituídas por suas freqüências relativas. Isto ocorre nos modelos adaptativo e semi-adaptativo.

Na compressão de dados normalmente são utilizados modelos que representam tanto a probabilidade de ocorrência dos símbolos individuais quanto a probabilidade de ocorrência de grupos de símbolos. O tamanho do grupo de símbolos utilizados no modelo determina a sua ordem. Normalmente, modelos de ordem mais alta são capazes de uma melhor compressão, à custa de um aumento exponencial na complexidade computacional.

Uma vez que o codificador aritmético consegue gerar uma seqüência de bits com o valor estabelecido pelas probabilidades utilizadas [6], a compressão dos dados fica determinada exclusivamente pelo modelo utilizado. Quando o modelo reflete com fidelidade a fonte aleatória, obtém-se boa codificação. A utilização de um modelo inadequado pode levar a uma codificação ineficiente e até mesmo expandir (usar mais bits) os dados. Assim, a realização de um sistema de compressão de dados eficiente é basicamente encontrar modelos fiéis para para as fontes de dados.

Codificador aritmético binário

É possível simplificarmos o codificador aritmético se, além de utilizarmos uma aritmética de precisão reduzida, utilizarmos um número menor de símbolos [33]. No caso extremo, podemos utilizar apenas dois símbolos, o que é denotado como codificador aritmético binário. Neste caso, o particionamento do intervalo atual quando um novo símbolo é emitido corresponde apenas a uma multiplicação pelo valor da probabilidade deste símbolo. A definição e a modelagem dos contextos também é simplificada, porque é necessário produzir apenas um valor de probabilidade (a outra probabilidade é calculada de forma que a soma seja 1), para cada contexto. A adaptação destes contextos também é muito simples, porque não é necessário armazenar e analisar a história dos símbolos emitidos através de uma estrutura de dados complexa. Estas simplificações levaram ao desenvolvimento de alguns codificadores aritméticos muito eficientes, em termos de velocidade de compressão e descompressão e utilização de memória, com uma perda muito pequena na compressão [31,73,79].

Para utilizarmos um codificador aritmético binário com uma fonte com mais de 2 símbolos, é necessário realizarmos um passo de binarização. A binarização corresponde a representar os símbolos da fonte com vários símbolos binários de forma que o símbolo possa ser recuperado no decodificador. A forma mais simples de binarização é representar os M símbolos da fonte com um código binário de comprimento fixo de dimensão $N = \lceil \log_2 M \rceil$. Porém, com esta binarização não há uma forma simples de definição dos contextos, haja visto que na representação binária convencional os bits individuais não são correlacionados.

Para a codificação de fontes em que os símbolos são números, e os números de menor valor são mais prováveis, uma binarização que tem se mostrado muito eficiente é representar o símbolo como uma seqüência de símbolos 0 de comprimento igual ao seu valor numérico seguido por um símbolo 1 [69,70,113]. Este esquema é conhecido como código unário. O código unário é um dos mais simples códigos utilizados na codificação de seqüências de números inteiros.

A codificação de seqüências de números inteiros com determinadas distribuições é am-

plamente estudada. Para algumas distribuições já foram encontrados, de forma fechada, códigos ótimos, ou seja, códigos com comprimento médio igual a entropia. A utilização destes códigos permite chegar aos mesmos resultados em termos de comprimento médio de código do que a utilização da codificação aritmética. No entanto, esses códigos são normalmente muito simples de serem construídos e podem ser utilizados quando a distribuição da fonte é conhecida. Estes códigos também podem ser utilizados como esquemas de binarização para aplicação de um código aritmético binário. Neste caso, em um esquema híbrido, é possível através do uso de contextos, uma melhor adaptação a fonte. O código unário, visto acima, é adequado a codificação de fonte com distribuição geométrica.

A perda de desempenho do código aritmético binário com relação a um código multi-símbolos é marginal [6].

Vimos nas últimas seções alguns aspectos da codificação de sinais sob o paradigma TQC. Neste contexto, as etapas de transformação, quantização e codificação entrópica operam, individualmente, sobre conjuntos de dados, modificando-os e adequando-os à etapa seguinte. Para a devida análise das características de cada um desses blocos funcionais, é mister avaliar o seu potencial quando inserido no contexto de um codificador completo, com os demais blocos fixados. Para tanto, faz-se necessário avaliar de alguma forma a qualidade das imagens codificadas, para diversas taxas. Este tema é tratado na Seção 2.9.

Veremos nas seções seguintes alguns codificadores de imagem encontrados na literatura.

2.7 Codificadores de blocos transformados encontrados na literatura

Nesta seção enumeramos alguns dos algoritmos de codificação de coeficientes de transformada de bloco, em especial da DCT, encontrados na literatura.

2.7.1 RL+VLC

Código de comprimento de seqüência de zeros e código de comprimento variável encontrado nos padrões de imagem JPEG [80,119] e de vídeo MPEG-1, MPEG-2, MPEG-4, H.263 [39,41,50,101]. O padrão de imagem JPEG é apresentado a partir da Seção 2.8. O padrão de vídeo MPEG-4 é analisado sucintamente na Seção 3.1.

2.7.2 CABAC

A sigla denota codificação aritmética binária com contextos adaptativos (*Context Adaptive Binary Arithmetic Coding*) e é usado no novo padrão de compressão de vídeo H.26l (também chamado de H.264 e JVT). Sua descrição pode ser encontrada no artigo de Marpe *et al.* [69,70]. O algoritmo baseia-se na codificação do comprimento de seqüência de zeros, amplitude e sinal de coeficientes quantizados, vetores da estimação de movimento e demais bits de sinalização através de um código aritmético binário baseado em contextos. Os contextos são adaptativos, de forma que a estatística dos elementos não precisa ser transmitida. Foi obtido um ganho no número de bits de até 30% com relação a RL+VLC. No entanto, é necessário investigar quanto deste ganho é obtido na codificação de textura e nos demais componentes do formato.

2.7.3 CEB

O artigo de Chengjie Tu e Trac Tran [113] engloba, na realidade, dois algoritmos, denominados E-CEB e L-CEB. CEB denota codificação entrópica baseada em contextos de coeficientes de transformada de bloco. L-CEB, ou CEB local opera sobre a imagem de bloco em bloco, sem armazenamento dos anteriores. E-CEB, ou CEB *embedded* (progressivo) comprime a imagem através de planos de bits. Todos os coeficientes são armazenados, mas os planos de bits são codificados de bloco em bloco. No L-CEB, após a quantização os coeficientes são varridos em zig-zag. A cada posição associa-se um conjunto formado por todos os elementos da posição atual até o último elemento. Se algum coeficiente deste conjunto for significante, o conjunto é dito significante. Neste caso, o conjunto é dividido

em dois subconjuntos: um conjunto formado pelo coeficiente da posição atual e outro conjunto formado pelos demais coeficientes na ordem de zig-zag. A significância, o sinal e a magnitude do coeficiente isolado são codificados. O algoritmo pára quando se informa que o conjunto não possui nenhum coeficiente significante. Cada codificação é realizada com um contexto diferente.

No E-CEB, o algoritmo é muito similar ao L-CEB, porém o algoritmo trabalha nos planos de bits separados. Todos os blocos são visitados para cada plano de bit. Por isso, faz-se necessário armazenar todos os blocos antes da codificação. A significância em um plano de bit é indicada quando o coeficiente é maior ou igual a um determinado limiar.

2.7.4 ARL

A codificação adaptativa de comprimento de seqüência, ARL (*Adaptive Runlength Coding*) [112] baseia-se em codificar separadamente o comprimento da seqüência de zeros e o nível (valor quantizado) do coeficiente não nulo após a seqüência com código aritmético de contexto adaptativo. Não há, portanto, muita diferença com relação ao CABAC.

2.7.5 EZ-DCT

A aplicação do método de codificação progressiva por árvores de zeros (*embedded zerotree*) aos coeficientes da transformada DCT foi primeiro sugerida por Xiong [126]. Neste trabalho os coeficientes DCT são reagrupados por banda de freqüências de forma semelhante às sub-bandas da codificação *wavelet*. O EZ-DCT é um codificador muito eficiente, com resultados de taxa distorção de cerca de 2 dB de PSNR melhores do que o JPEG para a imagem Bárbara.

2.8 JPEG

A codificação por transformada é a estrutura empregada pelo padrão de compressão de imagem JPEG [119]. A imagem é quebrada em blocos de 8 $\times$ 8 elementos de

imagem, que são transformados usando uma DCT bi-dimensional de 8 bandas. O bloco transformado é quantizado e então codificado. As amostras transformadas e quantizadas em um bloco são varridas e alinhadas em um vetor seguindo um padrão de zig-zag da banda de freqüência mais baixa para a banda de freqüência mais alta. O vetor varrido é aplicado a um codificador entrópico que usa uma combinação de contagem de seqüência de zeros (*run-length*) e codificação com comprimento variável para comprimir os dados. Estes passos serão vistos com mais detalhes adiante.

2.8.1 Representação de cores

Uma imagem colorida pode ser vista como um campo vetorial bidimensional, onde para cada posição discretizada, ou seja, para cada pixel, é associado um vetor de componentes. Este vetor representa as coordenadas em um espaço multidimensional, determinando a intensidade e matiz de cada pixel. Este espaço pode ser determinado por uma base e os seus elementos representados com relação a esta base. Uma das bases mais comuns para representação de sinal colorido é através de bandas espectrais com cores primárias vermelho, verde e azul (RGB, *red*, *green*, *blue*). Esta representação é utilizada em tubos de raios catódicos, em monitores de vídeo. Na codificação de imagem e vídeo é mais usual a representação baseada em luminância e crominância. Esta representação equivale aproximadamente a forma polar de um vetor, composta por seu comprimento e ângulos com relação a eixos de referência. A forma de representação com luminância e crominância é denotada como espaço YCbCr (também conhecido com YUV), onde Y representa a luminância e Cb e Cr representam a matiz. É possível realizar uma conversão entre representações de cor de um pixel com método similar à mudança de base.

A representação no espaço YCbCr para codificação de imagem é motivada pelo fato de o olho humano perceber melhor mudanças na luminância do que na crominância, principalmente sob condição de menor iluminação [110]. Além disso, a resolução espacial, determinada pelo número de células sensoras nos olhos, é bem maior para a luminância do que para a crominância. Para aproveitar este fato, os codificadores de imagem subamostram o sinal de crominância, obtendo uma representação menos detalhada espacial-

mente. Entretanto, esta simplificação não traz grandes alterações na qualidade percebida por um observador humano.

Na representação YCbCr 4:2:0 utilizada no padrão JPEG, a crominância é reduzida pela metade nas dimensões vertical e horizontal. Para cada bloco 8×8 de luminância Y estão associados dois blocos 4×4 de crominância Cb e Cr.

2.8.2 Quantização no padrão JPEG

A quantização aplicada aos coeficientes transformados pode ser vista como uma divisão seguida por um truncamento para inteiro. Especificamente, os coeficientes transformados são inicialmente divididos por uma matriz pré-especificada de inteiros que é ponderada por uma escala de quantização. Após a divisão, os resultados são truncados para valores inteiros.

Um exemplo de matriz de quantização é mostrada na Tabela 2.1. Os coeficientes podem ser especificados de forma a explorar as propriedades do sistema visual humano. Visto que o olho humano é mais sensível às freqüências espaciais baixas e menos sensível às freqüências espaciais altas, os coeficientes transformados correspondentes às freqüências espaciais altas podem ser quantizados de forma mais grosseira do que aqueles com freqüências espaciais mais baixas.

Varredura em zig-zag

Devido à quantização grosseira dos coeficientes correspondentes às freqüências espaciais mais altas, estes coeficientes são freqüentemente quantizados como zero. Uma maneira efetiva de codificar o conjunto resultante de coeficientes quantizados é com uma combinação de uma varredura em zig-zag como mostrado na Figura 2.6 e codificação por comprimento de corrida de zeros. Tipicamente, o coeficientes $DC = F(0,0)$ é codificado separadamente dos outros coeficientes e não é incluído na varredura de zig-zag.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	108	103	77
24	35	55	64	81	194	113	92
49	64	78	87	103	124	120	104
72	92	95	98	121	100	103	99

Tabela 2.1: Exemplo de uma tabela de quantização para o padrão JPEG.

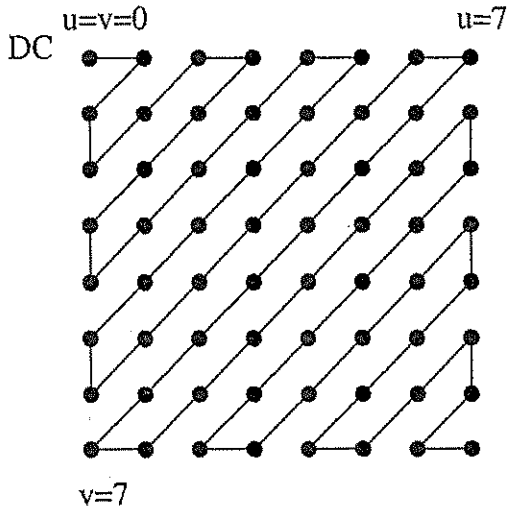


Figura 2.6: Ordem de varredura zig-zag dos coeficientes de um bloco DCT 8×8 no padrão JPEG.

Codificação entrópica dos coeficientes

Trataremos nesta seção da codificação entrópica do padrão JPEG mais usual, conhecida como *baseline*. O JPEG utiliza uma representação intermediária para os coeficientes que combina comprimentos de seqüência e valores de amplitude. Em cada bloco, os 63 coeficientes (obtidos com a exclusão do coeficiente DC) são varridos na ordem de zig-zag (Figura 2.6). O elemento básico para codificação é uma seqüência de zeros seguida de um coeficiente não nulo. Cada uma destas seqüências é codificada com o conjunto de símbolos $[(L_z, L_a), A]$. A variável L_z é o comprimento da seqüência consecutiva de zeros, codificado em 4 bits. Seu valor é limitado ao intervalo $[0, 15]$. Para possibilitar uma seqüência de zeros maior do que 15, o símbolo especial $(15, 0)$ é interpretado como uma seqüência de tamanho 16 seguida por outra seqüência de zeros. Como é muito comum a existência de seqüências de zeros que se estendem até o final do bloco, há um símbolo especial (EOB, *End of Block*) de marca de fim de bloco $(0, 0)$.

O símbolo L_a, codificado com 4 bits informa o número de bits usados para codificar o valor de amplitude do próximo coeficiente não nulo, A. Os valores possíveis estão entre $-2^{L_a} \ldots -2^{L_a-1}, 2^{L_a-1} \ldots 2^{L_a}$. Esta codificação aproveita o fato de que os coeficientes de maior amplitude são menos freqüentes, e por conseqüência codificados com mais bits.

Como mencionado anteriormente, o coeficiente DC da DCT de cada bloco da imagem é proporcional ao valor médio da imagem sobre o bloco. Como os blocos são pequenos, os valores médios de blocos adjacentes são relativamente semelhantes. Para aproveitar este fato, o codificador JPEG codifica o valor DC de cada bloco subtraindo-se o valor DC do bloco anterior na mesma linha de blocos. O coeficiente DC diferencial é codificado como $[(0, L_a), A)]$. Como no caso geral, L_a informa o número de bits usados para codificar A.

O conjunto de símbolos (L_z, L_a) é codificado de forma conjunta através de um código de Huffman porque os valores de L_z e L_a são normalmente correlacionados [80, 119]. O padrão também permite a utilização de um código aritmético como codificar entrópico em substituição ao código de Huffman estático.

2.9 Medidas de qualidade da imagem

Um dos grandes desafios em codificação digital de sinais é a concepção e o desenvolvimento de metodologias de avaliação de qualidade de sinais reconstruídos (obtidos com a aplicação de técnicas de compressão). De forma geral, as medidas utilizadas para avaliação da qualidade de sinais enquadram-se em duas classes: medidas de qualidade subjetivas e medidas de qualidade objetivas. As primeiras baseiam-se em comparações (realizadas por meio de testes de visualização), entre o sinal original e o sinal processado, realizadas por um grupo de pessoas, que subjetivamente classificam a qualidade do sinal processado segundo uma escala pré-determinada. As medidas objetivas, por sua vez, baseiam-se numa comparação matemática direta entre os sinais original e processado. As medidas de qualidade subjetiva não serão tratadas aqui.

Fundamentalmente espera-se restaurar o sinal de forma a ser percebido por um observador humano como quase idêntico ao sinal original. Isto implica a utilização de métricas perceptuais que possibilitem a otimização do código com respeito a nossa sensibilidade às degradações nos sinais. No entanto, por simplicidade, utiliza-se uma métrica como a norma quadrática do erro, que embora não quantifique a distorção percebida, está correlacionada com esta, no sentido de que ao se reduzir a distorção média quadrática geralmente reduz-se a distorção perceptual [115]. É possível utilizar distâncias médias quadráticas ponderadas que fornecem uma medição melhor dos erros perceptuais e podem ser otimizadas com as mesmas técnicas utilizadas para otimizar a norma média quadrática habitual.

O processo de codificação tenta minimizar a distorção média, que pode ser calculada como uma norma média-quadrática ou erro médio quadrático (MSE, *mean-squared error*). Para uma imagem I de dimensões $W \times H$, aproximada por uma imagem $\widehat{I}$, o $MSE(I,\widehat{I})$ é calculado como:

$$MSE(I,\widehat{I}) = \frac{1}{WH} \sum_{0 \le l < H} \sum_{0 \le c < W} (I(l,c) - \widehat{I}(l,c))^2, \tag{2.9}$$

em que $I(l,c)$ e $\widehat{I}(l,c)$ representam os valores de *pixels* das imagens original e reconstruída, l designa a l-ésima linha e c denota a c-ésima coluna de uma

A popularidade de métricas baseadas no MSE advém da tratabilidade desta medida de

distorção. O erro médio quadrático não apenas é fácil de calcular como pode ser analisado e otimizado pelo processo de compressão. No caso de codificação de imagem, utiliza-se normalmente uma métrica derivada da MSE, denominada PSNR, comentada a seguir.

2.9.1 Relação Sinal-Ruído de Pico (PSNR)

A métrica de qualidade baseada em distorção mais utilizada é a relação sinal-ruído de pico [25, 83], (PSNR-*Peak Signal to Noise Ratio*), medida normalmente em decibels, que para imagens em tons de cinza é definida como 10 vezes o logaritmo na base 10 da razão entre o quadrado do valor de pico da amplitude de entrada, (M^2), e o erro médio quadrático (MSE, *mean square error*):

$$PSNR = 10\log_{10}\frac{M^2}{\mathrm{MSE}(I,\widehat{I})}\ \mathrm{dB}, \tag{2.10}$$

em que M é o sinal de pico (o maior valor possível de um pixel). Para imagens com 8 bits por pixel normalmente encontradas o valor de M é 255.

Fixada a imagem e o método de compressão, a PSNR é uma indicação grosseira da qualidade esperada para uma determinada taxa. No entanto, a correlação entre a PSNR e a resposta do sistema visual humano (HVS - *human visual system*) não é tão boa. A razão é que a PSNR não leva em conta a estrutura do erro e cada pixel de erro tem a mesma importância. No sistema visual humano (HVS, *human visual system*), os erros em certas áreas da imagem, como em regiões uniformes, são menos tolerados do que em outras áreas, como em regiões de textura [55, 110, 115]. Normalmente uma imagem codificada com uma PSNR maior tem melhor qualidade do que a mesma imagem codificada com PSNR menor, haja visto que normalmente as técnicas de codificação de alguma forma utilizam a estrutura da imagem. Identifica-se um problema no fato de que para imagens com aproximadamente o mesmo PSNR, codificadas de maneiras diferentes, é possível encontrar uma grande diferença de qualidade subjetiva. Outra dificuldade encontrada no uso do PSNR é o fato de não permitir comparações de qualidade entre imagens diferentes.

Por exemplo, o fato de duas imagens diferentes possuírem o mesmo PSNR não indica que possuem qualidades similares.

A concepção/aplicação de medidas de distorção sintonizadas com critérios de percepção visual constitui importante área de estudo em codificação de imagens [44, 55, 77]. De fato, apesar de existirem diversas medidas de distorção objetivas [25] utilizadas para avaliar a qualidade de imagens comprimidas (reconstruídas), costumam ser relatados problemas apresentados por essas medidas, sendo freqüentemente apontada a incapacidade de modelarem de forma eficiente as características do sistema visual humano (HVS, *human visual system*) [44, 52, 87, 93], ou seja, a incapacidade de apresentarem elevada correlação com resultados de avaliação subjetiva da qualidade de imagens. Deste modo, apesar de as técnicas de codificação serem levadas a efeito no sentido de minimizar uma medida de distorção objetiva, não necessariamente se observa como resultado a obtenção de uma imagem reconstruída com elevada qualidade subjetiva. É importante mencionar, contudo, que algumas abordagens [29, 86, 88] têm sido utilizadas para introduzir aspectos do HVS em sistemas de codificação de imagens.

Ainda com respeito às medidas de distorção utilizadas para avaliar a qualidade de imagens, uma medida de qualidade objetiva denominada PQS (*picture quality scale*), foi apresentada por Miyahara *et al.* em [72]. Levando em consideração tanto distorções globais como degradações locais da imagem, PQS parece apresentar-se como uma alternativa promissora para avaliação de algoritmos de compressão de imagens. Segundo Miyahara *et al.*, a medida PQS apresenta-se fortemente correlacionada com resultados de avaliações subjetivas, sendo relatado em [72] um coeficiente de correlação de 0,92 entre PQS e a medida de qualidade subjetiva MOS (*mean opinion score*).

Capítulo 3

Codificação de vídeo

Um vídeo é basicamente uma seqüência de imagens, normalmente similares. A codificação de vídeo utiliza técnicas de codificação de imagem e técnicas que exploram a similaridade das imagens. Discutimos brevemente a codificação de vídeo e a compressão segundo o padrão MPEG-4.

3.1 Codificação de vídeo

Nesta seção veremos algumas aspectos gerais da codificação de vídeo natural. Nos atemos principalmente ao padrão MPEG-4, como um exemplo de codificador moderno.

3.1.1 Introdução à codificação de vídeo

Resumimos aqui alguns aspectos importantes sobre a codificação de vídeo. Como o tema é muito vasto e muito bem coberto pela literatura especializada [94, 95], nos atemos apenas a alguns pontos mais importantes para o trabalho de tese.

Redundância espacial

A redundância existe em uma seqüência de vídeo em duas formas: espacial e temporal. A primeira, também chamada de redundância intra-quadro, refere-se à redundância que existe dentro de um quadro único de vídeo, enquanto que a segunda, também chamada de redundância inter-quadro, refere-se à redundância que existe entre quadros consecutivos dentro de uma seqüência de vídeo.

Reduzir a redundância espacial é o foco de muitos algoritmos de compressão de imagem. Como vídeo não é mais do que uma seqüência de imagens, técnicas de compressão de imagem são diretamente aplicáveis a quadros de vídeo.

Redundância temporal

Quadros sucessivos em uma seqüência de vídeo são tipicamente bastante correlacionados, especialmente em cenas onde há pouco ou nenhum movimento. As técnicas de descorrelação espacial operam apenas no interior de um único quadro e não exploram a redundância que existe entre quadros.

A técnica mais usual de redução de redundância temporal é a estimação e compensação de movimento. Para estimação de movimento divide-se o quadro em regiões (normalmente blocos quadrados) e faz-se uma busca nos quadros anteriores já previamente codificados e enviados por uma região similar a atual. O método de busca mais utilizado é o casamento de blocos (*block matching*). Este método utiliza o modelo translacional de estimação e compensação de movimento, no qual apenas vetores de deslocamento são informados pelo codificador. A busca é realiza sobre uma região formada pelos pixels ao redor da posição do bloco atual, porém nos quadros anteriores. Esta busca pode ter precisão de pixel ou de sub-pixel. Nos casos de sub-pixel o bloco buscado é formado através de uma interpolação sobre os pixels. A compensação de movimento é realizada tanto no codificador quanto no decodificador e realiza a tarefa de construir uma imagem formada pelos blocos deslocados pelos respectivos vetores de movimento. A imagem formada é a predição temporal.

Um codificador de vídeo é dito híbrido ou heterogêneo se for utilizada uma associação de técnicas diferentes para a redução de redundância temporal e espacial. O exemplo

mais comum é o emprego de compensação de movimento para aproveitar a redundância temporal e a codificação através de transformada para reduzir a redundância espacial. O codificador de vídeo homogêneo emprega extensões para 3D de técnicas empregadas em imagens 2D. Todos os padrões internacionais de codificação de vídeo são híbridos.

3.2 Codificação de vídeo natural no padrão MPEG-4

Nesta seção revisamos de forma sucinta a codificação de vídeo natural segundo o padrão MPEG-4. Para maiores detalhes o leitor pode dirigir-se a descrições encontradas na literatura [50, 84, 94, 95].

Um diagrama de blocos simplificado da codificação híbrida de vídeo MPEG-4 pode ser visto na Figura 3.1.

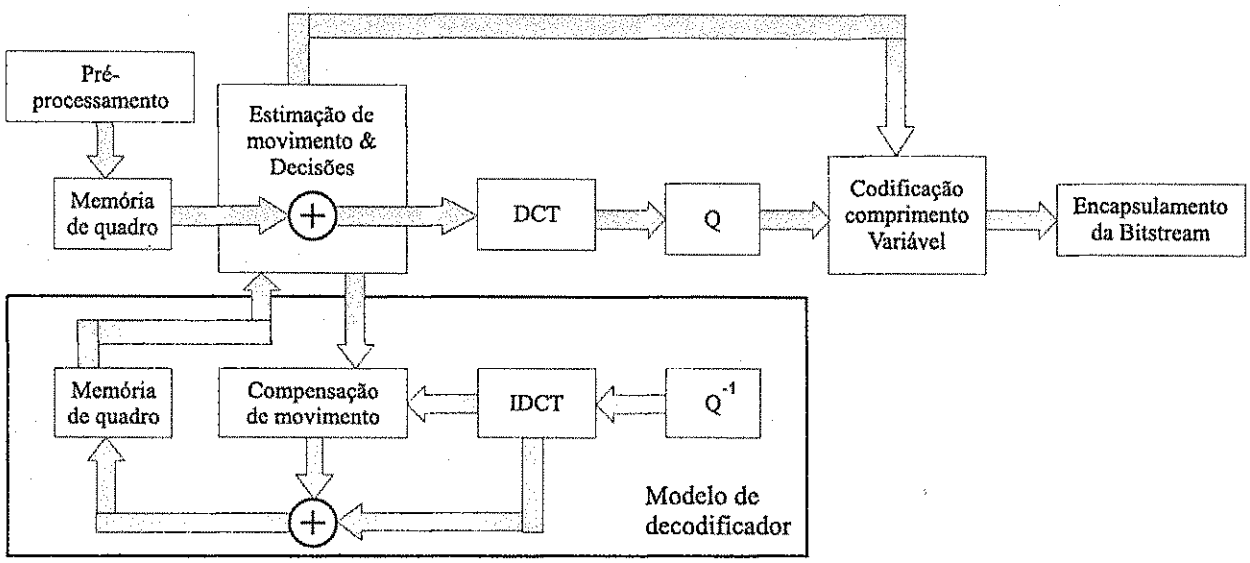


Figura 3.1: Codificador de vídeo MPEG-4.

O padrão MPEG-4 normalmente trabalha com seqüências de imagens no formato YCbCr 4:2:0, ou seja, 1 valor de luminância para cada pixel da imagem, e 1 par de valores de crominância (Cb e Cr) para um bloco de 2×2 pixels da imagem. Para a codificação, cada imagem é dividida em macroblocos. Um macrobloco tem dimensões de

16×16 pixels e é formado por 6 blocos 8×8, sendo 4 blocos de luminância Y e 2 blocos de crominância (Cb e Cr).

A codificação é realizada de forma independente para cada macrobloco. Nas imagens codificadas como *intra*, ou seja, sem referência a outra imagem da seqüência, utiliza-se um processo muito similar à codificação de imagem do padrão JPEG, onde cada um dos blocos de 8×8 pixels pertencentes aos macroblocos são transformados pela DCT. Os coeficientes DCT são então quantizados (efetivamente divididos por um número inteiro e arredondados). Em seguida os coeficientes passam por uma etapa de predição dos coeficientes DC e AC. Estes coeficientes são varridos e alinhados em um vetor segundo uma ordem pré-estabelecida, escolhida entre três opções (zig-zag, varredura horizontal e varredura vertical), da banda de freqüência mais baixa para a banda de freqüência mais alta. Com esta varredura aproveita-se a característica de agrupamento de energia da DCT. Faz-se uma codificação de corrida de zeros (*run-length*) e os valores obtidos são codificados usando uma tabela tridimensional de código de comprimento variável. Este processo é muito eficiente mas não explora, além do passo de predição, a dependência estatística entre os blocos.

Nas imagens codificadas como *inter*, realiza-se um processo de estimação de movimento entre a imagem atual e uma imagem de referência, onde associa-se a cada macrobloco (ou bloco, dependendo do modo escolhido) um vetor de deslocamento. Este vetor representa a predição do valor do macrobloco atual partindo da imagem de referência. Os vetores obtidos são codificados e enviados. À formação da imagem de predição dá-se o nome de compensação de movimento. A diferença entre a imagem atual e a imagem compensada denomina-se imagem resíduo. A imagem resíduo é codificada de forma semelhante à codificação de uma imagem *intra*. Realiza-se a transformação DCT seguida por um passo de predição (apenas para os macroblocos *intra*), quantização e codificação entrópica com corrida de zeros e código de comprimento variável. A alta correlação existente entre imagens subseqüentes leva a uma boa predição realizada pela estimação e compensação de movimento. Assim, a energia da imagem resíduo é pequena, o que leva a pequenos valores de coeficientes DCT, uma vez que a DCT é uma transformada ortonormal. Quando

os coeficientes são quantizados, dependendo do coeficiente de quantização, Q, utilizado, o número de zeros resultantes pode ser muito alto e os coeficientes não zero geralmente possuem valores bem pequenos. Isto motiva a codificação conjunta dos coeficientes na tentativa de representar o maior número de zeros com o menor número de bits.

Na decodificação de vídeo MPEG-4, para uma imagem *intra*, os bits de entrada após a demultiplexação são decodificados, através da inversão da codificação de comprimento variável e de corrida de zeros, obtendo-se os coeficientes quantizados transmitidos. O bloco é reformulado ao inverter-se a varredura. Os coeficientes aproximados são reconstituídos, através da inversão da quantização (as amostras são multiplicadas pelo mesmo número que foi usado para dividir as amostras transformadas antes de serem arredondadas durante o passo de quantização). A DCT inversa é aplicada e obtém-se a versão codificada da imagem transmitida. Nas imagens *inter*, os vetores de movimento são lidos e utilizados na compensação de movimento para obter a imagem de predição. Os coeficientes da imagem resíduo são obtidos de forma similar a imagem *intra* e, após reconstituídos, são somados a imagem de predição, formando a versão final da imagem transmitida. Esta imagem é armazenada como imagem de referência. O codificador MPEG realiza uma decodificação como parte do processo de codificação. Os blocos do codificador referentes à decodificação podem ser vistos agrupados por uma linha pontilhada na Figura 3.1.

A DCT bidimensional utilizada na codificação MPEG foi vista com mais detalhes na Seção 2.4.1.

3.2.1 Quantização

A quantização é realizada através da divisão dos coeficientes por uma tabela de quantização de tamanho 8×8. Esta divisão é feita elemento por elemento (quantização escalar). O padrão MPEG-4 permite o uso de duas tabelas pré-definidas, uma herdada do codificador H.263 [39], onde os fatores são todos iguais (com valor = 8), outra chamada MPEG-4, na qual os fatores refletem a resposta de freqüência do olho humano. Além dessas tabelas é possível utilizar outros valores, explicitando as tabelas para a codificação *intra* e para *inter* que devem ter seus valores armazenados no cabeçalho do arquivo codificado.

3.2.2 Codificação entrópica dos coeficientes

A codificação entrópica do padrão MPEG-4 é baseada no uso de códigos de comprimento de seqüência de zeros e de comprimento variável. Cada macrobloco pode ser classificado como ignorado (*skip*), ou seja, igual ao quadro anterior, não codificado (*not coded*), isto é, apenas com vetores, sem resíduo e codificado (*coded*), com vetores de movimento e resíduo. Além disso, os blocos codificados podem ter um vetor ou quatro vetores e cada um dos blocos 8×8, tanto de luminância quanto de crominância, podem não ser codificados, sendo gasto um bit para cada bloco nessa indicação. Estas decisões são codificadas em bits no início de cada macrobloco. Assim, é possível fazer uma codificação muito eficiente representando, às vezes, um macrobloco inteiro por apenas um bit, no caso dele ser ignorado.

Em seguida veremos com mais detalhes como é realizada a codificação entrópica dos macroblocos do tipo *intra* e do tipo *inter*.

Macroblocos *Intra*

A codificação de macroblocos do tipo intra inicia-se com a aplicação da DCT aos blocos 8×8 individuais, tanto de luminância quanto de crominância. Baseado nos valores dos coeficientes e nos valores dos blocos imediatamente adjacentes e causais na imagem, define-se qual bloco servirá como referência para a predição e se haverá predição dos coeficientes AC. A predição DC é realizada subtraindo-se o valor atual pelo valor DC do bloco de referência. A posição do bloco de referência é enviada no fluxo de bits (*bitstream*). A predição AC pode ser vertical ou horizontal, e utiliza o mesmo bloco de referência usado na predição DC. Neste caso, subtrai-se a primeira linha (predição horizontal) ou primeira coluna (predição vertical), pela linha ou coluna correspondente do bloco de referência.

Os coeficientes são enfileirados através de uma varredura, que pode ser de três tipos, a depender da predição realizada, como mostrado na Figura 3.2. A varredura pode ser (a) em zig-zag , (b) vertical e (c) horizontal.

Este vetor de elementos é codificado inicialmente com corrida de zeros, onde os zeros contíguos são representados por sua quantidade. Todo elemento é então representado

como um par de números formado pela quantidade de zeros que precede o seu valor. Estas seqüências de zero tendem a ocorrer nas altas freqüências, que ficam normalmente no final do vetor. Desta forma, torna-se necessária uma forma eficiente de explicitar que não há mais valores não nulos a serem codificados. No MPEG 4, este fato é codificado conjuntamente com o número de zeros e o valor do coeficiente através de um código de comprimento variável tridimensional (*VLC, variable length code*), como veremos logo adiante.

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

(a) zig-zag

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60
11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

(b) vertical

0	1	2	3	10	11	12	13
4	5	8	9	17	16	15	14
6	7	19	18	26	27	28	29
20	21	24	25	30	31	32	33
22	23	34	35	42	43	44	45
36	37	40	41	46	47	48	49
38	39	50	51	56	57	58	59
52	53	54	55	60	61	62	63

(c) horizontal

Figura 3.2: Formas de varredura dos coeficientes DCT.

Macroblocos *Inter*

Os macroblocos do tipo *inter* são codificados de maneira muito similar aos blocos do tipo *intra*. Mas como a imagem de resíduo é muito diferente de uma imagem natural, a estatística dos coeficientes DCT também é muito diferente. Não há uma aglutinação em torno das baixas freqüências de forma tão evidente. Também, é comum encontrar coeficientes de alta freqüência com valores mais significativos. Por isso, não há predição AC para os macroblocos *inter* e apenas a varredura zig-zag é utilizada. A tabela VLC é diferente e mais adequada à estatística destes coeficientes.

3.2.3 Código de comprimento variável

Os coeficientes DCT quantizados de um bloco da imagem são codificados através de um código de comprimento variável especializado em que três elementos são representados em uma única tabela. Um único conjunto de bits representa o número de zeros que precede um valor (*run*), seguido do próprio valor do coeficiente (*value*) e de um bit indicando se este é o último elemento não nulo (*last*). O sinal do coeficiente é acrescido ao código, como último bit. Esta tabela não cobre todos os casos, apenas os mais prováveis e os demais casos são codificados com um código especial (*escape*) seguido por um código de comprimento fixo.

Este método é muito eficiente e de baixa complexidade computacional, tanto no codificador quanto no decodificador.

Capítulo 4

Codificação por mapa de significância

Neste Capítulo investigaremos a codificação baseada na separação entre coeficientes nulos e não nulos através de um mapa de significância. Por meio de um algoritmo muito simples de codificação baseado em mapa de significância, observaremos a proporção de bits gastos para codificar a significância, o sinal e a amplitude dos coeficientes.

De posse destes resultados, poderemos ter uma noção de como melhorar a codificação. A importância relativa na quantidade de bits utilizados durante a codificação de sinal, magnitude e mapa de significância pode levar a novas técnicas de codificação de imagem, vídeo e outros sinais. Um exemplo disto é o casamento parcial de padrões. Outro conceito importante é o de plano de bits, muito utilizado em algoritmos que possuem a característica de codificação progressiva, como o SPIHT [96]. A relação entre mapa de significância e plano de bits também é apresentada. O contexto no qual o algoritmo proposto está inserido é investigado através da análise de outros trabalhos similares.

4.1 Introdução

Analisaremos em seguida a separação entre coeficientes nulos e não-nulos através do uso de mapa de significância.

4.2 Mapa de significância

Ao quantizar um sinal transformado, muitos coeficientes nulos são obtidos. Isto ocorre devido ao agrupamento e à ordenação por relevância levados a cabo pela transformada. Assim, muitos algoritmos tratam o símbolo zero de forma especial e, implícita ou explicitamente, codificam um mapa de significância. Lembramos que um elemento é dito significante se for não nulo.

Um mapa binário de significância pode ser construído explicitamente. Na Figura 4.1(a) vemos a imagem Lena. Na Figura 4.1(b) observa-se DCT 8×8 (normalizada para ser apresentada como uma imagem) e na Figura 4.1(c) o mapa de significância dos coeficientes DCT. Para obter-se este mapa de significância os coeficientes foram quantizados uniformemente (divididos e arredondados) pelo fator $Q = 150$. Os valores significantes (não nulos) são representados como pixels pretos. Observa-se claramente o agrupamento de coeficientes significantes ao redor da posição DC. Um fato importante a se observar é a correlação existente entre os blocos próximos. Isto ocorre devido à regularidade e localidade das imagens, que levam à semelhança na resposta espectral de blocos vizinhos.

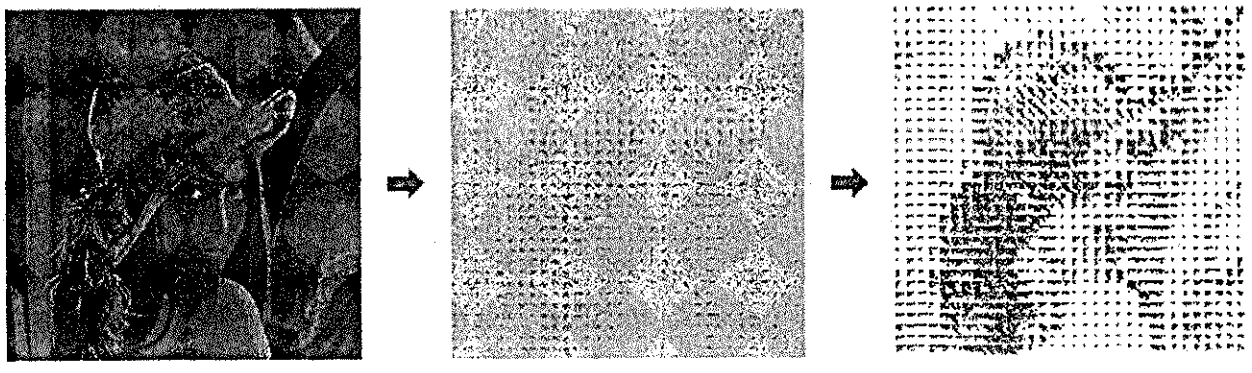
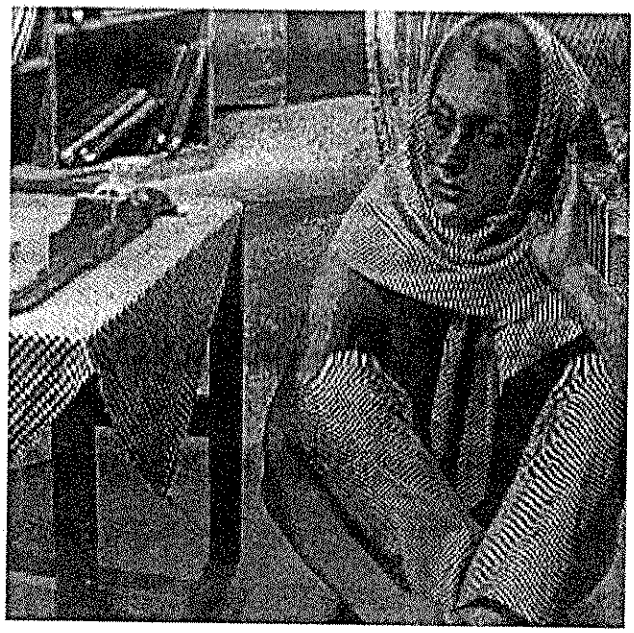


Figura 4.1: Obtenção do mapa de significância para a imagem Lena (256×256). Em (a) vemos a imagem original. Em (b) temos os coeficientes DCT e em (c) o mapa de significância dos coeficientes quantizados com $Q = 150$



(a)

(b)

Figura 4.2: Imagens Barbara e Lena

Um bloco de coeficientes quantizados pode ser descrito por meio do seu mapa de significância e da amplitude e sinal dos elementos não nulos. Esses elementos podem ser codificados separadamente ou de forma conjunta. Observamos que nos codificadores baseados em comprimento de seqüência de zeros, como o JPEG, a amplitude é codificada conjuntamente com o número de zeros, haja visto que esses valores são correlacionados. Nos codificadores baseados em contexto, como o ARL (*Adaptive Run Length*) [112] e CEB (*Context-Based Entropy Coding Of Block Transform Coefficients*) [113], estes fatores são codificados separadamente, permitindo uma melhor adaptação dos contextos.

A seguir introduzimos um algoritmo de codificação simples, baseado na codificação aritmética de contextos [123], em que os fatores representativos dos coeficientes, quais sejam significância, sinal e amplitude, são codificados separadamente e o número de bits relativo a cada fator é medido.

Entrada: Lista de blocos B de coeficientes transformados

```
1 início
2     para i = 0 até dim(B) faça
3         x ← zig-zag(B[i]);
4         para j = 0 até dim(x) faça
5             codifique(x[j] ≠ 0);
6             se x[j] ≠ 0 então
7                 codifique(x[j] > 0);
8                 codifique(binarize(|x[j]|));
9 fim
```

Figura 4.3: Algoritmo para análise do uso de bits na codificação de significância, sinal e amplitude.

4.2.1 Algoritmo de um codificador simples por mapa de significância

Para observarmos com clareza o número de bits utilizados na codificação de cada um dos elementos que formam um bloco de coeficientes, introduzimos o algoritmo baseado na codificação aritmética de contextos, que pode ser visto na Figura 4.3. Cada uma das codificações realizadas nos passos 5,7 e 8 do algoritmo utiliza um contexto diferente. Utilizamos um codificador aritmético binário adaptativo com contextos realizado de acordo com a descrição da Seção 2.6.2. A significância e o sinal, que são elementos binários, são codificados diretamente. O valor absoluto dos coeficientes é binarizado utilizando código unário. Utilizamos contextos separados apenas para os primeiros bins obtidos com o código unário. Os bins a partir de um determinado valor são todos codificados com um mesmo contexto. Este valor é um parâmetro do codificador. Antes da codificação, subtraímos 1 do valor do coeficiente quantizado para reduzir a sua faixa dinâmica. Isto é possível porque sabemos que o coeficiente é significativo, logo seu valor é pelo menos 1. O valor original é restaurado no decodificador. O processo de binarização com código unário pode ser resumido na Tabela 4.1. Nesta tabela X é o contexto inicial, $X+M$ é o último contexto reservado para binarização. A partir deste valor, os bins utilizam o mesmo contexto.

Valor Absoluto	Valor-1	Código	Contexto
1	0	1	X
2	1	01	$X+1$
3	2	001	$X+2$
M	$M-1$	0...01	$X+M$
...	...	...	$X+M$

Tabela 4.1: Construção do código unário.

4.2.2 Resultados

Nas Tabelas 4.2 e 4.3 são apresentados os resultados da aplicação do algoritmo às imagens Bárbara e Lena, mostradas na Figura 4.2, para algumas taxas, utilizando DCT. Para cada taxa está discriminado o número de bits usado na codificação de cada fator constituinte. Nessas tabelas também encontramos a qualidade objetiva em termos de relação sinal-ruído de pico (PSNR, *Peak Signal to Noise Ratio*). É interessante observar que mesmo esse algoritmo muito simples produz resultados melhores do que o JPEG, para todas as taxas medidas. Isto se deve à capacidade de adaptação da codificação aritmética por contextos. Se for utilizado um conjunto mais elaborado de contextos, de forma a aproveitar as estruturas inter e intra-blocos, os resultados podem ser melhores, como podemos observar no algoritmo L-CEB (*Context Based Entropy Coding of Block Transform Coefficients*, codificador entrópico de blocos baseado em contextos) [113].

Taxa (bpp)	Razão Comp.	Coefic. signif.	Codific Amplitude	Codific Sinal	Codific signif.	signif/total (%)	PSNR (dB)	PSNR JPEG
1,00	1:8	46052	10424	5987	16460	50,0	34,90	33,25
0,50	1:16	22455	3541	2922	9916	60,6	29,88	28,49
0,25	1:32	10875	1288	1424	5484	66,9	26,21	25,10

Tabela 4.2: Análise da proporção de bits utilizados para representar significância, sinal e amplitude, na codificação da imagem Bárbara, utilizando DCT.

A codificação separada dos fatores revela que a maior parte dos bits é gasta na representação do mapa de significância, para várias taxas. O sinal dos coeficientes transformados não apresenta estrutura simples e é normalmente considerado como incompressível, com entropia perto de 1 bit/símbolo. Esta suposição é especialmente válida quando a transformada é ortogonal, como a DCT. A codificação do sinal dos coeficientes transformados foi estudada em detalhes por Deever e Hernani [20]. No trabalho, através de um pro-

Taxa (bpp)	Razão Comp.	Coefic. signif.	Codific Amplitude	Codific Sinal	Codific signif.	signif/total (%)	PSNR (dB)	PSNR JPEG
1,00	1:8	44574	10462	5788	16626	50,6	38,48	37,96
0,50	1:16	21505	4833	2792	8775	53,6	34,95	34,86
0,25	1:32	10828	2090	1401	4718	57,6	31,40	31,40

Tabela 4.3: Análise da proporção de bits utilizados para representar significância, sinal e amplitude, na codificação da imagem Lena, utilizando DCT.

cesso complexo de projeção de coeficientes e para transformadas biortogonais, foi possível comprimir marginalmente a representação do sinal. Da mesma forma, considera-se que os valores absolutos dos coeficientes significativos não estão correlacionados. Esses valores absolutos são de difícil compressão e não há ganho significativo com a aplicação de contextos ou modelos mais elaborados.

Os resultados das Tabelas 4.2 e 4.3 mostram que a codificação eficiente do mapa de significância é essencial na compressão de imagens com transformada de bloco.

4.3 Conclusão

No presente capítulo foi apresentado um método simples para codificação de imagens por mapa de significância. O método apresentado é baseado em codificação aritmética de contextos e, apesar da simplicidade, apresenta uma superioridade sobre o JPEG. Para a imagem Bárbara, por exemplo, para a taxa de codificação de 0,25 bpp, um ganho de 1,11 db, em termos de relação sinal-ruído de pico da imagem reconstruída, é obtido ao se substituir a codificação entrópica do JPEG pelo método apresentado.

Verificamos através da aplicação de um algoritmo que separa o número de bits utilizado na codificação da significância, sinal e amplitude dos coeficientes dos blocos, que a maior

parte dos bits é gasta na representação da significância, e que o sinal e a amplitude são de difícil compressão. Observamos que estes mapas de significância dos blocos transformados apresentam uma estrutura de agrupamento, herdada da relação de amplitude dos coeficientes, que pode ser aproveitada para sua codificação.

Capítulo 5

Casamento parcial de padrões

Neste capítulo analisamos a técnica proposta de casamento parcial de padrões e sua utilização na codificação de imagem e vídeo. Para tal a técnica é aplicada a codificação sem perda de coeficientes oriundos de transformadas de bloco. Inicialmente, analisamos um codificador simples baseado em mapa de significância onde procuramos por similaridades entre os blocos. Este experimento mostra que o mapa de significância possui grande potencial para uma codificação utilizando casamento de padrões.

O casamento parcial de padrões tem sido utilizado com sucesso em diversos trabalhos de codificação de imagem, em particular nas images textuais. Uma revisão bibliográfica destes trabalhos é realizada. Analisamos também fatores como tipos de dicionários de padrões, que definem os possíveis tipos de codificadores; os métodos de busca; as métricas utilizadas; as formas de codificação dos índices e a correção necessária para tornar o método sem perda através da codificação de resíduos.

A técnica permite melhorar a codificação entrópica de sinais de vídeo ao explorar de forma explícita as redundâncias espacial e temporal. A utilização em um codificador de vídeo MPEG-4, substituindo apenas sua etapa de codificação entrópica, é mostrada no próximo capítulo. Os resultados experimentais são apresentados no Capítulo 7.

5.1 Teoria e aplicações de casamento de padrões

A técnica de casamento parcial de padrões visa identificar seqüências de símbolos que possuam algum grau de similaridade. Na codificação usando esta técnica, substitui-se a seqüência atual por aquela encontrada como similar. Há ganho de codificação quando a representação da informação sobre a similaridade utiliza menos bits do que a informação da própria seqüência.

O casamento imperfeito, ou parcial, de padrões pode ser aplicado a codificação sem perda se um novo passo de codificação, normalmente denominado de passo de refinamento, for aplicado ao resíduo obtido. O resíduo é a diferença entre os símbolos reais e os símbolos codificados.

Uma característica muito importante a ser analisada na codificação baseada em casamento de padrões, é a definição do dicionário (ou biblioteca) a ser utilizado. É necessário estudar as diversas formas de criação e manutenção destes dicionários.

Quanto à criação, podemos ter os dicionários estáticos, que já são conhecidos de alguma forma antes da codificação, tanto pelo codificador quanto pelo decodificador. Esta forma é muito similar à metodologia usada na quantização vetorial, diferindo no fato de que no nosso caso a codificação é sem perda. Uma variação do dicionário estático é o dicionário parametrizado, em que o dicionário é modificado, de acordo com um ou mais parâmetros, obtidos e enviados pelo codificador.

O dicionário também pode ser criado após uma análise de todos os dados. Neste caso dizemos que o dicionário é semi-adaptativo e o dicionário deve ser de alguma forma codificado e enviado para o decodificador. Este método é similar à codificação de Huffman, onde a tabela de código de comprimento variável construída precisa ser codificada e enviada. A terceira forma de criação de dicionário é a adaptativa.

O dicionário adaptativo é criado ao mesmo tempo em que o sinal é codificado a partir de um dicionário inicial, que pode ser nulo. O processo de criação é sincronizado no codificador e decodificador, de forma que os símbolos enviados podem ser decodificados. Para tanto, o codificador precisa usar na criação do dicionário apenas informações que ele tenha enviado previamente para o decodificador.

Os dicionários estáticos precisam de um prévio conhecimento da estatística da fonte e devido à falta de adaptabilidade não costumam produzir os melhores resultados. Por isso, este trabalho não contempla um estudo dos dicionários estáticos. Os dicionários adaptativos e semi-adaptativos possuem um maior potencial de capacidade de codificação e possuem mais variáveis que precisam ser estudadas. Os pontos principais são a codificação eficiente dos dicionários semi-adaptativos, haja visto que estes também precisam ser enviados, e a modelagem na criação e expansão dos dicionários adaptativos.

A técnica de casamento de padrões introduzida contrasta com as técnicas diretas, que sejam aquelas que codificam sinal, magnitude e mapa de significância em apenas um passo. Uma análise importante que necessita ser realizada é a de quando a codificação baseada em casamento de padrões fornece melhores resultados do que as técnicas diretas. No caso do casamento de padrões, são gastos bits para informar o índice selecionado no dicionário e na representação do resíduo. É preciso analisar a estatística dos coeficientes e dos mapas de significâncias, antes e depois da aplicação do casamento de padrões, como também a estatística dos índices do dicionário. O resultado total é a soma dos bits gastos para representar estas duas parcelas. O casamento de padrões revela-se mais eficiente quando esta soma é menor do que o número de bits utilizados na codificação direta.

Outro aspecto importante na análise da codificação por casamento de padrões é a formulação de uma métrica que determine a distância entre o elemento procurado e o os elementos do dicionário. A métrica usual quando se trata de objetos binários é a distância de Hamming, que é o número de bits que diferenciam, calculado como uma operação XOR entre os elementos e contagem do número de bits com valor "1". Existem outras métricas que podem ser utilizadas, como XOR ponderada, etc. A distância deve ser de alguma forma relacionada com a taxa de bits utilizados para codificar o bloco procurado.

É necessário também definir como o padrão encontrado será utilizado durante a codificação do bloco atual. As formas mais usuais são o SPM e o PM&S. O PM&S (*Pattern Matching and Substitution*) codifica a diferença entre os blocos, que deve ser de alguma forma minimizada. Por outro lado, no SPM (*Soft Pattern Matching*), o padrão encontrado é utilizado na construção dos contextos da codificação entrópica do bloco atual. Outras

técnicas podem ser encontradas na literatura sobre codificação de imagens binárias [8,32]

Com base na métrica e na técnica de codificação selecionadas, define-se como será realizado o processo de busca do padrão mais representativo. A busca exaustiva procura em todos os elementos do espaço de busca e seleciona a menor distância. A depender das dimensões do dicionário, também chamado de espaço de busca, e da complexidade da métrica utilizada a busca exaustiva pode possuir uma alta complexidade computacional. Esta busca permite a obtenção do melhor resultado possível, por definição, e é normalmente utilizada como referência de qualidade para outros métodos de busca. Um método simples, porém eficiente, de aceleração da busca exaustiva, originalmente proposto para quantização vetorial, é a busca por distâncias parciais [5]. Naquele algoritmo, durante a busca exaustiva, armazena-se o valor de distância mínima atual, que é comparado, durante o cálculo da distância, de forma a auxiliar a decidir rapidamente que um determinado elemento não pode ser menor que o mínimo atual. Várias outras formas de aceleração de busca exaustiva, tanto no contexto de quantização vetorial quanto na busca de vetores de movimento em codificação de vídeo, podem ser encontrados na literatura [34,36,85,100,106].

Se o espaço de busca possuir alguma estrutura de formação, esta pode ser aproveitada de forma que a busca seja realizada mais rapidamente. Neste caso, durante a busca o espaço de busca não é percorrido completamente, mas utiliza-se um percurso que, baseado na estrutura do dicionário, tenha a tendência de minimização da distância. Na literatura podemos encontrar diversos exemplos de busca estruturada usada na quantização vetorial, como no caso do *Lattice VQ* [97]. Também na codificação de vídeo, podemos encontrar vários exemplos de busca estruturada de vetores de movimento. Alguns exemplos clássicos são a busca em três passos (*Three step search*, TSS) [51], logarítmica [43] e busca em quatro passos (*four step search*) [82].

A busca exaustiva permite a utilização de uma métrica que englobe taxa-distorção. A minimização desta métrica significa a melhor codificação possível fixados o dicionário e a codificação entrópica. A utilização de buscas não exaustivas com métricas baseadas em taxa-distorção não é muito comum e precisa ser melhor desenvolvida.

O casamento parcial de padrões tem sido utilizado recentemente como a base para

diversos algoritmos de compressão. O trabalho de Bookstein e Klein [7] é direcionado à compressão de vetores (mapas) de bits correlacionados, tais como imagens binárias e imagens textuais [1].

Um exemplo claro da utilização de CPP é o padrão de compressão de imagens binárias JBIG2 [32, 47, 127, 128], baseado no formato DjVu da AT&T [8]. Este sistema de codificação atinge uma taxa de compressão significativamente maior com relação ao padrão anterior JBIG [40] graças ao uso de casamento parcial de padrões.

Outros trabalhos de codificação com perda ou sem perda de imagens textuais [30,37,38, 122] também são baseados no uso de elementos de referência, que formam um dicionário ou biblioteca de símbolos. Os elementos da imagem são codificados através da substituição dos símbolos. No caso sem perda, o resíduo, que é a diferença entre a imagem original e a formada pelos símbolos, também deve ser codificado.

O caso clássico da utilização de casamento de padrões na codificação entrópica é o codificador Ziv-Lempel [6]. Neste codificador as seqüências são casadas apenas se forem idênticas. O Ziv-Lempel faz parte da classe de codificadores entrópicos de dicionário, ou seja, que realiza busca por cadeias de símbolos. A cada cadeia de símbolos encontrada é associado um novo símbolo. O conjunto de novos símbolos e suas cadeias associadas formam um dicionário de busca. O algoritmo de busca guloso, que tenta casar o maior sub-cadeia possível, é o mais utilizado na classe de codificadores de dicionário.

Com princípios semelhantes foi desenvolvido o Ziv-Lempel com perda [26,56], no qual o casamento pode ser imperfeito. A aplicação do Ziv-Lempel com perda diretamente a conjuntos de pixels de uma imagem e entre os quadros de uma seqüência de vídeo tem sido avaliada [104, 105]. Nestes trabalhos não são utilizadas técnicas convencionais como transformadas e estimação de movimento. No entanto, foram obtidos resultados promissores, com boa qualidade de reconstrução aliada a uma complexidade do decodificador muito pequena.

Vale a pena reiterar que o presente trabalho, apesar de lidar com um casamento apenas parcial dos padrões, é utilizado como auxiliar da codificação sem perda, obtendo a exata

[1] Denotamos uma imagem textual como uma imagem binária formada principalmente por texto impresso.

descrição dos símbolos através do passo de refinamento.

5.2 Casamento de padrões na codificação de transformadas de bloco

De uma forma geral, o contexto da codificação de sinais de imagem e vídeo, utilizando a técnica de casamento parcial de padrões é muito vasto e deve ser melhor delimitado. É vantajoso trabalhar sobre um modelo baseado em transformadas de bloco, devido aos excelentes resultados obtidos, como visto no Capítulo 2. Devido à existência de técnicas já bem desenvolvidas de otimização da quantização e da seleção dos modos de codificação [76, 91, 92], nos atemos a codificação entrópica dos coeficientes já quantizados. No caso de vídeo, podemos encontrar grandes avanços das técnicas de estimação e codificação de movimento [27,50], como também de escolha dos diversos modos possíveis de codificação dos blocos [103, 121].

Foi vista no Capítulo 4 uma análise dos números de bits utilizados na codificação do mapa de significância, sinal e magnitude dos coeficientes de cada bloco transformado e quantizado de uma imagem. Observou-se que uma grande proporção dos bits é utilizada para codificar o mapa de significância. Ao mesmo tempo, conhece-se da literatura em diversos trabalhos que tratam da codificação usando mapa de significância [16, 18, 60, 96, 99, 125], que o sinal e a magnitude são de difícil compressão e os resultados obtidos possuem ganho apenas marginal. O mapa de significância é uma imagem binária, de fácil tratamento.

5.2.1 Redundâncias intra e inter bloco

Como vimos no Capítulo 2, na codificação por transformada os coeficientes quantizados formam um conjunto de dados que representam a imagem codificada. A este conjunto de dados aplica-se um codificador entrópico. O conjunto de dados possui uma estrutura

particular que pode ser aproveitada de forma a melhorar a codificação entrópica. Especificamente, no caso das transformadas de bloco, como a DCT, aplicadas às imagens naturais, os coeficientes tendem a possuir maior amplitude quando sua freqüência espacial, horizontal ou vertical, é menor. Esta estrutura pode ser vista como uma redundância intra-bloco.

Outra estrutura que pode ser considerada no projeto de um codificador é o fato de a transformada de bloco dividir o plano tempo-freqüência (no caso de imagem, espaço-freqüência), em regiões, denominadas sub-bandas, de mesma área. A posição do coeficiente no bloco transformado indica sua freqüência espacial. Os coeficientes de mesma posição de blocos adjacentes possuem, devido à regularidade e à localidade encontradas nas imagens naturais, valores correlacionados. Um codificador pode aproveitar essa redundância inter-blocos.

No entanto, podemos considerar a concepção de métodos que procurem explorar de forma conjunta as redundâncias intra e inter-blocos. O conceito envolveria encontrar alguma correlação entre blocos baseada em alguma característica de cada bloco. Um exemplo deste princípio poderia ser a busca de blocos que possuam mapas de significância de coeficientes transformados semelhantes.

O fato de estarmos explorando uma estrutura de mais alto nível, que abrange as dependências inter e intra-blocos, nos dá a esperança de conseguirmos algum ganho de codificação, com relação às técnicas tradicionais. No caso de vídeo, podemos ainda esperar que a exploração conjunta da redundância temporal inter-quadros promova também ganho de codificação. Cabe salientar que a codificação da imagem resíduo realizada desta forma, utilizando quadros anteriores como referência, não segue completamente o paradigma usual na codificação híbrida de vídeo. Naquele modelo, visto no Capítulo 2, procura-se reduzir a redundância temporal através da estimação e compensação de movimento e o resíduo desta compensação é codificado sem nenhuma referência aos demais quadros, ou seja, é tratado como uma imagem.

5.2.2 Redundâncias no mapa de significância

Como podemos observar na Figura 5.1, os elementos significativos estão na sua maioria agrupados no canto superior esquerdo de cada bloco. Este fato pode ser interpretado como uma estrutura intra-bloco, onde a significância dos coeficientes vizinhos é correlacionada, ou inter-bloco, onde a correlação ocorre entre coeficientes relativos a mesma sub-banda. Estas correlações podem ser aproveitadas de forma direta através da seleção dos contextos utilizados. A reutilização e a adaptação dos contextos entre blocos permite o aproveitamento indireto destas correlações.

Figura 5.1: Mapa de significância da imagem Lena, após a transformada DCT 8×8 e quantizada com $Q = 150$.

Desta feita, o escopo deste trabalho fica delimitado a codificar imagens e seqüências de imagens através do casamento de padrões utilizando apenas os mapas de significância de cada bloco. Além da simplicidade de tratamento, pode-se justificar a busca de padrões similares no interior da mesma imagem, formando dicionários adaptativos ou semi-adaptativos, aproveitando-se a regularidade e localidade inerente às imagens naturais, nas quais as es-

truturas tendem a se agrupar. Observam-se regiões de **textura**, identificadas pela existência de muitas bandas espectrais de alta freqüência; **bordas**, mais facilmente identificadas no domínio espacial, e as **zonas homogêneas**, onde há a predominância de baixas freqüências. Desta forma, podemos encontrar características espectrais semelhantes em blocos vizinhos e próximos de uma imagem. Como o mapa de significância revela a existência de uma determinada freqüência espacial naquele bloco com amplitude superior a um determinado limiar, é possível encontrarmos mapas de significância com certo grau de similaridade entre blocos próximos de uma imagem. Esta característica pode ser aproveitada durante a codificação.

5.2.3 Descrição resumida do método

Podemos resumir a codificação por casamento de padrões nos seguintes passos:

1. Encontre no espaço de busca (dicionário) um bloco *similar* ao atual;
2. Codifique seu índice no dicionário - utilize informações já obtidas para melhorar esta codificação;
3. Calcule e codifique o resíduo.

O espaço de busca, ou dicionário, é um conjunto de elementos pré-estabelecidos, indexados, que são varridos na busca pelo elemento mais adequado à codificação. Um exemplo de espaço de busca dependente da imagem é uma região bidimensional causal ao redor do bloco atual.

5.2.4 Considerações

A busca e a representação de padrões similares podem ser interpretadas como uma modelagem de mais alto nível das características estatísticas do conjunto de coeficientes quantizados. O nível de uma modelagem para condicionamento é relativo ao número de símbolos utilizados na formação de um contexto. Assim, como na busca de padrões

diversos elementos são varridos na procura da maior similaridade, há uma junção de informações de diversas fontes na composição dos contextos.

Um bloco de coeficientes pode ser representado por um mapa de significância, que indica que elementos são não nulos, e por suas respectivas magnitudes e sinais. No presente trabalho a ênfase é no casamento de padrões de mapas de significância. Isto é justificado pelo fato de a maior parcela dos bits serem utilizados na codificação deste mapa, para diversas taxas de bits, e a pouca estrutura apresentada pela magnitude e sinal dos coeficientes. Estes fatores foram investigados no Capítulo 4.

O CPP pode ser visto como um estágio de pré-processamento no qual os mapas de bits formam aglomerados que são casados entre si. Por exemplo, o casamento pode ser realizado através da minimização da distância de Hamming. Neste caso a diferença entre os blocos é obtida pela operação XOR. O número de bits 1 restante é o valor a ser minimizado. O resultado da operação XOR, ou seja, o resíduo, deve ser enviada juntamente com o índice do bloco no dicionário. Com esta métrica a diferença obtida possui um número menor de bits 1, o que é vantajoso. Para diversas técnicas de codificação de imagens binárias, ocorre uma relação linear entre o número de bits obtidos pela compressão e o número de bits 1 na imagem original [7]. Isto reforça a idéia de realizar um pre-processamento que reduza o número de bits 1 em um mapa binário. Para o caso específico de um mapa de significância de coeficientes DCT de imagens, é mostrado em [28] que a relação entre o número de coeficientes não nulos após a quantização e a taxa final também é aproximadamente linear.

No capítulo seguinte apresentamos a implementação de experimentos com o casamento de padrões para a codificação de vídeo.

Capítulo 6

Implementação

Neste capítulo apresentaremos as técnicas utilizadas e os detalhes de implementação do método de casamento de padrões como codificador entrópico em substituição ao código de comprimento variável VLC de um codificador MPEG-4.

6.1 Modificações no codificador MPEG-4

Como vimos no Capítulo 2, as informações de cabeçalho, de vetores de movimento e de textura (elementos DCT) são codificadas no MPEG-4 através de um código de comprimento variável. Na codificação dos elementos DCT utiliza-se também de código de comprimento de seqüencia e denominamos este codificador como RL+VLC. Implementamos um codificador baseado em casamento de padrões em substituição ao RL+VLC. Observa-se que apenas a textura, ou seja, os elementos DCT dos blocos são codificados com CPP.

Para realizar esta substituição de codificador, foram realizados dois tipos de implementação: (a) inserção de um sistema de coleta dos elementos dos blocos DCT e gravação em arquivo para tratamento externo e (b) inclusão direta dos dados codificados com CPP e código aritmético no *bitstream* do MPEG-4.

A implementação (a) é mais simples porque não envolve mudanças no processamento

do codificador MPEG-4. Uma vez retirados os elementos DCT, obtemos uma imagem digital com as mesmas dimensões dos quadros de vídeo com 16 bits de precisão por elemento. Este arquivo gerado é processado por um programa de casamento de padrões que gera um *bitstream* (arquivo) de saída. Este arquivo pode ser decodificado e deve levar ao mesmo resultado do arquivo original uma vez que estamos trabalhando com codificação (compressão) sem perdas.

Este método possui como vantagem além da simplicidade o fato de podermos realizar vários testes de sintonias de parâmetros do codificador e avaliar de forma simples o resultado, o tamanho do arquivo de saída. Além disso, o arquivo de elementos pode ser usado para fazer uma verificação rápida da taxa de compressão obtida com o uso de codificadores universais, como Ziv-Lempel, PPM ou BWT [6], facilmente encontrados, em comparação com a codificação dedicada realizada pelo MPEG-4 e pelo CPP.

Como desvantagens deste método podemos citar o fato de se perder a hierarquia entre macroblocos e blocos e, no caso de CPP, os quadros utilizados na construção do dicionário precisam ser aplicados juntamente com o quadro atual no codificador.

Na implementação (b) devemos observar cuidadosamente que elementos de sintaxe (cabeçalho geral, cabeçalhos dos macroblocos, vetor de movimento, textura, etc) devem ser codificados. Uma vez que o bitstream utilizado é o mesmo do MPEG-4 e como não se pretendia trocar a codificação de todos os elementos de sintaxe, faz-se necessário um processo de multiplexação de bitstreams. Por sorte, a codificação dos macroblocos, os únicos elementos de sintaxe codificados com CPP, são sempre os últimos campos a serem codificados no MPEG-4. Assim, o cabeçalho global é codificado diretamente usando as tabelas originais do MPEG-4. Não há mudança no código desta parte. A codificação dos macroblocos foi repetida para a inserção do código de CPP e codificador aritmético. Para tanto, utilizou-se um modo especial do codificador aritmético utilizado, o CABAC, visto mais adiante. Neste modo, chamado modo *skip*, codifica-se elementos em que se sabe que a probabilidade é perto de 0,5 e não é preciso realizar adaptação. Este modo é utilizado para codificar todos os bits que não sejam elementos DCT. Entre eles encontramos os elementos de cabeçalho dos macroblocos. Todos estes outros elementos são codificados

no modo *skip* e utilizam exatamente o mesmo número de bits que utilizariam se fossem codificados diretamente. Isto ocorre porque as tabelas de codificação VLC ainda são utilizadas, mesmo que não para saída direta dos bits, mas para inserção no codificador aritmético.

Os dados dos elementos DCT são os últimos codificados em um macrobloco. Nos macroblocos *inter* é necessário também codificar os vetores de movimento. Utilizou-se para tal o mesmo método de outros elementos de sintaxe, ou seja, aproveitar a tabela VLC e enviar os bits obtidos diretamente para o codificador aritmético.

6.2 Metodologia

A técnica de CPP foi avaliada em uma arcabouço de codificação de vídeo compatível com o padrão MPEG-4. Como vimos no Capítulo 2, os quadros do tipo *intra* são codificados como imagens, sem referência a outros quadros. Assim, utilizando o mesmo arcabouço podemos obter resultados para imagem (*intra*) e vídeo (*inter*).

Foram codificados o plano de luminância Y e nos planos de crominância Cb e Cr, tanto nos quadros *intra* quanto *inter*.

O CPP utiliza como etapa final um codificador aritmético otimizado para dois símbolos com possibilidade de seleção adaptativa de contextos [6, 123], o CABAC, utilizado no recente padrão de codificação de vídeo H.264, também conhecido com MPEG-4 AVC ou parte 10 [75]. Esta possibilidade de adaptação de contextos, se bem explorada, pode proporcionar uma diminuição no número de bits utilizados. No CPP, a seleção dos contextos foi realizada de forma similar ao CABAC original. Algumas modificações foram necessárias devido ao fato de que o padrão H.264 trabalha com blocos de DCT 4×4.

No CABAC, no lugar de contextos temos estados, que representam a probabilidade dos símbolos de forma aproximada. No H.264 são utilizados 276 estados para codificar todos os possíveis elementos de sintaxe [70]. Na implementação realizada aqui, os estados não são utilizados para codificar os elementos de sintaxe a menos dos coeficientes DCT. Desta forma, utilizamos um conjunto completamente diferente de estados. O conjunto de estados

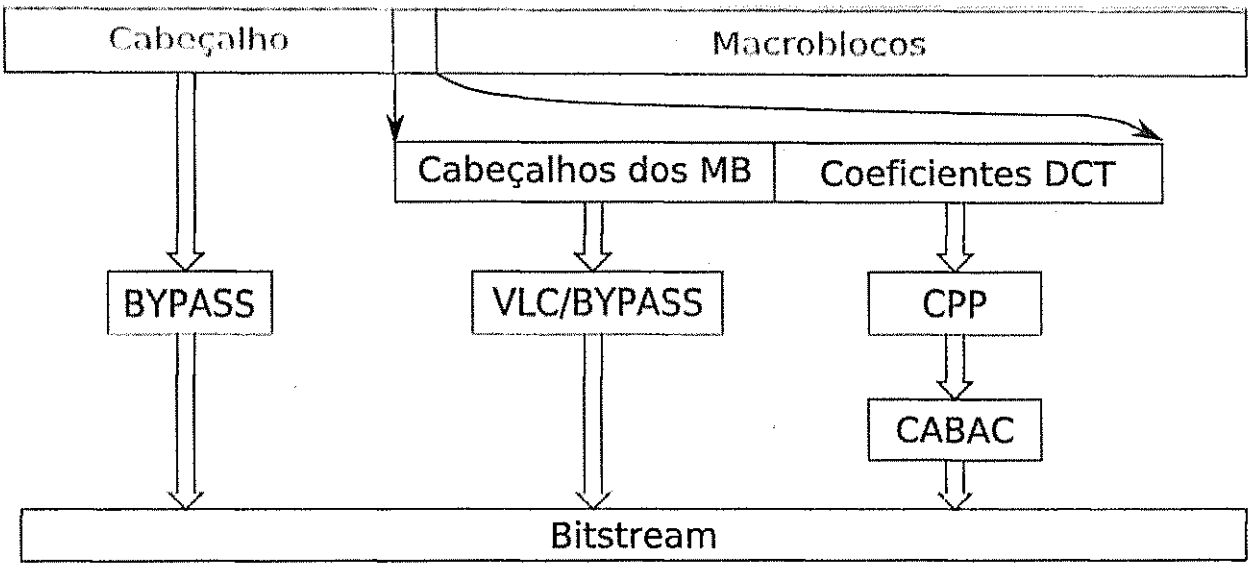


Figura 6.1: Modificações realizadas na etapa de codificação entrópica do MPEG-4 para acomodar a técnica CPP e o CABAC.

utilizados aqui pode ser visto na Tabela 6.1.

O valor dos coeficientes é codificado em dois passos. Em primeiro lugar é realizada uma binarização do valor utilizando um código exponencial de Golomb truncado [70]. Em seguida os *bins* obtidos são codificados com os estados selecionados. Nesta implementação o sinal dos coeficientes é codificado utilizando um modo especial do CABAC para símbolos não compressíveis. Isto acontece porque o sinal é considerado descorrelacionado.

Os índices encontrados durante a busca de casamento de padrões também são codificados utilizando código exponencial de Golomb truncado. Graças ao uso do *heap* como estrutura de dados de armazenamento do dicionário e da utilização da freqüência de uso como chave de busca, observa-se uma distribuição em que os elementos de índices menores ocorrem com muito mais freqüência.

Os demais elementos de sintaxe são codificados utilizando as próprias tabelas VLC do MPEG-4. Os elementos de cabeçalho são codificados diretamente para o bitstream de saída. Os elementos de sintaxe que fazem parte dos macroblocos, como a indicação de codificação, o campo CBP, os vetores de movimento passam pelo codificador CABAC no

Tabela 6.1: Estados associados com elementos de sintaxe.

Elementos de sintaxe	Estado
Valor dos coeficientes	0 - 19
significância sem CPP	20-83
último sem CPP	100-163
significância com CPP	180 - 313
Índice do CPP	320 - 336

modo SKIP, ou seja, com probabilidade 0,5, sem atualização de probabilidades e sem uso de estados.

6.3 Estrutura de dados do dicionário

Como estrutura de dados para armazenar o dicionário foi implementado um *heap*. Um *heap* é uma estrutura de dados que armazena coleções de objetos (com chaves de busca) e tem as propriedades de ser uma árvore binária completa e ter uma ordenação de *heap*. É normalmente implementada como uma matriz em que cada nó da árvore corresponde a um elemento da matriz.

Uma árvore binária completa é uma árvore binária que está preenchida completamente em todos os seus níveis, com uma possível exceção do último nível, que é preenchido da esquerda para a direita.

A propriedade de ordenação de *heap* pode ser definida como para cada nó v, que não seja a raiz, a chave armazenada em v é maior ou igual (ou menor ou igual) a chave armazenada no nó pai de v. Como exemplo, temos uma *heap* de máximo, em que o maior valor é armazenado na raiz e temos nos descendentes valores sempre menores, conforme ilustra a Figura 6.2.

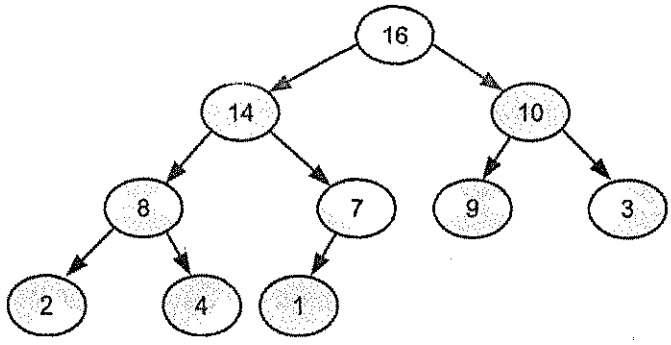


Figura 6.2: *Heap* de máximo.

1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

Figura 6.3: *Heap* armazenado em um arranjo.

6.3.1 Armazenamento do Heap em um arranjo

Um heap pode ser implementado diretamente utilizando uma estrutura de dados como uma lista encadeada, como normalmente se faz para as estruturas de árvore. No entanto, normalmente é relativamente simples implementar um heap sobre um arranjo (*array*). Para isso basta numerar os nós de um heap de cima para baixo, onde em cada nível, da esquerda para a direita os nós são numerados em ordem crescente. O i-ésimo número de cada nó corresponde a i-ésima posição no arranjo. O exemplo visto na Figura 6.2 pode ser armazenado em um arranjo como visto na Figura 6.3. Neste caso, a raiz da árvore binária é armazenada em $A[0]$, seu descendente esquerdo em $A[1]$ e seu descendente direito em $A[2]$, e assim por diante.

Para caminha-se pela árvore, podemos obter os índices do nó pai, do nó esquerdo e do nó direito de um elemento de índice i. O índice do nó pai é $i/2$; do nó esquerdo $2i$; e do nó direito $2i + 1$.

6.3.2 Operações com *heap*

A altura de um nó em uma árvore é definida como o maior caminho que pode ser percorrido do nó até uma folha. Uma folha é um nó terminal, sem descendentes. A altura de um *heap* de n elemento armazenado em uma árvore binária é $\log n$. As operações básicas em um *heap* são normalmente proporcionais à altura da árvore e têm complexidade $O(\log n)$.

As principais operações em um *heap* são a construção, inserção de um item, remoção de um item e busca. Estas operações normalmente dependem de uma operação básica que mantém a propriedade de ordenação de um *heap*. Chamaremos esta operação de ordenação. Em inglês esta operação é conhecida com *heapify*.

Ordenação

A operação de ordenação converte uma árvore binária em um *heap*. Suas entradas são um arranjo A e um índice i neste arranjo. A operação de ordenação ocorre na sub-árvore cujo nó de índice i é a raiz. É requisito do algoritmo que as duas sub-árvores direita[i] e esquerda[i] já sejam *heaps*. No entanto o valor do nó $A[i]$ pode ser maior do que seus descendentes, violando a propriedade de *heap*. A função de ordenação permite que o valor de $A[i]$ desça para um das sub-árvores de tal forma que a sub-árvore com raiz em i seja agora um *heap*. Pode-se demonstrar, usando método iterativo ou o teorema mestre que o tempo de execução, $T(n)$, do algoritmo de ordenação é dado por $T(n) = O(\log n)$. O algoritmo de ordenação pode ser visto na Figura 6.4.

Construção de um *heap*

Basicamente podemos construir um heap usando algoritmo de inserção começando com uma árvore vazia ou usar o algoritmo a seguir, visto na Figura 6.5. Neste algoritmo a rotina de ordenação é chamada de baixo para cima na árvore. Esta ordem de chamada garante que as sub-árvores com raiz no nó i são *heaps* antes de chamar a ordenação neste nó.

Cada chamada à ordenação custa tempo $O(\log n)$, sendo chamada em $O(n)$. O tempo

Entrada: Um arranjo A

Entrada: índice i

Saída: Sub-árvore com raiz em $A[i]$ ordenada

```
1  início
2      esquerdo ← FilhoEsquerdo(i);
3      direito ← FilhoDireito(i);
4      se esquerdo ≤ A.tamanho ∧ A[esquerdo] > A[i] então
5          maior ← esquerdo;
6      senão
7          maior ← i;
8      se direito ≤ A.tamanho ∧ A[direito] > A[maior] então
9          maior ←direito;
10     se maior ≠ i então
11         A[i] ← A[maior];
12         Chama ordena recursivamente;
13         Ordena(A,maior)
14 fim
```

Figura 6.4: Ordenação de um *heap*

total é no máximo em $O(n \log n)$. É possível fazer uma análise mais complexa e obter um limite superior de $O(n)$. Ou seja, a construção de um *heap* pode ser realizado em tempo linear.

Entrada: Um arranjo A

Saída: A como *heap*

```
1 início
2     A.tamanho ← dim(A);
3     para i = ⌊dim(A)/2⌋ decrescendo até 1 faça
4         Ordena(A,i);
5 fim
```

Figura 6.5: Constrói *heap* A

6.3.3 *heap* como uma fila de prioridade

Freqüentemente encontramos aplicações com prioridade entre chaves, ou seja, com ordenação parcial entre as chaves. Um exemplo é o escalonamento de tarefas para execução pela CPU. Outros exemplos são simulação de eventos, matrícula de alunos (concluintes têm prioridade), salas de emergência em um hospital.

Basicamente, em uma fila de prioridade os ítens são colocados na fila em qualquer ordem, mas o processamento ocorre sempre com o a chave de maior prioridade ou valor.

Um *heap* implementa diretamente uma fila de prioridade simples. Uma fila de prioridade deve implementar pelo menos duas operações: inserir um novo item e remover o item com a maior chave. Outras operações também podem ser suportadas, tais como a construção de uma fila de prioridades a partir de uma outra estrutura de dados, mudança de prioridades e remoção de ítens arbitrários. Vejamos em seguida os algoritmos para a

inserção e extração de máximo(ou mínimo) em uma fila de prioridade armazenada em um *heap*.

Inserção

O algoritmo para inserção pode ser visto na Figura 6.6.

Entrada: Um *heap A*

Entrada: Objeto

```
1 início
2 |  A.tamanho ← A.tamanho + 1;
3 |  i ← A.tamanho;
4 |  enquanto i > 1 ∧ A[pai(i)] < objeto faça
5 |  |  A[i] ← A[pai(i)];
6 |  |  i ← pai(i) ;
7 |  A[i] ← objeto;
8 fim
```

Figura 6.6: Extrai o máximo do *heap* A

O custo total é O($\log n$).

Obtenção e remoção do valor máximo de um *heap*

O algoritmo para inserção pode ser visto na Figura 6.7.

Entrada: Um arranjo A

Saída: Elemento máximo de A, $\max A$

```
1 início
2     se A.tamanho < 1 então
3         Erro;
4     maximo ← A[1];
5     A[1] ← A[A.tamanho];
6     A.tamanho ← A.tamanho − 1;
7     Ordena(A, 1);
8     retorna maximo;
9 fim
```

Figura 6.7: Extrai o máximo do *heap* A

Observa-se que a inserção de um objeto segue o percurso de uma folha até a raiz da árvore. Como a altura da árvore é dado por O(log n), obtemos o tempo de execução total de O(log$n$).

6.4 Implementação do dicionário para casamento de padrões

Na implementação do *heap* para o casamento de padrões, a chave utilizada foi o número de vezes em que determinado padrão foi utilizado na codificação de algum outro bloco. Cada vez que um bloco é identificado no dicionário como o mais adequado à codificação

de outro bloco, incrementa-se o contador de utilização e realiza-se um rearranjo de todo o *heap*, para manter a propriedade de ordem de *heap*. Como vimos, esse rearranjo é denominado *heapify*.

Para manter a ordem de *heap*, os valores de todos os filhos de um nó devem ser menores que o valor armazenado nesse nó.

A manutenção dos elementos mais utilizados com índices menores do que os elementos menos utilizados permite a aplicação de métodos de codificação de inteiros, como os códigos de Golomb-Rice [7], com grande eficiência.

Os mapas binários dos blocos 8×8 foram codificados como inteiros de 64 bits. Isto permite uma comparação mais rápida dos valores, através de uma única operação XOR em máquinas que possuam um tipo de dado de 64 bits.

6.5 Mecanismo de busca

Para determinarmos o mecanismo de busca e armazenamento no dicionário é necessário definirmos uma métrica de distância entre os símbolos armazenados. Nesta implementação foi utilizada a distância de Hamming obtida pela contagem do número de bits 1 residuais a utilização da operação XOR. Como visto no Capítulo 4 o número de bits gastos na representação de uma mapa binário possui alta correlação com o números de posições significativas (com valor 1) neste mapa. Como no CPP o número de bits necessários para representar um bloco é a soma do número de bits gasto na representação do índice no dicionário com o número de bits gastos para representar os resíduos, foi inserido no mecanismo de busca uma aproximação do números de bits devido a codificação do índice como sendo o número de níveis no *heap*.

Esta aproximação forma, efetivamente, um mecanismo de polarização que penaliza a utilização da utilização do processo de CPP, haja visto, que para utilizá-lo é necessário também o envio do índice. Esta heurística pode, através de sintonia de valores, ser aperfeiçoada, para obtermos menores números de bits.

De posse da aproximação do número de bits utilizados para codificar utilizando CPP,

o codificador pode decidir se irá codificar cada bloco de forma direta ou utilizando algum elemento do dicionário como predição.

Os padrões de codificação de vídeo determinam de forma rígida sobre a formatação dos bits que descrevem uma seqüencia compatível com o padrão, mas são flexíveis sobre quais ferramentas ou técnicas são usadas pelo codificador. Como exemplo observamos que o codificador decide o tipo de quadro, o tipo de macrobloco, a extensão da área de busca para estimação e compensação de movimento, etc. Desta forma, para um mesmo padrão observamos codificadores que possuem curvas diferentes de taxa-distorção. De forma similar, no caso do casamento de padrões o mecanismo de busca é responsável diretamente pela eficiência do método, juntamente com a estrutura de dados do dicionário. Podemos obter diferentes codificadores com eficiências diferentes. A implementação descrita neste capítulo não é única. Diversas formas de casamento parciais de padrões podem ser sugeridas e são um excelente tema para pesquisa.

Esta flexibilidade de que técnicas pode ser utilizada também permite tornar os codificadores baseados em casamento de padrões mais abrangentes do que os codificadores tradicionais. Como o casamento de padrões pode ser seletivamente desabilitado por unidade de codificação (no caso tratado, bloco transformado), podemos supor que a sua utilização, no pior caso, pode aumentar de apenas alguns bits o vídeo codificado. Nos casos comuns, no entanto, espera-se redução no número de bits totais utilizados.

No capítulo seguinte veremos os resultados obtidos através da codificação baseada em casamento de padrões e código aritmético em um arcabouço de codificação completo baseado no padrão MPEG-4.

Com essa correção obteve-se uma redução no número total de bits utilizados, como pode ser observado nos resultados dos testes apresentados no capítulo seguinte.

Capítulo 7

Resultados

Neste capítulo veremos o resultado da aplicação da técnica de casamento parcial de padrões para a codificação sem perda ou entrópica dos coeficientes que resultam da aplicação das transformadas de bloco em imagens e vídeo.

7.1 Seqüências utilizadas

As primeiras imagens das seqüências utilizadas para a realização dos testes de compressão podem ser vistos nas figuras 7.1 (*Foreman*), 7.2 (*Miss America*) e 7.3 (*Flower Garden*). As seqüências Foreman e Miss America estão no formato CIF, que corresponde a 352 × 288 pixels. A seqüência Flower Garden está no formato SIF, que possui 352 × 240 pixels. Estas seqüências estão no formato YCbCr 4:2:0 com 8 bits de precisão. As figuras mostram apenas o canal de luminância.

7.2 Resultados de casamento parcial de padrões

Para avaliar se os blocos transformados de uma imagem possuem mapas de significância similares dentro de uma certa vizinhança espacial, foi realizado um experimento de busca

Figura 7.1: Primeiro quadro da seqüência *Foreman*. Movimentação rápida do objeto principal com um fundo estático. Formato CIF.

Figura 7.2: Primeiro quadro da seqüência *Miss America*. Cena típica de vídeo-conferência, mostrando cabeça e ombros, com muito pouca movimentação. Formato CIF.

Figura 7.3: Primeiro quadro da seqüência *Flower Garden*. Cena com movimentação rápida de toda a imagem devido à movimentação da câmera com relação à imagem. Cena com muitas texturas. Formato SIF.

de padrões. Os testes foram realizados numa vizinhança causal de 8×8 blocos. A métrica utilizada foi a distância de Hamming. O total de de bits 1 antes e após a operação XOR é relatada na Tabela 7.1, na codificação da imagem Bárbara para algumas taxas de bits.

Taxa (bpp)	Razão compressão	Coeficientes significantes	Após XOR
1,00	1:8	46052	21132
0,50	1:16	22455	10153
0,25	1:32	10875	3912

Tabela 7.1: Avaliação do número de posições significantes antes e após a aplicação do casamento de padrões.

Nas Figuras 7.4, 7.5 e 7.6 observamos a codificação do primeiro quadro das seqüências de imagens *Miss America*, *Foreman* e *Flower Garden*, respectivamente, utilizando o casamento parcial de padrões.

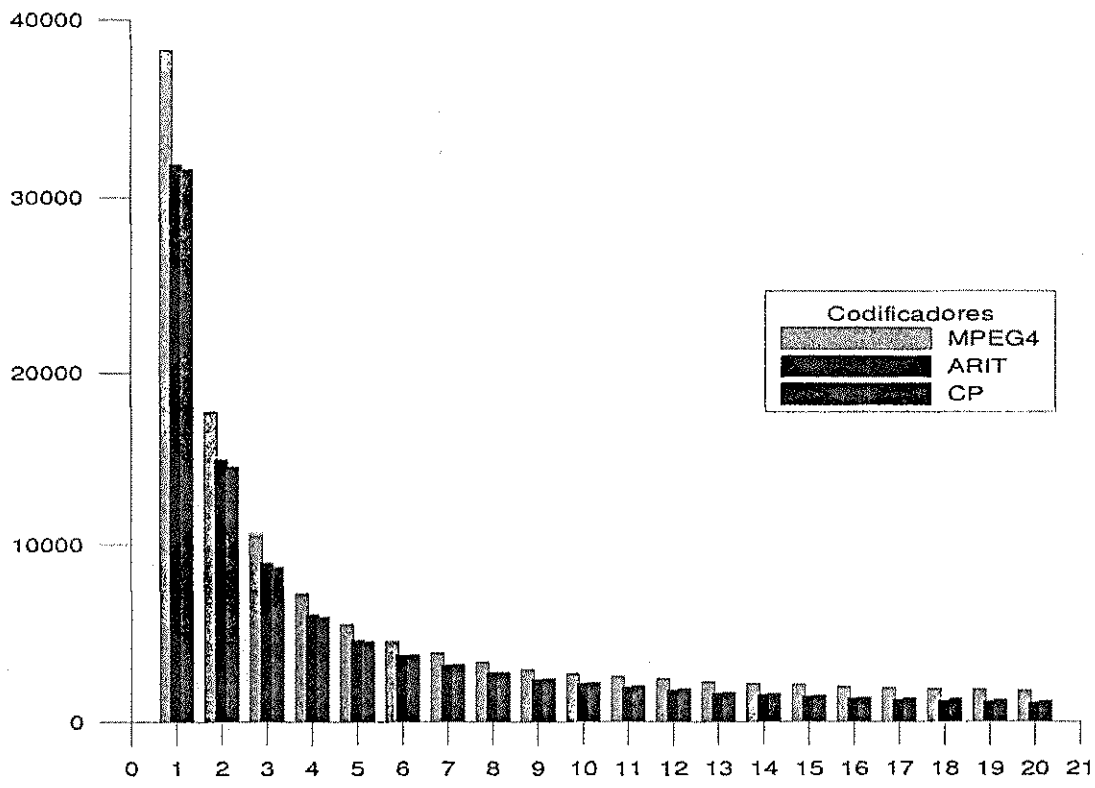


Figura 7.4: Codificação intra do primeiro quadro da seqüência *Miss America*, utilizando CPP. O gráfico representa o número de bytes utilizados na textura de acordo com o parâmetro de qualidade Qp.

Observamos no gráfico da Figura 7.4 pouco ganho de codificação com a utilização do codificador aritmético, como também pouquíssima variação com a inclusão do passo adicional de casamento de padrões. Esta seqüência possui muito pouca variação na imagem, sendo portanto de fácil codificação, permitindo ao codificador MPEG-4 obter excelentes resultados que não podem ser muito melhorados. Para este tipo de seqüência normalmente o codificador marca muitos macroblocos como *SKIP*, saltado, ou seja, completamente semelhante ao macrobloco co-sitiado na imagem anterior. Desta forma, não é necessário mais nenhum bit adicional para descrever o macrobloco. Os macroblocos saltados não são contabilizados como textura.

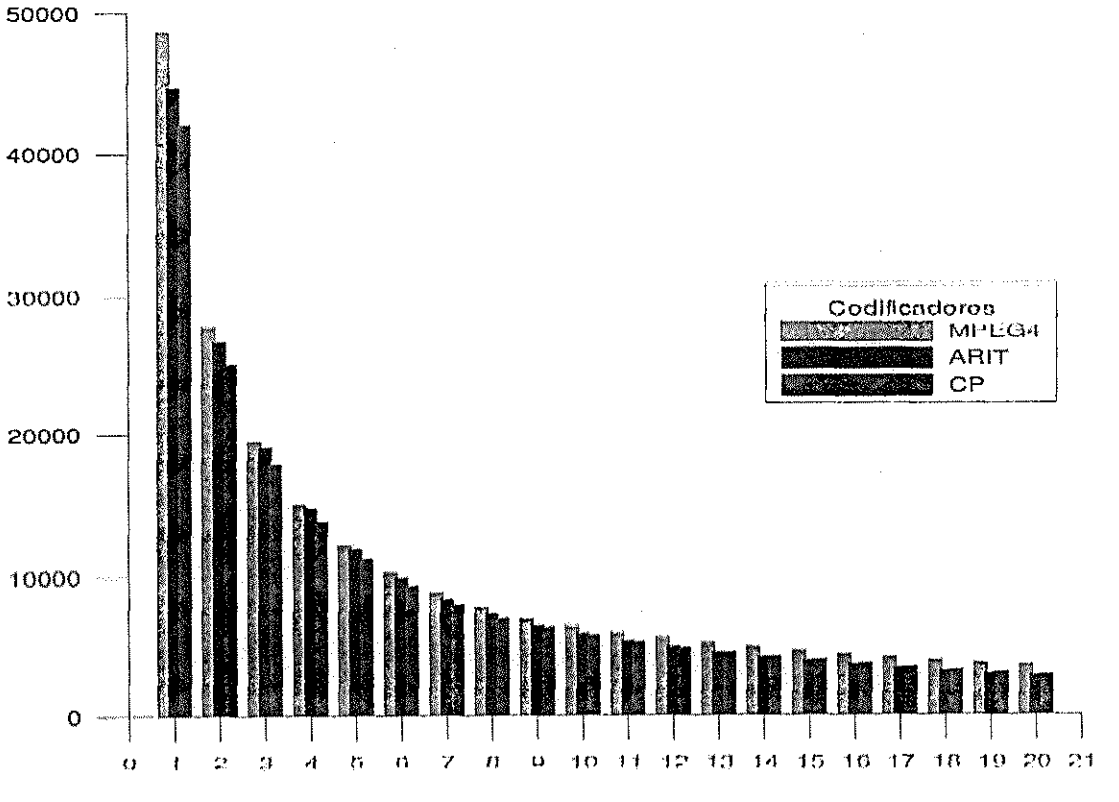


Figura 7.5: Codificação intra do primeiro quadro da seqüência *Foreman*, utilizando CPP. O gráfico representa o número de bytes utilizados na textura de acordo com o parâmetro de qualidade Qp.

Para a seqüência *Foreman* observamos que o codificador aritmético é capaz de obter

reduções maiores no número de bits utilizados. O passo de casamento de padrões possibilitou também uma redução mais significativa. Observamos que só há benefício da utilização do casamento de padrões para altas taxas, ou seja, para parâmetros de quantização baixos. Para taxas mais baixas, o número de coeficientes não nulos é pequeno. Embora seja possível encontrar padrões de semelhança nestas condições, o custo, ou seja, o número de bits usados para representar o padrão normalmente é maior do que para a codificação do bloco original. Nas taxas mais altas, há um número elevado de coeficientes significativos. Através do casamento de padrões é possível diminuir o número de bits significativos do mapa de significância.

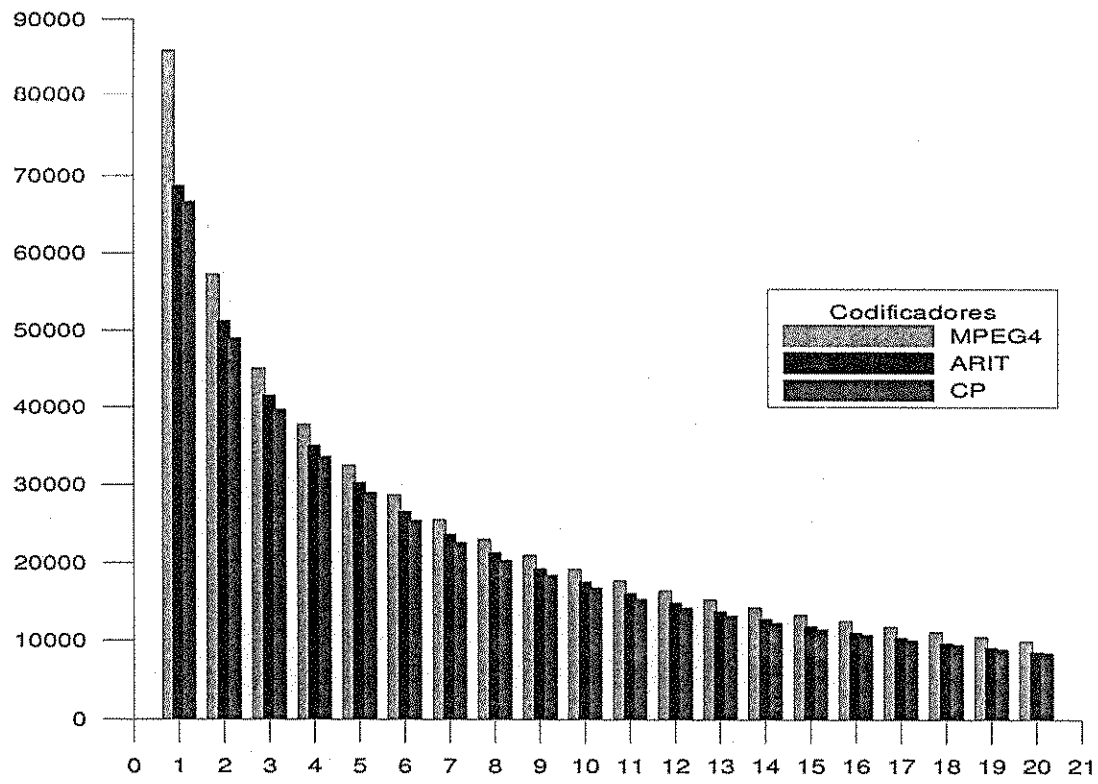


Figura 7.6: Codificação intra do primeiro quadro da seqüência *Flower Garden*, utilizando CPP. O gráfico representa o número de bytes utilizados na textura de acordo com o parâmetro de qualidade Qp.

7.3 Conclusões

Neste capítulo vimos alguns resultados selecionados da aplicação da técnica de CPP na codificação para transformadas de blocos.

No CPP, os resultados obtidos com este reagrupamento foram bons, principalmente para imagens (quadros *intra*) e resíduos (quadros *inter*) com muita textura e altas taxas. Em comparação com o codificador encontrado no padrão MPEG-4, pode-se observar que o CPP consegue realizar um melhor trabalho de codificação quando o número de coeficientes significativos é grande, sendo o MPEG-4 mais eficiente quando os blocos de coeficientes são esparsos.

Observa-se um ganho expressivo na codificação com a utilização apenas do CABAC sobre os coeficientes. Este resultado é de certa forma esperado por ser consistente com a literatura. Foram relatadas reduções no número de bits utilizados de cerca de 15% com o CABAC sobre o VLC no H.264. Observe-se que o H.264 utiliza um método especial de VLC chamado CAVLC, em que são utilizadas diversas tabelas para a codificação dos coeficientes DCT de forma adaptativa por contextos.

Capítulo 8

Conclusão

Apesar do aumento da capacidade de armazenamento e de transmissão dos equipamentos mais modernos, observa-se a exigência de métodos de compressão de sinais de multimídia cada vez mais eficientes, para promover novas formas de utilização antes impossíveis ou impróprias. Neste contexto estão as aplicações de comunicação móvel audiovisual, a televisão e o DVD de alta definição, o cinema digital, etc. A busca de métodos mais eficientes de codificação tem promovido um grande avanço nos domínios do processamento digital de sinais e da teoria da informação. Os avanços mais significativos atualmente estão associados ao uso de transformadas mais eficientes, para a codificação de imagem e áudio. Para o caso de vídeo, temos recentemente o aparecimento do novo padrão H.264/AVC/MPEG-4 parte 10, que inova nos métodos de estimação e compensação de movimento, transformada de bloco reversível e de tamanho reduzido, divisão e codificação hierárquica dos macroblocos e o uso de um codificador aritmético como estágio final de codificação.

Neste trabalho avaliamos técnicas eficientes para a codificação sem perda de coeficientes quantizados de transformadas de bloco, aplicados à compressão de imagem e vídeo. O *casamento parcial de padrões* (CPP), apresentado nesta tese, que pode como uma forma de pre-processamento para o codificador aritmético, ou como uma formulação de contextos de nível mais alto do que o usual, reduz sensivelmente o número de bits necessários para a

codificação sem perda, aumentando a razão de compressão. Este mecanismo não aumenta de forma significativa a complexidade do codificador nem do decodificador. Além desta técnica analisada com mais profundidade neste documento, outros estudos realizados até o momento resultaram na concepção de novas técnicas para a codificação de coeficientes quantizados em transformadas de bloco: o SPQ e o CBPB [74]. Estas técnicas de codificação foram avaliadas através de resultados de simulações envolvendo codificação de imagens e vídeo em um arcabouço de um codificador MPEG-4, modificado para acomodar estes algoritmos. Os resultados obtidos foram comparados com os do RL+VLC, utilizado normalmente no MPEG-4.

Este trabalho contempla uma avaliação comparativa de desempenho dos algoritmos CPP, e RL+VLC primariamente no aspecto da taxa de bits obtida, haja visto que por se tratarem de métodos de compressão sem perda, os demais aspectos, tal como a qualidade dos sinais reconstruídos, são mantidos. Alguns outros aspectos são considerados, tais como a complexidade computacional envolvida nas fases de codificação e de decodificação, a capacidade de adaptação e a adequação a sinais de outras origens.

Os resultados obtidos são importantes e sugerem que, com algumas otimizações do método seja possível obter uma redução ainda mais significativa do número de bits utilizados na codificação sem perda dos coeficientes transformados.

Os resultados dos experimentos demonstram que uma grande parte dos bits é utilizada para a codificação do mapa de significância, enquanto que uma parcela normalmente menor é utilizada para codificar o sinal dos coeficiente e a sua amplitude em valor absoluto. Estas quantidades são tipicamente imcompressíveis, ou seja, possuem alta entropia, o que nos permite focalizar o trabalho de casamento de padrões apenas no mapa binário de significância. Por ser binário, o mapa de significância é fácil de ser tratado, levando a algoritmos rápidos de busca e de operação.

Em seguida apresentamos algumas das contribuições originais deste trabalho e finalizamos com possíveis trabalhos futuros que podem dar continuidade a pesquisa.

8.1 Contribuições

Neste trabalho de tese podemos elencar algumas contribuições originais.

Foi introduzido o método de casamento de padrões. Este método constitui um estágio adicional para a codificação sem perda de coeficientes transformados. Como tal acrescenta mais um grau de liberdade no codificador, que pode seletivamente incluir ou não o casamento, a nível de macrobloco. Este método introduzido permite a redução no número de bytes utilizados sem nenhuma perda de qualidade. O método introduzido, embora não seja classificado como um dos métodos tradicionais de codificadores entrópicos permitiu esta redução. Uma classificação possível é então como um pré-processamento dos dados oriundos de transformadas de bloco, tornando estes dados mais adequados a aplicação a um codificador entrópico.

Para realizar os experimentos práticos, foi realizada a modificação de um codificador MPEG-4 existente. Esta modificação permitiu a análise de novos algoritmos de codificação baseados em casamento de padrões. Também foi possível a execução de todos os experimentos em um ambiente único, garantindo a exatidão na contagem de bytes utilizados e na medição de tempo de execução.

Outra contribuição original foi a inclusão do codificador CABAC, normalmente utilizado no padrão H.264, em um codificador MPEG-4. O CABAC teve que ser modificado para incluir os elementos de sintaxe do MPEG-4 e retirar os elementos de sintaxe únicos do H.264. Esta inclusão permitiu realizar testes originais de desempenho do MPEG-4 aplicado a um codificador aritmético em substituição ao codificador baseado em código de comprimento variável.

O método de casamento de padrões foi analisado tanto em quadros *intra* quanto *inter*. Isto permitiu extrapolar o método para codificadores de imagem, haja vista que um codificador de vídeo no modo intra é basicamente um codificador de imagens.

A estrutura de dados sugerida para a implementação do casamento de padrões foi o *heap*. O *heap* foi utilizado como dicionário de armazenamento dinâmico dos dados na implementação experimental. Foi realizada então uma avaliação do *heap* com relação a complexidade computacional.

O método de casamento de padrões não especifica como deve ser realizado o cálculo da distância entre os padrões e o método de busca. Neste trabalho foi realizada uma avaliação da distância de Hamming como métrica para a diferença dos padrões. Como a inclusão do método precisa ser sinalizada a nível de macrobloco ou bloco, para que a métrica seja mais realista quanto a quantidade de bits utilizados, a distância de Hamming foi acrescida de valores fixos, determinados experimentalmente. Através da inclusão destes fatores fixos, as decisões de inclusão ou não do casamento de padrões levaram a um melhor resultado na redução no número de bits utilizados para a codificação.

Um destes fatores fixos utilizados foi o valor aproximado do número de bits gastos para codificar o índice dos dicionários no cálculo da distância a ser minimizada. Desta forma, é possível a codificação direta do índice do dicionário utilizando um código que atribua palavras mais curtas para valores inteiros positivos menores. Diversos códigos especializados para codificação de inteiros são sugeridos na literatura. Entre estes destaca-se a classe de códigos Golomb-Rice. O armazenamento do número de bits como índice permite então a utilização destes códigos, geralmente muito mais simples.

Algumas outras métricas além da distância de Hamming também foram sucintamente analisadas para a utilização na definição da distância a ser minimizada na codificação dos blocos.

O método de casamento de padrões possui muitos parâmetros que precisam ser melhor investigados, buscando-se inclusive formalizar e encontrar limites de forma que se possa buscar métodos otimizados.

8.2 Perspectiva de trabalhos futuros

Entre os possíveis trabalhos que podem dar continuidade à pesquisa iniciada com esta tese, podemos citar a realização de otimização taxa-distorção, outras formas de codificação do resíduo decorrente do casamento, a aplicação do método a outros tipos de dados e a abordagem de outras estruturas de dados para a representação do dicionário.

Os codificadores de vídeo apresentam uma resposta típica entre a taxa de bits utilizada

e a distorção obtida (obviamente após a decodificação) do vídeo codificado. Esta curva taxa-distorção é dependente do sinal de vídeo de entrada. A resposta taxa distorção de um codificador que segue determinado padrão não é determinada a priori e depende das decisões de implementação. Por exemplo, dois codificadores diferentes e compatíveis com o padrão MPEG-4, podem possuir respostas diferentes. As decisões de implementação refletem-se no uso das ferramentas de codificação permitidas pelo padrão e que o codificador utiliza. Então a resposta geral do codificador depende das decisões tomadas sobre, por exemplo, se um quadro deve ser codificado como *intra* ou *inter*, se um macrobloco deve ser pulado ou codificado como idêntico ao bloco colocalizado no quadro anterior, etc. Além disso, com a inclusão do casamento parcial de padrões, a curva taxa-distorção do codificador MPEG-4 é modificada.

Dado um padrão de codificador, então é possível obter escolhas de ferramentas que otimizem a resposta taxa-distorção. Com a inclusão do casamento parcial de padrões, novas técnicas precisam ser analisadas para obter esta otimização.

O CPP apresenta a peculiaridade de obtenção da resposta sem perda através da codificação em dois passos. O primeiro passo indica o bloco com maior semelhança (sob uma métrica) através de um índice de dicionário. O passo seguinte codifica o resíduo, que seria a diferença entre o bloco que está sendo codificado e o bloco cujo índice foi enviado no passo anterior. Este resíduo possui características estatísticas diferentes de um bloco decorrente da encontrada nos elementos transformados, através da DCT de um bloco de quadros *intra* ou *inter* de um vídeo.

Neste trabalho contemplamos a codificação deste resíduo de CPP através de codificadores normalmente otimizados para elementos transformados. Esta diferença de estatística promove uma sub-optimalidade na codificação dos resíduos. Sugere-se portanto, como trabalho futuro, investigar outras formas de organizar os codificadores entrópicos para uso específico em resíduos de casamento parcial de padrões. Em particular, a ordem de varredura, normalmente zig-zag, utilizada pelos codificadores tradicionais, pode não ser adequada, devido ao fato de que os resíduos de CPP possuirem geralmente as posições de baixa freqüência (elementos próximos do $(0,0)$) vazias. Ou seja, um codificador para

resíduos de CPP poderia, em princípio, ser mais eficiente se começar a varredura da última posição significante do bloco de referência e fazer a varredura nos dois sentidos, para frente e para trás.

Outro trabalho de continuação desta tese que pode render bons frutos é a implementação da técnica a outros tipos de dados também codificados com transformada de bloco, como os encontrados nos diversos padrões de codificação de áudio, i.e. MP3, AAC, etc.

Além disso, um ponto que merece um estudo mais profundo é a estrutura de dados utilizada para armazenar o dicionário de referência. A atual tese contempla apenas o uso do *heap* para realizar este armazenamento. A depender da métrica utilizada para determinar as distâncias entre o bloco a ser codificado e as referências, é possível encontrar outras formas de armazenamento mais adequadas, do ponto de vista de espaço e tempo de busca e inclusão de novos dados.

Apêndice A

SPIHT pós-quantização

A.1 Introdução

A compressão de sinais, cujo objetivo fundamental é reduzir o número de bits necessários para representar adequadamente os sinais (voz, imagem, áudio, vídeo), desempenha um papel importante em aplicações que necessitam minimização dos requisitos de largura de faixa e/ou de capacidade de armazenamento, tais como: sistemas multimídia, redes digitais de serviços integrados, videoconferência, sistemas de resposta vocal, correio de voz, difusão de música, facsímile de alta resolução, televisão de alta definição (HDTV, *high definition television*), telefonia móvel, sistemas de armazenamento de imagens médicas e de impressões digitais e transmissão de imagens de sensoriamento remoto obtidas por satélites.

Em se tratando de compressão de imagem e vídeo, diversas técnicas podem ser utilizadas, destacando-se, por sua ampla utilização, a transformação dos dados através da projeção em funções de base, seguida da codificação dos coeficientes transformados.

As transformadas de bloco, como a DCT (*Discrete Cosine Transform*) e a LT (*Lapped Transform*), que realizam o processamento atuando em pequenas regiões da imagem, têm encontrado aplicação freqüente na compressão de imagem, haja visto que apresentam re-

duzida complexidade computacional e eficiência na compressão [18, 50, 119]. A transformada discreta de cosseno (DCT) figura no padrão JPEG [80] e nos padrões H.261, H.263 e MPEG [41, 50, 101].

O encadeamento de técnicas de transformação-quantização-codificação forma o paradigma TQC, amplamente utilizado na compressão com perda de variados tipos de sinais, como áudio [78, 111], imagem [1, 11, 12, 80, 119], vídeo [39, 50], ECG [4], etc.

Em um sistema de compressão de imagem ou vídeo baseado em transformada de bloco, os coeficientes obtidos pela transformação da imagem de entrada são quantizados, e os valores obtidos passam por uma codificação sem perda, também chamada codificação entrópica. Para que os melhores resultados sejam obtidos, deve-se buscar a otimização de todos os passos do sistema. A transformada deve ser adequada ao sinal a ser codificado, concentrando as informações essenciais para o sistema visual humano em poucos coeficientes significativos, de forma que, ao serem desprezados ou quantizados de forma mais grosseira os demais coeficientes, a representação obtida ainda seja satisfatória. A quantização deve ser realizada de forma a representar da melhor maneira possível (com o menor erro de representação) os coeficientes transformados. O papel da codificação entrópica é diminuir a redundância dos símbolos emitidos pelo quantizador.

Este apêndice apresenta uma técnica de codificação sem perda de coeficientes quantizados de vídeo, oriundos da aplicação de transformadas de bloco. O algoritmo proposto, denominado SPQ (*SPIHT pós quantização*), baseia-se no reagrupamento em sub-bandas sugerido inicialmente por Xiong *et al.* [126], do qual difere pelo fato de que não utiliza a quantização intrínseca do algoritmo SPIHT [96], aplicando a codificação aos elementos já quantizados. A separação entre quantização e codificação no SPQ permite a utilização de quantizadores otimizados, podendo levar a ganhos na codificação.

São apresentados resultados de simulações envolvendo codificação de vídeo em um arcabouço de um codificador MPEG-4, modificado para acomodar o algoritmo SPIHT. A codificação de vídeo natural no padrão MPEG-4 utiliza a seqüência de zeros, RL (*Run Length*), seguida por um código de comprimento variável, VLC (*Variable Length Coding*), com tabelas pré-estabelecidas, ou seja, sem adaptação. Denotaremos essa técnica como

RL-VLC. O trabalho contempla uma avaliação comparativa de desempenho dos algoritmos SPQ e RL-VLC no que diz respeito à taxa de bits obtida.

A.2 Codificação por *Zerotree*

A codificação progressiva (*embedded*) por árvore de zeros (*zerotree*) é freqüentemente associada à transformada *wavelet* diádica. Através das características multiresolucionais da transformada *wavelet* é possível desenvolver métodos simples, porém muito eficientes, de codificação de coeficientes transformados. A transformada *wavelet* e sua aplicação em imagens bidimensionais pode ser visto no corpo principal da tese no Capítulo 2, Seção 2.4.3.

Os codificadores EZW original (*embedded* por *zerotree* com *wavelet*) [99] e suas variações efetivamente ordenam os coeficientes por planos de bits e transmitem primeiramente os bits mais significativos ao explorar a relação entre o nó pai e seus descendentes em uma árvore *wavelet*. A importância do trabalho está na aplicação conjunta de dois conceitos muito importantes: codificação progressiva e árvore de zeros, resultando uma codificação eficiente com relação à taxa-distorção.

A base do EZW é a estrutura *zerotree*, a qual utiliza o princípio de que numa decomposição *wavelet* os elementos correspondentes à mesma posição da imagem, nos níveis de decomposição, têm o valor absoluto das suas amplitudes relacionadas. Os coeficientes *wavelet* das sub-bandas de detalhes possuem, em geral, menor amplitude que os coeficientes das sub-bandas de aproximação. Se um elemento é maior do que um determinado limiar, dizemos que o elemento é significativo. Numa decomposição *wavelet* de uma imagem, se um elemento não é significativo, há uma alta probabilidade de que os elementos correspondentes à mesma posição espacial não sejam significativos. Estes elementos podem ser visualizados como uma estrutura de árvore. Se toda a árvore é não significante, dizemos tratar-se de uma *zerotree*. Este esquema de codificação resulta uma seqüência de bits progressiva, ou seja, de refinamento sucessivo, aliado conjuntamente com muitas outras vantagens, como o controle exato de taxa de bits e a possibilidade de reconstrução perfeita (a reconstrução perfeita só pode ser obtida quando a transformada mapeia inteiros

em inteiros).

A abordagem de codificação progressiva baseia-se na idéia de que a informação mais importante (definida como aquela que mais diminui uma certa medida de distorção) deve ser transmitida primeiro. Assumindo que a medida de distorção é o erro médio quadrático (MSE, *mean square error*), que a transformada é ortogonal e que os coeficientes $C_{i,j}$ são transmitidos um a um, é conhecido [16, 99] que o MSE decai de $\frac{1}{N}|C_{i,j}|$, em que N é o número total de pixels. Se um bit é transmitido por vez, esta abordagem pode ser generalizada ao se ordenar os coeficientes por planos de bits e os bits mais significantes serem transmitidos primeiro. O esquema de transmissão progressiva resulta um fluxo de bits que pode ser truncado a qualquer ponto e o decodificador produz uma imagem reconstruída correspondente com qualidade próxima da que seria obtida com a otimização para aquele ponto. O algoritmo EZW pode ser considerado como uma combinação de um quantizador escalar com zona morta e um codificador entrópico para codificar coeficientes *wavelet*.

A.3 SPIHT

O codificador SPIHT (*Set Partitioning in Hierarchical Trees*) [96] pode ser visto como um aperfeiçoamento do codificador EZW, atingindo um melhor desempenho de taxa-distorção.

O algoritmo SPIHT utiliza três conceitos básicos: (1) codifica/transmite as informações mais importantes inicialmente através de uma representação por plano de bit dos pixels, (2) a transmite ordenadamente dos planos de bit de refinamento e (3) a codificação é realizada ao longo de caminhos/árvores chamadas árvores de orientação espacial, que exploram de forma eficiente as propriedades da imagem transformada pela *wavelet* 2D.

A maioria da energia de uma imagem está concentrada nas componentes de baixa freqüência [96]. Conseqüentemente, a energia (variância) diminui ao passo que nos movemos do nível mais alto da pirâmide (sub-banda de freqüência mais baixa) para os níveis mais baixos da pirâmide de sub-bandas (freqüência mais alta). Além disso, foi observado que há

uma auto-similaridade entre sub-bandas, e espera-se que os coeficientes fiquem ordenados em magnitude (decrescente) se nos movermos para baixo na pirâmide seguindo a mesma orientação espacial. Por exemplo, espera-se que se forem identificadas grandes áreas de pouca atividade no nível mais alto da pirâmide (baixa freqüência, aproximação), estes sejam replicados nos níveis mais baixo nas mesmas posições espaciais. Estes aspectos levaram à introdução do conceito de árvore de orientação espacial, visto a seguir.

O SPIHT consiste de dois estágios principais, ordenação e refinamento. No estágio de ordenação, SPIHT ordena os pixels por magnitude em respeito a um limiar, que geralmente é uma potência de dois, chamado nível de significância. Porém, esta ordenação é parcial, porque não há uma ordem pré-estabelecida entre os coeficientes com o mesmo nível de significância ou maior bit significante. Esta ordenação é baseada no teste de significância dos pixels ao longo das árvores de orientação espacial com raiz no nível mais alto da pirâmide na imagem transformada *wavelet*.

As árvores de orientação espacial foram introduzidas para testar a significância de um grupo de pixels para a compressão eficiente ao explorar as propriedades de auto-similaridade e localidade da magnitude de uma imagem transformada *wavelet* 2D. Em outras palavras, o SPIHT explora a propriedade primeiro observada por Lewis and Knowles [53] de que se um pixel no nível mais alto da pirâmide é insignificante, é muito provável que seus descendentes sejam insignificantes. A Figura A.1 descreve a relação pai-descendentes nas árvores de orientação espacial. Nas árvores de orientação espacial, cada nó consiste de 2×2 pixels adjacentes, e cada pixel no nó tem 4 descendentes, exceto no mais alto nível da pirâmide, em que um pixel em um nó indicado por '*' nesta figura não tem nenhum descendente.

Na implementação prática, o SPIHT mantém três listas, a lista dos pixels insignificantes (LIP, *List of Insignificant Pixels*), a lista dos pixels significantes (LSP, *List of Significant Pixels*) e a lista dos conjuntos insignificantes (LIS, *List of Insignificant Sets*). No estágio de inicialização, o SPIHT inicializa a LIP com todos os pixels do mais alto nível da pirâmide, a LIS com todos os pixels do mais alto nível da pirâmide exceto os pixels que não têm descendentes e a LSP como uma lista vazia. A função básica do algoritmo

de ordenação é particionar recursivamente os conjuntos que estão representados na LIS para localizar individualmente os pixels significantes, pixels insignificantes e conjuntos insignificantes menores e mover suas coordenadas para as listas apropriadas, LSP, LIP e LIS, respectivamente. Após cada estágio de ordenamento, o SPIHT apresenta os bits de refinamento do nível atual de significância dos bits dos pixels que foram movidos para a LSP nos limiares mais altos. Desta forma, as magnitudes dos pixels significantes são refinadas com os bits que mais decrescem o erro. Este processo continua, sendo o limiar diminuído sucessivamente por um fator de dois até que a taxa de bits ou a qualidade de imagem desejada seja atingida.

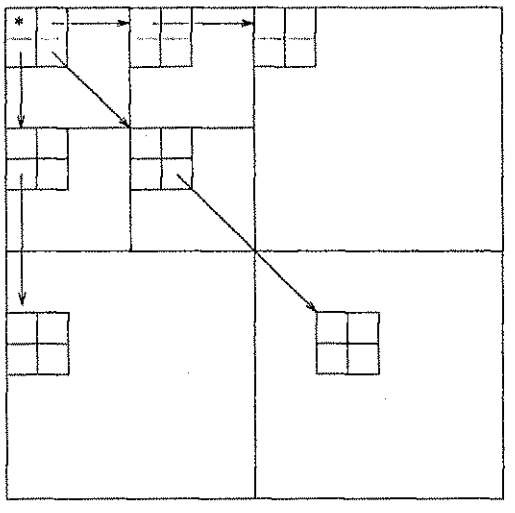


Figura A.1: Árvore de orientação espacial.

Durante a codificação, os pixels na LIP são testados e aqueles que são significantes no nível de quantização atual são movidos para a LSP. De forma similar, os conjuntos são avaliados seqüencialmente seguindo a ordem da LIS, e quando um conjunto é determinado como significante ele é removido da LIS e particionado em novos subconjuntos. Os novos subconjuntos com mais de um elemento são acrescentados de volta ao final da LIS, enquanto as coordenadas dos conjuntos que possuem um único elemento são acrescentados ao final da LIS ou da LSP, dependendo se eles são insignificantes ou significantes, respectivamente. Os conjuntos inseridos na LIS têm ainda uma outra classificação, como conjuntos do tipo A ou do tipo B. Os conjuntos do tipo A englobam toda uma árvore de

descendentes, sem incluir o nó pai, que indica apenas a posição inicial da árvore. Os conjuntos do tipo B possuem todos os descendentes excluindo-se os 4 descendentes imediatos. Esta classificação extra simplifica a implementação.

A decodificação é realizada por um algoritmo igual ao da codificação, substituindo-se a saída de bits por entrada. Isto ocorre porque o fluxo de controle do algoritmo é dominado pelas significâncias dos pixels e dos conjuntos, e são essas significâncias que são enviadas.

A.4 SPIHT Pós Quantização

Nesta seção veremos a o algoritmo de codificação de transformada de bloco baseado no reagrupamento dos coeficientes em sub-bandas e no codificador SPIHT. O algoritmo SPIHT engloba uma quantização escalar com zona morta e uma codificação entrópica dos coeficientes através de *zerotrees* e particionamento de conjuntos. A quantização intrínseca do SPIHT não é utilizada aqui. Em vez disso, utiliza-se, no presente trabalho, um quantizador escalar já disponível no MPEG-4. Esta separação permite que o quantizador seja otimizado para a estatística do sinal em questão, usando técnicas como o quantizador de Loyd-Max ou o quantizador ótimo, no sentido taxa-distorção de Ratnakar [90–92]. Como o codificador é aplicado após a quantização, denotamos este método por SPQ, ou SPIHT pós quantização.

As transformadas de bloco e as transformadas com sobreposição (*lapped*) produzem um particionamento uniforme do espectro, enquanto a transformada *wavelet* tem uma decomposição do sinal em oitavas. Todas as sub-bandas de uma transformada de bloco têm o mesmo tamanho. Um nó pai não teria quatro nós de descendentes como no caso de uma representação *wavelet*. Investigando a analogia entre a transformada *wavelet* e a transformada de bloco, como mostrado na Figura A.2, observa-se que o nó pai, os filhos (descendentes imediatos) e os demais descendentes em uma árvore *wavelet* cobrem a mesma localidade espacial, e isto também ocorre para os coeficientes de um bloco de transformada de bloco. De fato, uma árvore *wavelet* em uma decomposição em N níveis é análoga a um bloco de coeficientes de uma transformada de blocos com 2^N bandas.

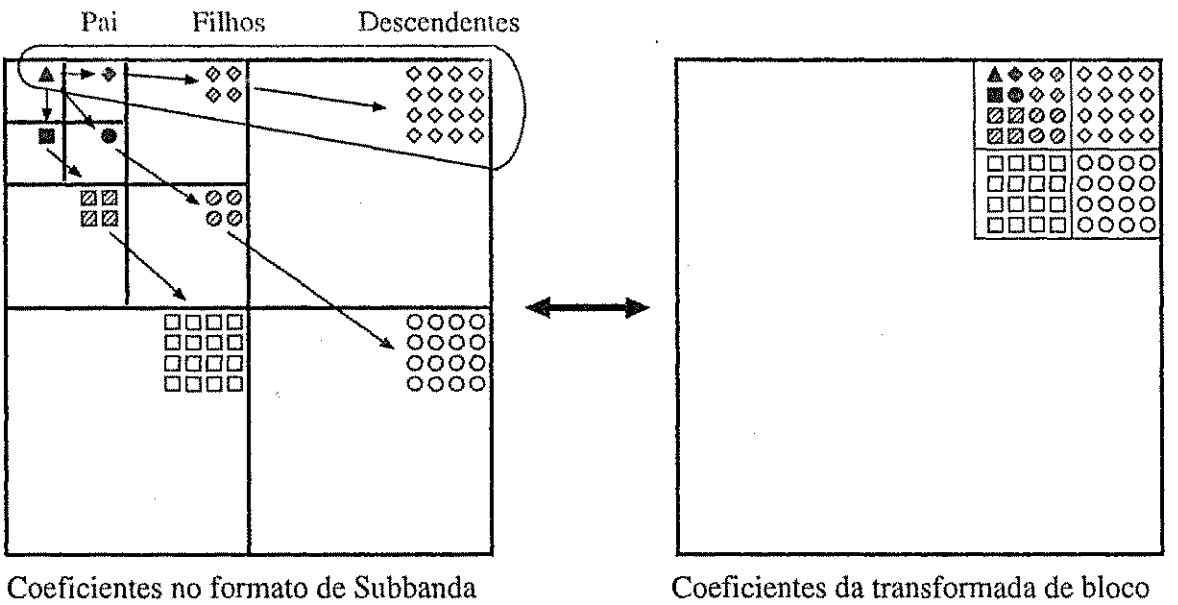


Figura A.2: Analogia entre transformadas de bloco e decomposição *wavelet*.

Essa analogia nos dá indicativos da utilização de métodos que tenham obtido sucesso na codificação baseada em *wavelet* sobre coeficientes de transformada de bloco. Encontramos esta abordagem nos trabalhos de Xiong [126] e de Queiroz e Tran [18]. Nestes trabalhos, para que os algoritmos de codificação *wavelet* baseados em *zerotree* possam ser utilizados na codificação de coeficientes de transformadas de bloco, a estrutura de dados (árvore espacial) da *zerotree* e a definição dos conjuntos de descendentes foram modificados ligeiramente. Denotaremos esta estrutura de árvore como árvore modificada e a árvore espacial definida no SPIHT como árvore original. A estrutura da árvore modificada pode ser vista na Figura A.3. A árvore original foi mostrada na seção A.3. Adiante veremos testes realizados com os dois tipos de árvore.

A técnica de SPQ foi avaliada em uma arcabouço de codificação de vídeo compatível com o padrão MPEG-4. Os quadros do tipo *intra* são codificados como imagens, sem referência a outros quadros. Utilizando o mesmo arcabouço, podemos obter resultados para imagem (*intra*) e vídeo (*inter*)[1]. O SPQ opera sobre toda a imagem, no plano de

[1]Desta forma, com poucas modificações, a técnica pode ser aplicada à codificação de imagens, como, por exemplo, no padrão JPEG.

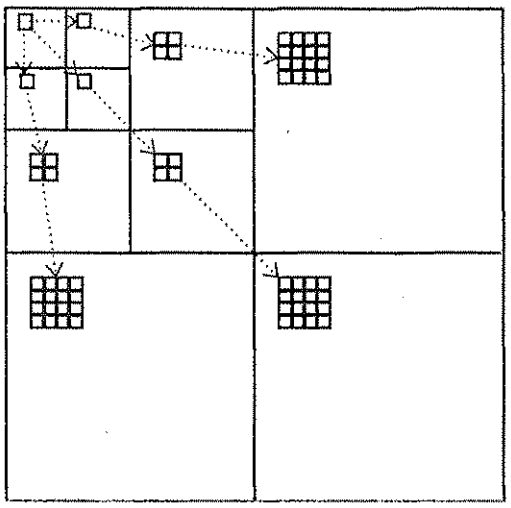


Figura A.3: Árvore espacial modificada.

luminância Y e nos planos de crominância Cb e Cr, tanto nos quadros *intra* quanto *inter*.

Adiante veremos os detalhes da substituição do codificador entrópico para os coeficientes DCT do MPEG-4 pelo SPQ.

Na implementação prática, o SPQ se distingue do SPIHT basicamente na condição de parada do algoritmo. O SPIHT normalmente opera até que um determinado número de bits seja atingido ou uma certa distorção seja atingida. No caso do SPQ, o algoritmo atravessa todos os planos de bits significativos que são encontrados.

O SPQ utiliza como etapa final um codificador aritmético otimizado para dois símbolos, com possibilidade de seleção adaptativa de contextos [6, 123]. Esta possibilidade de adaptação de contextos, se bem explorada, pode proporcionar uma diminuição no número de bits utilizados. No SPQ, a seleção dos contextos foi realizada de forma a aproveitar a correlação existente entre o número de pixels significativos em um conjunto de descendentes imediatos e a significância das árvores que são descendentes destes pixels.

Foi utilizado um conjunto de 14 contextos, sejam eles: 1 contexto para codificação de sinal; 1 contexto para codificação dos bits de refinamento; 4 contextos para codificação adaptativa das significâncias dos conjuntos tipo B; 4 contextos para codificação das significâncias dos conjuntos tipo A dependente da codificação dos conjuntos tipo B; e fi-

nalmente 4 contextos para a codificação dos pixels descendentes imediatos. As definições destes conjuntos e das listas foram vistas na Seção A.3.

O contexto para a codificação da significância dos pixels é selecionado de acordo com o número de pixels que já foram significantes no mesmo bloco 2×2. Esta escolha é armazenada na LIS junto com os conjuntos do tipo B. Durante a codificação da significância dos conjuntos do tipo A e do tipo B, este valor é recuperado e utilizado na seleção de novos contextos. Desta forma, há uma propagação ao longo do particionamento dos conjuntos do número de pixels significativos no começo da árvore. Só há um contexto para o sinal porque o sinal é considerado descorrelacionado. Pelo mesmo motivo, temos apenas um contexto para os bits de refinamento.

A codificação MPEG-4 prevê a predição de coeficientes DCT a partir de seus vizinhos causais. Para blocos do tipo *intra*, há predição dos coeficientes DC e AC. Para blocos do tipo *inter*, há apenas predição DC. Na predição DC, subtrai-se o coeficiente DC do bloco atual do coeficiente DC do bloco imediatamente anterior ou imediatamente acima (na linha de blocos anterior). A seleção de que bloco é usado na predição é baseada na direção de maior variação destes coeficientes. Os coeficientes AC da primeira linha e da primeira coluna são preditos a partir do mesmo bloco usado para o coeficiente DC. Esta predição melhora a codificação ao reduzir a amplitude dos coeficientes. No entanto, faz-se necessário avaliar melhor se esta predição produz bons resultados para a codificação com plano de bits e *zerotree*. Foram realizados testes com e sem predição.

Como proposto por Queiroz [18], a banda formada pelos coeficientes do nível DC ainda pode se tornar mais descorrelacionada, aplicando alguns níveis de decomposição *wavelet*, a depender das dimensões do quadro. Como o objetivo desta codificação é a reconstrução perfeita, haja visto que os coeficientes já foram quantizados, faz-se necessário o uso de *wavelets* que mapeiem inteiros em inteiros. Por simplicidade, optou-se por utilizar uma variação da *wavelet* de Haar, definida da seguinte forma. Seja $x = (x_0, x_1, \ldots, x_{N-1})$ o vetor de N elementos, N par, a ser transformado e $y = (y_0, y_1, \ldots, x_{N-1})$ o vetor de saída. O

resultado da filtragem passa-baixa a_i e passa alta d_i é dado por

$$\begin{aligned} a_i &= \lfloor \tfrac{x_i + x_{i+1}}{2} \rfloor \\ d_i &= x_i - x_{i+1}, \end{aligned} \tag{A.1}$$

para i par, e o vetor de saída é organizado como

$$\begin{aligned} y_{i/2} &= a_i \\ y_{\frac{i+N}{2}} &= d_i. \end{aligned} \tag{A.2}$$

A transformada inversa é calculada como

$$\begin{aligned} x_{2i} &= a_i + \lfloor \tfrac{d_i + 1}{2} \rfloor \\ x_{2i+1} &= x_i - d_i, \end{aligned} \tag{A.3}$$

sendo este procedimento realizado para as linhas e colunas da imagem a ser transformada. Este processo pode ser repetido para formar mais níveis de decomposição *wavelet*.

Os resultados das diversas combinações de codificação podem ser vistos em seguida.

A.5 Resultados

Nesta seção apresentamos os resultados práticos da codificação de vídeo utilizando o codificador SPQ. Foram realizados testes com os dois tipos de árvores (modificada e original), com e sem predição MPEG-4 e com a aplicação opcional de transformada *wavelet* na banda DC. Convém informar que os resultados são apresentados como bytes utilizados para a codificação, sem nenhuma referência à qualidade obtida, porque essa não é modificada em comparação com o codificador original do MPEG-4. Deve ser observado também que é contabilizada, na quantidade de bytes mostrada para o codificador RL-VLC do MPEG-4, a representação dos blocos não codificados, representados de forma eficiente com apenas um bit. No caso do SPQ, a área correspondente ao bloco 8×8 classificada como "não codificada" é zerada e codificada conjuntamente com os demais blocos, sem nenhuma referência explícita.

Na Tabela A.1, observamos um conjunto com os resultados de testes para a seqüência "Miss America", para o coeficiente de quantização $Q = 2$, que corresponde à alta qualidade (alta taxa de codificação). Na segunda coluna da tabela, temos o número de bytes

utilizados pela codificação MPEG-4 normal. Estes valores representam apenas os bytes utilizados na codificação da textura, excluindo-se os bits de controle e de representação dos vetores de movimento. O quadro de número 1 foi codificado como *intra*. Os demais foram codificados como *inter*. Observa-se que a maior redução no número de bytes neste codificador ocorre para os quadros intra, com a aplicação da decomposição normal (de SPIHT), o uso de predição e de nova decomposição *wavelet* dos coeficientes DC. O ganho não é tão significativo para os quadros *inter*. Um dos motivos para isto é o grande número de macroblocos em que a textura não é codificada ou utiliza-se apenas 1 ou 2 coeficientes. Isto ocorre quando a compensação atinge uma boa predição. Nestes quadros gasta-se menos bytes quando não se utiliza a transformada *wavelet*. Também neste caso a decomposição modificada não fornece os melhores resultados.

Na Tabela A.2 vemos a codificação da mesma seqüência para um coeficiente de quantização $Q = 15$. Isto corresponde a uma qualidade baixa e a uma pequena taxa de codificação. Observamos que a decomposição modificada forneceu os melhores resultados para o quadro *intra* sem uso de predição. Para o melhor caso utilizou-se também a predição e a decomposição *wavelet*. Para os quadros *inter*, observa-se que as duas decomposições produzem valores similares. Observa-se também resultados ligeiramente melhores sem o uso de *wavelet*.

Na Figura A.4 vemos o gráfico da variação do número de bytes usados na codificação do primeiro quadro da seqüência "Miss America", com codificação *intra*, com o coeficiente Q. O maior ganho, cerca de 20%, ocorre para $Q = 2$. Este ganho vai diminuindo com o aumento do Q (diminuição da qualidade). Foi usada neste teste a combinação que deu o melhor resultado para baixos valores de Q, ou seja, decomposição original, predição MPEG-4 e decomposição *wavelet* dos coeficientes DC.

Nas Figuras A.5 e A.6 analisamos a codificação da seqüência "Flower Garden" com a variação do Q, para o quadro *intra* e os quadros *inter*. Os codificadores de vídeo, em geral, não conseguem produzir uma baixa taxa de bits para essa seqüência, em virtude de ela possuir um movimento global e texturas complexos. A codificação desta seqüência foi beneficiada pelo uso do sistema proposto. Obtém-se um ganho de cerca de 37% para o

		Decomposição Normal			Decomposição Modificada	
Quadro	MPEG-4		sem *wavelet*	com *wavelet*	sem *wavelet*	com *wavelet*
1	17580	sem pred	14789	14689	15143	15033
		com pred	14108	14052	14826	14741
2	9512		8811	8970	9025	9199
3	9491		8772	8866	9023	9133
4	9963		9133	9235	9311	9427
5	10205		9376	9446	9550	9646
6	10016		9107	9233	9492	9628
7	9720		8918	8982	9298	9385
8	9763		8932	9095	9290	9449
9	9415		8583	8666	8907	9006
10	9762		8855	8908	9236	9337

Tabela A.1: Codificação da seqüência "Miss America" com $Q = 2$. Os números são em bytes.

quadro *intra* com $Q = 2$. Este ganho vai diminuindo com o aumento do Q. O ganho para o quadros *inter* é menor, porém é superior ao obtido na seqüência "Miss America". Neste caso, o maior ganho (redução de bits) ocorre quando se utiliza a decomposição modificada, sem predição MPEG-4 nem decomposição *wavelet* dos coeficientes DC.

Para seqüências com muita textura, onde particularmente a codificação padrão do MPEG-4 não realiza um bom trabalho, obtém-se uma grande redução no número de bytes utilizados. Os melhores resultados sempre ocorrem para altas taxas.

Para baixas taxas de codificação, os coeficientes DCT possuem valores pequenos e são

		Decomposição Normal			Decomposição Modificada	
Quadro	MPEG-4		sem *wavelet*	com *wavelet*	sem *wavelet*	com *wavelet*
1	2184	sem pred	2363	2288	2065	2051
		com pred	1818	1887	1830	1888
2	129		110	116	110	119
3	160		139	159	144	163
4	196		196	219	182	206
5	221		212	227	213	228
6	210		215	239	221	245
7	134		115	134	122	143
8	141		124	141	131	152
9	185		194	222	199	225
10	96		71	82	75	84

Tabela A.2: Codificação da seqüência "Miss America" com $Q = 15$. Os números são em bytes.

dispostos de forma esparsa. Este efeito se agrava nos quadros *inter*, em que a amplitude dos coeficientes é menor e a distribuição de freqüência é diferente, não havendo um agrupamento na vizinhança do nível DC. A codificação destes coeficientes torna-se basicamente uma codificação de posição dos coeficientes não zero. O melhor aproveitamento desta esparsidade pode levar a técnicas mais eficientes de codificação dos coeficientes DCT em vídeo.

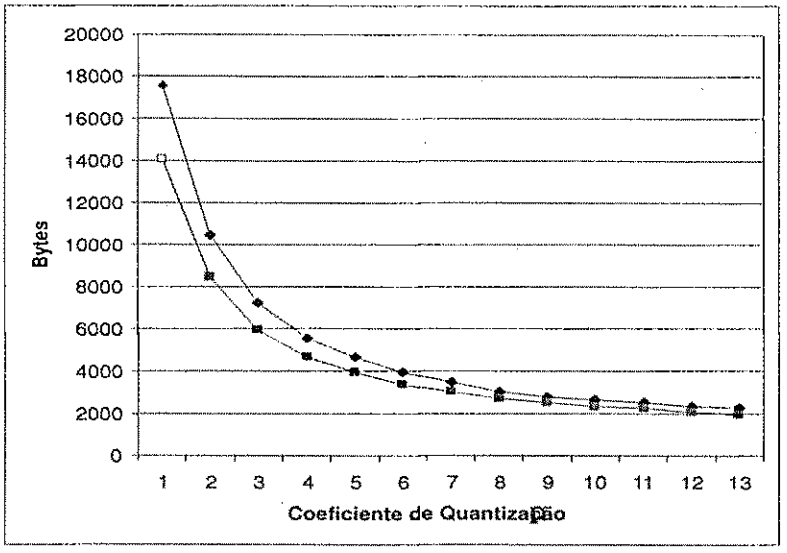


Figura A.4: Resultado da codificação intra para a seqüência "Miss America", variando-se o fator de qualidade Q. O codificador foi configurado com a decomposição original, predição dos coeficientes e decomposição *wavelet* dos coeficientes DC.

A.6 Avaliação do codificador

Neste apêndice vimos alguns resultados selecionados da aplicação da técnica de particionamento de conjuntos em árvores hierárquicas aplicado na codificação para transformadas de blocos rearranjados. O SPQ baseia-se no reagrupamento dos coeficientes da transformada de bloco segundo as suas sub-bandas, e na aplicação do algoritmo SPIHT aos coeficientes já quantizados. O reagrupamento baseia-se numa analogia entre transformada *wavelet* discreta multiresolucional e as sub-bandas das transformadas de bloco.

O algoritmo SPQ foi implementado como uma modificação de um codificador de vídeo MPEG-4. Isto permitiu uma análise da técnica para compressão de imagem parada (quadros *intra*) e vídeo (quadros *inter*). Vimos que o SPQ apresentou-se como uma boa alternativa para a codificação de quadros *intra*. Estes resultados eram de certa forma esperados, haja visto que a técnica tinha sido aplicada anteriormente, com excelentes resultados, na codificação de imagem [126]. Os resultados cientificamente mais relevantes são decorrentes da aplicação do algoritmo aos quadros *inter*. Neste caso codifica-se uma

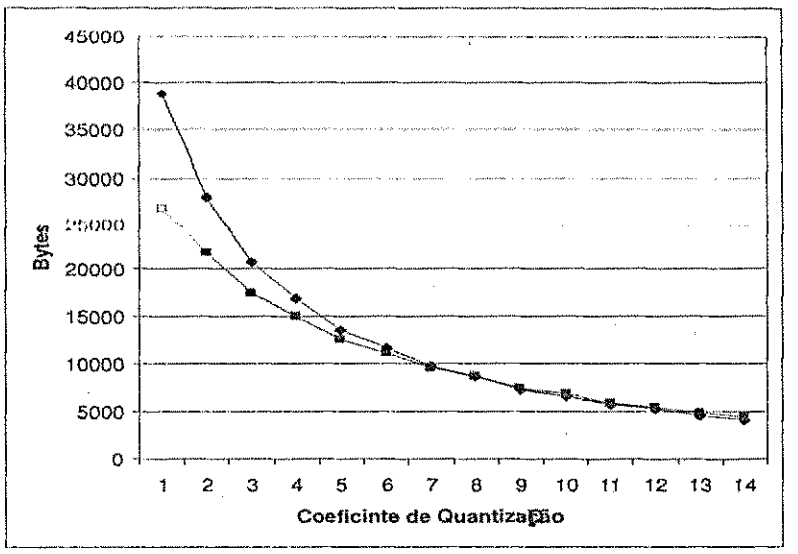


Figura A.5: Resultado da codificação inter para a seqüência "Flower Garden", variando-se o fator de qualidade Q. Configurou-se o codificador para utilizar a decomposição modificada, sem predição dos coeficientes e sem decomposição *wavelet* dos coeficientes DC.

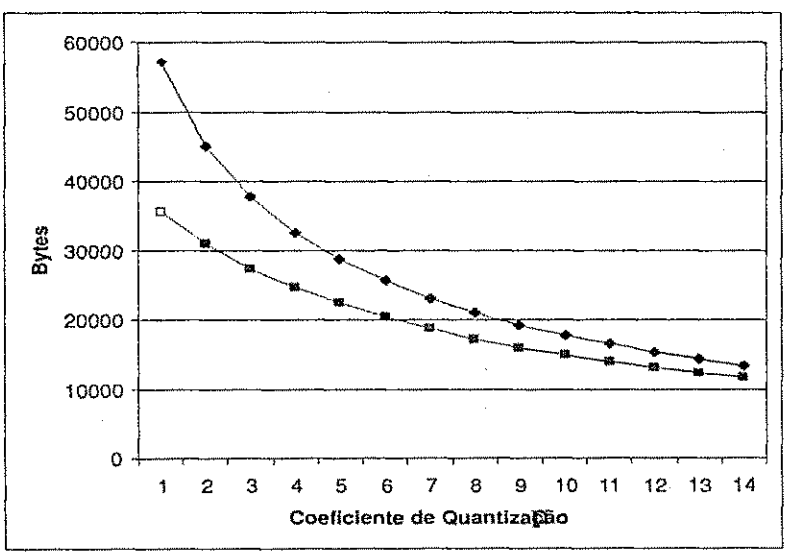


Figura A.6: Resultado da codificação intra para a seqüência "Flower Garden", de acordo com o fator de quantização Q. O codificador foi configurado para utilizar a decomposição modificada, sem a predição MPEG-4 de coeficientes e sem decomposição *wavelet* adicional da sub-banda DC.

imagem resíduo que é formada pela diferença entre o quadro atual e uma imagem formada pela compensação de movimento, que utiliza informações de quadros anteriores. A imagem resíduo possui características estatísticas muito diferentes das imagens naturais, além de possuir uma faixa dinâmica muito menor. Nestes quadros a vantagem da utilização do SPQ foi menor, principalmente para baixas taxas, ou seja, com baixa qualidade, o que implica em um fator de quantização Q elevado. Isto se deve ao fato de que na codificação original MPEG-4, um bloco cuja compensação de movimento tenha produzido um resultado já satisfatório, não seja codificado e esta informação seja indicada com o uso de apenas um bit. Estes blocos são codificados conjuntamente com os demais no SPQ.

Observou-se também que para seqüências consideradas simples, ou seja, com pouca textura e/ou pouco movimento, o SPQ apresenta vantagem pequena com relação ao MPEG-4. Estas seqüências produzem blocos transformados formados por coeficientes que são, na sua maioria, de baixa amplitude. Estes são quantizados para zero, formando blocos muito esparsos. Quando há pouco movimento, a compensação de movimento é mais eficiente e produz uma imagem resíduo também de baixa amplitude nos seus coeficientes. Esta alta esparsidade não é completamente aproveitada pelo SPQ.

A grande desvantagem do SPQ é que para haver a codificação, toda a imagem tem de ser armazenada, para que a varredura seja realizada sobre os planos de bits individuais.

A.7 Conclusão

Neste apêndice foi apresentado o SPQ (SPIHT pós quantização), um método para codificação sem perdas de coeficientes quantizados, oriundos de transformadas de bloco. Os resultados de simulação mostraram que o SPQ constitui uma alternativa adequada para a codificação de vídeo, e por extensão, de imagens. Para a seqüencia "Miss América", por exemplo, para codificação intra com parâmetro de quantização Q=2, o SPQ levou a uma economia de cerca de 20% do número de bytes em relação ao MPEG-4 convencional. Para a seqüencia "Flower Garden", obteve-se uma economia correspondente de cerca de 37% para codificação *intra*. Vale a pena salientar que os quadros codificados com *intra*

representam uma parcela muito grande dos bytes utilizados para codificação de vídeo, uma vez que eles não utilizam outros quadros como referência. Desta feita, o SPQ revela-se como um excelente codificador de imagens também. O uso da codificação com SPIHT após a quantização normal do MPEG-4 permite a utilização de técnicas otimizadas de quantização, como por exemplo, a quantização por codificação em treliça (*trellis code quantization, TQC*) [23, 24] e a codificação de coeficientes DCT otimizados globalmente e localmente introduzida por Ratnakar [90–92].

Apêndice B

Codificação de vídeo por Planos de Bits

B.1 Introdução

B.1.1 Objetivos

Verificar a codificação de coeficientes da DCT bidimensional por planos de bits sem reagrupamento, para codificação de imagens (*intra*) e de vídeo (*inter*).

B.1.2 Motivação

Uma das técnicas mais utilizadas para a codificação de imagem e de vídeo é a aplicação de transformadas de bloco, como a DCT, utilizada nos padrões de compressão de imagem e vídeo [41, 50, 80, 119]. Neste contexto particiona-se a imagem em pequenos blocos, normalmente quadrados ou retangulares, e aplica-se a DCT bidimensional a cada um destes blocos individualmente. Em seguida os coeficientes são quantizados e codificados de forma eficiente através de uma codificador entrópico.

O efeito de aglutinação da energia do sinal em poucos coeficientes realizada pela DCT, aliada a quantização, produz blocos com muitos coeficientes de valor zero. Esta característica é explorada realizando codificando conjuntamente os valores nulos de cada

bloco através da técnica de corrida de zeros (*runlength encoding*). Obtém-se com isso bons resultados com uma baixa complexidade.

No Apêndice A, a codificação foi realizada reagrupando-se os coeficientes DCT de forma similar a decomposição *wavelet*. Empregou-se então um codificador baseado em árvores de zeros *zerotrees* e planos de bits, obtendo-se resultados promissores. É necessário então avaliar se esta eficiência é decorrente do reagrupamento ou do uso de planos de bits.

Neste trabalho, propomos um codificador por planos de bits para os coeficientes DCT aplicado aos blocos individuais. Neste caso os coeficientes já estão quantizados e todos os planos de bits devem ser codificados (codificação sem perda). A esparsidade dos coeficientes não é explorada de forma global, através de regiões de zeros, mas sim de forma indireta através do uso de contextos para a codificação aritmética.

Esta forma de codificação por blocos individuais utiliza menos espaço de memória e normalmente o processamento é mais rápido devido ao efeito de localidade, com melhor aproveitamento da memória *cache* do processador. Além disso, é possível aplicar esta técnica a outros tipos de transformada de blocos, assim como a LOT, GenLOT, GLBT, etc [18, 102]. Estas novas transformadas de bloco possuem alto ganho de codificação [45] e quando associadas a uma codificação entrópica adequada levam a excelentes resultados, estando entre os melhores para codificação de imagens.

Este trabalho está dividido da seguinte maneira. A codificação com plano de bits é resumida na seção B.2. As modificações realizadas no codificador MPEG-4 são tratadas na seção B.3. Os resultados obtidos estão na seção B.4 e as conclusões podem ser encontrados na seção B.6.

B.2 Codificação por Planos de Bits

Um plano de bits é uma imagem binária, onde cada posição representa o n-ésimo bit de cada elemento x, $x \in X$, sendo $\mathbf{X}$ o conjunto dos coeficientes. Todos os planos de bits $\mathbf{P}_n$, para $0 < n < N$, $N = \lfloor log_2 M \rfloor$, onde $M = max(abs(x)), \forall x \in \mathbf{X}$, formam uma representação alternativa de $\mathbf{X}$.

Cada um dos planos de bits pode ser separado em três outros planos de bits, denominados de novos significantes $\mathbf{P}_n^N$, já significantes $\mathbf{P}_n^O$ e refinamento $\mathbf{R}_n$. Dizemos que um elemento é significante se o seu valor for maior ou igual a um determinado limiar T. A cada plano de bit $\mathbf{P}_n$, está associado um limiar $T = 2^n$. O plano de novos significantes $\mathbf{P}_n^N$ é formado pelos elementos maiores ou iguais a T e menores do que $2T$ (o limiar do plano $n+1$). O plano de já significantes $\mathbf{P}_n^O$ é formado pelos elementos maiores ou iguais a $2T$. O plano de refinamento $\mathbf{R}_n$ é formado pelos n-ésimos bits dos elementos. Ao conjunto dos novos significantes e já significantes chamamos de mapa de significância. Utilizando as operações booleanas '+' e '.' representando "ou" e "e" lógicos, podemos dizer que:

$$\mathbf{P}_n = (\mathbf{P}_n^O.\mathbf{R}_n) + \mathbf{P}_n^N \tag{B.1}$$

Esta separação dos planos de bits permite explorar a correlação existente entre os elementos novos significantes quando estes são o resultado de uma transformada com DCT ou *wavelet*. A codificação é realizada a partir do maior limiar, reduzindo-o sucessivamente. Observa-se que não é necessário codificar os elementos já significantes, haja visto que eles podem ser encontrados acumulando-se os novos significantes dos limiares anteriores. Os bits de refinamento possuem baixa auto-correlação e são codificados isoladamente. Uma representação esquemática da codificação dos planos de bits pode ser visto na Figura B.1.

A ordenação dos coeficientes por planos de bits e a transmissão primeiro dos bits mais significativos resulta em uma seqüência progressiva de bits (*embedded*). Esta seqüência de bits pode ser truncada e o resultado pode ser decodificado, levando a uma aproximação sucessiva dos valores enviados. Isto permite a quantização e a codificação em um único passo. No nosso caso, consideraremos que os elementos a serem codificados já estão quantizados e devem ser enviados sem perda. Assim, todos os planos de bits devem ser codificados.

É possível encontrar termos em comum aos algoritmos de codificação de planos de bits baseados na separação em mapa de significância e refinamento. A idéia central é explorar a correlação existente entre os elementos novos significantes, aproveitando o conhecimento dos já significantes. A codificação de um plano de bit esparso pode ser visto como a

	sinal	s	s	s	s	s	s	s	s	s	s	s	s	s
msb	5	1	1	0	0	0	0	0	0	0	0	0	0	0
	4	➢	➢	1	1	0	0	0	0	0	0	0	0	0
	3	➢	➢	➢	➢	1	1	1	1	0	0	0	0	0
	2	➢	➢	➢	➢	➢	➢	➢	➢	1	1	1	1	1
	1	➢	➢	➢	➢	➢	➢	➢	➢	➢	➢	➢	➢	➢
lsb	0	➢	➢	➢	➢	➢	➢	➢	➢	➢	➢	➢	➢	➢

Figura B.1: Codificação dos mapas de significância.

localização dos elementos minoritários (normalmente um). A diferença mais marcante entre os diversos algoritmos de codificação de planos de bits estudados está na forma como esta localização é feita. Os elementos majoritários (normalmente zero), normalmente possuem alguma estrutura que pode ser explorada. Na codificação de coeficientes de uma transformada *wavelet* ou DCT, os coeficientes positivos e negativos aparecem com a mesma probabilidade e de forma muito aleatória [2,3]. Assim o sinal é normalmente codificado de forma direta e indicado assim que a significância de um elemento é conhecida. Os bits de resíduos também não possuem uma estatística favorável e são codificados de forma direta, não importando muito a sua ordenação.

O algoritmo de codificação de coeficientes wavelet por árvores de zero (*zerotree*) denominado EZW [99], ao codificar um mapa de significância, aproveita o agrupamento dos zeros em grandes áreas e a tendência verificada de que se um valor é não significante, a árvore multiresolucional por ele encabeçada normalmente é toda insignificante. Esta árvore de elementos zeros é denominada *zerotree*. O EZW atribui um único símbolo a uma (T). Uma *zerotree* pode indicar um grande conjunto de pixels. Para codificar os elementos zero que não formam *zerotrees*, existe o símbolo de zero isolado (Z). Dois outros símbolos (P) e (N) indicam elementos significantes positivos e negativos.

O algoritmo denominado SPIHT [96] é um melhoramento do EZW. Ele consegue uma codificação mais eficiente ao ter dois tipos estruturas espaciais em árvore, com ou sem considerar os descendentes imediatos. Se uma destas estruturas espaciais possui apenas elementos nulos, isto é sinalizado e nenhum dos seus pixels é visitado novamente. Se

houver elementos significantes, a estrutura é divida. Se for uma árvore completa é divida em uma árvore sem descendentes imediatos e pixels isolados e a significância destes pixels é codificada juntamente com o sinal. Se é uma árvore incompleta, é dividida em várias árvores completas. Este processo se repete até que o plano de bit seja codificado como conjuntos e pixels isolados.

A codificação proposta é baseada nos princípios de plano de bits e refinamento, porém aplicados a blocos transformados já devidamente quantizados.

B.3 Implementação

O codificador entrópico dos blocos de coeficientes DCT de um codificador MPEG-4 foi substituído pelo algoritmo de planos de bits sugeridos. Neste capítulo veremos os detalhes desta implementação e os resultados obtidos serão vistos em seguida. Para a codificação entrópica propriamente dita foi utilizado um codificador aritmético otimizado para dois símbolos com possibilidade de seleção adaptativa de contextos [6, 123].

B.3.1 Algoritmo proposto

A entrada do algoritmo é bloco 8×8 de coeficientes DCT já quantizados. Os coeficientes, segundo o padrão MPEG-4, estão na faixa de -2047 a 2048, ou seja, 12 bits de magnitude e um bit de sinal. O maior valor absoluto é encontrado e o número de bits capaz de representar este valor é codificado. Os planos de bits são então varridos do mais significativo até o menos significativo. Para cada plano de bit 2 matrizes são criadas, correspondentes ao mapa de significância (acumulado) e aos novos significantes. As posições que ainda não são significantes são verificadas e se forem significantes, a matriz de novos significantes é atualizada e a significância e o sinal são codificados. Os bits das posições marcadas no mapa de significância são então enviados e o mapa de refinamento é atualizado.

Para a codificação aritmética foi utilizada a seguinte seleção de contextos:

- 4 contextos para a codificação do número de planos de bits enviados
- 64 contextos para a codificação dos coeficientes intra
- 64 contextos para a codificação dos coeficientes inter
- 1 contexto para o sinal
- 1 contexto para os bits de refinamento

A adaptação de contextos proporciona uma forma simples, porém muito eficiente de codificação entrópica, que quando bem explorada, proporciona uma diminuição no número de bits utilizados. A seleção dos contextos de significância foi planejada de forma a aproveitar a correlação existente entre a significância dos elementos de um bloco, e a correlação entre blocos é explorada de forma implícita através das probabilidades decorrentes da codificação dos blocos anteriores.

Para a codificação no número de planos de bits foram utilizados 4 contextos, sendo um para cada posição de bit.

O contexto para a codificação da significância dos pixels é selecionado de acordo com o a significância de alguns vizinhos espaciais, com pesos variados. Este conceito pode ser visualizado na Figura B.3. Se um pixel em determinada posição relativa ao pixel que está sendo codificado agora (X) é significante, o seu peso é acrescido à seleção de contexto. São possíveis 64 contextos para a codificar a significância, para macroblocos *intra* e outros 64 para macroblocos *inter*. Um contexto é usado para o sinal e outro para os bits de refinamento porque estes símbolos geralmente não seguem nenhuma estrutura.

B.4 Resultados

Apresentamos nesta seção os resultados da codificação de vídeo utilizando o codificador proposto. Como apenas o codificador entrópico para os coeficientes DCT foi substituído, o resultado final, em termos de qualidade visual dos quadros individuais é exatamente o

mesmo. Os resultados são apresentados na forma de bytes utilizados. Os valores apresentados representam apenas os bytes gastos na codificação dos coeficientes, não levando em conta os bits gastos nos cabeçalhos e na codificação das outras informações, tais como os vetores de movimento, os bits de controle e indicadores de modo, etc.

Na tabela B.2, observamos um conjunto os resultados dos testes para a seqüência "Miss America", para os coeficiente de quantização $Q = 2$ até $Q = 14$. Na primeira e terceira colunas temos os números de bytes utilizados pela codificação MPEG-4 normal, para os quadros *intra* e *inter* respectivamente. Na segunda e quarta colunas observamos o resultado da codificação proposta. Podemos constatar que para esta seqüência não obteve-se resultado satisfatório. Nas tabelas B.3 e B.1 temos os resultados para as seqüências "Foreman" e "Flower Garden". Observa-se uma redução significativa no número de bytes utilizados para a codificação dos quadros *intra* e *inter* da seqüencia "Flower", para todos os níveis de qualidade. Para a seqüencia "Foreman" só é observado um ganho na mais alta qualidade testada.

B.5 Análise comparativa do codificador

O codificador SPQ analisado no Apêndice A requer o armazenamento de todos os blocos de uma imagem ou quadro de vídeo para iniciar o processo de codificação. Para sobrepor esta dificuldade, a técnica de codificação em planos de bits foi aplicada aos blocos individuais da DCT, também no arcabouço de um codificador MPEG-4. Denominamos esta técnica de CBPB. Este experimento também teve o objetivo de verificar se a codificação em planos de bits, que tem dado bons resultados tanto na codificação baseada em *wavelets* quanto em transformada de blocos, também é eficiente quando aplicada aos blocos individuais.

Como a codificação no CBPB é realizada em blocos individuais, a imagem não precisa ser armazenada completamente. Os blocos determinados a não serem codificados também podem ser indicados utilizando apenas um bit. Os testes realizados no capítulo anterior contabilizam apenas os blocos realmente codificados. Um ganho maior de codificação

Tabela B.1: Codificação da seqüência "Flower Garden", para diversos fatores de qualidade (Q), com o codificador MPEG-4 e o codificador proposto. Os números para os quadros *inter* são a média dos 10 primeiros quadros, em bytes.

Q	Intra Proposto	Intra MPEG-4	Inter Proposto	Inter MPEG-4
2	49209	55411	30334	38344
3	40523	43275	23242	27428
4	34740	35991	17923	20311
5	30520	30906	15091	16576
6	27173	27128	12283	13211
7	24416	24108	10781	11418
8	22187	21637	9000	9441
9	20258	19575	8068	8382
10	18646	17846	6832	7054
11	17223	16371	6196	6377
12	15985	15104	5300	5428
13	14892	13981	4868	4969
14	13892	12947	4224	4306

Tabela B.2: Codificação da seqüência "Miss América", para diversos fatores de qualidade (Q), com o codificador MPEG-4 e o codificador proposto. Os números para os quadros *inter* são a média dos 10 primeiros quadros, em bytes.

Q	Intra Proposto	Intra MPEG-4	Inter Proposto	Inter MPEG-4
2	16627	16027	9333	9174
3	9857	8846	4776	4481
4	6845	5656	1818	1594
5	5237	4029	1214	1021
6	4289	3156	677	549
7	3598	2503	535	425
8	3078	2012	348	267
9	2657	1608	299	225
10	2384	1381	228	166
11	2156	1219	189	137
12	1974	1108	152	108
13	1752	962	138	100
14	1614	878	111	77

Tabela B.3: Codificação da seqüência Foreman, para diversos fatores de qualidade (Q), com o codificador MPEG-4 e o codificador proposto. Os números para os quadros *inter* são a média dos 10 primeiros quadros, em bytes.

Q	Intra Proposto	Intra MPEG-4	Inter Proposto	Inter MPEG-4
2	26216	25807	12400	12749
3	18980	17549	8135	8030
4	14801	13036	4611	4367
5	12095	10304	3594	3336
6	10225	8517	2455	2207
7	8878	7209	2072	1832
8	7797	6213	1522	1304
9	6915	5413	1372	1166
10	6264	4801	1065	878
11	5723	4291	960	784
12	5258	3873	763	607
13	4893	3527	701	550
14	4609	3227	591	457

poderia ser obtido se a informação sobre a codificação de cada bloco e os demais elementos da sintaxe do bitstream do MPEG-4 também fossem codificados por codificador aritmético baseado em contexto, como demonstra o CABAC, utilizado no novo padrão H.264, que fará parte do MPEG-4 e será denominado MPEG-4 versão 10 [70].

Os resultados da aplicação do CBPB foram bons, porém foram inferiores aos do SPQ. Da mesma forma, o ganho na codificação é mais expressivo nos quadros *intra* e nas seqüências com mais textura e movimento. Isto demonstra que o MPEG-4 não é muito eficiente nesta seqüências específicas. O ganho menor com relação ao SPQ está associado com o menor aproveitamento da correlação entre elementos de mesma sub-banda.

Para baixas taxas e seqüências de pouca textura, os coeficientes DCT quantizados são muito esparsos e de baixa amplitude. A codificação repetida do mapa de significância para cada plano de bit acaba por penalizar o CBPB e, nestes casos, a codificação é pior do que para o codificador original do MPEG-4.

B.6 Conclusão

Neste apêndice foi apresentado uma codificação dos coeficientes DCT que utiliza planos de bits bloco a bloco. Constatou-se que para seqüências de alta freqüência espacial em movimento o método proposto é vantajoso. Algumas deficiências foram encontradas na codificação de seqüências facilmente previsíveis. Nestas seqüências os coeficientes são dispostos de forma altamente esparsa. Espera-se, em trabalhos futuros, melhorar a codificação deste tipo de seqüência usando outras técnicas, como o casamento parcial de padrões.

Entrada: Lista de blocos B de coeficientes transformados

```
1  início
2      para i = 0 até dim(B) faça
3          M e S armazenas as significâncias novas e acumuladas, respectivamente;
4          M ← ∅, S ← ∅;
5          T é o limiar atual. N é o número de bits capaz de representar o maior limiar;
6          N = ⌊log2 (máx(abs(B[i])))⌋;
7          codifique(binarize(N));
8          x ← zig-zag(B[i]);
9          para T = 2^N decrescendo até 1 faça
10             para ∀j|j ∈ x ∧ j ∉ M faça
11                 codifique(x[j] ≥ T);
12                 se x[j] ≥ T então
13                     M ← M ∪ {j};
14                     codifique(x[j] > 0);
15             para ∀j|j ∈ S faça
16                 codifique(T/2 ≤ x[j] < T) ;
17             S ← S ∪ M;
18             T ← T/2;
19 fim
```

Figura B.2: Algoritmo de codificação por planos de bits.

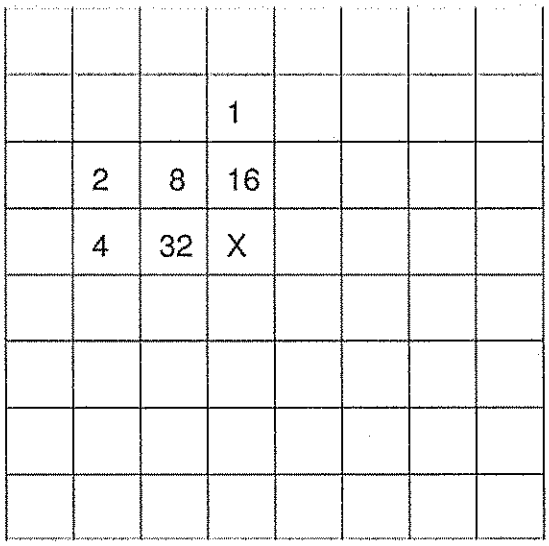


Figura B.3: Seleção de contexto para codificação da significância de cada plano de bit.

Referências Bibliográficas

[1] Michael D. Adams, Hong Man, Faouzi Kossentini, e Touradj Ebrahimi. JPEG2000: The Next Generation Still Image Compression Standard. Relatório técnico, Image Power Inc., Vancouver, BC, Canada, 2000.

[2] M. Antonini, M. Barlaud, P. Mathieu, e I. Daubechies. Image Coding Using Wavelet Transform. *IEEE Transactions on Image Processing*, 1(2):205–220, 1992.

[3] A. Averbuch, D. Lazar, e M. Israeli. Image Compression Using Wavelet Transform and Multiresolution Decomposition. *IEEE Transactions on Image Processing*, 5(1):4–15, January 1996.

[4] Leonardo Vidal Batista. *Compressão de Sinais Eletrocardiográficos Baseada na Quantização Ótima dos Coeficientes da Transformada Cosseno Discreta*. Tese de Doutorado, COPELE/UFPB, 2002.

[5] C. D. Bei e R. M. Gray. An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization. *IEEE Transactions on Communications*, (33):1132–1133, October 1985.

[6] Timothy C. Bell, John G. Cleary, e Ian H. Witten. *Text Compression*. Prentice Hall, Englewood Cliffs, NJ, USA, 1990.

[7] Abraham Bookstein e Samuel T. Klein. Compression of Correlated Bit-Vectors. *Information Systems*, 16(4):387–400, 1991.

[8] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, e Y. LeCun. High Quality Document Image Compression with DjVu. *Journal of Electronic Imaging*, 1998.

[9] C. S. Burrus, R. A. Gopinath, e H. Guo. *Introduction to Wavelets and Wavelet Transforms*. Prentice-Hall, New Jersey, 1998.

[10] P. Cassereau. A New Class of Optimal Unitary Transforms for Image Processing. Dissertação de Mestrado, Dept. of Elec. and Comp. Engineering, M.I.T., Cambridge, MA, May 1985.

[11] Charilaos Christopoulos, Touradj Ebrahimi, e Athanassios N. Skodras. JPEG2000: The New Still Picture Compression Standard. In *ACM Multimedia 2000*, Los Angeles, CA, USA, November 2000.

[12] Charilaos Christopoulos, Athanassios N. Skodras, e Touradj Ebrahimi. The JPEG2000 Still Image Coding System: an Overwiew. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127, November 2000.

[13] P. C. Cosman, S. M. Perlmutter, e K. O. Perlmutter. Tree-structured Vector Quantization with Significance Map for Wavelet Image Coding. *Proceedings of IEEE Data Compression Conference (DCC)*, pp. 33–41, April 1995.

[14] R. E. Crochiere, S. M. Webber, e J. K. L. Flanagan. Digital Coding of Speech in Sub-bands. *The Bell System Technical Journal*, 55:1069–1086, 1976.

[15] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, Philadelphia, PA, 1992.

[16] G. M. Davis e A. Nosratinia. Wavelet-based Image Coding: An Overview. *Applied and Computational Control, Signals, and Circuits*, 1(1), Spring 1998.

[17] R. L. de Queiroz, T. Q. Nguyen, e K. R. Rao. The GenLOT: Generalized Linear-Phase Lapped Orthogonal Transform. *IEEE Transactions on Signal Processing*, 40:497–507, March 1996.

[18] R. L. de Queiroz e T. D. Tran. *The Handbook on Transforms and Data Compression*, chapter Lapped transforms for image compression, pp. 197–265. CRC Press, October 2000.

[19] V. E. DeBrunner, L. Chen, e H.-L. Li. Lapped Multiple Bases Algorithms for Still Image Compression Without Blocking Effect. *IEEE Transactions on Image Processing*, 6(9):1316–1321, September 1997.

[20] Aaron Deever e Sheila S. Hemami. What's Your Sign? Efficient Sign Coding for Embedded Wavelet Image Coding. In *Proceedings of IEEE Data Compression Conference (DCC)*, pp. 273–282, 2000.

[21] P. Desarte, B. Macq, e D. T. M. Slock. Signal-Adapted Multiresolution Transform for Image Coding. *IEEE Transactions on Information Theory*, 38(2):897–904, March 1992.

[22] R. A. DeVore, B. Jawerth, e B. J. Lucier. Image Compression through Wavelet Transform Coding. *IEEE Transactions on Information Theory*, 38(2):719–746, 1992.

[23] R. E. Van Dyck. *Trellis coded quantization*. McGraw-Hill, yearbook of science & technology edition, 1997.

[24] R. E. Van Dyck, N. Moayeri, T. G. Marshall Jr., e M. Chin. Video coding using entropy-constrained trellis coded quantization. In *Proc. Applications of Digital Image Processing XVII, SPIE Inter. Symp. on Optics, Imaging and Instr.*, July 1994.

[25] A. M. Eskicioglu e P. S. Fischer. Image Quality Measures and Their Performance. *IEEE Transactions on Communications*, 3(12):2959–2965, December 1995.

[26] W. Finamore, M. Carvalho, e J. Kie. Lossy Compression with the Lempel-Ziv Algorithm. In *Proceedings of 11th Brasilian Telecommunication Conference*, pp. 141–146, 1993.

[27] M. Flierl, T. Wiegand, e B. Girod. Multihypothesis Pictures for H.26L. In *Proceedings International Conference on Image Processing, ICIP-2001*, Thessaloniki, Greece, October 2001.

[28] Zhihai He e S. Mitra. A Unified Rate-Distortion Analysis Framework for Transform Coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(12):1221–1236, December 2001.

[29] I. Höntsch, L. J. Karan, e R. J. Safraanek. A Perceptually Tuned Embedded Zerotree Image Coder. *Proceedings of ICIP'97*, pp. 41–44, 1997.

[30] P. G. Howard. Text Image Compression using Soft Pattern Matching. *The Computer Journal*, 40(2/3):146–154, 1997.

[31] P. G. Howard, L. Bottou, e Y. Bengio. The Z-Coder Adaptive Binary Coder. In *Proceedings of IEEE Data Compression Conference (DCC)*, 1998.

[32] P. G. Howard, Faouzi Kossentini, Bo Martins, Soren Forchhammer, William J. Rucklidge, e Fumitaka Ono. The Emerging JBIG2 Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7):838–848, November 1998.

[33] P. G. Howard e J. S. Vitter. *Images and Text Compression*, chapter Practical Implementations of Arithmetic Coding. Kluwer Academic Publishers, 1992.

[34] C.-M. Huang, Q. Bi, G. S. Stiles, e R. W. Harris. Fast Full Search Equivalent Enconding Algorithms for Image Compression Using Vector Quantization. *IEEE Transactions on Image Processing*, 1(3):413–416, July 1992.

[35] D. A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. In *Proceedings of the IRE*, volume 40-9, pp. 1098–1101, September 1952.

[36] Yeong-Cherng Lu Wen-Jyi Hwang e Yi-Ching Zeng. Fast Block-Matching Algorithm for Video Coding. *Electronics Letters*, 33(10):833–835, May 1997.

[37] Stuart Inglis. Lossless Document Image Compression, 1999.

[38] Stuart Inglis e Ian H. Witten. Compression-Based Template Matching. In *Proceedings of IEEE Data Compression Conference (DCC)*, pp. 106–115, 1994.

[39] ITU. Video Coding for Low Bit Rate Communication. ITU-T Recommendation H.263, March 1996.

[40] ITU-T. Information Technology - Coded Representation of Picture and Audio Information - Progressive Bi-Level Image Compression. ITU-T Recommendation T.82, 1993.

[41] ITU-T. Video Codec for Audiovisual Services at p×64 kbit/s, (International Telecommunication Union - Telecommunication Standardisation Sector) Recommendation H.261, 1993.

[42] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, USA, 1989.

[43] J. R. Jain e A. K. Jain. Displacement Measurement and its Application in Interframe Image Coding. *IEEE Transactions on Communication*, 29:1799–1808, December 1981.

[44] N. Jayant, J. Johnston, e R. Safranek. Signal Compression Based on Models of Human Perception. *Proceedings of the IEEE*, 81(10):1385–1422, 1993.

[45] N. S. Jayant e P. Noll. *Digital Coding of Waveforms*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[46] Glen G. Langdon Jr. An Introduction to Arithmetic Coding. *IBM Journal of Research and Development*, 28(2):135–149, 1984.

[47] ISO/IEC JTC1/SC29/WG1. JBIG2 Final Comitee Draft FCD-14492.

[48] J. Karlekar, P. G. Poonacha, e U. B. Desai. Image Compression Using Zerotree and Multistage Vector Quantization. *Proceedings of ICIP'97*, 2:610–613, 1997.

[49] A. Kjoelen, S. E. Umbaugh, e M. Zuke. Compression of Skin Tumor Images – Wavelet/Vector Quantization Methods for Reducing the Time, Cost and Bandwidth of Storing and Transmitting Data. *IEEE Engineering in Medicine and Biology*, pp. 73–80, June 1998.

[50] Rob Koenen. Overview of the MPEG-4 Standard. Relatório Técnico JTC1/SC29/WG11 N4030, ISO/IEC, March 2001.

[51] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, e T. Ishiguro. Motion-compensated Interframe Coding for Video Conferencing. In *Proceedings of NTC81*, pp. C9.6.1–9.6.5, Los Angeles, November 1981.

[52] M. Kunt, A. Ikonomopoulos, e M. Kocher. Second-Generation Image-Coding Techniques. *Proceedings of the IEEE*, 73(4):549–574, 1985.

[53] A. S. Lewis e G. Knowles. Image Compression Using the 2-D Wavelet Transform. *IEEE Transactions on Image Processing*, 1(2):244–250, April 1996.

[54] J. Liang e T. W. Parks. Image Coding Using Translation Invariant Wavelet Transforms with Symmetric Extensions. *IEEE Transactions on Image Processing*, 7(5):762–769, May 1998.

[55] J. Lu, V. R. Algazi, e R. R. Estes. Comparison of Wavelet Image Coders Using the Picture Quality Scale (PQS). In Harold H. Szu, editor, *Proceedings of SPIE - Wavelet Applications II*, volume 2491, pp. 1119–1130, April 1995.

[56] T. Luczak e W. Szpankowski. A Suboptimal Lossy Data Compression Based on Approximate Pattern Matching. *IEEE Transactions on Information Theory*, 43:1439–1451, 1997.

[57] F. Madeiro, M. S. Vajapeyam, M. R. Morais, B. G. Aguiar Neto, e M. S. de Alencar. Multiresolution Codebook Design for Wavelet/VQ Image Coding. *Proceedings ICPR'2000 (International Conference on Pattern Recognition), year =*.

[58] S. Mallat. Multifrequency Channel Decompositions of Images and Wavelet Models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):2091–2110, 1989.

[59] S. Mallat. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Transactions on Patt. Anal. and Mach. Intell.*, 11(7):674–693, July 1989.

[60] S. Mallat. *A wavelet Tour of Signal Processing - Second Edition.* Academic Press, 1999.

[61] Henrique S. Malvar. The LOT: a Link Between Block Transform Coding and Multirate Filter Banks. In *Proceedings International Symp. Circuits and Systems*, pp. 835–838, Espoo, Finland, June 1988.

[62] Henrique S. Malvar. Reduction of Blocking Effects in Image Coding with a Lapped Orthogonal Transform. In *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, pp. 781–784, Gasglow, Scotland, April 1988.

[63] Henrique S. Malvar. *Signal Processing with Lapped Transforms.* Artech House, Boston, MA, USA, 1992.

[64] Henrique S. Malvar. Biorthogonal and Nonuniform Lapped Transforms for Transform Coding with Reduced Blocking and Ringing Artifacts. *IEEE Transactions on Signal Processing, Special Issue Multirate Systems, Filter Banks, Wavelets, Applications*, 46:1043–1053, April 1998.

[65] Henrique S. Malvar. Progressive Wavelet Coding of Images. In *Proceedings of IEEE Data Compression Conference (DCC)*, pp. 336–343, Salt Lake City, UT, USA, March 1999.

[66] Henrique S. Malvar. *Optimal Pre- and Post-Filtering in Noisy Sampled-Data Systems.* Tese de Doutorado, Mass. Inst. Techn., 1986, August Cambridge, MA.

[67] Henrique S. Malvar e David H. Staelin. The LOT: Transform Coding Without Blocking Effects. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(4):553–559, April 1989.

[68] N. K. Mandal, S. Panchanathan, e T. Aboulnasr. Choice of Wavelets for Image Compression. *Lecture Notes in Computer Science*, 1133:239–249.

[69] D. Marpe, G. Blättermann, G. Heising, e T. Wiegand. Video Compression Using Context-based Adaptive Arithmetic Coding. In *Proceedings ICIP-2001 (IEEE International Conference on Image Processing)*, San Fransisco, CA, USA, October 2001.

[70] D. Marpe, H. Schwarz, e T. Wiegand. Context-Based Adaptive Binary Arithmetic Coding in the H.264 / AVC Video Compression Standard (invited paper). *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, 2003.

[71] Y. Meyer. *Wavelets: Algorithms and Applications*. Society of Industrial and Applied Mathematics, Philadelphia, 1993.

[72] M. Miyahara, K. Kotani, e V. R. Algazi. Objective Picture Quality Scale (PQS) for Image Coding. *IEEE Transactions on Communications*, 46(9):1215–1226, 1998.

[73] Alistair Moffat, Radford M. Neal, e Ian H. Witten. Arithmetic Coding Revisited. *ACM Transactions on Information Systems*, 16(3):256–294, 1998.

[74] Marcos R. A. Morais. Proposta de Tese: Codificação de Imagem e Vídeo com casamento Parcial de Padrões em Transformadas de Bloco, Fevereiro 2003.

[75] Joint Video Team of ISO/IEC JTC1/SC29/WG11 e ITU-T SG16/Q.6 Doc. JVT-G050. *Draft ITU-T Recommendation H.264 and Draft ISO/IEC 14496-10 AVC*, Mar 2003.

[76] Antonio Ortega e K. Ramchandran. Rate-Distortion Techniques in Image and Video Compression. *IEEE Signal Processing Magazine*, 15(6):23–50, 1998.

[77] W. Osberger, A. J. Maeder, e D. McLean. An Objective Quality Assessment Technique for Digital Image Sequences. *Proceedings of the IEEE International Conference on Image Processing (ICIP'96)*, I:897–900, 1997.

[78] Davis Yen Pan. Digital Audio Compression. *Digital Technical Journal of Digital Equipment Corporation*, 5(2):28–33, 1993.

[79] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, e R. B. Arps. An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder. *IBM Journal of Research and Development*, 32(6):717–726, 1988.

[80] William B. Pennebaker e Joan L. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, NY, USA, 1993.

[81] S. M. Perlmutter, K. O. Perlmutter, e P. C. Cosman. Vector Quantization with Zerotree Significance Map for Wavelet Image Coding. *Proceedings of 29th Asilomar Conf. on Signal, Systems and Computers, Pacific Grove, CA*, November 1995.

[82] L. Po e W. Ma. A Novel Four-Step Search Algorithm for Fast Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:313–317, June 1996.

[83] W. K. Pratt. *Digital Image Processing*. John Wiley & Sons, 2 edition, 1991.

[84] A. Puri, R. L. Schmidt, e B. G. Haskell. Performance Evaluation of the MPEG-4 Visual Coding Standard. In *Proceedings of the SPIE*, volume 3309, pp. 417–433, San Jose, CA, USA, January 1998.

[85] V. Ramasubramanian e K. K. Paliwal. Fast Nearest-neighbor Search Based on Voronoi Projections and Its Application to Vector Quantization Encoding. *IEEE Transactions on Speech and Audio Processing*, 7(2):221–226, March 1999.

[86] M. G. Ramos, S. S. Hemani, e M. A. Tamburro. Psyhovisually-Based Multiresolution Image Segmentation. *Proceedings of ICIP'97*, pp. 66–69, 1997.

[87] X. Ran e N. Farvardin. A Perceptually Motivated Three-Component Image Model – Part I: Description of the Model. *IEEE Transactions on Image Processing*, 4(4):401–415, 1995.

[88] X. Ran e N. Farvardin. A Perceptually Motivated Three-Component Image Model – Part II: Applications to Image Compression. *IEEE Transactions on Image Processing*, 4(4):430–447, 1995.

[89] K. Rao e P. Yip. *The Discrete Cosine Transform.* Academic Press, New York, 1990.

[90] Viresh Ratnakar. *Quality-Controlled Lossy Image Compression.* Tese de Doutorado, University of Wisconsin – Madison, 1997.

[91] Viresh Ratnakar e Miron Livny. RD-OPT: An Efficient Algorithm For Optimizing DCT Quantization Tables. In *Proceedings of IEEE Data Compression Conference (DCC)*, pp. 332–341, 1995.

[92] Viresh Ratnakar e Miron Livny. Extending RD-OPT with Global Thresholding for JPEG Optimization. In *Proceedings of IEEE Data Compression Conference (DCC)*, pp. 379–386, 1996.

[93] M. M. Reid, R. J. Millar, e N. D. Black. Second-Generation Image-Coding: An Overview. *ACM Computing Surveys*, 29(1):3–29, 1997.

[94] Iain E.G. Richardson. *Video Codec Design - Developing Image and Video Compression Systems.* Wiley, 2002.

[95] Iain E.G. Richardson. *H.264 and MPEG-4 Video Compression - Video Coding for Next-generation Multimedia.* Wiley, 2003.

[96] A. Said e W. A. Pearlman. A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, June 1996.

[97] D. G. Sampson, E. A. B. da Silva, e M. Ghanbari. Wavelet Transform Image Coding Using Lattice Vector Quantisation. *Electronics Letters*, 30(18):1477–1478, 1994.

[98] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[99] J. M. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[100] M. R. Soleymani e S. D. Morgera. An Efficient Nearest Neighbor Search Method. *IEEE Transactions on Communications*, 35(6):677–679, June 1987.

[101] ISO/IEC International standard 13818-2. Generic Coding of Moving Pictures and Associated audio information: Part2 - Video, 1995.

[102] G. Strang e T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[103] Gary J. Sullivan e Thomas Wiegand. Rate-Distortion Optimization for Video Compression. *IEEE Signal Processing Magazine*, pp. 74–90, November 1998.

[104] Wojciech Szpankowski, M. Atallah, e Y. Genin. Pattern Matching Image Compression: Algorithmic and Empirical Results. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:618–627, 1999.

[105] Wojciech Szpankowski, M. Atallah, e Y. Genin. 2D-Pattern Matching Image and Video Compression: Theory, Algorithms, and Experiments. *IEEE Transactions on Image Processing*, 11:318–331, 2002.

[106] S. C. Tai, C. C. Lai, e Y. C. Lin. Two Fast Nearest Neighbor Searching Algorithms for Image Vector Quantization. *IEEE Transactions on Communications*, 44(12):1623–1628, December 1996.

[107] D. Taubman. High Performance Scalable Image Compression with EBCOT. *IEEE Transactions on Image Processing*, 9(7):1158–1170, 2000.

[108] T. D. Tran, R. de Queiroz, e T. Q. Nguyen. The Generalized Lapped Biorthogonal Transform. In *Proceedings IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Seattle, WA, May 1997.

[109] Trac D. Tran e Truong Q. Nguyen. A Progressive Transmission Image Coder Using Linear Phase Uniform Filterbanks as Block Transforms. *IEEE Transactions on Image Processing*, 8(11), November 1999.

[110] Trac D. Tran e R. Safranek. A Locally Adaptive Perceptual Masking Threshold Model for Image Coding. In *IEEE Proceedings International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pp. 1882–1885, May 1996.

[111] Kyoya Tsutsui, Hiroshi Suzuki, Osamu Shimoyoshi, Mito Sonohara, Kenzo Akagiri, e Robert M. Heddle. ATRAC: Adaptive Transform Acoustic Coding for MiniDisc. In *Proceedings of the 93rd Audio Engineering Society Convention*, San Fransisco, CA, USA, October 1992.

[112] Chengjie Tu, J. Liang, e Trac D. Tran. Adaptive Runlength Coding. *IEEE Signal Processing Letters, to appear.*

[113] Chengjie Tu e Trac D. Tran. Context Based Entropy Coding of Block Transform Coefficients for Image Compression. *IEEE Transactions on Image Processing, to appear*, November 2002.

[114] M. Unser. Ten Good Reasons for Using Spline Wavelets. *Proceedings SPIE, Wavelets Applications in Signal and Image Processing V*, 3169:422–431, 1997.

[115] C. J. van den Branden Lambrecht e O. Verscheure. Perceptual Quality Measure using a Spatio-Temporal Model of the Human Visual System. In *Proceedings SPIE*, volume 2668, pp. 450–461, Mar 1996.

[116] M. Vetterli. Multi-dimensional Sub-band Coding: Some Theory and Algorithms. *Signal Processing*, 6:97–112, 1984.

[117] M. Vetterli e C. Herley. Wavelets and Filter Banks: Theory and Design. *IEEE Transactions on Signal Processing*, 40(9):2207–2232, September 1992.

[118] J. D. Villasenor, B. Belzer, e J. Liao. Wavelet Filter Evaluation for Image Compression. *IEEE Transactions on Image Processing*, 4(8):1053–1060, August 1995.

[119] G. K. Wallace. The JPEG Still Picture Compression Standard. *Communications of the ACM*, 34:30–44, April 1991.

[120] P. H. Westerink, D. E. Boekee, J. Biemond, e J. W. Woods. Sub-band Coding of Images Using Vector Quantization. *IEEE Transactions on Communications*, 36(6):713–719, 1988.

[121] T. Wiegand, M. Lightstone, D. Mukherjee, T. George, e S. K. Mitra. Rate-distortion Optimized Mode Selection for Very Low Bit Rate Video Coding and the Emerging H.263 Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(2):182–190, April 1996.

[122] I. Witten, T. Bell, H. Emberson, e S. Inglis. Textual Image Compression: Two-Stage Lossy/Lossless Encoding of Textual Images, 1994.

[123] I. Witten, R. Neal, e J. Cleary. Arithmetic Coding for Data Compression. *Communications of the ACM*, 30:520–540, 1987.

[124] J. W. Woods e S. D. O'Neil. Subband Coding of Images. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34:1278–1288, 1986.

[125] Z. Xiong, K. Ramchandran, e M. T. Orchard. Space-Frequency Quantization for Wavelet Image Coding. *IEEE Transactions on Image Processing*, 6(5):677–693, May 1997.

[126] Zixiang Xiong, Onur Guleryuz, e Michael T. Orchard. A DCT-based Embedded Image Coder. *IEEE Signal Processing Letters*, 3(11):289–290, 1996.

[127] Y. Ye e P. Cosman. Fast and Memory Efficient JBIG2 Encoder. In *Proceedings of the 2001 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001)*, Salt Lake City, Utah, May 2001.

[128] Y. Ye e P. Cosman. Speedup Techniques for Text Image Compression With JBIG2. In *Proceedings of the the 35th IEEE Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, November 2001.