UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

VEBit: um Algorítmo para Codificação de Vídeo com Escalabilidade

por

GASPARE GIULIANO ELIAS BRUNO

Dissertação submetida a avaliação,
como requisito parcial para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. José Valdeni de Lima
Orientador

Porto Alegre, abril de 2003.

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Bruno, Gaspare Giuliano Elias

VEBit: um Algorítmo para Codificação de Vídeo com Escalabilidade / por Gaspare Giuliano Elias Bruno. — Porto Alegre: PPGC da UFRGS, 2003.

95 f.: il.

Dissertação (mestrado) — Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Orientador: Lima, José Valdeni de.

1. Codificação de Vídeo. 2. Escalabilidade. 3. Transmissão em Camadas. 4. Bit-planes. 5. MPEG 4. I. Lima, José Valdeni de. II. Título.

Esta dissertação não teria sido possível sem a ajuda de Valter Roesler
e de meu orientador Valdeni de Lima.
Um agradecimento especial para minha namorada, Érica Marques,
por sua ajuda e compreensão.

Sumário

Lista de Abreviaturas

ALM	Adaptive Layered Multicast
CIF	Common Intermediate Format
CRT	Catode Ray Tube
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
FGS	Fine Granularity Scalability
GPS	Generalized Processor Sharing
HDTV	High Definition Television
JPEG	Joint Photographic Experts Group
MAD	Mean Absolute Difference
MJPEG	Motion JPEG
MPEG	Moving Picture Experts Group
MSE	Mean Squared Error
NTSC	National Television System Committee
PLM	Packet-pair Layered Multicast
PPM	Portable PixMap
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter CIF
RLC	Receiver-driven Congestion Control
RLE	Run Lenght Coding
RLM	Receiver-driven Layered Multicast
SAM	Sistema Adaptativo Multimídia
SDL	Simple Direct Layer
SNR	Signal-Noise Ratio
VLC	Variable Lenght Coding

Lista de Figuras

Lista de Tabelas

Resumo

A codificação de vídeo de modo a permitir a escalabilidade durante a transmissão tem se tornado um tópico de grande pesquisa nos últimos anos. Em conjunto com um algoritmo de controle de congestionamento, é possível a criação de um ambiente de transmissão multimídia mais apropriado.

Esta dissertação apresenta um algoritmo de codificação de vídeo escalável baseado em *bit-planes*. O modelo de codificação do vídeo utiliza 3D-DCT para codificação espacial e temporal e um quantizador escalar semelhante ao empregado no MPEG 4. A técnica de escalabilidade em *bit-planes* implementada permite a divisão da saída do codificador em taxas de bits complementares e com granularidade fina.

Este algoritmo é parte integrante do projeto SAM (Sistema Adaptativo Multimídia), que busca criar um ambiente de transmissão multimídia adaptativo em tempo real. Este projeto esta em desenvolvimento na tese de doutorado de Valter Roesler. O algoritmo proposto e implementado é capaz de transmitir de forma unidirecional vídeos de baixa movimentação.

Os resultados dos testes realizados com a implementação feita mostram que a solução proposta é flexível em relação a tecnologia disponível de transmissão através do ajuste no número de camadas e suas respectivas especificações de banda. Os testes realizados apresentaram um desempenho aceitável para codificação e decodificação de vídeo em tempo real. A taxa de compressão apresentou resultados satisfatórios na transmissão em ambientes de baixa velocidade para as camadas inferiores, bem como taxas de transmissão para ambientes baseados em ADSL, cable modem e rede local para as camadas superiores. Com relação a qualidade do vídeo, esta varia de acordo com o grau de movimentação do mesmo. Por exemplo, no modelo "*talking-head*", comum em videoconfêrencias, a qualidade se mostrou viável para ambientes de baixa velocidade (56 kbit/s).

Palavras-chave: Codificação de Vídeo, Escalabilidade, Transmissão em Camadas, Bit-planes, MPEG 4.

TITLE: "VEBIT: AN ALGORITM FOR VIDEO CODING WITH SCALABILITY"

Abstract

The video coding with scalability has become a topic of many studies in the last years. The use of an algorithm of congestion control in video coding with scalability makes possible to create a more appropriated environment of multimedia transmissions.

This dissertation presents an algorithm of video coding with scalability based on bit-planes. The model of video codification uses 3D-DCT for spatial and temporal coding and, a scalar quantizer based on MPEG 4. The scalability technique we have developed, based on bit-planes allows the output to be divided in complementary bit rates with fine granularity.

This algorithm is part of the SAM project, which tries to create an environment for adaptive multimedia transmission in real time. This project is under development in the doctor thesis of Valter Roesler. The proposed algorithm and the resulting implementation is capable of transmitting low movement videos in an unidirectional way.

The results of the implementation tests show that the proposed solution is flexible in relation to the technology of transmission. However, the adjustment of the number of layers and their respective bandwidth is necessary. The tests also have shown an acceptable performance for video coding and decoding in real time. The bandwidth rate has allowed the transmission in the low rates for lower layers, as well as, high rates in upper layers such as ADSL networks. In relation to the video quality, we can say it varies according to its own motion level. In "talking head" videos, for instance, the quality has shown to be suitable in low rates.

Keywords: Video Coding, Scalability, Layered Transmission, Bit-plane, MPEG 4.

1 Introdução

O conceito de distribuição de mídias surge simultaneamente com o estabelecimento da tecnologia de apresentação de dados multimídia nos computadores pessoais. Áudio e vídeo são digitalizados, codificados e armazenados em arquivos de computador, e para tornar possível a apresentação destas mídias há necessidade de um programa que decodifique e apresente o arquivo ao usuário.

O primeiro modelo para distribuição de mídias foi desenvolvido com a utilização do método de *download* destes arquivos [WU 2001]. Arquivos comprimidos eram disponibilizados em servidores na Internet, os usuários recebiam os arquivos em seus computadores e, após a descompressão, visualizavam-no usando um utilitário para apresentação multimídia. No entanto, esta não foi uma solução satisfatória para um elevado índice de usuários, devido ao espaço necessário para armazenamento e às conexões lentas, que exigiam do usuário um período de tempo que nem sempre poderia ser disponibilizado para aguardar a recepção completa do arquivo. A solução encontrada foi a criação de um método de transmissão *on-the-fly*, no qual o usuário é capaz de visualizar a mídia enquanto a recebe.

Com a utilização deste método e o crescente aumento na velocidade dos enlaces, a distribuição multimídia na Internet tem alcançado um grau crescente de popularidade. Contudo, a Internet foi concebida para comunicação de dados imutáveis no tempo, portanto satisfazer às necessidades de uma distribuição correta de mídias temporais é um desafio. Por exemplo: a Internet se caracteriza por grandes variações de banda e pela heterogeneidade de tecnologias de acesso (modem, cable, xDSL, entre outras). A mudança constante no estado da rede — como congestionamento de dados em roteadores — também é um fator de influência. A Internet ainda apresenta uma porcentagem alta de perdas de pacotes. Assim sendo, para atingir um nível aceitável para a distribuição de vídeo na Internet, é necessário vencer vários desafios técnicos em codificação e transmissão.

Os primeiros programas de transmissão de vídeo pela Internet operavam numa taxa fixa de bits por segundo. A maioria dos usuários contentavam-se com este sistema, mas há os que dispõem de mais recursos, como uma conexão mais rápida, e estes não podem aproveitar a capacidade adicional de banda disponível. Em contrapartida, há os que possuem uma conexão inferior à taxa de transmissão do servidor e não conseguem sequer visualizar corretamente a mídia.

Outro ponto problemático encontra-se na heterogeneidade dos equipamentos utilizados para recepção. O número de celulares, PDAs, notebooks e outros tipos de dispositivos com recursos limitados de hardware que são utilizados na recepção de dados multimídia é crescente, sendo que estes aparelhos não possuem recursos necessários para a melhor visualização da mídia.

Uma das soluções encontradas para minimizar este problema foi o uso de substituição de mídias[GOM 2000]. Caso o usuário não disponha de banda suficiente para receber a mídia ou não possua recursos de hardware, um vídeo pode ser substituído por uma imagem e/ou um áudio por um texto. No entanto, este processo pode incorrer em uma percepção deficiente pelo usuário da informação a ele apresentada, pois a imagem não contêm todos os recursos visuais presentes no vídeo, por exemplo.

Outra solução é a codificação da informação em várias taxas de bits [CON 2001].

O usuário seleciona a mídia que mais se adapta à sua característica de banda e de recursos. No entanto, este tipo de processo não leva em consideração as mudanças dinâmicas presentes na Internet. Uma conexão pode ficar congestionada durante a transmissão e prejudicar a visualização dos dados. Deve-se considerar também a necessidade de um processamento superior para codificar uma mídia em várias taxas, devido ao fato de que cada taxa gera uma codificação independente.

O primeiro programa comercial a incorporar uma técnica de codificação escalável surgiu no RealSystem G2 [CON 2001]. A metodologia aplicada foi a codificação de múltiplas representações do mesmo vídeo, cada uma com taxa de bits e qualidade diferentes. Um programa presente no receptor monitorava as condições de rede e selecionava a taxa que melhor aplicava-se àquele determinado momento. No caso da existência de congestionamento na rede ou perda de dados, o receptor instruía o servidor a reduzir a taxa para uma qualidade inferior. Esta tecnologia ficou conhecida como SureStream. A maneira como foi implementada exige que as várias taxas sejam codificadas de modo independente, o que gera um maior custo computacional. Além disso, ao ser detectado um problema, o cliente requisita ao servidor a interrupção da transmissão da mídia atual e solicita uma nova mídia em taxa inferior. Considerando que este processo de comunicação leva algum tempo, o usuário permanece com sua apresentação interrompida já que neste ínterim a nova mídia não é apresentada.

Em contraste, a codificação escalável atual, ou codificação em camadas, permite que se obtenha diferentes taxas de bits complementares [WU 2001]. A mídia original é codificada em um número discreto de camadas complementares, organizadas hierarquicamente de modo a prover um refinamento progressivo. Quanto maior o número de camadas que um usuário recebe, maior será a qualidade da mídia apresentada. Assim sendo, o usuário pode requisitar o número de camadas que melhor lhe convêm em determinado momento, aumentando ou diminuindo esse número de acordo com a variação na rede. Em geral, esta técnica necessita de um menor processamento do que a técnica apresentada anteriormente, onde cada taxa era codificada separadamente, pois as taxas são codificadas progressivamente. Quando cada camada for transmitida em uma conexão independente, é possivel evitar a interrupção na apresentação através de um controle do número de camadas recebidas. Em determinados casos, é possível obter adaptabilidade em outros dispositivos através da variação da resolução ou do processamento. Esta técnica de codificação, em conjunto com uma técnica de transmissão que seja capaz de selecionar o número ótimo de camadas em um determinado momento, de acordo com as características de rede, forma um ambiente de distribuição multimídia próximo ao ideal.

Esta pesquisa versa sobre a codificação de mídias de vídeo de maneira escalável. Os dois primeiros capítulos apresentam o estado da arte na área da codificação de vídeo e escalabilidade. São apresentadas as técnicas mais comuns empregadas na codificação para a compressão dos dados de um vídeo de maneira temporal e espacial com o objetivo de atualizar o leitor nestas técnicas. Apesar de alguns padrões de codificação, como o MPEG 4 (*Moving Picture Experts Group*, versão 4), trabalharem com a codificação de objetos em uma cena, este tipo de compressão encontra-se além do escopo deste trabalho. A seguir, no capítulo três, são apresentadas técnicas empregadas em alguns padrões de codificação para atingir escalabilidade.

O capítulo quatro apresenta alguns protocolos de transmissão em camadas. Mesmo sendo o foco desta dissertação a codificação, este ítem oferece ao leitor dados

básicos sobre pesquisas na área, objetivando apresentar de que modo um codificador pode atuar em sua máxima capacidade durante uma transmissão.

Para finalizar, apresenta-se uma implementação rústica de um codificador baseado na técnica de escalabilidade FGS [LI 2001] presente no padrão MPEG 4. O objetivo deste codificador é apenas apresentar a implementação de uma técnica e servir de ponto de partida para novos estudos, na busca de implementações mais avançadas. Ele é baseado na compressão tridimensional DCT (*Discrete Cosine Transform*), possui desempenho suficiente para transmissões em tempo real e qualidade aceitável para videoconferências. A utilização prevista para esta implementação é atuar como suporte à codificação de vídeo na plataforma SAM (Sistema Adaptativo Multimídia) [ROE 2002a], atualmente em desenvolvimento pelo professor Valter Roesler em sua tese de doutorado. Uma descrição deste sistema é apresentada no capítulo cinco.

2 Codificação de Vídeo

Na área da multimídia existem duas classes de mídias: as discretas e as contínuas. As mídias discretas são aquelas que representam informações estáticas, tais como textos e imagens. Mídias contínuas, também conhecidas como mídias temporais, possuem alteração no tempo: nesta classe encontram-se inseridos o áudio e o vídeo.

Os seres humanos somente são capazes de perceber sinais audiovisuais analógicos [ROC 90]. Nos sistemas de comunicação digital, estes sinais analógicos devem ser convertidos do meio analógico para o meio digital e codificados através de um codificador (*encoder*) e depois decodificados no decodificador (*decoder*) e retornar para o formato analógico de modo que o usuário possa ouvir o som ou ver a imagem.

Considerando que a quantidade de informação gerada tende a ser muito elevada, além da conversão de analógico para digital (A/D) e vice versa (D/A), um codificador/decodificador deve realizar a compressão dos dados [PAR 99]. Existem duas técnicas de compressão: sem perdas (*lossless*) e com perdas (*lossy*). A compressão sem perdas permite que o sinal digital, ao ser descompactado, seja igual ao sinal original digitalizado pelo codificador, enquanto que a compressão com perdas apresenta uma diferença no decodificador que é chamada de distorção. O objetivo de um compressor com perdas é aumentar a taxa de compressão do dado retirando informações do sinal original.

O compressor a ser usado varia conforme o tipo de mídia a ser transmitida. Em geral, compressores trabalham com os limites da percepção audiovisual humana. Esta técnica é conhecida como Codificação por Percepção (*Perceptual Coding*) e foi inicialmente empregada pela AT&T na telefonia pública. Isto deve-se ao fato de que o ser humano possui limites para a audição e para a visão [ROC 90]. Considerando-se que o olho humano possui maior percepção para a claridade (luminosidade) que para a crominância (intensidade da cor), este fator pode ser determinante para codificação de vídeos, assim, a retirada de algumas informações de cor reduz a quantidade de dados sem que o olho humano perceba grande alteração. O homem também sente dificuldades para perceber variações bruscas na freqüência do sinal.

A maneira mais habitual de reduzir-se uma informação visual é através da eliminação de redundâncias. Os dois modelos mais comuns de redundância são a espacial (mais conhecida como *intraframe*), que explora a idéia que os pixels vizinhos possuem uma forte correlação entre si [ROC 90, KUO 98], e a temporal (também chamada de *interframe*), onde quadros adjacentes de uma seqüência de vídeo podem apresentar pouca mudança [SEZ 93].

Várias técnicas podem ser empregadas para explorar estas redundâncias. A redundância espacial é mais explorada por transformadas. A redundância temporal pode ser removida utilizando-se técnicas como o DPCM (*Differential Pulse Code Modulation*) [KUO 98], que codifica apenas a diferença entre blocos adjacentes, ou pela técnica de compensação de movimento.

Normalmente, um codificador é formado por uma técnica de digitalização e por um compressor. O compressor costuma ser composto por uma técnica de transformada, um processo de quantização, um codificador temporal e um algoritmo de codificação por entropia [KUO 98]. O processo de quantização utilizado pelo codificador busca retirar a informação redundante ou excessiva do sinal. Este processo

é necessário porque a transformada não elimina essa informação, apenas a torna mais aparente. O processo de quantização é onde ocorre a maior parte da perda de informação espacial. O algoritmo de entropia é utilizado para aumentar a taxa de compressão e não causa perdas.

2.1 Digitalização

Em sistemas de telecomunicação tradicional, a informação é transmitida de maneira analógica, representada por um sinal contínuo no tempo. No entanto, na comunicação entre computadores, a maneira ideal de transmissão da informação audiovisual é no formato digital. A conversão de sinais analógicos em digitais é feita através de dois processos: amostragem e quantização [KUO 98].

Amostrar um sinal no tempo significa que o sinal analógico é medido em intervalos regulares de tempo. Este intervalo varia de acordo com o tipo de informação. O teorema de Nyquist informa que o intervalo ideal é igual ao dobro da freqüência da informação original. Por exemplo, a voz possui uma freqüência máxima de 3 kHz, portanto, a freqüência de amostragem ideal é de 6 kHz, ou 6000 vezes por segundo.

A quantização representa os valores amostrados em termos de valores discretos. O número de bits usados para representar cada valor amostrado determina a precisão do processo de quantização. Por exemplo, se a quantização tiver precisão de 3 bits, os valores amostrados pode assumir, no máximo, 8 valores diferentes. Na telefonia tradicional, o padrão empregado para digitalizar a voz humana é de 8 bits por amostra. A figura 2.1 representa um sinal contínuo amostrado no tempo. São definidos 9 níveis de quantização mas apenas 8 são utilizados, pois o último não é atingido. A cada escala de tempo T, o sinal é aproximado ao nível de quantização mais próximo. Cada ponto quantizado gera um valor discreto, que será usado para reconstruir o sinal no futuro. Quanto mais valores de quantização forem definidos, mais próximo o sinal ficará de um nível de quantização predeterminado e maior será a fidelidade do sinal de saída.

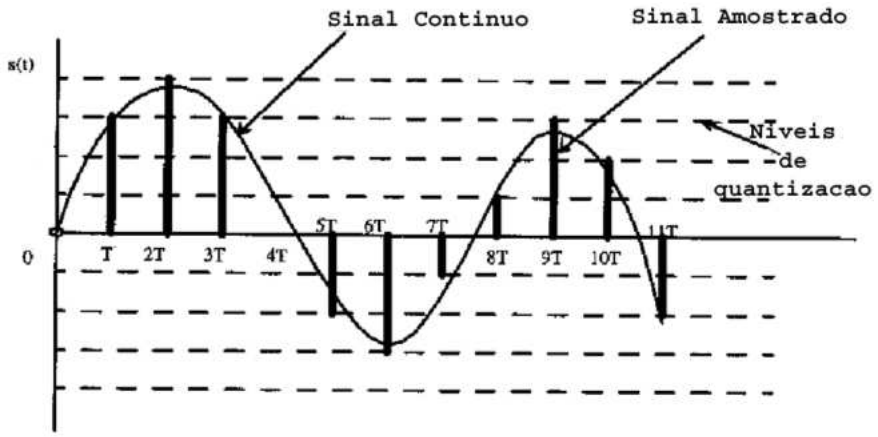

FIGURA 2.1 – Amostragem de um sinal

O processo de digitalização de um vídeo possui características semelhantes. Um vídeo é composto por uma seqüência de quadros estáticos, como fotografias. É

como se uma máquina fotográfica obtivesse fotos em intervalos de tempo regulares. Quando estas fotos são apresentadas na mesma taxa de tempo em que foram obtidas, então temos caracterizado um vídeo. Os filmes costumam utilizar uma taxa de exibição de 25 a 30 quadros por segundo.

A digitalização do vídeo ocorre através da digitalização de cada um destes quadros. A amostragem de cada quadro é feita através da análise de pontos com espaçamento fixo entre si. Uma imagem pode ser digitalizada utilizando 360 pontos (*pixels*) por linha e 288 linhas (padrão CIF). A qualidade final da imagem é definida pela precisão do valor de cada ponto e do espaçamento entre cada amostragem. Quanto menor o espaçamento entre os pontos, maior será a resolução e a qualidade da imagem digital e, conseqüentemente, do vídeo.

A quantização, assim como ocorre no processo de áudio, define a precisão de cada valor amostrado. Um vídeo preto e branco costuma ser caracterizado por um único sinal que define a intensidade de luz presente em cada ponto da imagem, enquanto isso, um vídeo colorido possui três ou mais sinais, dependendo do padrão utilizado. Os padrões mais comuns são o RGB, composto por três sinais com as intensidades de vermelho, verde e azul, e o YUV, composto por um sinal de luminosidade (Y) e dois de cor (U e V). Assim, em uma quantização de 8 bits por sinal, são necessários 24 bits para digitalizar cada pixel da imagem.

2.2 Representação das Cores

A representação de cada pixel de uma imagem depende de como é apresentado o sinal original. Uma imagem preto e branco necessita de apenas 1 bit para representar cada pixel, já uma imagem em tons de cinza (como de televisões preto e branco) necessita de 8 bits (o que gera uma escala de 256 tons, variando do preto ao branco). A representação colorida exige um processo diferente.

É normal a utilização de um sistema de cores primárias para gerar várias cores. O sistema de cores primárias mais usual é baseado nas cores vermelho, verde e azul (RGB). Este esquema é o mesmo usado nos monitores de computadores que utilizam tubo de raios catódicos (CRT), onde a combinação da intensidade de dois ou mais raios geram novas cores. Outro sistema de representação conhecido é baseado nas cores ciano, magenta e amarelo (CMY), que é amplamente utilizado em impressoras.

Em um processo de codificação de vídeo, nenhum destes sistemas é o ideal, pois o olho humano é mais sensível à quantidade de luz que à intensidade da cor em si [ROC 90], e ambos os formatos trabalham diretamente com a cor.

2.2.1 Sistema YUV

Um dos problemas do uso do sistema RGB para representação de vídeos é que o olho humano é mais sensível ao verde, menos ao vermelho e menos ainda ao azul [ROC 90]. Isto porque a cor verde representa de 60% a 70% da informação de luz recebida pelo olho, e o olho humano é mais sensível à luz que à cor. A redução na informação necessária para armazenamento de um pixel sem que o ser humano perceba grandes alterações é possivel a partir da redução das informações de cor. O sistema YUV [PAR 99, MIT 97] permite a separação dos sinais de luz e cor, o que facilita no processo de redução.

Em 1990, a União Internacional de Telecomunicações criou uma recomendação

[MJP 94] que define, através da equação 2.1, como computar a luminância (Luma, ou Y) através das primitivas do RGB.

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \tag{2.1}$$

O componente de luminância Y representa, em escalas de cinza, uma imagem no sistema RGB. As informações de crominância podem ser retiradas através da subtração do componente Y dos componentes R e B do sistema RGB ($R - Y$ e $B - Y$). Este processo gera dois valores de crominância (U e V). A equação completa para conversão de RGB para YUV é apresentada na equação 2.2 [SKO 2001].

$$\begin{vmatrix} Y \\ U \\ V \end{vmatrix} = \begin{vmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{vmatrix} \begin{vmatrix} R \\ G \\ B \end{vmatrix} \tag{2.2}$$

Uma representação no sistema YUV pode ser convertida no sistema RGB utilizando o inverso da matriz apresentada na equação 2.2, conforme apresentado na equação 2.3.

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1.402 \\ 1 & -0.34413 & -0.71414 \\ 1 & 1.772 & 0 \end{vmatrix} \begin{vmatrix} Y \\ U \\ V \end{vmatrix} \tag{2.3}$$

O sistema YUV, amostrado com todos os seus valores, necessita de 3 bytes para cada pixel, o mesmo necessário pelo sistema RGB. No entanto, é possível reduzir a quantidade de informações do sistema YUV durante o processo de amostragem.

Uma imagem é amostrada utilizando blocos de quatro pixels (blocos 2x2). A amostragem padrão, chamada 4:4:4, leva em conta todos os valores presentes nos quatro pixels. Este sistema é o mesmo utilizado na codificação RGB. O sistema YUV considera a característica do olho humano de ser menos receptivo às cores. Isso permite a redução do número de valores de crominância sem grandes alterações na percepção da qualidade. O processo de amostragem 4:2:2 codifica todos os quatro valores de Y, mas apenas dois valores para U e dois para V, o que resulta numa redução de 33%. O padrão 4:2:0, utilizado pelo MPEG I e II, amostra apenas um valor para U e um para V, o que gera uma redução de 50%. A tabela 2.1 [FOG 2002] apresenta a relação existente entre os padrões de amostragem. A redução dos valores depende da implementação a ser utilizada. O processo mais usado é uma simples média aritmética entre dois ou quatro valores.

TABELA 2.1 – Tabela de Padrões de Amostragem

Formato	Cols.	Lin.	Col. UV	Lin. UV	Red. H.	Red. V.
4:4:4	720	480	720	480	-	-
4:2:2	720	480	360	480	2:1	-
4:2:0	720	480	360	240	2:1	2:1
4:1:1	720	480	180	480	4:1	-
4:1:0	720	480	180	120	4:1	4:1

O posicionamento dos pixels na apresentação da imagem também varia. O MPEG I repete os valores de crominância dentro de cada pixel e o MPEG II retira

os valores apenas dos dois primeiros pixels do bloco, por exemplo. A figura 2.2 [POY 96] diagrama este processo.

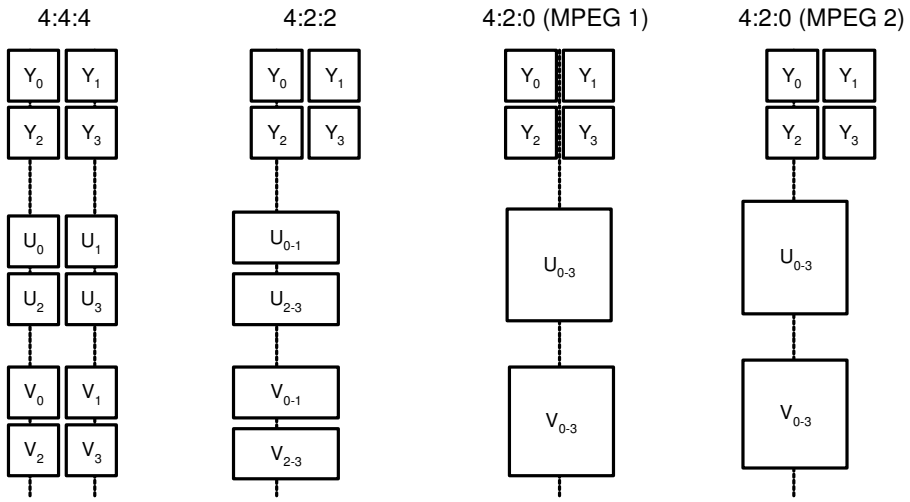

FIGURA 2.2 – Padrões de Amostragem de uma Imagem

2.3 Entropia

Uma das partes fundamentais em um sistema de codificação de vídeo é o algoritmo de entropia. A entropia utiliza a informação de freqüência de ocorrência de determinado símbolo de um alfabeto em um conjunto de dados. Para compactar a informação, é empregado o teorema de Shannon [SHA 48], que define: um símbolo que ocorra em probabilidade p pode ser codificado em $-log_2(p)$ bits, conforme a equação 2.4, onde L é o número de símbolos no alfabeto e o resultado, H, é informado em bits por símbolos.

$$H = \sum_{i=1}^{L} -p_i log_2(p_i) \tag{2.4}$$

A equação 2.4 demonstra que o menor código em bits pode ser associado ao símbolo mais freqüente, enquanto os de menor freqüência poderão ser associados a um maior número de bits, mas ainda assim menor (ou no máximo igual) à sua codificação original. A equação 2.4 é perfeitamente reversível, desde que haja acesso à probabilidade inicial de cada símbolo. Um algoritmo de entropia pode reconstruir o dado original sem perdas (*lossless*).

2.3.1 *Run Length Coding*

Neste modelo de codificação, uma seqüência do mesmo símbolo (*run length*) é representada por uma tupla formada pelo número de ocorrências do símbolo e um valor que represente o mesmo. Por exemplo, a seqüência $(0, 0, 0, 0, 0, 2, 0, 0, 5, 0, 0, 1)$ pode ser codificada nas tuplas $(5, 0), (1, 2), (2, 0), (1, 5), (2, 0), (1, 1)$.

Apesar desta técnica não seguir o teorema de Shannon, ela obtêm grande funcionalidade em imagens digitais. Uma imagem digital costuma apresentar grandes

seqüências de símbolos repetidos, principalmente o símbolo 0. Por isso, o emprego deste modelo de codificação auxilia os modelos baseados no teorema de Shannon a atingir maiores taxas de compressão. Uma variante da codificação por *run length* utilizada na codificação de imagens digitais é a redução apenas do símbolo 0. Por exemplo, se empregarmos o algoritmo na seqüência descrita acima, levando em conta apenas o símbolo 0, temos como resultado $(5,2),(2,5),(2,1)$. Neste caso, cada tupla é formada por um valor que representa a quantidade de zeros antes do símbolo e um valor representando o próprio símbolo. Esta variante é chamada *zero run length coding*.

A implementação real da codificação *run length* (RLE) varia de acordo com a padronização. Por exemplo, a padronização do JPEG (empregada também no MPEG I e II) emprega símbolos de 8 bits, onde os quatro primeiros bits do símbolo indicam o número de zeros consecutivos e os quatro últimos bits indicam o número de bits significativos que formam o número após a seqüência de zeros. Isso permite codificar uma seqüência de até 16 zeros seguidos em, no máximo, dois bytes significativos. Levando em conta que o JPEG codifica em blocos de 64 bytes, este algoritmo atinge altas taxas de compressão.

2.3.2 Codificação Huffman

A codificação Huffman [LIU 95, PAR 99] trabalha com uma tabela (chamada *codeword*) montada a partir da análise de freqüência de cada símbolo. Por exemplo, a seqüência ACBDABA possui a seguinte distribuição: O símbolo A aparece três vezes $(3/7 = 0,43)$, o símbolo B aparece duas vezes $(0,29)$, e os símbolos C e D apenas uma vez $(0,14)$.

O próximo passo é montar uma árvore de codificação. Esta árvore é montada alinhando-se os símbolos de acordo com a sua probabilidade. Conectam-se os símbolos de menor probabilidade entre si e cria-se um novo símbolo (imaginário), cuja probabilidade é a soma dos dois símbolos conectados. Repete-se a operação até que sobrem apenas dois símbolos. Durante o processo, a cada par de símbolos são atribuídos os bits 0 e 1. A figura 2.3 apresenta a árvore de codificação da seqüência ACBDABA. Ao símbolo mais frequente (A) é atribuído apenas um bit (0). O outro bit (1) representa o restante do alfabeto. Novamente, o segundo símbolo mais freqüente (B) recebe um bit (0). O processo se repete até que todos os símbolos estejam conectados a um galho da árvore.

A tabela de codificação é montada seguindo o caminho da ramificação mais externa da árvore, até que todos os símbolos sejam atingidos. No caso desta seqüência, o símbolo A tem como valor o bit 0, o símbolo B é igual a 10, o símbolo C é 110 e o D é 111. Por um processo de substituição, a seqüência inicial ACBDABA é codificada como 0110101110100, assim, são necessários apenas treze bits para armazenar a seqüência.

O número ótimo de bits a ser utilizado para cada símbolo é descrito por $log_2(1/p)$, onde p é a probabilidade de ocorrência de um determinado símbolo. Quando utilizados símbolos de 8 bits, o pior caso ocorrerá quando o símbolo aparecer uma única vez entre os 256 símbolos possíveis. Portanto, o número ótimo para este símbolo será definido por $log_2(1/256)$, ou seja, 8 bits. Se a probabilidade subir para 50% (1/2), o número ótimo de bits cai para 1. O desempenho da compressão dos dados depende da distribuição dos símbolos. O pior caso é encontrado quando a distribuição é uniforme.

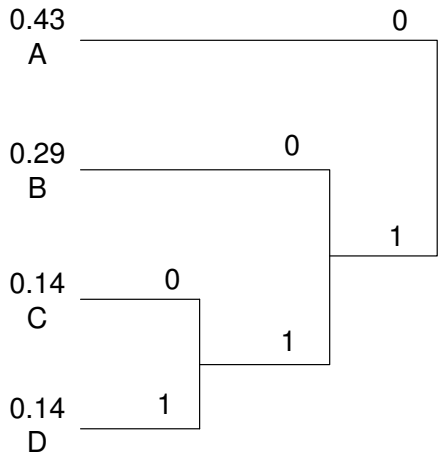

FIGURA 2.3 – Árvore de Codificação

O processo de decodificação requer um custo computacional menor, já que não é necessário realizar o cálculo da probabilidade de cada símbolo e nem a montagem da árvore de codificação. No entanto, para que um computador possa decodificar uma mensagem, ele deve ter acesso à árvore de codificação. A decodificação é feita por um algoritmo inverso ao da codificação, onde cada bit é levado à árvore de codificação até que atinja um símbolo. Este, então, é substituído pela seqüência de bits. Este processo é repetido até que todos os bits da mensagem sejam lidos.

2.3.3 Codificação Aritmética

Apesar do algoritmo de Huffman chegar quase ao limite do teorema de Shannon, ele possui uma limitação devido à sua característica de trabalhar com números inteiros de bits. Assim, se a probabilidade de um determinado símbolo for superior a 50%, o número ótimo de bits será igual a 1. A codificação aritmética [WIT 87, PAR 99] busca trabalhar com "frações de bits" de modo que, caso um símbolo tenha, por exemplo, 90% de freqüência, seja codificado da maneira mais próxima da ótima conforme o teorema de Shannon, ou seja, ocupe o valor mais próximo possível de 0.15 bits.

A codificação aritmética trabalha com a seqüência de símbolos na íntegra, e não mais através da codificação de símbolo por símbolo como Huffman. A seqüência de símbolos utilizada como entrada é substituída por um número fracionário (um único número de ponto flutuante). Este número é sempre menor que um e maior ou igual à zero. Quanto maior e mais complexa for a mensagem, maior será o custo computacional empregado no processo.

O primeiro passo, após o cálculo da probabilidade dos símbolos, é atribuir a cada símbolo uma faixa de valores. A tabela 2.2 apresenta o exemplo da seqüência BACA. Estes valores são números fracionários e o total das faixas de todos os símbolos juntos deve ser igual à 1. As faixas são distribuídas de acordo com a probabilidade de cada símbolo. Símbolos mais freqüentes possuem faixas maiores, enquanto que os de menor freqüência recebem uma faixa menor.

A cada símbolo é atribuída uma parte dos números entre 0 e 1, de acordo com a sua probabilidade dentro da mensagem original. As faixas variam do menor valor

TABELA 2.2 – Tabela de Distribuição de Faixas para Codificação Aritmética

Símbolo	Probabilidade	Faixa
A	0.50	0.00 - 0.50
B	0.25	0.50 - 0.75
C	0.25	0.75 - 1.00

até o maior valor atribuído, não incluindo o maior valor absoluto, ou seja, o símbolo A, por exemplo, possui a faixa entre 0.00 e 0.49999...

O algoritmo determina a cada símbolo a faixa de valores à qual ele tem direito dentro da mensagem. Como B é o primeiro símbolo da mensagem, a faixa atribuída é 0.50 - 0.75. Como o segundo símbolo é A, a faixa é reduzida entre os valores 0% e 50%, ou seja, o novo valor da mensagem passa a ser 0.50 - 0.625. O processo se repete até que todos os símbolos sejam codificados. O menor valor da faixa final é o valor da mensagem original. Abaixo segue o algoritmo da codificação aritmética. A variável *range* contém o valor total da faixa no início de cada interação. As variáveis *low* e *high* armazenam os limites inferior e superior da faixa, respectivamente.

```
Set low to 0.0
Set high to 1.0
While there are still input symbols do
    get an input symbol
    range = high - low.
    high = low + range*high_range(symbol)
    low = low + range*low_range(symbol)
End of While
output low
```

A tabela 2.3 apresenta a execução completa do algoritmo sobre a seqüência BACA. O valor final da variável *low*, 0.59375, é o valor atribuído à seqüência.

TABELA 2.3 – Exemplo de Codificação Aritmética

Símbolo	Range	low	high
		0	1
b	1	0.5	0.75
a	0.25	0.5	0.625
c	0.125	0.59375	0.625
a	0.03125	0.59375	0.609375

A decodificação realiza o processo inverso. Cada símbolo é extraído do valor da seqüência através de uma análise do valor na tabela de distribuição de faixas. O símbolo é retirado do valor de maneira inversa à como foi inserido, subtraindo-se o menor valor do valor da seqüência e multiplicando pelo tamanho de sua faixa. O novo valor obtido é utilizado como entrada. O algoritmo continua até que todos os valores sejam decodificados. O algoritmo de decodificação é apresentado a seguir.

```
get encoded number
Do
    find symbol whose range straddles the encoded number
    output the symbol
    range = symbol low value - symbol high value
    subtract symbol low value from encoded number
    divide encoded number by range
until no more symbols
```

A tabela 2.4 apresenta a execução do algoritmo de decodificação sobre a seqüência BACA usando como entrada o valor 0.59375.

TABELA 2.4 – Exemplo de Decodificação Aritmética

Símbolo	Range	Number
b	0.25	0.59375
a	0.5	0.375
c	0.25	0.75
a	0.5	0

Em seqüências muito extensas, a implementação deste algoritmo se transforma num problema, pois existe a limitação do tamanho de um valor com ponto flutuante nos computadores atuais. A alternativa mais comum de implementação baseia-se na utilização de valores inteiros de 16 ou 32 bits para as faixas. Ao invés de atribuir faixas entre 0 e 1, são atribuídas faixas entre 0 e 65535. Portanto, no exemplo da tabela 2.2, A pode assumir a faixa entre 0 e 32768.

2.3.4 Modelos Adaptativos

No processo de codificação por entropia normal, o codificador precisa calcular as estatísticas de probabilidade de cada símbolo dentro da seqüência a ser reduzida antes de iniciar o processo de codificação em si. Durante o processo, estas estatísticas mantêm-se inalteradas. Ao final, as estatísticas são armazenadas junto com a mensagem codificada para que o processo de decodificação possa retornar com a mensagem original.

Os modelos adaptativos [GAL 78, VIT 87] permitem que os algoritmos de entropia modifiquem suas estatísticas durante o processo de codificação. A utilidade desta variante reside no fato de que, à medida em que a codificação for sendo realizada, novos símbolos serão gerados com a união dos bits dos símbolos já processados e, assim, melhorando-se o grau de eficiência da compressão dos dados. Outro diferencial encontrado é a não existência da necessidade de enviar as estatísticas ao decodificador, pois ele pode recriar a tabela de estatísticas do último estado do codificador e recuperar a mensagem através da inversão do processo.

Na compressão adaptativa dos dados, tanto o codificador como o decodificador iniciam com o modelo estatístico no mesmo estado. Ambos processam um símbolo por vez, atualizando as probabilidades à cada interação. Este processo tem um custo computacional superior, se comparado aos processos de codificação por entropia normais, visto que é necessário um recálculo das estatísticas. Como o processo

de decodificação também precisa computar as estatísticas, o custo da decodificação torna-se igual ao da codificação. O custo maior justifica-se, entretanto, pois a eficiência obtida na compressão dos dados é compensatória.

2.4 Transformadas

A freqüência espacial de uma imagem normalmente refere-se à taxa com que a intensidade de um pixel muda. Freqüências altas podem ser apresentadas pela concentração de variações nesta intensidade enquanto que freqüências baixas podem ser apresentadas por grandes áreas de valores constantes ou próximos.

O olho humano tem a característica de perceber mais distintamente as variações suaves de freqüencia, representadas pelos componentes de freqüências baixas do sinal da imagem [ROC 90].

Como uma imagem é formada, basicamente, por grandes regiões com a mesma intensidade ou pequenas variações, a retirada de algumas informações das freqüências altas permite a redução da quantidade de informação presente na imagem, através da diminuição dos detalhes, sem que o olho humano sinta muita diferença. O problema reside no fato que uma imagem é representada por um vetor bidimensional de coeficientes, adquirido durante o processo de digitalização. Este vetor está no domínio espacial, onde cada posição apresenta a informação espacial de intensidade da imagem. É preciso identificar neste vetor as freqüências altas e baixas. Para isso, é utilizado uma transformada.

Uma transformada não apresenta perda de dados. Os coeficientes resultantes de uma transformada são tão válidos quanto os coeficientes do vetor de entrada original. A transformada apenas permite isolar os componentes das freqüências altas dos componentes das freqüências baixas. Um processo de quantização pode então ser aplicado para a redução da precisão dos coeficientes das freqüências mais altas. O processo de quantização é apresentado na seção seguinte.

Dois modelos de transformada são os mais utilizados: Transformada Discreta do Cosseno (DCT) e Transformada Discreta de Wavelet (DWT). Ambos são apresentados nos itens a seguir.

2.4.1 Transformada Discreta do Cosseno

A transformada do cosseno [AHM 74, MIT 97, MPE 93] é utilizada na conversão de um sinal no domínio do tempo (ou espaço, quando trata-se de uma imagem), para o domínio freqüência. Além disso, esta transformada ainda condensa a energia do vetor de entrada, permitindo que seja representado por menos bits.

A tendência é apresentar a maior parte das características de freqüencia baixa do vetor de entrada nas primeiras posições do vetor de saída. O primeiro valor do vetor de saída é o que contém a média de todas as freqüências do vetor de entrada. Ele é o que condensa a maior parte da energia. Este valor é chamado de DC. Os demais valores do vetor de saída são chamados de AC.

Para se converter N valores de entrada para o domínio-freqüência do cosseno, é necessário utilizar N funções base de cosseno. Cada função base é composta por N pontos e u funções. A equação 2.5 é utilizada para a criação das funções base, onde u varia de 0 a $N-1$ e x é o ponto da função base u a ser digitalizado, e também varia de 0 a $N-1$. A figura 2.4 apresenta as funções base para $N = 8$. O valor 8 para

o número de pontos é o de maior emprego nas implementações da DCT[MIT 97]. Cada função base possui uma freqüência, variando da freqüência mais baixa para a freqüência mais alta. A DCT compara os valores de um vetor de entrada com cada uma destas funções, e gera como saída a importância que cada função possui em relação a este vetor.

$$\cos \frac{(2x+1)u\pi}{16} \tag{2.5}$$

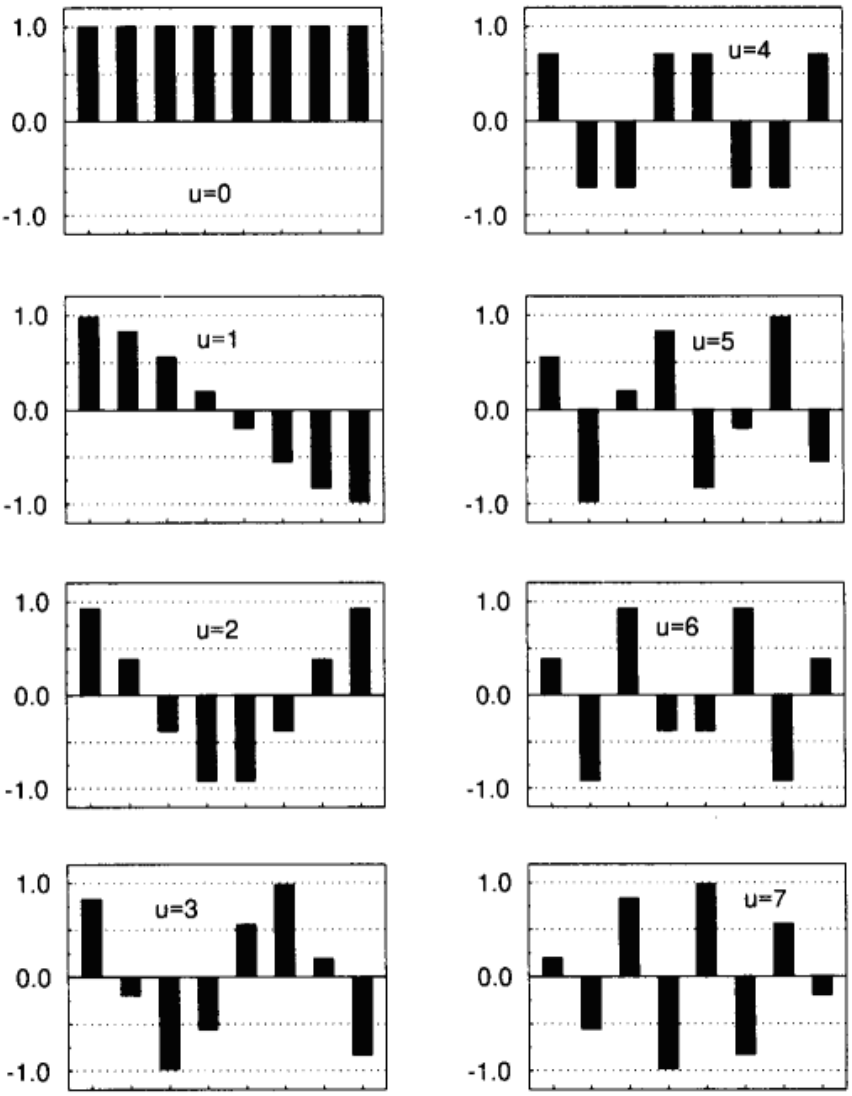

FIGURA 2.4 – Funções Base da DCT para $N = 8$

Dado um vetor de entrada, a DCT condensa as freqüências baixas nos primeiros coeficientes do vetor de saída, enquanto os últimos possuem as freqüências altas. Como uma imagem é composta, em sua grande maioria, por freqüências baixas, os primeiros valores no vetor da saída tendem a ser maiores e mais significativos. Os

valores do vetor de saída são os coeficientes da DCT. Estes coeficientes representam, também, o quanto cada função base deve ser escalada de modo a obter os N valores de entrada. A figura 2.5 mostra em (a) os valores de um vetor de entrada e em (b) os coeficientes DCT gerados. O coeficiente DC, o valor de freqüência média do vetor de entrada, é dado pelo primeiro coeficiente do vetor de saída.

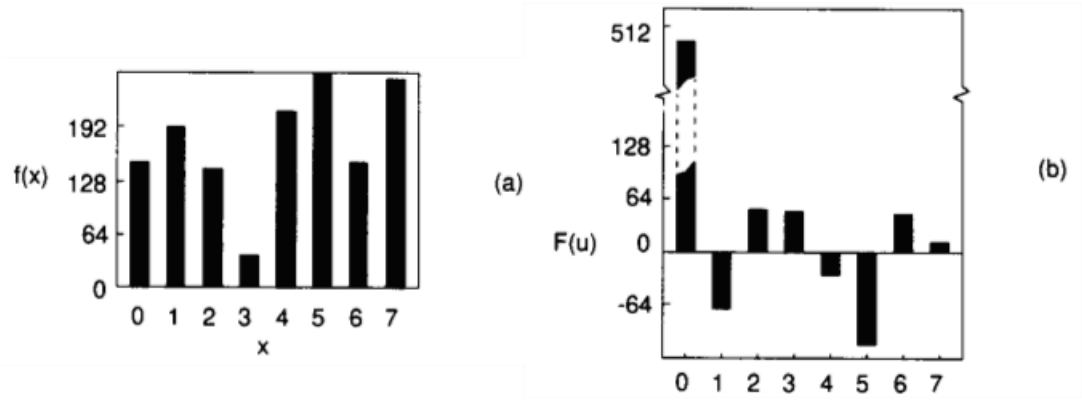

FIGURA 2.5 – Exemplo de aplicação da DCT

A equação 2.6 é utilizada para o cálculo dos coeficientes da DCT, onde $f(x)$ é o valor do vetor de entrada na posição x e $f(u)$ é o coeficiente DCT gerado para a posição u. A decomposição dos valores de entrada em coeficientes DCT é chamado de *Forward* DCT (FDCT). A reconstrução dos valores é alcançada através da função inversa da equação 2.6, que é encontrada na equação 2.7. O processo de reconstrução dos valores originais é chamado de *Inverse* DCT (IDCT).

$$f(u) = \frac{C(u)}{2} \sum_{x=0}^{7} f(x) \cos \frac{(2x+1)u\pi}{16} \tag{2.6}$$

$$\text{onde } C(u) = 1/\sqrt{2} \text{ para } u = 0$$
$$\text{e } C(u) = 1 \text{ para } u > 0$$

$$f(x) = \sum_{u=0}^{7} \frac{C(u)}{2} f(u) \cos \frac{(2x+1)u\pi}{16} \tag{2.7}$$

Quando se utiliza a DCT em uma imagem cujos valores estão armazenados em um vetor bidimensional, é comum aplicar a DCT nas duas dimensões. No caso, um vetor de 8x8 valores. O processo, chamado 2-D DCT, consiste em aplicar a equação da DCT aos vetores horizontais e, depois, aos verticais. Observa-se que a ordem de aplicação não influi no resultado, desde que seja aplicado em todos os vetores da mesma dimensão antes de ser aplicado na seguinte. O resultado é um vetor bidimensional com 64 coeficientes DCT, sendo que os coeficientes mais à esquerda e mais ao topo representam as freqüências baixas. As funções base também devem ter uma dimensão a mais, o que gera 64 funções. As equação 2.8 e 2.9 apresentam a 2D-FDCT e 2D-IDCT, respectivamente. A figura 2.6 mostra um exemplo de 2D-FDCT em uma matriz com $N = 4$.

$$f(u,v) = \frac{C(u)C(v)}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \tag{2.8}$$

$$f(x,y) = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{C(u)C(v)}{4} f(u,v) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \tag{2.9}$$

$$\begin{bmatrix} 22 & 24 & 27 & 28 \\ 23 & 24 & 28 & 29 \\ 24 & 25 & 29 & 30 \\ 25 & 27 & 30 & 32 \end{bmatrix} \rightarrow DCT \rightarrow \begin{bmatrix} 106.75 & -10.0604 & -0.25 & 8.8566 \\ -4.7887 & 0.4268 & -0.3266 & -0.1652 \\ 0.75 & -0.0560 & -0.25 & -0.6533 \\ 75193 & 0.5901 & -0.1353 & 0.4647 \end{bmatrix}$$

FIGURA 2.6 – Exemplo de 2D-FDCT

A compressão se dá no processo de quantização, que reduz a precisão dos coeficientes DCT e aumenta o número de valores iguais à zero. A matriz resultante é, então, fortemente compactada com o processo de entropia.

2.4.2 Transformada Discreta de Wavelet

Diferente do DCT, que utiliza intervalos fixos de tempo para o cálculo da freqüência do sinal, a Transformada Discreta de Wavelet (DWT) [RIO 91, USE 2001] permite representar a freqüência em intervalos variados.

Em um sinal, as freqüências altas possuem maior número de oscilações em menor intervalo de tempo, ou seja, podem ser melhor representadas através de uma escala com resolução maior no tempo. Já as freqüências baixas, que possuem menos oscilações através do tempo, são melhor representadas em uma escala com resolução maior em freqüência. A figura 2.7 apresenta um exemplo de sinal com tempo normalizado. Percebe-se que as freqüências altas possuem um intervalo de tempo curto, portanto, uma maior amostragem do tempo neste intervalo apresentará uma melhor representação destas freqüências.

A DWT consiste aplicar à um sinal diversos filtros para freqüências altas (*high-pass filters*) e baixas (*lowpass filters*). Estes filtros subdividem o sinal ao meio, de acordo com a sua escala de freqüências. Assim, se um sinal possui resolução máxima de 1000Hz, os filtros de freqüências altas extraem as freqüências no intervalo de 500-1000Hz, enquanto que os filtros de freqüências baixas extraem as freqüências no intervalo de 0-500Hz[1]. Este processo apenas reduz a freqüência do sinal, sendo que a escala permanece inalterada. No entanto, como agora a maior freqüência de cada intervalo é apenas a metade da maior freqüência anterior, é possível dividir também a escala pela metade. Por exemplo, se antes o sinal era representado por um vetor de 16 posições, a aplicação dos filtros resultará em 2 vetores de 16 posições. No entanto, é possível reduzir pela metade estes vetores, com pouca alteração no sinal,

[1] Foi utilizado HZ apenas para exemplificação. Em um sinal discreto, a unidade de freqüência é radianos.

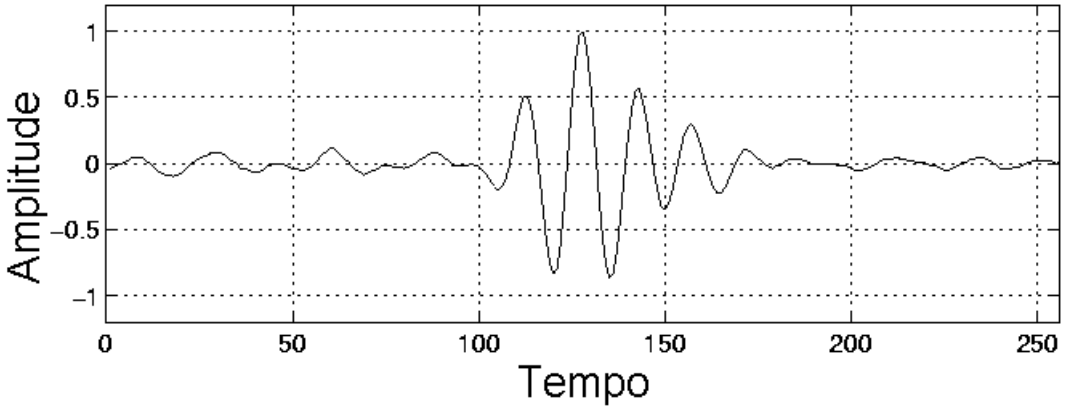

FIGURA 2.7 – Exemplo de Sinal no Tempo

realizando uma amostragem intercalada (posições 2, 4, 6, 8, ...). Isso resulta em 2 vetores de 8 posições. Uma DWT, portanto, é formada por aplicações de filtros para freqüências altas e baixas e por reduções na escala. O processo pode ser repetido diversas vezes em qualquer dos sinais resultantes. Costuma-se utilizar apenas uma repetição no sinal resultante das freqüências baixas no processo de compressão de imagens, pois as freqüências baixas são as mais importantes neste processo. A figura 2.8 apresenta o esquema para se computar a DWT com uma dimensão (1D-DWT), onde H é o filtro de freqüências baixas, G é o filtro de freqüências altas, S' é o sinal resultante de menor freqüência (no caso de uma imagem, a parte mais significativa da informação) e W_x são as freqüências altas de cada interação (no caso de uma imagem, o complemento da informação ou a parte menos significativa). A seta para baixo representa redução de escala com fator 2.

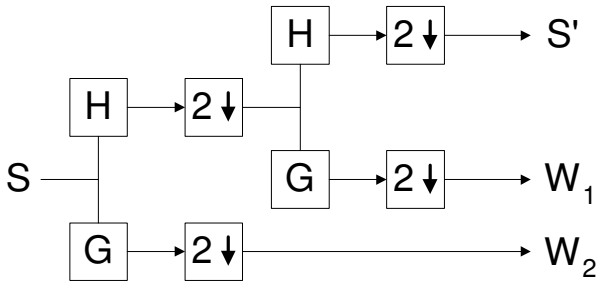

FIGURA 2.8 – Esquema do 1D-DWT

A figura 2.9 mostra a aplicação da DWT com 8 interações sobre o filtro H. O sinal de entrada é o mesmo apresentado na figura 2.7. Percebe-se que as faixas de freqüências que não possuem grande importância na informação original apresentam amplitudes próximas a zero, podendo ser descartadas sem grandes alterações. Neste exemplo, com 256 amostras no tempo, a primeira interação irá resultar em duas faixas de freqüências de 128 amostras cada. A última faixa (128-256), pertencente às freqüências altas, pode ser descartada sem muita alteração na amostragem inicial. Evidente que, em um sinal de imagem, a informação fica um pouco melhor

distribuída, mas o principio é o mesmo e o processo de quantização alcança grandes taxas de redução nas freqüências altas.

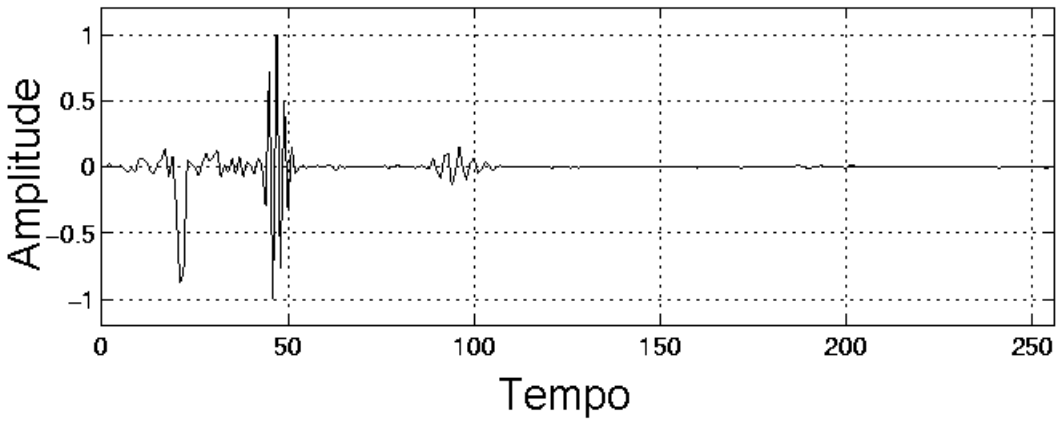

FIGURA 2.9 – Exemplo de DWT em um Sinal

A equação 2.10 apresenta a decomposição de um sinal (S) em freqüências baixas, enquanto a equação 2.11 trata das freqüências altas. L representa o tamanho do vetor do filtro utilizado (h para o filtro de freqüências baixas e g para freqüências altas) e n é o tamanho do sinal de entrada. As equações já estão computadas com a redução da escala por 2. A decomposição do sinal é também chamada de análise.

$$H(n) = \sum_{k=0}^{L-1} h(k)S(2n-k) \tag{2.10}$$

$$G(n) = \sum_{k=0}^{L-1} g(k)S(2n-k) \tag{2.11}$$

Existem vários estudos sobre filtros de wavelet [VET 92]. Cada filtro possui características distintas e é mais adequado para um determinado tipo de sinal que outro. A escolha do filtro certo para o sinal pode significar maior ou menor distribuição dos dados, afetando a compressão dos mesmos. No processo de compressão de imagens, costuma-se utilizar um mesmo filtro para toda a imagem. Os filtros mais conhecidos são as séries de Daubechies [DAU 88].

A recomposição do sinal é feita invertendo-se as equações e somando os coeficientes H e G. Este processo é também chamado de síntese e é apresentado na equação 2.12, à qual aplica-se a mesma legenda das equações imediatamente anteriores.

$$S(n) = \sum_{k} H(k)h(n-2k) + G(k)g(n-2k) \tag{2.12}$$

Em uma imagem, este processo deve ser aplicado nos dois eixos, horizontal e vertical. Ao aplicar a DWT nos vetores horizontais da imagem, os dois vetores resultantes (freqüências altas e baixas) de todos os vetores horizontais devem passar novamente pela DWT, agora na vertical. Este processo resulta em uma imagem como a apresentada na figura 2.10. O canto superior esquerdo (composto pelas freqüências baixas (LL)), contém a imagem original em metade da escala, já as demais regiões contêm as freqüências altas horizontais (HL), verticais (LH) e diagonais (HH).

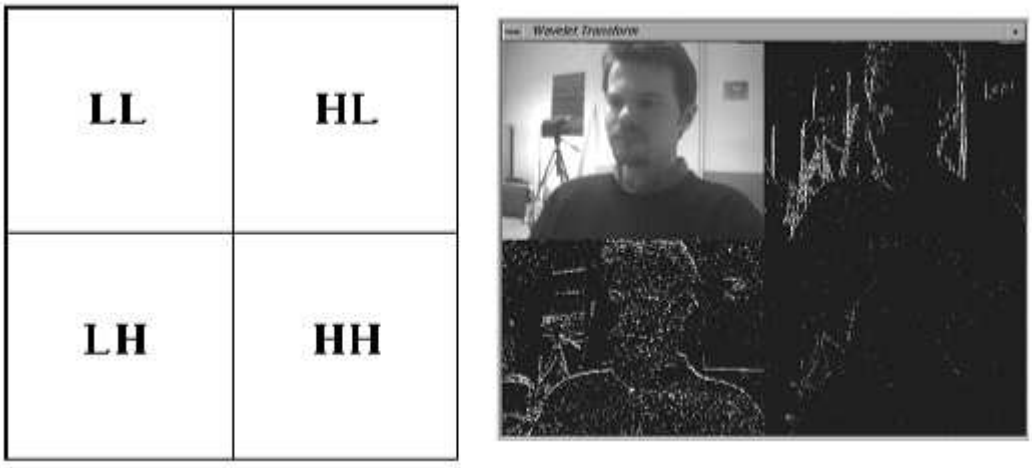

FIGURA 2.10 – Exemplo de 2D-DWT

2.5 Quantização

Um quantizador pode ser empregado para reduzir o número de bits necessários para armazenar os coeficientes resultantes das transformadas. Esta redução ocorre através de um processo de aproximação, onde um ou mais valores são substituídos por índices dentro de uma escala finita. Como este processo é realizado por aproximação, ele resulta em perda na qualidade da imagem. Esta perda depende da escala utilizada para gerar os índices e da transformada empregada. O processo de quantização é o responsável pela maior parte da perda na qualidade final de uma imagem e também pela maior parte da compressão dos dados.

Existem vários modelos de quantização [GRA 98, COS 93]. Basicamente, estes modelos podem ser divididos em duas categorias: escalares (SQ) e vetoriais (VQ). Quantizadores escalares tratam cada valor de entrada de maneira independente, gerando um valor de saída de menor precisão. Quantizadores vetoriais trabalham com uma seqüência de valores (um vetor) para gerar uma saída discreta.

A tabela de índices é otimizada quando o índice gerado é o que possui a menor distorção em comparação aos valores originais. Isso se obtém aproximando ao máximo o índice do centro dos valores. Esta condição é chamada de centróide (*centroid condition*). Um quantizador é otimizado quando é capaz de aproximar ao máximo um valor de entrada do índice de menor distorção (índice mais próximo). Esta condição é chamada *nearest neighbour condition*.

2.5.1 Escalares

O modo como são implementados os algoritmos de quantização também varia de acordo com o codec. Quantizadores em geral, por definição, trabalham com substituição de valores. Quando são utilizados quantizadores escalares [GRA 98], cada valor é substituído por um índice dentro de uma escala de aproximação. A figura 2.11 apresenta a subdivisão das escalas. Uma escala de índices é montada segmentando-se o valor máximo da entrada em faixas de tamanho r. Um índice n é retornado caso o valor de entrada pertença ao conjunto $[nr, (n+1)r)$.

Por exemplo, se os valores de entrada estiverem entre 0 e 256 (8 bits), uma

escala de aproximação pode ser gerada de modo que cada segmento tenha quatro valores ($r = 4$). Se for utilizado o valor 3 de entrada, ele será substituído pelo índice 1, pois pertence ao primeiro segmento. O valor 123 pertence ao trigésimo segmento, portanto seu índice será o número 30. Segundo este modelo, os índices possuem um limite de 64 valores, assim, apenas 6 bits são necessários para gerar a saída. Uma redução de 25%.

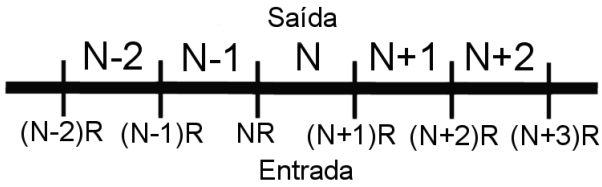

FIGURA 2.11 – Faixa de Cálculo para a Quantização Escalar

Um quantizador é uniforme (*Uniform Quantizer*) quando sua faixa de valores tem tamanho constante (r não varia). Se o tamanho da faixa variar, então o quantizador é não uniforme (*Non-uniform Quantizer*). Codificador e decodificador devem trabalhar com os mesmos valores de r para cada faixa.

2.5.2 Vetoriais

Quantizadores vetoriais [GRA 84, COS 93, NAS 88] baseiam-se numa base de dicionários, semelhante ao princípio empregado na compressão de textos. A figura 2.12 apresenta um esquema de quantização vetorial. Um vetor de entrada é comparado com uma base de padrões (*codebook*) previamente gerada. O melhor padrão (*codeword*) — aquele que apresentar a menor distorção dos valores de entrada — é escolhido, e apenas o índice deste padrão é transmitido ao decodificador, que deve dispor da mesma base de padrões que o codificador. O processo de decodificação consiste simplesmente na substituição do índice pelo vetor de valores da base. Estes são os valores de saída.

Utilizando vetores de entrada de 16 bytes e uma base com 256 padrões, apenas 1 byte (8 bits) é necessário para representar o índice. Uma redução de 16 vezes na quantidade de dados.

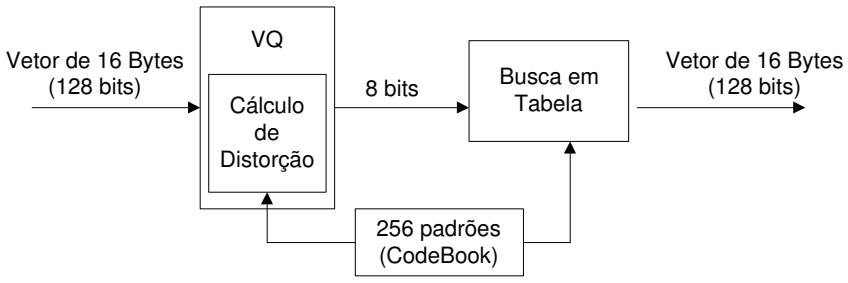

FIGURA 2.12 – Esquema de Quantização Vetorial

Para que um quantizador vetorial trabalhe de maneira otimizada, é preciso que a base de padrões apresente a menor distorção possível em uma variedade de

vetores de entrada (*centroid condition*) e um algoritmo de busca capaz de localizar dentro da base de padrões, qual aquele que apresenta menor distorção para uma determinada entrada (*nearest neighbour condition*).

A base de padrões é computada através do agrupamento de uma série de vetores de entrada. Os vetores de entrada são agrupados conforme sua similaridade, que pode ser calculada pela menor distância euclidiana entre eles, definida na equação 2.13, onde k é a dimensão dos vetores, x_j é o j componente do vetor de entrada e y_{ij} é o j componente do vetor y_i. Cada grupo é chamado de "região de Voronoi", conforme mostra a figura 2.13. É computado um vetor central (centróide) para cada grupo. Este vetor é uma média de todos os vetores pertencentes ao grupo. Os centróides formam a base de padrões.

$$d(x, y_i) = \sqrt{\sum_{j=1}^{k}(x_j - y_{ij})^2} \tag{2.13}$$

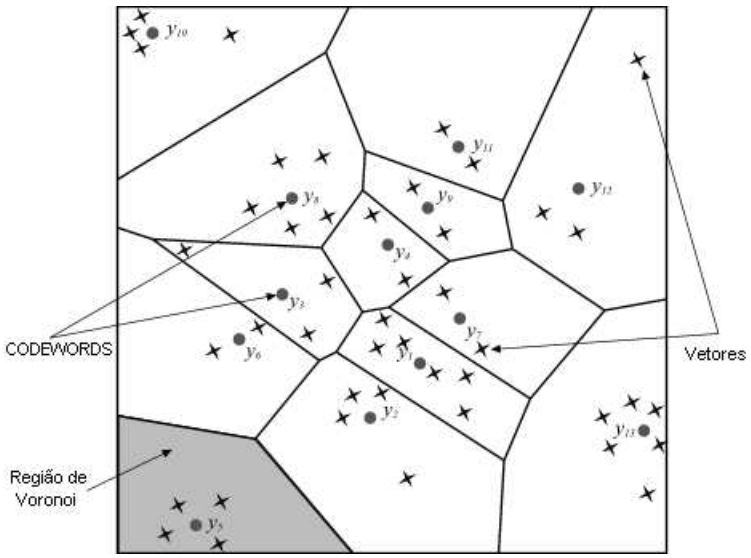

FIGURA 2.13 – Região de Voronoi

Uma maneira de realizar o cálculo da base de padrões é através do algoritmo de Linde-Buzo-Gray (LBG) [LIN 80]. Ele apresenta, como primeiro passo, a definição do tamanho da base (N). Quanto maior a base, menos distorções nos valores de saída mas maior a quantidade de dados na saída. A seguir, seleciona-se aleatoriamente N vetores de entradas que são utilizados como base inicial. Usando a medida de distância euclidiana, agrupam-se os vetores ao redor da base de padrões e calcula-se o vetor médio de cada grupo para ser utilizado como base. Este processo se repete até alcançar um ponto ótimo, com pouca ou nenhuma variação nos valores dos vetores de padrões.

O desenvolvimento de uma base de padrões que melhor represente uma série de vetores de entrada é um processo NP-completo. O custo para montar uma base de padrões é elevado, pois o processamento necessário para localizar a melhor base aumenta exponencialmente, de acordo com o tamanho da base. Várias pesquisas buscam a redução deste custo [ARY 93].

A base de padrões costuma ser montada com uma pequena base de vetores de entrada (vetores de treinamento). O problema encontra-se em como indexar os demais vetores. Uma maneira é a realização de uma pesquisa completa na base de padrões para cada vetor em busca do que possui a menor distorção. Este processo também tem um custo elevado. Uma alternativa é a busca binária.

2.6 Codificação Temporal

As técnicas apresentadas nas seções anteriores buscam atingir compressão de dados explorando a redundância e a correlação existente entre os pixels dentro de uma mesma imagem. A codificação temporal dos dados explora a redundância existente entre imagens consecutivas em um vídeo [MIT 97, SEZ 93].

Como dito anteriormente, um vídeo é formado por várias imagens apresentadas em seqüência. A freqüência de apresentação destas imagens varia de acordo com o padrão empregado e a forma como os dados foram inicialmente digitalizados. Cada imagem é chamada de quadro (*frame*) e a freqüência é informada em quadros por segundo. O padrão NTSC, por exemplo, utiliza a freqüência de 30 quadros por segundo, ou seja, 30 imagens devem ser apresentadas seqüencialmente, no período de um segundo.

A redundância pode ser explorada através da remoção de informações repetidas entre diversos quadros consecutivos da apresentação. Se a apresentação tratar-se de um telejornal ou uma vídeo conferência, por exemplo, poucas regiões serão alteradas de um quadro para outro. Na melhor das hipóteses, é possível que apenas a boca do apresentador apresente variação. As regiões repetidas entre os quadros não precisam ser codificadas, pois podem ser utilizadas as mesmas regiões do quadro anterior.

Assim como as demais técnicas, a codificação temporal trabalha com a noção de que o olho humano não percebe os erros existentes quando se utilizam freqüências altas para a apresentação [ROC 90, SEZ 93]. Portanto, é possível reduzir a qualidade de alguns quadros sem que sinta-se grandes diferenças na visualização do vídeo em geral.

São duas as técnicas mais utilizadas para codificação temporal dos dados: a compensação dos movimentos e as transformadas em 3D. A primeira técnica consiste em buscar a compensação de regiões alteradas de um quadro com informações existentes em quadros passados ou futuros. As transformadas em 3D empregam o mesmo princípio das transformadas vistas no capítulo anterior, no entanto, operam temporalmente.

2.6.1 Compensação de Movimento

Quadros sucessivos de um vídeo podem conter o mesmo objeto, parado ou em movimento. Se ele estiver parado, pode-se atingir compressão nos dados codificando-se uma única vez o objeto. Caso o objeto em questão esteja em movimento, vários quadros são alterados, dependendo de sua trajetória. Um dos processos mais utilizados para o tratamento destes casos é a compensação de movimento [PAR 99, MIT 97, MPE 93, SEZ 93]. Este processo é dividido em três módulos: detecção, estimativa e compensação.

Na figura 2.14, tem-se o diagrama do esquema de compensação de movimento. O processo de detecção de movimento busca localizar, dentro de uma seqüência de

quadros, quais as regiões que sofreram alterações e, então, o tratamento é realizado sobre estas regiões. A estimativa de movimento examina o movimento dos objetos dentro da seqüência de imagens na tentativa de obter vetores que representem o movimento em questão. A compensação do movimento usa estes vetores para atingir uma maior taxa de compressão nos dados.

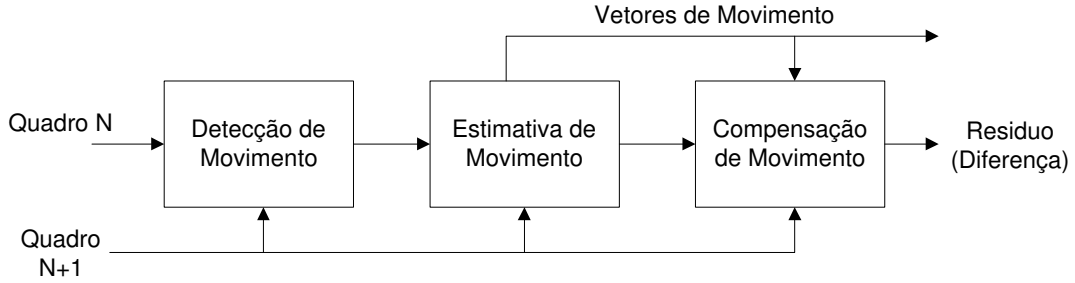

FIGURA 2.14 – Diagrama de Compensação de Movimento

Os algoritmos de compensação de movimento mais utilizados são baseados em blocos. Nesta situação, um quadro é dividido em blocos de medidas N por M, sendo que normalmente $N = M$, e todos os pixels deste bloco possuem, por definição, o mesmo movimento. Cada um destes blocos é comparado com os blocos do quadro anterior. Os blocos que exibirem a menor margem de erro e a menor distância entre si são considerados iguais e apenas um vetor, representando a distância entre eles, é codificado. O quadro é reconstituído tal como seria visto pelo decodificador, ou seja, os blocos são dispostos de acordo com seus respectivos vetores. Este quadro reconstruído é comparado com o quadro original e as diferenças são subtraídas: todo o quadro é representado por seus vetores e por esta diferença. Quanto mais apurada a estimativa dos vetores, menor será a diferença entre os quadros e menor a taxa de dados codificados.

O algoritmo mais conhecido, devido à sua simplicidade, é o da busca exaustiva (*full search*)[PAR 99]. Neste algoritmo, define-se uma região de busca para cada bloco de X blocos na horizontal e Y blocos na vertical. O primeiro passo é verificar a existência de alguma alteração neste bloco, entre um quadro e outro. Se há mudança, o bloco é comparado com seus vizinhos dentro da região de busca, de modo a localizar a estimativa de movimento do bloco. O grupo que possuir a menor diferença é localizado e um vetor é traçado entre o bloco original e o bloco localizado. Várias regras podem ser utilizadas para determinar qual o bloco que possui a menor diferença com relação ao original. O melhor bloco pode ser aquele que possui o menor erro médio (MSE, encontrado através soma das diferenças entre os pontos de cada bloco, dividido pelo número de pontos), de acordo com um mínimo aceitável. Este algoritmo varia muito entre implementações. No caso mais comum, para obter uma localização mais precisa, a varredura é realizada em pixels ao invés de blocos e o vetor é gerado de acordo com o deslocamento dos pixels.

A figura 2.15 demonstra como uma varredura em pixels localiza o vetor. A região em destaque do quadro anterior é verificada e uma busca ocorre no quadro atual em busca de uma região semelhante. Ao localizar a região que mais se aproxima da original, é traçado um vetor de movimentação. A região que mais se aproxima da original deve ficar dentro de um limiar definido por um cálculo de erro. O cálculo

de erro mais comum é o da diferença média absoluta (MAD)[MIT 97], dado pela equação 2.14, onde F é a matriz de pontos do bloco atual e G é a matriz do bloco anterior. O vetor é dado por (dx,dy).

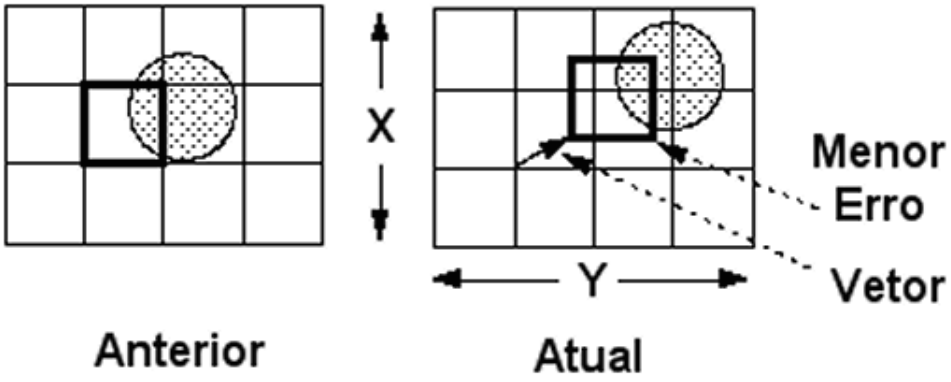

FIGURA 2.15 – Vetor de Movimento

$$MAD(dx, dy) = \frac{1}{NM} \sum_{i=-N/2}^{N/2} \sum_{j=-M/2}^{M/2} |F(i,j) - G(i+dx, j+dx)| \qquad (2.14)$$

A codificação por compensação de movimento tem um custo computacional muito alto. O tamanho de cada bloco e o de cada região de busca influenciam no processo, mas ainda assim, a compensação de movimento pode representar mais de 50% do processamento de codificação de um vídeo. Vários algoritmos que buscam acelerar a estimativa de movimento foram propostos [CHA 90, RIB 97], bem como algoritmos que buscam trabalhar com estimativas em pixels ao invés de blocos [MPE 93].

O custo computacional empregado pelo decodificador é muito inferior do codificador, visto que basta apenas reposicionar os blocos do quadro anterior de acordo com os vetores e aplicar a diferença. A maior parte dos algoritmos utiliza estimativa de movimento apenas no sinal da luminância (sinal Y) para reduzir o custo da computação.

2.6.2 Transformadas 3D

Assim como as transformadas isolam as freqüências altas nas imagem em duas dimensões, é possível se detectar as regiões que sofrerem alterações aplicando a transformada na dimensão do tempo. Uma alteração é representada pela mudança brusca na intensidade de um pixel de um quadro para outro. Por exemplo, uma cena que apresenta uma bola caindo mostra a bola em movimento sobre um fundo estático. A cada quadro a bola será apresentada em uma posição mais baixa do que a disposta no quadro anterior. Parte de região onde a bola foi apresentada num quadro anterior será substituída pelo fundo, enquanto que parte do fundo será substituído pela bola. Essa variação nos pixels entre os quadros durante a trajetória da bola produz o mesmo efeito que as variações da intensidade dos pixels em uma imagem.

Portanto, ao aplicar-se uma transformada na dimensão do tempo, é possível isolar as alterações, representadas pelas freqüências altas.

A codificação temporal realizada através das transformadas 3D é feita simultaneamente à codificação espacial dos quadros [SER 97, FUR 99]. A partir de uma seqüência de quadros utilizada como entrada, a transformada é aplicada à toda a seqüência e uma saída é gerada, com o mesmo tamanho da entrada, porém, com a distinção espacial e temporal das freqüências.

No caso da DCT, é comum utilizar oito quadros devido à característica da DCT de trabalhar com oito valores de entrada. Cada quadro é dividido em regiões de 8x8, formando um cubo que é usado como entrada para o cálculo da DCT. A redução da qualidade é dada no processo de quantização. As equações 2.15 e 2.16 apresentam a 3D-FDCT e a 3D-IDCT, respectivamente.

$$f(u,v,w) = \frac{C(u)C(v)C(w)}{8} \sum_{x=0}^{7} \sum_{y=0}^{7} \sum_{z=0}^{7} f(x,y,z) D(x,u) D(y,v) D(z,w) \qquad (2.15)$$

$$\text{onde } C(u) = 1/\sqrt{2} \text{ para } u = 0$$
$$\text{e } C(u) = 1 \text{ para } u > 0 \text{ e } D(x,u) = \cos \tfrac{(2x+1)u\pi}{16}$$

$$f(x,y,z) = \sum_{u=0}^{7} \sum_{v=0}^{7} \sum_{w=0}^{7} \frac{C(u)C(v)C(w)}{8} f(x,y,z) D(x,u) D(y,v) D(z,w) \qquad (2.16)$$

Este processo gera uma imagem em cubo de tamanho igual à imagem original de entrada. O coeficiente localizado mais acima, à esquerda e à frente dos blocos, o coeficiente DC, armazena a maior parte da força do sinal original de todo o bloco. As variações espaciais tendem a se localizar mais acima e à esquerda, enquanto a variação temporal se concentra nos blocos mais à frente e variam sua posição de acordo com a localização do movimento nos quadros.

Um processo semelhante pode ser empregado na DWT [TAU 94]. Todos os quadros da seqüência devem sofrer a aplicação da DWT em suas duas dimensões espaciais (horizontal e vertical). Os quadros resultantes passam novamente pela DWT, agora sofrendo sua ação na dimensão temporal.

O custo computacional para codificar uma transformada em 3D é inferior ao custo necessário para realizar a codificação temporal com o método de compensação de movimento. A taxa de compressão atingida geralmente é maior, dependendo da quantização empregada e da quantidade de movimento existente no trecho de vídeo codificado, no entanto, o custo computacional para decodificar um vídeo é o mesmo necessário para codificá-lo.

3 Codificação Escalável

Nas transmissões de vídeo em ambientes heterogêneos (com usuários utilizando diferentes recursos de hardware e/ou conectados entre sí com diferentes capacidades de banda), é usual a ocorrência de perda de dados na rede devido à latência na chegada dos pacotes e de congestionamento, entre outros fatores [WU 2001, CON 2001]. Em mídias temporais como vídeo, estes fatores acarretam degradação da qualidade de sua apresentação. Um fator importante em um codec é a sua capacidade de adaptação à estes fatores, degradando pontos específicos de um vídeo de modo a não prejudicar sua compreensão, por exemplo. A transmissão de uma mídia temporal em tempo real exige a possibilidade de adaptação da mesma aos diferentes receptores e às suas diferentes características de equipamento. Uma solução é a transmissão em taxas de bits diferentes, permitindo que o usuário receba a quantidade máxima permitida de dados, evitando o congestionamento na rede.

Algoritmos para compressão de um sinal de vídeo de modo escalável são uma característica necessária para a distribuição de um sinal de vídeo numa rede de computadores. Uma técnica de compressão é escalável quando oferece variedade de taxas de decodificação usando o mesmo algoritmo base [WU 2001]. Estas taxas devem ser progressivas e complementares. Cada taxa representa uma camada de transmissão. A camada mais baixa, chamada camada base, deve conter as informações básicas para a decodificação de um vídeo na menor taxa de bits possível, conseqüentemente, na menor qualidade. As demais camadas contêm informações que aumentam a qualidade do vídeo apresentado quando agregadas à camada base.

A degradação da qualidade pode ocorrer através da redução da resolução do vídeo, aumento no fator de quantização, redução da freqüência de quadros por segundo, entre outras técnicas. Estas técnicas podem agregar-se de modo a aumentar o desempenho do codificador quanto à quantidade de camadas geradas. Por exemplo, é possível utilizar a técnica de redução da resolução em conjunto com a técnica de aumento nos fatores de quantização. Isso permite a geração de mais camadas, algumas agregando informação de resolução espacial, outras aumentando a qualidade final da imagem. A existência de camadas dependentes apenas da camada base é uma possibilidade. As camadas adicionais não precisam ser complementares umas às outras, necessariamente. O padrão MPEG 2 define um perfil de codificação onde é possível aumentar a freqüência de quadros por segundo da camada base e/ou aumentar a resolução, dependendo da preferência do usuário.

A utilização de um codificador escalável permite a adaptação da mídia aos recursos de rede do usuário. Esta adaptação pode ser feita manualmente [CON 2001], a partir da seleção do número máximo de camadas pelo usuário, ou automaticamente, através de um algoritmo de controle de congestionamento [MCC 96] tais quais os apresentados no próximo capítulo, por exemplo. Como as camadas são complementares, a quantidade de dados presente na soma de todos os fluxos que o usuário requisitou é semelhante à taxa de bits máxima presente na codificação do mesmo sinal de vídeo na mesma qualidade por um codificador normal (não escalar) que empregue a mesma técnica de compressão de dados.

Algumas técnicas de codificação escalar também podem ser utilizadas para adaptação aos recursos de hardware presentes na máquina receptora [VER 2001]. Por exemplo, a técnica de escalabilidade espacial codifica um vídeo em diferentes

taxas de bits através da variação da resolução de seus quadros. Dispositivos que possuem limitação quanto à resolução podem fazer a requisição apenas da máxima qualidade possível para apresentação, independente de possuírem sobra de recursos de rede.

Vários esquemas de codificação de vídeo escalável já foram propostos e atualmente fazem parte de normas como o H263, MPEG 2 e MPEG 4. Em geral, estes esquemas costumam utilizar apenas duas camadas (duas taxas de bits distintas), uma em baixa qualidade e outra em alta qualidade. Dos cinco padrões apresentados a seguir, SNR, Espacial, Temporal e Particionamento estão presentes na norma MPEG 2 como recursos adicionais a um codificador, enquanto o último — escalabilidade granular fina — é parte do Adendo IV da norma MPEG 4.

3.1 Escalabilidade SNR

A escalabilidade pela relação sinal-ruído (SNR — *Signal-Noise Ratio*) [WIL 97, FOG 2002, MPE 94] é um método que trabalha no domínio espacial, codificando a imagem na mesma resolução de entrada, porém com diferentes qualidades, através da variação dos quantizadores.

Este modelo de codificação opera de maneira similar à uma codificação normal, exceto por possuir uma etapa de quantização adicional. O codificador quantiza os coeficientes do DCT com um grau de precisão que varia conforme a qualidade que se deseje atingir na camada base, reduz os dados em uma etapa de entropia e os transmite. A distorção introduzida pelo processo de quantização é novamente quantizada, com uma precisão superior, sendo logo após reduzida e transmitida. Estes dados formam uma camada adicional de qualidade. Informações necessárias para a decodificação, como os vetores de movimento, são transmitidas apenas na camada base.

A camada base pode ser decodificada pelo mesmo processo operacional de um decodificador não escalar. Para decodificar a camada base combinada com a camada adicional, ambas devem ter sido recebidas pelo decodificador. Os coeficientes da camada adicional são inversamente quantizados. Utiliza-se a mesma tabela de quantização anteriormente usada para reduzir sua precisão. Por fim, somam-se os coeficientes da camada base, já dequantizados. Os coeficientes resultantes são decodificados da mesma maneira que um decodificador normal.

O esquema da figura 3.1 mostra um codificador SNR simples. A imagem original passa por uma transformada (representada no esquema pela DCT), é quantizada na menor qualidade desejável e reduzida pela entropia (VLC - *Variable Length Coding*, este processo consiste no emprego do RLE mais uma entropia tipo Huffman ou Aritmética). Este é o processo que dá origem à camada base. Então, a imagem retornada pelo quantizador da camada base é inversamente quantizada, formando a imagem que o decodificador obtém apenas com a camada base. O sinal resultante da subtração desta imagem com a imagem original é novamente quantizada, agora com um quantizador de melhor qualidade. Os dados resultantes deste processo compõe a camada adicional. A compensação de movimento é feita com base na imagem formada pela soma dos dados de todas as camadas, e os vetores de movimento são transmitidos junto na camada base. O restante do processo de compensação de movimento segue o mesmo modelo básico apresentado no capítulo anterior, onde o

quadro atual é também utilizado para prever quadros futuros e apenas o sinal da subtração com o quadro futuro é transmitido.

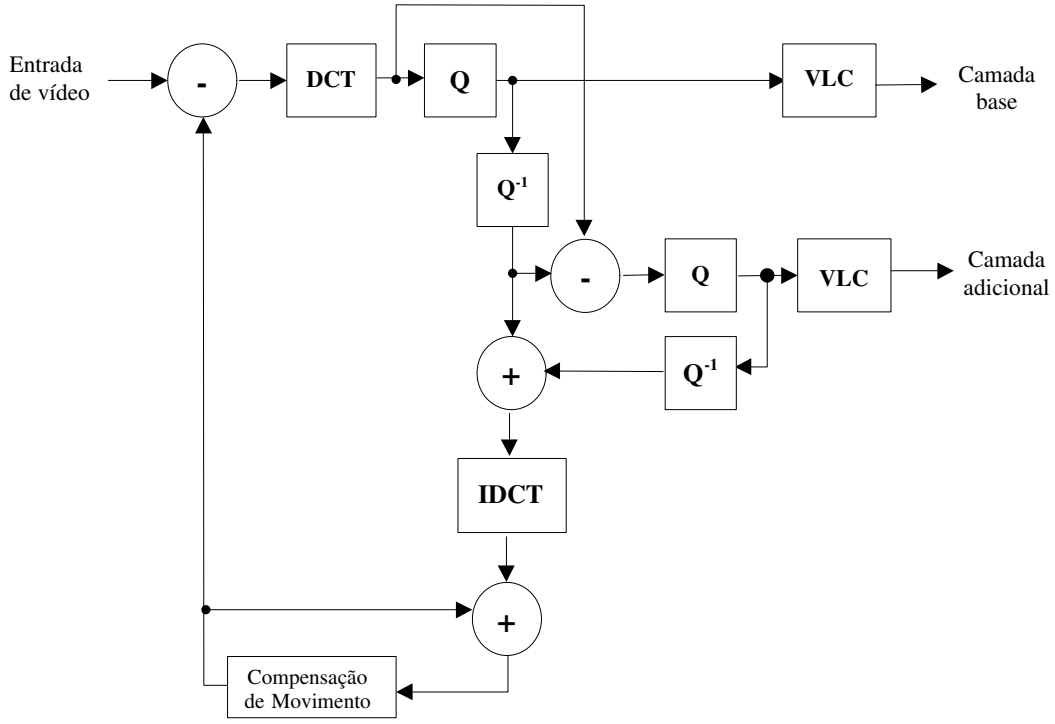

FIGURA 3.1 – Codificador Escalar SNR

A decodificação apresentada no esquema da figura 3.2, é o processo inverso do modelo de codificação. Os dados da camada base passam pelo processo inverso da entropia, quantização e transformada. São adicionados os dados presentes na compensação de movimento para formar a imagem básica. Se existirem dados de uma camada adicional, estes são adicionados aos dados da camada base, depois destes serem inversamente quantizados.

O padrão MPEG 2 [MPE 94] apresenta um perfil de codificação escalar SNR. Ele permite apenas duas camadas, a camada base e mais uma de refinamento. A camada base deve ser possível de ser decodificada por qualquer dispositivo capaz de decodificar um sinal MPEG 2, independente de possuir recurso de escalabilidade. Isto permite interoperabilidade com os dispositivos já existentes. A tabela 3.1 apresenta este perfil em maiores detalhes. Existem duas classes para o perfil escalar: Normal (*Main*) e baixa (*Low*). A classe baixa permite a transmissão e/ou armazenamento do vídeo em uma taxa de bits menor, seguindo o formato CIF.

3.2 Escalabilidade Espacial

A escalabilidade espacial [PUR 93, MPE 94] é caracterizada pelo uso de imagens de resolução mais baixa nas camadas inferiores, aumentando a resolução à medida que aumenta o número de camadas decodificadas. Se uma camada superior contém uma imagem em uma resolução alta, a camada inferior deve conter a mesma imagem com resolução reduzida através de algum algoritmo de redução escalar.

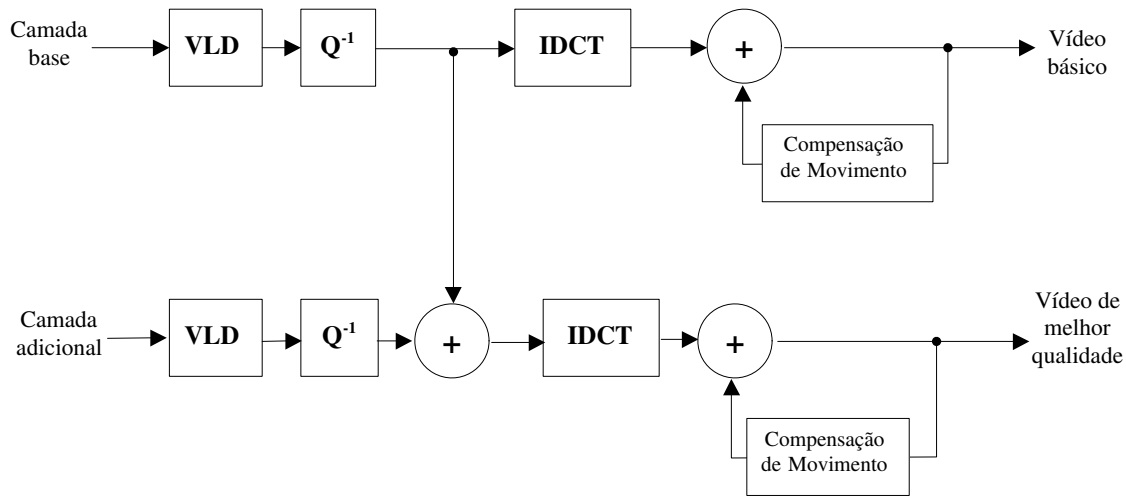

FIGURA 3.2 – Decodificador Escalar SNR

TABELA 3.1 – Tabela do perfil SNRProfile do MPEG 2

Formato de cor	4:2:0
Num. Max. de camadas	2 (Base + 1)
Max. Bit Rate @*Main*	Camada adicional: 15 Mbps Camada base: 10 Mbps
Resolução @*Main*	720x576x30
Max. Bit Rate @*Low*	Camada adicional: 4 Mbps Camada base: 3 Mbps
Resolução @*Low*	352x288x30

Os algoritmos de redução escalar mais empregados são o algoritmo de pirâmide laplaciana e as transformadas de wavelet. Em uma pirâmide laplaciana [BUR 83], a imagem é reduzida através de uma simples média aritmética de seus pixels. Por exemplo, uma região de quatro pixels é reduzida a um único pixel somando os quatro valores e dividindo por quatro. A parte inteira do valor resultante será o valor representante desta região. Para se retornar à imagem original, é preciso armazenar o resíduo, caracterizado pela subtração deste valor resultante pelos valores originais. Este resíduo necessita, normalmente, de uma menor quantidade de bits para ser representado do que a informação original. Quando usado para escalabilidade espacial de um vídeo, o valor resultante é transmitido na camada base, enquanto o resíduo é transmitido na camada adicional. A figura 3.3 apresenta uma pirâmide laplaciana de três níveis. O nível mais baixo forma a camada base, enquanto os resíduos, representados na pirâmide laplaciana pelos níveis superiores, formam as duas camadas adicionais.

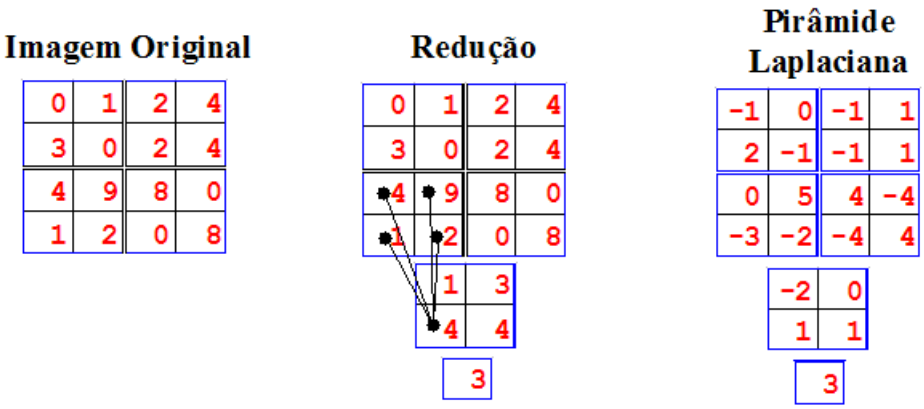

FIGURA 3.3 – Redução Espacial baseada em Piramide Laplace

No algoritmo de wavelet [SHE 99], a imagem é submetida à transformada de wavelet, que a separa em freqüências altas e baixas, conforme mostra a figura 3.4. A região das freqüências baixas, localizada no canto superior esquerdo, forma a camada base, e as demais freqüências formam a camada adicional.

Para se codificar um vídeo, dois laços operam com diferentes resoluções de imagem para produzir uma camada base e camadas adicionais. A camada base produz uma saída compatível com um decodificador não escalável. A camada adicional possui um complemento da camada base com resolução superior. Este complemento é a diferença existente entre a imagem da camada base, ampliada até a resolução da camada atual, e a imagem original na resolução da camada atual. Em determinados casos, é necessário um recálculo dos vetores de movimento. Em situações em que a imagem é composta por um único tom, toda preta ou toda branca por exemplo, não há variação entre as camadas pois a imagem permanece inalterada em qualquer resolução. Quando a imagem possui um ganho considerável entre as camadas, um recálculo dos vetores de movimento pode trazer maior compressão dos dados e uma maior qualidade final. Esta decisão é tomada através do cálculo de uma função W que verifica se as alterações entre as camadas passaram de um certo limiar para compensar o recálculo do movimento. A definição desta função W depende da im-

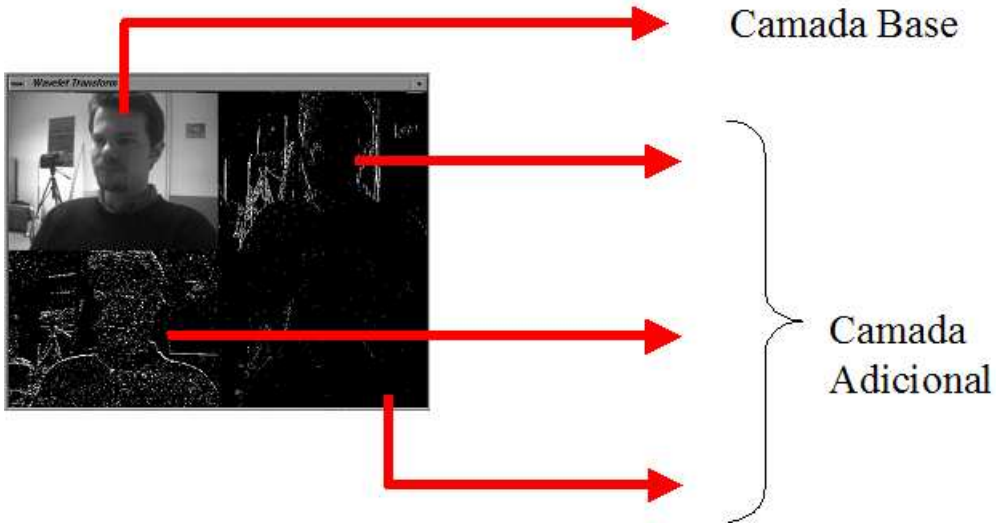

FIGURA 3.4 – Redução Espacial baseada em Wavelet

plementação. Ela pode variar desde uma simples média entre os pontos até um cálculo mais complexo da relação sinal-ruído.

Assim como a escalabilidade SNR, a camada base pode ser decodificada por um decodificador não escalar. Em uma combinação da camada base com camadas adicionais, a camada base é decodificada primeiro e ampliada até a resolução da camada adicional. A esta é somada a informação contida na camada adicional. Ambas as imagens são comparadas utilizando a mesma função W para verificar se estão entre um certo limiar e definir como utilizar os vetores de movimento.

A escalabilidade espacial pode ser empregada na transmissão de dados em camadas, para a adaptabilidade quanto à banda disponível no usuário, assim como para adaptabilidade de dispositivos com resoluções diferentes, como uma transmissão simultânea para PDAs e para computadores de mesa [VER 2001].

O esquema apresentado a seguir, na figura 3.5, trata-se de um típico codificador escalar espacial. Uma cópia do quadro atual é reduzida de tamanho e sofre um processo de compressão que é composto por uma transformada, pela quantização e pela entropia. Este resultado é uma parte da camada base. Em seguida, a imagem é descompactada através do processo inverso e é ampliada para sua resolução original. A imagem ampliada é transferida para a função W, que realiza uma comparação com a imagem original. Se a diferença encontrar-se acima de um certo limite, a diferença entre as duas imagem é compactada e é criada uma camada adicional. O cálculo de compensação de movimento é feito em ambos os casos. A função W irá informar quando as camadas adicionais precisam, ou não, conter informações de compensação de movimento adicionais.

A decodificação escalar espacial, apresentada na figura 3.6, realiza a descompressão da imagem na camada base, amplia sua escala e transfere a imagem para a camada adicional. A distorção presente na camada adicional é decodificada e adicionada à imagem vinda da camada base. Informações adicionais ligadas à compensação de movimento serão então utilizadas, caso existam.

O padrão MPEG 2 [MPE 94] define um perfil de codificação espacial que permite a codificação e decodificação espacial e SNR. Uma entrada pode ser composta

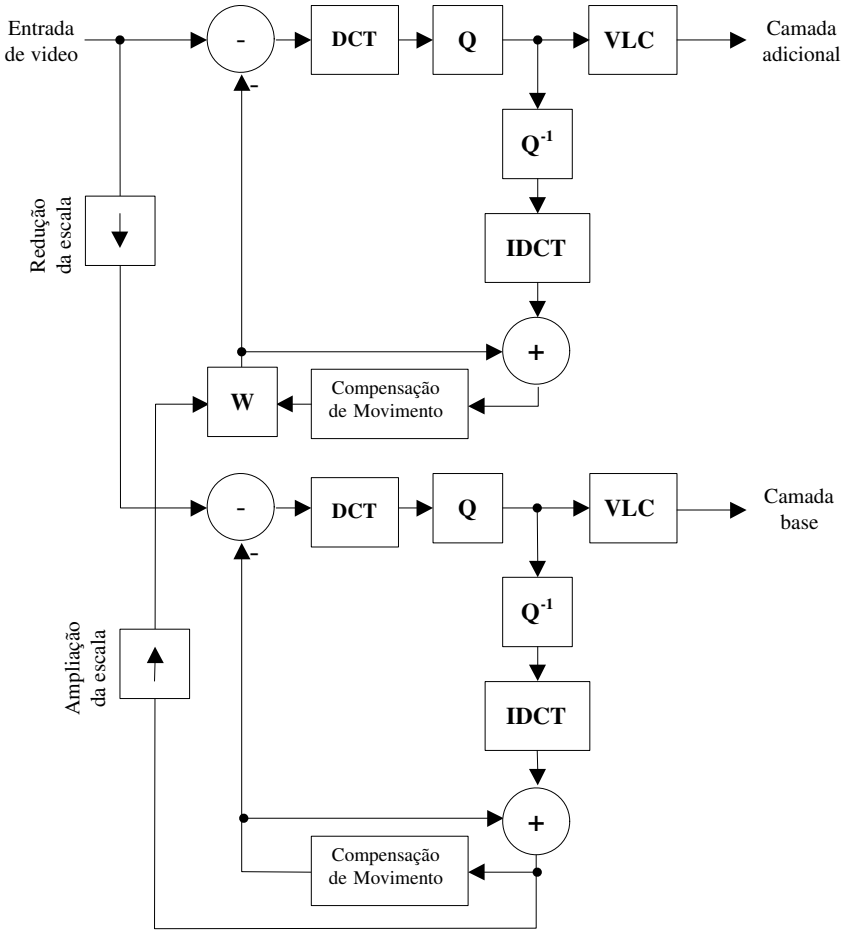

FIGURA 3.5 – Codificador Escalar Espacial

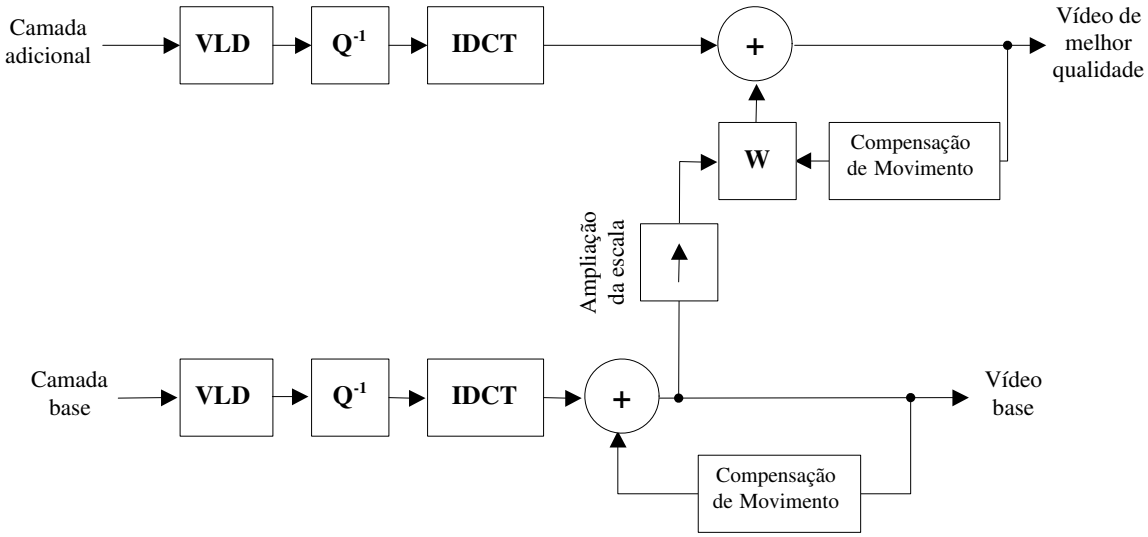

FIGURA 3.6 – Decodificador Escalar Espacial

de três camadas, no máximo. Quando utilizado apenas duas camadas, a base e uma adicional, um decodificador que esteja em conformidade com o perfil de escalabilidade espacial do MPEG 2 deve ser capaz de tratar tanto espacial quanto SNR na camada adicional. Se forem utilizadas as três camadas, as duas camadas adicionais devem ser espacial e SNR. A ordem das camadas não tem influência, mas não podem ser codificadas pelo mesmo modelo de escalabilidade.

Este modelo foi criado para permitir a compatibilidade em dispositivos HDTV. Um dispositivo que possua apenas a escalabilidade SNR pode, por exemplo, receber também uma entrada do perfil espacial e decodificar até a segunda camada apenas, assumindo que a terceira seja a espacial. Todos os dispositivos devem ser capazes de decodificar a camada base, independente de implementação de escalabilidade. A tabela 3.2 apresenta maiores detalhes sobre o perfil de codificação espacial do MPEG 2. O perfil espacial apresenta apenas uma classe de codificação, Alto-1440 (*High-1440*). Esta classe define resolução máxima de 1440x1152. O perfil Alto do MPEG 2 define, também, o uso de escalabilidade espacial nas classes Alto (High), Alto-1440 e Base (*Main*). Quando utilizadas duas camadas, a separação temporal é realizada juntamente com as demais, com excessão do perfil Alto na classe Base. A escalabilidade temporal é apresentada a seguir.

TABELA 3.2 – Tabela do perfil SpatialProfile e HighProfile do MPEG 2

Formato de cor	Spatial: 4:2:0 High: 4:2:2
Escalabilidade	SNR e/ou Espacial
Num. Max. de camadas	3 (Base + 2)
Max. Bit Rate @High-1440	Até a camada 3: 60 Mbps Até a camada 2: 40 Mbps Camada base: 15 Mbps
Resolução @High-1440	Camada adicional: 1440x1152 60Hz Camada base: 720x576 30Hz
Max. Bit Rate High@High	Até a camada 3: 100 Mbps Até a camada 2: 80 Mbps Camada base: 25 Mbps
Resolução High@High	Camada adicional: 1920x1152 60Hz Camada base: 960x576 30Hz
Max. Bit Rate High@High-1440	Até a camada 3: 80 Mbps Até a camada 2: 60 Mbps Camada base: 20 Mbps
Resolução High@High-1440	Camada adicional: 1440x1152 60Hz Camada base: 720x576 30Hz
Max. Bit Rate High@Main	Até a camada 3: 20 Mbps Até a camada 2: 15 Mbps Camada base: 4 Mbps
Resolução High@Main	Camada adicional: 720x576 30Hz Camada base: 352x288 30Hz

3.3 Escalabilidade Temporal

Um codificador escalável temporal permite que se extraia um vídeo em múltiplas taxas de quadros por segundo. A camada base possui uma taxa reduzida de quadros por segundo que, somada às demais camadas, atinge a mesma taxa em que o vídeo foi digitalizado. Por exemplo, a camada base pode transmitir apenas os quadros pares, enquanto que uma camada superior transmite os quadros impares.

Existem várias técnicas para obter a escalabilidade temporal de um vídeo [CON 99, YAN 2000]. Estas técnicas estão diretamente relacionadas ao algoritmo de codificação temporal empregado. No modelo de compressão baseado em compensação de movimento, a técnica mais empregada é a proposta pelo padrão MPEG 2 [MPE 94]. Neste padrão, a compressão temporal é realizada com a compensação de movimento gerando 3 tipos de quadros:

- Quadros I: formados pela imagem completa, como se fosse compactada com o modelo JPEG. Este tipo de quadro é usado como referência, não possuindo nenhuma dependência com relação a qualquer outro tipo de quadro. É o que necessita de mais espaço para ser armazenado.

- Quadros P: Quadros de previsão, formados pela previsão entre o quadro atual e um quadro I ou P anterior. Estes quadros são compostos pelo algoritmo de compensação de movimento em si. Usa-se como referência um quadro I ou um quadro P anterior e, a partir disso, geram-se os vetores de movimento e a diferença. Se forem utilizados muitos quadros P como referência para gerar outros quadros P, a distorção tende a se propagar entre estes e a qualidade vai degradando-se. recomenda-se que a previsão seja gerada, pelo menos, com um quadro I por segundo.

- Quadros B: Quadros de previsão bidirecional. Sua função é preencher a lacuna existente entre um quadro I e um quadro P. Para reduzir processamento e espaço de armazenamento, o padrão MPEG costuma utilizar uma distância média de dois quadros entre um I e um P ou entre dois Ps. Por exemplo, quando a freqüência for de 30 quadros por segundo, compacta-se apenas um I e nove P. Os demais quadros são compostos pela previsão bidirecional. Esta previsão interpola os dois quadros de referência através de uma média simples. O resultado desta interpolação é subtraído do quadro original e a distorção resultante forma o quadro B. Esta distorção ainda é reduzida com maior quantização para alcançar um maior índice de compactação, mas gerando um ruído ainda maior. Os quadros B são os que ocupam menor espaço de armazenamento.

Em geral, o MPEG segue uma ordem de compressão através de um grupo de imagens (GOP) e são compactados de modo a formar a seguinte ordem de quadros: I, B, B, P, B, B, P, B, B, P [MIT 97]. Esta ordem é variável de acordo com a implementação. A ordem de codificação e decodificação dos quadros não obedece à mesma ordem de entrada. Para formar um quadro B, é preciso ter codificados os quadros I e P anteriormente. A ordem de codificação seria a seguinte: I, P, B, B, P, B, B, P, B ,B.

No caso de uma transmissão escalável temporal, a camada base deve conter os quadros que não necessitarem de nenhum tipo de dependência. No caso, os quadros I. As outras camadas podem conter os demais quadros, conforme a dependência.

Um quadro P depende de um I e um quadro B depende de I e P. Um exemplo de codificação escalável temporal utilizando compensação de movimento pode ser visto no esquema da figura 3.7, onde os quadros I e P são transmitidos na camada base, e os quadros B são transmitidos em uma camada adicional.

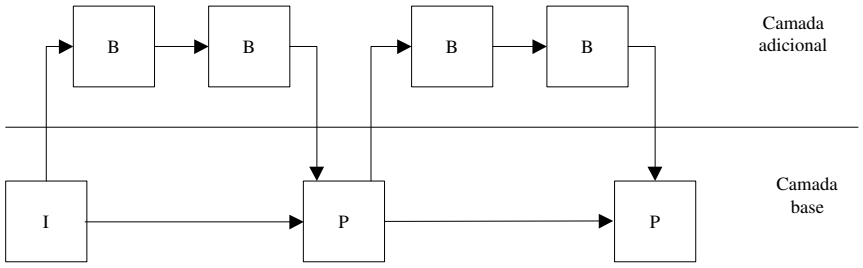

FIGURA 3.7 – Escalabilidade Temporal

Outro modelo de compressão escalável temporal pode ser empregado no caso de utilização da transformada 3D de wavelet como algoritmo de compressão temporal [TAU 94]. Neste caso, uma série de quadros é usada como entrada e a transformada de wavelet é aplicada no sentido temporal. Isto irá gerar a divisão desta série de quadros em freqüências altas e baixas. As freqüências, no domínio temporal, são dadas pela alteração existente em cada quadro. Quando existir muita alteração de um quadro para outro, esta alteração será mais visível em sua freqüência alta. Com esta divisão feita, é possível que duas camadas temporais sejam geradas: uma com informações de freqüência baixa, formando a camada base, e outra com informações de freqüência alta, que formará as adicionais. Uma deficiência deste esquema é que, quando a seqüência de quadros possuir muita alteração (um filme de ação, por exemplo), a camada base não refletirá esta alteração de maneira correta pois grande parte da "ação" ficará na camada adicional. Uma solução é utilizar este método para divisão das camadas e o emprego conjunto de um método de compensação de movimento mais suave.

3.4 Escalabilidade por Particionamento de Dados

Também conhecido como escalabilidade por freqüência, este processo é similar ao modo progressivo do JPEG. Este método divide os dados do domínio de freqüências em camadas distintas. Estes dados são resultados da transformada, normalmente DCT, aplicada nos valores de entrada [MPE 94]. Como a DCT costuma ser dividida em blocos de 8x8, a saída apresenta-se com 64 valores no domínio freqüência. Estes valores, após serem quantizados, são particionados em duas ou mais camadas. A primeira, a camada base, contém os valores mais críticos, compostos pelas freqüências baixas: valores como o DC e os ACs mais significativos. Os demais valores AC são transmitidos em outras camadas.

Esta técnica é utilizada para visualização progressiva de uma imagem em um dos modelos JPEG. Os valores DC são transmitidos em primeiro lugar, o que resulta em uma imagem nebulosa, mas o usuário tem uma idéia de como será apresentada a imagem final. À medida em que os demais valores ACs são apresentados, a imagem torna-se mais nítida.

Em uma mídia temporal, como é o caso de um vídeo, não há possibilidade de esperar que um quadro chegue por completo. A freqüência de um vídeo, normalmente 24 quadros por segundo, exige que cada quadro seja apresentado em alguns milisegundos. A seleção do número de camadas é feita de modo a possibilitar a apresentação de cada imagem no seu devido tempo. O resultado, dependendo do número de camadas, é um vídeo constantemente nebuloso, porém do qual se consegue ter noção sobre o que está sendo apresentado. À medida em que a capacidade da rede for aumentando, o número de camadas tambem irá aumentar e, assim como a visualização progressiva do JPEG, o vídeo se tornará mais nítido.

A escalabilidade por particionamento de dados pode ser implementada com pouca complexidade, se comparada às demais técnicas de codificação escalar. Os 64 valores são organizados de modo a deixar as freqüências baixas mais à frente de um vetor. Esta leitura é chamada *zigzag*. Um ponto de particionamento é definido conforme mostra a figura 3.8. Este ponto varia de acordo com a quantidade de camadas a serem criadas. O vetor é interrompido exatamente neste ponto e, então, começa a leitura da próxima camada. Ambos vetores passam por uma compressão de entropia.

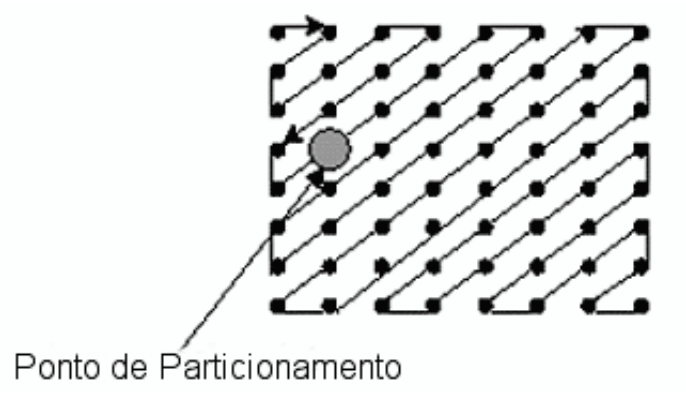

FIGURA 3.8 – Ponto de Particionamento

O padrão MPEG 2 define a criação de, no máximo, duas camadas para esta técnica de particionamento. Esta técnica costuma ser utilizada juntamente com outras, como SNR e espacial, para aumentar o desempenho. Os valores dos vetores de movimento também devem ser incluídos na camada base para alcançar a correta decodificação dos dados.

3.5 Escalabilidade Granular Fina

Um dos problemas nos modelos apresentados anteriormente é a sua característica de codificar no máximo duas camadas. Apesar do modelo de escalabilidade espacial do MPEG 2 apresentar três camadas, são utilizadas duas técnicas distintas (espacial e SNR). O objetivo da Escalabilidade Granular Fina [LI 2001] (FGS - *Fine Granular Scalability*) é permitir um número maior de camadas adicionais. O aumento deste número permite um ajuste mais preciso da qualidade de recepção durante a transmissão.

A escalabilidade fina é uma funcionalidade almejada por vários codificadores. Várias técnicas foram propostas para atingí-la. O padrão definido na norma MPEG 4 utiliza escalabilidade por *bit-plane* devido à sua simplicidade de implementação e eficiência na codificação, se comparada às demais técnicas. A técnica apresentada nesta seção baseia-se no padrão definido no Adendo IV da norma MPEG 4.

A intenção primordial é codificar um vídeo de maneira que ele possa ser visualizado por qualquer dispositivo MPEG 4, independente ou não de possuir recurso de escalabilidade. Para isso, utiliza-se uma das codificações apresentadas nas seções anteriores para gerar uma camada base e uma camada adicional. A camada base pode ser decodificada por qualquer dispositivo MPEG, enquanto a camada adicional é truncada em camadas de refinamento. O algoritmo utilizado para truncar a camada adicional é denominado *bit-plane*.

Em uma codificação normal baseada na transformada DCT, como o MPEG 4, os coeficientes são quantizados e codificados utilizando um algoritmo de entropia baseado em RLE e Huffman. Portanto, uma seqüência de valores como $2, 0, 0, 0, 0, 3, 0, 0, 0$ pode ser reduzida por RLE para $(2, 3), (3, 3)$, onde o primeiro valor representa o coeficiente e o segundo valor representa a quantidade de zeros seguidos existente. No caso da codificação por *bit-plane*, cada coeficiente quantizado é considerado um número binário.

Um bloco 2D-DCT de 8x8 pode ser visualizado como um vetor de 64 coeficientes. Um *bit-plane* deste bloco é definido como um vetor de 64 bits. O número de *bit-planes* possível para este bloco é definido pelo *bit-plane* mais significativo (MSB), aquele que contém o bit mais significativo do maior coeficiente no vetor. Começando pelo MSB, cada coeficiente é decomposto em bits e dispostos em vetores.

O seguinte exemplo ilustra o procedimento. Assumindo o seguinte vetor como sendo o bloco 2D-DCT, já quantizado e computado pelo codificador:

```
10, 0, 6, 0, 0, 3, 0, 2, 2, 0, 0, 2, 0, 0, 1, 0, ... , 0, 0
```

O maior valor encontrado no vetor é 10 e o número máximo para se representar este número em binário é 4 (1010). Portanto, para representar o vetor acima são necessários 4 *bit-planes*. O MSB é o que contém o bit mais significativo do maior valor, no caso, o quarto *bit-plane*. Os demais *bit-planes* irão conter os valores dos demais coeficientes decompostos em bits, sendo um bit em cada *bit-plane*. A composição dos bit-planes para o exemplo acima é apresentada a seguir:

```
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0 (MSB)
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0 (MSB-1)
1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, ..., 0, 0 (MSB-2)
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ..., 0, 0 (MSB-3)
```

Estes quatro vetores podem ser comprimidos utilizando as técnicas de entropia apresentadas anteriormente. Como os valores de cada vetor são binários, é possível atingir uma compressão superior através de uma variação na codificação RLE. O MPEG 4 define o método de compressão RLE para cada bit-plane utilizando apenas um valor que indica o número de zeros consecutivos antes do próximo bit significativo. Por exemplo, o terceiro bit-plane (MSB-2) pode ser codificado da seguinte maneira: $0, 1, 2, 1, 0, 2$. Estudos estatísticos [LI 2001] demonstram que esta técnica apresenta resultados de compressão superior à técnica de codificação normal dos coeficientes por RLE e Huffman.

A técnica de *bit-plane* permite a separação da informação da camada adicional de modo a permitir a existência de várias camadas. O limite de camadas depende do maior número presente no vetor. No exemplo acima, é possivel criar 4 camadas adicionais, cada uma contendo um *bit-plane*, duas camadas com dois *bit-planes* cada, e assim por diante. O processo de distribuição varia de acordo com a implementação. Em geral, a distribuição do número de camadas depende da taxa de bits que deseja-se atingir, da granularidade nas camadas e da qualidade, entre outros fatores.

A diferença entre camadas está na qualidade final da imagem apresentada. Cada camada possui uma parte da informação dos coeficientes DCT. Evidente que, ao executar a operação inversa do DCT para recuperar a imagem original, nem todos os valores podem estar presentes e muitos se apresentarão com precisão inferior à original. Por exemplo, se o usuário receber somente a primeira camada (MSB) do exemplo acima, apenas o primeiro valor estará presente. Ainda assim, do valor original (10) apenas parte será decodificado (8), já que apenas o bit mais significativo foi informado. Este processo produz um resultado semelhante ao encontrado na escalabilidade por SNR. A qualidade da imagem final codificada com escalabilidade FGS, em comparação com SNR na mesma taxa de bits, apresenta resultados superiores [LI 2001].

Uma técnica empregada para melhorar a qualidade final da imagem é o uso de *bit-plane shifting*[LI 2001]. Como foi apresentado anteriormente, os coeficientes mais significativos do DCT representam as freqüências baixas e estão localizados no início do vetor, enquanto que os coeficientes que representam as freqüências altas estão localizados mais ao final e tendem, em sua maioria, a apresentar valores mais baixos que as freqüencias baixas. O *bit-plane shifting* rotaciona os bits dos primeiros valores de modo que estes aumentem sua precisão. Por exemplo, o valor 6 no vetor apresentado anteriormente é rotacionado em um bit e passa a valer 12(1100). Isso permite que este valor seja incluído na primeira camada (MSB), aumentando a quantidade de informação transmitida por ela. Evidente que o decodificador deverá rotacionar o coeficiente em um bit no sentido inverso.

A figura 3.9 apresenta um codificador escalar FGS, conforme apresentado no Adendo IV do MPEG 4. A camada base é codificada de maneira normal, apenas com uma qualidade reduzida em relação ao vídeo original, de modo que possa ser utilizada em qualquer decodificador MPEG 4. A camada adicional executa uma subtração do vídeo original com o vídeo presente na camada base. O sinal resultante passa pela transformada de DCT e têm seus bits rotacionados (*bit-plane shifting*) e compactados. A decodificação, apresentada na figura 3.10, executa o processo inverso.

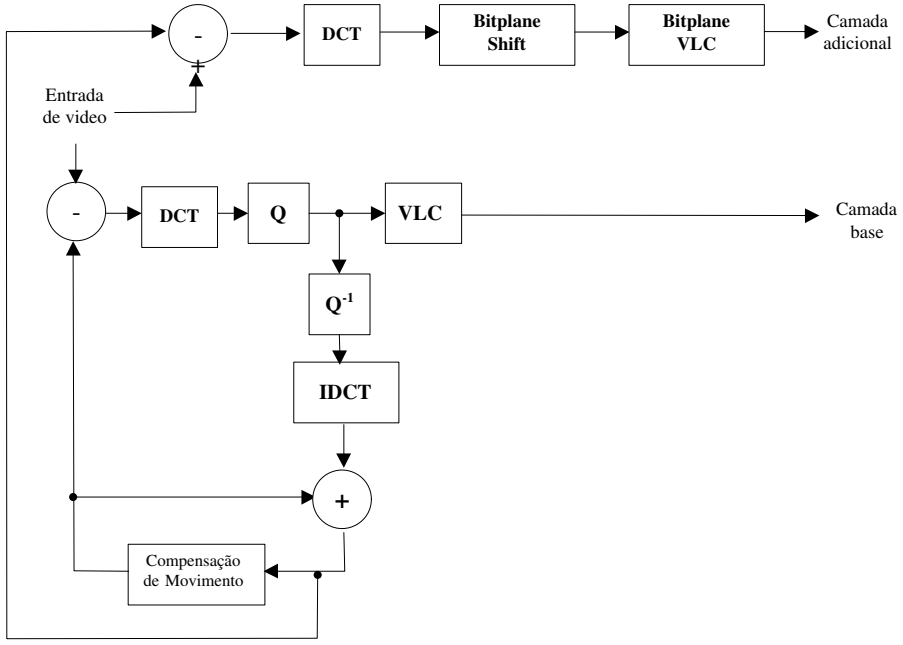

FIGURA 3.9 – Codificador Escalar FGS

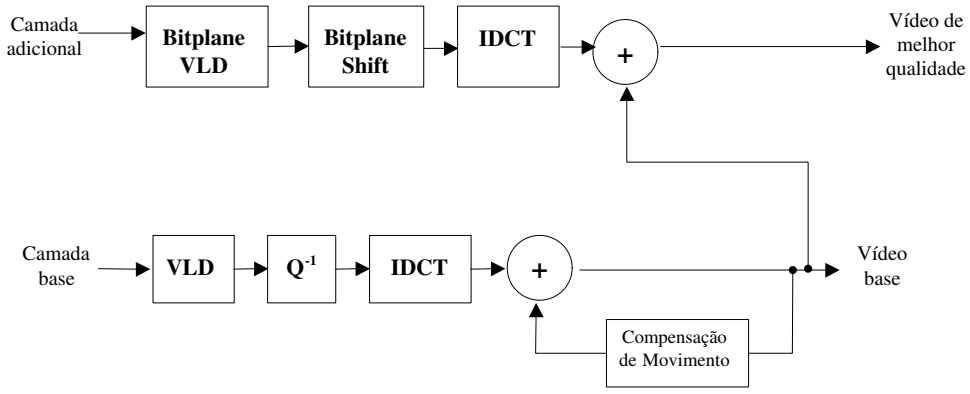

FIGURA 3.10 – Decodificador Escalar FGS

4 Transmissão em Camadas

A chave para o envio eficiente de dados multimídia pela Internet está na conciliação da taxa de transmissão e da banda disponível [LI 99, MCC 97]. Isso pode ser um grande problema, já que existem diversos ambientes e cada um possui características de rede diferentes. No único da decada de 90, a banda *Rolling Stones* transmitiu uma apresentação ao vivo através do Mbone (*Multicast Backbone*), uma rede que se utiliza da infra-estrutura da Internet para trafegar dados em Multicast. Apesar da tecnologia multicast ser eficiente em economia de banda, não é adaptativa, já que os usuários costumam estar conectados à Internet em taxas de recepção diferentes. Um usuário conectado à Internet por um cable modem teria a mesma qualidade de recepção que outro conectado por meio de uma linha telefônica. Um exemplo comum é o da infra-estrutura da Universidade de Berkeley, apresentada na figura 4.1, que transmite seus seminários para o campus e para o Mbone. Para ser capaz de maximizar a qualidade recebida pelos usuários do Mbone, os seminários são transmitidos à velocidade de 128kbit/s, que torna inviável o recebimento para usuários de modem e subutiliza a capacidade da rede do campus.

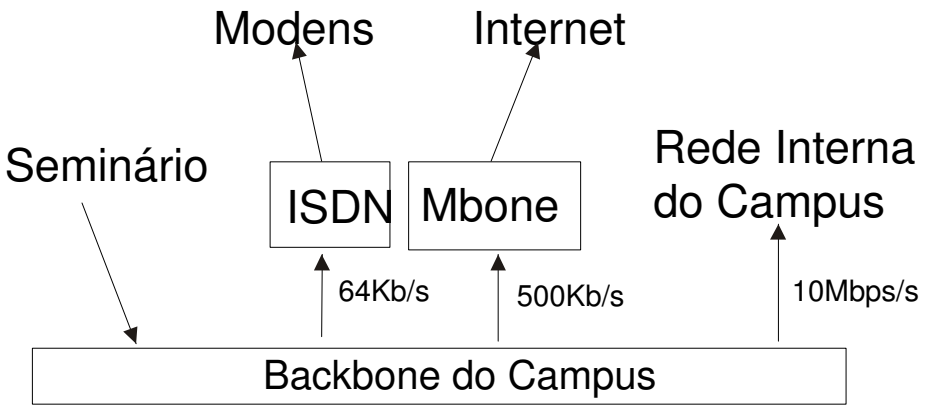

FIGURA 4.1 – Backbone da Universidade de Berkeley

Uma solução melhor é ajustar a taxa de transmissão de modo que ela se adapte às capacidades da rede dinamicamente. Um estudo foi realizado com vídeo em taxas adaptativas [KAN 93] e com o ajuste feito pelo servidor. Apesar de ser um avanço, pois consegue adaptar a taxa de transmissão para que haja uma melhor recepção pela maior parte dos usuários, o fato de existir uma única taxa de transmissão (em qualquer momento) ainda irá gerar um gargalo para aqueles que possuem pouca banda de acesso, ou má utilização de recursos para os que possuem maior banda [BAJ 98].

Shacham [SHC 92] propôs a utilização de um algoritmo de codificação em camadas num ambiente de transmissão controlado. O objetivo é codificar o vídeo em várias camadas com diferentes taxas de transmissão de ordem progressiva, onde a primeira conteria a estrutura mais simples do vídeo, o básico necessário para se assistir à transmissão, e cada camada superior conteria um refinamento na qualidade. É possivel observar que a banda B_i necessária para uma determinada camada i é a soma das camadas iguais e menores a ele ($B_i = \sum_{j=0}^{j=i} C_j$, onde C_j é a banda da

camada j). No caso da primeira camada ter uma taxa de 20 kbit/s e a segunda ter uma taxa de 40 kbit/s, seriam necessários 60 kbit/s de banda disponível na rede para que um usuário receba até a segunda camada. Este algoritmo possibilita o controle da heterogeneidade quando aliado à uma técnica para transmissão que permita a diminuição das camadas em determinados pontos da rede através de um mecanismo de controle.

Apesar desta abordagem ser bastante eficiente, é necessária uma alteração no mecanismo da rede: a inclusão de pontos de controle nos roteadores. Para solucionar isto, Deering [DEE 93] propôs a utilização de multicast como o meio de controle e distribuição das camadas de vídeo. Cada camada é distribuída utilizando um grupo multicast diferente. Cabe ao receptor informar a quais camadas deseja assistir, controlando o gargalo na conexão através da sua inscrição e retirada de grupos multicast.

A figura 4.2 mostra um modelo com 3 camadas (L1 a L3) transmitindo em grupos multicast diferentes. Os receptores de R4 a R6 se associam somente à mais baixa, a camada base, recebendo uma qualidade baixa e onerando pouco a rede, enquanto os receptores de R1 a R3 se associam a todos os fluxos, recebendo uma qualidade teoricamente igual à soma de todas as larguras de banda utilizadas, porém, precisando de uma rede mais rápida.

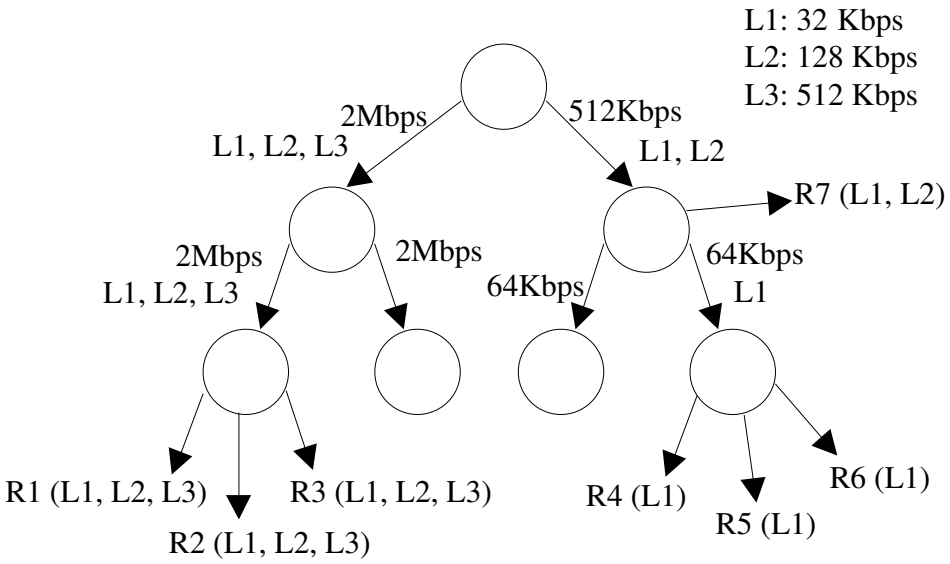

FIGURA 4.2 – Transmissão Multicast usando Camadas

Neste modelo, nenhuma sinalização precisa ser usada entre os receptores e os roteadores, ou entre o receptor e o transmissor, tornando esta solução perfeitamente escalável. Mas este modelo somente pode ser utilizado plenamente se os receptores conseguirem perceber e se adaptar automaticamente e de maneira dinâmica aos recursos da rede. Para isso, McCanne [MCC 96] propõe a criação de um mecanismo de controle de congestionamento, onde cada receptor tem um módulo que detecta as alterações na rede em termos de congestionamento e se adapta, buscando aliviar o tráfego e melhorar o fluxo.

4.1 Controle de Congestionamento

O controle de congestionamento apresentado por McCanne [MCC 96] tem como função informar ao receptor o momento de inscrever-se em um novo grupo multicast e, assim, melhorar a qualidade de recepção, ou sair de um grupo e reduzir sua qualidade.

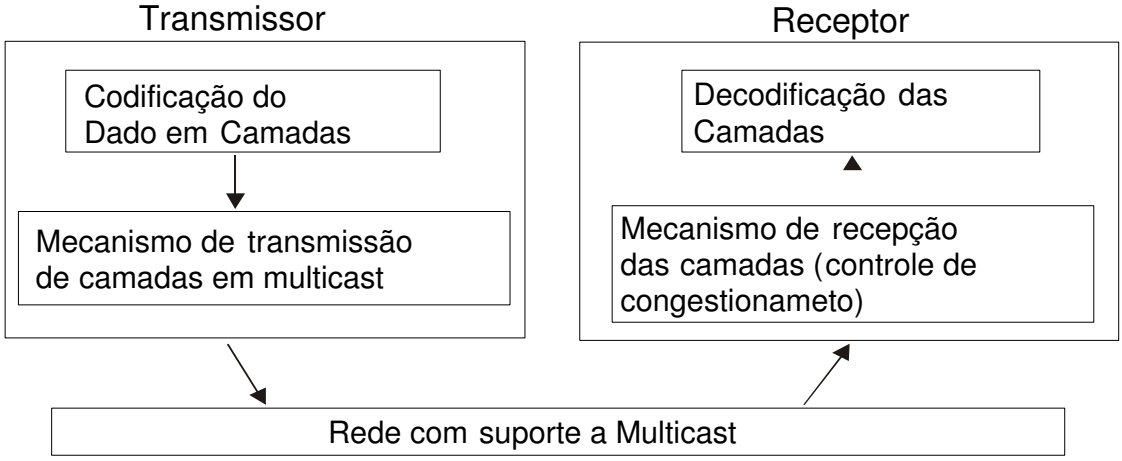

FIGURA 4.3 – Modelo de Controle de Congestionamento

Para isso, este controle utiliza-se de meios que possam detectar os recursos disponíveis na rede, como banda disponível, taxa de perda de dados, entre outros fatores.

Um mecanismo de controle de congestionamento para aplicações multimídia possui requerimentos bastante específicos: o usuário deseja sempre obter a maior qualidade possível (alta velocidade com baixa perda) e deseja evitar constantes alterações na qualidade recebida.

O mecanismo de controle de congestionamento ideal deve seguir algumas regras [LEG 99]:

1. Estabilidade: o mecanismo não pode ter muitas oscilações entre as camadas (consecutivas entradas e saídas de grupos).

2. Eficiência: deve permitir que o receptor atinja o melhor aproveitamento possível da taxa de transmissão.

3. Justo: ele deve ser justo com os demais fluxos, tanto da transmissão (demais camadas) quanto qualquer outro tráfego na rede. Uma transmissão não pode interferir significativamente com os demais dados que trafegam no mesmo caminho.

4. Robusto: deve ser sólido, sem o risco de quedas ou variações de desempenho.

5. Escalável: deve permitir a inclusão e retirada de receptores sem grande interferência na transmissão dos demais.

6. Viável: deve ser de fácil implantação, sem a necessidade de alteração no meio de transmissão (roteadores, conexões, etc.).

A seguir, são apresentados quatro mecanismos de controle de congestionameto: RLM, RLC, PLM e ALM.

4.1.1 RLM

O *Receiver-driven Layered Multicast* (RLM) [MCC 96] foi o primeiro mecanismo de controle de congestionamento a ser proposto e é considerado a base para estudos em protocolos de camadas.

Tem a vantagem de não precisar de nenhum tipo de alteração na rede, seus únicos requisitos são:

- Uma rede com política de "melhor esforço", onde todos os pacotes são tratados de maneira igual pelos roteadores, ou seja, sem a necessidade de ordenação de pacotes, banda mínima reservada, etc. Este é o modelo usado na Internet atualmente.
- A implementação de um protocolo multicast eficiente.
- Uma comunicação de grupos, onde o transmissor não saiba da existência de receptores e cada receptor possa entrar e sair de grupos de modo eficiente.

Não é exigida nenhuma alteração na concepção inicial de um transmissor de multicast em camadas, ou seja, o transmissor precisa apenas codificar o vídeo em camadas e transmitir cada camada em um grupo multicast separado.

Toda a lógica do mecanismo está no receptor que, basicamente, se inscreve nas camadas da transmissão até detectar congestionamento na rede e, então, retorna para o nível anterior ao congestionamento.

A capacidade de detectar uma rede congestionada fundamenta-se na percepção de perda de pacotes nas camadas. A capacidade de aumentar a qualidade através da inscrição em novas camadas é dada através de experimentos (*join-experiment*). Caso o experimento cause congestionamento na rede, a última camada é liberada, caso contrário, o experimento é considerado um sucesso e o usuário tem sua qualidade de recepção aumentada com a inclusão de uma nova camada.

O experimento é executado de acordo com um contador no receptor (*join-timer*). Para evitar constantes tentativas fracassadas, e conseqüente congestionamento constante na rede, a definição deste contador de tempo cresce logaritimicamente. Quando um experimento retorna um fracasso, então o tempo para executar um novo experimento aumenta.

A figura 4.4 mostra o funcionamento deste procedimento através de um gráfico de camadas no tempo. O receptor obtém sucesso nas três primeiras camadas, representadas no gráfico por uma linha que atinge L3. Ao atingir a quarta camada, L4, é detectado o congestionamento e o algoritmo do RLM volta para L3. O contador para o próximo experimento é aumentado e é feita uma nova tentativa, que novamente fracassa. O contador é incrementado mais uma vez, agora com um tempo superior ao anterior. E assim prossegue até que não seja mais detectado perda ao aumentar uma camada. O tempo de intervalo entre cada tentativa ficará maior a cada fracasso. Se a última camada nunca puder ser atingida, a tendência é que o intervalo alcance um valor suficientemente alto para não interferir na trasmissão com consecutivos congestionamentos.

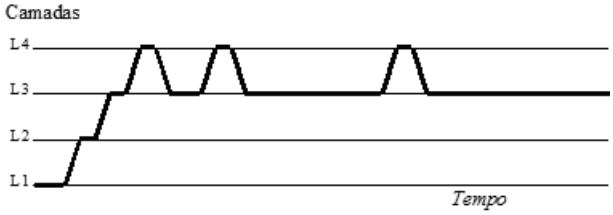

FIGURA 4.4 – Contador do RLM (*join-timer*)

Para detectar a ocorrência do sucesso de uma experiência, é utilizado um tempo de detecção (*detection-time*). Se a experiência demorar mais que o tempo de detecção sem apresentar congestionamento, então é considerada bem sucedida. Este tempo é, inicialmente, desconhecido. À medida em que forem ocorrendo fracassos, este valor é atualizado através da média entre o tempo do início da experiência e o tempo até o retorno do congestionamento. Caso seja detectada perda de mais de 25% de pacotes, então uma camada é abandonada. Apenas uma camada é incluída ou retirada a cada intervalo de detecção.

Para evitar que receptores dentro da mesma rede local iniciem tentativas com poucos instantes de diferença, gerando um congestionamento quase constante, o autor propõe a utilização de um "aprendizado compartilhado"(*shared learning*), onde um receptor envia uma mensagem multicast para todo o grupo avisando que irá fazer uma experiência. No caso de falha, todos os receptores detectam o congestionamento. Como eles foram avisados de que uma experiência está em andamento, todos reinicializam seus contadores e aguardam a hora de executar uma nova experiência. Nenhum sinal é emitido em caso de sucesso, cabe a cada receptor detectar se pode ou não se inscrever em uma nova camada.

4.1.2 RLC

O *Receiver-driven Layered Congestion control* (RLC) [VIC 98] se utiliza de uma técnica que lhe permite alcançar resultados semelhantes aos atingidos pelo TCP para controle de congestionamento [JAC 88]. O objetivo é produzir um algoritmo que seja justo com os fluxos TCP e atinja um sincronismo entre os receptores, sem a necessidade de haver um protocolo para o controle de grupos de receptores atrás do mesmo roteador, como é o caso do RLM.

O algoritmo de controle de congestionamento é orientado ao receptor, muito parecido com o do RLM: quando uma perda de dados é detectada, decresce uma camada, caso nenhuma perda seja detectada em um período t, incrementa uma camada. Seu grande diferencial está no método utilizado para solucionar alguns problemas presentes no seu antecessor.

Quando mais de um receptor está atrás do mesmo gargalo na rede (na mesma rede local), é fundamental que exista uma sincronia, pois:

1. O ato de um receptor retirar-se de uma camada não causa efeito a não ser que nenhum outro receptor esteja recebendo a mesma camada.

2. Se um receptor causa congestionamento ao se inscrever em uma nova camada, outro receptor pode detectar esta perda e, consequentemente, se retirar de

uma camada;

3. Todos os receptores atrás do mesmo gargalo podem receber as mesmas camadas.

Para solucionar estes problemas, é proposta a utilização de um pacote especial, enviado pelo transmissor, chamado de ponto de sincronização (SP - *Synchronisation Point*). Este pacote tem o objetivo de sincronizar os receptores quanto às decisões a serem tomadas. Cada receptor deve basear suas decisões apenas nas informações recebidas após um SP.

Para evitar o segundo efeito descrito anteriormente, os SP também irão definir a hora em que cada receptor irá fazer uma tentativa de aumentar uma camada. Cada SP é enviado em uma camada junto com o SP da sua camada inferior. Isso faz com que, quando um receptor tente aumentar sua qualidade, todos os receptores com qualidade igual ou inferior tentem também, solucionando o terceiro problema.

Quanto mais alta for a camada, maiores são as chances de que, ao tentar se inscrever, seja gerado congestionamento. Para evitar uma tentativa mal sucedida de aumentar a qualidade, os SP são enviados a intervalos de tempo cada vez mais esparsos, conforme mais altas forem suas camadas, de maneira exponencial.

Para otimizar o algoritmo, é proposta a utilização de sondas com a função de informar quando é mais provável que uma tentativa de aumentar a qualidade seja bem sucedida, evitando congestionamento desnecessário. A sonda consiste de rajadas enviadas em tempos regulares pelo transmissor, que envia o dobro do número de pacotes de tempos em tempos à todas as camadas. Desta forma, na existência de algum tipo de gargalo na rede por onde esta rajada trafegar, a probabilidade de ocorrer perda de pacotes ou atraso na entrega dos mesmos é maior, aumentando a chance destes eventos serem detectados pelo receptor. Tais sinais não são identificados pelo receptor como sinais para reduzir uma camada, mas sim como uma pista para não aumentar. Para tentar evitar interferência em outros fluxos, após cada rajada com duração t, o transmissor fica o mesmo tempo t sem emitir nenhum pacote. Os pontos de sincronização são enviados após cada sonda, o que permite que o receptor possa inferir a melhor hora de aumentar um nível. A figura 4.5 mostra, utilizando 5 camadas (L0-L4), o envio de uma sonda em cada camada, dentro de um período de tempo. O período da sonda é dado na região de *burst* (em destaque), onde cada pacote é representado por um bloco preto. Os blocos brancos logo após a sonda representam o tempo de espera, onde o transmissor não envia nenhum pacote. A figura 4.6 apresenta os pontos de sincronização, onde cada linha vertical representa o envio de uma sonda e cada bloco representa um ponto de sincronismo. Percebe-se que as camadas inferiores possuem maior probabilidade de aumentar uma camada, pois possuem um maior número de pontos de sincronismo.

Este processo visa evitar inscrições desnecessárias em uma camada evitando a saída do receptor de um grupo multicast. Esta fase é lenta pois, quando um receptor envia uma mensagem de requisição de saída, o roteador executa uma consulta para verificar se existe algum receptor dentro da subrede e, caso nenhuma máquina responda em um determinado período de tempo, a transmissão deste grupo multicast é interrompida para esta subrede pelo roteador. Este período de tempo pode ser grande o suficiente para gerar um congestionamento desnecessário. Para evitar que os receptores detectem este período como congestionamento, após todos saírem de uma determinada camada, é definido um período de surdez (*deaf period*) onde,

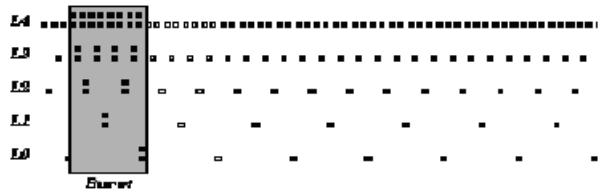

FIGURA 4.5 – Sonda RLC

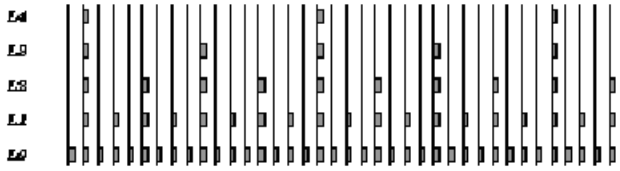

FIGURA 4.6 – Pontos de Sincronismo do RLC

depois da saída de um receptor de um grupo multicast, ele não irá reagir à qualquer turbulência na rede por um período pré-estabelecido; um pouco maior que o tempo de saída de um grupo.

4.1.3 PLM

O *packet Pair receiver-driven cumulative Layered Multicast* (PLM) [LEG 2000] tem como objetivo ser um protocolo que garanta a rápida convergência das camadas sem induzir à perda de pacotes.

A principal intenção é prever o congestionamento antes mesmo que ele aconteça. Para isso, é preciso descobrir a banda existente entre dois pontos utilizando o mecanismo do par de pacotes (*packet pair*) que apresenta um método onde o transmissor, através da emissão de pacotes em pares, pode identificar o tamanho da banda do menor enlace existente entre ele e um destino. Para funcionar em um ambiente multicast, é preciso que este mecanismo seja adaptado de modo que o receptor possa inferir a banda. Com isso:

1. O transmissor envia dois pacotes com distância zero entre si (um logo após o outro, sem nenhum tempo de latência), marcando cada par de pacote, de modo que o receptor possa identificar o primeiro e o segundo como pertencentes ao mesmo grupo.

2. O receptor calcula a banda (B) do menor enlace através do espaçamento existente entre os pacotes na chegada (T) e o tamanho dos mesmos (S) utilizando a fórmula $B = S/T$.

Para que o congestionamento possa ser previsto pelo receptor, o PLM assume que a rede tenha o suporte de um escalonador justo [LEG 99] (*Fair Scheduler* - FS) em cada roteador. Com isso, cada pacote de cada fluxo é tratado em igual tempo pelo roteador e liberado de maneira justa, ou seja, sem prioridade ou interpolação

dos fluxos. Este paradigma é conhecido como *Generalized Processor Sharing* (GPS) [PAR 93].

Na figura 4.7, apresenta-se um exemplo deste modelo, onde 2 fluxos são enviados para o roteador, 1 comum (F1) e 1 PLM (F2). Cada bloco representa um pacote. Assumindo que os pacotes possuem o mesmo tamanho e que a banda do link é B, a banda inferida na saída é de $B/2$. Quando um terceiro fluxo (F3) chega ao roteador, a banda inferida passa a ser $B/3$. Detectando esta diferença, o receptor pode prever um possível congestionamento. FS representa o fluxo de saída do roteador, onde no primeiro momento cada pacote PLM (PF2) possui um pacote intercalado do fluxo 1 (PF1). Em um segundo momento, o fluxo F3 é inserido no meio dos pacotes, aumentando a distância entre cada pacote PLM.

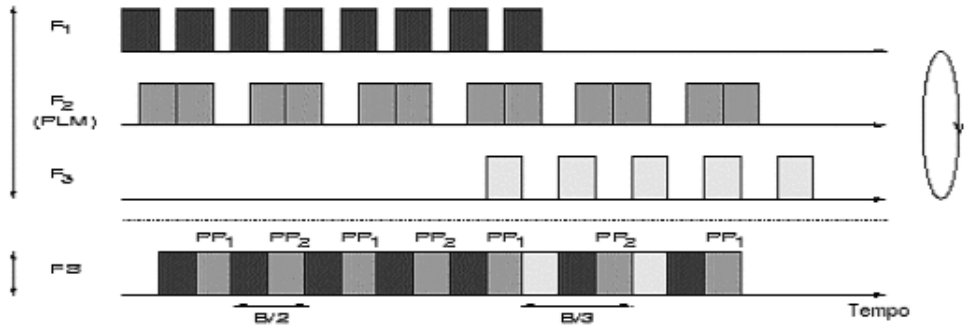

FIGURA 4.7 – Paradigma FS

O receptor precisa ter conhecimento prévio da banda transmitida em cada camada. Isto pode ser obtido através de um anúncio de sessão (*Session Announcement*). Para receber um transmissão de vídeo, por exemplo, o receptor precisa primeiro receber o anúncio da sessão com as informações da apresentação (codificadores de áudio e vídeo utilizados, número de camadas, banda em cada camada, entre outras).

Assumindo B_n como a banda obtida através do somatório da banda de todas as camadas inscritas ($B_n = \sum_{i=1}^{n} L_i$, onde a camada i possui banda igual a L) , B_i como a banda inferida pelo par de pacotes, C como o tempo de checagem (*check time*) e T_c como o tempo máximo entre cada par de pacotes, pode-se observar que quando o receptor recebe um par de pacotes no tempo t:

```
Se Bi < Bn
 Tc = t + C
 Repetir enquanto Bi < Bn
  subtrai uma camada n
  n = n - 1
Senão, se Bi >= Bn e Tc < t
 E = menor banda inferida entre Tc - C e t
 Tc = t + C
 Se E >= Bn então: enquanto Bn+1 < E
  adiciona uma camada n
  n = n + 1
```

Em resumo, subtrai camadas toda a vez que tiver uma estimativa de banda menor que a soma da banda de todas as camadas inscritas, até que a soma das bandas seja menor que a estimada. Adiciona camadas de acordo com a menor banda estimada dentro de um período de checagem C, se esta for maior que a soma das bandas das camadas. De acordo com simulações executadas pelo autor [LEG 2000], o valor ideal para C é de 1 segundo.

No caso de alto congestionamento na rede (um número alto de novos fluxos aparece em um período menor que C), é possível que os pares de pacotes (ou, apenas um dos pacotes do par) sejam descartados e nunca cheguem até o receptor. Para evitar esta perda, o PLM implementa um mecanismo semelhante ao RLC, para evitar congestionamentos sem a presença dos pares de pacotes.

Se não for recebido nenhum par de pacotes por um período t, então subtrai-se uma camada e aguarda-se mais um período t. Se forem recebidos dados referentes às camadas mas não for recebido nenhum par de pacotes, então estima-se a porcentagem de perda de pacotes nestes fluxos. Se esta for superior a um valor P, subtrai-se uma camada e aguarda-se um tempo L, chamado de período cego (*blind period*), para recomeçar o processo. As variáveis P e L foram empiricamente apresentadas por Legeout [LEG 2000] baseadas em simulações. Para L deu-se o valor fixo de 500ms e P inicia em 10% e aumenta à medida em que a perda de pacotes cresce.

Diferente dos outros protocolos, não é necessário um sincronismo explícito entre os receptores atrás do mesmo roteador, pois para sincronizar a descida de uma camada de todos os receptores ao mesmo tempo pode-se utilizar o próprio par de pacotes, então assume-se que todos recebem a mesma informação ao mesmo tempo. Já a subida em uma camada não é tão simples, visto que os receptores podem entrar a qualquer momento na transmissão. Não é proposto nenhum modelo de sincronismo neste evento, pois o receptor já estará sincronizado com os demais ao atingir a camada máxima permitida pela rede.

4.1.4 ALM

O método de controle de congestionamento empregado pelo algoritmo *Adaptive Layered Multicast* (ALM) [ROE 2001, ROE 2002, ROE 2003] é baseado em uma variável (*bwshare*) que deve refletir a largura de banda disponível na rede para aquele receptor (de forma eqüitativa entre todos receptores), e deve fazer isso sem precisar se comunicar com os outros receptores. Para alcançar este objetivo, o algoritmo busca "aumentar mais a banda para os tráfegos que têm menos" e "diminuir mais a banda para os tráfegos que têm mais", resultando num equilíbrio entre os receptores. Assim, o receptor cuja variável *bwshare* é menor vai ter um acréscimo maior (e um decréscimo menor) em relação ao receptor concorrente.

No algoritmo ALM, a adaptação ocorre no lado do receptor, e não é necessária uma interação de controle entre o transmissor e os receptores. Com o uso da adaptação no lado do receptor, o sistema em geral fica mais simples, gerando menos tráfego na rede.

O receptor se inscreve em tantas camadas quanto a variável *bwshare* permitir, e deve fazer mais de uma adição ou subtração de camadas por vez, caso seja necessário. Os principais parâmetros do algoritmo são:

- Intervalo de Execução: tempo entre duas execuções consecutivas do algoritmo. Em cada execução do algoritmo, a variável *bwshare* é incrementada ou decre-

mentada, e o receptor decide adicionar ou subtrair uma camada baseado nesta variável;

- Taxa de incremento: quantidade de banda adicionada à variável *bwshare* a cada intervalo de execução quando não ocorrem perdas;
- Taxa de decremento: quantidade de banda subtraída da variável *bwshare* a cada intervalo de execução quando ocorrem perdas.

Um resumo do código do algoritmo ALM será exposto a seguir, e explicado logo após, com o objetivo de mostrar os pontos principais da implementação.

```
Init ()  ;# rotina de inicialização das variáveis
  add-layer (camada1) ;# começa na camada 1
  cngmode = StartState
  Alm_EI() ;# executa agora e cada Intervalo de Execução     (1)

Alm_EI () ;# código executado cada Intervalo de Execução
  # repete essa função cada $ALM_EI_delay                    (2)
  $ns_ at [$ns_now + $ALM_EI_delay]  "AlmEI ()"

  switch [$cngmode]
    "StartState" { ;# Fase Start-State                       (3)
      if {$numloss==0}
        bwshare = $bwshare + ($bwshare / 4)
      else  ;# perdas detectadas
        # divide banda e vai para fase steady-state
        bwshare = $bwshare/2
        cngmode = SteadyState

    "SteadyState"  ;# fase de regime permanente             (4)
      if ($numloss==0)
        bwshare = $bwshare + (1e9/$bwshare)
      else ;# perdas detectadas
        bwshare = $bwshare - ($bwshare*5%)

  # decide se receptor pode fazer join                       (5)
  if ($bwshare > $bwlevelup)
    # adiciona tantas camadas quanto possível (add-layer)
    add-layer ($bwshare)

  # decide se receptor deve fazer leave                      (6)
  if ($bwshare < $bwused)
    # retira tantas camadas quanto necessário (leave-layer)
    leave-layer ($bwshare)
    # configura banda na média entre camadas adjacentes     (7)
    bwshare = ($bwused+$bwlevelup)/2
```

Inicialmente, a sub-rotina *init* é chamada (número 1 no código). Ela inicializa algumas variáveis do ALM, se inscreve na camada base e chama a sub-rotina

ALM_EI, que é o intervalo de execução (conforme explicado anteriormente). A sub-rotina ALM_EI é repetida consecutivamente de acordo com a variável global ALM_EI_delay, que fornece o intervalo entre duas execuções consecutivas do algoritmo (ver número 2 no código).

A mudança da variável ALM_EI_delay modifica a velocidade de adaptação do receptor e a estabilidade do sistema. Com um tempo relativamente pequeno (por exemplo, 100ms), o sistema alcança sua fatia eqüitativa no compartilhamento mais rapidamente, mas não é tão estável quanto seria com um intervalo de execução maior, como 1s, por exemplo. Isso acontece pois quanto mais vezes o algoritmo é executado por minuto, mais vezes vai incrementar ou decrementar a variável *bwshare*.

A forma empregada pelo receptor para atingir rapidamente sua largura de banda eqüitativa com os outros receptores e permanecer estável depois disso é através da divisão do algoritmo em duas fases: fase *start-state* e fase *steady-state*, conforme explicado a seguir.

Durante a fase *start-state* (ver número 3 no código), o algoritmo tenta alcançar rapidamente sua banda eqüitativa com os fluxos concorrentes, e faz isso incrementando rapidamente *bwshare* até a detecção de perdas. Quando perdas são detectadas, o algoritmo considera que sua banda passou o valor eqüitativo (por causa das perdas), assim, divide *bwshare* por um fator (no caso, 2) e vai para a fase *steady-state*.

A detecção das perdas é realizada com base no número de seqüência existente em cada pacote transmitido (semelhante ao protocolo RTP). Caso o receptor detecte em qualquer camada uma falha na seqüência recebida, ele incrementa uma variável de perda.

Durante a fase *steady-state* (ver número 4 no código), o algoritmo busca "dar mais largura de banda para quem tem menos", e "tirar mais largura de banda de quem tem mais". Isso significa que, havendo dois fluxos compartilhando o mesmo gargalo, aquele com menos banda vai incrementar mais rapidamente em relação ao outro, e ambos vão tender a um ponto de equilíbrio. No caso de perdas, o fluxo com maior largura de banda vai decrementar mais rapidamente em relação ao outro, também gerando uma tendência de equilíbrio.

Tanto a fase *start-state* como a fase *steady-state* mudam a variável *bwshare*, incrementando ou decrementando de acordo com o nível de congestionamento medido (visto através das perdas). Depois disso, a variável *bwshare* é usada para descobrir se o receptor pode receber novas camadas (ver número 5 no código) ou se ele deve se retirar de algumas camadas (ver número 6 no código). Quando um receptor se inscreve numa camada, ele configura a variável *bwshare* para a média aritmética entre as duas camadas adjacentes (superior e inferior - ver número 7 no código). Isso evita que o receptor adicione ou remova camadas em curtos intervalos de execução (o que deixaria o sistema menos estável).

É importante entender que para se adaptar eqüitativamente, um receptor não precisa conhecer a variável *bwshare* dos outros receptores a fim de que a mesma seja semelhante. Isso acontece por que o cálculo realizado leva à uma sincronização e ao equilíbrio dessa variável entre os receptores, conforme explicado a seguir.

As taxas de incremento e decremento são proporcionais à largura de banda permitida ao receptor. Assim, se um receptor tem *bwshare* igual a 500 kbit/s e outro tem *bwshare* igual a 100 kbit/s, e perdas aconteçam a ambos, ambos vão decrementar 5%, por exemplo. O decréscimo do primeiro receptor será de 25 kbit/s,

e o do segundo será 5 kbit/s. Isso vai reduzir a diferença entre a variável *bwshare* de ambos (de 400 kbit/s para 380 kbit/s). O mesmo acontece ao incrementar a variável, mas em escala menor. Isso explica o motivo dos receptores não necessitarem de comunicação para possuírem uma banda eqüitativa entre eles. Conclui-se, então, que a banda permitida a cada receptor vai tender a um ponto de equilíbrio, mesmo que os receptores estejam localizados em diferentes sessões ou localizados na mesma sub-rede.

A figura 4.8 mostra graficamente a evolução da variável *bwshare*. Na figura, há cinco camadas exponenciais (32 kbit/s, 64 kbit/s, 128 kbit/s, 256 kbit/s, 512 kbit/s) - o eixo vertical representa o total de banda usada pelo receptor, que é a soma das larguras de banda de cada camada na qual o receptor está inscrito.

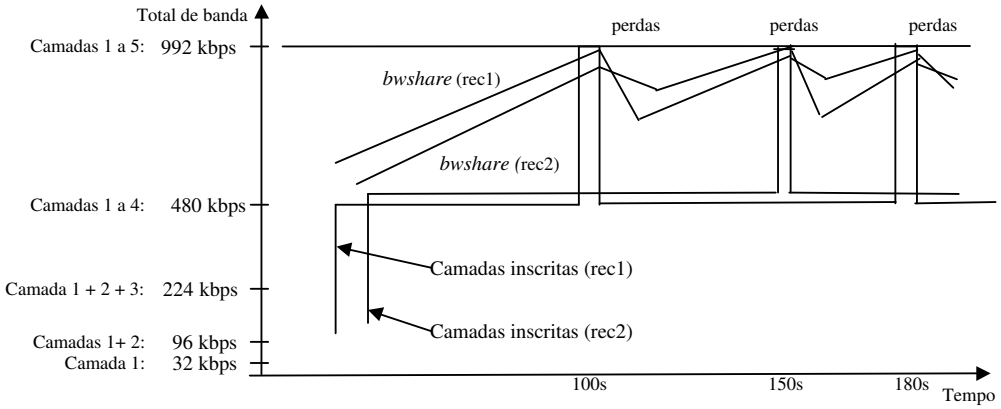

FIGURA 4.8 – Visão geral do ALM em termos da variável bwshare

As linhas inferiores na figura, apontadas como sendo as camadas inscritas, indicam a quantidade de banda que o receptor 1 (*rec1*) e o receptor 2 (*rec2*) estão utilizando. Essa quantidade é relacionada diretamente com a quantidade de camadas em que cada receptor se inscreveu. Pode-se observar que os receptores estão estáveis quando inscritos nas primeiras quatro camadas, e as tentativas de se inscrever na quinta camada geram perdas.

A largura de banda permitida aos receptores incrementa ou decrementa a cada intervalo de execução. Isso é representado na figura através das linhas superiores (*bwshare*). Essas linhas representam os resultados obtidos para a variável *bwshare* em cada intervalo de execução.

Caso a largura de banda permita (linhas superiores na figura atingindo uma largura de banda suficiente), o receptor vai se inscrever em outra camada. Caso essa tentativa gere perdas, o receptor vai abandonar uma camada.

Como pode ser inferido, se a variável ALM_EI_delay é maior, o sistema vai ficando mais lento, levando mais tempo para causar instabilidade na rede.

O algoritmo do ALM encontra-se ainda em desenvolvimento por Valter Roesler, em sua tese de doutorado. A implementação gerada até o momento baseia-se em simulações de rede. Uma implementação real está em desenvolvimento e irá atuar como suporte a transmissão multimídia na plataforma SAM. Esta plataforma é apresentada no capítulo a seguir.

5 Projeto SAM

O projeto SAM [ROE 2002a, ROE 2002] (Sistema Adaptativo Multimídia) tem por finalidade criar uma ferramenta para transmissão e recepção de apresentações multimídia em ambientes heterogêneos. Um dos objetivos é que a plataforma final permita facilmente implementar novas tecnologias de transmissão e recepção escaláveis, bem como a novos modelos de mídias.

A figura 5.1 apresenta a arquitetura proposta para o sistema. Uma camada inferior (camada 1) é composta de rotinas de acesso ao sistema operacional. Estas rotinas são responsáveis pelo tratamento da rede, interface com o usuário e acesso a recursos multimídia (câmera, microfone, vídeo). Todas as rotinas dependentes do sistema operacional devem ser apresentadas nesta camada. Isso facilita a portabilidade do SAM para outros sistemas operacionais, bastando alterar estas rotinas.

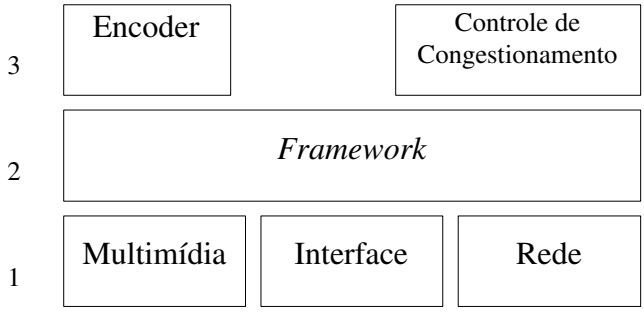

FIGURA 5.1 – Arquitetura do Sistema SAM

A camada 2 compõe-se do *framework* do sistema. Este *framework* permite a fácil comunicação dos módulos de transmissão e manipulação de mídias com o sistema operacional. Ele é composto por diretivas básicas de controle de rede e manipulação de recursos multimídia, de modo que os módulos na camada 3 não precisem preocupar-se com tarefas mais comuns, como por exemplo, a criação de rotinas para tratamento de camadas e *multicast*, com diretivas do tipo "subir_camada" e "descer_camada". Isso permite a fácil comunicação de um módulo de controle de congestionamento com a rede. O *framework* também possibilita a comunicação entre os módulos da camada 3. Um codificador pode gerar um vídeo em camadas e enviar diretamente ao *framework*, sem preocupar-se com a finalidade com que estes dados serão utilizados. O *framework*, por sua vez, pode entregar estes dados diretamente a um transmissor em camadas ou a um conformador de tráfego antes da transmissão. Cabe ao *framework* tarefas de sincronismo de mídias no lado do receptor. Um decodificador de áudio e um de vídeo, por exemplo, não necessitam controlar o tempo de apresentação. Basta enviar os dados decodificados ao *framework* que este se encarrega de apresenta-los.

A última camada é composta por módulos individuais para controle de tarefas como transmissão, codificação, conformidade de tráfego, estatísticas de dados, entre outros. Cada módulo se comunica com o *framework* de maneira individual e desconhece as demais características do sistema. Isso permite a reutilização do sistema para a implementação de novas tecnologias de controle de congestionamento e codificação de mídias. Um decodificador pode receber dados do *framework*, decodifica-los

e devolver ao *framework* sem preocupar-se com o método empregado para recepção destes dados nem sequer com como será realizada a apresentação dos dados ao usuário.

Como a plataforma irá operar em um ambiente onde os dados são gerados e transmitidos em tempo real, a latência entre a comunicação de dados dos módulos deve ser suficientemente pequena para evitar que o sistema se torne um gargalo na apresentação. O objetivo final é produzir uma ferramenta que possa facilmente ser adaptada às novas tecnologias de transmissão e codificação escalável.

Esta ferramenta encontra-se atualmente em desenvolvimento. Os primeiros módulos a serem acoplados serão o controle de congestionamento ALM, em desenvolvimento por Valter Roesler, e o módulo de codificação de vídeo, em desenvolvimento nesta dissertação. A aplicação de teste inicial baseia-se em uma tele aula. Espera-se, ao final do desenvolvimento, obter uma ferramenta adaptativa que permita a um professor apresentar uma aula remotamente a um grupo de estudantes em ambientes diversos.

6 Modelo e Implementação do VEBit

Para incluir suporte a codificação de vídeo no projeto SAM, foi proposto e implementado um modelo de codificação e decodificação de vídeo escalável batizado de VEBit (Vídeo Escalável por Bitplanes). Como o objetivo inicial do projeto é uma tele aula, e para evitar o excesso de informação trafegando no sistema em um primeiro momento, os moldes adotados são de uma transmissão de vídeo simples baseada em resolução QCIF (176x144) à 24 quadros por segundo. O modelo foi implementado de maneira independente, devido ao fato que o sistema SAM ainda está com seu *framework* em desenvolvimento. No entanto, as rotinas desenvolvidas são de fácil portabilidade, não devendo ser um problema o desenvolvimento do modelo como um módulo na ferramenta SAM.

O modelo é baseado em transformada DCT tridimensional por questões de simplicidade de programação [SER 97]. Neste primeiro momento, foi dispensada maior atenção à escalabilidade dos dados e não ao desempenho do algoritmo de compressão. O objetivo era atingir um codificador capaz de realizar a transmissão em tempo real de vídeos de baixo movimento, como videoconferências. O uso de uma transformada tridimensional facilitou este processo por permitir o uso de um codificador simples e eficiente para este modelo de aplicação.

Também foi levado em consideração o desempenho do sistema. Em um sistema baseado em compensação de movimento, a codificação necessita de um tempo superior ao da decodificação, pois o codificador precisa fazer a busca pelos vetores de movimento. Para uso em um codificador em tempo real, a característica da transformada tridimensional de possuir tempos de codificação e decodificação semelhantes foi considerada, apesar do tempo de decodificação ser superior ao de modelos baseados em compensação de movimento.

O modelo de escalabilidade utilizado é baseado em *bit-planes*. A escolha deste modelo justifica-se por sua simplicidade de implementação e pela sua característica de escalabilidade granular fina. Durante o projeto, um dos objetivos era atingir o maior número de receptores heterogêneos. Assim, o sistema se adaptaria à capacidade de usuários de baixa velocidade até redes locais. Evidente que a gama de receptores nesta faixa é muito grande, então, o foco foi concentrado em usuários de baixa capacidade de banda. A escalabilidade FGS também apresenta uma melhor adaptabilidade em redes como a Internet, onde as alterações no estado da rede ocorrem constantemente. A incorporação do sistema espacial e temporal juntamente com o FGS ainda está sendo analisada.

A figura 6.1 apresenta o modelo implementado. Cada vídeo é dividido em blocos de oito quadros, que são utilizados como entrada no sistema. Cada quadro é convertido de seu formato original para um formato YUV 4:2:2 e cada uma destas primitivas (Y, U e V) é utilizada como entrada para o algoritmo de codificação. Cada bloco de oito quadros de cada uma das primitivas deve ser seccionado em blocos menores de 8x8 pixels de modo a criar um cubo de 8x8x8. Este cubo é utilizado de entrada para codificação 3D-DCT. Os coeficientes resultantes são quantizados e utilizados como entrada para um modelo escalar fino baseado em *bit-planes*, que subdivide cada coeficiente em vetores de bits e os comprime. Cada vetor forma uma camada de acordo com uma certa distribuição, conforme visto adiante. Cada um destes módulos é apresentado com mais detalhes a seguir. A figura 6.2 apresenta o

modelo de decodificação do vídeo.

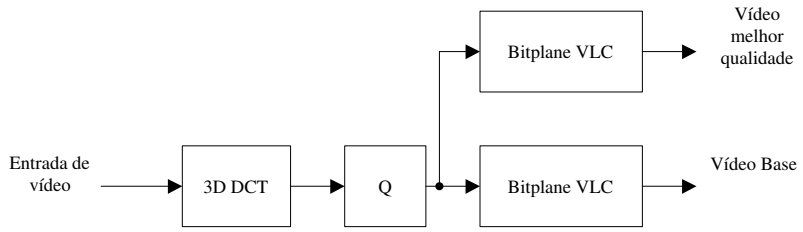

FIGURA 6.1 – Modelo de Codificador Proposto

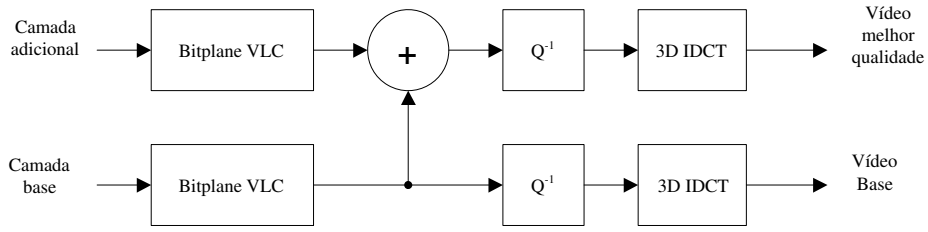

FIGURA 6.2 – Modelo de Decodificador Proposto

6.1 Implementação

A implementação foi realizada em ambiente Linux 2.4.20 na plataforma gráfica XFree86 4.2.1. O compilador utilizado foi o GCC 3.2.1. Para permitir portabilidade da implementação final para outros sistemas, os recursos gráficos foram implementados utilizando a biblioteca gráfica SDL 1.2.4 (*Simple Direct Layer*). Esta biblioteca possui funções base para acesso a hardware de vídeo, tratamanto de múltiplas tarefas (*threads*) e controle de dispositivos de entrada como teclado e mouse.

A entrada dos dados no sistema é feita através de arquivos PPM (*Portable Pixmap*). Estes arquivos são representações gráficas de imagens. O usuário informa ao sistema um diretório que contenha uma série de arquivos PPM, cada um contendo um quadro do vídeo a ser codificado. Os arquivos devem seguir uma nomenclatura formada por um prefixo e por uma série de quatro dígitos. Estes dígitos são seqüenciais, iniciando em zero. Se o prefixo informado for "frame", o arquivo "frame0000.ppm" contém o primeiro quadro do vídeo, "frame0001.ppm" o segundo quadro, e assim consecutivamente. O tamanho máximo permítido é de 10000 quadros, o que gera, aproximadamente, sete minutos de vídeo a 24 quadros por segundo. Esta não é uma limitação do algoritmo e pode ser facilmente expandida.

Cada arquivo PPM é formado por dois bytes de identificação (*magic number*). Estes bytes devem conter sempre a informação "P6". Os próximos dois bytes contém o tamanho da imagem (altura e largura) e, logo após, encontra-se uma seqüência de três bytes por pixel da imagem. Cada um destes bytes contém a informação no modelo RGB. O emprego de arquivos PPM para a codificação permite que uma

imagem seja lida no seu formato original, sem perdas. Isso permite uma análise posterior da eficiência do algoritmo de compressão.

O sistema lê oito arquivos PPM a cada interação. Estes arquivos são compactados e subdivididos em taxas complementares pelo sistema. A saída é armazenada em um arquivo conforme mostra a figura 6.3. Os sete primeiros bytes armazenam as informações de largura (W), altura (H), número de camadas (C) e quadros por segundo (FPS, em milisegundos). Após, cada grupo de oito quadros forma um grupo de camadas (GOL). O número de GOLs existentes depende do número de quadros existente no vídeo, onde n é igual os número de quadros dividido por oito. Cada GOL possui informações separadas para cada camada. Os primeiros bytes de um GOL possui as informações de tamanho para cada camada (TC), variando de 1 até C. Após o tamanho de cada camada é informado os dados que compõe a camada em si (GOP). Cada GOP é subdividido nas primitivas Y, U e V.

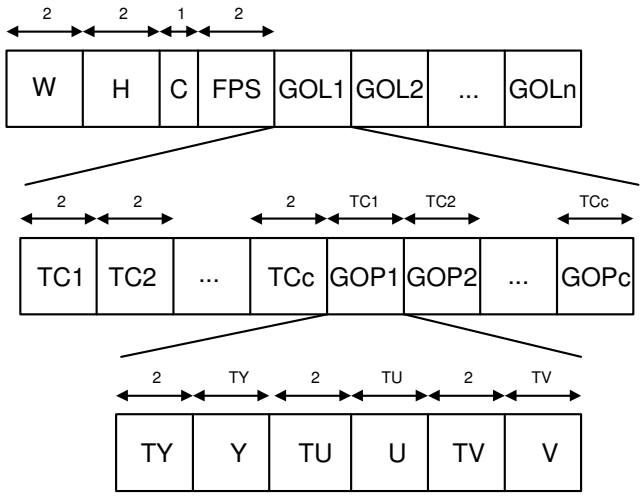

FIGURA 6.3 – Formato do Arquivo VEBit

A decodificação é feita a partir da leitura do arquivo, dependendo do número de camadas que deseja-se atingir. Por exemplo, se apenas uma camada for decodificada, apenas o GOP referente aquela camada será lido e utilizado pelo sistema. Os demais GOPs existente em cada GOL são descartados. Uma dificuldade na decodificação consiste no fato de que a cada interação são gerados oito quadros. Durante a apresentação deste vídeo, o decodificador deve esperar que os oito quadros seja apresentados antes que possa decodificar os próximos oito quadros. Isso porque as regiões de memória utilizadas para armazenar os quadros decodificados são reutilizadas. O efeito na visualização é a interrupção do vídeo a cada oito quadros. Para solucionar este problema, foram desenvolvidas duas tarefas distintas. Uma realiza o processo de decodificação e outra o processo de visualização. A cada interação do processo de decodificação, oito quadros são gerados e armazenados em uma fila circular. O processo de visualização recupera desta fila cada um destes quadros os e apresenta em seu devido momento. Para evitar que o processo de decodificação rode ao mesmo tempo que o processo de visualização, causando um travamento na seqüência do vídeo, é utilizado um "semáforo" entre as duas tarefas. A tarefa de decodificação é executada apenas no intervalo de visualização de cada quadro.

Isso permite que o vídeo seja apresentado em sua seqüência de digitalização, sem interrupções.

O processo de conversão dos arquivos de entrada do formato RGB para o formato YUV 4:2:2 empregado pelo sistema é baseado na função de conversão apresentada na equação 2.2. Uma característica desta função é o emprego de números inteiros nos cálculos e rotacionamento de bits para divisão, o que aumenta a velocidade de execução do sistema. A desvantagem apresentada é que o emprego de números inteiros em cálculos causa perda de informação por imprecisão. Para tentar reduzir esta perda, são empregados números de 16 bits para cada cálculo. Um efeito semelhante é visualizado na implementação da DCT, que será vista a seguir. A conversão de YUV para RGB novamente, durante o processo de decodificação, acontece utilizando-se as funções do SDL. Estas funções empregam a conversão automática em *hardware* existente na maioria das placas de vídeo, então, a perda neste caso é ínfima.

A implementação foi realizada utilizando um modelo de cinco camadas fixas. Cada vídeo de entrada é codificado sempre em cinco camadas. A razão para usar este número fixo é o emprego das distribuições das tabelas de Huffman, usadas na compressão, e do número de *bit-planes* em cada camada. Maiores detalhes podem ser vistos na implementação da codificação *bit-plane* apresentada a seguir.

6.1.1 3D-DCT

A entrada no módulo de transformada 3D-DCT é um vetor de 512 valores. Este vetor foi criado pelo processo de subdivisão de cada uma das primitivas em blocos de 8x8x8, conforme explicitado anteriormente. Este vetor contém informações horizontais e verticais de uma das primitivas, e informações temporais de oito quadros para a região seccionada.

Para a execução da 3D-DCT neste bloco de 512 valores, é preciso aplicar a DCT à cada uma das dimensões do cubo, conforme apresentado na equação 2.15. A DCT calcula vetores de oito valores a cada vez, retornando os coeficientes DCT referentes a estes oito valores, conforme apresentado nas equações 2.6 e 2.7. Uma nova divisão é executada para retirar cada um destes vetores de cada uma das dimensões. Portanto, 64 vetores são transformados por dimensão. No total, a função DCT é executada 192 vezes por cubo.

Para agilizar o processo de divisão do cubo em vetores, apenas uma referência a cada uma das posições de entrada do vetor inicial é encaminhado para a função DCT. Isso evita uma movimentação desnecessária dos dados e reduz o tempo de processamento da transformada. Um vetor de oito ponteiros para a posição a ser calculada sobre o vetor original de 512 valores é informado para entrada da função DCT, que executa a transformada diretamente sobre estas posições. O resultado é o mesmo vetor de entrada, agora com 512 coeficientes DCT, sendo que a primeira posição do vetor contém o coeficiente DC.

O processo de decodificação segue o mesmo algoritmo, com a diferença que a função aplicada é a DCT inversa, conforme a equação 2.16. O módulo de quantização retorna um vetor decodificado de 512 valores que são inversamente transformados. Portanto, o tempo de execução da codificação e decodificação da DCT é praticamente o mesmo.

A função DCT deve ser fortemente otimizada, pois ela representa grande parte do processamento do sistema. A implementação foi realizada utilizando-se a otimiza-

ção proposta por Loeffler, Ligtenberg e Moschytz (LLM)[LOE 89], que descreve um algoritmo DCT com 11 multiplicações e 29 adições com complexidade $O(NlogN)$. Para aumentar o desempenho do algoritmo, o número de variáveis temporárias empregadas foi reduzido para oito. Isso permite uma maior otimização por parte do compilador, pois cada variável pode ser representada em um registrador do processador. O algoritmo final implementado da FDCT consiste de 16 multiplicações e 26 adições, já a IDCT apresenta 16 multiplicações e 25 adições.

Assim como a conversão de RGB para YUV, as funções DCT utilizam cálculos com números inteiros para reduzir o tempo de execução. Novamente, o problema de utilizarem-se números inteiros é a precisão dos dados. Cada valor de ponto flutuante foi convertido em um número inteiro através de um aumento na escala. Este aumento é encontrado pela multiplicação do número de ponto flutuante por 2^n (onde n é a escala). Divisões podem ser substituídas por multiplicações ($3/2 = 3 * 0.5$) e operações em geral são executadas com estes valores inteiros. Ao final, os valores resultantes são escalados novamente, agora sendo divididos por 2^n. Os processos de escala podem ser substituídos por simples movimentação de bits. Dependendo do número de ponto flutuante, o processo de escala pode perder precisão. Quanto maior a precisão da escala, mais bits são necessários para armazenar o número inteiro e maior é a precisão do valor final.

A utilização deste processo na DCT tem um agravante: como alguns valores do resultado do cosseno possuem precisão muito grande, são necessários muitos bits para representar este valor em números inteiros. Acontece, então, que os números têm sua precisão reduzida, na maioria das vezes. Esta redução causa um erro no vetor final transformado e este erro propaga-se à medida em que a função DCT é executada repetidamente. Assim, o erro resultante no vetor de saída de 512 coeficientes tende a ser elevado, já que a DCT é executada 192 vezes, gerando erro acumulativo.

Para tentar reduzir o erro ao máximo, foram utilizados valores inteiros de precisão de 12 bits na implementação. No entanto, estes valores são multiplicados entre si e, à cada dimensão que a DCT for aplicada, a quantidade de bits para representar cada valor aumenta. Isso exige que o vetor tenha capacidade de manipular o máximo de número de bits possível para evitar propagação do erro pela necessidade de redução da precisão para o armazenamento. O vetor de 512 valores emprega variáveis de 32 bits durante o processo de 3D-DCT. Somente ao final do processo, após as 192 transformadas, é que os valores são escalados para sua representação final.

Os valores finais da DCT, os coeficientes DC e AC, costumam ser de maior magnitude que os coeficientes de entrada, já que estes condensam a energia nas primeiras posições do vetor. No caso da transformada 3D-DCT, estes valores ficam concentrados na região superior esquerda dianteira do cubo. Na implementação, os valores atingem um máximo de 10 bits para o coeficiente DC, e 9 bits para os AC, após o processo de quantização.

6.1.2 Quantização

O processo de quantização empregado segue os princípios da quantização escalar. Uma tabela contendo 512 valores serve de referência para o processo de redução na precisão dos coeficientes e, conseqüentemente, uma redução na quantidade de informação a ser armazenada ou transmitida.

Após o processo de transformada, o vetor de 512 coeficientes DCT passa para o módulo de quantização. Cada posição neste vetor possui uma função diferente e deve ser tratada de maneira diferenciada. O primeiro valor, o coeficiente DC, é o de maior magnitude e também o que contém a maior parte da energia do vetor. Uma redução brusca na representação deste valor causa uma elevada perda na qualidade final da imagem. Os primeiros 64 valores após o coeficiente DC condensam a maior parte da informação do bloco. Os demais valores contém informações temporais. Uma alta redução nos primeiros 64 valores afeta a qualidade final dos oito quadros que formam o cubo de entrada. Um redução muito grande nos demais valores afeta a movimentação existente entre cada quadro.

A tabela empregada para o processo de quantização é semelhante à tabela empregada no MPEG 4 para a compressão espacial dos quadros, com o diferencial de que é empregada uma maior compressão temporal. A tabela do MPEG 4 possui 64 valores de quantização espacial. Estes valores são os mesmos utilizados nas primeiras 64 posições do vetor. As demais posições sofrem um aumento nestes valores à medida em que vão distanciando-se das posições iniciais. O fator de aumento segue a equação: $2^n + 1$, onde n é a distância do vetor inicial. Cada posição tem seu fator aumentado em relação à posição anterior. Por exemplo, a primeira posição do primeiro plano de 64 valores é 8, a primeira posição do segundo plano é 9 $(8 + 1)$, a terceira é 12 $(9 + 3)$, e assim repete-se consecutivamente até a primeira posição do oitavo e último plano, que é 55. Isto permite uma redução dos valores finais do vetor. Como o vetor possui seus valores condensados mais à frente, esta redução é possível sem incorrerem grandes alterações na qualidade final.

O processo de quantização executa uma divisão do valor original vindo do DCT pelo valor contido na tabela ($F_q(x, y, z) = \frac{F(x,y,z)}{Q(x,y,z)}$, onde F é o coeficiente DCT e Q é o valor na tabela de quantização). O valor resultante (F_q) é a valor final quantizado. Este processo é o único que utiliza um processo de divisão direta de seus valores, todos os demais utilizam divisão por rotacionamento de bits. Como resultado, este processo, juntamente com o do DCT, é o que apresenta maior parte do processamento total do algoritmo, pois a divisão é uma operação com custo de processamento elevado. O mesmo não acontece com a decodificação, já que o processo inverso é apenas uma multiplicação do valor com a sua posição na tabela. Esta relação é a que apresenta a maior parte da diferença entre os tempos de codificação e decodificação, como será visto mais adiante.

Este processo também é responsável pelo recolhimento de algumas estatísticas, como o maior valor AC. Estas estatísticas são utilizadas para codificação *bit-plane*.

6.1.3 Codificação *Bit-Plane*

Este módulo é o responsável pela subdivisão dos dados em camadas e por grande parte da compactação dos dados. O modelo empregado segue o padrão apresentado anteriormente de escalabilidade granular fina utilizado pelo MPEG 4. A entrada é dada pelo vetor de 512 valores, já quantizados, e estes valores são divididos em *bit-planes*.

O primeiro passo é a codificação diferenciada do coeficiente DC. Este coeficiente é o maior valor do vetor e deve ser tratado de maneira diferente pelo fato de ser o que contém a maior parte da energia. Em um primeiro momento, é armazenada a informação de precisão do DC. Este valor varia entre 0 e 10. A maior precisão

possível do coeficiente DC são 10 bits. Um estudo estatístico em diversos vídeos mostrou que a maior parte dos coeficientes DC estão localizados entre os valores 7 e 9, sendo que a precisão 9 foi apresentado em mais de 50% dos casos. Para reduzir a quantidade de dados armazenados, estes dados estatísticos foram empregados para a criação de uma tabela para compressão por Huffman. O primeiro e segundo anexo apresentam, respectivamente, as distribuições e as tabelas Huffman criadas.

A distribuição do coeficiente DC varia de acordo com a quantidade de bits necessários para armazená-lo. Apenas as 3 primeiras camadas possuem informações relevantes ao coeficiente DC. A primeira camada contém os bits mais significativos, a segunda, os bits centrais e a terceira, os bits menos significativos. O primeiro bit nunca é transmitido, pois a precisão do DC, armazenada anteriormente, indica a posição do primeiro bit significativo. A tabela 6.1 mostra como foi realizada esta distribuição. Se a precisão do DC for 10 bits, o segundo bit significativo é sempre ignorado. Uma análise do valor máximo do DC apresenta este valor nunca ultrapassando 1500, portanto, o segundo bit é sempre 0. Precisão de valores 0 e 1 não possuem bits mais significativos, pois a referência à precisão do DC já os codifica. O objetivo deste processo é aumentar a compressão dos dados. Por exemplo, se o número de bits do coeficiente DC for 9, a primeira camada precisa apenas de 4 bits (um para representar o valor 9 e três como os mais significativos). A conseqüência é uma perda na qualidade final da imagem. Mas, como a maior parte da informação é transferida, esta perda não representa grande prejuízo. Uma análise da qualidade do DC apresenta que, se o usuário tiver acesso a duas camadas, mais de 98% da informação do DC é decodificada corretamente. Uma vantagem apresentada é a redundância de dados. As três primeiras camadas possuem o valor de precisão do DC. Durante uma transmissão, caso ocorra uma perda de dados em uma das camadas, as demais podem recuperar parte do valor original do DC.

TABELA 6.1 – Distribuição do Coeficiente DC nas Camadas

Bits	Camada 1	Camada 2	Camada 3
2	1	-	-
3	2	-	-
4	2	1	-
5	2	2	-
6	3	2	-
7	3	3	-
8	3	3	1
9 e 10	3	3	2

O próximo passo após a codificação do coeficiente DC é a codificação dos 511 coeficientes AC. Esta codificação dá-se através da divisão de cada um dos coeficientes em até 5 *bit-planes*, sendo que cada bitplane possui até 3 bits significativos. Cada camada leva 1 *bit-plane* distribuídos de acordo com a tabela 6.2. Por exemplo, se o valor do maior AC apresentar precisão de 9 bits, a camada 1 leva apenas o bit mais significativo, enquanto que o *bit-plane* da camada 3 leva os três menos significativos.

Para a codificação dos dados é utilizado um cabeçalho de 4 bits, onde os 2 primeiros bits representam a quantidade de bits no *bit-plane* (de 0 a 3) enquanto o segundo representa um código referente a codificação presente na camada. O código

TABELA 6.2 – Distribuição dos Coeficientes AC nas Camadas

Camadas	9	8	7	6	5	4	3	2	1
1	1	1	1	1	-	-	-	-	-
2	1	1	1	1	1	-	-	-	-
3	2	2	1	1	1	1	-	-	-
4	2	2	2	1	1	1	1	-	-
5	3	2	2	2	2	2	2	2	1

varia de acordo com a camada e serve para identificar em qual posição os bits deste *bit-plane* devem ser inseridos. A tabela 6.3 apreseta a distribuição destes códigos.

TABELA 6.3 – Código dos *Bit-Planes* por Camada

Camadas	0	1	2	3
1	6	7	8	9
2	5	6	7	8
3	4	5	6	7
4	2	3	4	5
5	1	2	3	4

Realizou-se uma análise de distribuição destes 4 bits e constatou-se que a maior parte possui o valor 0, que representa a inexistência de um *bit-plane* para aquela camada. Mais de 50% dos valores entram nesta categoria. A segunda maior foi a existência de apenas um bit no *bit-plane* com o código 0, isso pode ser facilmente explicado através da distribuição apresentada na tabela 6.2. A distribuição da precisão do valor AC apresenta mais de 50% entre 6 e 1. Com base nesses dados, uma tabela de Huffman foi gerada para compactar a informação do cabeçalho.

O próximo passo é a codificação do *bit-plane* em si. Cada valor do vetor de 511 coeficientes de entrada é analisado de acordo com o seu maior valor e a distribuição da tabela 6.2. Um *bit-plane* é codificado utilizando o padrão *run-lenght*. Uma seqüência de zeros é identificada dentro da seqüência de bits de entrada. Esta seqüência é codificada em 4 bits, significando que até 16 zeros podem ser armazenados em um código *run-lenght*. Um primeiro bit é inserido no vetor de bits de saída informando ser a próxima seqüência é um código *run-lenght* ou uma seqüência de bits válidos. Cada camada pode ter até 3 bits válidos, mais um bit de sinal. Para que um valor seja identificado como 0, é preciso que todos os seus bits válidos sejam zero. Em camadas com apenas um bit válido, este bit é identificado apenas pelo valor do sinal.

Novamente, tanto a codificação *run-lenght* quanto a seqüência de bits válidos apresentaram uma distribuição que permitiu uma maior compressão dos dados através de tabelas Huffman. No caso da codificação *run-lenght*, mais de 50% dos padrões encontrados foram de seqüências de 16 zeros. Isso permite uma compressão superior configurando apenas um bit para representar esta seqüência. O caso da codificação dos bits válidos apresentou apenas uma distribuição otimizada nas camadas com dois bits válidos. Neste caso, três bits foram codificados, os dois válidos mais o

sinal. As tabelas de distribuição e Huffman podem ser vistas nos anexos finais.

Para evitar uma codificação desnecessária, os dados são codificados até o último bit válido de cada *bit-plane*. Ao final, um código especial é inserido indicando o final da codificação.

6.2 Resultados

Os resultados foram gerados com base em quatro vídeos, todos no formato QCIF e normalizados a 24 quadros por segundo:

- Carphone: 376 quadros. Apresenta uma pessoa falando diretamente para a câmera dentro de um carro em movimento. Ela gesticula e movimenta a cabeça constantentemente. O carro possui um fundo azul listrado e à direita do vídeo é visível uma janela que apresenta a paisagem. Esta paisagem se altera durante o filme, com a maior parte da atualização acontecendo dos quadros 190 em diante (7 segundos de vídeo). Neste período, uma seqüência de árvores é apresentada e a atualização nesta região é constante.

- Forest: 288 quadros. Apresenta uma clareira dentro de uma floresta. A câmera executa uma aproximação até uma pessoa. Esta aproximação dura em torno de 200 quadros (8 segundos). No final, a pessoa executa uma acrobacia, indo da direita para esquerda do vídeo. A câmera mantém a pessoa centralizada e a acompanha durante a execução desta acrobacia.

- Claire: 488 quadros. Apresenta uma pessoa sobre um fundo azul. Esta pessoa fala diretamente para a câmera, com pouco movimento da cabeça. Grande parte do movimento é apenas da boca.

- Night: 200 quadros. Panorâmica em uma cidade à noite. A câmera sobrevoa a cidade gerada por computador, o que produz grande atualização de tela e de maneira constante.

Os quadros de entrada apresentam-se no formato PPM, descompactados e sem perda de qualidade em relação a digitalização original.

6.2.1 Desempenho

A análise de desempenho do algorimo foi realizada em um computador com processador Athlon 1200 Mhz com 256 Megabytes de memória. A linha de comando utilizada pelo compilador GCC foi "-O3 -mmmx". Estas opções habilitam a utilização de otimização máxima do código executável, bem como o emprego de instruções MMX sempre que o compilador achar necessário. Em todos os testes apresentados foram feitas 10 análises e geradas médias dos resultados.

As rotinas de codificação e decodificação são as principais rotinas da implementação. A rotina de codificação recebe a entrada de oito quadros no formato RGB e produz a saída de uma seqüência de bits. Já a rotina de decodificação recebe esta seqüência de bits e reconstrói os quadros originais, no formato YUV.

A tabela 6.4 apresenta o desempenho da rotina de codificação. O valor informado, em segundos, é referente a 10 execuções da rotina, suficiente para codificar

80 quadros. As resoluções dos vídeos de entrada estão no formato QCIF (176x144) e CIF (352x288). Para a execução da codificação em tempo real de um vídeo a 24 quadros por segundo, são necessárias apenas três interações da rotina de codificação. Dois vídeos foram utilizados para entrada de dados: forest e night. A principal diferença entre eles é a quantidade de diferença entre cada um dos 8 quadros utilizados como entrada. O vídeo "forest" possui pouca alteração enquanto o vídeo "night" possui um número maior de alterações. A quantidade de atualizações de tela presente influenciou bastante nos testes. Um vídeo em formato CIF com muitas alterações pode inviabilizar a execução da rotina em tempo real. Já os vídeos no formato QCIF apresentaram resultados satisfatórios em ambos os casos.

TABELA 6.4 – Desempenho da Rotina de Codificação

Resolução	forest	night
176x144	0.530	0.681
352x288	1.665	2.078

A tabela 6.5 apresenta os resultados da decodificação apenas do vídeo "night". Novamente, os resultados são referentes a 10 execuções da rotina, suficiente para decodificar 80 quadros. Percebe-se uma grande diferença de tempo na última camada. Isso pode ser explicado pelo fato que a quantidade de dados necessários para o processamento desta camada é superior às demais. Os resultados apresentados mais adiante apresentam essa discrepância com maior detalhamento.

TABELA 6.5 – Desempenho da Rotina de Decodificação

Camada	176x144	352x288
1	0.274	0.828
2	0.277	0.840
3	0.286	0.856
4	0.289	0.884
5	0.324	0.977

Uma análise mais detalhada apresentou quais as rotinas que consomem mais tempo durante a execução da codificação. A tabela 6.7 apresenta as quatro principais rotinas. O vídeo de entrada utilizado é, novamente, o "night". A DCT é a que apresenta a maior parte do tempo de execução. O processo de quantização também absorve um tempo considerável devido a utilização de uma divisão normal. Uma grande diferença de tempo se percebe entre os formatos na rotina de codificação dos valores AC. Isso se deve ao fato que a quantidade de valores AC diferentes de zero é maior, pois o vídeo possui muita atualização. O mesmo não se percebe na resolução QCIF, pois a quantidade de blocos é menor.

No processo de decodificação, apresentado na tabela 6.7, novamente a rotina DCT é a que ocupa maior parte do tempo de execução. A quantidade de dados existente no vídeo CIF causou a inclusão das rotinas de decodificação de *run-lenght* e dos ACs.

TABELA 6.6 – Tempo de Execução por Rotina de Codificação

Rotina	% do tempo total QCIF	% do tempo total CIF
fdct1D	48.91	35.90
Quantization	17.39	20.05
rgb_to_yuv	7.61	9.09
encode_ac	5.43	16.20

TABELA 6.7 – Tempo de Execução por Rotina de Codificação

Rotina	% do tempo total QCIF	% do tempo total CIF
idct1D	64.71	45.31
decode_run	0.00	7.04
decode_ac	1.96	6.57
DeQuantization	7.84	5.16

Estas estatísticas tendem a variar dependendo do vídeo. O exemplo apresentado é um dos piores casos, onde a atualização existente entre cada quadro é elevada. Isso aumenta a quantidade de valores ACs existente em cada bloco de dados, o que, além de prejudicar a compactação, aumenta o número de vezes que as rotinas de codificação do *bit-plane* são executadas. A função DCT e a função de quantização são as únicas a serem executadas o mesmo número de vezes, variando apenas de acordo com a resolução, independente da distribuição dos dados no vídeo.

A função DCT implementada apresenta uma alta complexidade computacional ($O(N^3)$), onde $N = \sqrt[3]{N_c N_l N_q}$, N_c é o número de colunas, N_l o número de linhas e N_q o número de quadros de entrada. Na implementação realizada, $N = 8$).

A tabela 6.8 apresenta o desempenho da transformada 3D-DCT implementada. Os valores são dados vezes 1000 execuções por segundo. Para fins de comparação, um vídeo no formato QCIF, com resolução de 176 por 144, necessita de 396 3D-DCT a cada 8 quadros, o que dá 1188 execuções por segundo, assumindo 24 quadros por segundo. A análise foi feita sobre uma mesma entrada de dados, com valores aletórios entre 50 e 80.

TABELA 6.8 – Desempenho da Tranformada 3D-DCT

Transformada	Kdct3D/s
fdct3D	21.7
idct3D	26.9

Apesar de ser a rotina que ocupa maior parte do tempo tanto no processo de codificação quanto na decodificação, o desempenho geral da rotina mostra-se satisfatório. Foi dada uma maior atenção apenas na perda de precisão dos dados. Os resultados da relação sinal-ruido, apresentados a seguir, mostram mais detalhes.

6.2.2 Compressão

Os teste de compressão tem a finalidade de medir a taxa de compressão atingida por cada um dos vídeos. O cálculo do número de bits necessários para se armazenar um pixel foi realizado somando-se o número de pixels existente no vídeo e dividindo pelo total de bits gerados por cada camada. A taxa de compressão é dada em porcentagem do tamanho original necessário para armazenar o vídeo. O fator de compressão apresenta a relação existente entre o tamanho do vídeo original e o tamanho do vídeo compactado. Todos os resultados são dados através de um somatório com as camadas superiores. O resultado apresentado na camada 5 é calculado somando-se a quantidade de dados presente em todas as camadas.

TABELA 6.9 – Compressão do Vídeo carphone

Camada	Bits por Pixel	% do original	Fator de Compressão
1	0,0266	0,1	903:1
2	0,0654	0,2	367:1
3	0,1173	0,4	204:1
4	0,1995	0,8	120:1
5	0,5276	2,0	45:1

TABELA 6.10 – Compressão do Vídeo forest

Camada	Bits por Pixel	% do original	Fator de Compressão
1	0,0266	0,11	903:1
2	0,0630	0,26	381:1
3	0,1179	0,49	203:1
4	0,2244	0,93	107:1
5	0,6846	2,85	35:1

As tabelas 6.9 e 6.10 apresentam os resultados de compressão dos vídeos "carphone" e "forest", respectivamente. Estes vídeos apresentam características distintas de movimento, entretanto apresentaram resultados muito semelhantes.

A tabelas 6.11 apresenta as estatísticas para o vídeo "claire". Este vídeo é composto de baixa movimentação. O resultado é uma forte compactação dos dados. Em contraste, a tabela 6.12 apresenta o resultado para o vídeo "night". A alta movimentação presente neste vídeo resultou em uma baixa compactação da informação.

Em todos os casos, percebe-se que a última camada possui um aumento superior a 100% em relação à camada anterior. Isso é devido à distribuição dos bits presentes no *bit-plane* desta camada. Após a quantização dos dados, a tendência é que a informação contida nas últimas posições do vetor resultante seja composta de valores baixos. Principalmente em vídeos de alta movimentação. Como a última camada é a responsável pelo transporte dos valores mais baixos atingidos pelos coeficientes AC, a quantidade de informação presente nesta camada é superior às demais.

TABELA 6.11 – Compressão do Vídeo claire

Camada	Bits por Pixel	% do original	Fator de Compressão
1	0,0203	0,084	1183:1
2	0,0475	0,198	505:1
3	0,0810	0,337	296:1
4	0,1237	0,515	194:1
5	0,2529	1,054	94:1

TABELA 6.12 – Compressão do Vídeo night

Camada	Bits por Pixel	% do original	Fator de Compressão
1	0,046	0,19	527:1
2	0,106	0,442	226:1
3	0,201	0,839	119:1
4	0,395	1,647	61:1
5	1,04	4,33	23:1

6.2.3 Distribuição da Banda

A distribuição da banda apresenta a quantidade de dados necessária para a transmissão ou armazenamento das informações presentes em cada camada. Os gráficos foram gerados com três amostras por segundo de vídeo. A informação apresentada é referente apenas a camada em questão. Para se transmitir a camada 2, por exemplo, é preciso banda suficiente para a camada 1 e a 2 somadas.

O gráfico 6.4 apresenta a distribuição da banda para o vídeo "carphone". Percebe-se um grande aumento da banda na última camada, em torno de 7 segundos de apresentação. Este aumento é resultado do início de uma movimentação mais elevada em uma região do vídeo, no caso, as alterações vistas na janela do carro. As demais camadas também apresentam esta movimentação, mas de maneira mais suave, chegando a ser quase inexistente na camada base. O motivo é o mesmo apresentado na seção anterior, onde valores baixos dos ACs durante uma movimentação geram um aumento na informação contida na última camada. O mesmo percebe-se no vídeo 6.5, entre os tempos 3 e 5 segundos, onde a aproximação é feita de maneira mais acelerada, e no final, durante a acrobacia.

Os gráficos 6.6 e 6.7 apresentam a distribuição de banda para os vídeos "claire" e "night". O vídeo "claire" apresentou uma taxa constante nos dados de cada camada, com exceção da última camada. O mesmo percebe-se com o vídeo "night". Em vídeos que não possuem muita variação na sua taxa de movimentação, as taxas de banda resultante também não apresentam grande variação. As camadas mais baixas, em geral, tendem a não sofrer com a movimentação em um vídeo, como mostra os gráficos anteriores.

O gráfico 6.8 apresenta a comparação entre a banda necessária para a transmissão ou armazenamento de cada vídeo. Cada ponto no gráfico representa a quantidade de banda disponível necessária para o tráfego da camada em questão, já somada com as camadas inferiores. A camada base é representada pelo índice 0, enquanto que

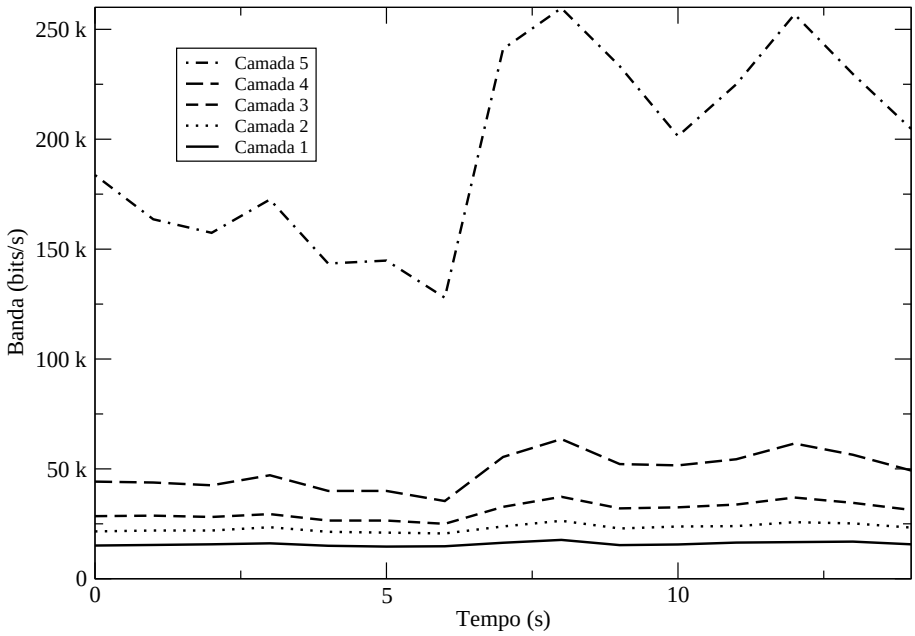

FIGURA 6.4 – Banda por Camada do vídeo carphone

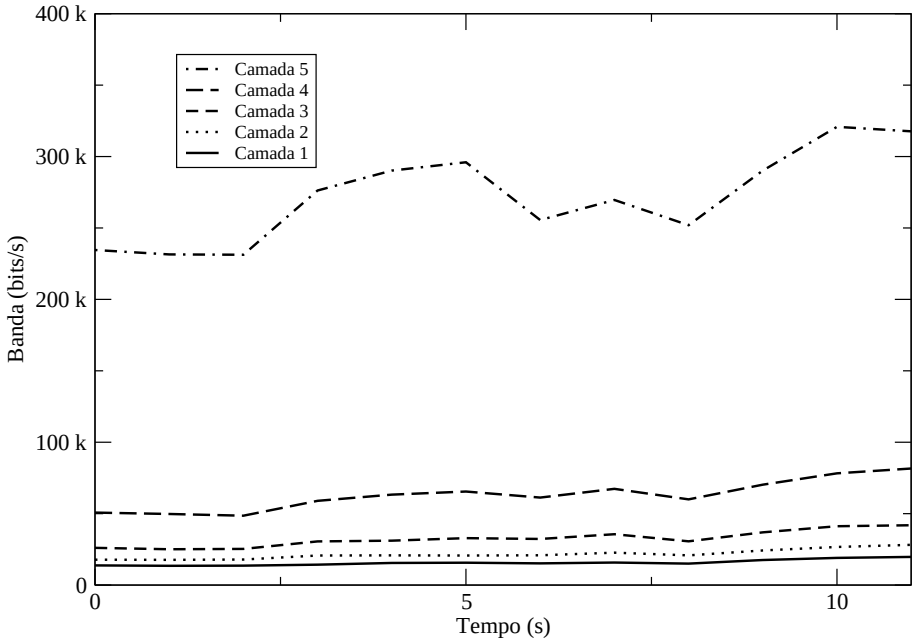

FIGURA 6.5 – Banda por Camada do vídeo forest

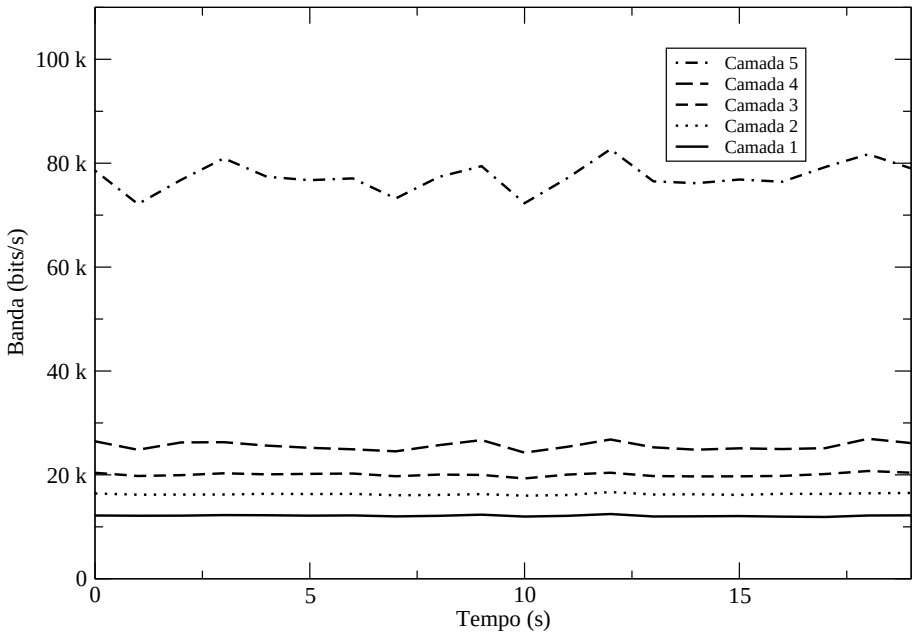

FIGURA 6.6 – Banda por Camada do vídeo claire

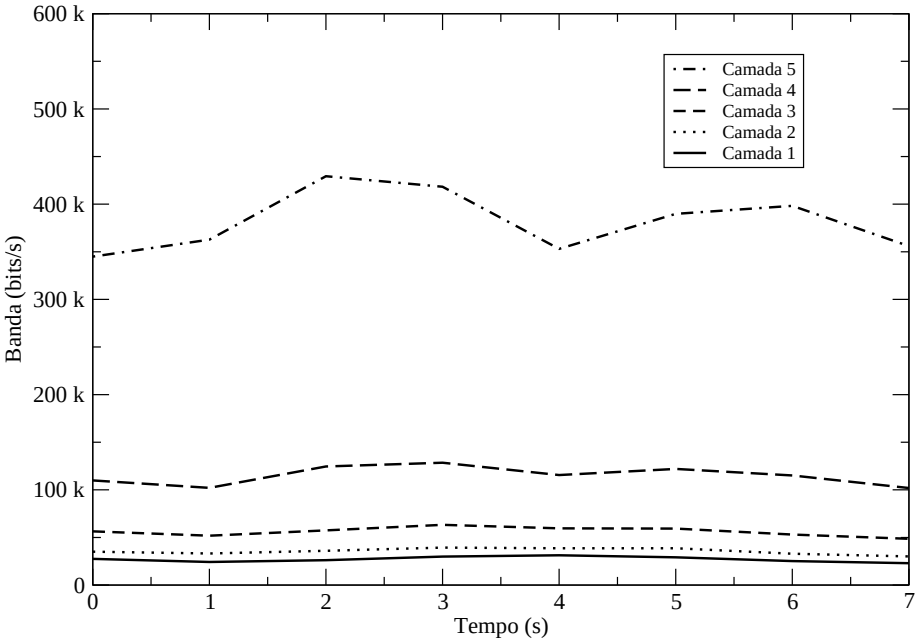

FIGURA 6.7 – Banda por Camada do vídeo night

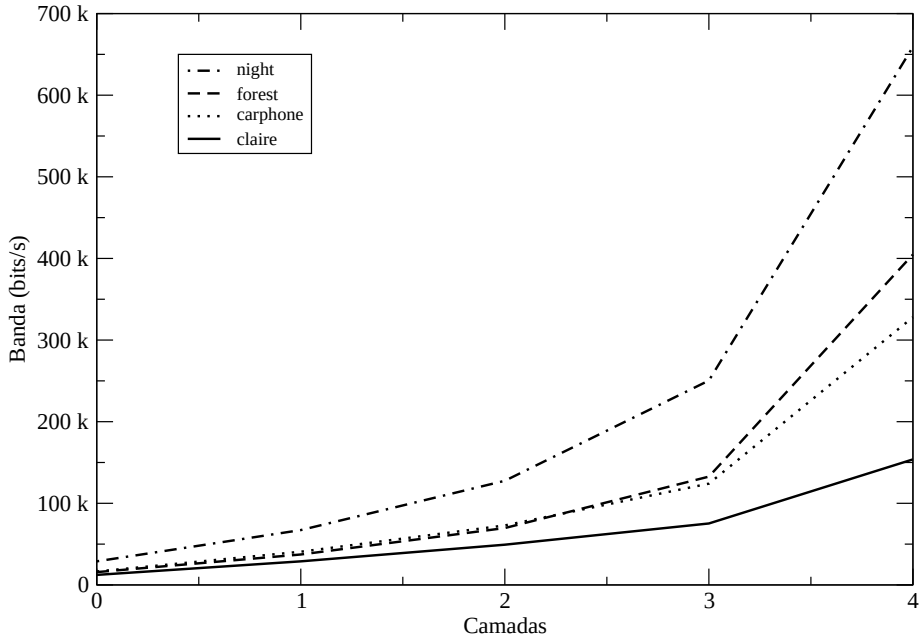

FIGURA 6.8 – Comparação da Banda por Camada entre os Vídeos

as camadas adicionais são apresentadas de 1 a 4.

Percebe-se que, com exceção do vídeo "night", todos os vídeos apresentam pelo menos duas camadas abaixo do índice de 50 kbit/s, sendo que o vídeo "claire" apresenta três camadas e o vídeo "night" apresenta apenas a camada base.

6.2.4 Qualidade

Os testes de qualidades foram realizados através de uma comparação entre os quadros gerados pelo decodificador e os quadros originais. Cada quadro decodificado foi comparado com seu respectivo quadro original. O método utilizado para comparação entre os dois quadros é o cálculo PSNR (*Peak Signal to Noise Ratio*). A formula 6.1 apresenta a equação utilizada, onde N_l e N_c são os números de linhas e colunas, respectivamente, de cada quadro, $p(y,x)$ é um pixel do quadro original e $p'(y,x)$ é o pixel no quadro decodificado na mesma posição dada pelo pixel do quadro original.

$$PSNR = 10log_{10}\frac{255^2}{\frac{1}{N_l}\frac{1}{N_c}\sum_{y=0}^{N_l-1}\sum_{x=0}^{N_c-1}[p'(y,x)-p(y,x)]^2} \tag{6.1}$$

O valor resultante é a relação existente entre o sinal do quadro original e o sinal do quadro decodificado, em dB. Quanto mais semelhantes os dois sinais forem, maior será este valor. Quanto mais distircido for o sinal da imagem decodificada, maior será a diferença entre os dois sinais e menor será o valor resultante.

Um cálculo foi executado para identificar a quantidade de ruído inserida pela imprecisão do cálculo da DCT. Uma matriz de dados aleatórios foi informada como entrada para a 3D-FDCT desenvolvida. A matriz resultante foi inversamente transformada com a 3D-IDCT e calculada a relação entre as duas matrizes através do PSNR. A matriz resultante apresenta uma relação de 50.675 dB em relação a ma-

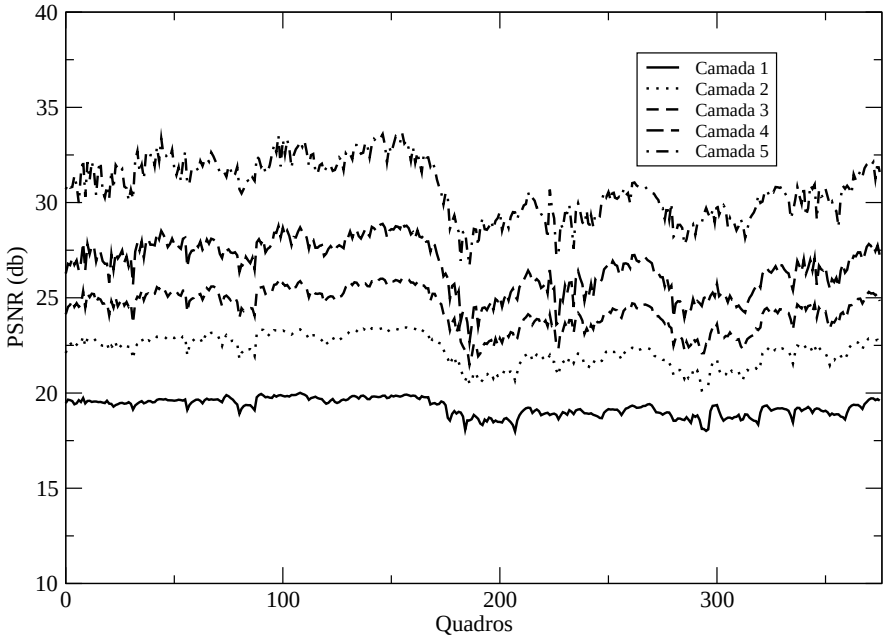

FIGURA 6.9 – PSNR do vídeo carphone

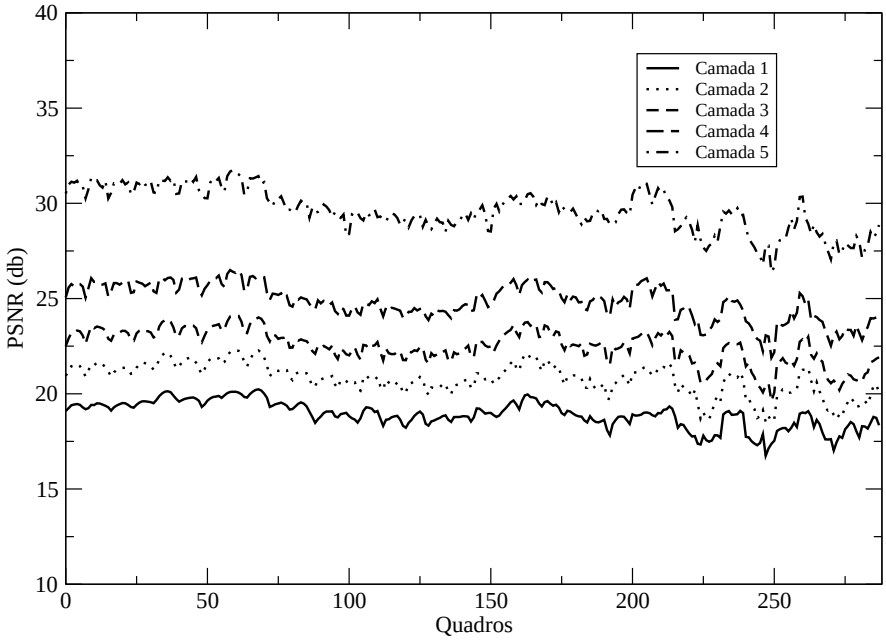

FIGURA 6.10 – PSNR do vídeo forest

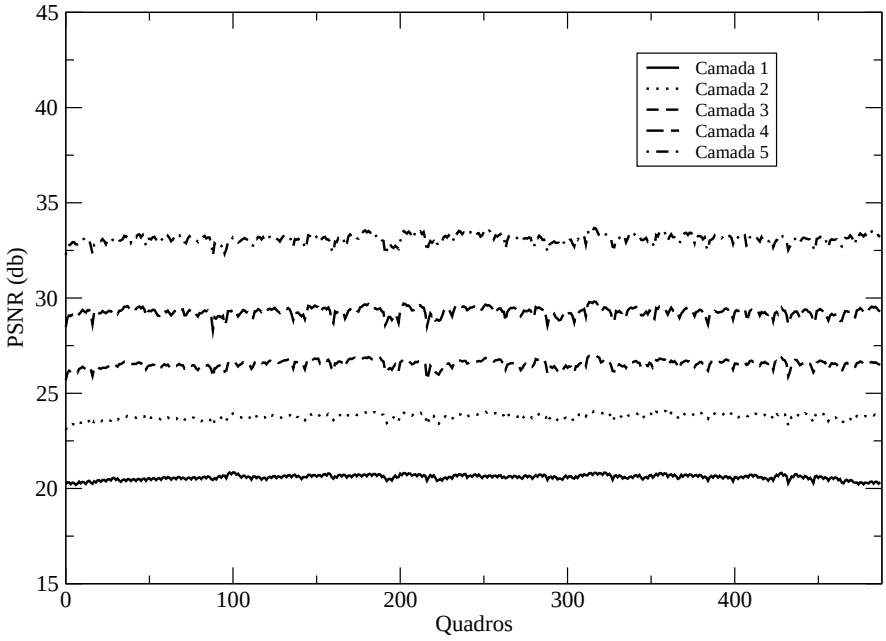

FIGURA 6.11 – PSNR do vídeo claire

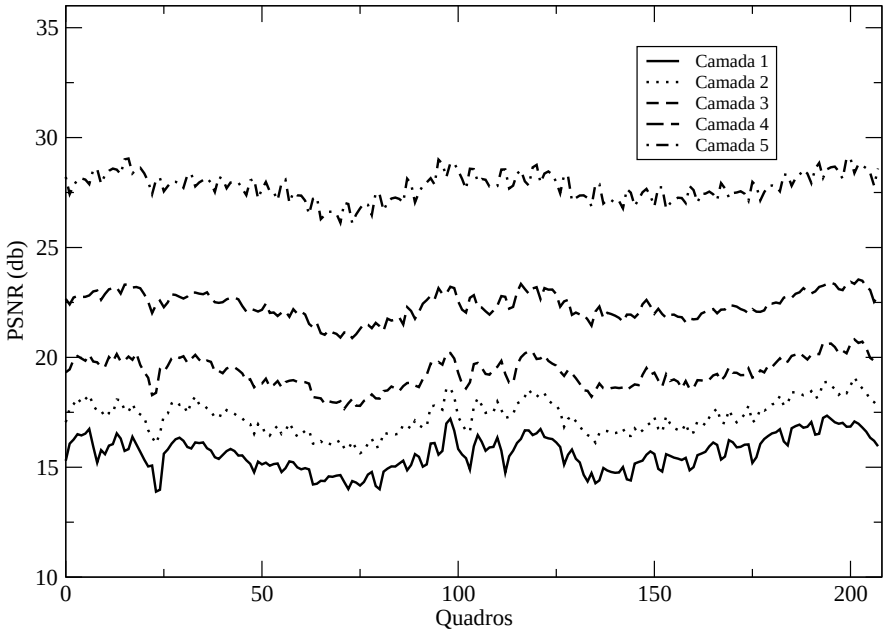

FIGURA 6.12 – PSNR do vídeo night

triz original. Este valor é considerado um erro baixo, mas, em relação às outras implementações, este valor ainda pode ser melhorado.

O vídeo 6.9 apresenta a variação de qualidade do vídeo "carphone". Percebe-se que, em torno do quadro 190 em diante, acontece uma grande perda na qualidade dos quadros. A perda é causada pela introdução de um movimento mais brusco na cena. Esta perda tende a se recuperar mais ao final, quando o movimento na janela do carro se apresenta mais estável.

O gráfico 6.10 apresenta os resultados do cálculo de qualidade para o vídeo "forest". Em torno do quadro 200 até o final do vídeo percebe-se uma grande variação na qualidade. Isso se dá devido a movimentação causada por um objeto. Diferente do vídeo "carphone", que possui apenas uma região da tela em alteração, o vídeo "forest" tem a movimentação de um objeto, no caso uma pessoa durante a acrobacia, da direita para a esquerda do vídeo. O efeito na qualidade final dos quadros é uma variação em torno de 5 dB.

O vídeo "claire", gráfico 6.11, assim como os resultados de distribuição de banda, apresentou baixa variação na qualidade dos quadros. Percebe-se, também, que em baixa movimentação é possível se atingir, além de altas taxas de compactação, uma maior qualidade nos quadros finais.

O vídeo "night", gráfico 6.12, apresenta o resultado da freqüente movimentação nos quadros: baixa qualidade e alta variação no sinal de saída. Está variação esta mais localizada nas camadas inferiores. A qualidade dos quadros de saída tende a se tornar mais constante a medida que aumenta-se o número de camadas.

7 Conclusão

Transmissão de vídeo é um importante componente em aplicações atuais tais como ensino a distância, divulgação de filmes, entre outras. Entretanto, a estratégia denominada de "melhor esforço" utilizada na Internet não é suficiente para garantir que uma transmissão de vídeo seja sempre compreensível pelo espectador. A infra-estrutura de rede disponível neste contexto é muito heterogênea, podendo variar desde ambientes de baixa velocidade com modems que permitem um acesso máximo de 56 kbit/s, até redes ADSL e *cable modems* ou redes locais. Isto sem contar com os diferentes periféricos de materialização de vídeo.

Na busca de transmissão em tempo real em ambientes heterogêneos, foi idealizado o projeto SAM. Dentro deste projeto, se mostrou urgente a necessidade de definição de um algoritmo e implementação de um protótipo capaz de codificar vídeos de forma escalável e transmitir em camadas em função da banda presente disponível e periférico de materialização. Esta dissertação buscou desenvolver um algoritmo de codificação de vídeo escalável. O sistema de transmissão será feito em conjunto com a tese em desenvolvimento por Valter Roesler, que propõe a criação de um algoritmo de controle de congestionamento denominado ALM.

No desenvolvimento do modelo de codificação, foi empregado a transformada 3D-DCT devido a sua simplicidade de desenvolvimento e a facilidade quanto a manipulação dos dados junto ao algoritmo de escalabilidade empregado. O algoritmo de escalabilidade é baseado em *bit-planes*, inspirado no modelo empregado pelo MPEG 4. Dos algoritmos estudados, ele foi escolhido por ser o que apresenta melhores resultados e por permitir uma granularidade fina das camadas.

A implementação foi feita em linguagem "C" e testada em ambiente Linux pelo fato de se possuir uma grande experiência neste sistema operacional e pela grande disponibilidade de bibliotecas de programação em domínio público. O desenvolvimento foi feito de modo a permitir a portabilidade para outros sistemas operacionais através do uso de bibliotecas portáveis, como SDL.

Os testes realizados comprovam que o protótipo é flexível para diferentes recursos de banda, podendo adaptar-se através do recurso de escalabilidade.

Quanto ao desempenho do protótipo, foi possível medir através de consecutivas execuções das rotinas de codificação e decodificação tendo como entrada dados aleatórios, conforme apresentado na seção 6.2. Foram realizados testes específicos com vídeos no formato QCIF, mostrando a viabilidade do protótipo para codificação e decodificação em tempo real. Analisando os resultados se descobriu que o gargalo do mesmo se encontra nas rotinas de DCT e quantização. Isto significa que, para se conseguir melhores resultados, é necessário melhorar o desempenho das referidas rotinas.

A distribuição da banda nas camadas também apresentou resultados favoráveis. Os testes realizados foram feitos com a distribuição dos dados em cinco camadas. Todos os vídeos apresentaram capacidade de transmissão em redes de baixa velocidade para duas ou três camadas. As taxas de transmissão apresentadas possuem pouca variação no tempo, o que torna mais fácil para um algoritmo de controle de congestionamento, como o ALM, atingir a estabilidade na rede.

Apesar de ser possível a transmissão em baixas taxas de bits, compatíveis com modems de 56 kbit/s, a qualidade na primeira camada ainda é muito inferior para

a correta compreensão do usuário. Os testes realizados mostram uma qualidade aceitável do vídeo a partir da terceira camada. Uma exceção a este fato se da em vídeos de baixa movimentação, como videoconferências (*Talking-Head*). A maior taxa de compressão dos dados atingida neste tipo de vídeo permite que uma banda de 56 kbits/s receba duas a três camadas. A qualidade destas camadas também é melhor devido a baixa movimentação, o que torna o modelo proposto viável para uma aplicação do tipo tele-aula, um dos objetivos do projeto SAM.

7.1 Trabalhos Futuros

Apesar do modelo atual ser aceitável, muitos pontos ainda devem ser melhor estudados. A implementação atual possui escalabilidade máxima de cinco camadas. Esta limitação está no fato de que as estatísticas geradas para compressão foram todas realizadas levando em conta apenas cinco camadas. O modelo proposto permite a utilização de camadas dinâmicas, desde que a distribuição dos bits dos coeficientes AC e DC seja melhor calculada. Um exemplo de distribuição pode ser visto em [SMO 96].

A implementação foi gerada tendo em vista usuários de baixa capacidade de banda, como modems, cable modem, ISDN e ADSL. Diferentes características de banda podem ser exploradas, novamente levando-se em conta a distribuição dos dados nos *bit-planes*.

A qualidade final do vídeo deve ser melhorada, principalmente nas camadas inferiores. O ideal seria atingir uma maior qualidade com a mesma taxa de transmissão. Para isso, estão sendo estudados novos recursos de compressão. Grande parte da perda da qualidade está relacionado a movimentação existente no vídeo. É possível atingir maior qualidade em modelos baseados em 3D-DCT através da compressão apenas do resíduo de movimentação [SER 97]. É gerado um quadro de referência e os demais quadros a serem utilizados como entrada na transformada são uma diferença deste quadro inicial. Com este modelo, a camada base pode transportar o quadro de referência enquanto que as demais transportam os coeficientes transformados do resíduo do movimento. O efeito é que, na camada base, um vídeo de alta qualidade espacial mas baixa qualidade temporal é apresentado, enquanto que cada camada adiciona recursos temporais.

Está sendo estudada, também, a substituição do codificador principal por uma transformada tridimensional baseada em wavelet. A transformada é executada um número de interações suficiente para que a camada base possa transmitir as freqüências baixas em uma taxa de bits estipulada. As freqüências altas são, então, divididas em *bit-planes* e transmitidas nas camadas superiores [ORD 98]. Um dos problemas presente neste modelo é o custo computacional envolvido na transformada tridimensional de wavelet.

Uma API de programação esta sendo desenvolvida de modo a permitir a fácil integração da implementação realizada em outras aplicações. Uma delas é o *framework* do projeto SAM. Assim que estiver concluído, pretende-se realizar novos estudos em um ambiente real de transmissão escalável.

Anexo 1 Tabelas de Distribuição dos Coeficientes

TABELA A.1 – Distribuição do Coeficiente DC

	0	1	2	3	4	5	6	7	8	9	10
carphone	0.00	0.00	0.00	0.00	0.00	0.02	1.69	12.02	18.62	61.23	6.42
forest	0.00	0.00	0.00	0.00	0.05	2.01	9.35	27.78	21.49	38.15	1.16
claire	6.20	0.11	0.19	0.31	0.21	5.59	2.26	1.98	9.88	71.24	2.03
night	0.00	0.00	0.00	0.00	0.00	0.00	0.59	20.85	24.51	48.75	5.30
mobile	0.00	0.00	0.01	0.09	0.35	0.71	1.72	5.04	10.77	63.54	17.76
clouds	0.00	0.00	0.00	0.00	0.00	0.00	0.64	10.49	4.33	71.07	13.47
news	0.00	0.00	0.00	0.00	0.00	0.00	4.85	22.45	18.07	50.54	4.09
salesman	0.00	0.00	0.00	0.00	0.00	0.00	0.38	19.47	31.97	48.19	0.00
tempete	0.00	0.00	0.00	0.00	0.00	0.51	4.88	11.31	25.21	55.45	2.64
foreman	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	5.99	66.02	27.95

TABELA A.2 – Distribuição dos Valores do Cabeçalho

	0000	0100	0101	0110	1001	1011
carphone	42.65	37.76	1.59	0.04	17.94	0.02
forest	49.80	28.98	2.36	0.27	18.45	0.14
claire	48.79	31.63	3.12	0.31	15.99	0.16
night	40.72	28.20	8.30	1.81	20.07	0.91
mobile	33.50	44.92	2.05	0.03	19.49	0.01
clouds	69.39	16.62	1.33	0.00	12.66	0.00
news	50.81	31.26	1.89	0.06	15.94	0.03
salesman	50.11	32.49	0.30	0.00	17.10	0.00
tempete	44.98	37.52	0.46	0.00	17.04	0.00
foreman	46.79	35.61	1.33	0.06	16.17	0.03

TABELA A.3 – Distribuição dos Valores *Run-Lenght*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
carphone	11.31	3.40	2.93	3.21	4.69	6.57	11.63	1.99	1.47	0.57	0.83	0.77	1.40	37.78	11.45
forest	13.91	5.37	4.13	4.09	5.23	5.89	7.61	1.90	1.16	0.69	1.03	0.95	1.59	38.09	8.36
claire	16.75	5.37	3.77	4.07	5.18	5.71	13.33	1.57	1.49	0.49	0.54	0.52	0.77	16.76	23.68
night	9.93	3.51	2.86	3.34	4.72	6.13	6.40	1.53	0.77	0.56	0.91	1.08	1.77	50.65	5.85
mobile	16.99	6.85	5.93	6.27	7.33	7.72	8.03	2.69	1.48	0.82	0.91	0.97	1.41	26.26	6.32
clouds	10.08	3.71	3.03	3.50	5.35	7.76	13.01	1.66	0.78	0.46	0.88	0.89	1.89	29.01	17.97
news	19.08	6.57	5.03	4.39	4.95	5.77	12.84	1.64	1.56	0.45	0.60	0.58	0.90	19.48	16.15
salesman	20.98	8.16	5.38	5.12	6.29	7.23	11.49	1.96	1.11	0.47	0.40	0.39	0.80	11.66	18.55
tempete	14.29	5.49	4.36	4.45	5.86	7.32	10.12	2.31	1.43	0.64	0.93	0.82	1.55	32.21	8.21
foreman	10.42	3.51	3.11	3.91	5.80	8.27	9.20	2.64	1.16	0.64	0.88	0.93	1.80	38.90	8.82

TABELA A.4 – Distribuição dos Valores dos bits Válidos para bits=2

	0	1	2	3	4	5	6	7
carphone	0.00	0.00	29.63	31.45	11.98	12.34	6.80	7.80
forest	0.00	0.00	30.57	32.30	11.80	12.28	6.45	6.61
claire	0.00	0.00	26.33	29.56	12.38	14.92	7.39	9.43
night	0.00	0.00	29.33	30.37	12.53	12.79	7.48	7.51
mobile	0.00	0.00	29.50	30.51	12.62	12.95	7.12	7.30
clouds	0.00	0.00	29.28	31.11	13.79	11.98	7.31	6.53
news	0.00	0.00	26.85	28.29	13.31	14.39	8.00	9.17
salesman	0.00	0.00	28.60	29.99	12.13	13.58	7.23	8.47
tempete	0.00	0.00	30.13	31.42	12.22	12.62	6.70	6.91
foreman	0.00	0.00	30.33	31.93	11.81	12.48	6.47	6.98

Anexo 2 Tabelas Huffman

TABELA B.1 – Tabela Huffman para Coeficiente DC

Valor	Código
9	0
8	100
7	101
10	11000
6	11001
5	11010
0	11011
4	11100
3	11101
2	11110
1	11111

TABELA B.2 – Tabela Huffman para o Cabeçalho

Valor	Código
0	0
4	100
9	101
5	1100
6	1101
10	11100
7	11110
11	11101

TABELA B.3 – Tabela Huffman para *Run-Lenght*

Valor	Código
14	0
1	1000
7	1001
3	11000
4	11001
8	11010
9	11011
6	10100
2	10101
5	10110
15	10111
13	11100
12	11101
11	11110
10	11111

TABELA B.4 – Tabela Huffman para Valores de bits Válidos para bits=2

Valor	Código
3	00
2	01
5	100
4	101
7	110
6	111

Bibliografia

[AHM 74] AHMED, N.; NATARAJAN, T.; RAO, K. R. Discrete cosine transform. **IEEE Transactions of Computers**, New York, v.C-23, p.90–93, 1974.

[ARY 93] ARYA, S.; MOUNT, D. M. Algorithms for fast vector quantization. In: DATA COMPRESSION CONFERENCE, 1993, Utah, EUA. **Proceedings...** New Jersey: IEEE, 1993. p.381–390.

[BAJ 98] BAJAJ, S.; BRESLAU, L.; SHENKER, S. Uniform versus priority dropping for layered video. **ACM Computer Communication Review**, New York, EUA, v.28, n.6, p.131–143, Oct. 1998. Trabalho apresentado na ACM SIGCOMM Conference, 1998, Vancouver.

[BUR 83] BURT, P. J.; ADELSON, E. H. The laplacian pyramid as a compact image code. **IEEE Transactions on Communications**, New York, v.31, n.4, p.532–540, 1983.

[CHA 90] CHAN, M. H.; YU, Y. B.; CONSTANTINIDES, A. G. Variable size block matching motion compensation with applications to video coding. **Proceedings of the IEEE**, New York, v.137, n.4, Aug. 1990.

[CON 2001] CONKLIN, G. et al. Video coding for streaming media delivery on the internet. **IEEE Transactions on Circuits and Systems for Video Technology**, New York, v.11, n.3, p.269–280, Mar. 2001.

[CON 99] CONKLIN, G.; HENAMI, S. A comparison of temporal scalability techniques. **IEEE Transactions on Circuits and Systems for Video Technology**, New York, v.9, n.6, p.909–919, Sept. 1999.

[COS 93] COSMAN, P. C. et al. Using vector quantization for image processing. **Proceedings of the IEEE**, New York, v.81, n.9, p.1326–1341, Sept. 1993.

[DAU 88] DAUBECHIES, I. Orthonormal bases of compactly supported wavelets. **Communications on Pure and Applied Mathematics**, New York, v.41, p.909–996, 1988.

[DEE 93] DEERING, S. **Internet multicast routing**: state of art and open research issues. Estocolmo, Suécia: [s.n.], 1993. Multimedia Integrated Conferencing for Europe (MICE), Seminário no Swedish Institute of Computer Science.

[FOG 2002] FOGG, C. **The mpeg-faq version 3.2**: mpeg-2. Disponível em: <http://bmrc.berkeley.edu/projects/mpeg/faq/mpeg2-v38>. Acesso em: maio 2002.

[FUR 99] FURHT, B.; WESTWATER, R.; ICE, J. Multimedia broadcasting over the internet: part ii-video compression. **IEEE Multimedia**, New York, v.6, n.1, p.85–89, Jan. 1999.

[GAL 78] GALLAGER, R. G. Variations on a theme by huffman. **IEEE Transactions of Information Theory**, New York, v.24, n.6, p.668–674, 1978.

[GOM 2000] GOMES, R. L. et al. Suporte à apresentação adaptativa de aplicações multimídia em sistemas distribuídos. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, SBRC, 18., 2000, Belo Horizonte. **Anais...** Belo Horizonte: Universidade Federal de Minas Gerais, 2000. p.489–504.

[GRA 84] GRAY, R. M. Vector quantization. **IEEE Transactions on Acoustics Speech and Signal Processing**, New York, v.1, n.2, p.4–29, Apr. 1984.

[GRA 98] GRAY, R. M.; NEUHOFF, D. L. Quantization. **IEEE Transactions on Information Theory**, New York, v.44, n.6, Oct. 1998.

[MPE 93] ISO/IEC. **11172-2**: Coding of moving pictures and associated audio for digital storage media at up to 1.5 Mbit/s, Part 2 - Video, MPEG-1 Standard. Switzerland, 1993.

[MPE 94] ISO/IEC. **13818-2**: Generic Coding of Moving Pictures and Associated Audio – Recommendation H.262. Switzerland, 1994.

[MJP 94] ITU-R. **BT.601-4**: Encoding parameters of digital television for studios. Switzerland, 1994.

[JAC 88] JACOBSON, V. Congestion avoidance and control. **ACM Computer Communication Review**, New York, v.18, n.4, p.314–329, Aug. 1988. Trabalho apresentado na ACM SIGCOMM Conference, 1988, Standford.

[KAN 93] KANAKIA, H.; MISHRA, P.; REIBMAN, A. An adaptive congestion control schene for real-time packet transport. **ACM Computer Communication Review**, New York, v.23, n.4, p.20–31, Sept. 1993. Trabalho apresentado na ACM SIGCOMM Conference, 1993, San Francisco.

[KUO 98] KUO, F.; EFFELSBERG, W.; GARCIA-LUNA-ACEVES, J. J. **Multimedia communications**: protocols and applications. New Jersey, EUA: Prentice Hall, 1998. 243p.

[LEG 2000] LEGOUT, A.; BIERSACK, E. Plm: fast convergence for cumulative layered multicast transmission schemes. **ACM Performance and Evaluation Review**, New York, v.28, n.1, p.13–22, June 2000. Trabalho apresentado na ACM SIGMETRICS Conference, 2000, Santa Clara.

[LEG 99] LEGOUT, A.; BIERSACK, E. W. **Beyond tcp-friendliness**: a new paradigm for end-to-end congestion control. France: EURECOM, 1999. Relatório Técnico. Disponivel em: <http://

www.eurecom.fr /btroup /FormerPages /legout_htdoc /Research /np.ps>. Acesso em: abr. 2003.

[LI 2001] LI, W. Overview of fine granularity scalability in mpeg-4 video standard. **IEEE Transactions on Circuits and Systems for Video Technology**, New York, v.11, n.3, p.301–316, Mar. 2001.

[LI 99] LI, X.; AMMAR, M.; PAUL, S. Video multicast over the internet. **IEEE Network Magazine**, New York, v.13, n.2, p.46–60, Apr. 1999.

[LIN 80] LINDE, Y.; BUZO, A.; GRAY, R. An algorithm for vector quantizer design. **IEEE Transactions on Communications**, New York, v.28, n.1, p.84–95, Jan. 1980.

[LIU 95] LIU, L. Y. et al. Huffman coding. **IEEE Proc. Computer Digital Technology**, New York, v.142, n.6, p.411–418, Nov. 1995.

[LOE 89] LOEFFLER, C.; LIGTENBERG, A.; MOSCHYTZ, C. S. Practical fast 1d dct algorithm with eleven multiplications. In: INTERNATIONAL CONFERENCE OF ACOUSTICS SPEECH AND SIGNAL PROCESSING, 1989, Glasgow, Escócia. **Proceedings...** New York: IEEE, 1989. p.988–991.

[MCC 96] MCCANNE, S.; JACOBSON, V.; VETTERLI, M. Receiver-driven layered multicast. **ACM Computer Communication Review**, New York, v.26, n.4, p.117–130, Aug. 1996. Trabalho apresentado na ACM SIGCOMM Conference, 1996, Stanford.

[MCC 97] MCCANNE, S.; VETTERLI, M.; JACOBSON, V. Low-complexity video coding for receiver-driven layered multicast. **IEEE Journal on Selected Areas in Communications**, New York, v.16, n.6, p.983–1001, Aug. 1997.

[MIT 97] MITCHELL, J. L. et al. **Mpeg video compression standard**. New York, EUA: Chapman & Hall, 1997. 470p.

[NAS 88] NASRABADI, N. M.; KING, R. A. Image coding using vector quantization: a review. **IEEE Transactions on Communications**, New York, v.36, n.8, p.957–971, Aug. 1988.

[ORD 98] ORDENTLICH, E.; WEINBERGER, M. J.; SEROUSSI, G. A low-complexity modeling approach for embedded coding of wavelet coefficients. In: DATA COMPRESSION CONFERENCE, 1998, Utah, EUA. **Proceedings...** New York: IEEE, 1998. p.408–417.

[PAR 93] PAREKH, A. K.; GALLAGER, R. G. A generalized processor sharing approach to flow control in integrated services networks. In: INFOCOM, 1993, São Francisco, EUA. **Proceedings...** New York: IEEE, 1993. p.521–530.

[PAR 99] PARHI, K. K.; NISHITANI, T. **Digital signal processing for multimedia systems**. New York, EUA: Marcel Dekker, 1999. 860p.

[POY 96] POYNTON, C. **A technical introduction to digital video**. New York, EUA: John Wiley & Sons, 1996.

[PUR 93] PURI, A.; WONG, A. Spatial domain resolution scalable video coding. In: CONFERENCE OF VISUAL COMMUNICATIONS AND IMAGE PROCESSING, 1993, Boston, EUA. **Proceedings...** New York: SPIE, 1993. v.2094, p.718–729.

[RIB 97] RIBAS-CORBERA, J.; NEUHOFF, D. L. On the optimal block size for block-based, motion compensated video coders. **SPIE Proceedings of Visual Communications and Image Processing**, New York, v.3024, p.1132–1143, Feb. 1997.

[RIO 91] RIOUL, O.; VETTERLI, M. Wavelets and signal processing. **IEEE Signal Processing Magazine**, New York, v.8, n.4, p.14–38, Oct. 1991.

[ROC 90] ROCK, I. **The perceptual world**: readings from scientific american magazine. New York, EUA: W. H. Freeman and Company, 1990. 200p.

[ROE 2001] ROESLER, V.; BRUNO, G.; LIMA, V. Alm - adaptive layering multicast. In: SIMPÓSIO BRASILEIRO DE SISTEMAS MULTIMÍDIA E HIPERMÍDIA - SBMIDIA, 7., 2001, Florianópolis, Brasil. **Anais...** Florianópolis: Universidade Federal de Santa Catarina, 2001. p.107–121.

[ROE 2002] ROESLER, V.; BRUNO, G.; LIMA, V. Análise de estabilidade em um algoritmo para controle de congestionamento de transmissões multimídia em camadas. In: SIMPÓSIO BRASILEIRO DE SISTEMAS MULTIMÍDIA E HIPERMÍDIA - SBMIDIA, 8., 2002, Fortaleza, Brasil. **Anais...** Fortaleza: Universidade Federal do Ceará, 2002.

[ROE 2003] ROESLER, V.; BRUNO, G.; LIMA, V. A new receiver adaptation method for congestion control in layered multicast transmissions. In: INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, ICT, 2003, French Polinese. **Proceedings...** New York: IEEE, 2003.

[ROE 2002a] ROESLER, V. et al. Uma ferramenta adaptativa para transmissão e recepção de sinais multimídia ao vivo. In: SIMPÓSIO BRASILEIRO DE SISTEMAS MULTIMÍDIA E HIPERMÍDIA - WORKSHOP DE FERRAMENTAS E APLICAÇÕES, 8., 2002, Fortaleza, Brasil. **Anais...** Fortaleza: Universidade Federal do Ceará, 2002.

[SER 97] SERVAIS, M.; JAGER, G. de. Video compression using the three dimensional discrete cosine transform (3d-dct). In: SOUTH AFRICAN

SYMPOSIUM ON COMMUNICATIONS AND SIGNAL, COMSIG, 1997, Grahamstown, South Africa. **Proceedings...** New York: IEEE, 1997. p.27–32.

[SEZ 93] SEZAN, M. I.; LAGENDIJK, R. L. **Motion analysis and image sequence processing**. Boston, EUA: Kluwer Academic, 1993. 489p.

[SHA 48] SHANNON, C. E. A mathematical theory of communication. **Bell System Technical Journal**, New York, v.27, p.379–423, July 1948.

[SHC 92] SHACHAM, N. Multipoint communication by hierarchically encoded data. In: INFOCOM, 1992, Florença, Italia. **Proceedings...** New York: IEEE, 1992. p.2107–2114.

[SHE 99] SHEN, K.; DELP, E. Wavelet based rate scalable video compression. **IEEE Transactions Circuits System and Video Tecnology**, New York, v.9, n.1, p.109–122, Feb. 1999.

[SKO 2001] SKODRAS, A.; CHRISTOPOULOS, C.; EBRAHIMI, T. The jpeg 2000 still image compression standard. **IEEE Signal Processing Magazine**, New York, v.18, n.5, p.36–57, Sept. 2001.

[SMO 96] SMOOT, S.; ROWE, L. Study of dct coefficient distributions. In: SYMPOSIUM ON ELECTRONIC IMAGING, 1996, San Jose, EUA. **Proceedings...** New York: SPIE, 1996. v.2657.

[TAU 94] TAUBMAN, D.; ZAKHOR, A. Multirate 3-d subband coding of video. **IEEE Transactions on Image Processing**, New York, v.3, n.5, p.572–588, Sept. 1994.

[USE 2001] USEVITCH, B. E. A tutorial on modern lossy wavelet image compression: fundations of jpeg 2000. **IEEE Signal Processing Magazine**, New York, v.18, n.5, p.22–35, Sept. 2001.

[VER 2001] VETRO, A.; SUN, H. Media conversions to support mobile users. In: CCECE, 2001, Ontario, Canada. **Proceedings...** New York: IEEE, 2001. p.607–613.

[VET 92] VETTERLI, M.; HERLEY, C. Wavelets and filter banks: theory and design. **IEEE Transactions on Signal Processing**, New York, v.40, n.9, p.2207–2232, Sept. 1992.

[VIC 98] VICISANO, L.; CROWFORD, J.; RIZZO, L. Tcp like congestion control for layered multicast data transfer. In: INFOCOM, 1998, San Francisco, EUA. **Proceedings...** New York: IEEE, 1998. p.996–1003.

[VIT 87] VITTER, J. S. Design and analysis of dynamic huffman codes. **Jornal of ACM**, New York, v.34, n.4, p.825–845, 1987.

[WIL 97] WILSON, D.; GHANBARI, M. Optimization of two-layer snr scalability for mpeg-2 video. In: INTERNATIONAL CONFERENCE OF

ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 1997, Monique, German. **Proceedings...** New York: IEEE, 1997. p.2637–2640.

[WIT 87] WITTEN, I. H.; NEAL, R. M.; CLEARY, J. G. Arithmetic coding for data compression. **Communications of the ACM**, New York, v.30, n.6, p.520–540, June 1987.

[WU 2001] WU, D. et al. Streaming video over the internet: approaches and directions. **IEEE Transactions on Circuits and Systems for Video Technology**, New York, v.11, n.3, p.282–300, Mar. 2001.

[YAN 2000] YANG, T. et al. **Temporal scalability in h.26l**. [S.l.]: ITU-T Video Coding Experts Group, 2000. <Disponível em: http://standard.pictel.com/video-site/0005 Osa/q15j45.doc>. Acesso em: jan. 2003.