Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e Computação
Departamento de Comunicações

Sistema de Codificação de Vídeo Baseado em Transformadas Tridimensionais, Rápidas e Progressivas

Autora:
Vanessa Testoni

Orientador:
Prof. Dr. Max Henrique Machado Costa

Co-orientador:
Prof. Dr. Leonardo de Souza Mendes

Dissertação apresentada à Faculdade de Engenharia Elétrica e Computação da Unicamp como parte dos requisitos exigidos para a obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA.

Campinas, fevereiro de 2007

Sistema de Codificação de Vídeo Baseado em Transformadas Tridimensionais, Rápidas e Progressivas

Autora:
Vanessa Testoni

Orientador:
Prof. Dr. Max Henrique Machado Costa

Co-orientador:
Prof. Dr. Leonardo de Souza Mendes

Dissertação apresentada à Faculdade de Engenharia Elétrica e Computação da Unicamp como parte dos requisitos exigidos para a obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA.

Banca Examinadora:

Prof. Dr. Max Henrique Machado Costa	FEEC/Unicamp
Prof. Dr. Marcus Vinicius Lamar	Dep. de Ciência da Computação/UnB
Prof. Dr. Dalton Soares Arantes	FEEC/Unicamp
Prof. Dr. Amauri Lopes	FEEC/Unicamp

Campinas, fevereiro de 2007

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

Testoni, Vanessa
T289s Sistema de codificação de vídeo baseado em transformadas tridimensionais, rápidas e progressivas. / Vanessa Testoni. --Campinas, SP: [s.n.], 2007.

Orientadores: Max Henrique Machado Costa, Leonardo de Souza Mendes
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Videodigital – Codificação. 2. Compressão de dados (Telecomunicações). 3. Teoria da codificação. 4. Videodigital. I. Costa, Max Henrique Machado. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Video coding system based on three dimensional, fast and progressive transforms
Palavras-chave em Inglês: Video coding, Three dimensional coding, Progressive coding, Hadamard transform
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Marcus Vinicius Lamar, Dalton Soares Arantes e Amauri Lopes
Data da defesa: 09/02/2007
Programa de Pós-Graduação: Engenharia Elétrica

A você que agora está lendo este trabalho e fazendo com que possa cumprir a sua real função, que não é a de apenas servir para o meu crescimento intelectual ou como prova do conhecimento científico por mim adquirido, mas a de contribuir efetivamente para a nossa sociedade, seja no incentivo à formação de novos pesquisadores, seja no desenvolvimento de novos projetos tecnológicos ou na simples disseminação do conhecimento e de novas idéias.

AGRADECIMENTOS

À todos que de alguma forma contribuíram para que este trabalho pudesse ser realizado, seja com palavras, atitudes ou meios materiais. Como considero estas contribuições igualmente importantes e sempre as recebi de muitas pessoas realmente especiais, prefiro não nomeá-las aqui, uma vez que certamente seria injusta ao não mencionar todas elas.

De forma geral, devo agradecer a inspiração e a possibilidade de realização deste trabalho aos professores que tive nos vários níveis de minha formação (especialmente ao meu orientador – Max Henrique Machado Costa – cuja orientação foi tão especial que já está tendo seqüência nos estudos de doutorado) e aos meus colegas de trabalho/universidades (entre os quais vários amigos). O entusiasmo, a dedicação, a inteligência, as idéias criativas, as discussões lógicas que geravam soluções inicialmente improváveis e as atitudes comportamentais de muitos de vocês me inspiram a continuar sempre buscando novos conhecimentos, me ensinaram muito, me fizeram e me fazem admirar cada vez mais este nosso mundo de cálculos e bits, que me conquistou e que eu pretendo tentar compreender melhor a cada dia.

Agradeço também à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro.

RESUMO

As pesquisas na área de codificação de vídeo buscam técnicas que alcancem taxas de compressão cada vez mais altas. O aumento da compressão é obtido ao custo do aumento da complexidade dos algoritmos de codificação, que é suportado pelo também constante aumento da capacidade dos processadores. Entretanto, em alguns cenários de codificação e transmissão de vídeo, a utilização destes processadores de alta capacidade não é possível ou desejada. Isso exige o desenvolvimento de codificadores de vídeo focados na obtenção de tempos de processamento reduzido e na utilização de poucos recursos computacionais, tais como o sistema de codificação apresentado neste trabalho. Para o desenvolvimento deste sistema foi utilizada a transformada de Hadamard tridimensional implementada de forma otimizada e um codificador adaptativo de Golomb por planos de bits que acrescenta ao sistema a desejável característica de ser progressivo. A implementação do sistema é adaptada para realizar somente operações matemáticas rápidas e alocar pouca memória computacional. Mesmo com a utilização destas técnicas focadas em rapidez, foram obtidos bons resultados experimentais em termos da razão de sinal de pico por ruído em função da taxa de bits por pixel.

ABSTRACT

The research on video coding systems has always been looking for techniques that can reach the highest possible compression rate. This compression rate increase is generally achieved by means of increased coding complexity, which is supported by the continuous increase verified in computational power. However, in some video coding and transmission situations, the use of high capacity processors is not possible or desirable. These situations require the development of video coders focused on the achievement of reduced execution times and on the requirement of few computational resources, just as the video coding system proposed in this dissertation. The proposed system uses three dimensional Hadamard transforms, implemented in an efficient way, and adaptive entropy coding with Golomb codes applied to bit planes, whichs adds to the system the desirable characteristic of being progressive. The computational system implementation is designed to perform only fast mathematical operations and to require small computational memory. Even with the use of these constrained techniques, good experimental results, in terms of peak signal to noise ratio (PSNR) versus pixel bit-rate were achieved.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1
INTRODUÇÃO

O objetivo deste trabalho é descrever um sistema computacional de codificação de vídeo desenvolvido especificamente para ser utilizado em transmissões de arquivos de vídeo em uma rede de fibra ótica. Devido à grande disponibilidade de banda na rede, ao contrário dos principais codificadores de vídeo, o foco deste sistema está na obtenção de tempo de processamento reduzido e não na obtenção de altas taxas de compressão.

Sendo assim, as redundâncias temporais entre os vários quadros dos arquivos de vídeo não foram exploradas com as eficientes, porém custosas, técnicas de estimação e compensação de movimento. No lugar destas técnicas, foram utilizados métodos de transformadas tridimensionais que dividem o arquivo de vídeo em blocos de quadros processados em conjunto.

Em busca de uma redução ainda maior no tempo de processamento, a transformada de Hadamard foi adotada no lugar da tradicional DCT (*Discrete Cosine Transform*). Embora não apresente a mesma capacidade de compactação de energia da DCT, a transformada de Hadamard foi escolhida devido ao seu baixo custo computacional, pois seu cálculo envolve apenas adições e subtrações.

Ainda em busca de rapidez de processamento, todas as operações de multiplicação e divisão do sistema foram aproximadas para serem realizadas por múltiplos de dois e implementadas com deslocamentos binários. Certamente, estas aproximações geram perdas de eficiência, mas o processamento torna-se muito mais rápido, uma vez que somente são realizadas operações matemáticas de adição, subtração e deslocamentos binários.

Outra característica adicionada ao sistema foi a capacidade de realizar codificação progressiva. Esta capacidade é importante porque os arquivos de vídeo transmitidos pela rede de fibra ótica podem chegar a diversos tipos de receptores, bem como atravessarem enlaces de maior ou menor capacidade. A codificação progressiva foi adicionada ao sistema através do uso de um codificador de entropia de Golomb adaptativo, onde cada plano de bits dos coeficientes da transformada é processado separadamente a partir do mais significativo [20].

Devido a todas as características descritas anteriormente, o sistema de codificação de vídeo descrito neste trabalho foi denominado FHVC (*Fast Hadamard Video Codec*) e será referenciado ao longo do trabalho através desta sigla.

A motivação inicial para o desenvolvimento do FHVC foi a necessidade da utilização de um codificador de vídeo de baixa complexidade e bom desempenho em um projeto idealizado pelo LARCOM - Laboratório de Redes de Comunicações da Faculdade de Engenharia Elétrica e de Computação da Unicamp. O projeto visa a construção de um equipamento denominado *Set-Top Box Universal* para servir de interface entre uma rede de fibra ótica e seus usuários. O equipamento recebe sinais digitais, extrai as informações de vídeo, áudio e dados e as envia a um dispositivo de saída. Entre outras funcionalidades, tais como conexão com a Internet e voz sobre IP, o *set-top box* está apto a receber e enviar sinais de vídeo provenientes, por exemplo, de aplicações de vídeo sob demanda ou de vídeo conferência. Estas funcionalidades originaram a necessidade de se adicionar o desenvolvimento de um codificador rápido de vídeo ao projeto. Este codificador seria inserido no protocolo SIP (*Session Initiation Protocol*) que, através do protocolo SDP (*Session Description Protocol*), possui mecanismos de extensibilidade que suportam a inserção de novos codificadores de áudio e vídeo [1]. O SIP é um protocolo padrão de sinal desenvolvido pela IETF (*Internet Engineering Task Force*) para estabelecer chamadas e conferências através de redes via IP. A configuração da sessão, mudança ou término é independente do tipo de mídia ou aplicação usada na chamada e cada chamada pode utilizar diferentes tipos de dados, incluindo áudio e vídeo. Já o SDP é um protocolo descritor de sessão para sessões multimídias abertas pelo protocolo SIP.

Devido às dificuldades encontradas pela equipe de construção do hardware do *set-top box*, o projeto foi suspenso temporariamente pelo LARCOM e o codificador de vídeo acabou sendo desenvolvido para ser executado em um ambiente computacional pessoal. Todas as características iniciais desejadas para o codificador foram mantidas e, para uma melhor execução neste novo ambiente, acrescentou-se a possibilidade de configuração e execução dos processos de codificação e decodificação de vídeo através de interfaces gráficas. Uma vez que o codificador foi desenvolvido como um sistema orientado a objetos, esta camada de entrada/saída realizada por interfaces gráficas, pode ser modularmente substituída por outra. Sendo assim, quando o projeto do *set-top box* for retomado, o codificador ainda poderá ser adaptado ao mesmo, apenas com a substituição da interface de entrada/saída pela definida para o hardware.

Organização do texto

Inicialmente, o Capítulo 2 descreve as várias técnicas utilizadas no FHVC separadas e ordenadas de acordo com as etapas do próprio processo de codificação de vídeo. São descritos os padrões de vídeo utilizados, a transformada de Hadamard tridimensional, a forma de ordenação dos coeficientes da transformada, a codificação de entropia adaptativa por Golomb e como o sistema implementa a codificação progressiva.

Em seguida, o Capítulo 3 contém toda a modelagem computacional do FHVC feita de acordo com as técnicas de Engenharia de *Software* e com a utilização de uma ferramenta CASE (*Computer-Aided Software Engineering*).

O Capítulo 4, por sua vez, apresenta os resultados obtidos para vários arquivos de vídeo comumente utilizados na literatura da área. Os resultados são expressos em termos de PSNR (*Peak Signal-to-Noise Ratio*) do arquivo de vídeo decodificado em função da taxa de bits por pixel do arquivo de vídeo codificado. Também são apresentados alguns quadros dos arquivos de vídeo decodificados para avaliação da qualidade visual obtida.

Finalmente, o Capítulo 5 contém as considerações finais e as sugestões de melhorias que podem ser realizadas em trabalhos futuros.

Capítulo 2
SISTEMA PROPOSTO

Supõe-se que o leitor já possui um conhecimento prévio sobre conceitos fundamentais de sinais de vídeo e sobre as várias técnicas de codificação dos mesmos. Sendo assim,optou-se pela não inclusão neste trabalho de uma extensa revisão bibliográfica destes assuntos e este capítulo já descreve detalhadamente somente as técnicas de codificação de vídeo utilizadas efetivamente no sistema de codificação de vídeo proposto.

2.1 INTRODUÇÃO

O sistema FHVC realiza um processo de codificação de vídeo com perdas, denominado de compressão. De forma geral, a codificação digital resulta na representação de um sinal de vídeo através de uma seqüência de bits. Quando o objetivo é utilizar o menor número de bits possível para representar o sinal de forma que quando reconstruído seja idêntico ao original, a codificação é denominada de compactação [10]. Quando são aceitáveis distorções na imagem reconstruída, o processo é denominado de compressão. Neste caso, as perdas na qualidade do sinal reconstruído são justificadas pelas taxas mais baixas de bits por pixel alcançadas.

A Fig. 2.1.1 abaixo apresenta a seqüência de execução das técnicas utilizadas em cada uma das etapas da codificação e da decodificação de vídeo implementadas no FHVC:

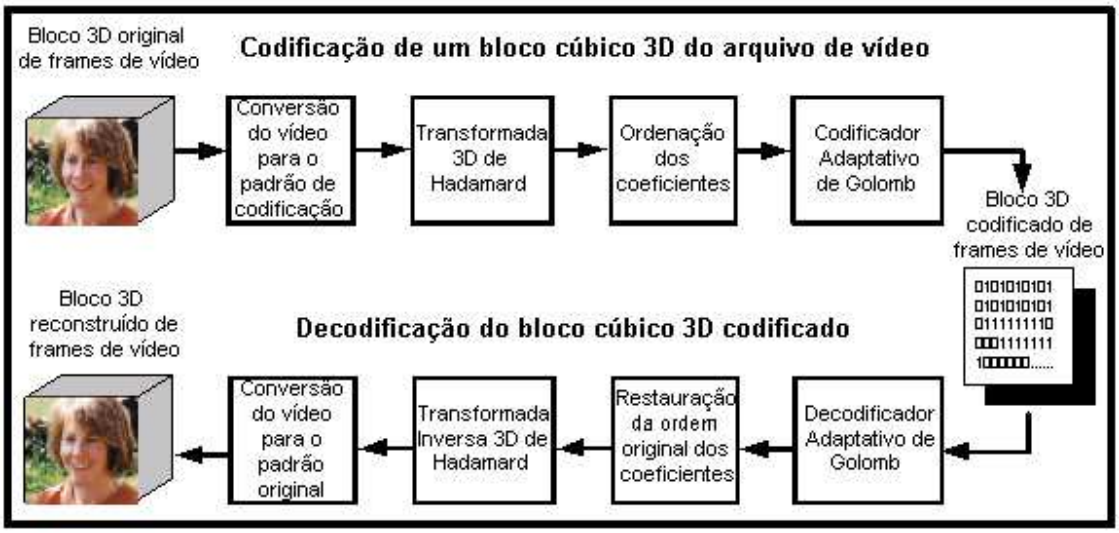


Fig. 2.1.1- Processo de codificação e decodificação de vídeo realizado pelo FHVC.

Cada uma das técnicas tem o objetivo básico de reduzir um determinado tipo de redundância encontrada em sinais de vídeo. A diminuição destas redundâncias é que resulta na compressão do sinal, pois este é representado com menor taxa de bits por pixel. A seguir, as redundâncias dos sinais de vídeo são relacionadas às técnicas utilizadas no FHVC para diminuí-las:

- **Redundância psico-visual:** alguns componentes dos sinais de vídeo são irrelevantes para o sistema visual humano. Sendo assim, com base nas características do olho humano, foram desenvolvidos padrões mais eficientes de representação de vídeo. A Seção 2.2 discute estas características e apresenta, entre outros, os padrões RGB e YUV 4:2:0 que são usados no FHVC. Vários outros padrões de espaços de cores podem ser encontrados em [8], [9], [45] e [46];

- **Redundância espacial**: a utilização da transformada de Hadamard, descrita na Seção 2.3.2, visa reduzir a redundância espacial existente entre pixels dentro do mesmo quadro, caracterizada pelas altas correlações entre pixels vizinhos. A transformada de Hadamard pode ser vista como uma simplificação da transformada discreta do co-seno (DCT), normalmente utilizada em codificações de imagens. Várias outras técnicas podem ser utilizadas para a redução da redundância espacial, tais como as transformadas Wavelet [11], a codificação em sub-bandas [10] e as várias transformadas espaciais [4], [10], entre elas a própria DCT, a transformada discreta de Fourier e a transformada ótima de *Karhunen-Loève*. Alguns trabalhos que exploram o uso da transformada de Hadamard na codificação de imagens e vídeo podem ser obtidos em [47], [48] e [49];

- **Redundância temporal**: a transformada de Hadamard foi implementada de forma tridimensional para reduzir também as correlações existentes nos valores dos pixels vizinhos entre quadros consecutivos. A Seção 2.3.3 descreve a transformada tridimensional de Hadamard implementada. Outros trabalhos que realizam codificação tridimensional de vídeo podem ser encontrados em [16], [17], [18], [19], [37], [39], [40] e [41];

- **Redundância entrópica**: o codificador de entropia adaptativo de Golomb explora o fato de que alguns valores de coeficientes ocorrem com mais freqüência que outros para obter uma forma mais eficiente da representação dos coeficientes da transformada. As características do codificador de entropia implementado no FHVC são detalhadas na Seção 2.5. Outros codificadores de entropia podem ser encontrados em [10], [11], [50] e [51]. Antes de passarem pelo codificador de entropia, os coeficientes são lidos de acordo com uma forma de ordenação especialmente definida para reduzir ainda mais a redundância estatística do sinal. O FHVC utiliza uma forma própria de ordenação dos coeficientes que está descrita na Seção 2.4.

2.2 SISTEMA VISUAL HUMANO E PADRÕES DE VÍDEO

Nesta seção são apresentados os aspectos do sistema visual humano que são comumente explorados nas técnicas de codificação para reduzir a quantidade de dados em seqüências de vídeo sem perdas significativas de qualidade do sinal.

2.2.1 Representações de cores

A representação de cores pode ser feita através de dois sistemas: o aditivo e o subtrativo. No sistema de cores aditivo, as cores são constituídas por formas de onda luminosas de diferentes comprimentos de onda que são adicionadas para a geração de novas cores. Neste sistema, inicia-se do preto (que corresponde a ausência de forma de onda) e combinam-se as cores primárias vermelho, verde e azul (*RGB - Red Green Blue*) para gerar as cores secundárias amarelo, ciano e magenta (*YCM - Yellow Cyan Magenta*), as quais também são sucessivamente combinadas até a geração da cor branca (resultante da combinação de todas as cores). Já no sistema de cores subtrativo, as cores resultam do processo de subtração de comprimentos de onda realizado por pigmentos que absorvem partes particulares do espectro. Neste sistema, a combinação de tipos diferentes de pigmentos resulta em formas de onda refletidas cujo espectro de comprimentos de onda é reduzido. Inicia-se, então, do branco (correspondente a nenhum pigmento, nenhuma forma de onda absorvida) e combinam-se as cores primárias (YCM no sistema subtrativo) para gerar as

cores secundárias (RGB) que são combinadas até a geração do preto (onde os pigmentos absorvem todas as cores e, portanto, nenhuma é refletida).

Nos dois sistemas de representação de cores, são utilizadas apenas três cores primárias. De forma geral, cada pixel presente em imagens coloridas também é caracterizado por apenas três componentes de cor. Esta representação surgiu da teoria tricromática da visão humana em cores [2] que afirma ser esta produzida pelas três classes de células cone do olho humano (os foto-receptores de cor). Uma descrição detalhada destas e de outras características do olho humano pode ser encontrada em [3]. Os foto-receptores de cor têm sensibilidades diferentes aos comprimentos de onda do espectro visível, mas são principalmente sensíveis aos comprimentos de onda correspondentes às cores vermelho, verde e azul (daí o surgimento do padrão RGB). A sensibilidade a cada uma destas cores também varia e é apresentada na Fig. 2.2.1.

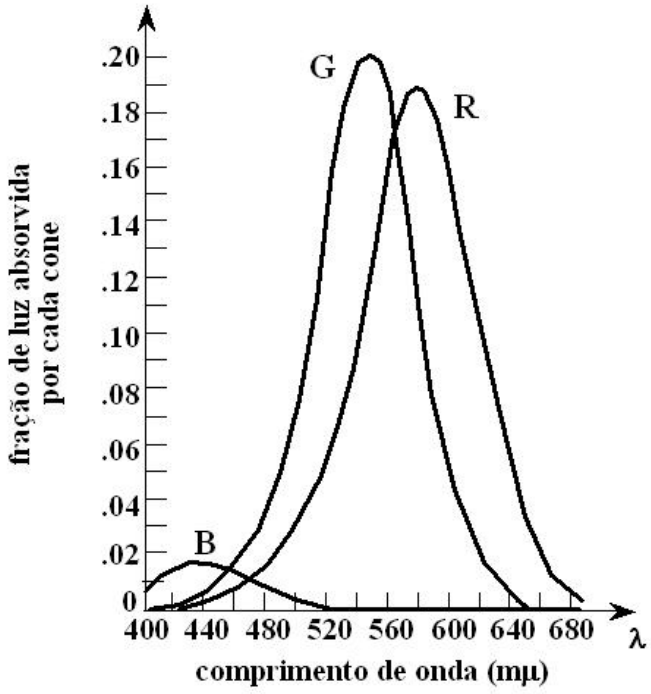


Fig. 2.2.1 - Sensibilidade do olho humano aos comprimentos de onda [36].

Observa-se que o pico da sensibilidade para a cor verde é aproximadamente 5% maior do que o pico para a cor vermelha e que estas sensibilidades são aproximadamente dez vezes maiores do que a sensibilidade para a cor azul. Estas diferenças de sensibilidade do olho humano interessam aos sistemas de compressão de imagens porque não há razão em se reproduzir

detalhes que não serão percebidos. Sendo assim, a utilização do padrão RGB em sistemas de compressão não é vantajosa, uma vez que representa as três componentes de cor com o mesmo número de bits.

Uma forma de se tirar vantagem destas diferenças de sensibilidade do olho humano é representar a imagem em termos de luminância e crominância. A luminância está relacionada à percepção de brilho da imagem e a crominância está associada à percepção de saturação e matiz das cores. As três componentes de cor RGB participam do cálculo da luminância Y [11] através da expressão

$$Y = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B\,, \tag{2.1}$$

em que são ponderadas de acordo com a sensibilidade do olho à cada cor primária. Sendo assim, o peso dado à componente de cor verde é maior do que o peso dado à componente de cor vermelha e ambos os pesos são maiores do que o atribuído à componente de cor azul. O valor destes pesos é definido na Recomendação BT.601 da ITU-R (*International Telecommunication Union - Radiocommunication Sector*) [44].

Para o cálculo da Eq. (2.1) considera-se que as componentes RGB foram normalizadas para uma escala entre 0 e 1, de forma que os tons de cinza são produzidos quando R=G=B. Sendo assim, o branco corresponde à R=G=B=1 e o preto corresponde à R=G=B=0. Como os fatores de ponderação utilizados para o cálculo da luminância somam 1, esta também será definida em uma escala entre 0 e 1 [4][5].

Embora seja possível caracterizar saturação e matiz diretamente, é mais conveniente do ponto de vista computacional, definir as componentes de crominância U e V como as seguintes diferenças de cores:

$$\begin{aligned} U &= B - Y; \\ V &= R - Y. \end{aligned} \tag{2.2}$$

Para eliminar os possíveis valores negativos das componentes U e V e também para defini-las em uma escala entre 0 e 1, estas são escalonadas e deslocadas, dando origem às componentes C_B e C_R definidas por:

$$\begin{aligned} C_B &= \left(\frac{U}{2}\right) + 0{,}5; \\ C_R &= \left(\frac{V}{1{,}6}\right) + 0{,}5. \end{aligned} \tag{2.3}$$

Para a representação usual de 8 bits por componente, faz-se a multiplicação dos valores Y, C_R e C_B por 255 [7]. O padrão YC_RC_B resultante desta multiplicação é o padrão geralmente utilizado em sistemas de vídeo. Entretanto, ficou erroneamente conhecido como YUV, sendo esta definição utilizada até em padrões de amostragem consagrados (tais como o YUV 4:2:0, que será explicado ainda nesta seção).

Além de representar melhor as cores utilizando ponderações, outra vantagem do padrão YC_RC_B é que as componentes de crominância podem ser amostradas por uma taxa espacial menor do que a utilizada para a componente de luminância, sem degradação perceptível do sinal. Isto pode ser feito porque o olho humano tem sensibilidades diferentes às componentes de luminância e crominância, que podem ser observadas na Fig. 2.2.2. Nesta figura, o contraste é definido como a amplitude pico-a-pico da variação senoidal da luminância dividida pela luminância média e a sensibilidade ao contraste é definida como o inverso do contraste para uma mudança perceptível [3]. Portanto, mudanças muito pequenas para serem percebidas estão localizadas acima das curvas e mudanças visíveis estão localizadas abaixo. As curvas mostram os limiares para luminância à crominância zero (tons de cinza) e para crominância à luminância constante.

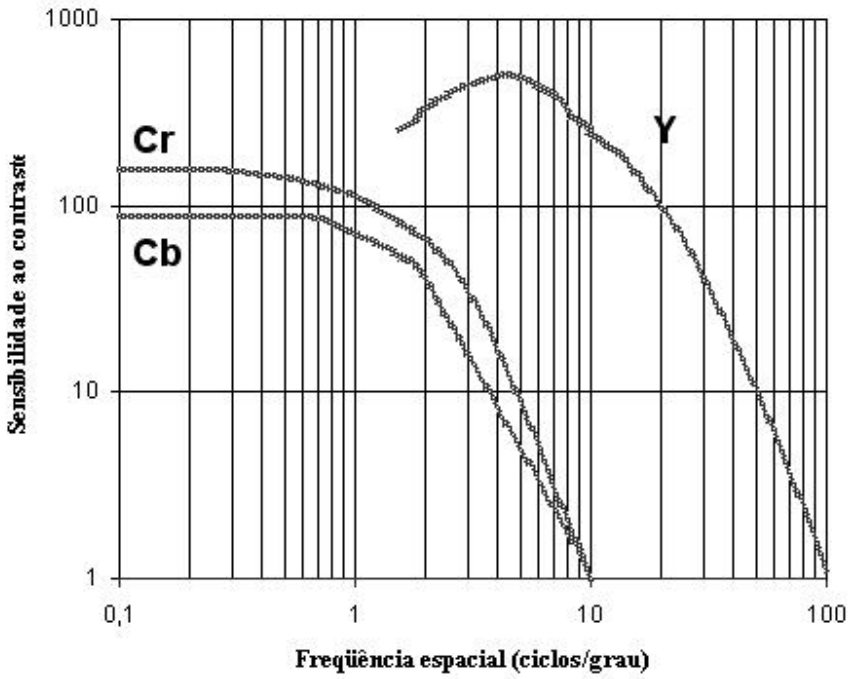


Fig. 2.2.2 – Sensibilidade do olho humano às componentes de luminância e crominância [3].

A Fig. 2.2.2 mostra que a maior sensibilidade do olho humano está na componente de luminância em baixas freqüências. Observa-se também que tanto a resposta à luminância quanto a resposta à crominância caem bruscamente em altas freqüências, mas que este caimento se inicia muito antes para as componentes de crominância (enfatizando que as freqüências estão representadas em escala logarítmica). Uma vez que o maior conteúdo de informação das imagens típicas está em baixa freqüência e a faixa superior de freqüências da Fig. 2.2.2 corresponde à informações de altas freqüências nestas imagens, a necessidade de uma resolução espacial alta nas amostras de crominância é significativamente reduzida.

Os componentes de alta freqüência de uma imagem correspondem às suas bordas, detalhes, limites e ruídos e, portanto, estarão mais presentes na componente de luminância do que nas componentes de crominância. Isto pode ser observado ao se realizar uma filtragem passa-baixa em uma imagem. Após a filtragem, a imagem perderá a nitidez e o ruído, mas as componentes de crominância quase não serão afetadas.

A possibilidade de amostragem das componentes de crominância com taxa inferior à utilizada para a componente de luminância gerou os três formatos de representação de vídeo mais utilizados, ilustrados na Fig. 2.2.3. Os nomes dados aos formatos não implicam em resoluções espaciais específicas, somente em relações relativas entre as componentes de luminância e crominância.

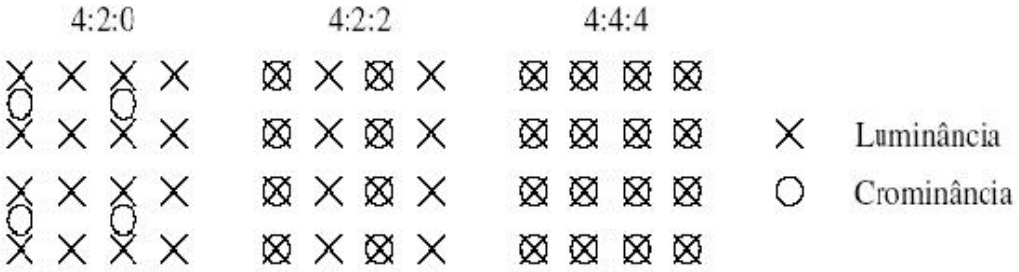


Fig. 2.2.3 – Formatos de vídeo YUV 4:2:0, YUV 4:2:2 e YUV 4:4:4 [6].

No formato YUV 4:4:4 nenhuma informação é descartada, pois o número de amostras utilizado para as componentes de crominância é o mesmo utilizado para a luminância. Este formato é pouco utilizado porque gera uma representação com 24 bpp (bits por pixel), assim como o padrão RGB. Já no formato YUV 4:2:2, as componentes de crominância são sub-amostradas por um fator de 2 na direção horizontal, gerando 16 bpp. Quando esta sub-

amostragem por um fator de 2 também é realizada na direção vertical, tem-se o formato YUV 4:2:0. Este é o formato mais utilizado, pois gera apenas 12 bpp.

Existem várias formas de armazenamento computacional destas amostras de luminância e crominância, o que originou muitos formatos de vídeo diferentes [6][8]. Estes formatos são divididos, basicamente, em formatos empacotados e planares. Em um formato empacotado, as componentes Y, U e V estão armazenadas em um único vetor e os pixels estão organizados em grupos de macropixels. Em um formato planar, as componentes Y, U e V são armazenadas em três planos diferentes.

A maioria dos formatos de vídeo YUV está associada a um código FOURCC (*Four Character Code*) [8], que é um código bem conhecido de 4 caracteres ASCII (*American Standard Code for Information Interchange*). Entre os formatos empacotados estão o UYVY e o YUY2, sendo este último o formato YUV 4:2:2 mais utilizado e cuja regra de armazenamento está apresentada na Fig. 2.2.4.

Fig. 2.2.4 – Formato de arquivo empacotado YUY2 [6].

Entre os formatos planares estão o IMC1, o IMC3, o YV12 e o NV12. O NV12 é o preferido entre os formatos YUV 4:2:0 e está ilustrado na Fig. 2.2.5.

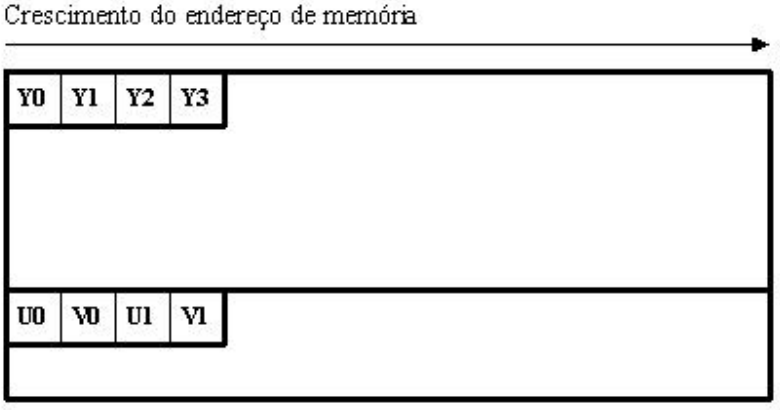


Fig. 2.2.5 – Formato de arquivo planar NV12.

No formato NV12, inicialmente todos os valores das componentes Y de um quadro do arquivo de vídeo são armazenados. Em seguida, os valores das componentes U e V são armazenados alternadamente. Como no formato YUV 4:2:0 o número de componentes U é quatro vezes menor do que o número de componentes Y e o número de componentes V é igual ao número de componentes U, a área de memória destinada à componente Y é duas vezes maior do que a área destinada às componentes U e V juntas.

O sistema FHVC foi implementado para ler arquivos de vídeo de entrada nos formatos RGB e YUV 4:2:0 [1]. A forma de armazenamento implementada foi a ilustrada na Fig. 2.2.6.

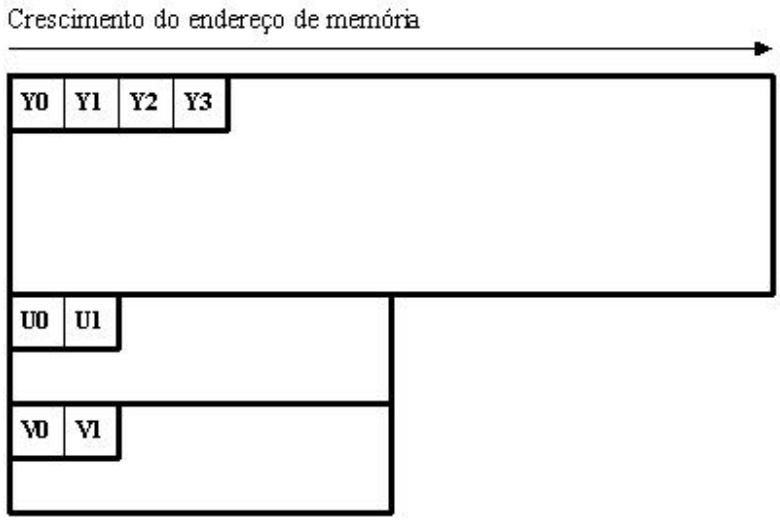


Fig. 2.2.6 – Formato de arquivo planar implementado no FHVC.

Optou-se pelo formato da Fig. 2.2.6 no FHVC porque os arquivos de vídeo utilizados para os testes descritos no Capítulo 4 já estavam previamente armazenados desta maneira. Neste formato, todos os valores das componentes Y (ou R) de um quadro são inicialmente lidos, seguidos de todos os valores das componentes U (ou G) e de todos os valores das componentes V (ou B). Este processo de leitura é feito em cada quadro do arquivo de vídeo.

Para se comprovar a eficiência da utilização do padrão YUV 4:2:0 com relação aos demais padrões anteriormente citados, também inclui-se na implementação do FHVC a possibilidade de se converter o arquivo de vídeo original (lido nos formatos RGB ou YUV 4:2:0)

[1] A Fig. 3.4.3 mostra a interface gráfica do FHVC com os possíveis formatos de arquivos de vídeo de entrada.

para os formatos YUV 4:2:2, YUV 4:4:4 e YC_oC_g [2]. Como esta conversão é feita logo após a leitura do quadro do arquivo de vídeo original, para o restante do processo de codificação, o arquivo de vídeo original já está armazenado neste novo formato.

O formato do arquivo de vídeo original e o formato para o qual o arquivo foi convertido antes da codificação são armazenados no cabeçalho do arquivo de vídeo codificado. Estas informações são necessárias porque a decodificação deve ser realizada no formato para o qual o arquivo foi convertido, mas antes do arquivo reconstruído ser armazenado, este deve ser convertido novamente para o seu formato original. Como o formato para o qual o arquivo de vídeo é convertido é utilizado apenas dentro dos processos de codificação e decodificação, este é chamado no FHVC de espaço de cores interno. Este espaço de cores deve ser escolhido na tela de opções internas do sistema[3] antes do início do processo de codificação. A utilização destas conversões permite comparar, por exemplo, a qualidade e o tamanho do arquivo codificado para o mesmo arquivo de vídeo original nos formatos YUV 4:2:0 e YUV 4:2:2 ou ainda nos formatos YUV 4:2:0 e RGB, entre outras combinações.

Todas as formas de conversão para os formatos mencionados anteriormente, com sub-amostragem ou super-amostragem das componentes de crominância, foram implementadas no FHVC. Para a sub-amostragem (necessária, por exemplo, para a conversão entre os padrões YUV 4:2:2 e YUV 4:2:0), utilizou-se a simples eliminação das amostras. Já para a super-amostragem (realizada, por exemplo, na conversão entre os padrões YUV 4:2:0 e YUV 4:2:2), cada valor foi calculado pela interpolação ponderada da curva entre os quatro valores vizinhos, com pesos maiores para os dois valores mais próximos. Este método de interpolação é conhecido como interpolação por convolução cúbica [6].

Em muitos casos, mais de uma conversão deve ser realizada. Para a conversão entre os padrões YUV 4:2:0 e RGB, por exemplo, é necessário converter de YUV 4:2:0 para YUV 4:2:2, de YUV 4:2:2 para YUV 4:4:4 e finalmente de YUV 4:4:4 para RGB.

Existem vários outros formatos para representação de cores, além do formato YUV e suas várias formas de amostragem. Um dos formatos mais eficientes é o YC_oC_g (*luminance* + *offset orange* + *offset green*)[9], que está implementado no FHVC. Este espaço de cores utiliza 24 bpp, assim como os formatos RGB e YUV 4:4:4, mas apresenta ganhos de codificação em relação a

[2] O formato YC_oC_g é descrito ainda nesta seção.
[3] A Fig. 3.4.6 mostra a tela de opções internas do sistema.

estes. Outra vantagem é a forma de conversão do espaço RGB para o espaço YC_oC_g que necessita apenas de adições e deslocamentos, conforme mostra a matriz:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} Y \\ C_o \\ C_g \end{bmatrix}. \tag{2.4}$$

Uma variação do formato YC_oC_g é o formato YC_oC_g–R [9], que é completamente reversível ao custo de um bit adicional para o armazenamento das componentes C_o e C_g.

Além das características já mencionadas do sistema visual humano exploradas nos sistemas de codificação de vídeo, duas outras características merecem ser citadas. A primeira é o fato do olho humano ter maior sensibilidade aos padrões horizontais e verticais do que aos padrões diagonais. Desta forma, as técnicas usadas na codificação (tais como as transformadas espaciais) são aplicadas horizontalmente e verticalmente nos quadros dos arquivos de vídeo. A segunda característica enfatiza a sensibilidade do olho humano tanto aos componentes de magnitude quanto aos de fase das imagens, sendo esta maior que aquela, conforme consta em [2]. Isto possibilita uma análise objetiva dos codificadores de vídeo, com medidas numéricas tais como as apresentadas no Capítulo 4, uma vez que as formas de onda de entrada e saída (ou valores de pixels, para o caso digital) devem ser iguais.

2.2.2 Formatos de vídeo

Além do formato da representação de cores do arquivo de vídeo, a resolução deste também deve ser informada no FHVC. Alguns formatos bem conhecidos, tais como CIF (*Common Intermediate Format*), QCIF (*Quarter Common Intermediate Format*), 4CIF e Sub-QCIF, já são apresentados ao usuário. Porém, arquivos de vídeo com resoluções maiores, de até 4000 x 4000 pixels - o que abrange os arquivos de vídeo no formato SHDTV (*Super High-Definition TV)* com 2000 x 2000 pixels - também podem ser lidos[4]. A única restrição é que a altura e a largura dos quadros sejam múltiplos do dobro do tamanho do cubo que será utilizado durante o processo tridimensional de codificação (conforme especificado na Seção 2.3.3).

[4] A Fig. 3.4.3 mostra a interface gráfica do FHVC com os possíveis formatos de resolução do arquivo de vídeo de entrada.

Assim como para os espaços de cores, optou-se por possibilitar o uso dos formatos 4CIF, CIF, QCIF e Sub-QCIF porque os arquivos de vídeo utilizados para os testes descritos no Capítulo 4 já estavam previamente armazenados nestes formatos (principalmente em QCIF e CIF). Os formatos operam com quadros não-entrelaçados, a uma freqüência de 30/1,001 (aproximadamente 29,97 quadros por segundo) e com uma tolerância de ±50 ppm. Uma comparação visual entre estes formatos é apresentada na Fig. 2.2.7.

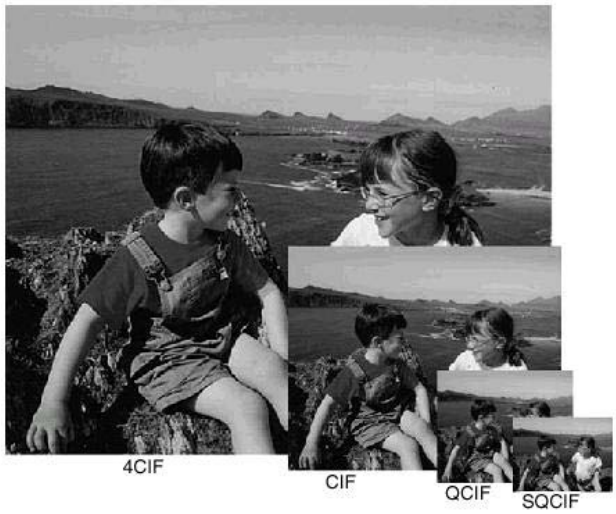


Fig. 2.2.7 – Formatos de arquivos de vídeo [11].

As diferenças entre o número de bits por quadro necessários para cada um dos formatos da Fig. 2.2.7 podem ser observadas na Tabela 2.2.1. A resolução considerada é para a componente de luminância e o padrão é o YUV 4:2:0. O número de bits por quadro é calculado pela simples multiplicação de 12 pelo número de pixels por quadro, uma vez que o padrão YUV 4:2:0 contém 12 bpp.

Formato	Resolução	Bits por quadro
Sub-QCIF	128x96	147.456
QCIF	176x144	304.128
CIF	352x288	1.216.512
4CIF	704x576	4.866.048

Tabela 2.2.1– Comparação de bits por quadro para vários formatos de vídeo.

2.3 TRANSFORMADA 3D DE HADAMARD

A seção anterior apresentou a forma de leitura dos quadros dos arquivos de vídeo e as conversões entre seus espaços de cores feitas no FHVC para redução da redundância psico-visual destes arquivos. Esta seção trata das próximas redundâncias a serem reduzidas no processo de codificação, a espacial e a temporal. A redundância espacial será reduzida com a aplicação da transformada de Hadamard. Para a redução da redundância temporal, esta mesma transformada também será aplicada tridimensionalmente, ou seja, entre os quadros.

2.3.1 Codificação por transformadas

Transformadas lineares podem ser utilizadas para reduzir a redundância espacial existente entre pixels dentro do mesmo quadro (caracterizada pelas altas correlações entre pixels vizinhos) porque produzem seqüências de variáveis (coeficientes) com correlação tipicamente reduzida a partir destes pixels. De modo geral, uma transformada é um mapeamento linear, unitário e inversível entre dois espaços vetoriais. No caso das transformadas aplicadas a quadros de arquivos de vídeo, o espaço de origem é formado por vetores cujos elementos são os valores dos pixels do quadro que se deseja codificar e o espaço destino é formado pelos coeficientes resultantes da transformada.

A aplicação de uma transformada em um quadro de vídeo envolve a divisão do quadro em blocos *N*x*N*. A transformada é, então, aplicada a cada um destes blocos separadamente. Como os blocos de pixels são bidimensionais, a transformada também é dita bidimensional. A redução da correlação observada nos coeficientes resultantes desta transformada geralmente resulta em uma redistribuição da energia do sinal em um pequeno conjunto de coeficientes, característica conhecida como propriedade de compactação da energia. Desta forma, muitos coeficientes apresentam valores praticamente nulos e podem ser descartados antes do próximo passo do processo de codificação, que no caso do FHVC, é a codificação entrópica. É importante ressaltar que a aplicação da transformada não realiza compressão no sinal, apenas reduz a correlação entre os valores originais dos coeficientes e compacta a energia em poucos coeficientes. A compressão é obtida pela quantização explícita dos coeficientes transformados, ou no caso FHVC, pela

quantização implícita realizada através de envio de planos de bits, conforme explicado na Seção 2.5.2.

Existem várias formas de interpretação das transformadas unitárias realizadas em blocos de *N*x*N* pixels [10]. Uma abordagem é considerar a transformada como uma rotação dos eixos coordenados definidos pelos pixels do bloco. Outra visão, adotada neste trabalho, é que a transformada decompõe o bloco original em um conjunto de *N*x*N* funções base ortogonais (cada uma com *N*x*N* amostras) através do produto interno entre o bloco e as funções base [21],

$$\hat{x}[k,l] = \left\langle X, W_{k,l} \right\rangle = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x[m,n] w_{k,l}[m,n], \quad \text{para } 0 \le k, l \le N-1, \quad (2.5)$$

onde o vetor x contém os valores originais dos pixels do bloco *N*x*N*, o vetor $w_{k,l}$ contém as amostras *N*x*N* de uma das funções base da transformada e a variável $\hat{x}[k,l]$ contém o coeficiente relacionado à função base $w_{k,l}$. Segundo esta visão, cada coeficiente resultante da transformada está relacionado a uma função base e representa a similaridade do sinal que está sendo codificado com esta função. Sendo assim, para a reconstrução de um bloco do sinal original, basta somar as funções base ortogonais ponderadas pelos coeficientes resultantes da transformada.

Se além de ortogonais, as funções base da transformada bidimensional forem normalizadas, a transformada é dita ortonormal (ou unitária) e a equação de reconstrução é uma simples combinação linear dos vetores da base, tendo como coeficientes os próprios coeficientes da transformada [21], conforme mostra a equação:

$$x[m,n] = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} \hat{x}[k,l] w_{k,l}[m,n], \qquad \text{para } 0 \le m, n \le N-1. \quad (2.6)$$

A característica de compactação de energia do sinal obtida pela aplicação das transformadas é importante para que o sinal possa ser comprimido através da quantização dos coeficientes transformados. Porém, também é muito importante que esta energia seja apenas compactada ou redistribuída, não perdida. A energia do sinal deve ser preservada no domínio da transformada para que se possa fazer a reconstrução do sinal a partir de seus coeficientes. A utilização de transformadas ortonormais, como as descritas acima, garante a preservação da energia do sinal transformado. Para estas transformadas, o teorema de Parseval [2] afirma que:

$$\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\left|x[m,n]\right|^2 = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\left|\hat{x}[m,n]\right|^2 . \tag{2.7}$$

Uma transformada bidimensional é dita separável caso suas funções de base possam ser escritas como produtos tensoriais[5] das funções base de duas transformadas unidimensionais [21], isto é,

$$W_{k,l} = u_k \otimes v_l = w_{k,l}[n,n] = u_k[n]v_l[n], \tag{2.8}$$

onde u_k (com $0 \le k \le N-1$) e v_l (com $0 \le l \le N-1$) são os vetores base de duas transformadas unidimensionais com N valores cada. A utilização de transformadas separáveis é importante do ponto de vista computacional porque a transformada bidimensional pode ser calculada inicialmente para as linhas do bloco que está sendo codificado e posteriormente para as colunas (a ordem inversa também pode ser usada).

As funções base diferem para cada tipo de transformada e, em alguns casos, dão nome às mesmas, a exemplo da transformada mais comumente utilizada em codificação de imagens, a DCT (*Discrete Cosine Transform*), que utiliza funções base co-senoidais. A DCT apresenta uma boa compactação de energia do sinal, porém a transformada considerada ótima é a KLT (*Karhunen- Loève Transform*). O uso da KLT não é tão difundido quanto o da DCT porque as funções base da KLT são os auto-vetores da matriz de correlação da imagem. Isto torna o cálculo das funções base dependente da imagem, o que requer o envio destas funções base para o decodificador e dificulta o desenvolvimento de um algoritmo rápido. Já a DFT (*Discrete Fourier Transform*) possui um algoritmo de cálculo rápido, mas não é muito utilizada em imagens devido à baixa compactação da energia em poucos coeficientes. Além disso, os coeficientes gerados são complexos.

A DFT pode ser utilizada para uma interessante interpretação que pode ser estendida para as demais transformadas [10]. Uma vez que as funções base da DFT consistem em senos e co-senos de diferentes freqüências, é natural interpretar esta transformada como uma decomposição espectral da imagem original. Esta interpretação pode ser estendida para as demais transformadas

[5] O produto tensorial entre dois vetores a e b pode ser definido como:

$$[\mathbf{a} \otimes \mathbf{b}] = \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{bmatrix} \Rightarrow [\mathbf{a} \otimes \mathbf{b}] = \begin{bmatrix} a_1 \\ a_2 \\ a_2 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} = \mathbf{a}\mathbf{b}^T.$$

se o conceito de freqüência for generalizado para incluir outras funções além de senos e co-senos. Assim, por exemplo, na transformada WHT (*Walsh-Hadamard Transform*), formas de onda retangulares com número crescente de mudanças de sinal são utilizadas como funções base. Cada coeficiente da transformada é proporcional à fração de energia da imagem original que corresponde à função espectral que está sendo considerada.

De todas as transformadas citadas anteriormente, a transformada WHT é a que apresenta a menor capacidade de descorrelação entre os pixels dos quadros e, portanto, a pior compactação de energia. Ao se calcular o ganho da WHT para um processo de entrada de Markov estacionário de ordem 1 (também conhecido como processo AR(1)) com coeficiente de correlação $\rho = 0,9$ e N = 4 (onde N é o comprimento da transformada), obtém-se o valor de 5 dB. O mesmo cálculo de ganho produz o valor de 5,39 dB [52] para a DCT e de 5,41 dB para a KLT. O resultado obtido para a DCT está de acordo com a afirmação de que a DCT é aproximadamente ótima para um processo AR(1) com alto valor positivo de coeficiente de correlação [53].

As diferenças de ganho entre as transformadas crescem significativamente com o aumento da ordem do processo de entrada e do comprimento N da transformada. A WHT possui o menor ganho de transformada, porém apresenta a forma mais simples e rápida de implementação. Devido à esta característica foi a transformada escolhida para ser implementada no FHVC e será descrita na próxima seção.

2.3.2 Transformada de Hadamard

A transformada de Hadamard tem uma forma rápida de implementação porque os elementos das suas funções base consistem somente de valores +1 e –1 e, portanto, o cálculo da transformada não requer multiplicações [4]. As matrizes H_n de Hadamard são matrizes NxN onde $N = 2^n$. Estas matrizes são geradas através da matriz base

$$H_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.9}$$

e do produto recursivo de Kronecker

$$H_n = H_{n-1} \otimes H_1 = H_1 \otimes H_{n-1} = \frac{1}{\sqrt{2^n}}\begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}. \tag{2.10}$$

Os vetores base da matriz de Hadamard também podem ser gerados através da amostragem de uma classe de funções retangulares chamada de funções *Walsh* ilustrada na Fig. 2.3.1. Estas funções também só possuem amplitudes +1 e –1 e formam uma base ortonormal completa. Por esta razão, a transformada de Hadamard é também chamada de transformada de Walsh-Hadamard.

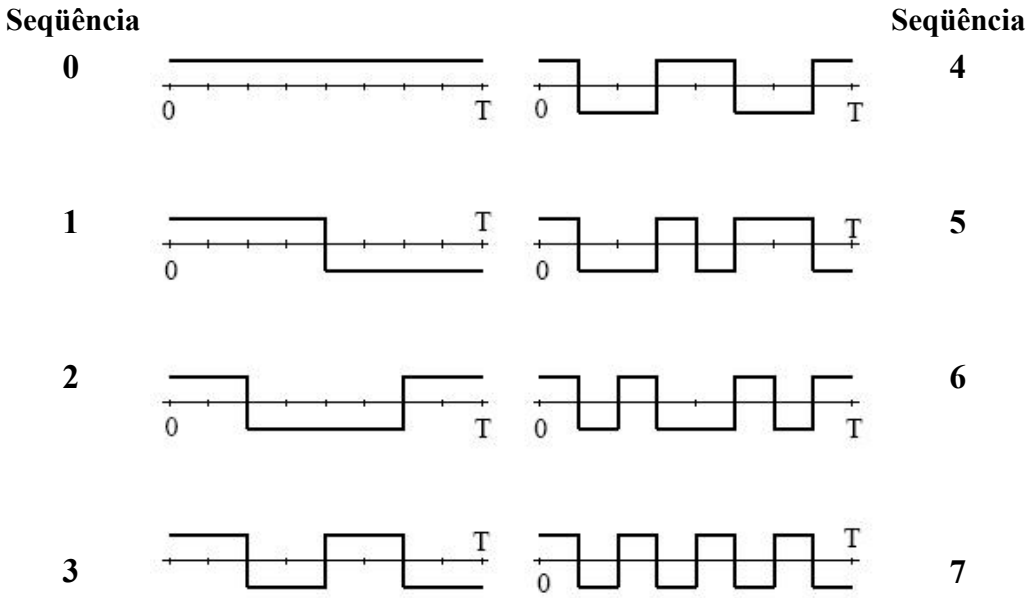


Fig. 2.3.1 – Funções de Walsh que são amostradas para a geração da matriz de Hadamard.

O número de cruzamentos de zeros da função Walsh é chamado de seqüência. Porém, as matrizes de Hadamard geradas de acordo com a Eq. (2.10) não possuem as linhas ordenadas de acordo com esta seqüência. Um exemplo de uma matriz de Hadamard gerada desta forma para *n=3* é apresentado por:

$$H_3 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{matrix} \text{Seqüência} \\ 0 \\ 7 \\ 3 \\ 4 \\ 1 \\ 6 \\ 2 \\ 5 \end{matrix} . \qquad (2.11)$$

A ordem (seqüência) em que as linhas estão ordenadas na matriz da Eq. (2.10) é chamada de ordem de Hadamard.

A transformada direta de Hadamard de um vetor u de tamanho N e a transformada inversa do vetor v são dadas por:

$$v = H_n * u ; \tag{2.12}$$

$$u = H_n * v . \tag{2.13}$$

A mesma matriz H_n pode ser utilizada no cálculo das transformadas direta e inversa porque a matriz de Hadamard é real, simétrica e ortogonal, ou seja,

$$H_n = H_n^{*} = H_n^{T} = H_n^{-1} . \tag{2.14}$$

A transformada direta unidimensional de Hadamard da Eq. (2.12) é uma transformada rápida porque pode ser implementada por $N * log_2 N$ adições e subtrações no lugar de N^2 operações. Várias formas de cálculo rápido da transformada de Hadamard já foram desenvolvidas e algumas delas estão em [12], [13] e [14]. A forma escolhida para ser implementada no FHVC considera o fato da matriz de Hadamard H_n poder ser escrita como um produto de n matrizes esparsas $\widetilde{H}$, de forma que,

$$H_n = \widetilde{H}^n \text{, onde } n = log_2 N. \tag{2.15}$$

Substituindo-se a Eq. (2.15) na Eq. (2.12), tem-se que:

$$v = \widetilde{H}^n * u = \underbrace{\widetilde{H} * (\widetilde{H} ... * (\widetilde{H} * u))}_{n} . \tag{2.16}$$

Para $n = 3$, por exemplo, a Eq. (2.16) pode ser escrita como $v = \widetilde{H} * (\widetilde{H} * (\widetilde{H} * u)))$ e o resultado da primeira multiplicação do vetor u pela matriz esparsa $\widetilde{H}$ é dado por:

$$\underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}}_{\tilde{H}} * \begin{bmatrix} u[1] \\ u[2] \\ u[3] \\ u[4] \\ u[5] \\ u[6] \\ u[7] \\ u[8] \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} u[1]+u[2] \\ u[3]+u[4] \\ u[5]+u[6] \\ u[7]+u[8] \\ u[1]-u[2] \\ u[3]-u[4] \\ u[5]-u[6] \\ u[7]-u[8] \end{bmatrix}. \tag{2.17}$$

Esta primeira multiplicação implica em $N = 2^n = 8$ adições/subtrações. Considerando que esta multiplicação será repetida $n = log_2 N = 3$ vezes, o número total de operações necessárias é de $N * log_2 N = 24$. Após todas as aplicações recursivas da matriz $\tilde{H}$, o vetor v resultante é dado por:

$$v = \frac{1}{\sqrt{8}} \begin{bmatrix} u[1]+u[2]+u[3]+u[4]+u[5]+u[6]+u[7]+u[8] \\ u[1]-u[2]+u[3]-u[4]+u[5]-u[6]+u[7]-u[8] \\ u[1]+u[2]-u[3]-u[4]+u[5]+u[6]-u[7]-u[8] \\ u[1]-u[2]-u[3]+u[4]+u[5]-u[6]-u[7]+u[8] \\ u[1]+u[2]+u[3]+u[4]-u[5]-u[6]-u[7]-u[8] \\ u[1]-u[2]+u[3]-u[4]-u[5]+u[6]-u[7]+u[8] \\ u[1]+u[2]-u[3]-u[4]-u[5]-u[6]+u[7]+u[8] \\ u[1]-u[2]-u[3]+u[4]-u[5]+u[6]+u[7]-u[8] \end{bmatrix}. \tag{2.18}$$

Observa-se que, conforme esperado, o vetor v em (2.18) é exatamente o que seria gerado através das $N^2 = 64$ operações necessárias para a aplicação da equação (2.12) com H_3. Sendo assim, com o uso da Eq. (2.16), o número de adições/subtrações necessárias foi reduzido de 64 para apenas 24.

A compactação de energia obtida com a aplicação da transformada de Hadamard varia de bom para muito bom quando comparada com outras transformadas, tais como a DCT, a qual é classificada como excelente [4]. Isto significa que a energia está mais espalhada pelos coeficientes, não se concentrando tanto nos coeficientes de Hadamard de baixa freqüência. O coeficiente de freqüência zero contém o valor DC, ou valor médio, do sinal. A eficiência da

transformada de Hadamard aumenta quando são consideradas imagens de alta definição, uma vez que estas são altamente correlacionadas espacialmente [15].

Além da implementação rápida, outra vantagem da transformada de Hadamard é o baixo índice de perdas que a transformada acrescenta ao processo de codificação, uma vez que é implementada apenas com adições e subtrações. O único fator gerador de perdas é a divisão dos valores dos coeficientes por $\sqrt{2}$ a cada aplicação recursiva da Eq. (2.16). Para evitar a geração de coeficientes fracionários, as divisões por $\sqrt{2}$ são agrupadas de forma a eliminar a operação de raiz quadrada e implementadas por deslocamentos binários no FHVC. Este processo será mencionado na próxima seção, que descreve como a transformada de Hadamard foi implementada de forma tridimensional no FHVC.

2.3.3 Codificação inter-quadros

Seqüências de vídeo típicas, com taxas maiores que 15 quadros/segundo, apresentam um alto índice de correlação entre quadros vizinhos. Um processo de codificação inter-quadros visa explorar esta redundância temporal e geralmente é realizado com técnicas de estimação e compensação de movimento entre os quadros.

Uma outra forma de codificação inter-quadros consiste na visualização de um arquivo de vídeo como uma seqüência de dados tridimensional e na extensão das técnicas bidimensionais aplicadas na codificação intra-quadros para a dimensão temporal. Esta forma de codificação tridimensional apresenta uma alta qualidade do arquivo reconstruído e elimina alguns problemas causados pela estimação de movimentos, tais como os atrasos devidos aos pesados algoritmos de busca e casamento de blocos e as dificuldades em arquivos de vídeo que apresentam movimentos rápidos ou que tenham rotação, *zoom* e mascaramento de objetos. Já as desvantagens da codificação tridimensional estão na geração de atrasos em implementações de tempo real devido à necessidade de se ter vários quadros do vídeo para realizar o processamento e na necessidade de maior quantidade de memória para armazenamento dos coeficientes resultantes da codificação, os quais passam de *N*x*N* por bloco (onde *N* é o tamanho do bloco) para *N*x*N*x*N*. Evidentemente, estas dificuldades podem ser contornadas com a utilização de blocos de tamanho menor em relação aos utilizados nas técnicas bidimensionais.

O processo de codificação tridimensional permite que transformadas diferentes sejam utilizadas em cada dimensão. Sendo assim, com base nas características do arquivo de vídeo que está sendo codificado, pode-se aplicar a técnica mais apropriada nas linhas do arquivo, outra técnica nas colunas e outra entre os quadros. Uma vez que um requisito importante do FHVC é a rapidez e simplicidade de processamento, optou-se pela aplicação da transformada de Hadamard, cuja forma de implementação adotada foi descrita na seção anterior, nas três dimensões do arquivo de vídeo.

A seqüência de vídeo que está sendo codificada é particionada em cubos e a transformada de Hadamard tridimensional é calculada em cada cubo separadamente. Este procedimento é realizado para cada uma das componentes do espaço de cores interno que está sendo utilizado. A Fig. 2.3.2 ilustra a divisão dos quadros de um arquivo de vídeo em cubos de tamanho 8x8x8.

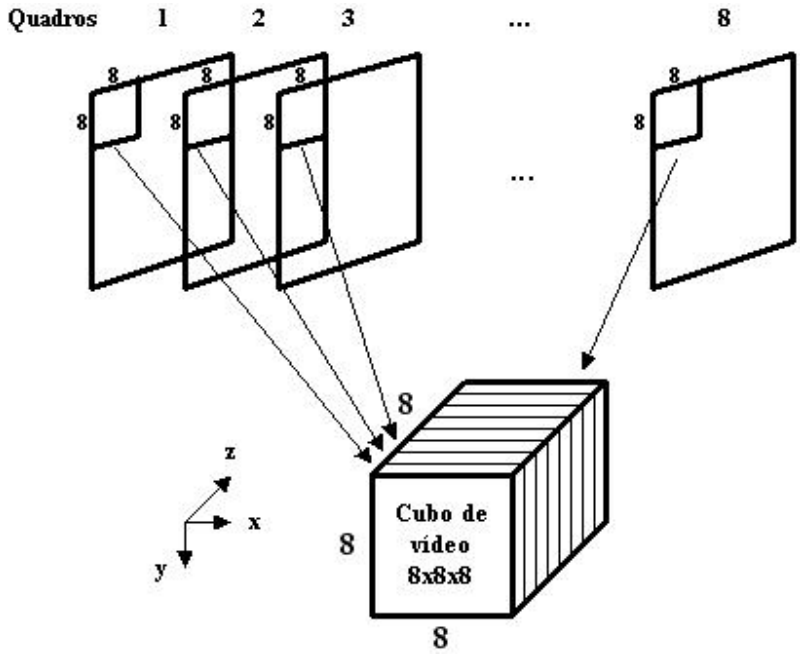


Fig. 2.3.2 – Formação de um cubo de vídeo 8x8x8 [17].

O tamanho adotado para o cubo tem uma grande influência no resultado final da codificação. Um cubo de tamanho 8x8x8 explora melhor a redundância temporal e a redundância espacial do arquivo de vídeo do que um cubo de tamanho 4x4x4, por exemplo. Porém, um cubo 8x8x8 requer maior quantidade de memória para armazenamento dos coeficientes e eleva o atraso em implementações de tempo real. Sendo assim, em arquivos de vídeo com pouco movimento e

conteúdo de informação uniforme, a utilização de cubos 8x8x8, em geral, apresenta melhores resultados do que a utilização de cubos de tamanho menor. Para avaliar o efeito do tamanho do cubo no processo de codificação, o FHVC pode ser executado com cubos de tamanho 4x4x4 ou 8x8x8 [6]. Outras estruturas poderiam ser utilizadas para a leitura dos quadros de vídeo, tais como paralelepípedos 8x8x4 em seqüências com muito movimento e paralelepípedos 4x4x8 em seqüências com quadros ricos em detalhes. A utilização destas estruturas não é possível no FHVC devido ao requisito de rapidez de processamento, que é alcançado pela leitura de cubos por possibilitar o reuso das mesmas estruturas computacionais nas três dimensões da seqüência.

Uma vez que a transformada tridimensional de Hadamard é separável, esta foi implementada no FHVC inicialmente nas colunas dos cubos de vídeo, depois nas linhas e finalmente entre os quadros, porém qualquer outra ordem poderia ser sido utilizada para a geração do mesmo resultado. Se os arquivos de vídeo forem de imagens que apresentam uma grande correlação temporal e a transformada não for totalmente separável, pode ser mais vantajoso aplicá-la inicialmente entre os quadros, conforme demonstrado em [16], no qual se utiliza uma transformada wavelet na codificação de arquivos de vídeo com imagens médicas.

De acordo com a Eq. (2.10), cada implementação da transformada gera a multiplicação dos valores dos coeficientes resultantes por um fator de $1/\sqrt{N}$, onde N é o tamanho da lateral do cubo. Isto é feito na transformada de Hadamard para a conservação da energia da imagem (conforme a Eq. 2.7). De forma a evitar a geração de coeficientes fracionários após a divisão dos coeficientes por $\sqrt{N}$, o FHVC realiza duas implementações da transformada de Hadamard e só então aplica o fator de conservação de energia através de deslocamentos binários.

Inicialmente a transformada de Hadamard é aplicada nas colunas dos cubos de vídeo e em seguida nas linhas, o que gera uma divisão dos coeficientes por $\sqrt{N} * \sqrt{N} = N$. Considerando que os valores de N utilizados no FHVC são múltiplos de dois, cada divisão por N é realizada simplesmente com $log_2 N$ deslocamentos binários para direita.

Uma implementação realizada no FHVC para reduzir a magnitude dos coeficientes resultantes da aplicação da transformada tridimensional de Hadamard considera que, como a transformada inversa de Hadamard é idêntica à transformada direta, antes da primeira aplicação da transformada inversa os coeficientes deveriam ser divididos por $\sqrt{N}$. Esta primeira divisão

[6] O tamanho do cubo pode ser escolhido na tela de opções internas do sistema ilustrada na Fig. 3.4.6.

que seria realizada durante a decodificação dos coeficientes é trazida para a codificação e agrupada com a aplicação da transformada entre os quadros, o que gera uma nova divisão dos coeficientes por N ainda na codificação.

Resumidamente, no processo de codificação do FHVC, a transformada de Hadamard é aplicada nas colunas e nas linhas dos cubos de vídeo e os coeficientes sofrem $log_2 N$ deslocamentos binários para direita. Em seguida, a transformada é aplicada entre os quadros e os coeficientes sofrem novamente $log_2 N$ deslocamentos binários para direita. Estes deslocamentos não podem ser agrupados e realizados antes da aplicação da transformada entre os quadros porque os valores dos coeficientes se tornam muito pequenos e perdem muita precisão. Também não podem ser realizados de forma agrupada depois da transformada entre os quadros porque, neste caso, os valores dos coeficientes se tornam muito grandes. Sendo assim, esta forma de implementação da transformada tridimensional com deslocamentos binários intercalados às aplicações de cada transformada unidimensional possibilita que a faixa dinâmica dos coeficientes da transformada não aumente muito durante o processo de codificação, de forma que o FHVC pôde ser completamente implementado utilizando aritmética de apenas 16 bits. Já na decodificação, inicialmente a transformada é aplicada entre os quadros dos cubos de vídeo e os coeficientes sofrem os últimos $log_2 N$ deslocamentos binários para direita. Finalmente, a transformada é aplicada nas linhas e nas colunas dos cubos.

Os coeficientes obtidos no final da codificação poderiam ter valores ainda menores se os deslocamentos que ainda são realizados na decodificação também passassem para a codificação. Entretanto, se um cubo de tamanho 8x8x8 estiver sendo utilizado, os valores dos coeficientes sofrem seis deslocamentos binários para direita ao invés de três, o que resulta em coeficientes de valor muito pequeno, que não podem ser recuperados satisfatoriamente durante a decodificação, dada a precisão finita do sistema computacional.

A seguir, apresenta-se uma análise do aumento da faixa dinâmica proporcionada pela transformada tridimensional de Hadamard implementada conforme explicado anteriormente.

Inicialmente, considera-se que os valores dos pixels dos quadros dos arquivos do vídeo são representados por 8 bits, variando, portanto, entre 0 e 255. Uma forma de se reduzir o máximo valor absoluto possível é diminuir 128 de todos os pixels dos quadros. Com esta diminuição, os pixels dos quadros passam a variar entre –128 e 127 e a ter média nula. Uma forma menos custosa computacionalmente de se retirar o valor médio dos pixels é fazer a

diminuição somente no valor do coeficiente DC (de freqüência zero) após o primeiro cálculo da transformada. Uma vez que o coeficiente DC contém o valor médio do sinal inteiro, o valor a ser reduzido aumenta de 128 para 128 * *N*, onde *N* é o tamanho da transformada (no caso do FHVC, *N = 4* ou *N = 8*).

Uma vez que a matriz de Hadamard apresenta apenas valores 1 e –1, o ganho de faixa dinâmica proporcionado pela aplicação da transformada unidimensional é de $(N * 1) / \sqrt{N}$. Se *N = 8*, por exemplo, conforme apresentado na Eq. (2.11), o ganho de faixa dinâmica, evidenciado pela primeira linha da matriz, é de $8 / \sqrt{8}$. Considerando que a transformada calculada é tridimensional e a primeira divisão por $\sqrt{N}$ da decodificação foi implementada na codificação, o ganho total de faixa dinâmica é de $8^3 / (\sqrt{8})^4 = 8$. Como $log_2 8 = 3$, são necessários apenas 3 bits a mais para armazenar os coeficientes do que para armazenar os valores dos pixels. A análise com *N = 8* é válida e suficiente para o FHVC porque o máximo valor permitido para o tamanho do cubo é de 8x8x8.

2.4 ORDENAÇÃO DOS COEFICIENTES DA TRANSFORMADA

A aplicação da transformada tridimensional de Hadamard descrita na Seção 2.3 gera um cubo de coeficientes de mesmo tamanho e mesma energia que o cubo de pixels original que foi codificado. Porém, a compactação de energia obtida com a aplicação da transformada faz com que a energia do cubo não esteja mais dispersa entre seus valores e se concentre nos coeficientes de baixa freqüência. Sendo assim, se os eixos *x*, *y* e *z* apresentados na Fig. 2.3.2 forem interpretados como eixos de direção de crescimento da freqüência para o cubo de coeficientes, o primeiro coeficiente do canto esquerdo superior do cubo é o coeficiente de freqüência (0,0,0) e, portanto, corresponde ao coeficiente DC do cubo. Os coeficientes AC próximos ao coeficiente DC nas três dimensões são os coeficientes de freqüência baixa que concentram a maior parte da energia do cubo. Os demais coeficientes AC estão associados às freqüências mais altas do cubo e possuem valores nulos ou muito próximos de zero. Quanto maior for a capacidade de compactação de energia da transformada, maior será a quantidade de coeficientes de alta freqüência com valores nulos que podem ser descartados durante o processo de codificação.

A leitura dos coeficientes do cubo de forma decrescente, ou decrescente na média, é importante para a formação de uma seqüência ordenada de coeficientes para ser utilizada no

próximo passo do processo de codificação, que geralmente é a etapa de quantização, ou no caso do FHVC, a codificação de entropia adaptativa por código de Golomb com codificação por planos de bits no lugar da quantização. A formação de uma seqüência esparsa, contendo inicialmente poucos coeficientes não-nulos agrupados seguidos de um grande número de coeficientes nulos, aumenta muito a eficiência da codificação de entropia.

Ainda não existem formas padronizadas de leitura de coeficientes 3D, tais como as tabelas de quantização ou a leitura em *zig-zag* usadas para a 2D-DCT implementada em padrões como JPEG (*Joint Photographic Experts Group*) e MPEG (*Motion Pictures Experts Group*). Porém, algumas formas de leitura já foram estudadas e publicadas, tais como as apresentadas em [18], [19] e [20]. Uma vez que nestes artigos, a forma de leitura é voltada para a realização de uma quantização mais eficiente dos coeficientes, os métodos apresentados procuram dividir o cubo em regiões de coeficientes significativos, que são lidas com uma forma de quantização mais precisa, e regiões de coeficientes não-significativos, que geralmente são desprezadas. A classificação dos coeficientes em significativos ou não-significativos e o cálculo dos valores de quantização utilizados para estes coeficientes são feitos através de equações que definem a forma das regiões de coeficientes significativos e de valores de limiar que determinam a significância dos coeficientes dentro da região.

As formas de leitura dos coeficientes do cubo apresentadas nos artigos [18], [19] e [20] são computacionalmente custosas porque são dependentes do conteúdo do cubo. Além disso, requerem cálculos de valores de significância de cada coeficiente e comparação com valores de limiar. Este tipo de leitura não pôde ser utilizado no FHVC devido à necessidade de rapidez de implementação. Desta forma, desenvolveu-se uma forma de leitura fixa, independente do conteúdo do cubo. Para o desenvolvimento desta forma de leitura foi realizada uma análise do conteúdo de informação do cubo de coeficientes para alguns arquivos de vídeo e identificou-se uma forma aproximadamente comum de espalhamento da energia no cubo. Esta análise utilizou as mesmas técnicas apresentadas em [18], onde o espalhamento de energia do cubo foi estudado para coeficientes resultantes da aplicação da 3D-DCT. A forma de leitura resultante, implementada no FHVC, também utiliza os conceitos de particionamento de conjuntos tridimensionais em árvores hierárquicas (SPIHT – *Set Partitioning in Hierarchical Trees*) apresentados em [19].

2.4.1 Ordem de leitura dos coeficientes DC

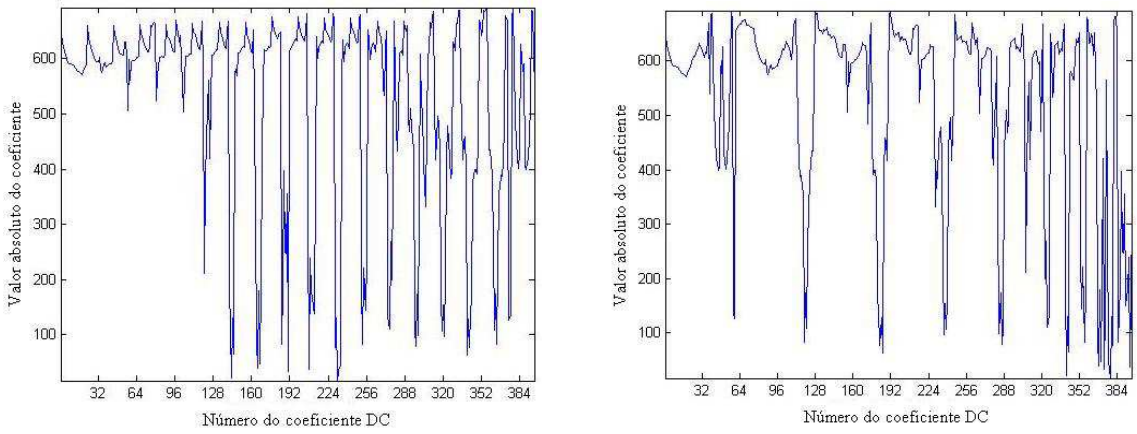
A análise foi feita inicialmente para os valores dos coeficientes DC dos vários cubos de um bloco de quadros do arquivo de vídeo. Uma vez que estes são os maiores valores de coeficientes do bloco, podem ser lidos conjuntamente no início da seqüência, de forma a agrupar a maior energia do sinal. Blocos de quadros que possuem cubos com coeficientes DC de valor alto correspondem aos blocos da seqüência de vídeo com pouca atividade, enquanto que os blocos com coeficientes DC de valor baixo correspondem aos blocos com alta atividade, movimentos complexos, tais como *zoom* ou mudanças de cena. Tais valores baixos para os coeficientes DC são causados pelo espalhamento da energia do bloco nos coeficientes AC. Sendo assim, blocos de quadros correspondentes a cenas com alta atividade possuem tipicamente valores maiores para os coeficientes AC.

Para a ilustração da forma de leitura implementada no FHVC será utilizada a seqüência de vídeo “Miss America” no formato QCIF por ser esta uma seqüência muito conhecida e utilizada para testes na área de codificação de vídeo. A seqüência utilizada está no formato YUV 4:2:0 para que os resultados da forma de leitura dos componentes Y, U e V possam ser analisados separadamente. O processamento da seqüência foi feito com cubos de tamanho 8x8x8 e foram escolhidos os quadros de 64 a 71 para a análise porque este bloco apresenta mais movimento que os blocos iniciais e, portanto, pode-se fazer uma análise melhor do espalhamento de energia entre os coeficientes AC. Alguns quadros deste bloco são mostrados na Fig. 2.4.1.

(a) (b) (c)

Fig. 2.4.1 – Quadros #64 (a), #67 (b) e #71 (c) da seqüência de vídeo "Miss America".

Uma vez que o arquivo de vídeo está no formato QCIF, possui 176 colunas por 144 linhas, resultando em $\frac{176}{8} * \frac{144}{8} = 22 * 18 = 396$ cubos de coeficientes 8x8x8 para cada bloco de 8 quadros da seqüência de vídeo. A leitura por linhas dos valores absolutos dos coeficientes DC dos 396 cubos é apresentada no gráfico da Fig. 2.4.2(a).



(a) (b)

Fig. 2.4.2 – Leitura por linhas (a) e em espiral (b) da componente Y dos coeficientes DC do bloco de quadros.

O valor dos coeficientes DC dos cubos pode ser negativo porque os valores dos pixels foram deslocados para a faixa de –128 a 127, conforme explicado anteriormente na Seção 2.3.3. É importante que a análise da leitura dos coeficientes DC seja feita para os seus valores absolutos porque a seqüência final ordenada é passada para o codificador de entropia do FHVC e este considera apenas os valores absolutos dos coeficientes, armazenando e transmitindo seus sinais separadamente. Sendo assim, aproximadamente os primeiros 110 coeficientes apresentados no gráfico da Fig. 2.4.2(a) têm valores próximos a 600 que são, na realidade, negativos, pois correspondem às primeiras 40 linhas dos quadros que têm pixels escuros, conforme pode ser observado na Fig. 2.4.1, e portanto, com baixos valores de intensidade. Isto pode ser concluído porque cada linha do quadro está dividida entre 22 cubos e cada cubo abrange 8 linhas do quadro. Tem-se, portanto, 22 valores DC a cada 8 linhas do quadro e 110 (onde 110 = 22 * 40/8) valores DC a cada 40 linhas.

Aproximadamente a partir do centésimo coeficiente começam a ser observados valores menores gerando picos que se repetem de forma periódica. Estes valores menores, na maioria

próximos ou abaixo de 100, são realmente positivos e representam o conteúdo de freqüência da região central mais clara dos quadros analisados para a seqüência “Miss America”.

A análise anterior e os gráficos foram repetidos para outros arquivos de vídeo. Para os arquivos de vídeo do mesmo tipo da seqüência "Miss America", com uma região fixa de fundo e conteúdo disposto centralmente, verificou-se que a distribuição dos valores DC é aproximadamente a mesma da Fig. 2.4.2(a). Os baixos valores absolutos DC encontrados na região central dos quadros também se devem ao movimento maior que as seqüências de vídeo do tipo da seqüência "Miss America" apresentam nesta região. Se, ao contrário da seqüência "Miss America", a seqüência de vídeo contiver uma região de fundo clara, os coeficientes DC desta região terão valores altos, neste caso, já incluindo o sinal. Estes valores também serão altos porque o movimento das regiões de fundo é praticamente inexistente de forma geral.

Após a observação das análises descritas anteriormente, desenvolveu-se para o FHVC uma leitura dos valores dos coeficientes DC de forma espiral, com início no canto superior esquerdo do bloco de quadros e término no centro do bloco. A Fig. 2.4.3 ilustra esta forma de leitura para um bloco de 8 quadros da seqüência “Miss America”. Na figura, o bloco aparece dividido nos 396 cubos comentados anteriormente e alguns valores DC destes cubos estão representados por pequenos quadrados brancos somente para melhor visualização do processo de leitura, uma vez que estes são os valores lidos pela espiral que percorre o bloco e seus cubos.

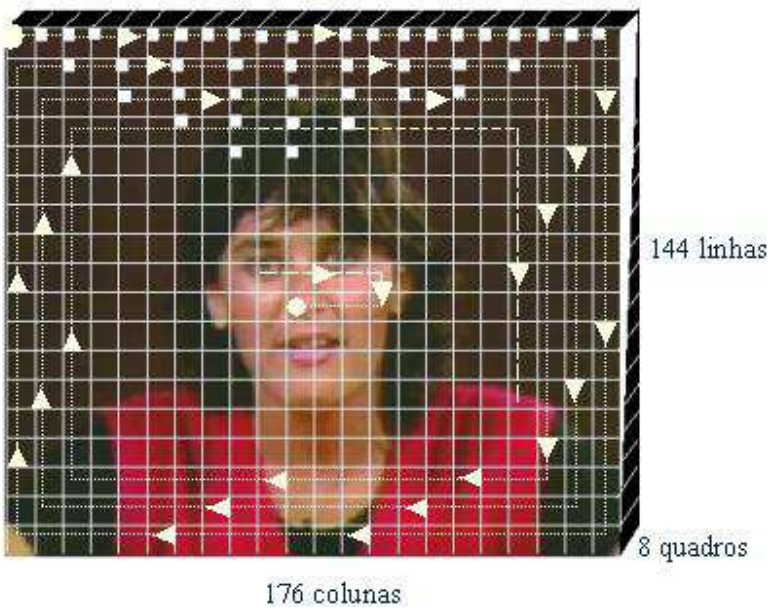


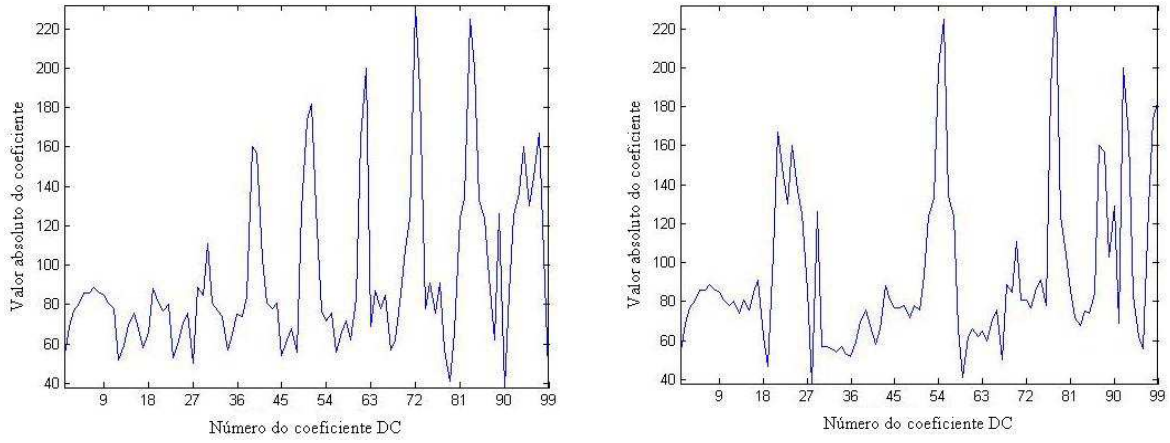
Fig. 2.4.3 – Ilustração do processo de leitura em espiral dos coeficientes DC de um bloco de quadros de vídeo.

A espiral apresentada na Fig. 2.4.3 não foi desenhada de forma completa para evitar sobrecarga de informações na figura. Sendo assim, utilizou-se a linha tracejada para indicar que a espiral continua até o centro da figura, onde a circunferência branca representa o seu término.

A seqüência resultante desta leitura em forma espiral é apresentada no gráfico da Fig. 2.4.2(b). Percebe-se que os valores similares estão mais agrupados, uma vez que o gráfico apresenta menor variação, com menos picos. Isto ocorre porque inicialmente estão sendo lidos valores da região de fundo dos quadros e por fim os valores centrais. O primeiro vale observado no gráfico da Fig. 2.4.2(b) deve-se à região mais clara do canto inferior esquerdo da seqüência, correspondente ao braço da "Miss America" e os demais vales já se referem à região do pescoço. Desta análise, conclui-se que quanto mais centralizado estiver o conteúdo da seqüência de vídeo, mais agrupados estarão os coeficientes DC de valores próximos e menor variação haverá na seqüência ordenada resultante.

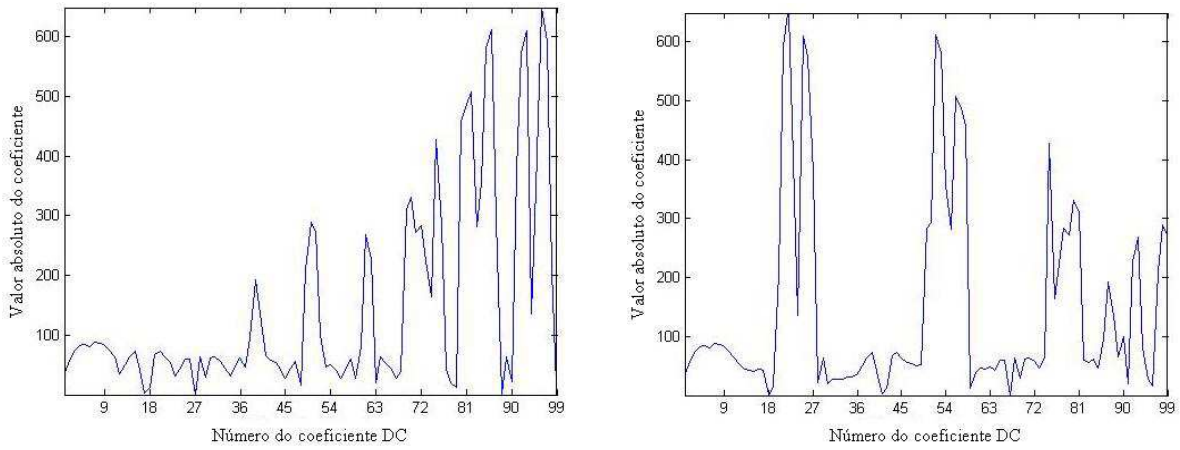
A mesma comparação feita na Fig. 2.4.2 para a leitura em linhas e em espiral da componente Y dos valores dos coeficientes DC do bloco de quadros pode ser repetida para as componentes U e V. Os resultados da leitura das seqüências podem ser vistos na Fig. 2.4.4 e na Fig. 2.4.5. Assim como para a componente Y, percebe-se uma redução na variação dos valores dos coeficientes quando estes são lidos de forma espiral.

As componentes U e V apresentam quatro vezes menos valores DC do que a componente Y porque a amostragem da seqüência foi feita no padrão YUV 4:2:0. Sendo assim, tem-se 88 colunas por 72 linhas nestas componentes, resultando em $\frac{88}{8} * \frac{72}{8} = 11 * 9 = 99$ cubos de coeficientes 8x8x8 para cada bloco de 8 quadros da seqüência de vídeo. O mesmo tamanho de cubo 8x8x8 é utilizado para as componentes U e V porque não há sub-amostragem destas componentes na dimensão temporal, o que impede a utilização de cubos 4x4x4 que refletiriam a sub-amostragem das dimensões espaciais. Neste caso, seria necessária a utilização de uma estrutura em forma de paralelepípedo com tamanho 4x4x8, porém a forma de implementação adotada para o FHVC só permite a utilização de cubos.



(a) (b)

Fig. 2.4.4 – Leitura por linhas (a) e em espiral (b) da componente U dos coeficientes DC do bloco de quadros.



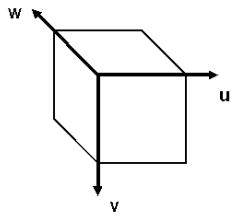
(a) (b)

Fig. 2.4.5 - Leitura por linhas (a) e em espiral (b) da componente V dos coeficientes DC do bloco de quadros.

2.4.2 Ordem de leitura dos coeficientes AC

De acordo com [18], a energia dos coeficientes AC está concentrada nos valores de mais baixa freqüência do cubo que correspondem aos seus eixos. O gráfico da Fig. 2.4.6 é apresentado em [18] para comprovar esta afirmação e corresponde à leitura dos 511 coeficientes AC de um cubo 8x8x8 do bloco de quadros 9-16 do arquivo de vídeo “Football”. Os coeficientes apresentados no artigo foram gerados através da aplicação da DCT e seus números foram

definidos através da Eq. (2.19), onde *u* corresponde às colunas do cubo, *v* corresponde às linhas e *w* corresponde aos quadros:



$$\text{Número do coeficiente AC} = u * 64 + v * 8 + w. \tag{2.19}$$

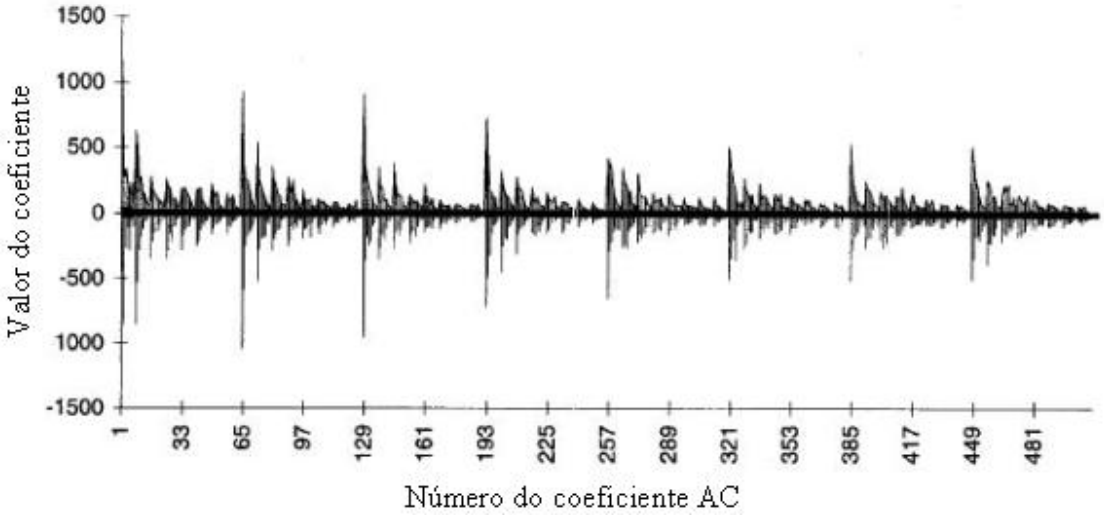


Fig. 2.4.6 – Leitura dos coeficientes AC de um cubo do arquivo de vídeo "Football" [18].

Portanto, a ordem de leitura dos coeficientes é feita por quadros e agrupa todos os coeficientes correspondentes à mesma linha e coluna dos vários quadros do cubo.

Percebe-se, através da análise do gráfico da Fig. 2.4.6, que realmente existe uma periodicidade no sinal, com picos grandes a cada 64 coeficientes. Cada um destes picos corresponde à leitura de um coeficiente do eixo de colunas *u* do cubo (para $v = w = 0$), o que comprova que os coeficientes AC de maior valor estão realmente localizados neste eixo. Também podem ser observados picos menores, com periodicidade de 8 coeficientes, que correspondem aos coeficientes lidos no primeiro quadro do cubo ($w = 0$) quando ocorre um incremento no valor da linha que está sendo lida. Isto também indica que há uma concentração de energia dos coeficientes AC no eixo de linhas do cubo.

Conforme já mencionado, o gráfico apresentado na Fig. 2.4.6 apresenta coeficientes gerados através da DCT. Sabe-se que a DCT apresenta uma compactação de energia maior do que a transformada de Hadamard utilizada no FHVC. Sendo assim, para se validar as conclusões de concentração de energia dos coeficientes AC nos eixos do cubo, foram gerados gráficos para os coeficientes obtidos com a transformada de Hadamard lidos da mesma forma que os coeficientes da DCT através da Eq. (2.19). O gráfico com os coeficientes AC gerado para o mesmo bloco de quadros utilizado na Seção 2.4.1 anterior para a análise dos coeficientes DC é apresentado na Fig. 2.4.7. O cubo lido foi retirado do meio do bloco de quadros, tendo o canto superior esquerdo posicionado na coluna e na linha 80 e sendo identificado, portanto, como o cubo 10x10, uma vez que os cubos têm tamanhos 8x8x8.

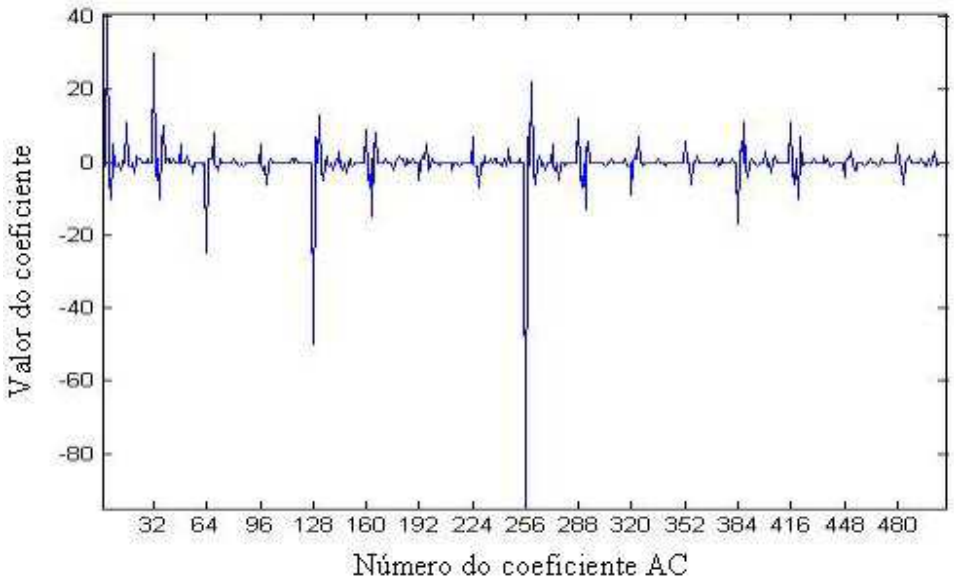


Fig. 2.4.7 – Leitura dos coeficientes AC de um cubo do arquivo de vídeo "Miss America".

A análise do gráfico da Fig. 2.4.7 sugere que a energia dos coeficientes AC da transformada de Hadamard do cubo apresenta uma distribuição mais heterogênea do que a distribuição de energia dos coeficientes AC da DCT apresentada na Fig. 2.4.6. Mesmo assim pode ser identificada uma certa periodicidade, com picos a cada 64 coeficientes aproximadamente, principalmente no início da seqüência. De fato, os maiores picos de energia correspondem aos coeficientes localizados nas regiões de freqüências baixas do cubo, especialmente no eixo de colunas u do cubo (para $v = w = 0$). Além disso, a amplitude dos

coeficientes realmente diminui com o aumento do número do coeficiente, o que comprova a pouca concentração de energia do cubo de coeficientes de Hadamard nas regiões de alta freqüência.

Sendo assim, procurou-se desenvolver para o FHVC uma ordem de leitura dos coeficientes AC que explorasse esta concentração de energia nos coeficientes de baixa freqüência, de forma que os coeficientes fossem lidos de maneira aproximadamente decrescente ou que, pelo menos, os coeficientes com valores altos e médios similares fossem lidos de forma agrupada, evitando variações bruscas e obtendo regiões de valores uniformes na seqüência. Os cubos 8x8x8 foram divididos em oito regiões conforme a Fig. 2.4.8 (a oitava região está encoberta pelas outras na figura). Por apresentar a maior concentração de energia, a região do cubo 1 foi novamente dividida em oito regiões gerando sub-cubos de tamanho 2x2x2. Quando a codificação é realizada com cubos de tamanho inicial 4x4x4, obviamente, realiza-se apenas uma divisão do cubo em oito regiões, a qual já gera sub-cubos de tamanho 2x2x2.

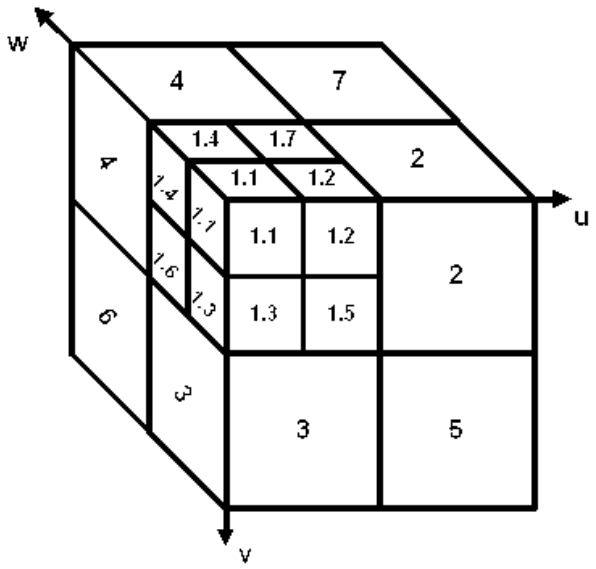


Fig. 2.4.8 – Divisão em oito regiões implementada no FHVC para o cubo de coeficientes.

De forma a explorar também a correlação existente entre valores de coeficientes AC localizados na mesma posição em cubos vizinhos, os coeficientes correspondentes às posições de maior concentração de energia do cubo foram lidos de forma espiral (assim como feito para os coeficientes DC). Sendo assim, inicialmente os valores DC dos cubos são lidos, conforme apresentado na Seção 2.4.1 anterior. Em seguida, são lidos os coeficientes AC dos sub-cubos 1.1,

1.2 e 1.3 localizados nos eixos de linha e coluna de forma alternada. Conforme mencionado, esta leitura também é feita de forma espiral, ou seja, primeiro o coeficiente (0,0,1) de todos os cubos do bloco de quadros é lido de forma espiral, depois o coeficiente (0,1,0) de todos os cubos do bloco é lido seguido da leitura do coeficiente (0,0,2) de todos os cubos e assim por diante. A notação (*w,v,u*) utilizada corresponde a (quadro, linha, coluna). O próximo coeficiente a ser lido, também de forma espiral para todos os cubos do bloco, corresponde ao coeficiente de maior valor do segundo quadro do bloco, situado na posição (1,0,0). A leitura dos coeficientes de maior valor do cubo encerra-se com os quatro coeficientes localizados no eixo principal do sub-cubo 2 e dos quatro coeficientes localizados no eixo principal do sub-cubo 3, seguidos dos quatro coeficientes restantes do sub-cubo 1.1.

Os demais coeficientes, que ainda não foram lidos, dos sub-cubos 2 e 3 e os coeficientes do sub-cubo 4 geralmente apresentam valores médios e terão formas de leitura particulares que priorizam a leitura dos coeficientes próximos aos eixos principais do cubo. Estes coeficientes aparecem de forma destacada nos sub-cubos ilustrados na Fig. 2.4.9. A leitura destes coeficientes também é feita de forma espiral para todos os cubos do bloco de quadros.

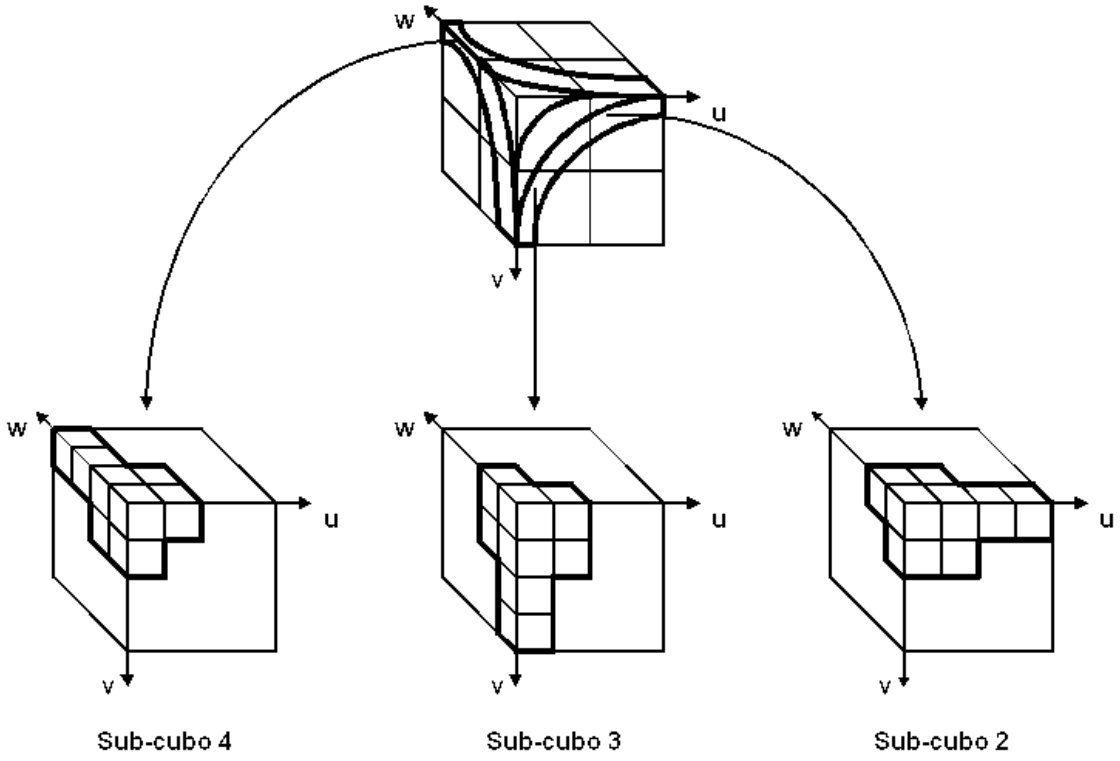


Fig. 2.4.9 – Regiões de coeficientes de maior valor dos sub-cubos 2, 3 e 4.

As ordens de leitura dos coeficientes dos sub-cubos 2, 3 e 4 foram implementadas no FHVC de forma recursiva, ou seja, podem ser utilizadas para sub-cubos de tamanho 4x4x4 ou 2x2x2. Isto evita o desenvolvimento de uma forma de leitura específica para cada tamanho de cubo. Sendo assim, as ordens de leitura também são utilizadas para a leitura dos coeficientes dos sub-cubos 1.2, 1.3 e 1.4 que ainda não foram lidos por não estarem nos eixos principais do cubo.

Já as regiões de coeficientes de alta freqüência do cubo foram lidas na seguinte ordem: 1.5, 1.6, 1.7, 1.8, 5, 6, 7 e 8. Todas estas regiões foram lidas de acordo com a mesma ordem, também desenvolvida de forma recursiva. Esta ordem é feita por colunas, seguida da variação de linhas e, finalmente, de quadros.

Os coeficientes de alta freqüência dos vários cubos não são lidos de forma espiral por coeficiente, assim como implementado para os coeficientes de baixa e média freqüência. Estes coeficientes são lidos de forma agrupada por sub-cubo e espiral por cubo, ou seja, todos os coeficientes do sub-cubo 1.5 de um cubo são lidos conjuntamente, depois todos os coeficientes do sub-cubo 1.5 do cubo vizinho (determinado de forma espiral) são lidos e assim por diante. Esta forma de leitura dos coeficientes de alta freqüência foi realizada de forma agrupada, porque segundo consta em [21], a variância dos coeficientes tende a decair conforme a freqüência aumenta. Desta forma, assim como observado para os coeficientes de baixa freqüência, os valores dos coeficientes de alta freqüência tendem a apresentar uma alta correlação com os valores dos coeficientes vizinhos, ou seja, se um coeficiente tem valor alto, espera-se que seu vizinho também tenha um valor alto e que a probabilidade de se ter coeficientes de alta freqüência com valores altos isolados seja baixa.

Conclui-se, portanto, que a forma de leitura dos coeficientes implementada no FHVC não somente organiza os coeficientes de baixa e média freqüência em ordem decrescente de variância, mas também agrupa os coeficientes de alta freqüência de cada cubo, mantendo a alta correlação ainda existente entre eles.

A Fig. 2.4.10 apresenta o resultado da leitura de todos os 202.752 coeficientes dos 396 cubos 8x8x8 dos quadros de #64-#71 da seqüência de vídeo “Miss America”. As regiões dos coeficientes correspondentes aos vários sub-cubos também são apresentadas. Os primeiros 396 coeficientes do gráfico da Fig. 2.4.10 correspondem aos valores dos coeficientes DC do bloco de quadros. A fim de se analisar de forma mais detalhada a ordem de leitura destes coeficientes, a figura também apresenta um gráfico somente com os primeiros 6.500 coeficientes lidos.

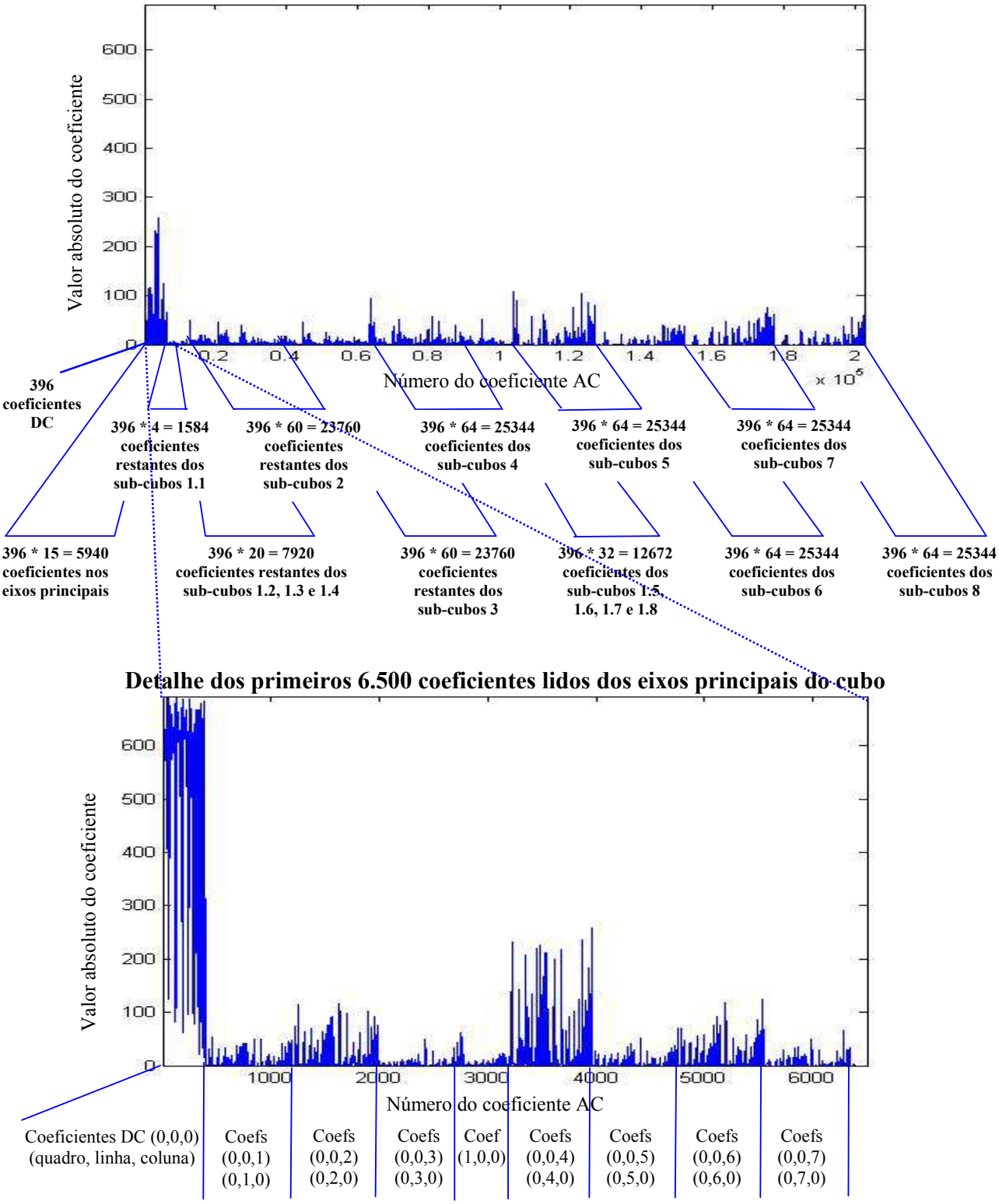


Fig. 2.4.10 – Ordem de leitura do FHVC da componente Y dos coeficientes dos quadros #64-#71 da seqüência de vídeo "Miss America".

Através do gráfico da Fig. 2.4.10 percebe-se que a seqüência de quadros analisada apresenta boa quantidade de energia nos coeficientes de alta freqüência. Isto pode ser evidenciado pelos altos valores apresentados pelos coeficientes lidos nos sub-cubos 5 e 6 e também pelos valores lidos nos eixos principais dos sub-cubos 2 e 3 que são até maiores do que os valores lidos nos eixos principais dos sub-cubos 1.1, 1.2 e 1.3. Este espalhamento de energia pode ser atribuído ao fato da seqüência de quadros analisada apresentar certa quantidade de movimento, conforme comentado na Seção 2.4.1.

Uma vez que os valores dos coeficientes DC apresentados na Fig. 2.4.10 são muito superiores aos valores dos coeficientes AC, estes são visualizados na figura de maneira muito reduzida. Para que a distribuição destes valores possa ser melhor analisada, a Fig. 2.4.11 apresenta somente o resultado da leitura dos coeficientes AC.

Como forma de comparação e avaliação da ordem de leitura implementada no FHVC, a Fig. 2.4.12 apresenta um gráfico gerado através da leitura linha por linha dos mesmos cubos lidos na Fig. 2.4.11.

A análise dos gráficos da Fig. 2.4.11 e da Fig. 2.4.12 comprova que a ordem de leitura implementada no FHVC e descrita nesta seção realmente agrupa, na média, os coeficientes de maior valor no início da seqüência. Também poderiam ser implementadas formas de ordenação absolutas dos coeficientes, porém, até mesmo os algoritmos mais rápidos de ordenação apresentam complexidade da ordem de $O(n)$ (onde n é o tamanho da seqüência, que é de 202.752 para o caso analisado). Além disso, formas de ordenação dependentes dos valores dos coeficientes exigem que a ordem utilizada seja transmitida ao decodificador, resultando em *overhead* no arquivo codificado.

Toda a análise apresentada nesta seção foi feita para a componente Y da seqüência de vídeo “Miss America”. Os gráficos da Fig. 2.4.13 e da Fig. 2.4.14 apresentam o resultado da leitura do FHVC da componente U e da componente V. Devido ao padrão YUV 4:2:0 utilizado, estas componentes apresentam quatro vezes menos valores do que a componente Y. Observa-se que as conclusões obtidas para componente Y também se aplicam as componentes U e V, uma vez que estas também apresentam maior concentração de energia nos primeiros coeficientes e uma boa quantidade de energia nos coeficientes dos sub-cubos 5 e 6.

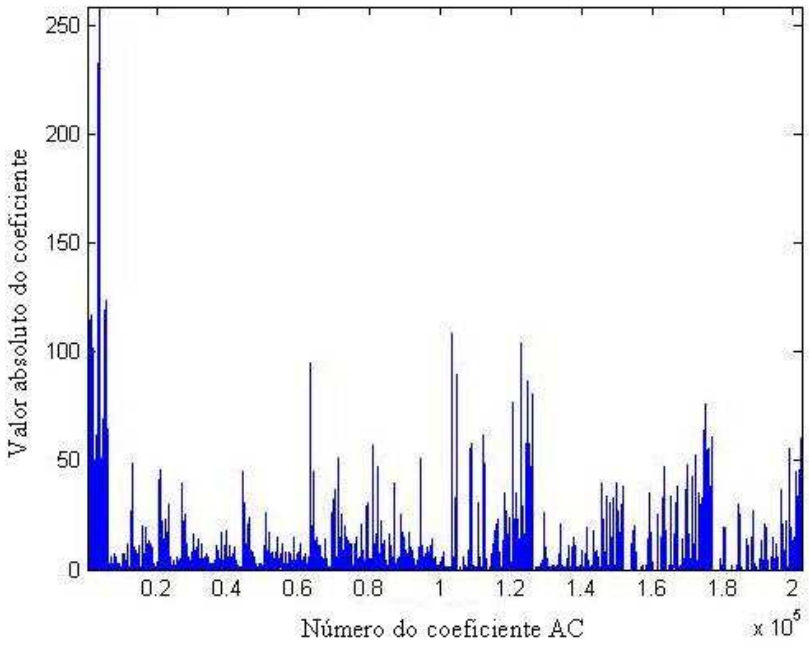


Fig. 2.4.11 – Coeficientes AC apresentados na Fig. 2.4.10.

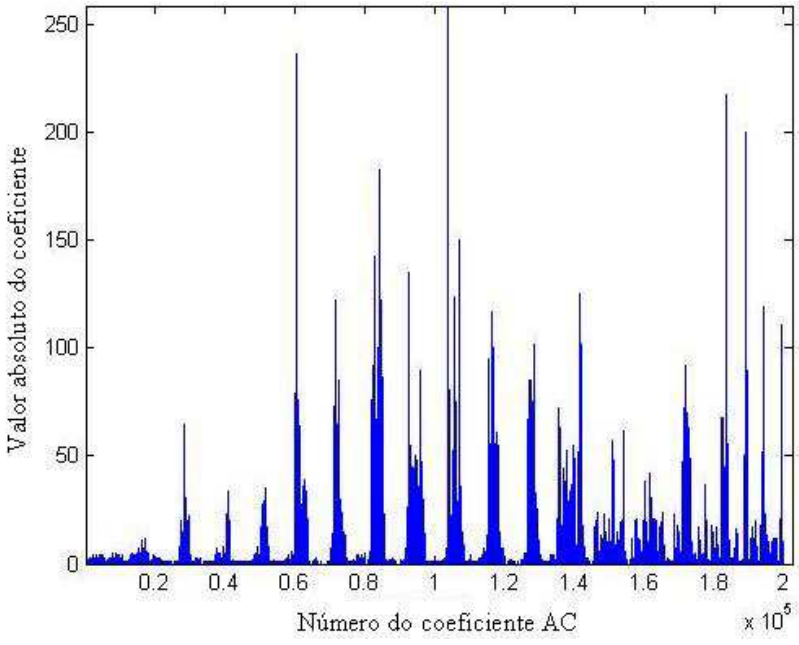


Fig. 2.4.12 – Ordem de leitura por linhas dos coeficientes AC dos quadros 64-71 de "Miss America".

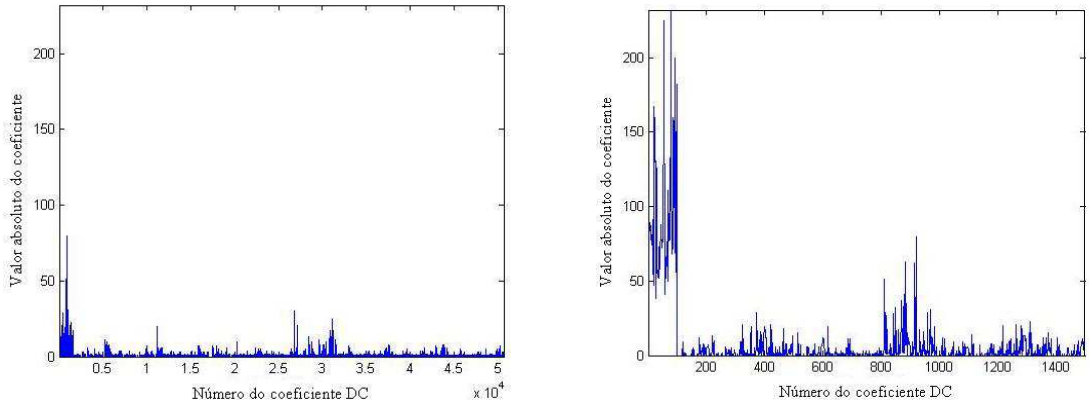


Fig. 2.4.13 – Ordem de leitura do FHVC da componente U dos coeficientes dos quadros 64-71 de "Miss America" e detalhe dos primeiros 1500 coeficientes lidos.

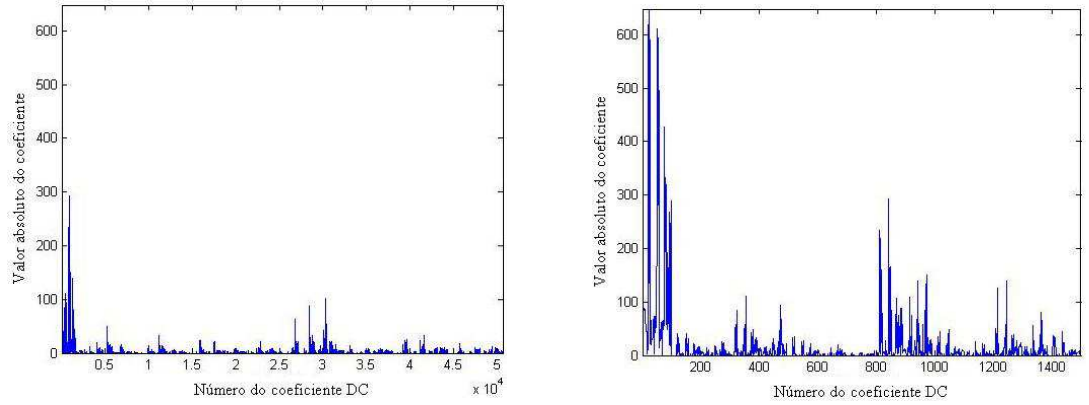


Fig. 2.4.14 – Ordem de leitura do FHVC da componente V dos coeficientes dos quadros 64-71 de "Miss America" e detalhe dos primeiros 1500 coeficientes lidos.

2.5 CODIFICAÇÃO ADAPTATIVA POR CÓDIGO DE GOLOMB

Após o cálculo da transformada 3D de Hadamard descrita na Seção 2.3, os coeficientes de todos os cubos de um bloco de quadros são reorganizados, de acordo com o processo apresentado na Seção 2.4, para formar uma seqüência unidimensional contendo os valores absolutos dos coeficientes e outra seqüência contendo apenas seus sinais. A seqüência com os valores absolutos é, então, passada para a última etapa do processo de codificação implementado no FHVC, que é o codificador de entropia.

A codificação de entropia utiliza um código de comprimento variável para representar de maneira eficiente os valores de uma variável ou de um vetor aleatório discreto através da minimização do comprimento médio da seqüência codificada. Esta forma de codificação é dita

entrópica porque a representação digital da variável ou do vetor aleatório é reduzida a um comprimento médio próximo do valor da sua entropia [21]. O código de comprimento variável se baseia na distribuição de probabilidade do objeto que está sendo codificado, de forma que, palavras curtas são utilizadas para representar símbolos mais freqüentes e palavras longas representam os símbolos mais raros.

Na maioria dos sistemas de codificação de vídeo, a codificação de entropia é realizada após a etapa de quantização dos coeficientes ordenados resultantes da transformada. A quantização reduz o espaço de valores possíveis para os coeficientes, o que resulta em uma maior compactação do arquivo de vídeo. Porém, também é responsável pelas maiores perdas no processo de codificação, uma vez que é irreversível.

O FHVC não realiza a quantização dos valores dos coeficientes antes da codificação de entropia. Sendo assim, o processo de codificação do FHVC pode ser considerado sem perdas e a seqüência de vídeo codificada pode ser totalmente recuperada. Evidentemente, as taxas de compactação alcançadas por um processo de codificação com perdas são maiores do que as alcançadas por um processo sem perdas. Dependendo da aplicação, estas taxas de compactação maiores podem ser até necessárias e as perdas são aceitáveis. Para também poder ser utilizado neste tipo de aplicação, o FHVC apresenta a possibilidade de ser executado com perdas, com o sinal de vídeo sendo reconstruído com ótima qualidade sem que a seqüência codificada seja totalmente decodificada. Isto é possível porque a codificação de entropia é feita no FHVC por plano de bits, ou seja, um plano de bits dos valores absolutos dos coeficientes é codificado a cada iteração do algoritmo, iniciando-se pelo plano de bits mais significativo. Este processo de codificação de entropia por planos de bits gera uma seqüência de bits embutida, onde códigos de taxa menor estão "embutidos" no início da seqüência. Sendo assim, o arquivo de vídeo codificado pode ser decodificado com a taxa desejada. Outra possibilidade é controlar a taxa de bits durante o próprio processo de codificação, o que já possibilita a geração do arquivo codificado com a taxa desejada. A Seção 2.5.3 descreve como a codificação progressiva e o controle de taxa de bits foram implementados no FHVC.

A codificação de entropia pode ser implementada através de vários métodos já consagrados na literatura, tais como o código de Huffman, a codificação aritmética e o código de Golomb. O FHVC implementa a codificação de Golomb adaptativa por plano de bits descrita em [21] e [22], que será apresentada nas Seções 2.5.1 e 2.5.2. Esta forma de codificação utiliza

conceitos de métodos consagrados baseados em transformadas wavelets, tais como o EZW (*Embedded Zerotree Wavelet*) [23] e o SPIHT (*Set Partitioning in Hierarchical Trees*) [24].

2.5.1 Código adaptativo de corrida de zeros de Golomb

O código de Golomb foi concebido para codificar variáveis aleatórias com distribuição de probabilidade geométrica. Uma variável aleatória X com distribuição de probabilidade geométrica é uma variável que assume apenas valores inteiros não-negativos com probabilidade:

$$prob\{X = k\} = (1 - p)p^k, \quad \text{para } k \geq 0. \tag{2.20}$$

Uma seqüência de variáveis randômicas geométricas independentes e identicamente distribuídas (i.i.d) pode ser gerada a partir de uma seqüência de variáveis randômicas de Bernoulli i.i.d. que assumem o valor "0" (fracasso) com probabilidade p e o valor "1" (sucesso) com probabilidade *1-p*. Isto pode ser feito através da contagem do número de fracassos (número de símbolos "0") antes de um sucesso (símbolo "1"). Sendo assim, pode-se caracterizar o código como de corrida de zeros porque são codificadas as seqüências de símbolos "0" que ocorrem entre dois símbolos "1". Conclui-se que existe um mapeamento biunívoco entre seqüências de variáveis aleatórias de Bernoulli e seqüências de variáveis com distribuição geométrica e que o código de Golomb pode ser utilizado para codificar seqüências de qualquer um dos tipos de variáveis [21].

O método de codificação de Golomb permite que ocorram variações na distribuição de probabilidade das variáveis de Bernoulli, desde que sejam lentas e conservem a propriedade de "favorecer" sempre o mesmo resultado (neste caso, o valor "0"). Devido à esta necessidade de variação lenta da distribuição de probabilidade das variáveis é que seqüências mais uniformes, tais como a apresentada na Fig. 2.4.2(b), são melhor codificadas do que as seqüências com maior variância, tais como a apresentada na Fig. 2.4.2(a). Esta distribuição de probabilidade é utilizada no ajuste do principal parâmetro do codificador de Golomb, que é o comprimento do codificador, representado por um número inteiro não negativo ℓ. Para considerar a variação no tempo da distribuição de probabilidade, o parâmetro ℓ é ajustado de forma adaptativa durante o processo de codificação. O algoritmo e as condições em que este parâmetro é ajustado estão descritos em [21]. É importante ressaltar que esta adaptatividade do comprimento do codificador é baseada no

comportamento da própria seqüência de entrada através do cálculo do valor esperado da seqüência de variáveis aleatórias modeladas como geométricas que está sendo codificada. Desta forma, dispensa-se o envio de informações adicionais para o decodificador e evita-se *overhead* na seqüência de vídeo codificada. A implementação necessária para a adaptatividade é rápida porque necessita apenas do ajuste de um parâmetro relacionado ao comprimento do codificador.

O codificador de Golomb com parâmetro ℓ codifica uma corrida de *k* símbolos "0" com uma palavra de código binária formada por três partes:

- uma seqüência de $\lfloor k/\ell \rfloor$ símbolos "0", onde $\lfloor \bullet \rfloor$ denota a parte inteira do argumento. Nota-se que, cada "0" enviado representa uma corrida de ℓ zeros;
- um símbolo "1" para indicar o fim da corrida de zeros, ou seja, o encontro de um símbolo "1" na seqüência que está sendo codificada;
- uma palavra de código para representar o resto da divisão k/ℓ. Esta palavra de código é determinada por um código de Huffman otimizado para distribuições geométricas descrito em [21].

2.5.2 Codificação por planos de bits

O código descrito na seção anterior foi escolhido para ser implementado no FHVC porque se adapta às características da seqüência de dados que precisa ser codificada. Conforme mencionado anteriormente, esta seqüência é formada por cada plano de bits dos valores ordenados dos coeficientes resultantes da transformada 3D de Hadamard. Uma vez que a seqüência está ordenada, apresenta inicialmente bits com valor "1" que se tornam menos freqüentes no decorrer da seqüência, até que esta apresenta apenas bits com valor "0". Identifica-se, portanto, a formação de corridas de zeros e de uma seqüência de variáveis aleatórias binárias independentes cuja distribuição de probabilidade "favorece" um dos dois possíveis resultados da variável (neste caso, o valor "0"). Uma vez que a probabilidade *p* de ocorrência de um símbolo "0" aumenta à medida que os planos de bits dos coeficientes são codificados, o comprimento ℓ do codificador adaptativo também aumenta.

A Tabela 2.5.1 apresenta um exemplo ilustrativo de uma possível seqüência de coeficientes resultantes da transformada 3D de Hadamard ordenados na média. Se os planos de

bits fossem codificados de forma completa, todos os bits do plano de bits mais significativo (identificado como B_{10}) seriam lidos na forma de uma seqüência unidimensional (100100000000) e passados para o codificador de entropia. Em seguida, todos os bits do plano B_9 seriam lidos e codificados e o processo se repetiria até o plano B_1.

Uma forma de melhorar muito esta codificação é observar que a distribuição de probabilidade só "favorece" um dos dois valores ("0" ou "1") para o bit mais significativo de cada coeficiente. Assim, por exemplo, a probabilidade dos valores dos coeficientes da seqüência estarem entre "512" e "1023" (e portanto apresentarem o valor "1" para o bit mais significativo no plano de bits B_{10}) é alta no início da seqüência, mas diminui no decorrer da mesma. Já os demais bits dos coeficientes não apresentam nenhuma correlação com os valores dos bits dos coeficientes vizinhos e podem ser "0" ou "1" com aproximadamente a mesma probabilidade, isto é, 0,5. Desta forma, a probabilidade do primeiro bit (localizado no plano B_{10}) dos primeiros valores de coeficientes da Tabela 2.5.1 ser "1" é bastante alta. Já a probabilidade do segundo bit (localizado no plano B_9) destes coeficientes (que já apresentaram o primeiro bit igual a "1") ser "1" é a mesma de ser "0". Esta análise sugere o envio dos bits de menor significância dos coeficientes sem qualquer codificação.

Valor absoluto do coeficiente	**Valor binário do coeficiente separado em planos de bits**									
	B_{10}	**B_9**	**B_8**	**B_7**	**B_6**	**B_5**	**B_4**	**B_3**	**B_2**	**B_1**
693	1	0	1	0	1	1	0	1	0	1
500	0	1	1	1	1	1	0	1	0	0
325	0	1	0	1	0	0	0	1	0	1
513	1	0	0	0	0	0	0	0	0	1
310	0	1	0	0	1	1	0	1	1	0
217	0	0	1	1	0	1	1	0	0	1
130	0	0	1	0	0	0	0	0	1	0
52	0	0	0	0	1	1	0	1	0	0
28	0	0	0	0	0	1	1	1	0	0
12	0	0	0	0	0	0	1	1	0	0
16	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	1	1

Tabela 2.5.1- Exemplo ilustrativo de uma seqüência de coeficientes ordenada na média.

De forma a explorar as características de correlação entre os planos de bits analisadas anteriormente, a codificação de entropia foi implementada no FHVC de acordo com os seguintes passos:

1. Inicialmente determina-se o número de plano de bits que devem ser codificados como $log_2 (max |coeficiente|)$, ou seja, determina-se o coeficiente de maior valor absoluto da seqüência de coeficientes que está sendo codificada e calcula-se quantos bits são necessários para a sua representação. De forma a evitar uma custosa comparação entre todos os valores da seqüência, o maior valor do coeficiente já é determinado durante a etapa de ordenação dos coeficientes. Sendo assim, o codificador de entropia já recebe da etapa anterior do processo de codificação o número de planos de bits que deve ser codificado.

2. A codificação começa pelo plano de bits mais significativo (B_{10} no exemplo da Tabela 2.5.1). Calculam-se os valores máximo e mínimo para que os coeficientes possuam o primeiro bit “1” neste plano de bits. No caso do plano B_{10}, o valor dos coeficientes com o bit mais significativo neste plano deve estar entre o valor mínimo de “512” (1000000000) e o valor máximo de “1023” (1111111111). Compara-se, então, o valor de cada coeficiente da seqüência com estes valores de máximo e de mínimo. Esta comparação pode gerar um dos seguintes resultados:

    - *o valor do coeficiente é menor do que o valor de limiar mínimo*. Neste caso, o plano de bits mais significativo do coeficiente ainda não foi codificado e este apresenta um bit “0” no plano que está sendo codificado atualmente. Este zero, e todos os zeros consecutivos a ele, serão contados na corrida de zeros até que um bit “1” seja encontrado no plano. Este é o caso, por exemplo, da codificação do plano B_{10} dos coeficientes de valor "500" e "325" apresentados na Tabela 2.5.1. Estes dois coeficientes estão abaixo do valor de limiar mínimo para o plano B_{10}, que é de “512”, e portanto, formam uma corrida de 2 zeros finalizada pelo próximo coeficiente de valor "513", que já está acima do valor de limiar mínimo;

- *o valor do coeficiente é maior ou igual do que o valor de limiar mínimo e menor ou igual do que o valor de limiar máximo*. Neste caso, o primeiro bit “1” deste coeficiente está no plano de bits que está sendo codificado e, portanto, finaliza uma corrida de zeros (que também pode ter comprimento zero) dos coeficientes anteriores. Esta corrida de zeros é, então, codificada pelo codificador adaptativo de Golomb descrito na Seção 2.5.1. Ao final desta codificação, o sinal do coeficiente é transmitido;

- *o valor do coeficiente é maior do que o valor de limiar máximo*. Neste caso, o coeficiente já foi codificado, pois apresentou o primeiro bit “1” em planos de bits anteriores que já foram codificados. O valor do bit que o coeficiente apresenta no plano atual será enviado depois, sem qualquer forma de codificação, juntamente com os demais bits deste plano dos coeficientes que estiverem na mesma situação, conforme explicado no Item 4 a seguir. A situação descrita neste caso, obviamente, nunca ocorre durante a codificação do plano mais significativo, o plano B_{10} da Tabela 2.5.1, uma vez que nenhum coeficiente da seqüência apresentará valor maior do que o valor de limiar máximo deste plano de bits. Já durante a codificação do plano B_9, o primeiro coeficiente, de valor "693", se encaixa neste caso, pois já apresentou o primeiro bit “1” no plano B_{10}. É importante ressaltar que o valor deste bit não é contado nas corridas de zeros, ou seja, o bit “0” que o coeficiente de valor "693" apresenta no plano B_9 não é contabilizado nestas corridas.

3. Todos os valores dos coeficientes sofrem as comparações descritas no Item 2 para a codificação do plano de bits B_{10} e todas as corridas de zeros são codificadas da mesma forma pelo codificador adaptativo de Golomb. Apenas na última corrida de zeros a codificação do resto k / ℓ não é realizada. Isto pode ser feito porque tanto o codificador quanto o decodificador conhecem a quantidade de coeficientes que estão sendo codificados, ou seja, o comprimento das seqüências binárias B_n é finito e conhecido.

4. Após o término da codificação de todos os coeficientes que apresentam o bit mais significativo no plano de bits atual, ocorre o envio dos bits deste plano dos coeficientes

que foram codificados em planos de bits anteriores. Conforme mencionado anteriormente, estes bits de menor significância dos coeficientes apresentam distribuição uniforme e, portanto, são enviados sem qualquer forma de codificação. Como o plano B_{10} é o primeiro, nenhum coeficiente teria sido codificado em planos de bits anteriores e não há envio de bits de menor significância. Sendo assim, o primeiro envio de bits de menor significância ocorre ao final da codificação do plano B_9, onde são enviados o bit "0" que o coeficiente de valor "693" apresenta no plano B_9 e o bit "0" que o coeficiente de valor "513" também apresenta neste plano. Uma vez que os valores dos coeficientes são armazenados na forma decimal e não binária, a detecção do valor do bit do coeficiente em um determinado plano é feita através de uma rápida operação lógica E binária entre o valor do coeficiente e o valor de limiar mínimo do plano. Se o resultado da operação lógica for maior que zero, o coeficiente possui um bit "1" no plano atual. Por exemplo, o valor de limiar mínimo do plano B_9 é de "256". Para se detectar o valor do bit do coeficiente "693" no plano B_9 faz-se o seguinte E binário entre "256" (0100000000) e "693" (1010110101):

$$\begin{array}{cccccccccc} B_{10} & B_9 & B_8 & B_7 & B_6 & B_5 & B_4 & B_3 & B_2 & B_1 & \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 256 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 693 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}, \qquad (2.21)$$

onde o resultado "0" indica que o coeficiente "693" realmente possui um bit "0" em B_9.

5. Se a codificação for realizada sem perdas, todos os planos de bits devem ser codificados e os passos de 2 a 4 precisam ser repetidos até o plano de bits B_1. Se perdas forem aceitáveis, pode-se aumentar a taxa de compactação do codificador através da redução da taxa de bits. O controle da taxa é feito durante o próprio processo de codificação, conforme explicado na Seção 2.5.3.

Todo o processo de codificação de planos de bits implementado no FHVC e descrito nesta seção é apresentado em [22] com a utilização de vetores auxiliares ao processo de codificação.

São mencionados o vetor *v*, que contém os coeficientes não-significativos, e o vetor *s*, que contém os coeficientes significativos e que está inicialmente vazio. Quando o plano de bits do bit mais significativo do coeficiente é codificado, este passa do vetor *v* para o vetor *s*, ou seja, o coeficiente se torna significativo depois que seu bit mais significativo é codificado. Esta forma de implementação, com vetores auxiliares, exige duas estruturas de memória de tamanho igual ao número máximo de coeficientes ao invés de somente uma. Isto ocorre porque os coeficientes são movidos de uma estrutura para outra, mas devem manter a posição da estrutura original.

Devido às restrições de memória existentes para o ambiente inicial de execução do FHVC (conforme comentado no Capítulo 1), optou-se pela não utilização dos vetores auxiliares. Esta decisão foi baseada no fato do tamanho da seqüência de coeficientes ser sempre alto. Até mesmo seqüências de vídeo no formato de baixa resolução QCIF, tais como a utilizada como exemplo na Seção 2.4, já produzem seqüências de 202.752 coeficientes. Desenvolveu-se, então, a forma de implementação rápida do codificador de entropia apresentada nesta seção, que utiliza apenas comparações e operações lógicas, sempre no próprio vetor inicial de coeficientes.

2.5.3 Controle da taxa de bits

A codificação por planos de bits implementada no FHVC também é chamada de codificação progressiva, uma vez que os planos de bits podem ser adicionados progressivamente ao sinal codificado até que a qualidade desejada para o sinal reconstruído seja atingida. Por outro lado, se uma taxa de bits reduzida for mais importante do que a obtenção de um sinal com poucas perdas, é possível estabelecer uma taxa de bits máxima e codificar os planos de bits até que esta taxa seja atingida.

Esta capacidade de codificação progressiva é importante para o FHVC porque os arquivos de vídeo codificados serão transmitidos pela rede de fibra ótica e podem chegar a diversos tipos de receptores, bem como atravessar enlaces de maior ou menor capacidade. Sendo assim, pode ser necessário reduzir o tamanho do arquivo durante a sua transmissão e isto pode ser feito porque a *bitstream* comprimida gerada no FHVC é completamente embutida. Isto significa que um único arquivo codificado de forma embutida para uma seqüência de vídeo pode fornecer vídeo de qualidade progressiva, isto é, o algoritmo de decodificação pode ser finalizado para

qualquer tamanho de arquivo ou pode ser executado até o fim para que uma reconstrução sem perdas seja obtida.

A codificação embutida pode ser comparada à representação binária de precisão finita de números reais [23]. Todos os números reais podem ser representados por *strings* de dígitos binários. Para cada bit adicionado à direita, mais precisão é obtida. Da mesma forma, quanto mais bits da seqüência embutida codificada são processados pelo decodificador, menos distorcida se torna a versão reconstruída do sinal.

O FHVC apresenta três possibilidades de controle e alteração da taxa de bits:

- *Codificar toda a seqüência sem controlar a taxa de bits*. A seqüência comprimida é uma seqüência embutida com todos os planos de bits codificados e o sinal de vídeo é reconstruído sem perdas;

- *Definir a taxa de bits máxima desejada para a seqüência comprimida durante a codificação*. Esta definição deve ser feita entre as opções de taxas de bits apresentadas pelo FHVC no início do processo de codificação[7];

- *Diminuir a taxa de bits da seqüência sem precisar decodificá-la e codificá-la novamente com a nova taxa máxima desejada.* Isto pode ser feito através da funcionalidade de redução de taxa de bits implementada no FHVC[8]. Na tela de redução de taxa de bits, a escolha da nova taxa máxima é livre, ou seja, é possível informar qualquer valor, desde que menor do que a taxa de bits atual do arquivo.

Uma vez que o processo de codificação e decodificação de arquivos de vídeo no FHVC não ocorre em tempo real e é feito a partir de seqüências já armazenadas, é necessário calcular o número máximo de bits codificados permitido para cada bloco da seqüência antes do início do processo de codificação. Este cálculo é feito por blocos porque todo o processo de codificação é tridimensional, e portanto, realizado em blocos de quadros no FHVC. O número máximo de bits permitidos para cada bloco codificado é calculado de acordo com a taxa máxima de bits desejada para o arquivo de vídeo codificado através dos seguintes passos:

[7] As opções de taxas de bits disponibilizadas pelo FHVC são apresentadas na tela mostrada na Fig. 3.4.3.

[8] A tela da funcionalidade de redução de taxa de bits do FHVC é mostrada na Fig. 3.4.7.

1. Calcula-se o número de quadros que deverão ser codificados (considerando os quadros que foram repetidos ao final do arquivo para completar o último bloco).

2. Calcula-se o número de pixels no arquivo de vídeo original de acordo com o espaço de cores interno escolhido, o formato do vídeo e o número de quadros obtido acima.

3. A taxa de bits por pixel deve ser, então, dividida entre as componentes do espaço de cores do arquivo. No FHVC, esta divisão foi feita através de pesos para se considerar a representatividade da informação contida em cada componente. Esta questão foi discutida na Seção 2.2. Sendo assim, para o espaço de cores YUV 4:2:0, a taxa é dividida de acordo com os pesos: 3/4 para a componente Y, 1/8 para a componente U e 1/8 para a componente V. Já para o espaço RGB, a taxa é dividida igualmente com o peso de 1/3 para cada componente. Uma vez que a codificação de cada componente do espaço de cores é realizada separadamente durante a codificação de cada bloco, o número máximo de bits permitido para o bloco é dividido entre as componentes da seguinte forma:

   - *Espaço de cores YUV 4:2:0* - multiplica-se o peso de 3/4 pela taxa de bits por pixel e pelo número de pixels do arquivo de vídeo original calculado anteriormente para se obter o máximo número de bits no arquivo codificado que estarão preenchidos por valores de luminância. Em seguida, multiplica 1/8 pela taxa de bits por pixel e pelo número de pixels do arquivo de vídeo para se obter o mesmo para as crominâncias;

   - *Espaço de cores RGB* – multiplica-se o peso de 1/3 pela taxa de bits por pixel e pelo número de pixels do arquivo de vídeo original para se obter o número máximo de bits no arquivo codificado destinados aos valores codificados de cada componente.

4. Divide-se o número de quadros que serão codificados pelo tamanho que foi adotado para o cubo de codificação. Por exemplo, se o tamanho do cubo que será utilizado é de 8x8x8, o número de quadros será dividido por 8. O resultado é o número de blocos de quadros que devem ser codificados.

5. Divide-se o número máximo de bits no arquivo codificado calculado para cada componente conforme o Item 3 pelo número de blocos de quadros calculado de acordo

com o Item 4. Tem-se, finalmente, o número máximo de bits que a codificação de cada componente de cada bloco pode alcançar.

Desde o início desta seção foram utilizados os termos: taxa *máxima* de bits desejada e número *máximo* de bits permitido por bloco codificado. Isto foi feito porque não é possível atingir uma taxa de bits exata no FHVC devido à forma de controle da taxa de bits. Este controle da taxa de bits ocorre durante a codificação no FHVC em dois momentos:

- Após a codificação dos bits mais significativos de um plano de bits compara-se o número de bits já codificados com o número máximo permitido para a componente de cor e bloco que estão sendo codificados. Se aquele for superior a este, a codificação do último plano de bits significativos é descartada e o processo é finalizado;

- O mesmo procedimento é realizado após a codificação dos bits de menor significância de um plano de bits. Observa-se que o descarte da codificação dos bits de menor significância de um plano não gera tanta distorção na reconstrução do arquivo de vídeo do que o descarte da codificação dos bits mais significativos de um plano de bits. A razão é que os bits de menor significância são apenas bits de refino de coeficientes que já tiveram, pelo menos, o bit mais significativo codificado. Neste caso, já se tem uma aproximação da magnitude do valor destes coeficientes. A reconstrução do valor de coeficientes que não tiveram nenhum bit codificado será analisada em detalhes na Seção 2.5.4.

Uma vez que o número de bits codificados para cada bloco do arquivo de vídeo é apenas limitado por um limiar de máximo, o decodificador não sabe e não tem informações suficientes para calcular o valor exato de bits que foram enviados como resultado da codificação de cada bloco. Ressalta-se também que o codificador de entropia gera códigos de tamanho variável e que, desta forma, a taxa de compactação de cada bloco varia dependendo do seu conteúdo. Sendo assim, é necessário armazenar o número de planos de bits e o número de bits codificados por bloco no arquivo de vídeo codificado. Esta informação gera um *overhead*, por bloco, de 1 byte para o armazenamento do número de plano de bits mais 4 bytes para o armazenamento do número de bits codificados que devem ser lidos para a decodificação de cada bloco.

2.5.4 Decodificação por planos de bits

Os planos de bits mais significativos contêm mais informação estrutural e são altamente compressíveis, pois estão muito correlacionados, o que gera muita redundância na representação destes planos. Já os planos de menor significância se comportam quase como ruído branco [10]. Como conseqüência, a quantidade de compressão é grande para o plano de bits mais significativo e decai no decorrer da codificação dos demais planos de bits. Desta forma, se houver limite de taxa de bits máxima, geralmente os planos de bits de menor significância não são armazenados no arquivo de vídeo codificado.

A decodificação, ou reconstrução, de um coeficiente do arquivo de vídeo pode ocorrer de duas formas:

- Se o plano de bits do bit mais significativo do coeficiente não foi codificado, significa que o valor do coeficiente é provavelmente bem baixo e por isso, será reconstruído com o valor zero. Caso contrário, a taxa de bits desejada é tão restritiva que nem os coeficientes de maior valor foram codificados e o arquivo de vídeo reconstruído terá baixa qualidade;

- Se somente alguns planos de bits dos coeficientes foram codificados, utiliza-se a média entre o máximo e o mínimo valor que o coeficiente poderia ter. Por exemplo, se um coeficiente tiver valor “45”, seis planos de bits devem ser codificados, uma vez que o valor binário do coeficiente é de “101101”. Se somente os dois primeiros planos de bits forem recebidos e decodificados, o valor reconstruído poderia variar entre “10*0000*” = “32” e “10*1111*” = “47” Sendo assim, o valor é reconstruído pela média entre estes dois valores, calculada como $\left\lceil \frac{32+47}{2} \right\rceil = 40$. A representação binária do valor “40” é “101000”. Percebe-se, então, que a média não precisa ser calculada explicitamente, basta adicionar o bit “1” na posição do primeiro plano de bits que não foi codificado.

Nesta forma de decodificação, cada plano de bits recebido reduz o valor médio do erro relativo ao coeficiente pela metade. Conforme os bits são decodificados, a reconstrução do valor do coeficiente se torna mais precisa, com o intervalo de incerteza se tornando cada vez menor. Esta diminuição do intervalo de incerteza se torna clara com a observação da Fig. 2.5.1. A figura

apresenta uma variável uniformemente distribuída entre –128 e 128. Com a decodificação do primeiro bit, o intervalo de incerteza é reduzido para [-128,0] ou [0,128]. Cada bit decodificado posteriormente reduz os intervalos de incerteza pela metade. É importante observar que no FHVC, o sinal do coeficiente é enviado em um bit logo após a codificação do bit mais significativo do coeficiente, o que já define o intervalo inicial em que este estará localizado.

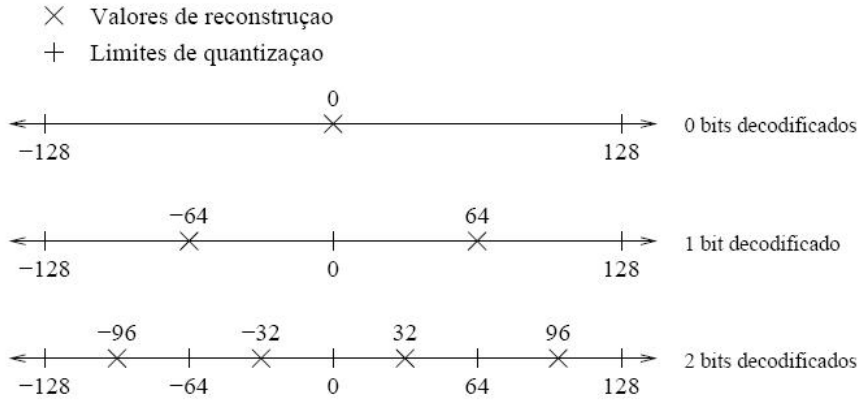


Fig. 2.5.1 – Exemplo de decodificação progressiva [21].

2.5.5 Propagação de erros [21]

Pode-se observar que o método de codificação proposto ainda apresenta um benefício colateral contra os possíveis erros de um canal de transmissão. Tal benefício resulta do modo como os coeficientes da transformada são codificados.

Como os bits de refinamento, isto é, os bits de menor significância transmitidos ao final da codificação de cada plano de bits, são transmitidos sem codificação, os erros que venham a ocorrer durante esta transmissão não se propagam para o resto da seqüência binária. Como conseqüência, os sintomas desses erros na imagem reconstruída representam apenas alterações de magnitude nos pixels que dependem do coeficiente afetado. E essas alterações são da ordem de magnitude correspondente ao bit errôneo. Já os erros que afetam os bits produzidos pelo codificador de Golomb, podem gerar perdas de sincronismo entre o codificador e o decodificador e, assim, produzir defeitos mais graves na imagem reconstruída.

Portanto, um esquema de proteção contra erros para a seqüência comprimida poderia levar em conta estas características e proporcionar níveis de proteção distintos, de acordo com a sensibilidade a erros de cada conjunto de bits.

Capítulo 3
MODELAGEM DO SISTEMA COMPUTACIONAL

Após a apresentação das técnicas de codificação de vídeo utilizadas no sistema FHVC, este capítulo descreve a conversão destas técnicas em um sistema computacional. Antes de ser implementado, o sistema passou por uma fase de modelagem de acordo com os princípios da engenharia de software. Todos os conceitos e abordagens utilizados na modelagem e na implementação computacional do FHVC são apresentados neste capítulo.

3.1 INTRODUÇÃO

O sistema FHVC foi modelado e implementado computacionalmente de acordo com as técnicas de engenharia de *software* orientada a objetos. Esta abordagem foi escolhida porque a orientação a objetos, conforme consta em [28], permite construir modelos que refletem um domínio, tal como um sistema de codificação de vídeo, de uma forma natural, usando a terminologia do domínio. Dessa forma, as técnicas utilizadas no FHVC são modeladas e implementadas computacionalmente utilizando a mesma nomenclatura utilizada no Capítulo 2. Além disso, quando os modelos orientados a objetos estão prontos e corretos, as alterações, expansões, validações e verificações que se fizerem necessárias são facilmente realizadas, pois os modelos são flexíveis. Pode-se, ainda, criar elementos reusáveis e verificar de uma forma mais fácil se os requisitos do sistema estão sendo atendidos no modelo.

A engenharia de *software* orientada a objetos segue os mesmos passos da abordagem estruturada convencional [28]. A primeira fase da modelagem corresponde à análise dos requisitos do sistema. A fase seguinte, chamada de fase de análise, identifica os objetos e classes que são relevantes para o domínio do sistema e os relacionamentos entre elas. Em seguida, a fase de *design* ou projeto define a arquitetura, a interface gráfica e os componentes do sistema. Após o término da fase de projeto, considera-se que a modelagem do sistema está completa e a fase de implementação (utilizando uma linguagem orientada a objetos) transforma o modelo de projeto em código. Finalmente, a fase de testes valida a arquitetura orientada a objetos do sistema, as interfaces e os componentes.

A realização da modelagem de um sistema computacional antes de sua implementação é importante porque melhora a visualização, a especificação, a construção e a documentação da estrutura e do comportamento da arquitetura do sistema [29]. A linguagem UML (*Unified Modeling Language*) foi escolhida para a modelagem porque é uma linguagem padrão e pode descrever qualquer tipo de sistema em termos de diagramas orientados a objetos. Para a realização da modelagem foi utilizado o "*Rational Rose 98 Enterprise Edition"*, uma vez que fornece uma ótima estrutura de diagramas e documentação. O uso de ferramentas de modelagem visuais (tais como o "*Rational Rose*") facilita o gerenciamento da complexidade dos modelos de requisitos, análise e projeto, pois além de expor ou esconder detalhes dos modelos, ajuda a manter a consistência entre os modelos e a implementação.

Este capítulo apresenta os modelos criados para as três fases de modelagem do FHVC mencionadas anteriormente (requisitos, análise e projeto). Também são descritas as formas de implementação e testes utilizadas no sistema.

3.2 MODELO DE REQUISITOS

Para representar os requisitos do sistema foram utilizados diagramas de casos de uso, uma vez que possibilitam uma maneira clara de representar os atores que têm interesse no sistema (pessoas, máquinas e outros sistemas) e a funcionalidade que estes desejam (os casos de uso).

Os requisitos gerais identificados para o sistema FHVC são apresentados na Fig. 3.2.1 e detalhados abaixo:

- **Realização da codificação e da decodificação seqüencial automática do arquivo de vídeo:** o usuário inicia apenas a fase de codificação e a fase de decodificação começa assim que a codificação terminar. A utilização do sistema desta maneira armazena os arquivos de vídeo codificado e decodificado na mesma máquina do arquivo de vídeo original. Portanto, pode-se comparar visualmente os arquivos de vídeo original e decodificado para avaliar subjetivamente a qualidade do codificador;

- **Realização somente da codificação ou da decodificação do arquivo de vídeo:** para os casos em que não se deseja que os arquivos de vídeo codificado e decodificado estejam na

mesma máquina ou que o processo de codificação e decodificação seja realizado de forma seqüencial. Se o arquivo de vídeo for transmitido, por exemplo, a sua codificação será feita no transmissor e a decodificação será feita no receptor. Já se o objetivo for reduzir o espaço necessário para o armazenamento do arquivo de vídeo, a codificação e a decodificação até podem ser realizadas na mesma máquina, mas não de forma seqüencial, uma vez que o arquivo codificado será armazenado e somente será decodificado quando sua visualização for necessária;

- **Redução da taxa de bits por pixel do arquivo de vídeo já comprimido:** um arquivo de vídeo previamente codificado pelo sistema FHVC pode ser ainda mais comprimido sem a necessidade de se realizar o processo completo de decodificação/nova codificação com menor taxa de bits por pixel. Isto é possível devido à característica de codificação progressiva do sistema que torna o arquivo de vídeo codificado uma seqüência embutida de informação (conforme já descrito na Seção 2.5.3). Logicamente, a redução da taxa de bits por pixel reduz a qualidade do arquivo de vídeo decodificado. Porém, muitas vezes esta perda de resolução e de detalhes não é nem percebida pelo olho humano;

- **Obtenção da taxa de relação sinal-ruído do arquivo de vídeo decodificado:** a avaliação qualitativa do codificador é feita através da PSNR (calculada conforme descrito na Seção 4.2). A apresentação desta medida ao usuário é importante pois a partir dela a codificação pode ser realizada novamente com uma taxa de bits por pixel maior ou menor sem a necessidade de se avaliar a qualidade do arquivo de vídeo decodificado visualmente;

- **Modificação das configurações de operação interna do sistema:** a fim de se modificar a taxa de compactação ou a qualidade dos processos de codificação e decodificação, duas configurações internas do sistema podem ser alteradas: o número de quadros por cubo (que pode ser de 4 ou 8, conforme já mencionado na Seção 2.3) e o espaço de cores interno do sistema (que pode ser RGB, YC_oC_g, YUV 4:2:0, YUV 4:2:2 ou YUV 4:4:4). As implicações da utilização de espaços de cores internos ao sistema diferentes dos espaços de cores dos arquivos de vídeo originais foram apresentadas na Seção 2.2. Estas duas configurações de operação interna do sistema não são armazenadas em arquivos de

configuração externos ao sistema, nem persistidas em arquivos de banco de dados. Portanto, estas configurações devem ser realizadas todas as vezes que o sistema FHVC é iniciado.

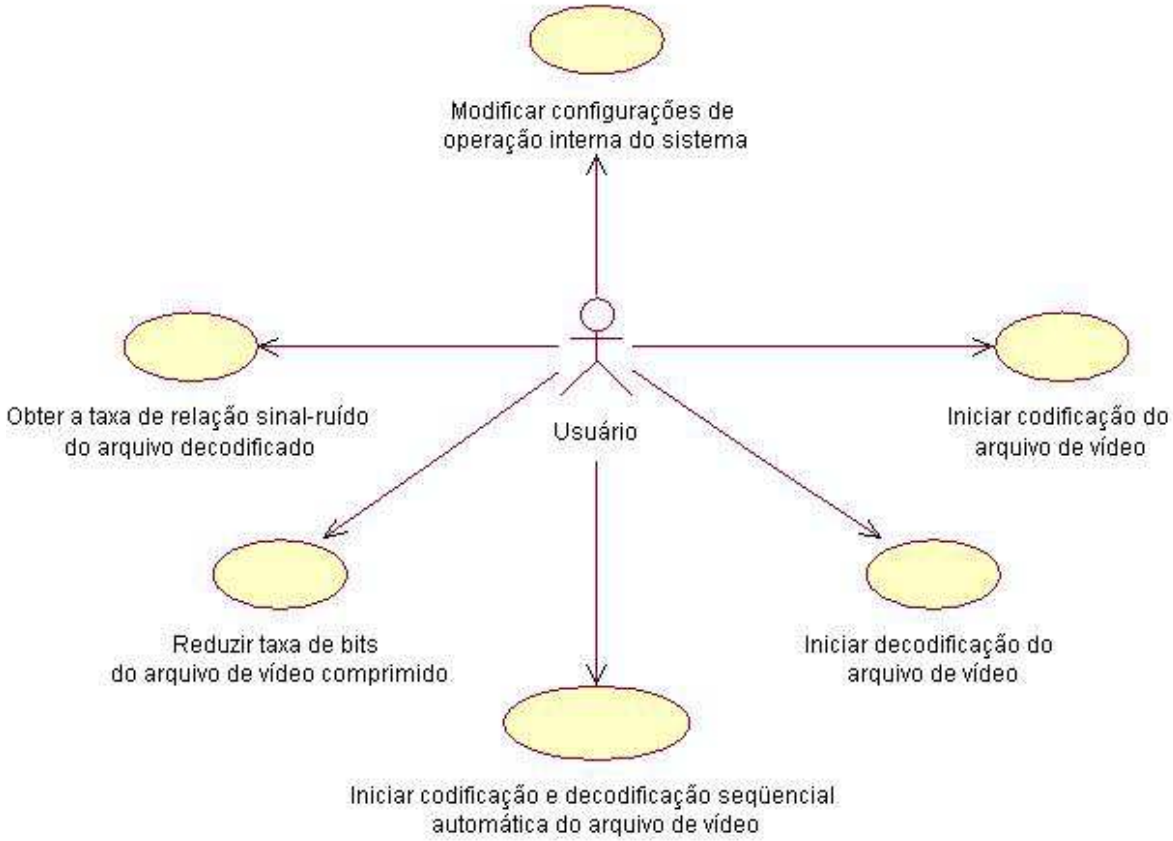


Fig. 3.2.1 - Diagrama de casos de uso do FHVC.

A seguir são mencionados os requisitos identificados especificamente para o processo de codificação (estes requisitos são apresentados na Fig. 3.2.2):

- **Informações sobre o arquivo de vídeo original:** o usuário deve especificar a resolução, o número de quadros e o formato do arquivo de vídeo original. Estas especificações são feitas entre as várias opções apresentadas pelo sistema na tela de Codificação ilustrada na Fig. 3.4.3. Além disso, o usuário deve informar a localização do arquivo de vídeo original;

- **Informações sobre o arquivo de vídeo codificado:** o usuário deve informar a taxa de bits por pixel desejada para o arquivo codificado. Porém, também existe a opção de não se limitar esta taxa. Neste caso, o processo de codificação é realizado de forma completa e a taxa de bits por pixel alcançada é sempre a maior possível. Como na maioria das vezes, é possível se reduzir esta taxa sem perda perceptível de qualidade do arquivo de vídeo decodificado, a mesma tela de Codificação mencionada no item anterior apresenta algumas opções de taxas de bits por pixel que podem ser selecionadas. O arquivo de vídeo codificado sempre será criado com o mesmo nome e na mesma localização do arquivo de vídeo original, porém terá a extensão .fhvc para diferenciá-lo do arquivo original.

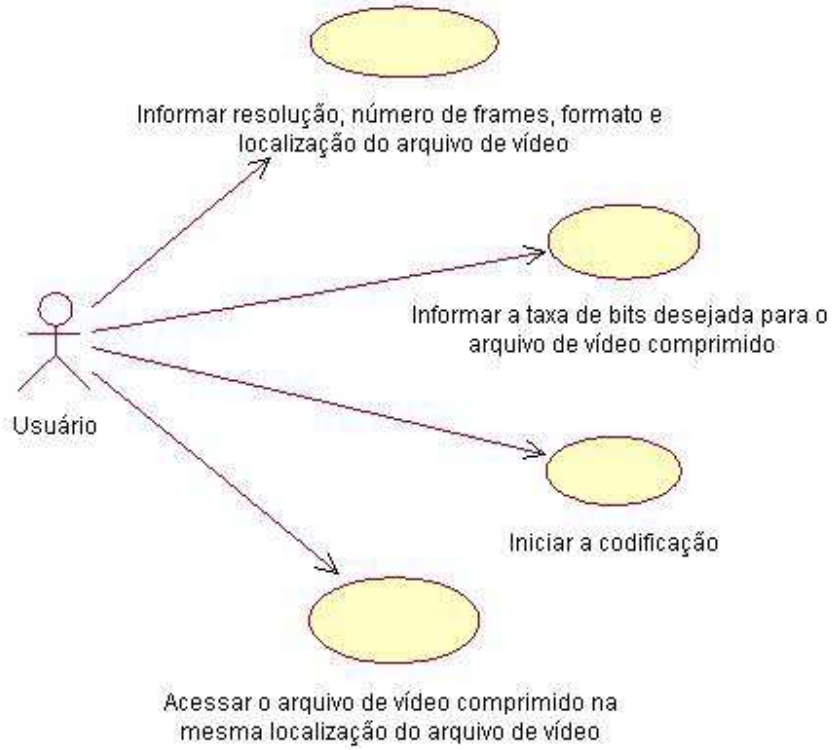


Fig. 3.2.2 - Diagrama de casos de uso da codificação.

Assim como a codificação, a decodificação possui seus requisitos específicos. Estes serão apresentados na Fig. 3.2.3 e detalhados abaixo:

- **Informações sobre o arquivo de vídeo codificado:** o usuário deve especificar apenas a localização do arquivo de vídeo codificado. Assim que esta especificação é feita, o

cabeçalho do arquivo codificado é lido e as informações de resolução, número de quadros, formato e taxa de bits por pixel do arquivo codificado são apresentadas na tela de Decodificação ilustrada na Fig. 3.4.4;

- **Informações sobre o arquivo de vídeo decodificado:** o usuário não precisa especificar a localização do arquivo decodificado ou reconstruído, pois este será criado na mesma localização do arquivo codificado com a identificação "*_Reconstructed*" adicionada ao final do nome do arquivo e com a mesma extensão do arquivo de vídeo original (uma vez que esta informação também é armazenada no cabeçalho do arquivo de vídeo codificado e lida no início da decodificação). Se a identificação "*_Reconstructed*" não fosse adicionada e os arquivos de vídeo original e codificado estivessem armazenados na mesma localização, o arquivo reconstruído sobrescreveria o arquivo de vídeo original porque teria exatamente o mesmo nome e extensão deste.

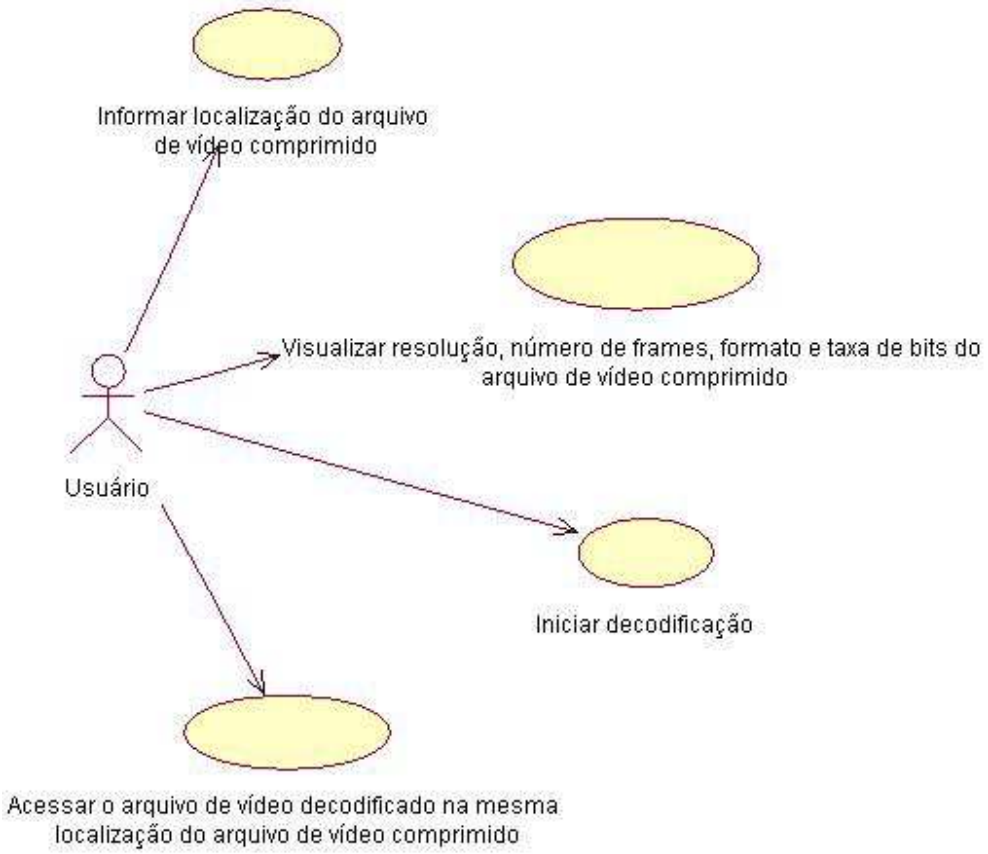


Fig. 3.2.3 - Diagrama de casos de uso da decodificação.

A seguir apresenta-se um exemplo da forma de localização e nomenclatura dos arquivos de vídeo codificado e decodificado adotada pelo sistema FHVC:

Arquivo original: C:\Temp\VideoApresentacaoPalestra.qcif
Arquivo codificado: C:\Temp\VideoApresentacaoPalestra.fhvc
Arquivo decodificado: C:\Temp\VideoApresentacaoPalestra_Reconstructed.qcif

3.3 MODELO DE ANÁLISE

Conforme consta em [28], a análise está completa quando a hierarquia de todas as classes, com seus atributos, operações e relacionamentos, está definida e o modelo comportamental está criado.

Para representar a hierarquia de classes, os relacionamentos entre as classes e as próprias classes foram utilizados diagramas de classes. Para representar o modelo comportamental dinâmico do projeto, pode-se utilizar os seguintes diagramas equivalentes: diagrama de seqüência, colaboração ou atividades em UML. O diagrama de seqüência foi escolhido porque, além de ser melhor suportado pelo ambiente escolhido para a modelagem, o "*Rational Rose*", é o que define o comportamento do sistema da forma mais simples e clara.

3.3.1 Diagramas de classes

Uma classe geralmente é descrita como o modelo ou a forma a partir da qual um objeto é criado e cada objeto deve ser responsável pela realização de um conjunto específico de tarefas relacionadas. Um objeto nunca deveria manipular diretamente os dados internos de outro objeto e toda comunicação entre eles deve ser através de mensagens (chamadas a métodos). O modelo de classes criado para o sistema FHVC, apresentado na Fig. 3.3.1, segue estas orientações. Isto assegura que as vantagens de encapsulamento, modularização e reusabilidade do uso da orientação a objetos serão realmente obtidas.

Uma vez que os modelos orientados a objetos podem refletir o domínio usando a sua terminologia, as classes do sistema FHVC foram criadas e nomeadas de acordo com os vários

elementos do sistema de codificação apresentados no Capítulo 2. Desta forma, além de se obter clareza conceitual no modelo, obtém-se uma boa modularização ou uma separação adequada das várias atividades do processo de codificação entre as classes. A classe *CHadamardFast3DTransform*, por exemplo, realiza apenas a transformação de Hadamard 3D (descrita na Seção 2.3) nos dados que foram lidos do arquivo de vídeo original pela classe *CVideoFile.* Se posteriormente, a utilização de outra transformada for desejada, basta substituir a classe *CHadamardFast3DTransform* por outra que contenha a implementação da nova transformada e a mesma interface de entrada e saída (assinatura dos métodos) para que as outras classes do modelo possam interagir com a nova classe da mesma forma que interagiam com a classe substituída. Já a classe *CVideoFile*, por exemplo, encapsula todas as operações relacionadas à leitura, escrita e conversão de formatos de todos os arquivos de vídeo envolvidos no processo (o original, o codificado, o decodificado e o codificado com taxa de bits reduzida). Se a forma de implementação da conversão de um formato de arquivo em outro for modificada, somente o método desta classe que faz a conversão será modificado.

Todas as classes implementadas para o FHVC estão no diagrama de classes da Fig. 3.3.1 organizadas de acordo com as várias camadas do sistema. Geralmente, sistemas computacionais são divididos em quatro camadas: camada de interface do usuário, camada de formatação de dados, camada da aplicação e camada de persistência. No FHVC, a classe *CFast3DVideoCodecApp* é a principal classe de interface gráfica e gerencia todas as outras seis classes gráficas (identificadas com o prefixo *Cfrm*). Na camada de formatação de dados, a classe *CFast3DVideoCoder* prepara dados e gerencia os processos realizados pelas classes da camada da aplicação (*CHadamardFast3DTransform, CHadamard3DCoefficientsOrdering e CGolombAdaptativeEncoding*). Finalmente, na camada de persistência, a classe *CVideoFile* é responsável pelas operações de leitura/escrita de todos os dados externos ao sistema.

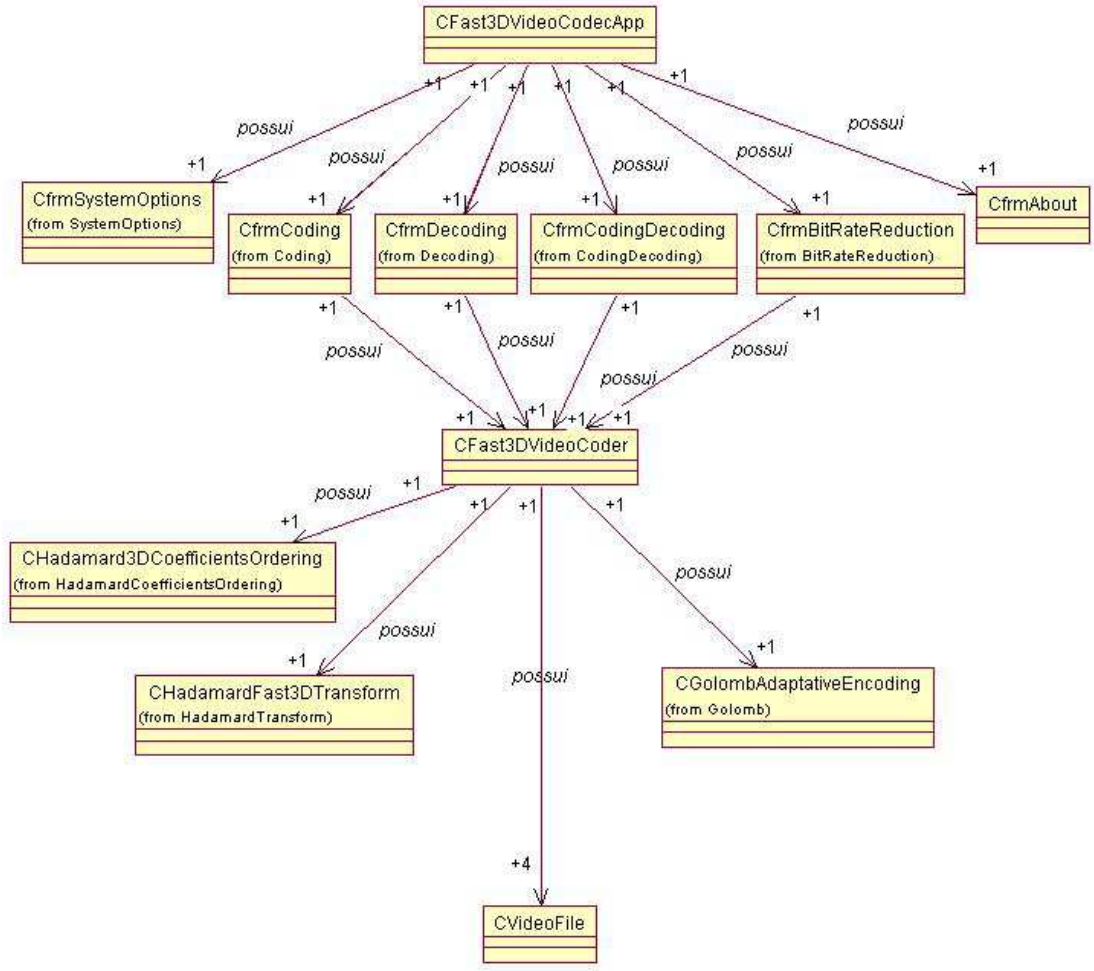


Fig. 3.3.1 - Diagrama de classes do FHVC.

3.3.2 Diagramas de seqüência

Os objetos (instâncias das classes dos modelos orientados a objetos) possuem atributos (que definem o estado de cada objeto) e métodos (que definem o comportamento do objeto e, portanto, a forma como seus atributos são modificados). Para possibilitar a visualização de forma global e limpa, o diagrama de classes do FHVC apresentado na Fig. 3.3.1 não expõe os atributos e métodos de cada classe. Porém, para que o comportamento do sistema seja entendido, os diagramas de seqüência apresentados a seguir contêm algumas chamadas de métodos de classes. Cada uma destas chamadas será detalhada, assim como cada um dos passos dos diagramas.

O diagrama principal de seqüência do FHVC é mostrado na Fig. 3.3.2 e define o passo-a-passo de um processo de codificação e decodificação seqüencial automático de um arquivo de vídeo. O usuário inicia a interação com a aplicação escolhendo a opção "*Coding and Decoding*" do menu principal. A aplicação retorna uma tela com os dados do arquivo de vídeo que devem ser informados pelo usuário. Após preencher estes dados, o usuário pressiona o botão "*Start*" e o controle da aplicação passa para a classe de interface gráfica do processo (*CfrmCodingDecoding*). Esta classe realiza a validação dos dados informados pelo usuário. Se estes não forem válidos, o processo é finalizado e o usuário recebe uma mensagem de texto solicitando a correção dos dados inválidos. Se a validação for bem sucedida, as classes de negócio (relacionadas ao sistema de codificação) são acessadas. A classe *CFast3dVideoCoder* centraliza e coordena todas as tarefas realizadas por estas classes internas. Assim, inicialmente esta classe inicia a codificação do arquivo de vídeo e se esta for finalizada corretamente, inicia imediatamente a decodificação do arquivo de vídeo. Os detalhes operacionais da codificação e da decodificação são apresentados nos diagramas de seqüência específicos destes processos apresentados, respectivamente, na Fig. 3.3.3 e na Fig. 3.3.4. Após o término da decodificação, a classe *CFast3dVideoCoder* calcula a taxa de bits por pixel alcançada para o arquivo codificado e a PSNR obtida para o arquivo decodificado. Estas duas medidas são apresentadas para o usuário e o processo é finalizado. O usuário pode acompanhar o *status* da execução de todos os passos descritos anteriormente na tela da Fig. 3.4.5 que lhe é apresentada.

O diagrama de seqüência específico da codificação é mostrado na Fig. 3.3.3. Após o preenchimento dos dados do arquivo de vídeo original pelo usuário, a codificação é iniciada com a abertura deste arquivo e com a criação do arquivo de vídeo codificado que será preenchido durante o processo. O cabeçalho, cujo formato é descrito posteriormente na Seção 3.4.1, é a primeira informação gravada neste arquivo. Os próximos passos da codificação serão repetidos para cada bloco de quadros do arquivo de vídeo e são detalhados a seguir:

1. Leitura do bloco de quadros (em número de 4 ou 8).
2. Se o espaço de cores do arquivo de vídeo for diferente do espaço de cores interno do codificador, realiza-se a conversão do bloco de quadros para o espaço de cores interno e grava-se o bloco convertido em um novo arquivo de vídeo.

3. Para cada um dos três componentes de cores do bloco aplica-se a transformada 3D de Hadamard, ordenam-se os coeficientes resultantes da transformada e codificam-se estes coeficientes ordenados com o código de Golomb.
4. Gravação do bloco de quadros codificado no arquivo de vídeo codificado.
5. Informação ao usuário de que a codificação do bloco foi finalizada com sucesso.

Quando todos os blocos do arquivo de vídeo original são lidos, a taxa de bits por pixel alcançada para o arquivo codificado é calculada e informada ao usuário.

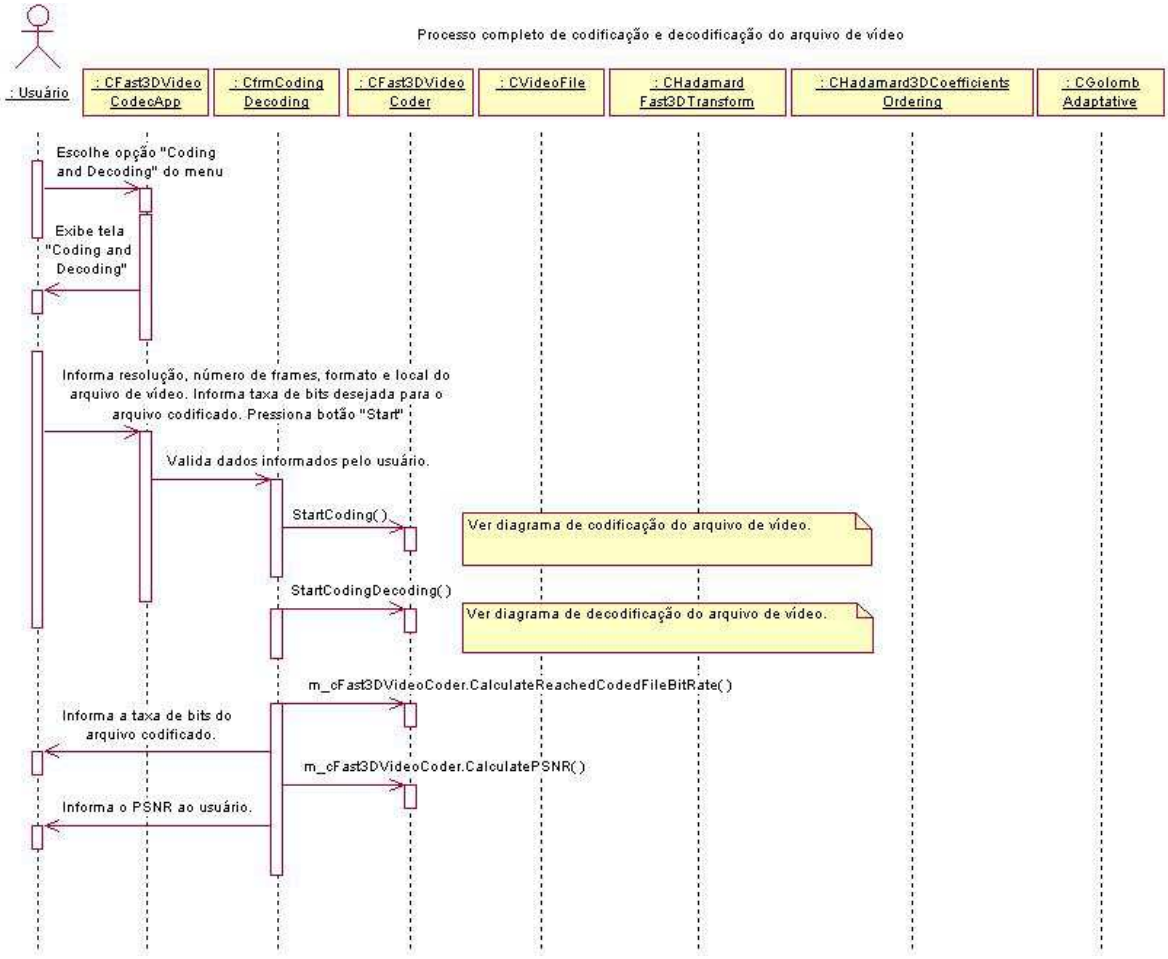


Fig. 3.3.2 - Diagrama de seqüência do FHVC.

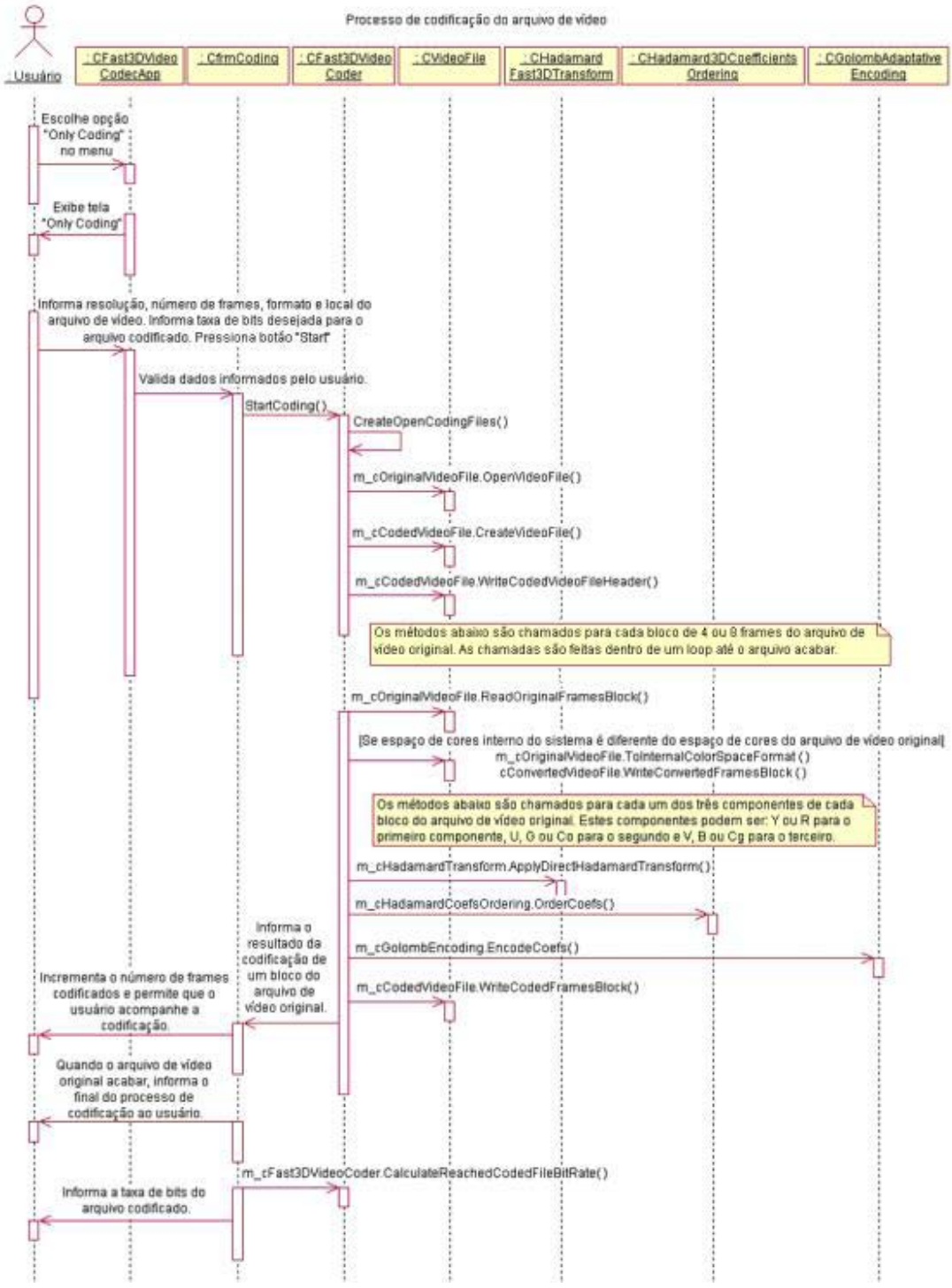


Fig. 3.3.3 - Diagrama de seqüência da codificação.

A Fig. 3.3.4 mostra o diagrama de seqüência específico da decodificação. Inicialmente, os dados do cabeçalho do arquivo codificado informado pelo usuário são lidos e apresentados. A decodificação é iniciada com a criação do arquivo de vídeo decodificado (ou reconstruído) que será preenchido durante o processo. Os passos da decodificação são repetidos para cada bloco de quadros do arquivo de vídeo e para cada um dos três componentes de cores do arquivo de vídeo. Estes passos são detalhados a seguir:

1. Leitura do bloco de quadros codificado.
2. Recuperação dos coeficientes de Hadamard através da decodificação dos coeficientes por Golomb.
3. Restauração da ordem original dos coeficientes (ordem em que os coeficientes de Hadamard foram inicialmente calculados durante o processo de codificação).
4. Aplicação da transformada 3D de Hadamard inversa.
5. Se o espaço de cores do arquivo de vídeo era diferente do espaço de cores interno durante a codificação, realiza-se a conversão do bloco de quadros decodificado para o espaço de cores do arquivo de vídeo original.
6. Gravação do bloco de quadros decodificado no arquivo de vídeo decodificado.
7. Informação ao usuário de que a decodificação do bloco foi finalizada com sucesso.

Quando todos os blocos do arquivo de vídeo são decodificados, a PSNR do arquivo decodificado é calculada e informada ao usuário. O cálculo da PSNR só será possível se o arquivo de vídeo original estiver disponível na mesma localização do arquivo de vídeo codificado.

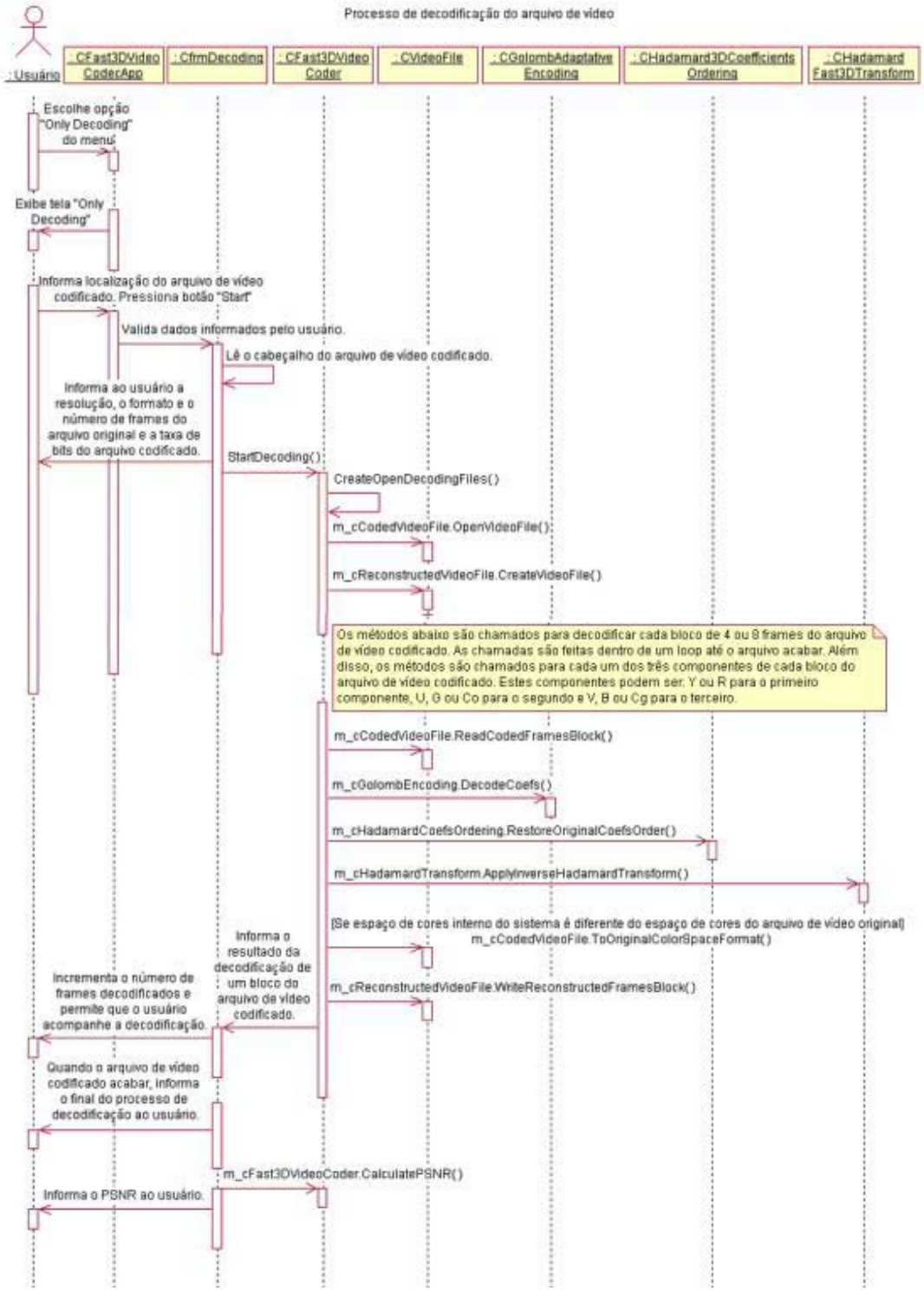


Fig. 3.3.4 - Diagrama de seqüência da decodificação.

3.4 MODELO DE PROJETO

Na fase de projeto, o resultado da análise é expandido em uma solução técnica que será utilizada para a fase de implementação. Primeiramente, algumas definições particulares do sistema FHVC são apresentadas, tais como o formato escolhido para o cabeçalho dos arquivos de vídeo codificados. Em seguida, divide-se o projeto em duas partes: projeto do objeto e projeto do sistema, conforme consta em [28].

A fase do projeto do objeto abrange a definição da interface do usuário através de um *layout* das telas de interface gráfica**.** A fase do projeto do sistema começa com a separação das classes em subsistemas, que também são posteriormente separados em camadas. Um diagrama em camadas foi utilizado para mostrar a divisão destes subsistemas e os relacionamentos entre os mesmos. O projeto do sistema também pode conter um dicionário de dados, que apresenta definições e detalhes de cada classe, de seus atributos e de seus métodos. Por ser bastante extenso, o dicionário de dados do FHVC não foi incluído neste documento.

3.4.1 Formato do cabeçalho

O cabeçalho do arquivo de vídeo codificado pelo FHVC contém informações fundamentais que precisam ser lidas pelo decodificador para reconstruir corretamente o arquivo de vídeo. O cabeçalho contém 23 bytes e deve ser implementado com as informações e ordem apresentadas na Fig. 3.4.1, onde os bytes aparecem numerados.

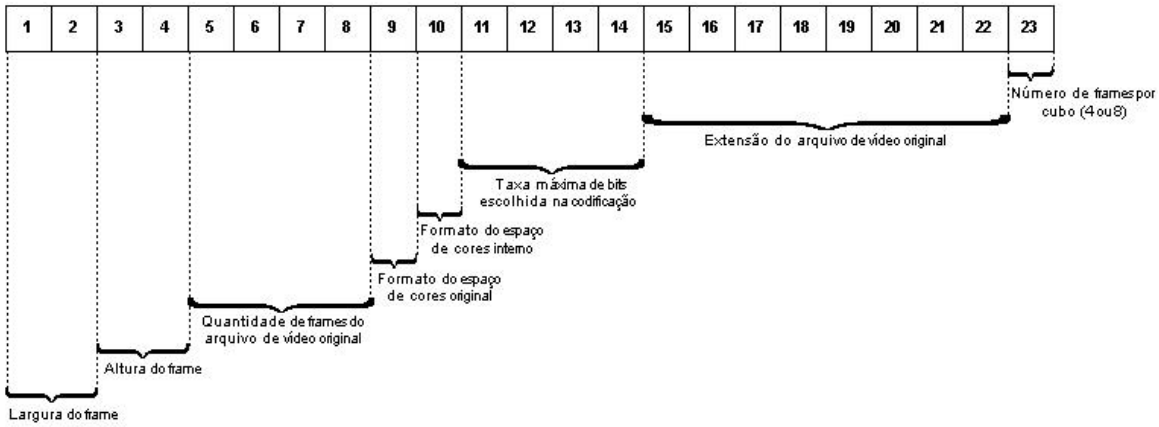


Fig. 3.4.1 - Formato do cabeçalho do arquivo de vídeo codificado no FHVC.

O formato do espaço de cores original é armazenado como 0, se for RGB, e 1 se for YUV 4:2:0. Já o formato do espaço de cores interno é identificado pelos seguintes valores:

- 0 para RGB;
- 1 para YUV 4:2:0;
- 2 para YUV 4:2:2;
- 3 para YUV 4:4:4;
- 4 para YC_oC_g.

A *string* com a extensão do arquivo de vídeo original é armazenada no cabeçalho para que o arquivo de vídeo seja reconstruído com a mesma extensão do arquivo original.

3.4.2 Interface gráfica

A tela inicial do FHVC é apresentada na Fig. 3.4.2. Esta tela apresenta apenas um menu principal e uma região de fundo onde serão abertas as telas escolhidas a partir deste menu.

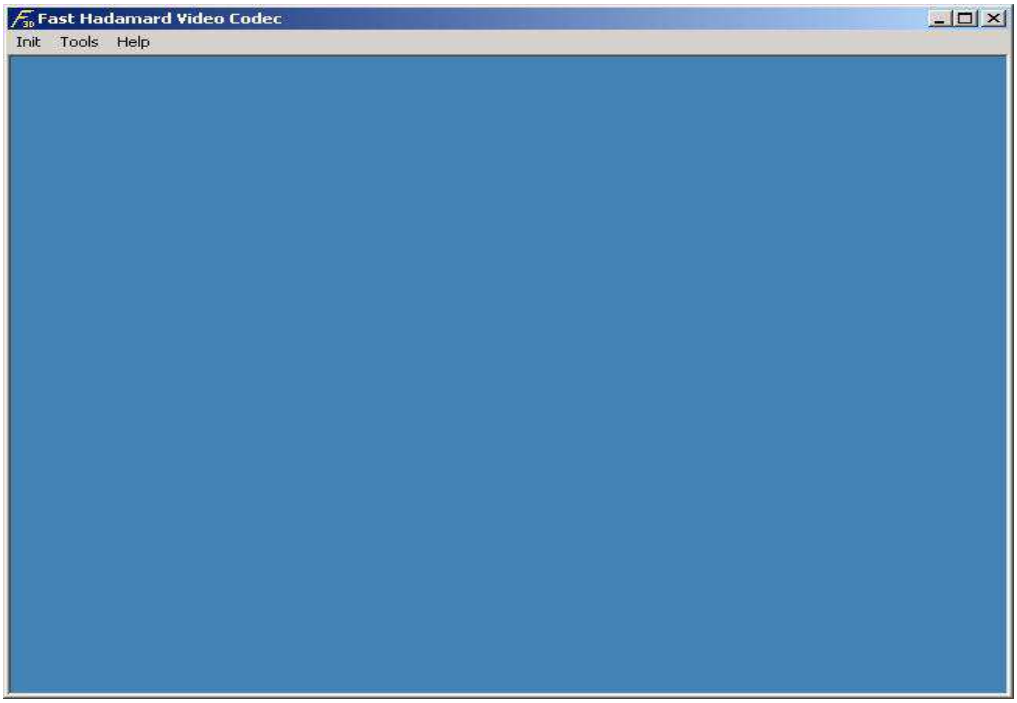


Fig. 3.4.2 - Tela inicial do FHVC.

Através dos seguintes itens do menu principal podem ser acessadas todas as funcionalidades implementadas:

- ***Init -> Only Coding****:* ao se pressionar este item de menu, a tela da Fig. 3.4.3 é apresentada. Todas as informações a respeito do arquivo de vídeo original e do arquivo codificado resultante devem ser preenchidas pelo usuário, conforme já detalhado na Seção 3.2 do modelo de requisitos. Após o preenchimento destas informações, o usuário deve pressionar o botão "*Start*". O processo de codificação é iniciado e o *status* de sua execução pode ser acompanhado na região da tela identificada como "*Coding Status*". O término da codificação de cada bloco do arquivo de vídeo é informado. Assim, por exemplo, se o tamanho do bloco de quadros for de 8, a tela apresentará as seguintes mensagens:

  - "*Frames from 1 to 7 read and coded.*"
  - "*Frames from 8 to 15 read and coded.*"
  - "*Frames from 16 to 23 read and coded.*"

  e assim por diante até o término do arquivo de vídeo, que é a situação apresentada na Fig. 3.4.3. Se a codificação for bem sucedida, uma mensagem informando a localização do arquivo codificado é apresentada e o valor da taxa de bits por pixel alcançada é informado. Se acontecerem problemas durante a codificação, estes também serão informados na região "*Coding Status*";

- ***Init->Only Decoding****:* ao se pressionar este item de menu, a tela da Fig. 3.4.4 é apresentada. O arquivo de vídeo codificado é escolhido e suas informações, lidas a partir de seu cabeçalho, aparecem na tela. O usuário deve, então, solicitar ou não o cálculo da PSNR ao final do processo de decodificação e pressionar o botão "*Start*". O processo de decodificação é iniciado e pode ser acompanhado através das mensagens apresentadas na região "*Decoding Status*", que seguem o mesmo padrão de formação das mensagens já descritas anteriormente para a codificação. Se tiver sido solicitado e o processo de decodificação for finalizado com sucesso, o valor obtido para a PSNR é apresentado ao usuário;

- ***Init -> Coding and Decoding****:* através deste item de menu o usuário pode realizar a codificação e a decodificação de um arquivo de vídeo de forma seqüencial. A tela apresentada na Fig. 3.4.5 é apresentada posicionada na região de fundo do FHVC. O usuário deve preencher todas as informações a respeito dos arquivos de vídeo original, codificado e decodificado e pressionar o botão "*Start*". As mensagens apresentadas seguem o padrão descrito anteriormente. A figura apresenta o *status* de término de uma execução bem sucedida do processo;

- ***Exit:*** fecha a aplicação;

- ***Tools -> System Options****:* as configurações de operação interna do sistema descritas na Seção 3.2 do modelo de requisitos podem ser feitas na tela da Fig. 3.4.6, apresentada ao se pressionar este item de menu;

- ***Tools -> Bit Rate Reduction****:* ao se pressionar este item de menu, a tela da Fig. 3.4.7 é apresentada. O arquivo de vídeo codificado, que deve ter a taxa de bits por pixel reduzida, é escolhido e suas informações aparecem na tela. Após informar a nova taxa de bits desejada (necessariamente menor do que a taxa atual), o usuário deve pressionar o botão "*Start*". O padrão das mensagens de *status* apresentadas segue o mesmo descrito para as mensagens do processo de codificação;

- ***Help -> About***: este item de menu apresenta informações sobre o FHVC, tais como versão, autor e ano de liberação do sistema.

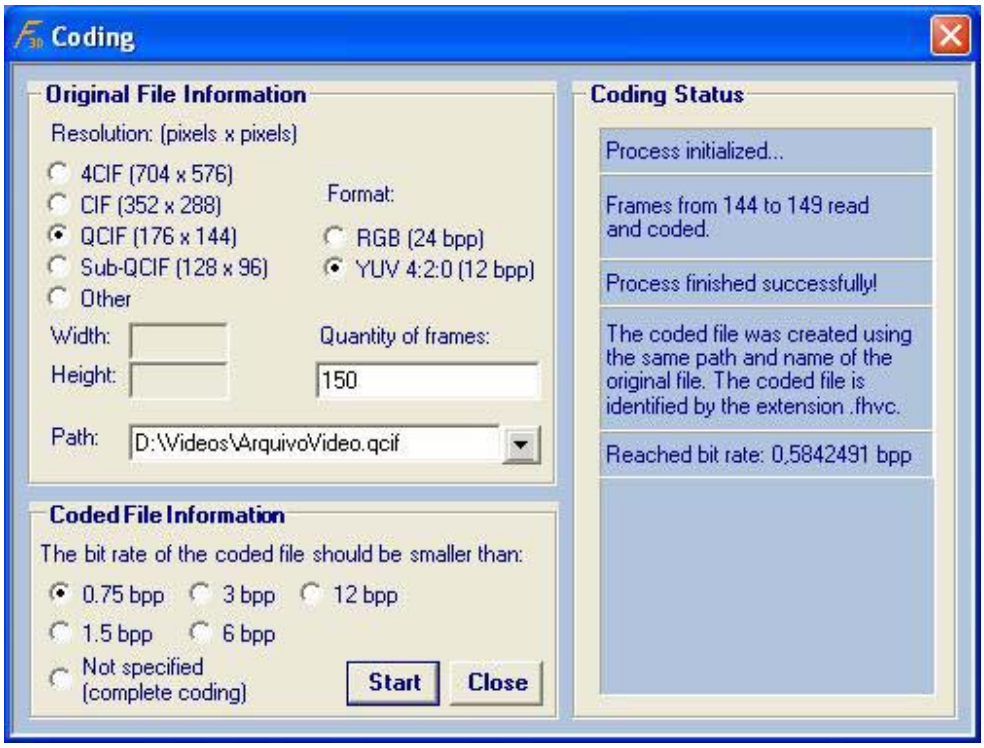


Fig. 3.4.3 - Tela de codificação extraída da tela de fundo.

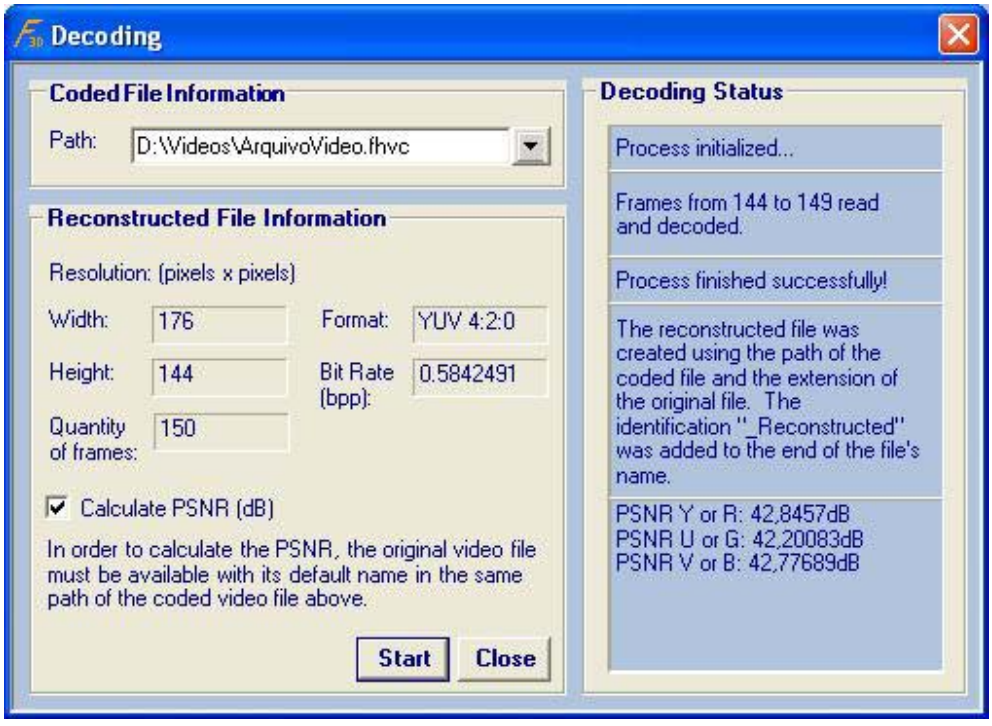


Fig. 3.4.4 - Tela de decodificação extraída da tela de fundo.

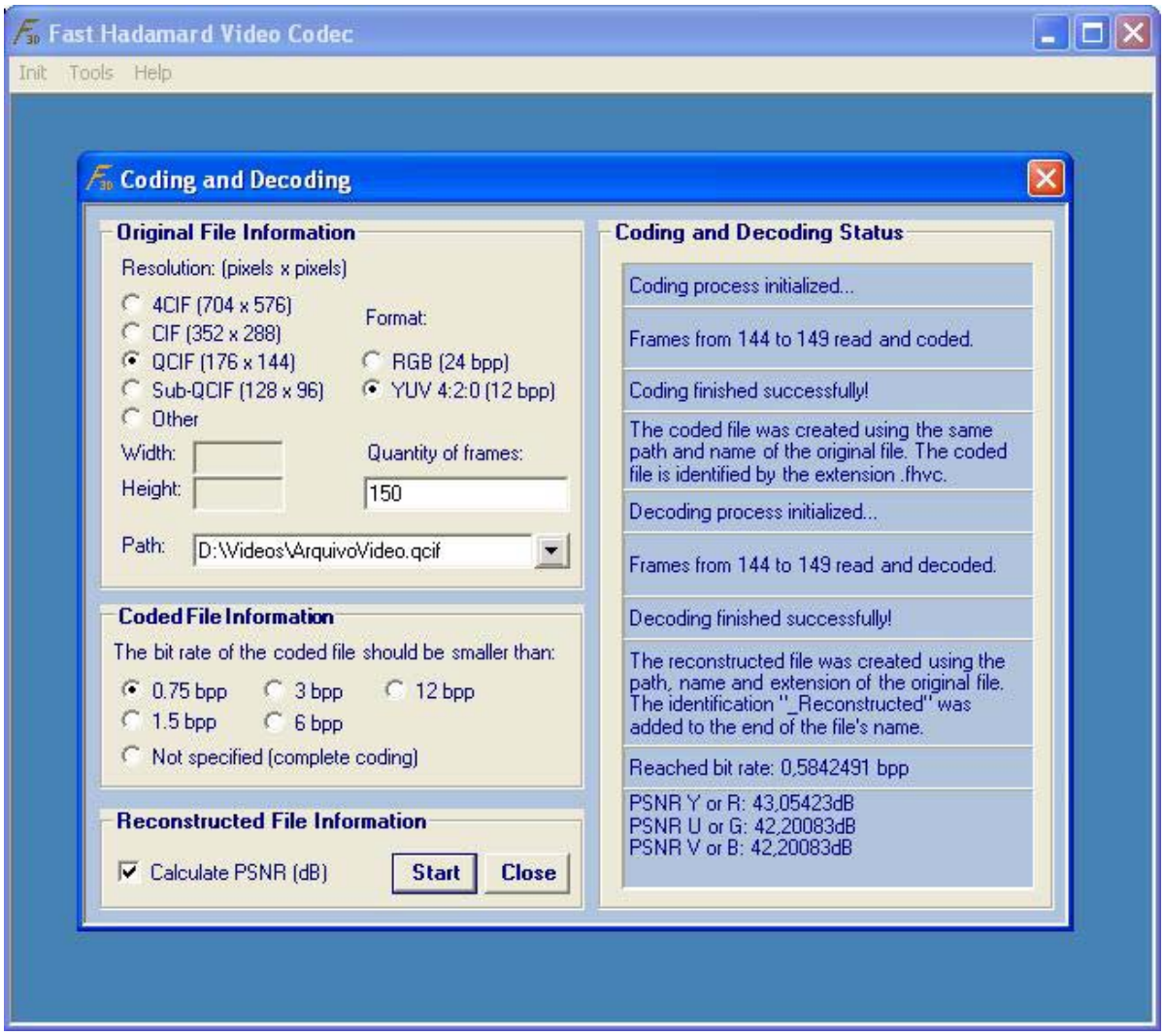


Fig. 3.4.5 - Tela de codificação e decodificação posicionada na tela de fundo principal.

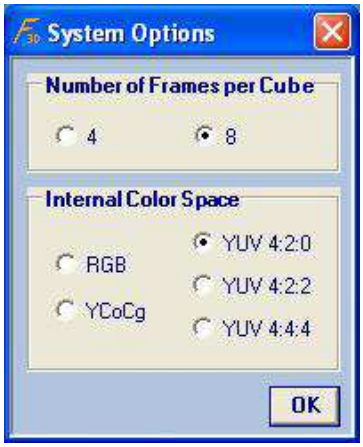


Fig. 3.4.6 - Tela de opções internas do sistema.

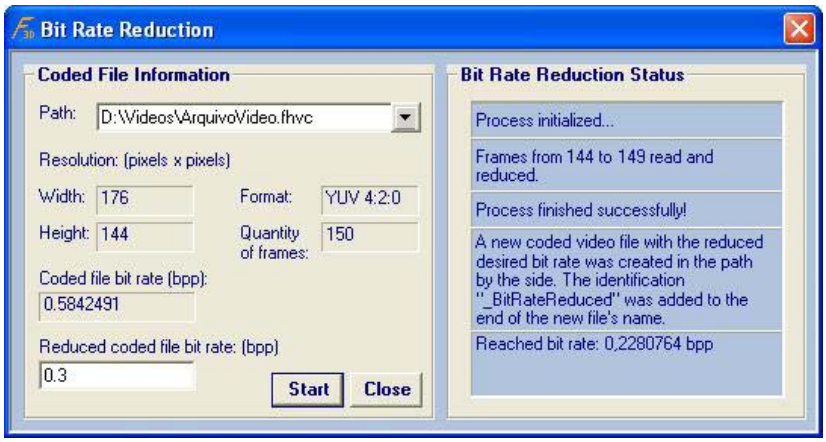


Fig. 3.4.7 - Tela de redução de taxa de bits do arquivo codificado.

3.4.3 Diagrama em camadas

Conforme mencionado na Seção 3.3.1, as classes do FHVC foram desenvolvidas de forma modularizada e funcional, sendo que cada uma realiza apenas uma parte do processo de codificação. Portanto, estas classes, bem como todo o sistema, podem ser divididas nas camadas apresentadas na Fig. 3.4.8. Observa-se que cada camada apresenta uma dependência unicamente com a camada inferior. As classes do FHVC pertencentes à cada camada foram nomeadas na Seção 3.3.1. De forma a manter a independência entre as classes de cada camada, cada classe foi modelada por um componente separado dentro de cada camada.

Fig. 3.4.8 - Diagrama em camadas do FHVC.

3.5 IMPLEMENTAÇÃO

A linguagem utilizada na implementação do FHVC foi o C#. Esta linguagem foi escolhida por ser orientada a objetos e eficiente para a aplicação de codificação de vídeo em questão, uma vez que gera códigos de execução rápida. Além disso, o código gerado pelo C# é multi-plataforma e só necessita de sua máquina virtual instalada no ambiente. Esta característica é necessária para o FHVC, uma vez que este foi desenvolvido inicialmente para ser executado no hardware de um *set-top box*, conforme mencionado no Capítulo 1.

O ambiente de desenvolvimento utilizado foi o "*Microsoft Visual C# .NET*", o qual foi escolhido por possibilitar implementações rápidas e fáceis de interfaces gráficas e por já ser um ambiente conhecido pela autora. Uma boa referência sobre programação em C# no ambiente .NET pode ser encontrada em [30].

O sistema operacional utilizado foi o "*Microsoft Windows XP Professional*" Versão 2002, que foi escolhido por já estar instalado nos computadores utilizados para a implementação e por ser compatível com o "*Rational Rose*" e com o "*Microsoft Visual C# .NET*". Entretanto, outros sistemas operacionais da *Microsoft* poderiam ser utilizados, tais como "*Windows NT/2000*".

3.6 TESTES

Não foram desenvolvidos testes unitários, de integração e de validação para o FHVC, conforme recomendados pelas técnicas de engenharia de *software* para o teste de sistemas especificadas em [28]. Esta abordagem não foi aplicada porque o desenvolvimento de uma metodologia completa de casos de teste demandaria muito tempo e deveria ser feita pela própria autora do código. Sendo assim, certamente não seriam testados todos os aspectos do código que seriam testados por outra pessoa que não conhecesse os detalhes de implementação do sistema e que, portanto, modelaria os testes de forma mais eficiente do ponto de vista do usuário.

Considerando as restrições acima, os testes do sistema FHVC foram feitos de forma funcional, focando os aspectos necessários para se fazer uma avaliação do desempenho das técnicas de codificação implementadas no sistema. Estes testes funcionais incluíram a utilização de vários tipos de seqüências de vídeo, de diversos formatos e tamanhos, que foram escolhidas para serem codificadas e decodificadas de forma seqüencial ou somente codificadas e depois

decodificadas. As seqüências codificadas também passaram pela redução de taxa de bits e as várias opções de formato interno de espaço de cores e tamanho do bloco de quadros codificado foram utilizadas.

Além disso, foram testados vários aspectos da interação do usuário com a interface gráfica, como por exemplo, o comportamento do sistema ao ser executado sem que todas as informações necessárias tivessem sido preenchidas na tela de forma válida. Em todos estes casos, verificou-se que o sistema responde de forma adequada, apresentando mensagens de erro solicitando a correção do dado que foi informado de forma inválida, ou que simplesmente não foi informado, e bloqueando a continuidade da execução do processo até que a correção necessária seja feita.

De forma geral, todos as funcionalidades implementadas no FHVC foram testadas para seqüências de vídeo válidas. Os resultados da codificação e decodificação de algumas destas seqüências são apresentados e avaliados no Capítulo 4 a seguir.

Capítulo 4
RESULTADOS OBTIDOS

Este capítulo apresenta os resultados experimentais obtidos pelo sistema computacional descrito no Capítulo 3, que é a implementação do método de codificação de vídeo proposto no Capítulo 2. Os resultados são comparados aos resultados obtidos com outros métodos de codificação descritos na literatura da área.

4.1 INTRODUÇÃO

O FHVC foi executado com as seqüências de vídeos mais utilizadas na área de codificação para publicação de resultados em artigos. Estas seqüências foram obtidas nas referências [33], [34] e [35], estão nos formatos CIF e QCIF com 30 fps (quadros por segundo) e no padrão de amostragem YUV 4:2:0.

Os resultados da codificação e decodificação destas seqüências foram avaliados de acordo com a metodologia descrita na Seção 4.2. Para a avaliação visual dos arquivos de vídeo reconstruídos através da comparação com os arquivos de vídeo originais foram utilizados os *players* de vídeo "*YUV Player Deluxe*" obtido em [31] e "*YUV Viewer*" disponível em [32].

4.2 METODOLOGIA

Conforme mencionado no início deste trabalho, dois requisitos importantes para o FHVC são a rapidez de execução e a utilização de pouca memória computacional, o que demandou cuidados na forma de implementação do codificador. Para a avaliação da eficiência temporal do algoritmo, foram registrados os tempos de codificação e de decodificação de cada seqüência de vídeo processada. Todos estes tempos foram obtidos em uma máquina com processador Pentium 4 3.20 GHz com 3Gb de memória executando exclusivamente o FHVC e são apresentados na Seção 4.3. A quantidade de memória computacional utilizada também foi monitorada durante a execução do FHVC através da aba "Desempenho" do Gerenciador de tarefas do "*Windows"*.

Verificou-se que para todas as seqüências de vídeo codificadas, o uso da CPU esteve em torno de 50%. A utilização de 100% da capacidade de processamento da CPU reduziria os

tempos necessários para codificação e decodificação, porém isto não pode ser obtido com o FHVC devido ao tempo gasto com as operações freqüentes de leitura do arquivo de vídeo original e de escrita no arquivo de vídeo codificado. Estas operações de leitura/escrita são custosas computacionalmente e limitam o desempenho do processador, uma vez que a execução de processos consiste de uma fase de execução da CPU e de uma fase de espera por dados de entrada/saída. Sendo assim, o uso da capacidade do processador seria otimizado se o FHVC fosse executado sem a necessidade de realizar operações de leitura/escrita em arquivos.

O uso da memória física também se manteve reduzido e em torno de 15%. Não há variação perceptível da quantidade de memória utilizada entre os processos de codificação e decodificação. Este comportamento já era esperado, uma vez que os processos de codificação e decodificação implementados no FHVC são simétricos.

Como forma de avaliação da qualidade do arquivo de vídeo reconstruído, optou-se pela medida de distorção mais utilizada na área, que é a PSNR (*Peak Signal to Noise Ratio*). Esta medida fornece a relação entre a máxima energia do sinal de vídeo e a energia do ruído. A maioria dos artigos e livros da área de codificação apresenta a seguinte fórmula para o cálculo da PSNR:

$$PSNR = 10\log_{10}\frac{255^2}{MSE}, \tag{2.22}$$

onde MSE (*Mean Squared Error*) é o erro quadrático médio entre a imagem original e a reconstruída calculado por:

$$MSE = \frac{1}{N_1 N_2}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}(x[n_1,n_2]-y[n_1,n_2])^2, \tag{2.23}$$

onde $x[n_1,n_2]$ é a imagem original de tamanho N_1 x N_2 e $y[n_1,n_2]$ é a imagem reconstruída de mesmo tamanho.

Entretanto, observa-se que a medida apresentada não pode ser aplicada diretamente ao FHVC porque foi feita para imagens em níveis de cinza (8 bits/pixel) e o FHVC usa arquivos de vídeo nos formatos YUV 4:2:0 ou RGB. Sendo assim, além de ser necessário considerar a forma de cálculo de uma única medida para os vários quadros do arquivo, deve-se considerar que o padrão de amostragem YUV 4:2:0 não é igual para as três componentes de cores e que apresenta 12 bpp.

A solução encontrada para o FHVC foi calcular separadamente o erro médio quadrático por quadro de cada uma das três componentes do espaço de cores. Este erro médio é utilizado para o cálculo da PSNR por quadro e por componente conforme a Eq. 2.22. Em seguida, calcula-se a média das medidas de PSNR de todos os quadros da seqüência e os três valores finais de PSNR são obtidos, um para cada componente da seqüência.

Esta solução baseia-se no fato da maioria dos artigos pesquisados na área apresentar valores de PSNR separados para as componentes Y, U e V calculados conforme a Eq. 2.22. Porém, outras fórmulas também são utilizadas na literatura, tal como a apresentada na Eq. 2.24 retirada de [25], onde as três componentes do espaço de cores são utilizadas:

$$PSNR = 10\log_{10}\frac{255^2}{(MSE(Y)+MSE(U)+MSE(V))/3}. \tag{2.24}$$

Esta variedade de formas de cálculos da PSNR para arquivos de vídeo no formato YUV 4:2:0 dificulta a avaliação de resultados obtidos pelos algoritmos que codificam este tipo de seqüência, uma vez que nem sempre se sabe exatamente como a PSNR foi calculada.

De modo geral, boas reconstruções de vídeo têm valores de PSNR em torno de 30 dB. Porém, muitas vezes a medida da PSNR não reflete a qualidade visual das seqüências de vídeo reconstruídas, uma vez que não captura o efeito de bloco e a perda de detalhes devido à quantização [18]. Além disso, uma medida baixa de PSNR não implica, necessariamente, em qualidade visual ruim. Estas afirmações são comprovadas pelos resultados apresentados na Seção 4.3, onde algumas seqüências de vídeo apresentam PSNR baixa, mas têm boa qualidade visual.

Além da PSNR, outra medida importante para a avaliação de codificadores de vídeo é a taxa de bits por pixel alcançada para os arquivos codificados. Para o arquivo de vídeo original, tem-se a taxa de quadros por segundo, a taxa de bits por quadro (em bits/pixel) e o tamanho do quadro (em pixels). Enquanto todas estas medidas são fixas no arquivo de vídeo original, para o arquivo de vídeo codificado, a taxa de bits/pixel varia, uma vez que o codificador FHVC é um codificador de tamanho de palavra de código variável.

Sabe-se que existe uma estreita relação entre a PSNR e a taxa de bits por pixel da seqüência de vídeo. De forma geral, quanto maior a taxa de bits, maior a PSNR. Gráficos relacionando estas medidas são extensamente apresentados nos artigos da área e também são apresentados na Seção 4.3 para as seqüências de vídeo utilizadas.

Outras formas de avaliação dos codificadores de vídeo analisam, por exemplo, a complexidade computacional dos algoritmos implementados, tais como as apresentadas em [26] e [27]. As medidas calculadas nestes artigos estabelecem pesos para cada tipo de operação aritmética realizada no algoritmo e levam em consideração as quantidades de memória física e dinâmica das máquinas em que os algoritmos estão sendo executados. Este tipo de medida de complexidade computacional não foi calculado para o FHVC devido ao grande tamanho do código e, principalmente, ao fato de não se ter encontrado artigos com as mesmas medidas calculadas para outros algoritmos de codificação. Desta forma, a medida da complexidade computacional do FHVC seria calculada, mas não se poderia afirmar se representaria um algoritmo de alta ou de baixa complexidade, devido à falta de meios de comparação.

4.3 RESULTADOS

Vários aspectos e configurações do FHVC foram testados e a avaliação dos resultados obtidos é apresentada a seguir. Inicialmente, na Seção 4.3.1, são analisadas seqüências de vídeos com características variadas. Para todas as seqüências são mostrados os seguintes resultados:

- Curvas de taxa de bits por pixel x PSNR (calculada conforme a Seção 4.2);
- Tempos de codificação e de decodificação;
- Alguns quadros da seqüência para avaliação de qualidade visual.

Estes resultados são comparados com os apresentados em [37], [38], [39], [40] e [41]. Todos estes artigos descrevem algoritmos que utilizam técnicas tridimensionais de codificação de vídeo, assim como o FHVC.

Para efeito de comparação de resultados, também foi utilizada uma implementação pronta do MPEG 4 – versão 10 (H.264). Apesar das técnicas utilizadas neste formato serem muito diferentes das utilizadas no FHVC, considerou-se interessante a comparação com o padrão de codificação de vídeo que apresenta o melhor desempenho atualmente. A implementação utilizada corresponde ao *software* de referência oficial H.264/AVC (*Advanced Video Codec*).

É importante ressaltar que existem implementações otimizadas do padrão H.264 e, portanto, de execução mais rápida do que o *software* de referência oficial. Mesmo assim, optou-se pela comparação do FHVC com o *software* de referência por ser este não-proprietário e

disponibilizado sempre íntegro e sem restrições em [42] pelo HII (*Heinrich-Hertz-Institut*), que é o instituto responsável pela manutenção do *software* de referência. Além disso, a implementação do FHVC também não é otimizada para o hardware em que está sendo executada, uma vez que o código em C# é interpretado e não foi gerada uma versão compilada do FHVC.

Enfatiza-se ainda que, a comparação de desempenho entre os dois codificadores foi realizada apenas para se obter um parâmetro de avaliação para o FHVC que pudesse ser reproduzido para a avaliação de qualquer outro codificador. Ou seja, os tempos de codificação/decodificação de qualquer outro codificador podem ser comparados aos tempos obtidos com o *software* de referência oficial (uma vez que este é disponibilizado livremente) e assim indiretamente comparados aos tempos obtidos com o FHVC.

A maioria dos parâmetros utilizados no arquivo de configuração da implementação do *software* de referência oficial são os identificados como *default* no manual oficial do *software* [43] desenvolvido pelo JVT (*Joint Video Team*). A utilização da maioria dos parâmetros configurados com seus valores *default* impede que o H.264 alcance o seu melhor desempenho, pois muitos de seus recursos não são utilizados. Por outro lado, esta escolha reduz a complexidade dos algoritmos e aumenta a velocidade de execução do codificador. Obtém-se, assim, um bom ajuste entre desempenho e velocidade de execução. Os parâmetros que não foram configurados como *default* no arquivo de configuração do *software* estão especificados com os seguintes valores:

- Perfil: *Main*;
- Nível: 2.0;
- GOP: 15;
- Tipo da seqüência: I-B-B-P-B-B-P-B-B-P-B-B-P-B-B-I;
- Transformada de Hadamard: utilizada para todas as posições de sub-pixel;
- Número de quadros de referência: 5;
- Codificador de entropia: CABAC (*Context-based Adaptive Binary Arithmetic Coding*);
- Otimização de taxa-distorção: utilizada no modo de complexidade alta;
- Esquema de estimação de movimento: busca completa com faixa de busca de 16 pixels sem restrições;
- Arredondamento adaptativo: habilitado com base em JVT-N011.

Outro aspecto do FHVC, analisado na Seção 4.3.2, é a possível utilização de um espaço de cores interno de codificação diferente do espaço de cores do arquivo de vídeo original. Esta análise é feita para a seqüência de vídeo "Miss America", uma vez que esta foi a seqüência utilizada no Capítulo 2 para a descrição do sistema proposto neste trabalho. Ainda para a seqüência "Miss America", analisa-se o desempenho do codificador com a utilização de 4 e de 8 quadros por cubo. As implicações do uso de cada um destes tamanhos de cubo na codificação são apresentadas na Seção 4.3.3.

4.3.1 Análise do desempenho do FHVC para seqüências de vídeo com diferentes características

Esta seção apresenta resultados de codificação/decodificação para as seqüências de vídeo QCIF "Miss America", "Foreman" e "Hall Monitor" no formato YUV 4:2:0. Estas seqüências foram escolhidas porque cobrem uma variedade satisfatória de movimentos de câmera e objetos. O desempenho do FHVC com seqüências de vídeo de maior resolução também é analisado com a utilização da seqüência CIF "Mobile" no formato YUV 4:2:0.

Os resultados apresentados nesta seção foram obtidos com a execução do FHVC com o mesmo espaço de cores interno do arquivo de vídeo original, ou seja YUV 4:2:0. Para as seqüências QCIF foram utilizados 8 quadros por cubo e para a seqüência CIF foram utilizados 4 quadros por cubo. Esta escolha foi feita devido ao conteúdo das seqüências e não em função das resoluções. Coincidentemente, as seqüências QCIF apresentadas nesta seção apresentam movimento reduzido ou moderado e regiões de fundo fixas. Portanto, foram melhor codificadas com a utilização de 8 quadros por cubo. Já a seqüência CIF utilizada apresenta bastante movimento e imagens com muitos detalhes e cores, sendo, portanto, melhor codificada com a utilização de 4 quadros por cubo. Ressalta-se, entretanto, que seqüências QCIF também podem ser codificadas com 4 quadros por cubo. Assim como, seqüências CIF podem ser codificadas com 8 quadros por cubo.

As diferenças de resultados obtidos com o FHVC utilizando 4 e 8 quadros por cubo para a codificação da seqüência "Miss America" são apresentadas na Seção 4.3.3.

Uma vez que os componentes de luminância e crominância contêm informações distintas, os seus resultados são apresentados nesta seção em gráficos separados. Porém, o arquivo foi

codificado com as três componentes em conjunto, ou seja, não foi feita uma divisão do arquivo original em três outros arquivos (um com cada componente) antes da codificação.

4.3.1.1 Seqüência de vídeo "Miss America"

A seqüência "Miss America" é uma seqüência de 150 quadros com movimentos lentos concentrados na cabeça e nos lábios de uma pessoa que aparece sempre localizada centralmente no vídeo. Uma vez que os movimentos são lentos e não ocorrem mudanças de cena na seqüência, a redundância temporal é alta. Nestes casos, os métodos tridimensionais de codificação geralmente apresentam ótimos resultados, ultrapassando o desempenho dos métodos bidimensionais, conforme ilustrado no gráfico da Fig. 4.3.1 e nas imagens da Fig. 4.3.3 que foram obtidas de [37]. O gráfico e as imagens destas figuras correspondem à codificação da seqüência "Miss America" QCIF em escala de cinzas (8 bpp). Conforme consta em [37], os resultados dos métodos bidimensionais comparados na Fig. 4.3.1 provêm de implementações simples da DCT (2D DCT) e da compensação de movimentos com busca exaustiva no quadro anterior (2D MC- *Motion Compensation*). Já os métodos tridimensionais utilizam a DCT nas três dimensões do vídeo (3D DCT), com a implementação da compensação de movimentos em cubos de quadros (3D MC) e com a utilização da 3D DCT - PCA *(Principal Component Analysis*), que identifica os componentes principais de cada cubo de quadros e codifica o resíduo entre a predição e o dado de vídeo real através da 3D DCT.

A Fig. 4.3.2 apresenta os resultados obtidos com o FHVC e o H.264 para a codificação da seqüência "Miss America" QCIF no formato YUV 4:2:0 (12 bpp). Verifica-se nesta figura que o H.264 sempre apresenta resultados superiores aos do FHVC em termos de taxa de bits por pixel x PSNR e que esta diferença diminui para as componentes de crominância, especialmente para a componente U. Já em termos de qualidade visual, verifica-se que o desempenho do FHVC pode ser comparado ao do H.264 até para uma taxa baixa de 0,1 bit/pixel, conforme mostrado nos quadros da Fig. 4.3.4. Isto acontece porque mesmo à taxa de 0,1 bit/pixel, o FHVC obtém uma seqüência de vídeo reconstruída de boa qualidade com 36 dB para a componente de luminância Y.

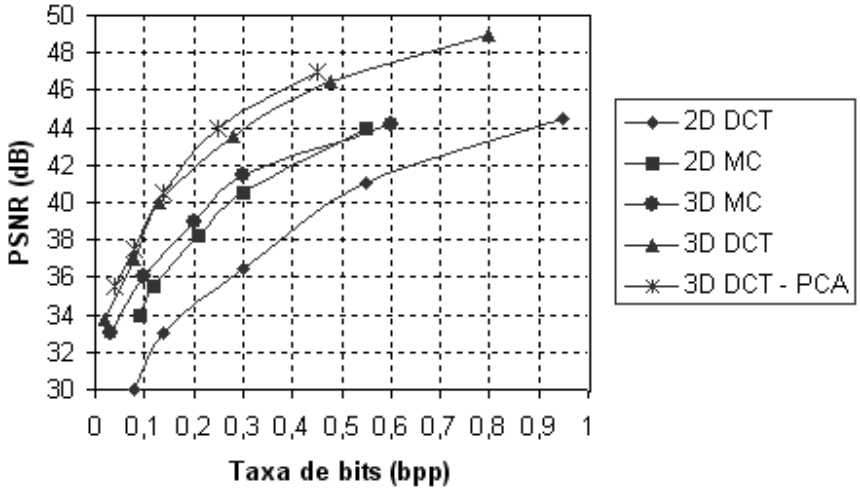


Fig. 4.3.1 - Comparação de resultados entre métodos bidimensionais e tridimensionais para a seqüência "Miss America" [37].

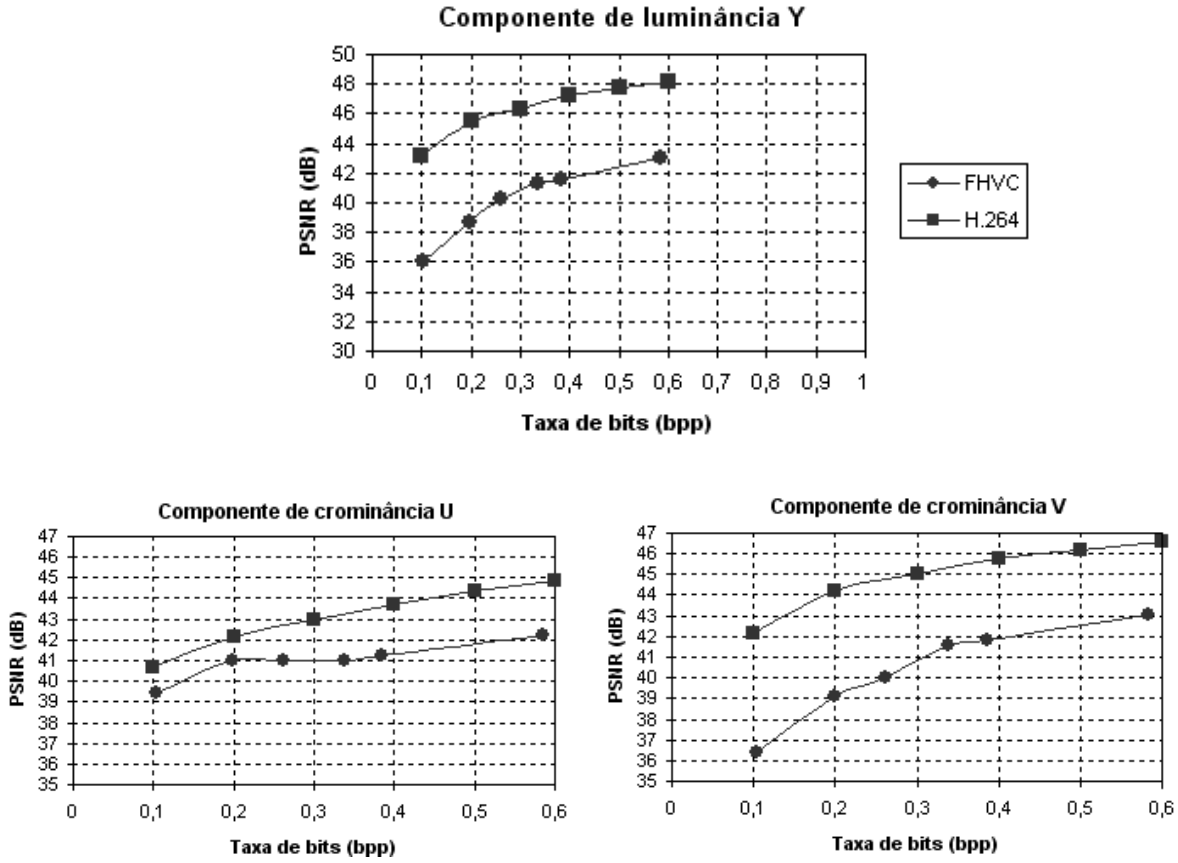


Fig. 4.3.2 - Resultados experimentais para a seqüência "Miss America".

Já a comparação entre o FHVC e os demais métodos de codificação tridimensional de vídeo apresentados na Fig. 4.3.1 não é adequada porque os resultados dos métodos tridimensionais mostrados referem-se à seqüência de vídeo “Miss America” com 8 bpp (só luminância), enquanto os resultados do FHVC mostrados na Fig. 4.3.2 referem-se à seqüência com 12 bpp (YUV 4:2:0), ou seja, com mais informação a ser comprimida. De fato, percebe-se que a qualidade visual do quadro codificado pelo FHVC com 0,1 bit/pixel apresentado na Fig. 4.3.4(a) é superior a de todos os quadros apresentados na Fig. 4.3.3 que foram codificados com outras técnicas bidimensionais e tridimensionais a 0,14 bit/pixel.

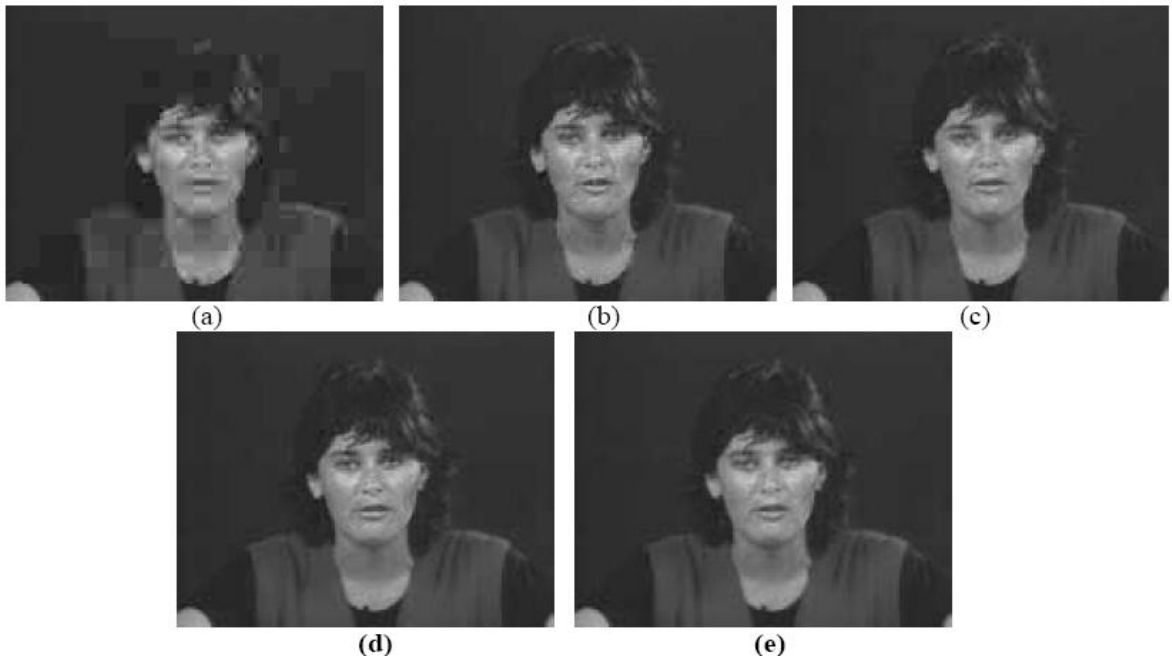


Fig. 4.3.3 - Quadro reconstruído da seqüência "Miss America" codificada a 0,14 bits/pixel utilizando (a) 2D DCT, (b) 2D MC, (c) 3D MC, (d) 3D DCT e (e) 3D PCA [37].

Fig. 4.3.4 - Componente de luminância Y de quadro reconstruído da seqüência "Miss America" codificada a 0,1 bit/pixel utilizando (a) FHVC e (b) H.264.

Os tempos de codificação e de decodificação obtidos com o FHVC foram registrados conforme especificado na Seção 4.2 e comparados aos tempos obtidos com o H.264 para as mesmas taxas de bits por pixel. Estes resultados são apresentados na Tabela 4.3.1. Verifica-se que os tempos de codificação e decodificação do FHVC e do H.264 diminuem sempre com a redução da taxa de bits por pixel utilizada. O FHVC apresenta os mesmos tempos de codificação e de decodificação, uma vez que é um codificador simétrico. Já o H.264 possui um tempo de codificação aproximadamente 22 vezes superior ao tempo de decodificação. Na comparação entre os codificadores, observa-se que o FHVC é sempre mais rápido do que o H.264, sendo aproximadamente 160 vezes mais rápido na codificação e 8 vezes mais rápido na decodificação.

CODIFICAÇÃO		
Taxa (bpp)	Tempo por quadro (milisegundos)	
	FHVC	H.264
0,60	19,68	3.111,65
0,40	17,47	3.062,68
0,30	16,81	3.040,45
0,20	15,96	2.990,62
0,10	15,07	2.909,48

DECODIFICAÇÃO		
Taxa (bpp)	Tempo por quadro (milisegundos)	
	FHVC	H.264
0,60	16,85	135,35
0,40	16,12	135,05
0,30	15,70	136,38
0,20	16,22	131,68
0,10	14,86	128,54

Tabela 4.3.1 - Resultados de tempos de codificação e decodificação para a seqüência "Miss America".

Uma vez que o desempenho do H.264 em relação ao FHVC é sempre inferior em termos de tempo de execução, mas é sempre superior em termos de taxa de bits por pixel x PSNR, procurou-se analisar a que taxa o FHVC obtém uma seqüência de qualidade visual semelhante à obtida pelo H.264 para então se comparar os tempos de execução.

A análise utilizou os três quadros originais mostrados na Fig. 4.3.5(a). Estes quadros forem escolhidos de forma a representar regiões com diferentes tipos e graus de movimento. O quadro #16 está no início da seqüência e corresponde a um período de pouquíssimo movimento concentrado na região dos lábios. O quadro #83 está centrado na seqüência e no período de maior movimento da mesma. Este período corresponde a um movimento lateral da cabeça para direita que começa a partir do quadro #60 e se estende até o quadro #110. Já o quadro #143 está no final da seqüência localizado em um período de movimento apenas dos olhos e dos lábios. O resultado

da codificação e da decodificação destes quadros à taxa de 0,1 bit/pixel é apresentado na Fig. 4.3.5(b) para o FHVC e na Fig. 4.3.5(c) para o H.264.

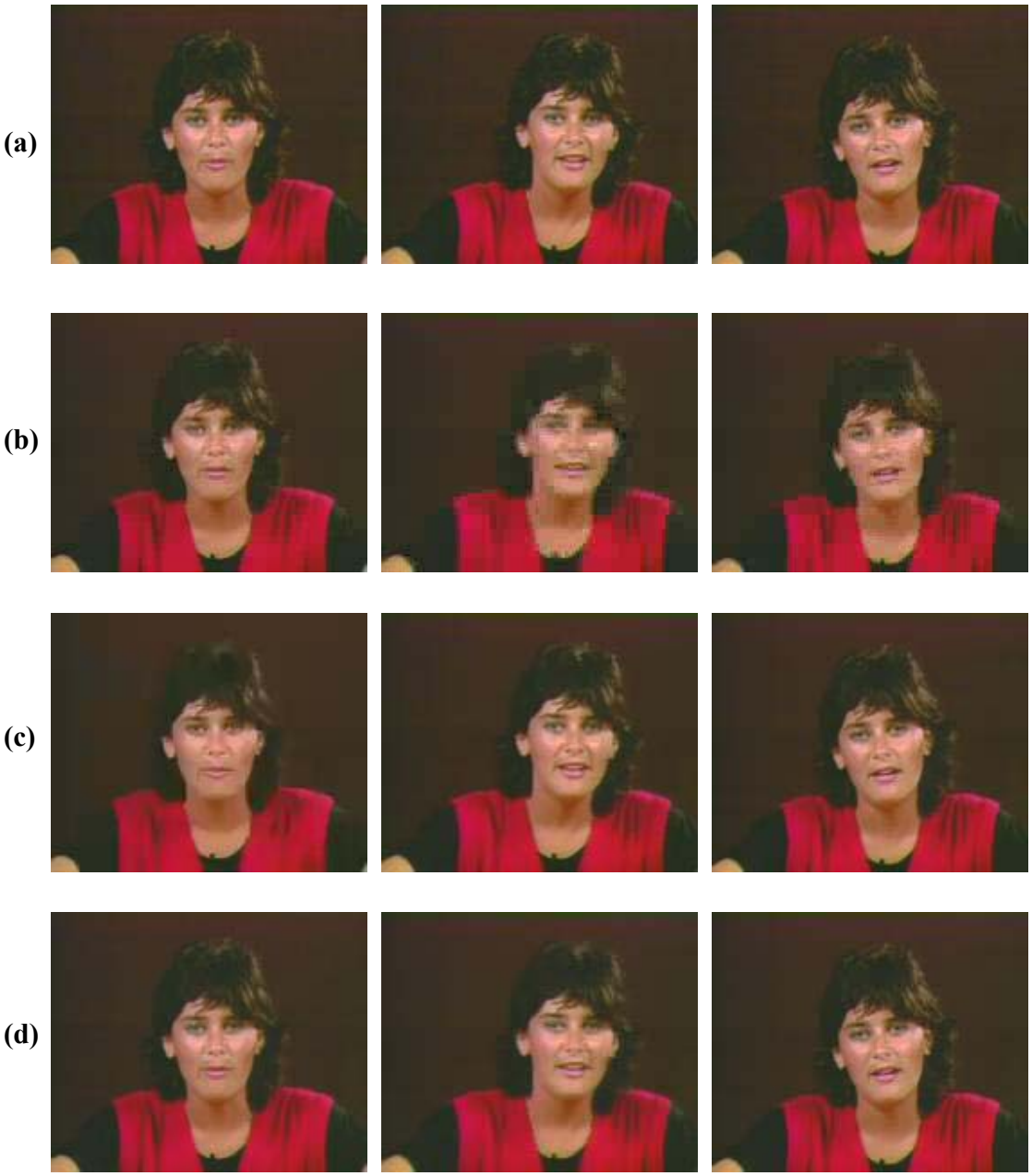


Fig. 4.3.5 – Quadros #16, #83 e #143 originais em (a) e quadros reconstruídos com todas as componentes da seqüência "Miss America" codificada a 0,1 bit/pixel utilizando FHVC em (b) e H.264 em (c) e a 0,38 bit/pixel utilizando FHVC em (d) para comparação da qualidade visual com as imagens obtidas em (c).

Conforme esperado, o FHVC apresenta um bom resultado para o quadro #16, pois este explora a grande redundância temporal existente no período inicial da seqüência. Porém, não consegue manter esta mesma qualidade para os demais quadros, especialmente para o quadro #83, proveniente da região de maior movimento. Já o H.264 apresenta resultados excelentes, o que também já era esperado, uma vez que obtém uma PSNR superior a 40 dB para todas as componentes (conforme mostra a Fig. 4.3.2). A Fig. 4.3.2 também mostra que à taxa de aproximadamente 0,4 bit/pixel, o desempenho do FHVC já pode ser equiparado ao do H.264 à taxa de 0,1 bit por pixel. De fato, a qualidade visual dos quadros apresentados na Fig. 4.3.5(d), codificados à taxa de 0,38 bit/pixel pelo FHVC, pode ser comparada à dos quadros da Fig. 4.3.5(c).

De acordo com a Tabela 4.3.1, o H.264 necessita de 2.909,48 ms por quadro para a codificação à taxa de 0,1 bit/pixel. Já o FHVC necessita de 17,47 ms para a codificação de um quadro de qualidade visual semelhante à taxa de 0,4 bit/pixel. Conclui-se que, ao custo de uma redução de 4 vezes na taxa de compressão obtida pelo H.264, obtém-se uma codificação 166 vezes mais rápida com o FHVC. Outra importante conclusão é que o tempo de 17,47 ms de codificação por quadro torna possível a codificação em tempo real de uma seqüência de vídeo original à taxa de 30 quadros/segundo.

4.3.1.2 Seqüência de vídeo "Foreman"

A seqüência “Foreman” é uma seqüência com movimento não-uniforme causado pela câmera e pelos movimentos da face e da cabeça de uma pessoa. A seqüência contém 300 quadros e apresenta uma mudança completa de cena a partir do 190^{o} quadro. Mesmo com uma boa quantidade de movimento, as superfícies uniformes dos quadros ainda proporcionam uma grande quantidade de redundância temporal e espacial que é adequadamente explorada pelo método de codificação tridimensional. Sendo assim, o FHVC alcança um bom desempenho na codificação desta seqüência, embora inferior à obtida para a seqüência “Miss America” analisada na seção anterior. Isto ocorre porque a seqüência “Miss America” é mais uniforme e apresenta menos movimento do que a seqüência “Foreman”.

Os artigos referenciados em [38] e [39] descrevem métodos de codificação embutida (progressiva) de vídeo, por isso foram utilizados para comparação de resultados com o FHVC. Além disso, o método apresentado em [39] também utiliza técnicas tridimensionais.

O método de codificação de vídeo descrito em [38] se baseia em wavelets e obtém taxa de bits "escalável", sendo chamado de SAMCoW (*Scalable Adaptive Motion Compensated Wavelet*). O termo "escalável" se refere à um processo de decodificação ou transmissão em que apenas uma parte da seqüência embutida pode ser utilizada de acordo, por exemplo, com uma taxa de bits específica, produzindo seqüências de vídeo com diferentes níveis de qualidade visual, resolução espacial ou taxa de quadros. O método SAMCoW utiliza compensação de movimento para redução da redundância temporal e uma abordagem similar ao método EZW (*Embedded Zerotree Wavelet*) para codificação dos quadros de erro de predição e dos quadros intra-codificados. Esta abordagem "EZW" também é utilizada para explorar a interdependência entre as componentes de luminância e crominância.

Os resultados obtidos pelo SAMCoW na codificação da seqüência "Foreman" foram extraídos de [38] e comparados com os resultados obtidos com o FHVC e com o H.264 em termos de taxa de bits por pixel x PSNR, na Fig. 4.3.6, e em termos de qualidade visual, na Fig. 4.3.7. Observa-se na Fig. 4.3.6, que o FHVC alcança um resultado muito próximo ao SAMCoW na codificação da componente de luminância e que ultrapassa o SAMCoW na codificação das componentes de crominância. De fato, o bom resultado obtido na codificação das componentes U e V pelo FHVC se mostra comparável até mesmo ao resultado obtido pelo H.264. Já a comparação dos resultados da codificação da componente Y entre o FHVC e o H.264, demonstra que este último sempre atinge resultados 10dB superiores, em média.

A qualidade visual dos quadros apresentados na Fig. 4.3.7 está de acordo com os resultados apresentados nos gráficos de taxa de bits por pixel x PSNR da Fig. 4.3.6. Observa-se, assim, que os métodos SAMCoW e FHVC apresentam boa qualidade visual para taxas maiores de codificação (a partir de 1 bit/pixel). Os quadros codificados com esta taxa apresentam 34 dB de PSNR, em média, para a componente de luminância e são mostrados na Fig. 4.3.7(a). Verifica-se, também, que nesta taxa, a qualidade visual obtida pelo FHVC pode ser comparada com a qualidade obtida pelo H.264. Já os quadros codificados com taxas menores, a 0,5 bit/pixel apresentados na Fig. 4.3.7(b) e a 0,25 bit/pixel apresentados na Fig. 4.3.7(c), não apresentam boa qualidade visual para os métodos SAMCoW e FHVC. Os quadros codificados pelo SAMCoW apresentam borramento e o efeito de blocagem aparece nos quadros codificados pelo FHVC. Já o H.264 sempre apresenta quadros com boa qualidade visual na Fig. 4.3.7. Isto ocorre porque, até

mesmo o quadro codificado na taxa mais baixa mostrada na figura, que é de 0,25 bit/pixel, apresenta alta PSNR para a luminância (de 38dB).

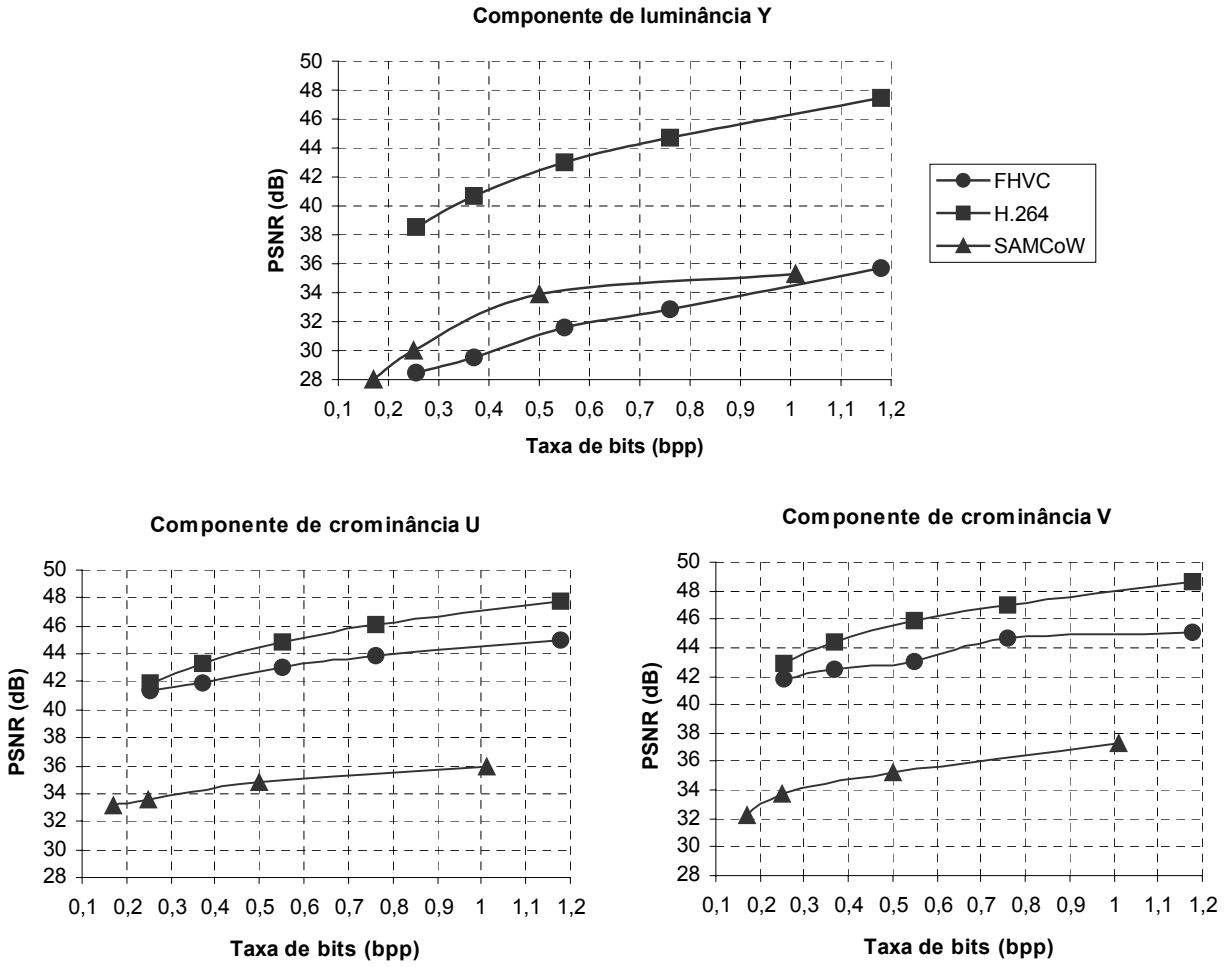


Fig. 4.3.6 - Resultados experimentais para a seqüência "Foreman" (resultados para o SAMCoW de [38]).

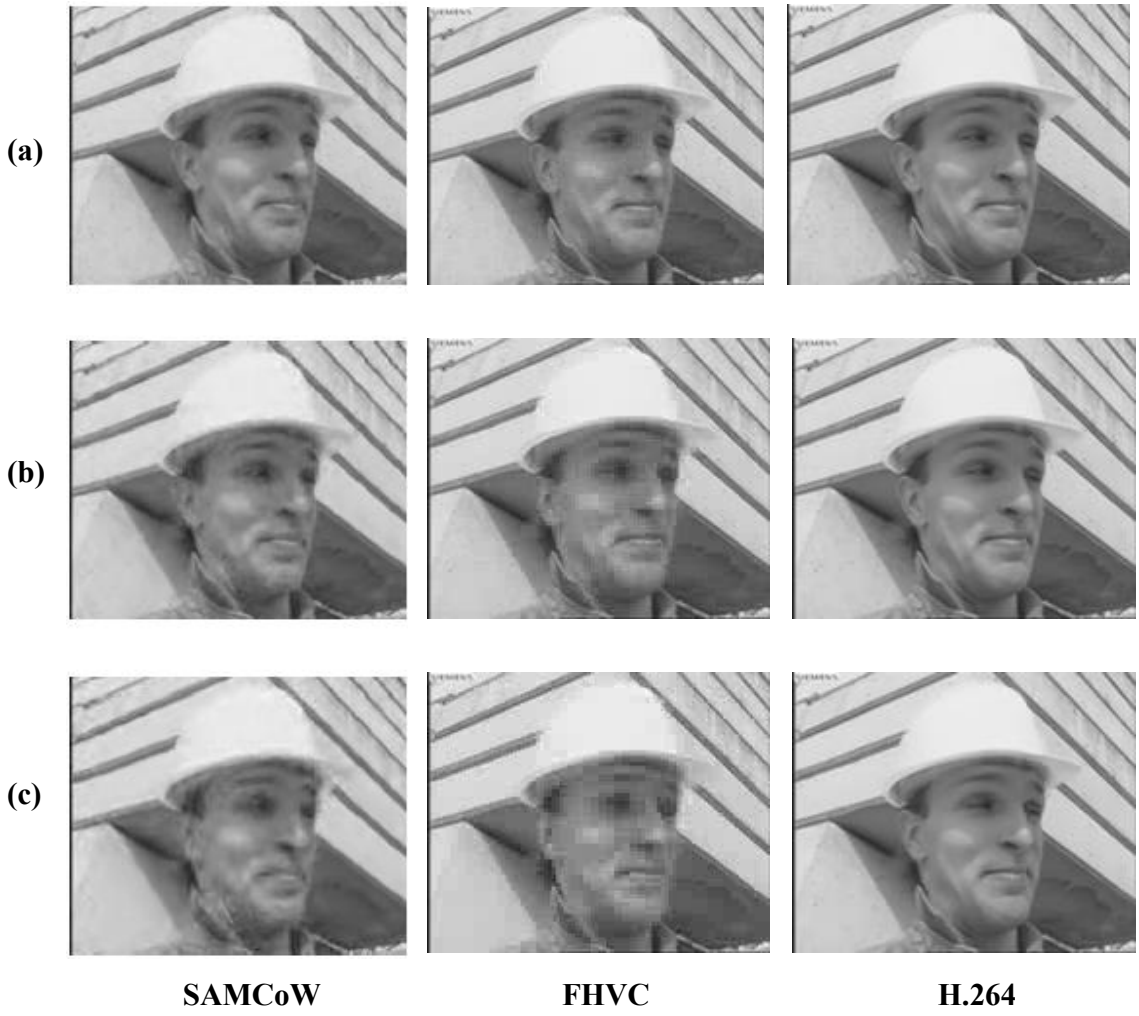


Fig. 4.3.7 - Componente de luminância Y de quadro reconstruído da seqüência "Foreman" codificada a aproximadamente (a) 1 bit/pixel, (b) 0,5 bit/pixel e (c) 0,25 bit/pixel com SAMCoW[38], FHVC e H.264.

A Tabela 4.3.2 apresenta os tempos de codificação e de decodificação obtidos com o FHVC e com o H.264 para as mesmas taxas de bits por pixel. Verifica-se que as características de redução de tempo x redução da taxa de bits comentadas para a seqüência “Miss America” na seção anterior também são válidas para a seqüência “Foreman”. Além disso, o tempo de codificação do H.264 continua aproximadamente 24 vezes superior ao tempo de decodificação.

A maior diferença observada com relação aos tempos de codificação obtidos pelo H.264 está na análise que será feita a partir dos resultados da Tabela 4.3.3. Todos os resultados apresentados nesta tabela foram obtidos com a utilização da taxa de codificação de 0,37 bit/pixel porque a partir desta taxa, o FHVC apresenta PSNR superior a 30dB para todas as componentes.

CODIFICAÇÃO		
Taxa (bpp)	Tempo por quadro (milisegundos)	
	FHVC	H.264
1,18	22,29	3.845,98
0,76	19,89	3.588,71
0,55	18,98	3.491,14
0,37	17,94	3.341,81
0,25	14,03	3.206,08

DECODIFICAÇÃO		
Taxa (bpp)	Tempo por quadro (milisegundos)	
	FHVC	H.264
1,18	19,11	149,68
0,76	17,24	145,11
0,55	16,20	142,96
0,37	15,62	137,29
0,25	15,16	136,25

Tabela 4.3.2 - Resultados de tempos de codificação e decodificação para a seqüência "Foreman".

	Tempo de codificação por quadro (milisegundos)		PSNR (dB) da componente de luminância Y	
	FHVC	H.264	FHVC	H.264
Seqüência "Miss America"	16,15	3.000,00	41,5	47
Seqüência "Foreman"	17,94	3.341,81	29,5	40,5
Comparação	Aumento de 1,8%	Aumento de 10,2%	Queda de 12 dB	Queda de 6,5 dB

Tabela 4.3.3 – Comparação entre resultados de tempos de codificação e PSNR obtidos pelo FHVC e pelo H.264 para as seqüências "Miss America" e "Foreman" à taxa de 0,37 bit/pixel.

Observa-se que, tanto o FHVC quanto o H.264, não conseguem obter valores de PSNR tão bons para a seqüência "Foreman" quanto os obtidos para a seqüência "Miss America". A queda do valor de PSNR apresentada pelo H.264 é de 6,5 dB, enquanto a queda apresentada pelo FHVC é de 12 dB. Entretanto, verifica-se que, esta queda menor do H.264 acontece ao custo de um aumento de 10,2% do seu tempo de codificação, enquanto o tempo de codificação do FHVC aumenta somente em 1,8%.

Assim, observa-se que, como a seqüência "Foreman" apresenta muito mais movimento e informação do que a seqüência "Miss America", tanto o FHVC quanto o H.264, precisaram aumentar o tempo de codificação e não obtiveram os mesmos valores de PSNR. Verifica-se também, que a diferença de PSNR entre o FHVC e o H.264, que era de 5,5 dB para a seqüência "Miss America", subiu para 11 dB para a seqüência "Foreman" devido ao aumento de tempo de codificação realizado pelo H.264 (10,2%), que não foi proporcional ao aumento realizado pelo FHVC (1,8%).

A análise dos dados da Tabela 4.3.2 mostra ainda que os tempos de decodificação obtidos pelo FHVC e pelo H.264 também aumentaram em relação aos tempos obtidos para a seqüência

"Miss America". Entretanto, o aumento foi proporcional para os dois decodificadores e, portanto, a decodificação realizada pela FHVC continua 8 vezes, em média, mais rápida que o H.264.

A comparação da qualidade visual obtida para um quadro reconstruído com todas as suas componentes utilizando o FHVC e o H.264 pode ser feita através da Fig. 4.3.8.

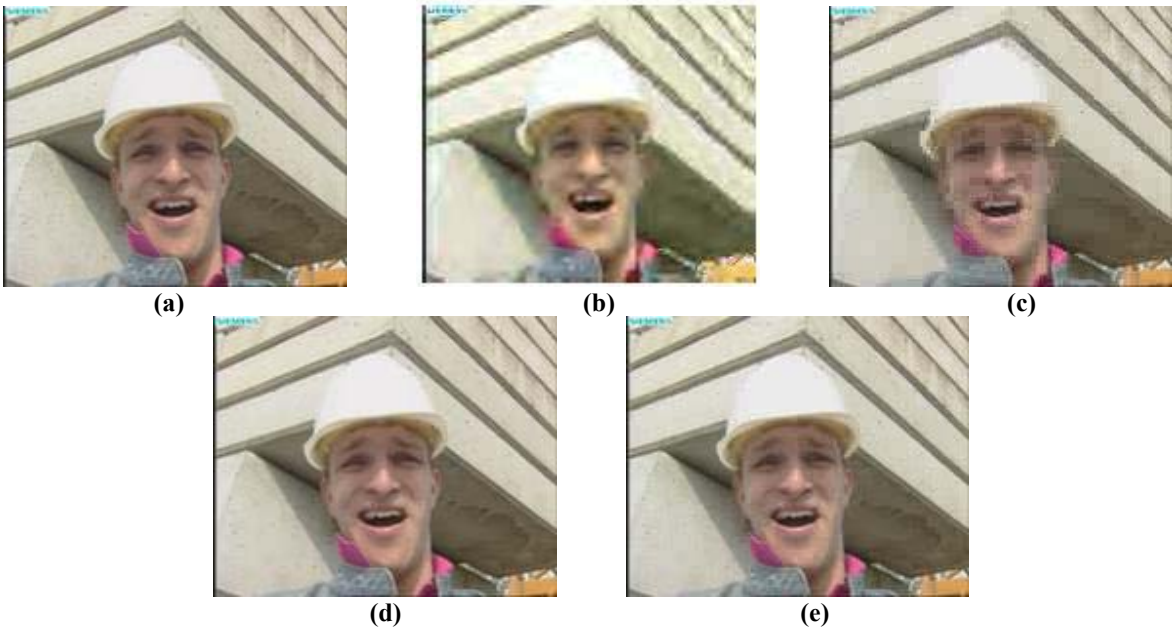


Fig. 4.3.8 - Quadro #30 original em (a) e quadro reconstruído com todas as componentes da seqüência "Foreman" codificada a 0,37 bit/pixel utilizando o codificador escalável 3D [39] em (b), o FHVC em (c) e o H.264 em (d) e a 1,18 bit/pixel utilizando o FHVC em (e) para comparação com a imagem obtida em (d).

A Fig. 4.3.8(b) apresenta um quadro extraído de [39]. O artigo [39] descreve um método de codificação de vídeo tridimensional baseado em decomposição por sub-bandas. Neste método, grupos de quadros são temporalmente filtrados usando compensação de movimentos e, em seguida, espacialmente decompostos com wavelets. Os coeficientes resultantes são, então, comprimidos com uma estratégia baseada em SPIHT denominada de "*Fully Scalable Zerotree"* em conjunto com um codificador de entropia aritmético. Desta forma, o método obtém escalabilidade espacial, temporal e de PSNR.

O quadro codificado com o FHVC está apresentado na Fig. 4.3.8(c). Na taxa de 0,37 bit/pixel, utilizada para a codificação dos quadros da Fig. 4.3.8, o FHVC obtém apenas 29 dB para a componente de luminância Y. Este valor baixo de PSNR confirma a baixa qualidade visual da imagem, que apresenta grande efeito de blocagem. Já a imagem da Fig. 4.3.8(b) apresenta

PSNR ainda inferior, de 27,44 dB, conforme consta em [39]. As cores desta imagem também estão modificadas com relação às da imagem original.

O quadro codificado pelo FHVC na Fig. 4.3.8(e) à taxa de 1,18 bit/pixel apresenta qualidade visual comparável ao quadro obtido pelo H.264 na Fig. 4.3.8(d) à taxa de 0,37 bit/pixel. Para a codificação do quadro da Fig. 4.3.8(d), o H.264 gasta 6.717,04 ms e para a codificação do quadro da Fig. 4.3.8(e), o FHVC gasta apenas 44,80 ms. Conclui-se que, ao custo de uma redução de 3 vezes na taxa de compressão obtida pelo H.264, obtém-se uma codificação 150 vezes mais rápida com o FHVC.

4.3.1.3 Seqüência de vídeo "Hall Monitor"

A seqüência “Hall Monitor” é uma seqüência típica de monitoração de ambientes com 320 quadros. O fundo da seqüência é sempre fixo e alguns objetos (pessoas) aparecem e desaparecem. Esta característica não existia nas seqüências “Miss America” e “Foreman” analisadas anteriormente e adiciona certa quantidade de movimento não-contínuo à seqüência. Mesmo com este tipo de movimento, como a maior parte da informação dos quadros da seqüência correspondente à região de fundo, que é fixa, o FHVC obtém bons resultados na codificação. Estes resultados são inferiores aos obtidos para a seqüência “Miss America”, mas são superiores aos obtidos para a seqüência “Foreman”.

A Fig. 4.3.9 apresenta gráficos de taxa de bits por pixel x PSNR obtidos com o FHVC, o H.264 e com os métodos H.263, MC 3D SPIHT e 3D SPIHT, cujos resultados obtidos para a seqüência “Hall Monitor” foram transcritos de [40]. O artigo [40] descreve o método de codificação tridimensional 3D SPIHT, que emprega o algoritmo SPIHT nas três dimensões do arquivo de vídeo. Obtém-se uma seqüência embutida baseada em sub-bandas com qualidade equivalente à produzida pelo padrão H.263 e que é ainda melhorada quando se adiciona compensação de movimento, gerando o método MC 3D SPIHT. A comparação com os resultados obtidos pelo FHVC para a componente de luminância Y demonstra que estes são apenas 4 dB, em média, inferiores aos resultados apresentados em [40].

Já a comparação entre os resultados obtidos para a codificação da componente de luminância Y pelo FHVC e pelo H.264 mostra que, a taxas superiores a 0,5 bit/pixel, o desempenho do FHVC se aproxima bastante do desempenho do H.264 e se mantém apenas 4 dB

abaixo deste. De fato, entre todas as seqüências de vídeo analisadas, verifica-se que, embora a seqüência “Miss America” tenha obtido valores de PSNR superiores tanto para o H.264 quanto para o FHVC, a seqüência “Hall Monitor” é a que apresenta os resultados de codificação do FHVC mais próximos dos resultados obtidos com o H.264. Esta afirmação é reforçada quando os gráficos de taxa de bits por pixel x PSNR são analisados para as componentes de crominância U e V, pois verifica-se nestes gráficos que os desempenhos obtidos com o H.264 e com o FHVC podem ser considerados equivalentes.

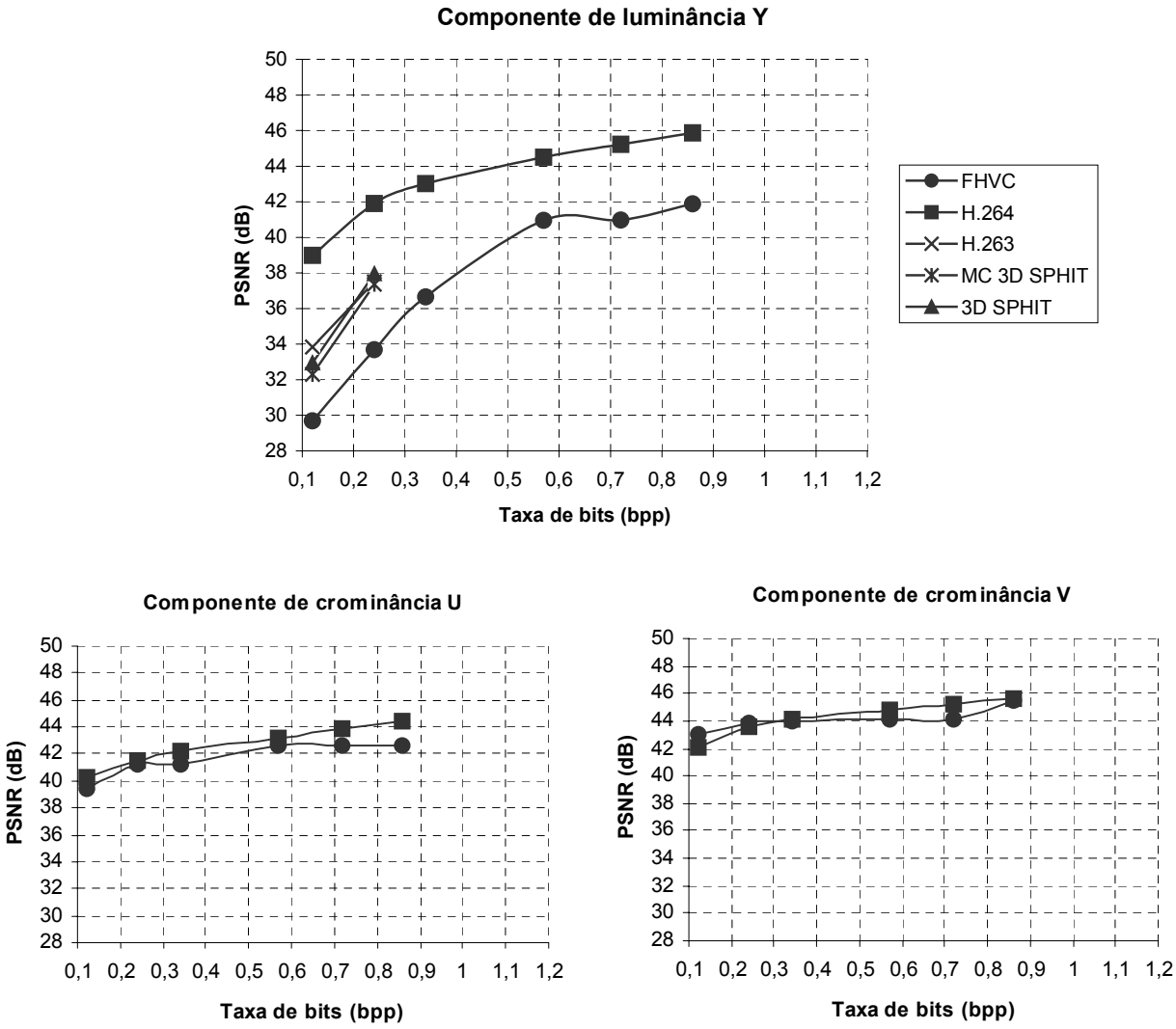


Fig. 4.3.9 - Resultados experimentais para a seqüência "Hall Monitor" (resultados da componente de luminância Y para o H.263, o MC 3D SPIHT e o 3D SPIHT transcritos de [40]).

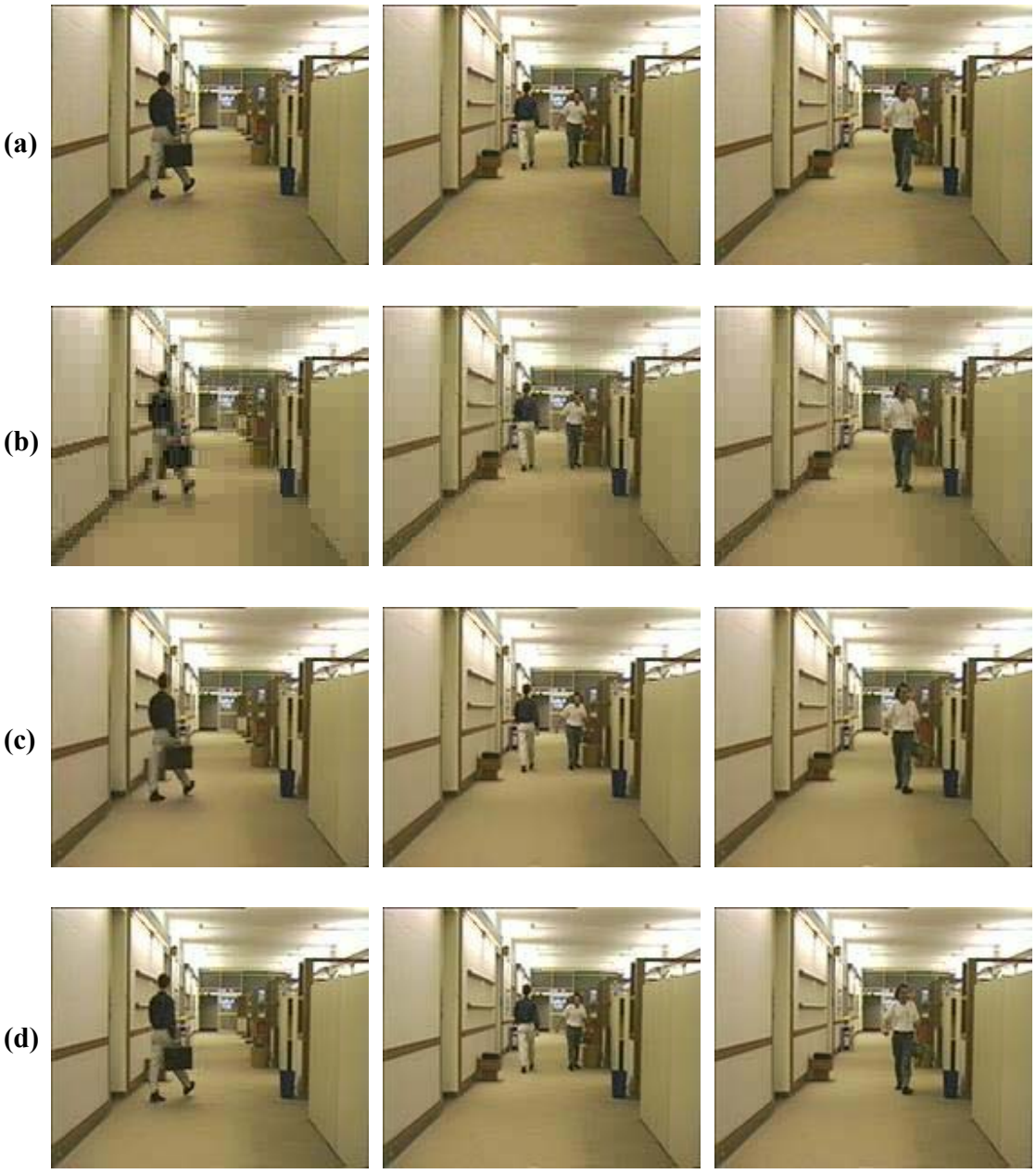


Fig. 4.3.10 - Quadros #42, #180 e #264 originais em (a) e quadros reconstruídos com todas as componentes da seqüência "Hall Monitor" codificada a 0,12 bit/pixel utilizando FHVC em (b) e H.264 em (c) e a 0,57 bit/pixel utilizando FHVC em (d) para comparação da qualidade com as imagens obtidas em (c).

A avaliação da qualidade visual obtida pelo FHVC e pelo H.264 para a mesma taxa de bits por pixel pode ser feita através dos quadros mostrados na Fig. 4.3.10. Nesta figura são analisados: o quadro #42, que corresponde a uma região de bastante movimento logo no início da seqüência; o quadro #180, que também está em um período de boa quantidade de movimento,

mas com maior proporção de região de fundo fixo; e o quadro #264 que está localizado na porção final da seqüência, em um período de movimento menor.

Para a taxa de 0,12 bit/pixel, o FHVC alcança 29 dB para a componente de luminância Y e produz as imagens que estão apresentadas na Fig. 4.3.10(b). A localização temporal dos quadros em períodos da seqüência com características de movimento distintos justifica as diferenças na qualidade visual obtida com a utilização do FHVC na codificação dos quadros #42, # 180 e #264. Já as imagens obtidas com o H.264 não apresentam diferenças de qualidade visual perceptíveis entre os quadros, conforme pode ser observado na Fig. 4.3.10(c). Isto acontece porque o H.264 atinge um valor de PSNR alto de 39 dB para a taxa de 0,12 bit/pixel utilizada.

O gráfico da Fig. 4.3.9 mostra que à taxa de 0,34 bit/pixel, o FHVC já produz imagens reconstruídas de boa qualidade com 37 dB para a componente de luminância Y. Supõe-se, portanto, que o desempenho do FHVC à taxa de 0,34 bit/pixel pode ser comparado ao do H.264 à taxa de 0,12 bit/pixel. Entretanto, a análise da qualidade visual dos quadros produzidos com o FHVC não confirmou esta hipótese e verificou-se que, à taxa de 0,34 bit/pixel os quadros ainda apresentam efeito de blocagem perceptível. Identifica-se aqui um caso em que o valor obtido para a PSNR não representa de forma adequada a qualidade visual das imagens. Neste caso, uma imagem de PSNR alta apresenta qualidade visual ruim, mas em outros casos, como o que será apresentado na próxima seção, imagens de PSNR de valor baixo apresentam boa qualidade visual. Sendo assim, considerou-se que o desempenho em termos de qualidade visual do H.264 à taxa de 0,12 bit/pixel só pode ser comparado ao desempenho do FHVC quando utilizado com taxas superiores a 0,5 bit/pixel. Foram utilizados, então para a comparação de resultados, os quadros apresentados na Fig. 4.3.10(d), que foram codificados com 0,57 bit/pixel.

Os tempos de codificação e de decodificação por quadro obtidos com o FHVC e com o H.264 estão registrados na Tabela 4.3.4. Observa-se que, assim como obtido para as seqüências “Miss America” e “Foreman”, a codificação executada pelo FHVC é aproximadamente 165 vezes mais rápida, em média, do que a codificação executada pelo H.264. Os tempos de decodificação gastos pelo FHVC também se mantiveram 8 vezes menores, na média, do que os tempos gastos pelo H.264.

Na Fig. 4.3.10, a qualidade visual dos quadros codificados pelo H.264 à taxa de 0,12 bit/pixel foi equiparada à qualidade visual dos quadros codificados pelo FHVC à taxa de 0,57 bit/pixel. A codificação executada pelo H.264 à taxa de 0,12 bit/pixel requer 2.855,75 ms por

quadro, enquanto a codificação executada pelo FHVC à taxa de 0,57 bit/pixel requer apenas 19,18 ms. Conclui-se assim, que ao custo de uma redução de 4,75 vezes na taxa de compressão obtida pelo H.264, obtém-se uma codificação aproximadamente 150 vezes mais rápida com o FHVC. Ressalta-se que, assim como para as seqüências "Miss America" e "Foreman", o FHVC pode ser utilizado para a codificação em tempo real da seqüência "Hall Monitor" à taxa original de 30 quadros/segundo.

CODIFICAÇÃO		
Taxa (bpp)	Tempo por quadro (milisegundos)	
	FHVC	H.264
0,86	21,09	3.455,90
0,72	19,70	3.446,74
0,57	19,18	3.237,77
0,34	18,08	3.148,40
0,24	17,92	3.083,59
0,12	17,19	2.855,75

DECODIFICAÇÃO		
Taxa (bpp)	Tempo por quadro (milisegundos)	
	FHVC	H.264
0,86	17,48	141,60
0,72	16,75	141,55
0,57	16,60	136,72
0,34	15,28	132,86
0,24	14,94	130,86
0,12	14,30	126,07

Tabela 4.3.4 - Resultados de tempos de codificação e decodificação para a seqüência "Hall Monitor".

4.3.1.4 Seqüência de vídeo "Mobile"

A seqüência "Mobile" é uma seqüência complexa com grande quantidade de movimento, cores e detalhes espaciais. Podem ser identificados movimentos de objetos de forma horizontal e vertical, oclusão de objetos e *zoom-out* do início até o meio da seqüência. Estas características tornam difícil a obtenção de boas taxas de compressão por todos os codificadores de vídeo que exploram as redundâncias temporais inter-quadros e as redundâncias espaciais intra-quadros, uma vez que estas estão muito reduzidas. Por outro lado, seqüências de vídeo com muitos detalhes espaciais podem apresentar, após a decodificação, valores de PSNR considerados baixos e, ainda assim, apresentar boa qualidade visual. Isso ocorre, por exemplo, com seqüências comprimidas por codificadores que processam os quadros divididos em blocos, tais como o FHVC. O efeito de blocagem que costuma aparecer nas imagens reconstruídas, após serem codificadas por este tipo de codificador, não é tão evidente em seqüências de vídeo com muitas cores e detalhes espaciais do que em seqüências que apresentam grandes regiões uniformes.

Sendo assim, uma vez que a seqüência “Mobile” apresenta pouca redundância temporal e espacial, o FHVC apresentou melhores resultados quando executado com a utilização de cubos menores de 4x4x4. A Fig. 4.3.11 apresenta estes resultados e possibilita a comparação dos mesmos com os obtidos com o H. 264 e com os métodos MPEG-1, MCP (*Motion-Compensated Prediction*) e MC-3DSBC (*Motion-Compensated Three-Dimensional Subband/Wavelet Coding*), cujos resultados para a seqüência “Mobile” foram transcritos de [41]. O artigo [41] descreve o método MC-3DSBC e afirma que o seu desempenho supera o alcançado pelo padrão MPEG-1 e por outros codificadores do tipo MCP. O MC-3DSBC utiliza transformadas wavelets na dimensão espacial e compensação de movimento na dimensão temporal. Os coeficientes resultantes destes processos são quantizados tridimensionalmente pelo 3DB-FSSQ (*3-D Subband- Finite State Scalar Quantization*).

A comparação com os resultados obtidos pelo FHVC na Fig. 4.3.11 demonstra que estes estão muito próximos dos obtidos com o MPEG-1 e com o MCP para a componente de luminância Y e, aproximadamente, 3,5 dB abaixo do resultado obtido com o MC-3DSBC para taxas próximas de 0,5 bit/pixel. Já para as componentes de crominância U e V, verifica-se que o desempenho do FHVC é sempre superior ao apresentado pelos métodos MPEG-1, MCP e MC-3DSBC.

Ao se comparar o desempenho do FHVC em termos de taxa de bits por pixel x PSNR com o desempenho do H.264 para a componente de luminância Y, verifica-se que aquele está, aproximadamente, 12 dB abaixo deste para todas as taxas de bits/pixel mostradas na Fig. 4.3.11. Este desempenho inferior do FHVC em termos de taxa de bits por pixel x PSNR já era esperado devido ao fato das técnicas adotadas pelo FHVC serem tridimensionais e a seqüência “Mobile” apresentar pouca redundância espacial e temporal. Entretanto, este baixo desempenho é compensado pelo alto desempenho em termos de qualidade visual dos quadros reconstruídos, conforme a análise que será apresentada posteriormente nesta seção. Já na comparação do desempenho dos codificadores para as componentes de crominância, observa-se que o FHVC se aproxima bem mais do H.264.

Uma análise interessante a ser feita nos gráficos da Fig. 4.3.11 é a de que, de forma geral, o valor obtido para a PSNR cresce uniformemente com o aumento da taxa de bits/pixel utilizada. Este crescimento uniforme também pôde ser observado para as seqüências “Miss America”, “Foreman” e “Hall Monitor” analisadas anteriormente. Apenas para esta última seqüência,

verifica-se na Fig. 4.3.9, certa saturação do valor do PSNR obtido para a componente de luminância Y, que aumenta pouco para taxas superiores à 0,5 bit/pixel. Esta tendência de saturação é apresentada tanto pelo H.264 quanto pelo FHVC e indica que, para a seqüência "Hall Monitor", taxas superiores a um limiar de 0,5 bit/pixel não precisam ser utilizadas porque não proporcionam seqüências reconstruídas de melhor qualidade.

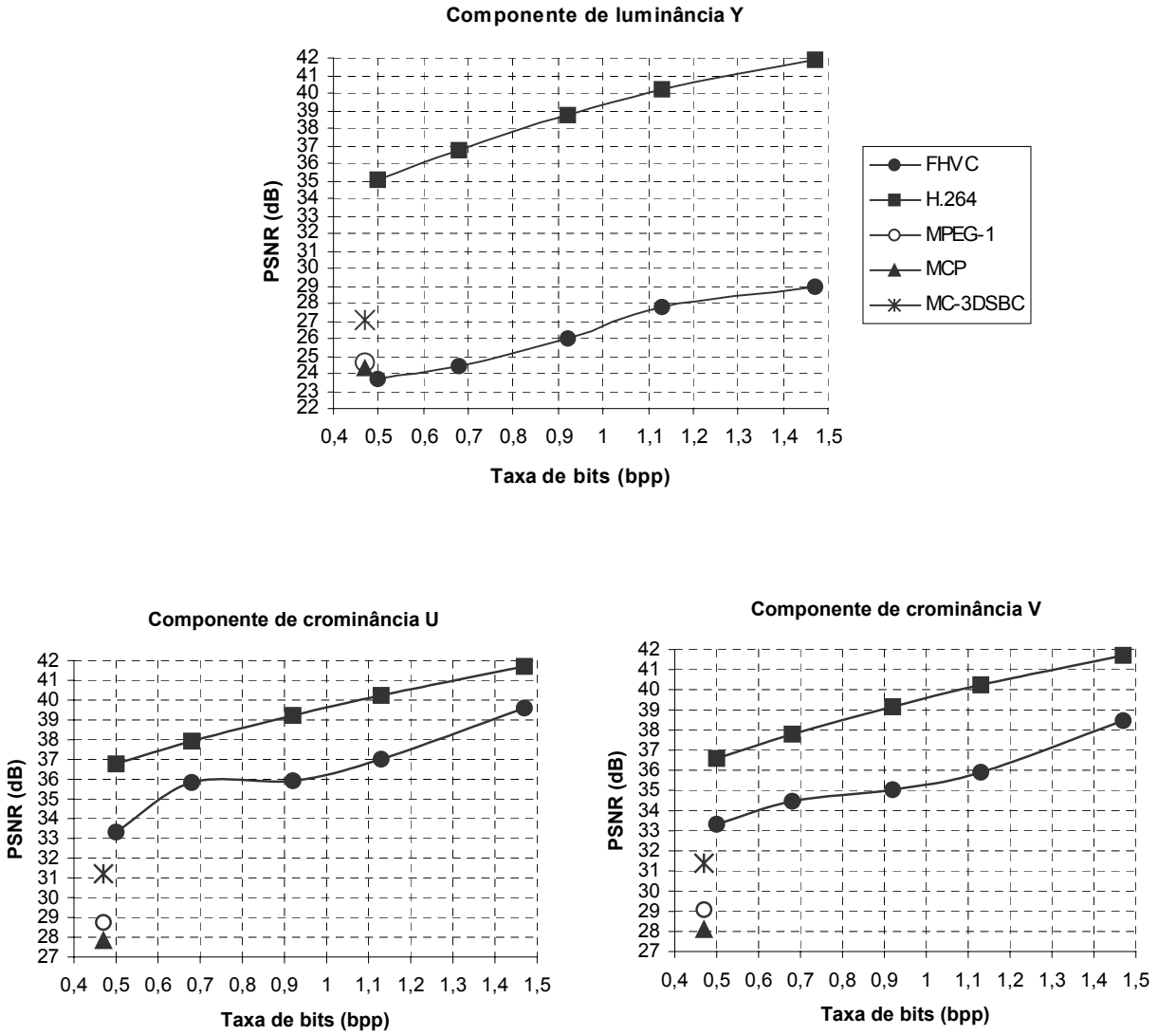
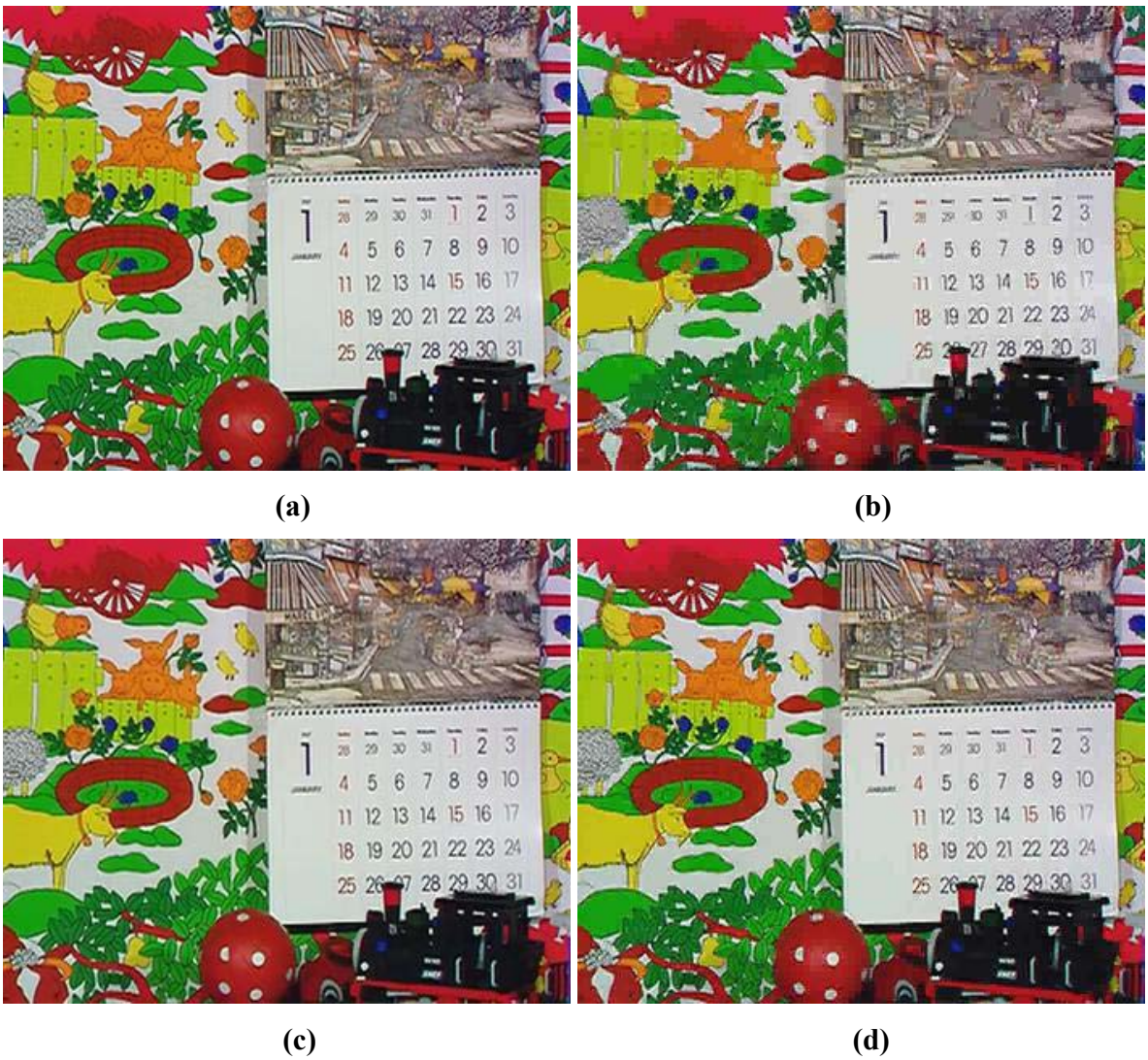


Fig. 4.3.11 - Resultados experimentais para a seqüência "Mobile" (resultados para o MPEG-1, o MCP e o MC-3DSBC transcritos de [41]).

A avaliação da qualidade visual obtida pelo FHVC e pelo H.264 para a mesma taxa de bits por pixel pode ser feita através dos quadros da Fig. 4.3.12 e da Fig. 4.3.13.

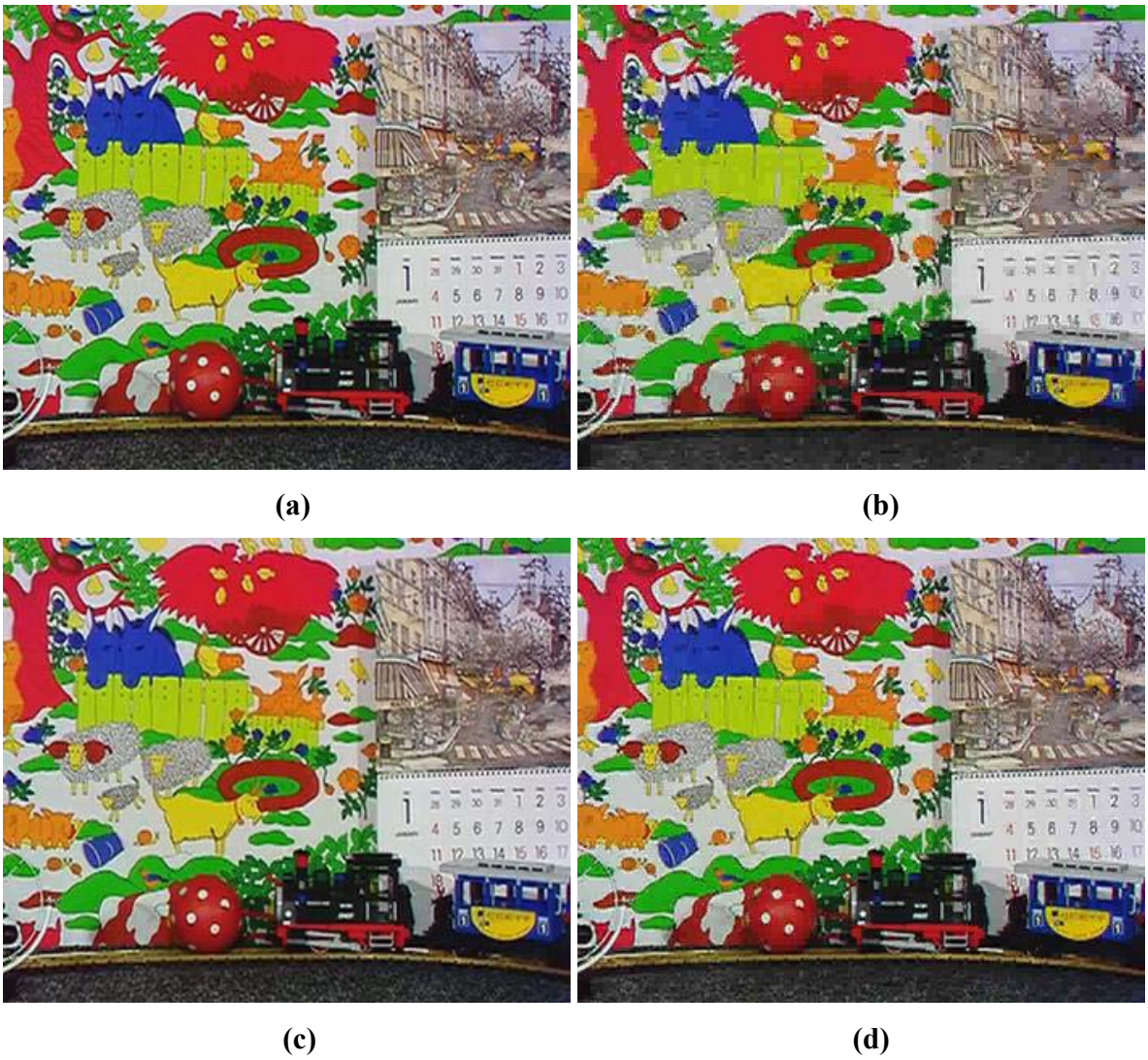
O quadro #40 apresentado na Fig. 4.3.12(a) corresponde a um quadro do início da seqüência localizado em um período de bastante movimento horizontal, vertical e de *zoom*. O quadro #159, por sua vez, apresentado na Fig. 4.3.13(a), está em um período da seqüência em que o movimento de *zoom* termina, mas que um objeto da direita (calendário) desaparece e um novo objeto girante aparece à esquerda. Desta forma, os dois quadros analisados estão localizados em períodos de bastante movimento, mas com características distintas.



(a) **(b)**

(c) **(d)**

Fig. 4.3.12 - Quadro #40 original em (a) e quadros reconstruídos com todas as componentes da seqüência "Mobile" codificada a 0,5 bit/pixel utilizando FHVC em (b) e H.264 em (c) e a 1,13 bit/pixel utilizando FHVC em (d) para comparação da qualidade com a imagem obtida em (c).



(a) **(b)**

(c) **(d)**

Fig. 4.3.13 - Quadro #159 original em (a) e quadros reconstruídos com todas as componentes da seqüência "Mobile" codificada a 0,5 bit/pixel utilizando FHVC em (b) e H.264 em (c) e a 1,13 bit/pixel utilizando FHVC em (d) para comparação da qualidade com a imagem obtida em (c).

A Fig. 4.3.12(b) e a Fig. 4.3.13(b) apresentam os quadros #40 e #159 reconstruídos após terem sido codificados pelo FHVC à taxa de 0,5 bit/pixel. Observa-se que, embora haja perda de detalhes espaciais e nitidez, as imagens destas figuras ainda alcançam boa qualidade visual mesmo apresentando valores de PSNR de 24 dB para a componente de luminância Y. Isto comprova o que foi mencionado no início desta seção e configura um caso em que imagens com valores de PSNR baixo apresentam boa qualidade visual.

Para a comparação com as imagens obtidas pelo H.264, apresentadas na Fig. 4.3.12(c) e na Fig. 4.3.13(c), a taxa de bit/pixel do FHVC foi aumentada para 1,13 bit/pixel. Considera-se

que, à esta taxa, as imagens obtidas com o FHVC, apresentadas na Fig. 4.3.12(d) e na Fig. 4.3.13(d), têm qualidade visual que se aproxima da qualidade obtida pelo H.264 à taxa de 0,5 bit/pixel, embora ainda haja uma certa perda de detalhes nas imagens produzidas pelo FHVC.

Os tempos de codificação e de decodificação por quadro obtidos com o FHVC e com o H.264 estão registrados na Tabela 4.3.5. Observa-se que, a codificação executada pelo FHVC é aproximadamente 184 vezes mais rápida, na média, do que a codificação executada pelo H.264. Este valor indica que a diferença entre os tempos de codificação obtidos pelo FHVC e pelo H.264 aumentou em 10% para a seqüência "Mobile", uma vez que para as seqüências de vídeos analisadas anteriormente, o FHVC era 160 vezes, em média, mais rápido do que o H.264. Este aumento no tempo de codificação gasto pelo H.264 também justifica o distanciamento maior de 12 dB verificado entre os valores de PSNR obtidos pelo H.264 e pelo FHVC para a componente de luminância Y. Já para a decodificação, este aumento de tempo não foi encontrado. Ao contrário, a diferença entre os tempos de decodificação obtidos pelo FHVC e pelo H.264 foi reduzida em 50%, de forma que para a seqüência "Mobile", a decodificação realizada pelo FHVC é apenas 4 vezes mais rápida do que a decodificação realizada pelo H.264.

As diferenças no tempo de decodificação são justificadas pelo fato da seqüência "Mobile" estar no formato CIF (352x288) e portanto, apresentar um tamanho 4 vezes maior do que as seqüências de vídeo analisadas anteriormente, que estavam no formato QCIF (176x144). De fato, observa-se que os tempos de codificação e decodificação obtidos pelo FHVC para a seqüência "Mobile" sofreram exatamente um aumento de 4 vezes com relação aos tempos obtidos para as seqüências QCIF analisadas anteriormente. Isto leva à interessante conclusão de que o FHVC apresenta um relação de aumento uniforme entre tempo de execução e quantidade de pixels a serem codificados e possibilita uma previsão de quantos milisegundos serão necessários por quadro para a codificação de seqüências de vídeo de maior resolução. Este aumento de tempo de fator 4 na codificação de seqüências CIF também pode ser aproximadamente observado para a codificação realizada pelo H.264, mas não é encontrado na sua decodificação. A análise dos tempos de decodificação apresentados pelo H.264 para a seqüência "Mobile" apresenta um aumento de fator 2 apenas em relação às seqüências QCIF analisadas anteriormente. Isto ocorre porque o H.264 possui a característica de ser assimétrico e, sendo assim, a diferença entre os tempos de decodificação obtidos pelo H.264 e pelo FHVC diminuiu para a seqüência "Mobile".

CODIFICAÇÃO		
Taxa (bpp)	Tempo por quadro (milisegundos)	
	FHVC	H.264
1,47	81,04	15.267,63
1,13	79,54	14.658,41
0,92	74,60	1.4323,38
0,68	72,48	13.861,55
0,5	68,22	13.357,19

DECODIFICAÇÃO		
Taxa (bpp)	Tempo por quadro (milisegundos)	
	FHVC	H.264
1,47	70,08	297,43
1,13	66,20	281,85
0,92	63,51	277,59
0,68	59,33	260,71
0,50	54,92	255,84

Tabela 4.3.5 - Resultados de tempos de codificação e decodificação para a seqüência "Mobile".

Na Fig. 4.3.12 e na Fig. 4.3.13, a qualidade visual dos quadros codificados pelo H.264 à taxa de 0,5 bit/pixel foi equiparada à qualidade visual dos quadros codificados pelo FHVC à taxa de 1,13 bit/pixel. A codificação executada pelo H.264 à taxa de 0,5 bit/pixel requer 13.357,19 ms por quadro, enquanto a codificação executada pelo FHVC à taxa de 1,13 bit/pixel requer apenas 79,54 ms. Conclui-se assim, que ao custo de uma redução de 2,23 vezes na taxa de compressão obtida pelo H.264, obtém-se uma codificação 168 vezes mais rápida com o FHVC.

É interessante analisar o tempo total de codificação dos 300 quadros da seqüência de vídeo “Mobile” considerando as condições que estão sendo comparadas. Esta análise aponta que para a codificação da seqüência completa à taxa de 0,5 bit/pixel o H.264 levaria 4.007.157 ms (aproximadamente 1 hora e 7 minutos). Já a codificação da mesma seqüência pelo FHVC, com taxa de 1,13 bit/pixel e qualidade visual semelhante à obtida pelo H.264, levaria aproximadamente apenas 24 segundos.

4.3.2 Análise da variação do espaço de cores interno do codificador

A utilização de um espaço de cores de codificação interno diferente do espaço de cores do arquivo original se mostra muito vantajosa para arquivos de vídeos que estão em formatos que requerem grande quantidade de bits por pixel, tais como o RGB e o YUV 4:4:4, que utilizam 24 bpp. Isto ocorre porque a conversão do formato do arquivo do vídeo ocorre em uma etapa anterior ao início do processo de codificação, logo após a leitura dos valores dos pixels dos quadros do arquivo original. Sendo assim, um arquivo de vídeo no formato RGB, por exemplo, com 24 bpp, pode ser convertido internamente pelo codificador para o formato YUV 4:2:0, com

apenas 12 bpp, e ser codificado como se o formato YUV 4:2:0 fosse o formato original do arquivo de vídeo. Esta conversão não gera perda de qualidade visual perceptível nos quadros do vídeo. Porém, requer tempos maiores de codificação e também de decodificação porque, após o término desta, o arquivo de vídeo é convertido novamente para o seu formato original. Isto é possível porque, tanto o formato do espaço de cores do arquivo de vídeo original quanto o formato do espaço de cores utilizado internamente no codificador, são armazenados no cabeçalho do arquivo de vídeo codificado.

O FHVC suporta apenas os formatos de vídeo originais RGB e YUV 4:2:0, mas para o espaço de cores interno do codificador são suportados os formatos YUV 4:4:4, YUV 4:2:2, YUV 4:2:0, RGB e YC_oC_g. As características destes formatos podem ser encontradas na Seção 2.2.1.

Se o arquivo de vídeo original já estiver no espaço de cores YUV 4:2:0, não haverá ganho em se utilizar outro espaço de cores interno para a codificação porque o formato YUV 4:2:0 é o que utiliza a menor quantidade de bits por pixel entre todos os formatos disponíveis. Esta afirmação pode ser comprovada através da análise dos dados da Tabela 4.3.6, que correspondem à codificação da seqüência "Miss America" no formato original YUV 4:2:0.

Espaço de cores interno	**Taxa (bpp)**	**PSNR (dB)**		
		Componente Y	**Componente U**	**Componente V**
YUV 4:2:0	0,584	43,05	42,20	42,20
YUV 4:2:2	0,574	42,73	41,62	41,62
YC_oC_g	0,543	39,42	42,76	42,76
YUV 4:4:4	0,605	40,24	43,32	43,32
RGB	0,553	40,77	41,96	41,96

Tabela 4.3.6 – Comparação entre os desempenhos de taxa x PSNR obtidos com a utilização de diferentes espaços de cores internos na codificação da seqüência "Miss America" com o formato original YUV 4:2:0.

Porém, se o arquivo de vídeo original estiver no formato RGB, haverá um bom ganho com a sua conversão inicial para o formato YUV 4:2:0 e algum ganho com a conversão para os formatos YUV 4:2:2 e YC_oC_g. Estes ganhos podem ser analisados em termos de taxa de bits por pixel x PSNR nos dados apresentados na Tabela 4.3.7. Estes dados correspondem à codificação da seqüência "Miss America" no formato original RGB. Realmente, verifica-se que, para taxas de bits por pixel semelhantes, a conversão inicial da seqüência para o formato YUV 4:2:0

proporciona o alcance de valores de PSNR maiores do que os obtidos se o próprio espaço de cores RGB for utilizado como o espaço de cores interno do codificador.

Espaço de cores interno	Taxa (bpp)	PSNR (dB)		
		Componente Y	Componente U	ComponenteV
YUV 4:2:0	0,376	35,289	38,572	34,815
YUV 4:2:2	0,385	33,894	37,735	34,485
YC_oC_g	0,376	34,193	36,445	34,250
YUV 4:4:4	0,376	34,140	36,270	34,139
RGB	0,385	33,883	36,884	34,520

Tabela 4.3.7 – Comparação entre os desempenhos de taxa x PSNR obtidos com a utilização de diferentes espaços de cores internos na codificação da seqüência "Miss America" com o formato original RGB.

4.3.3 Análise da variação da quantidade de quadros por cubo codificado

O FHVC apresenta a possibilidade de poder ser executado com cubos de tamanho 4x4x4 ou 8x8x8. De forma geral, cubos maiores, de tamanho 8x8x8, exploram melhor as redundâncias temporal e espacial existentes nos quadros. Por outro lado, o uso destes cubos maiores pode gerar efeitos de blocagem perceptíveis em seqüências com regiões coloridas uniformes e de baixa resolução, tais como as seqüências no formato QCIF. Já o uso de cubos menores, de tamanho 4x4x4, produz bons resultados em seqüências com muitos detalhes e que não apresentem um alto grau de redundância temporal e espacial. Além disso, se a codificação em tempo real for considerada, o atraso causado pelo armazenamento de quatro quadros da seqüência é duas vezes inferior ao atraso causado pelo armazenamento de oito quadros.

Sendo assim, o uso de cada um dos tamanhos de cubos tem suas vantagens e desvantagens que devem ser balanceadas de acordo com as características particulares de cada seqüência de vídeo. Dentre as seqüências de vídeo analisadas neste trabalho, enquanto as seqüências no formato QCIF "Miss America", "Foreman" e "Hall Monitor" foram codificadas com cubos de tamanho 8x8x8, a seqüência "Mobile" no formato CIF foi codificada com cubos de tamanho 4x4x4.

Os gráficos apresentados na Fig. 4.3.14 mostram os resultados de desempenho em termos de taxa de bits por pixel x PSNR obtidos pelo FHVC na codificação da seqüência QCIF "Miss

America" no formato YUV 4:2:0. Observa-se que, tanto para a componente de luminância Y quanto para as componentes de crominância U e V, os valores obtidos com a utilização de cubos com 4 quadros foram superiores aos resultados obtidos com a utilização de cubos com 8 quadros. Este resultado já era esperado, uma vez que a seqüência "Miss America" apresenta grande quantidade de redundância temporal e espacial.

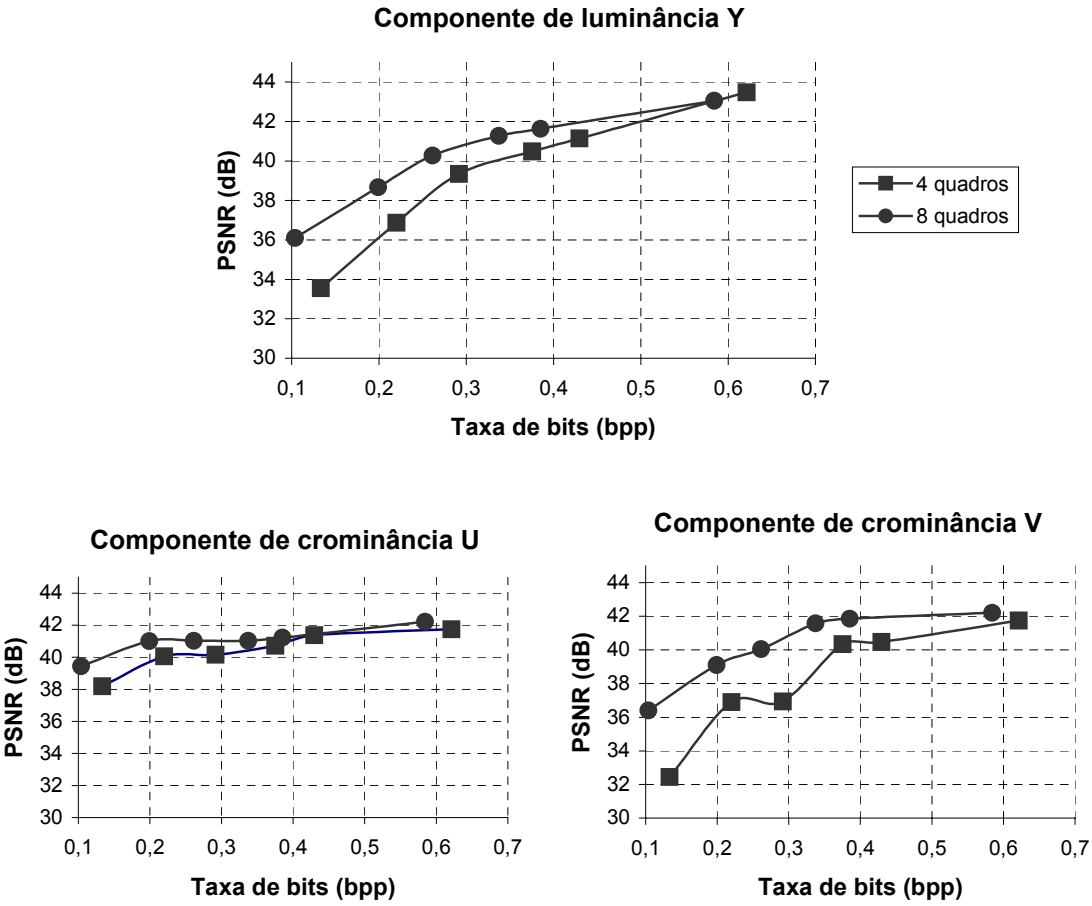


Fig. 4.3.14 – Resultados experimentais obtidos com a seqüência "Miss America" para comparação entre codificação com cubos 4x4x4 e cubos 8x8x8.

Capítulo 5
CONSIDERAÇÕES FINAIS

O objetivo deste capítulo é apresentar as considerações finais sobre os aspectos mais significativos deste trabalho e apresentar algumas propostas de trabalhos futuros para o desenvolvimento desta linha de pesquisa.

5.1 CONCLUSÃO GERAL

A principal contribuição deste trabalho foi o desenvolvimento de um sistema completo de codificação e decodificação de vídeo focado na obtenção de tempos de processamento reduzidos e no uso de poucos recursos computacionais. Para atingir tais objetivos, foram pesquisadas e implementadas técnicas já existentes adequadas a cada uma das etapas do processo de codificação. Para a leitura dos coeficientes obtidos através da transformada de Hadamard implementada de forma tridimensional foi desenvolvida uma nova técnica. Uma nova forma de implementação também foi desenvolvida para a obtenção de um sistema de baixo custo computacional, através da alocação otimizada de recursos de memória e da utilização somente de operações matemáticas inteiras de adição, subtração e deslocamentos binários. Devido a estas características, o sistema foi denominado FHVC (*Fast Hadamard Video Codec*).

Para a execução dos processos de codificação, decodificação e redução de taxas de bits de arquivos já codificados, o FHVC possui interfaces gráficas, onde todos os parâmetros de execução necessários podem ser configurados. A modelagem computacional do sistema foi feita de acordo com a linguagem UML, seguiu os princípios de engenharia de *software* e está descrita neste trabalho.

O objetivo de redução do tempo de execução do codificador foi atingido através do uso da transformada rápida de Hadamard implementada de forma eficiente e utilizada tridimensionalmente para a exploração tanto das redundâncias espaciais quanto das redundâncias temporais das seqüências de vídeo.

Outro objetivo do FHVC é a utilização de seqüências de vídeo coloridas, uma vez que este poderia ser utilizado para transmissão de vídeo real em uma rede de grande disponibilidade de banda. Para explorar também as diferentes características dos vários formatos existentes para

arquivos de vídeo coloridos, foi acrescentada ao sistema a capacidade de conversão entre os formatos.

Uma característica muito interessante adicionada ao sistema foi a capacidade de produzir seqüências codificadas de forma embutida, onde seqüências comprimidas com taxas maiores possuem internamente seqüências de menores taxas. Esta codificação progressiva está presente no FHVC devido ao uso de um codificador de entropia de Golomb adaptativo onde cada plano de bits dos coeficientes da transformada de Hadamard tridimensional é processado separadamente a partir do mais significativo. Antes de serem processados pelo codificador de entropia, os coeficientes de cada cubo são lidos e organizados de forma a melhorar o desempenho do codificador. A característica do FHVC de realizar codificação progressiva permite, por exemplo, que a taxa de bits por pixel de um arquivo de vídeo codificado seja reduzida conforme o arquivo seja transmitido através de uma rede que pode apresentar nós (gargalos) com diferentes capacidades de transmissão.

Em relação à complexidade relativa das etapas de transformada tridimensional e de codificação de entropia para todos os planos de bits dos coeficientes, pode-se considerar os resultados obtidos para a codificação da seqüência “Mobile” no formato CIF. Estes resultados indicam percentuais de tempos de 37% e 63%, respectivamente para as etapas mencionadas, excluindo-se os tempos de leitura do arquivo de vídeo original, ordenação dos coeficientes da transformada e escrita do arquivo de vídeo codificado. Ressalta-se que o tempo referente à codificação de entropia pode ser significativamente reduzido se a codificação não for realizada para todos os planos de bits dos coeficientes.

Os resultados obtidos com o FHVC foram comparados com os resultados obtidos com o padrão H.264 e com outros resultados provenientes de vários artigos que também descrevem métodos de codificação tridimensional de vídeo. Verificou-se que, para a codificação das componentes de luminância dos arquivos de vídeo, o desempenho do FHVC, em termos de taxas de bits por pixel x PSNR, foi próximo ao desempenho apresentado para outros métodos de codificação tridimensionais. O bom desempenho do FHVC é verificado especialmente para a codificação das componentes de crominância.

Mas o melhor resultado de desempenho do FHVC está em termos do tempo de codificação e decodificação x qualidade visual da seqüência de vídeo reconstruída. Observou-se que o FHVC obtém tempos de codificação, em média, 160 vezes mais rápidos do que o *software*

de referência oficial do H.264 para imagens de qualidade visual semelhante e taxa de compressão reduzida por um fator de 2 a 5 em relação à taxa obtida pelo H.264. Além disso, verificou-se que os tempos de codificação e decodificação reduzidos do FHVC tornariam possíveis a codificação e decodificação em tempo real de seqüências QCIF implementadas por *software* em computadores pessoais, tais como o computador em que os resultados deste trabalho foram obtidos.

5.2 PROPOSTAS DE TRABALHOS FUTUROS

Algumas sugestões são feitas a seguir para melhorar o desempenho do FHVC ou para adicionar outras funcionalidades e possibilidades de uso ao mesmo.

- **Filtro de blocagem**: o maior problema verificado nas imagens reconstruídas após a codificação com o FHVC é o efeito de blocagem, que aparece, principalmente quando a seqüência de vídeo é codificada a baixas taxas de bits por pixel ou apresenta grande quantidade de movimento. O uso de um filtro de blocagem nas seqüências de vídeo reconstruídas poderia reduzir este problema e aumentar a qualidade das imagens obtidas;

- **Aumento da resolução dos arquivos de vídeo**: os testes feitos com o FHVC só utilizaram seqüências de vídeo nos formatos QCIF e CIF que apresentam baixa resolução. Para a utilização com arquivo de vídeo reais, o FHVC deveria ter seu desempenho analisado para seqüências de maior resolução. Para que o FHVC continuasse a ter tempos de execução reduzidos poderiam ser realizadas implementações em *hardware* ou testes em máquinas de maior capacidade;

- **Utilização do FHVC em tempo real**: uma adaptação interessante seria mudar a implementação do FHVC de forma que a leitura dos quadros fosse feita através de uma câmera de vídeo e estes fossem codificados em tempo real, transmitidos para o decodificador em *streaming* de vídeo e decodificados, em seguida, também em tempo real;

- **Desenvolvimento de uma versão preditiva do FHVC**: na versão implementada neste trabalho, o uso de codificação preditiva (através de técnicas de estimação e compensação de movimentos) foi evitado com o objetivo de simplificar o algoritmo. No entanto, sabe-se que o uso de codificação preditiva produz resultados atraentes em termos da relação taxa x distorção. Sendo assim, poderia ser desenvolvida uma versão MC-FHVC, em que os cubos transmitidos transportariam os valores dos resíduos em relação às predições encontradas.

- **Estudo para distribuição de IPTV**: na IPTV (*Internet Protocol Television*) tem-se a TV distribuída sobre uma rede IP de banda larga. Neste tipo de serviço, também está prevista a situação em que o próprio usuário, originalmente o receptor da informação, se transforma em produtor de conteúdo. O uso do FHVC poderia ser estudado para codificação do conteúdo a ser distribuído pelo usuário. Além disso, se o usuário desejar assistir ao conteúdo da IPTV em um aparelho convencional de TV, o FHVC poderia ser adicionado ao decodificador, uma vez que já foi desenvolvido para ser executado em um *set-top box*.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] SILVA, Davison Gonzaga. **Implementação de um Sistema SIP para o Sistema Operacional Linux.** 2003. 1 v. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas, 2003.

[2] LIM, Jae S.. **Two-dimensional Signal and Image Processing.** New Jersey, USA: Prentice-Hall Information Processing Series, 1990.

[3] PENNEBAKER, William B.; MITCHELL, Joan L.; FOGG, Chad E.; LEGALL, Didier J. **MPEG Video Compression Standard.** New York, USA: Chapman & Hall, 1996.

[4] JAIN, Anil K. **Fundamentals of Digital Image Processing**. New Jersey, USA: Prentice-Hall Information and System Sciences Series, 1989.

[5] PENNEBAKER, William B.; MITCHELL, Joan L. **JPEG Still Image Data Compression Standard**. New York, USA: Van Nostrand Reinhold, 1993.

[6] SULLIVAN, Gary; ESTROP, Stephen. **Video Rendering with 8-bit YUV Formats**. Redmond, WA, USA: Microsoft Digital Media Division, 2003.

[7] WILSON, Dave. **RGB/YUV Pixel Conversion.** FOURCC.org. Disponível em: <http://www.fourcc.org/fccyvrgb.php>. Acesso em: 01 ago. 2005.

[8] WILSON, Dave. **Compressed Formats**. FOURCC.org. Disponível em <http://www.fourcc.org/>. Acesso em: 01 ago. 2005.

[9] MALVAR, Henrique; SULLIVAN, Gary. **YC_oC_g-R: A Color Space with RGB Reversibility and Low Dynamic Range**. Redmond, WA, USA: Microsoft Corporation, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 2003.

[10] RABBANI, Majid; JONES, Paul W. **Digital Image Compression Techniques**. Washington, USA: SPIE -The International Society for Optical Engineering, 1991.

[11] RICHARDSON, Iain E. G. **H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia**. UK: John Wiley & Sons, 2003.

[12] PARK, Neungsoo; PRASANNA, Viktor. Cache Conscious Walsh-Hadamard Transform. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, 2001, v. 2, p. 1205 - 1208.

[13] JOHNSON, Jeremy; PÜSCHEL, Markus. In Search of the Optimal Walsh-Hadamard Transform. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, 2000, v. 6, p. 3347-3350.

[14] JOHNSON, Jeremy. **What is the WHT anyway, and why are there so many ways to compute it?**. Philadelphia, PA, USA: Drexel University MSC Society, 2003.

[15] LAM, W.-M.; LU, L. Memory Reduction for HDTV Decoders. **IBM Journal Of Research And Development,** p. 545-553. 1 jul. 1999.

[16] BERNABE, Gregorio; GONZALEZ, Jose; GARCIA, Jose Manuel; DUATO, Jose. A New Lossy 3-D Wavelet Transform for High-Quality Compression of Medical Video. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY APPLICATIONS IN BIOMEDICINE, 2000, p. 226-231.

[17] FURHT, Borko; WESTWATER, Raymond. Three-Dimensional DCT Video Compression Technique Based on Adaptive Quantizers**.** In: PROCEEDINGS OF THE SECOND IEEE INTERNATIONAL CONFERENCE ON ENGINEERING OF COMPLEX COMPUTER SYSTEMS, 1996, p. 189 - 198**.**

[18] CHAN, Raymond K. W.; LEE, M. C. 3D-DCT Quantization as a Compression Technique for Video Sequences. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON VIRTUAL SYSTEMS AND MULTIMEDIA, Geneva, Switzerland, 1997, p. 188-196.

[19] KIM, Beong-Jo; PEARLMAN, William A. An Embedded Wavelet Video Coder Using Three-Dimensional Set Partitioning in Hierarchical Trees (SPIHT). In: PROCEEDINGS OF THE DATA COMPRESSION CONFERENCE, 1997, p. 251-260.

[20] BAUER, Mark; SAYOOD, Khalid. Video Coding using 3 Dimensional DCT and Dynamic Code Selection. In: PROCEEDINGS OF THE DATA COMPRESSION CONFERENCE, 1995, p. 451.

[21] OLIVEIRA, Fabrício Corrêa de Araújo. **Compressão Progressiva de Imagens Usando DCT e Código de Golomb**, 2002. 1 v. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas, set. 2002.

[22] OLIVEIRA, F. C. de A.; COSTA, M. H. M. Embedded DCT Image Encoding. In: PROCEEDINGS OF THE INTERNATIONAL TELECOMMUNICATIONS SYMPOSIUM, Natal, RN, Brazil, 2002.

[23] SHAPIRO, J. M. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, 1993, v. 41, n. 12, p. 3445 - 3462.

[24] SAID, A.; PEARLMAN, W. A. A New Fast and Efficient Image Coded Based on Set Partitioning in Hierarchical Trees. In: PROCEEDINGS OF THE IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, 1996, v. 6, p. 243 – 250.

[25] SHEN, K.; DELP, Edward J. Color Image Compression Using an Embedded Rate Scalable Approach. In: PROCEEDINGS OF IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 1997, v. 3, p. 34 - 37.

[26] LAMAR, Marcus Vinicius. **Codificação de Vídeo Utilizando Decomposição Quadtree para Estimação de Movimentos**. 1996. 1 v. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Florianópolis, 1996.

[27] MARCA, José Roberto B. An LSF Quantizer for the North-American Half-Rate Speech Coder. In: PROCEEDINGS OF THE IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, 1994, v. 43, n. 3, p. 413 - 419.

[28] PRESSMAN, Roger S. **Software Engineering: a practioner's approach.** 5. ed. New York, USA: McGraw-Hill, 2001.

[29] RATIONAL UNIVERSITY. **Object-oriented Analysis and Design Using the UML**. Student Manual Book 1. USA, 2000.

[30] RICHTER, Jeffrey. **Applied Microsoft .NET Framework Programming**. Washington, USA: Microsoft Press, 2003.

[31] YUVPLAYER.COM. **YUV Player Deluxe**. Disponível em: <http://www.yuvplayer.com/>. Acesso em 01 set. 2005.

[32] YUVVIEWER. **YUVViewer.exe**. Disponível em: <http://ftp3.itu.ch/av-arch/jvt-site/software_tools/>. Acesso em: 01 ago. 2005.

[33] INSTITUTO MILITAR DE ENGENHARIA. **Seqüências de vídeo**. Disponível em: <http://ginfo05.dee.ime.eb.br/seqs.html>. Acesso de: 01 mai. 2005 a 01 nov. 2005.

[34] VIDEO TRACES RESEARCH GROUPS. **YUV 4:2:0 Video Sequences**. Disponível em: <http://trace.eas.asu.edu/yuv/yuv.html>. Acesso de: 01 mai. 2005 a 01 nov. 2005.

[35] RENSSELAER POLYTECHNIQUE INSTITUTE. **CIPR QCIF Sequences**. Disponível em: <http://www.cipr.rpi.edu/resource/sequences/qcif.html>. Acesso em 01 set. 2006.

[36] BERTULANI, C.A. **Luz e Cor.** Disponível em: <http://www.if.ufrj.br/teach/luz/cor.html>. Acesso em: 01 set. 2006.

[37] GOKTURK, Salih Burak; AARON, Anne Margot F. **Applying 3D Methods to Video for Compression**. Stanford EE 392J Final Report. USA, 2002.

[38] SHEN, Ke; DELP, Edward J. Wavelet Based Rate Scalable Video Compression. In: PROCEEDINGS OF THE IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, 1999, v. 9, n. 1, p. 109 – 122.

[39] BOTTREAU, Vincent; BÉNETIÈRE, Marion; FELTS, Boris; PESQUET-POPESCU, Béatrice. A Fully Scalable 3D Subband Video Coded. In: PROCEEDINGS OF THE IEEE CONFERENCE ON IMAGE PROCESSING, 2001, v. 1, p. 1017 – 1020.

[40] KIM, Beong-Jo; XIONG, Zixiang, PEARLMAN, William A. Low Bit-Rate Scalable Video Coding with 3D Set Partitioning in Hierarchical Trees (3D SPIHT). In: PROCEEDINGS OF THE IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, 2000, v. 10, n. 8, p. 1374 – 1387.

[41] CHOI, Seung-Jong; WOODS, Hohn W. Motion-Compensated 3D Subband Coding of Video. In: PROCEEDINGS OF IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 1999, v. 8, n. 2, p. 155 – 167.

[42] **H.264/AVC reference software version JM 11.0.** Disponível em: <http://iphome.hhi.de/suehring/tml/download/>. Acesso em: 01 dez. 2006.

[43] **H.264/AVC reference software manual (JVT-Q042).** Disponível em: <http://iphome.hhi.de/suehring/tml/>. Acesso em: 01 dez. 2006.

[44] ITU-T. **Recommendation ITU-R BT.601-5:** Studio encoding paremeters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, 1995.

[45] CONNOLLY, C.; FLEISS, T. A study of efficiency and accuracy in the transformation from RGB to CIELAB color space. In: PROCEEDINGS OF IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 1997, v. 6, n. 7, p. 1046 – 1048.

[46] POYNTON, Charles. New foundations for video technology. In.: PROCEEDINGS OF THE SMPTE ADVANCED TELEVISION AND ELECTRONIC IMAGING CONFERENCE, 1995, p. 167-180.

[47] PRATT, W. K.; KANE, J.; ANDREWS, H. C. Hadamard transform image coding. In.: PROCEEDINGS OF IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 1969, v. 57, n. 1, p. 58 – 68.

[48] LEE, Moon Ho. The center weighted Hadamard transform. In.: PROCEEDINGS OF THE IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. 1989, v. 36, n. 9, p. 1247 – 1249.

[49] JENG, J. H.; TRUONG, T. K.; SHEU, J. R. Fast fractal image compression using the Hadamard transform. In: PROCEEDINGS OF IEEE INTERNATIONAL CONFERENCE ON IMAGE AND SIGNAL PROCESSING, 2000, v. 147, n. 6, p. 571 – 574.

[50] CHRYSAFIS, Christos; ORTEGA, Antonio. Efficient Context-Based Entropy Coding Lossy Wavelet Image Compression. In.: PROCEEDINGS OF IEEE DATA COMPRESSION CONFERENCE, 1997, p. 241 – 250.

[51] LEI, S. -M.; SUN, M.-T. .An entropy coding system for digital HDTV applications. In.: PROCEEDINGS OF THE IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. 1991, v. 1, n. 1, p. 147 – 155.

[52] MALVAR, Henrique; HALLAPURO, Antti; KARCZEWICZ, Marta; KEROFSKY, Louis. Low-complexity transform and quantization with 16-bit arithmetic for H.26L. In.:

PROCEEDINGS OF THE IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. 2003, v. 13, n. 7, p. 598-603.

[53] JAYANT, N. S.; NOLL, Peter. **Digital Coding of Waveforms – Principles and Applications to Speech and Video**. New Jersey, USA: Prentice-Hall Signal and Processing Series, 1984.