COMPRESSÃO EFICIENTE DE VÍDEO

Daniel Monteiro de Barros Pinheiro

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

Aprovada por:

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

Prof. Gelson Vieira de Mendonça, Ph.D.

Prof. Abraham Alcaim, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

FEVEREIRO DE 2005

PINHEIRO, DANIEL

Compressão Eficiente de Vídeo [Rio de Janeiro] 1999

xi, 75 pp 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1.Compressão de Vídeo

2.Compressão sem Perdas

3.Imagens Estéreo

4.Padrão H.264

I.COPPE/UFRJ II.Título (série)

Agradecimentos

Os meus agradecimentos são para todos aqueles que direta ou indiretamente colaboraram para a elaboração dessa tese. Em especial, gostaria de agradecer a:

Meus pais, José Luiz e Maria Verônica, que me deram o todo o apoio dividindo os momentos mais importantes da minha vida.

Meus irmãos Leonardo e Patrícia que estiveram sempre do meu lado com amizade, consideração, respeito e apoio.

Meu orientador, Professor Eduardo, por todo o incentivo, suporte e paciência durante essa difícil jornada.

Meu chefe Sílvio Pereira pelo constante incentivo e apoio que possibilitaram a realização desse trabalho de forma simultânea com meu emprego.

Aos amigos do DEPED (Departamento de Pesquisa e Desenvolvimento) da TV Globo por todo o apoio, incentivo e amizade.

Ao amigo Eduardo Barbosa, pelas infindáveis dicas e macetes no LaTeX.

Ao amigo Danillo Graziosi, pelas sugestões, dicas, críticas e colaborações durante o desenvolvimento desse trabalho.

Todos meus amigos que me apoiaram nos momentos mais difíceis, comemoraram cada conquista e foram peças fundamentais para que esse trabalho fosse concluído.

Enfim, a todas essas pessoas, deixo meu muito obrigado, na esperança de que esse trabalho esteja correspondendo a todas as espectativas.

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

COMPRESSÃO EFICIENTE DE VÍDEO

Daniel Monteiro de Barros Pinheiro

Fevereiro/2005

Orientador: Eduardo Antônio Barros da Silva

Programa de Engenharia Elétrica

A codificação de vídeo é um problema cuja solução deve ser projetada de acordo com as necessidades da aplicação desejada. Nessa tese novos métodos de compressão são apresentados. Esses métodos visam tanto melhorias nos formatos de compressão atuais quanto novas propostas e conceitos aplicados em compressão. Em primeiro lugar dois métodos inéditos para compressão de vídeo sem perdas são apresentados. Em seguida mostramos a aplicação do padrão H.264, que é destinado a compressão de vídeo, para compressão de pares de imagens estéreo. Além disso um estudo de um problema de *flickering* existente em um dos modos de codificação do H.264 é realizado e um método capaz de atingir uma quase eliminação do efeito é apresentado. Os resultados apresentados dentro de cada capítulo mostram a eficácia de cada um dos métodos propostos bem como idéias para melhorias em trabalhos futuros.

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

EFFICIENT VIDEO COMPRESSION

Daniel Monteiro de Barros Pinheiro

February/2005

Advisor: Eduardo Antônio Barros da Silva

Department: Electrical Engineering

A solution to a video coding problem should be developed according to the requirements of the application. In this thesis, new methods for video compression are presented. These methods aiming at improving the current compression formats and also new also presenting new proposals and concepts applied to compression. At first, two new methods for lossless video compression are presented. After that, we show the use of the H.264 standard, which is designed for video compression, to the compression of stereo pairs. Also a study on a *flickering* problem observed in one of the compression modes of the H.264 standard is presented and a method that can almost eliminate the effect is proposed. The results presented in each chapter show the efficiency of each method and present ideas for improvements in future works.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com a evolução da eletrônica e a explosão da informática, vemos uma sociedade cada vez mais globalizada, clamando por mais conforto e qualidade de vida. Da mesma forma, a evolução das telecomunicações vem unindo povos de todo o mundo, sem distinção de raça, credo ou cor.

Nesse contexto, a necessidade de se comunicar tem buscado esforços para que novas formas de interação sejam disponibilizadas para sociedade, não somente com a função de integração social, mas também contribuindo para a evolução de outras ciências, como a medicina.

Desde os primeiros *beeps* do telégrafo nos idos de 1830, passando para os sons do rádio e logo em seguida para as imagens em uma televisão, é notável a necessidade da disponibilização de novas mídias, garantindo uma interação cada vez mais real entre as pessoas.

Dessa forma, nos últimos anos muitas atenções têm sido depositadas na transmissão de imagens mais realísticas e convincentes, além da necessidade de se transmitir informações visuais até mesmo em meios onde o som é o propósito principal, como nos telefones celulares. Logo, um crescente aumento na quantidade de informações necessárias para a transmissão ou armazenamento das imagens tem sido observada, acarretando na necessidade da elaboração de mecanismos mais evoluídos e complexos para processá-las.

O vídeo constitui uma mídia com uma quantidade muito grande de informação, e por isso formas de compactação dessa informação são buscadas incessantemente por pesquisadores. Agregando-se a essa necessidade, são buscados métodos

que primam pela fidelidade visual, robustez a erros de transmissão e também complexidade de processamento, já que poderiam ser utilizados por dispositivos portáteis com poder de processamento e consumo de energia reduzidos.

Essa tese visa a exploração de novos métodos para compressão de vídeo, seja explorando métodos já consagrados e propondo aprimoramentos, mas também apresentando novos algorítmos de compressão. Aqui apresentamos dois métodos originais para compressão de vídeo sem perdas, assunto que é abordado no capítulo 3. No capítulo 4 mostramos que o emergente padrão H.264 apresenta considerável eficiência na codificação de imagens estéreo, que é uma aplicação não proposta pelo padrão. Em seguida, no capítulo 5, apresentamos um problema em um dos modos de codificação do H.264 que acarreta em um efeito de *flickering* extremamente desagradável. Nesse capítulo uma solução que praticamente elimina esse efeito é apresentada. Finalmente, no capítulo 6 concluímos o trabalho e propomos sugestões para pesquisas e aprimoramentos futuros.

Capítulo 2

Princípios de Codificação de Vídeo

2.1 Introdução

O processo de captação de vídeo com posterior digitalização resulta em uma sequência de imagens amostradas nas direções horizontal e vertical. Esses pontos de amostragem são denominados *pixels*. Dessa forma, cada pixel representa um elemento luminoso da imagem que contém uma cor específica. Assim, armazenar essas imagens significa armazenar a cor de cada um de seus pixels.

Considerando o padrão de cores primárias RGB, com 8 bits para armazenar cada uma das componentes e uma frequência de quadros por segundo em 30Hz, para o qual as mudanças entre quadros são imperceptíveis ao sistema visual humano, teríamos a quantidade de informação a ser armazenada de acordo com a tabela 2.1.

Tabela 2.1: Resoluções típicas de imagem

Resolução	Pixels	Bits/segundo	Equivalente Analógico
176x144	25344	17 Mbps	Video-conferência
352x288	101376	69 MBps	Qualidade VHS
720x486	349920	240 Mbps	TV padrão NTSC
1920x1080	2073600	1422 Mbps	HDTV

Como podemos ver, mesmo imagens muito pequenas como as de vídeo-conferência possuem quantidade de informação muito superior a que qualquer meio de transmissão ou armazenagem pode suportar, sendo inviável utilizá-las da forma

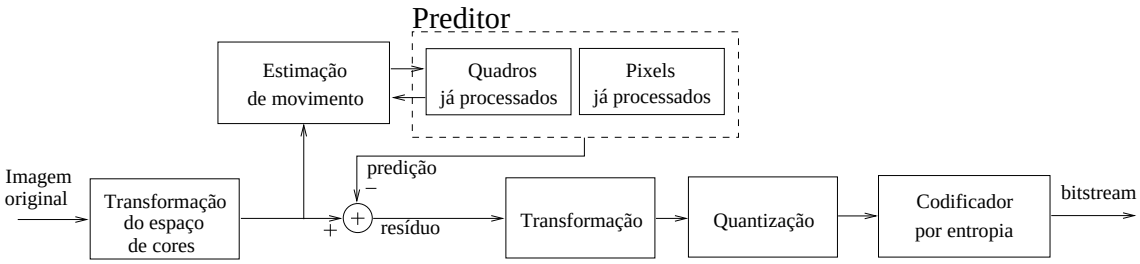


Figura 2.1: Diagrama em blocos de um codificador

como são captadas. Assim, os processos de compressão visam reduzir a quantidade de dados, transformando-as em códigos que possam facilmente trafegar pelos meios de comunicação acessíveis nos dias de hoje.

Um processo de compressão como na figura 2.1 consiste na digitalização da imagem seguido de uma transformação no seu espaço de cores. O espaço RGB possui excessiva correlação entre suas componentes e por isso espaços de cores mais eficientes podem ser utilizados (seção 2.2). Em seguida, uma modelagem da imagem utilizando-se de preditores e estimadores é realizada. Nessa predição podem ser utilizados pixels que já foram processados e mesmo quadros anteriores armazenados no codificador. Os quadros anteriores passam por um processo de estimação de movimento (seção 2.3, visando torná-los mais redundantes em relação ao quadro a ser processado e assim melhorar a eficiência do processo de predição. Após o processo de modelagem, a imagem que originalmente era descrita por pixels no formato RGB é agora um conjunto de descritores com os quais o decodificador seria capaz de reconstituí-la. Essa reconstituição pode ser idêntica à imagem original, e teríamos assim um codificador *sem perdas* (*Lossless*). Caso a imagem reconstruída apresente diferenças em relação original, temos um codificador *com perdas* (*Lossy*).

Em seguida, o processo de *codificação* é responsável por atribuir códigos aos símbolos gerados pela *modelagem*. O sinal resultante do processo de modelagem contém símbolos que possuem determinada probabilidade de ocorrência. Em geral, em imagens naturais, códigos residuais em torno de zero são bastante prováveis, já que uma boa modelagem acarreta em uma boa predição para o pixel em questão, fazendo com que o resíduo de codificação tenha uma distribuição bem concentrada em torno de zero. Atribuindo-se uma quantidade pequena de bits para os símbolos mais prováveis, no caso os símbolos em torno de zero, temos uma boa taxa de compres-

são. O processo de codificação pode se dar através de tabelas de código montadas experimentalmente, ou também através de métodos adaptativos, que computam as distribuições de probabilidades dinamicamente e gerando tabelas otimizadas para cada imagem específica.

2.2 Transformação do Espaço de Cores

A percepção das cores pelo sistema visual humano é causada pelos *cones*, que são células sensíveis à cor e ficam localizadas na retina, mais especificamente em uma região denominada fóvea. Também na retina, junto dos cones, ficam situados os *bastonetes*, que são células sensíveis à niveis de luminosidade, mas não são capazes de distinguir entre diferentes comprimentos de onda das cores [1] [5].

As cores são percebidas como uma combinação de 3 comprimentos de onda: um longo, um medio e um curto, que representamos pelas três cores primárias: vermelho, verde e azul (padrão RGB). Entretanto, outras representações para as cores podem ser utilizadas. Espaços de cor que possuem uma das componentes indicando o nível de luz do sinal (*luminância*) e as outras duas representando a cor correspondente têm sido propostos. Esse sistema de coordenadas aproxima a representação de um elemento visual à percepção do sistema visual humano.

O sistema visual humano possui maior resolução espacial na percepção da luminância do que na crominância, um processamento diferenciado para ambos proporcionam eficiência aos algorítmos de compressão. Esses algorítmos se utilizam de espaços de cor alternativos, como Y-Cb-Cr (seção 2.2.1) e Y-Co-Cg (seção 2.2.2) destinando menor resolução espacial e menor número de *bits* para as componentes de crominância do que para a luminância.

2.2.1 Y-Cb-Cr

O espaço Y-Cb-Cr está entre os mais populares entre os codificadores de vídeo [6]. Y representa a componente de luminância da imagem e é calculada como uma média ponderada das componentes R, G e B. A informação de crominância é representada como a diferença entre cada canal (R, G e B) e a componente de luminância Y.

A conversão é descrita na tabela 2.2.

Tabela 2.2: Conversão RGB $\leftrightarrow$ YCbCr

RGB $\leftarrow$ YCbCr	YCbCr $\rightarrow$ RGB
$Y = 0.299R + 0.587G + 0.114B$	$R = Y + 1.402Cr$
$Cb = 0.564(B - Y)$	$G = Y - 0.344Cb - 0.714Cr$
$Cr = 0.713(R - Y)$	$B = Y + 1.772Cb$

O espaço Y-Cb-Cr tem como desvantagem as operações com números em ponto flutuante, tornando a transformação computacionalmente mais lenta, e impactando na sua reversibilidade.

2.2.2 Y-Co-Cg

A transformação YCoCg proporciona um bom descorrelacionamento entre as bandas de cor [12] e tem a propriedade de ser completamente reversível, o que a torna adequada a aplicações onde perdas não são aceitáveis. Além disso, tem a característica de ser computacionalmente muito rápida, pois pode ser implementada utilizando somente adições e deslocamentos de bits.

$$\begin{bmatrix} Y \\ Co \\ Cg \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & -\frac{1}{2} \\ 1 & 0 & -1 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.1}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2} & -\frac{1}{2} \\ 1 & 0 & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} Y \\ Co \\ Cg \end{bmatrix} \tag{2.2}$$

A conversão pode ser feita com a sequência de operações descrita na tabela 2.3, onde o operador >> indica a operação de deslocamento de bits para a direita (*shift right*).

O espaço YCoCg tem como única desvantagem o aumento da faixa dinâmica das componentes de cor (Co e Cg) em 1 bit, não havendo alteração para a componente Y. Porém esse aumento da faixa dinâmica é compensado pelo bom descorrela-

Tabela 2.3: Conversão RGB $\leftrightarrow$ YCoCg

RGB $\leftarrow$ YCoCg	YCoCg $\rightarrow$ RGB
$Co = R - B$	$t = Y - (Cg >> 1)$
$t = B + (Co >> 1)$	$G = Cg + t$
$Cg = G - t$	$B = t - (Co >> 1)$
$Y = t + (Cg >> 1)$	$R = B + Co$

cionamento proporcionado, o que resulta numa boa performance de codificação em se tratando de redundância espectral.

2.3 Estimação de Movimento

A exploração de redundâncias temporais em um codificador de vídeo é feita utilizando quadros anteriores ou posteriores no processo de predição. Entretanto, em uma sequência de vídeo, os quadros são compostos por objetos que se movem com o tempo ou mesmo um fundo que se movimenta devido ao movimento de câmera na captação. Dessa forma, na tentativa de se obter um quadro de referência mais redundante em relação ao quadro a ser codificado, uma estimação de movimento é realizada [2] [6] [3].

Nos codificadores abordados nessa tese são modelados apenas os movimentos de translação dos objetos e do fundo de câmera. O processo é realizado dividindo-se a imagem em blocos e para cada um desses blocos é buscado no quadro de referência o bloco mais adequado. A distância entre os dois blocos é representada através de um vetor de movimento que deve ser armazenado no *bitstream*. A medida do grau de semelhança entre dois blocos é em geral dada pela *Soma das Diferenças Absolutas* (SAD) de todos os pixels no bloco.

O método de implementação mais simples e que é capaz de encontrar o melhor vetor de movimento para um determinado bloco é o método da *Busca Exaustiva* (*Full Search*). Como esse método tem sérios problemas de velocidade, outros métodos rápidos como a *Busca em 3 passos* (*Three-Step-Search*) podem ser empregados. Esses métodos são descritos nas seções 2.3.1 e 2.3.2.

2.3.1 Busca Exaustiva

O método da busca exaustiva consiste em testar todos os vetores de movimento possíveis dentro da faixa de busca estabelecida pelas normas do codificador. Esse método consome um poder de processamento elevado, devido à quantidade de operações necessárias para o resultado. Como todas as possibilidades são testadas, o mínimo global é sempre encontrado e esse é o vetor de movimento ótimo para o macrobloco.

Se tomarmos uma janela de busca de (-15, 15) pixels, as componentes do vetor podem assumir 31 possibiliadades. Assim teríamos 31 × 31 (961) vetores a serem testados para cada macrobloco da imagem.

2.3.2 Busca em 3 passos

O método da busca em três passos é um método rápido que visa encontrar o vetor de movimento que fornece um menor valor de SAD em um menor número possível de testes. Como não são testadas todas as possibilidades de vetores, é possível que um mínimo local seja atingido ao invés do mínimo global, como acontece no método da *Busca Exaustiva*. Entretanto a velocidade de processamento do algorítmo compensa a diferença para o mínimo global em aplicações práticas.

O método funciona escolhendo-se uma janela de busca de $+/-(2^N-1)$ pixels.

1. Computar o SAD para a posição (0,0)
2. Setar o passo de busca $S = 2^{N-1}$
3. Computar o SAD para 8 posições $+/-S$ ao redor de (0,0)
4. Das 9 posições computadas, marcar como origem a de menor SAD
5. Setar o passo $S = S/2$
6. Repetir estágios 3-5 até termos $S = 1$

A figura 2.2 demonstra o funcionamento da busca para uma janela de $+/-7$ pixels. Os números nos pixels representam em qual passo o vetor correspondente é testado. O círculo em negrito representa o vetor com menor SAD [6].

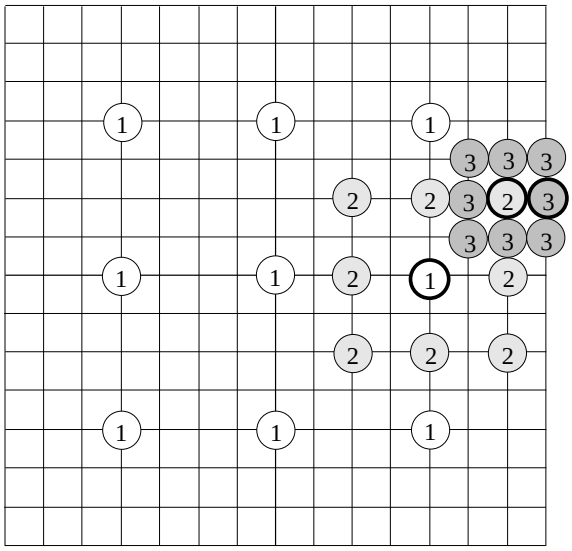


Figura 2.2: Busca em Três Passos

O método genericamente necessita de $(8N - 1)$ comparações para convergir. Assim, para uma janela de $+/-15$ pixels, teríamos 33 comparações para cada macrobloco, que corresponde a 3% do número de comparações do método de *Busca Exaustiva*.

2.4 Codificador Aritmético

Após o estágio de modelagem de um quadro da sequência, os símbolos que o descrevem devem ser codificados e armazenados no *bitstream*. Entretanto, ganhos de compressão podem ser obtidos já que alguns símbolos provavelmente se repetem e dessa forma um menor número de *bits* podem ser a eles atribuidos. Essa compressão deve ser feita sem perdas (*lossless*), já que no decodificador devemos recuperar os símbolos exatamente como eles foram gerados no codificador, para que o modelo seja reproduzido com fidelidade e exista sincronismo entre codificador e decodificador [1].

Os codificadores por entropia funcionam através de modelos estatísticos que computam a distribuição de probabilidades dos símbolos e atribuem um menor número de *bits* para símbolos com alta probabilidade e um maior número de *bits* para símbolos com baixa probabilidade. O codificador aritmético trabalha de forma adaptativa, de modo que as tabelas com as probabilidades dos símbolos são alteradas ao longo do processo de codificação. Dessa forma, o codificador se adapta à estatística

independente do conteúdo [11], não sendo necessárias tabelas com probabilidades fixas coletadas em processos experimentais. A taxa de compressão se aproxima da entropia, já que o número de *bits* utilizados no código é dependente da probabilidade do símbolo a ser codificado. Entretanto, o codificador aritmético tem como desvantagem o alto custo computacional, tanto para a codificação quanto para a decodificação.

A idéia básica por trás do codificador aritmético é mapear uma sequência de entrada em uma palavra de código (*codeword*). Embora uma palavra de código seja representada por um número inteiro de *bits*, o número médio de *bits* por símbolo é obtido dividindo-se o comprimento da palavra de código pelo número de símbolos de entrada codificados nessa palavra. Assim, para uma sequência de M símbolos, a taxa média de *bits* satisfaz a relação da equação (2.3) e se aproxima da quantidade ótima (2.4) à medida que o tamanho M da sequência se torna muito grande. Em (2.3) e (2.4), representamos como H a entropia do sinal e como B_c a taxa média de *bits*.

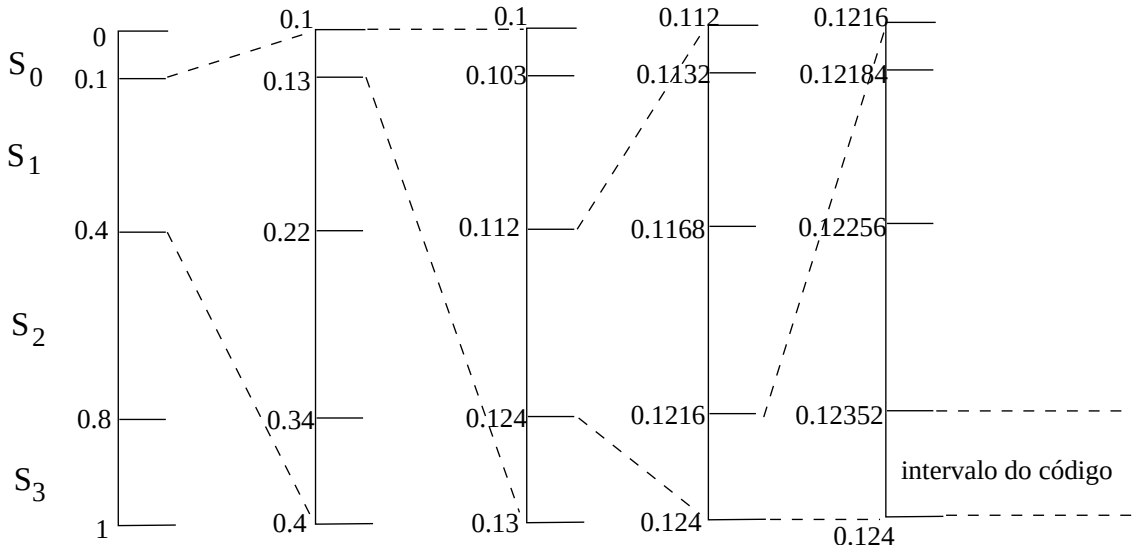
$$\frac{H(S^{(M)})}{M} \leq B_c \leq \frac{H(S^{(M)})}{M} + \frac{1}{M} \tag{2.3}$$

$$\lim_{M \to \infty} \frac{H(S^{(M)})}{M} \tag{2.4}$$

Seja S o alfabeto da fonte com N símbolos $s_0, ..., s_{N-1}$. Seja $p_k = P(s_k)$ a probabilidade de ocorrência do símbolo s_k, com $0 \leq k \leq (N-1)$. Como inicialmente temos apenas um símbolo ($M = 1$), o codificador aritmético começa dividindo o intervalo [0,1) em N subintervalos, onde cada intervalo é associado com um símbolo s_k e tem tamanho igual à sua probabilidade p_k. Seja $[L_{s_k}, H_{s_k})$ o intervalo associado ao símbolo s_k, onde $p_k = H_{s_k} - L_{s_k}$. Na prática, os limites de subintervalo L_{s_k} e H_{s_k} para o símbolo s_k são iguais à soma das probabilidades como nas equações (2.5) e (2.6).

$$L_{s_k} = \sum_{i=0}^{k-1} p_k \quad ; \qquad 0 \leq k \leq (N-1) \tag{2.5}$$

$$H_{s_k} = \sum_{i=0}^{k} p_k \quad ; \qquad 0 \leq k \leq (N-1) \tag{2.6}$$



Assim, o algorítmo de codificação é baseado na seguinte sequência:

1. $L_c = 0$; $H_c = 1$
2. Calcular o comprimento do subintervalo: $comp = H_c - L_c$
3. Ler próximo símbolo de entrada s_k
4. Atualizar o intervalo
   $L_c = L_c + comp \times L_{s_k}$
   $H_c = L_c + comp \times H_{s_k}$
5. Repetir do passo 2 até que a sequência esteja completamente codificada

Podemos visualizar um exemplo do processo de codificação na figura 2.4. Nesse exemplo, a sequência $s_1 s_0 s_2 s_3 s_3$ foi codificada. Ao codificar o símbolo s_0 o intervalo de codificação passa a ser [0.1, 0.4]. Assim, os novos intervalos são calculados de acordo com as probabilidades dos símbolos. Ao ser codificado o símbolo s_1 o intervalo passa a ser [0.1, 0.13]. E assim seguindo até que ao codificar último símbolo s_3 utilizamos o intervalo $[0.12352, 0.124]$ que identifica o código da sequência.

Qualquer número real no intervalo $[L_c, H_c)$ pode ser utilizado como representação da sequência codificada.

O processo de decodificação é realizado da forma:

1. $L_c = 0$; $H_c = 1$

2. Calcular o comprimento do subintervalo: $comp = H_c - L_c$

3. Encontrar o subintervalo de símbolo $[L_{s_k}, H_{s_k})(0 \leq k \leq N-1)$ para o qual $L_{s_k} \leq \frac{codeword - L_c}{comp} < H_{s_k}$

4. Saída do símbolo s_k

5. Atualizar o subintervalo
   $L_c = L_c + comp \times L_{s_k}$
   $H_c = L_c + comp \times H_{s_k}$

6. Repetir do passo 2 até que o último símbolo seja decodificado

2.5 Compressão de Imagens sem Perdas

2.5.1 Introdução

A compressão de imagens sem perdas visa a representação de uma imagem utilizando uma menor quantidade de bits, através de um processo completamente reversível, isto é, a imagem recuperada deve ser uma cópia fiel da imagem original.

Ao contrário da compressão com perdas, onde é almejada a maior fidelidade possível dada uma taxa de compressão desejada, a compressão sem perdas não possui medidas de fidelidade como parâmetro, e as análises e processamentos utilizados não utilizam critérios perceptuais, já que distorções mesmo imperceptíveis ao sistema visual humano não são aceitáveis.

Com isso, taxas muito inferiores de compressão, como da ordem de dois ou três para um, são alcançadas se compararmos com algorítmos que admitem perdas. Entretanto, os algorítmos de compressão sem perdas possuem importante atuação em aplicações onde não são permitidas perdas de fidelidade no sinal, como imagens médicas, imagens para análises científicas ou em aplicações de arquivamento de imagens sem perda de qualidade.

2.5.2 Método

Os algoritmos de compressão de imagem sem perdas consistem basicamente de 2 estágios: *modelagem* e *codificação*.

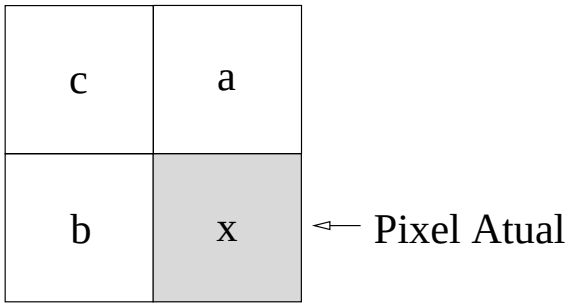


Figura 2.3: Grade de predição do algoritmo LOCO-I

Grande parte das imagens naturais são formadas por objetos ou pessoas que se localizam por cima de um fundo, geralmente uma paisagem. Tanto os objetos quanto o fundo normalmente apresentam uma textura homogênea, isto é, pixels adjacentes possuem características de cor e iluminação semelhantes, exceto se constituirem parte de uma borda do objeto em questão. Considerando-se que bordas são menos frequentes do que regiões planas de uma imagem, boas aproximações podem ser alcançadas para determinado pixel bastando uma observação dos pixels que o circundam. O processo de *modelagem* consiste em um algoritmo capaz de realizar as previsões do pixel, considerando-se pixels que já são conhecidos dado um sistema causal.

2.5.3 LOCO-I: *Low Complexity, Context-Based, Lossless Image Compression Algorithm*

O algoritmo LOCO-I [8] consiste de um preditor causal, como mostrado na Figura 2.3, onde o pixel a ser codificado é denotado por X e os pixels a, b e c denotam os pixels de sua redondeza.

Esse preditor atua gerando uma estimativa $\hat{x}$ do valor do pixel x, através de uma detecção de bordas bem simples. Em primeiro lugar um teste é realizado nos pixels a, b e c para averiguar a existência de uma aresta vertical ou horizontal. Caso uma aresta seja detectada, a predição segue a direção onde supostamente existe homogeneidade na imagem. Caso não sejam detectadas bordas horizontais ou verticais, o algoritmo assume um preditor plano, já que uma suavidade é esperada na imagem.

$$\hat{x} = \begin{cases} \min(a,b) & \text{se } c \geq \max(a,b) \\ \max(a,b) & \text{se } c \leq \min(a,b) \\ a+b-c & \text{outros casos} \end{cases} \tag{2.7}$$

O resíduo da predição x - $\hat{x}$ é codificado através de dicionários seleiconados de acordo com um contexto determinado por medidas de gradiente, utilizando também pixels da redondeza de x. A utilização de variados contextos condiciona a probabilidade de ocorrência de determinado resíduo à textura da região da imagem em questão. No caso do LOCO-I, os gradientes são utilizados para modelar a textura de uma redondeza de x.

2.6 Codificação de Pares de Imagens Estéreo

Os sistemas estereoscópicos proporcionam uma percepção de profundidade ao observador e tem tido aplicações diversas, destacando-se a área médica e entretenimento. Assim, surge a necessidade de um método capaz de codificar de forma eficiente o par estéreo, e isso é feito explorando-se as redundâncias binoculares entre as imagens [13].

O sistema visual humano é capaz de ter a sensação de profundidade através de um par de imagens captadas pelos olhos que têm uma distância que os separam. A geometria 3D de um ponto da cena é calculado a partir de sua projeção no par estéreo. A diferença entre as posições dos objetos entre as duas imagens que compõem o par é chamada de *disparidade*.

A obtenção dessas medidas de disparidade do par estéreo constituem no problema central dos codificadores estéreo. Para isso, o estimador de disparidades precisa selecionar um ponto no espaço de uma das imagens e identificar a mesma posição na outra imagem do par. Após medidas as posições devemos calcular a distância entre as duas medidas. Entretanto, essa estimação de disparidades enfrenta problemas que são difíceis de serem modelados, como oclusão de objetos, ou seja, regiões visíveis em uma das imagens mas não visíveis na outra. Essas oclusões são ocasionadas pela diferença no ângulo de captação das imagens, já que as vistas são separadas por uma certa distância.

Uma solução mais simples para se codificar o par estéreo seria codificar cada uma das imagens de forma independente utilizando um codificador qualquer [15]. Entretanto, nessa abordagem as redundâncias entre as imagens que formam o par estéreo não são exploradas. Os codificadores estéreo então codificam uma das imagens como se fosse uma imagem estática. Já a segunda imagem do par é codificada utilizando-se um mapa de disparidades para a imagem de referência (a vista esquerda, por exemplo) [14]. A imagem residual resultante da compensação da disparidade, bem como o mapa de disparidades são codificados e armazenados no *bitstream*. Entretanto, a modelagem e a estimação do mapa de disparidades é um problema complexo devido à oclusão e uma boa estimação é fundamental para uma boa eficiência do algorítmo de codificação.

2.7 O Padrão H.264

Depois de finalizados os trabalhos do padrão H.263, projetado especialmente para videotelefonia, pois realizava codificação a taxas muito baixas (a partir de 64Kbps), o VCEG (Video Coding Experts Group) iniciou um trabalho de aprimoramento na norma original. Um dos trabalhos consistia no desenvolvimento de novos modos de codificação para o H.263, chamado de "short-term solution"e que deu origem ao H.263 Versão 2 ou H.263++. Também foi estipulado o início de um trabalho mais elaborado, uma solução de longo prazo, onde se esperava que trouxesse grandes inovações em codificação de vídeo. Essa norma levaria o nome de H.26L (Long term solution). Em 2001, o grupo MPEG da ISO/IEC, que já estava com o desenvolvimento do MPEG-4 em andamento, se juntou com o VCEG e levou à criação do JVT (Joint Video Team). Esse novo grupo passou a conduzir o desenvolvimento da norma H.26L, que agora é chamada de MPEG-4 Parte 10/AVC (Advanced Video Coding) ou H.264 [21].

O H.264 consegue qualidades de vídeo muito superiores a padrões anteriores, utilizando para isso taxas muito menores. Por exemplo, para uma mesma qualidade de imagem o H.264 é capaz de codificar com uma taxa 50% menor do que padrões consagrados como MPEG-2.

As inovações trazidas pelo H.264 possibitam a sua utilização em um grande

número de diferentes sistemas. Isso se deve à flexibilidade encontrada no padrão, principalmente porque o H.264 apresenta excelente desempenho tanto para vídeos em baixíssimas taxas (videofone) quanto a vídeos a altas taxas (televisão de alta definição). As principais características do H.264 são [22]:

- **Até 50% de redução de banda comparado com H.263++ ou MPEG-4:** Os algoritmos de predição espacial e temporal, com maior capacidade de adaptação à textura das imagens e do vídeo, possibilitam uma compressão muito mais eficiente.

- **Vídeo de alta qualidade em todas as taxas:** Contando com filtros que afetam a qualidade subjetiva das imagens, o H.264 proporciona imagens agradáveis ao observador, mesmo a taxas muito baixas.

- **Suporte a vídeo entrelaçado:** Esse suporte, que é inexistente no H.263, possibilita a utilização do padrão H.264 na codificação de imagens em ambientes de televisão.

- **Resistência a erros de transmissão:** Com algoritmos de detecção de erro embutidos, o H.264 consegue ser robusto a canais de transmissão onde exista uma probabilidade de erros considerável, como canais de telefonia celular.

- **"Network Friendliness":** A implementação independente do protocolo de rede possibilita que o H.264 seja utilizado em uma gama muito maior de aplicações.

- **Alta complexidade:** O grande número de possibilidades de codificação aliado à complexidade aritmética do algoritmo causa uma dificuldade extra tanto na implementação por hardware quanto por software.

Em essência o H.264 segue os mesmos princípios gerais comuns à maioria dos codificadores de vídeo padronizados hoje: divisão das imagens em macroblocos, estimação de movimento, DCT e codificação por entropia com códigos de comprimento variável [21]. Entretanto, o H.264 contém aprimoramentos dessas ferramentas, conduzindo a um algoritmo capaz de codificar vídeo com melhor qualidade a uma taxa mais baixa que os padrões atuais. Abaixo estão listadas as principais inovações técnicas do algoritmo:

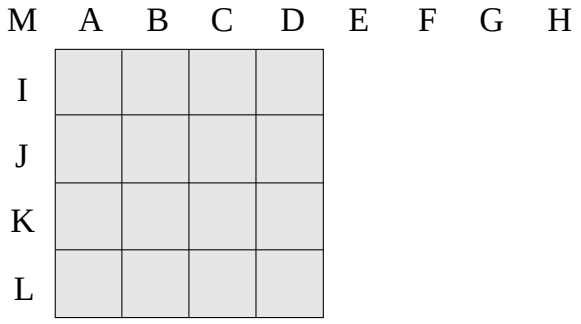


Figura 2.4: Vizinhança para predição espacial INTRA (bloco 4x4)

- Predição espacial direcional para codificação INTRA
- Compensação de movimento com blocos de tamanho variável
- Estimação de movimento com precisão de $^1/_4$ de pixel
- Utilização de múltiplas imagens de referência na estimação de movimento
- Filtro para remoção de efeito de blocos dentro do ciclo de codificação
- Transformada de baixa complexidade e com inversa "exata"
- Codificação aritmética com contextos adaptativos

Nessa tese, o modo de predição INTRA é abordado no capítulo 5. Assim uma descrição mais detalhada desse recurso do H.264 é feita a seguir, na seção 2.7.1.

2.7.1 Predição Espacial INTRA

Os blocos codificados como INTRA são blocos que não possuem informação temporal agregada. Esses blocos devem ser codificados como se o quadro se tratasse de uma imagem estática. Para que a textura da imagem seja utilizada como predição para o bloco, o H.264 define 9 tipos de predição para um bloco com dimensão 4x4 e 4 modos para blocos com dimensão 16x16. Essas predições são calculadas utilzando-se os pixels que já foram decodificados (à esquerda e acima do pixel em questão), ou seja, que estão na vizinhança "causal"da imagem.

A Figura 2.4 define a vizinhança causal usada para a predição INTRA em um bloco 4x4.

Na Figura 2.5, pode-se ver os seguintes 9 modos de predição:

- Modo 0 (Vertical)
- Modo 1 (Horizontal)
- Modo 2 (DC)
- Modo 3 (Diagonal Esquerda-Abaixo)
- Modo 4 (Diagonal Direita-Abaixo)
- Modo 5 (Vertical-Esquerda)
- Modo 6 (Horizontal-Abaixo)
- Modo 7 (Vertical-Direita)
- Modo 8 (Horizontal-Acima)

Para cada bloco de dimensão 4x4, o codificador escolhe qual a melhor predição para esse bloco. O índice da predição é transmitido para o decodificador como informação lateral. Em seguida, o codificador computa o resíduo, subtraindo o bloco real da sua predição. Nesse resíduo são aplicadas a transformada, a quantização e a codificação por entropia.

O mesmo processo se aplica em blocos 16x16, porém 4 modos de predição são definidos:

- Modo 0 (Vertical)
- Modo 1 (Horizontal)
- Modo 2 (DC)
- Modo 3 (Plano)

Os modos *Horizontal*, *Vertical* e *DC* são semelhantes aos utilizados nos blocos 4x4. Já no modo *Plano*, uma função linear é aplicada nos pixels da vizinhança causal do bloco. Esse modo de predição é eficiente em blocos onde as variações na luminância são suaves.

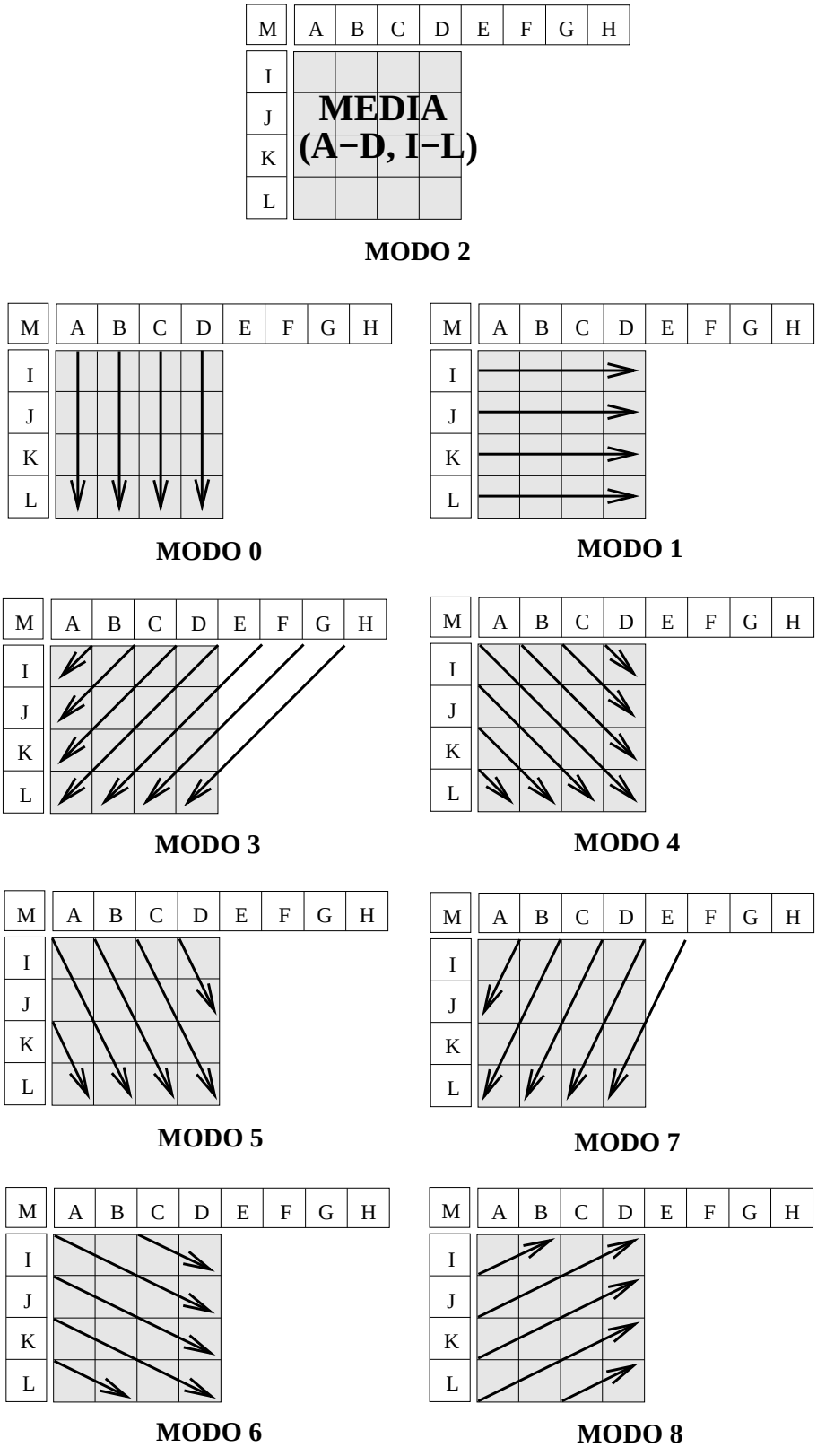


Figura 2.5: Modos de predição INTRA

Capítulo 3

Codificação de Vídeo sem Perdas

Utilizando-se dos conceitos de compressão sem perdas para imagens estáticas, nesse capítulo são abordados dois métodos desenvolvidos para compressão de vídeo sem perdas.

Os algorítmos se caracterizam basicamente pela utilização das predições espaciais dos algoritmos sem perdas para imagens estáticas, acrescido de uma análise das redundâncias temporais, ou seja, redundâncias existentes entre dois quadros consecutivos de vídeo. A exploração dessas redundâncias temporais no processo de predição age de forma a reduzir a entropia do sinal e consequentemente a uma redução na taxa de bits necessárias para a sua representação.

Em primeiro lugar é apresentado o Método do Preditor Tridimensional, onde temos uma extensão espaço-temporal do preditor LOCO-I, apresentado na seção 2.5.3. Após verificar problemas de performance nesse preditor, um novo método que utiliza informação lateral para indicar o melhor preditor para cada região da imagem é apresentado. Chamamos esse método de Método do Chaveamento de Preditores.

Em ambos os métodos, o quadro anterior é utilizado nas análises da predição temporal. Assim, de maneira a utilizar um quadro anterior com conteúdo mais próximo do quadro atual, uma compensação de movimento é realizada. Nessa compensação, a imagem é dividida em macroblocos de 16x16 pixels e para cada um desses macroblocos um vetor de movimento é associado. Limitamos as componentes horizontal e vertical do vetor de movimento de cada macrobloco a valores inteiros na faixa (-16, 16). Nas simulações realizadas do codificador, o método de busca exaustiva *Full Search* apresentado na seção 2.3.1 foi utilizado como estimador de

movimento. Os vetores de movimento são codificados como informação lateral no *bitstream*.

Chamando de INTER o quadro codificado com os métodos aqui apresentados, definimos também um quadro INTRA, no qual todos os pixels são codificados utilizando o preditor espacial LOCO-I convencional. O código INTRA é equivalente ao código gerado por métodos de compressão sem perdas para imagem estáticas, e portanto a comparação do desempenho do quadro INTER em relação ao quadro INTRA mostra a eficiência da utilização das informações temporais na geração da predição.

O *bitstream* é gerado utilizando-se um codificador aritmético adaptativo (seção 2.4), com contextos independentes para os resíduos de predição dos pixels e as componentes dos vetores de movimento dos macroblocos.

Os métodos são aqui apresentados para codificação de uma sequência com apenas um canal (luminância). Para codificação de sequências coloridas, utilizamos uma transformação de cor para um espaço de cores com boa descorrelação das componentes e a aplicação do mesmo método para cada componente. Nas simulações utilizamos o espaço Y-Co-Cg [12] apresentado na seção 2.2.2.

Para as simulações foram utilizadas as sequencias de teste *Boy and Toys, Cognac and Fruit, Container, Foreman, Hall Monitor, Mobile and Calendar* e *Tennis* (ver Apêndice A).

3.1 Método do Preditor Tridimensional

O Preditor Tridimensional consiste em uma extensão espaço-temporal do preditor puramente espacial LOCO-I, apresentado na seção 2.5.3. Essa extensão visa realizar a predição de um pixel na imagem utilizando não somente os pixels de sua vizinhança causal, mas também pixels do quadro anterior da sequência [7] [9], já que quadros consecutivos apresentam conteúdos semelhantes na grande maioria dos casos.

Devido à sua concepção, o Preditor Tridimensional adaptativamente escolhe qual a direção mais apropriada (espacial ou temporal) para se extrair a redundância de cada pixel, e dessa forma não necessita de qualquer informação lateral na formação

do *bitstream*.

3.1.1 O Preditor Tridimensional

Extendendo o preditor espacial LOCO-I para um preditor tridimensional, onde consideramos o eixo temporal, chegamos a um preditor capaz de detectar texturas não só espaciais mas também de movimentos em quadros distintos no tempo. Como não há partição da imagem em blocos, os pixels são codificados na ordem tradicional de varredura, de cima para baixo e da direita para a esquerda.

Definindo a vizinhança do pixel atual tanto no quadro em questão quanto no quadro anterior, temos a vizinhança tridimensional da figura 3.1. Os pixels considerados na predição podem ser então mais facilmente visualizados na figura 3.2.

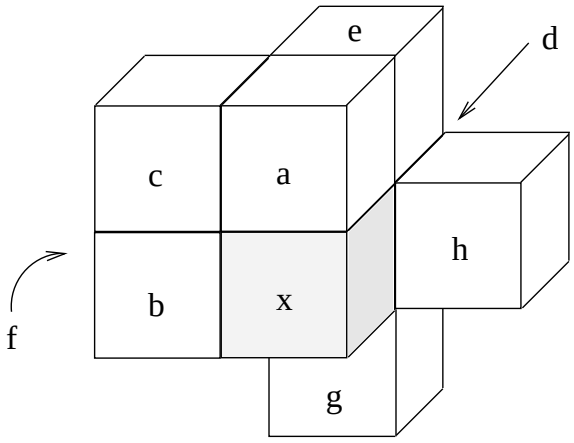


Figura 3.1: Cubo de predição com pixels da vizinhança causal

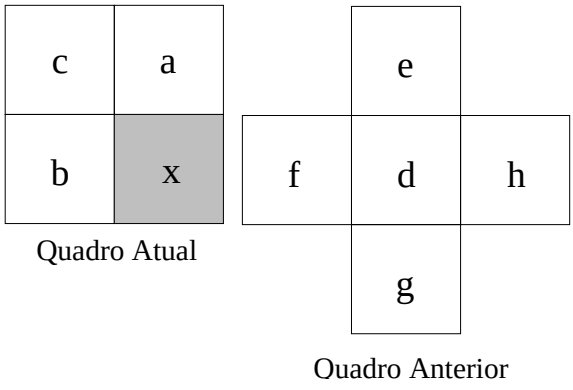


Figura 3.2: Pixels utilizados pelo Preditor Tridimensional

Observando o pixel atual x como um cubo (figura 3.3) tendo suas faces fazendo fronteiras com os pixels que o circundam, ao olharmos para cada uma dessas faces, vemos uma grade de predição LOCO-I diferente (figura 3.4).

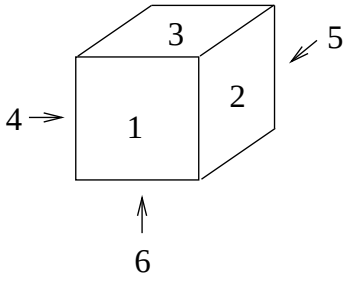


Figura 3.3: Pixel atual: cubo de predição

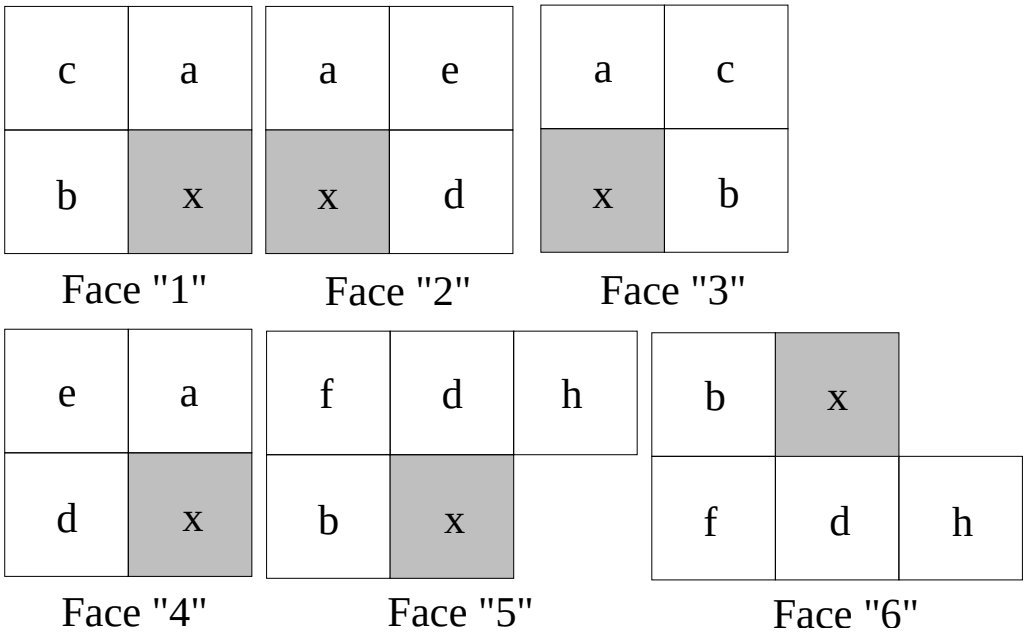


Figura 3.4: As 6 faces de predição

Entretanto, podemos observar que das 6 faces, temos apenas 3 predições distintas, pois a face 1 equivale à face 3, a face 2 equivale à face 4 e a face 5 equivale à face 6. Assim, temos as seguintes grades de predição para o Preditor Tridimensional (figura 3.5):

- Faces 1 e 3 - Predição Horizontal-Vertical (HV)
- Faces 2 e 4 - Predição Vertical-Temporal (VT)

- Faces 5 e 6 - Predição Horizontal-Temporal (HT)

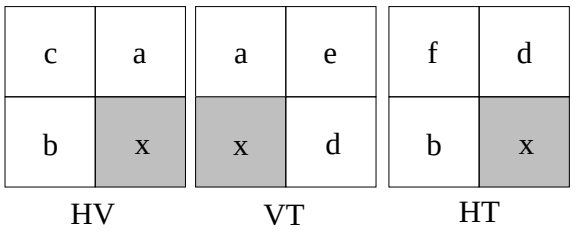


Figura 3.5: Preditores HV, VT e HT

A escolha de qual face será utilizada pelo Preditor Tridimensional é dada pelo menor gradiente. Essa medida de gradiente detecta possíveis arestas na superfície do preditor e é um bom indicativo de qual direção a ser seguida.

Os gradientes d_{hv}, d_{vt} e d_{ht} são definidos pelas equações 3.1, 3.2 e 3.3.

$$d_{hv} = \begin{cases} c - \max(a,b) & \text{se } c \geq \max(a,b) \\ \min(a,b) - c & \text{se } c \leq \min(a,b) \\ -1 & \text{outros casos} \end{cases} \tag{3.1}$$

$$d_{vt} = \begin{cases} e - \max(a,d) & \text{se } e \geq \max(a,d) \\ \min(a,d) - e & \text{se } e \leq \min(a,d) \\ -1 & \text{outros casos} \end{cases} \tag{3.2}$$

$$d_{ht} = \begin{cases} f - \max(b,d) & \text{se } f \geq \max(b,d) \\ \min(b,d) - f & \text{se } f \leq \min(b,d) \\ -1 & \text{outros casos} \end{cases} \tag{3.3}$$

Se pelo menos um dos gradientes acima resultar em um valor maior ou igual a zero, a predição do pixel é feita utilizando o LOCO-I da grade de predição correspondente à direção com o menor gradiente.

Entretanto, se todos os gradientes resultarem em -1, passamos para a utilização de um dos *Preditores Diagonais*, definidos pelas equações 3.4, 3.5 ou 3.6.

$$P_{hv} = a + b - c \tag{3.4}$$

$$P_{vt} = a + d - e \tag{3.5}$$

$$P_{ht} = b + d - f \tag{3.6}$$

A escolha de qual preditor diagonal será utilizado é feita considerando-se a direção com mínimo *Gradiente Diagonal*, que é definido por 3.7, 3.8 e 3.9.

$$\gamma_{hv} = \frac{a+b}{2} - c \tag{3.7}$$

$$\gamma_{vt} = \frac{a+d}{2} - e \tag{3.8}$$

$$\gamma_{ht} = \frac{b+d}{2} - f \tag{3.9}$$

Como a escolha da direção de predição para cada pixel é completamente reversível no decodificador, não há necessidade de transmissão de qualquer tipo de informação lateral.

3.1.2 Codificador

Após realizada a estimação de movimento no quadro anterior, as componentes horizontal e vertical dos vetores de movimento são codificadas no *bitstream*. Para isso utilizamos um codificador aritmético com contextos distintos para ambas as componentes.

A seguir, seguindo a ordem tradicional de varredura, para cada pixel é gerada a predição e o resíduo é codificado no *bitstream*. No caso de imagens coloridas, utilizamos um contexto para cada componente de cor. Nas nossas simulações utilizamos a transformação Y-Co-Cg.

Os contextos do codificador aritmético são descritos na tabela 3.1.

3.1.3 Resultados

A tabela 3.2 mostra os ganhos mínimo, médio e máximo de codificação INTER em relação a INTRA para cada uma das sequências simuladas, onde o ganho mínimo representa o menor ganho de quantização para todos os quadros da sequência. O ganho médio representa a média aritmética dos ganhos e o ganho máximo representa o maior ganho obtido, também para todos os quadros da sequência.

Tabela 3.1: Contextos do codificador aritmético

Contexto	Utilização
MV_X	Componente Horizontal do Vetor de Movimento
MV_Y	Componente Vertical do Vetor de Movimento
PIX_Y	Resíduo de predição do canal Y
PIX_CO	Resíduo de predição do canal Co
PIX_CG	Resíduo de predição do canal Cg

Observamos que o Preditor Tridimensional pode atingir um ganho máximo bastante eficiente, chegando a ordem de 41%. Porém ganhos mínimos negativos indicam que em algumas situações o código INTER é muito menos eficiente do que o INTRA, chegando a ser 22% menos eficiente em um pior caso. O ganho médio mostra que nos casos *Foreman* e *Hall Monitor* a codificação INTRA é mais eficiente do que a INTER para a sequência como um todo, já que esse ganho médio apresenta valor negativo.

Analisando os gráficos, observamos que o comportamento do preditor 3D é bastante sensível ao conteúdo da imagem, com quedas acentuadas e oscilações na taxa de compressão ao longo dos quadros.

A sequência *Container* (Fig. 3.8), que tem conteúdo bastante estático, apresenta bom resultado, com ganhos INTER/INTRA positivos para toda a sequência. Por outro lado, a sequência com captação ruidosa *Hall Monitor* (Fig. 3.10) apresenta ganho INTER/INTRA negativo para todos os quadros, constituindo um caso de ineficiência do método.

3.2 Método do Chaveamento de Preditores

Em uma sequência de vídeo, os quadros apresentam tanto redundância espacial, que é explorada pelos pixels de uma vizinhança, quanto temporal, já que quadros consecutivos tendem a apresentar conteúdo muito semelhante. O método do Chaveamento de Preditores parte do princípio de que em um quadro existem regiões onde a predição espacial seja mais eficiente do que a temporal. Da mesma

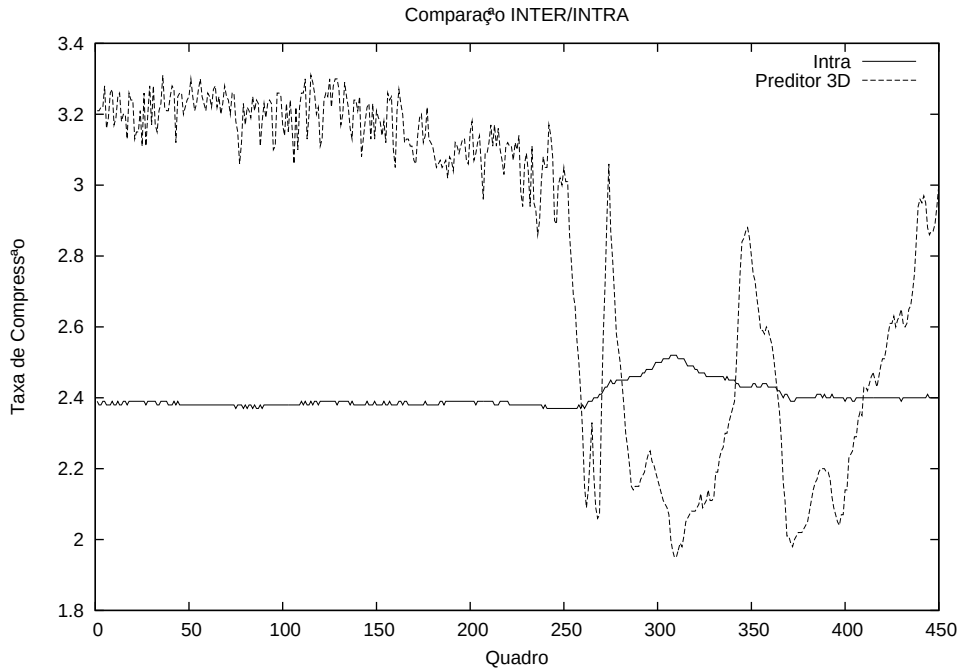


Figura 3.6: *Boy and Toys* - Taxa de Codificação para os modos INTRA e INTER

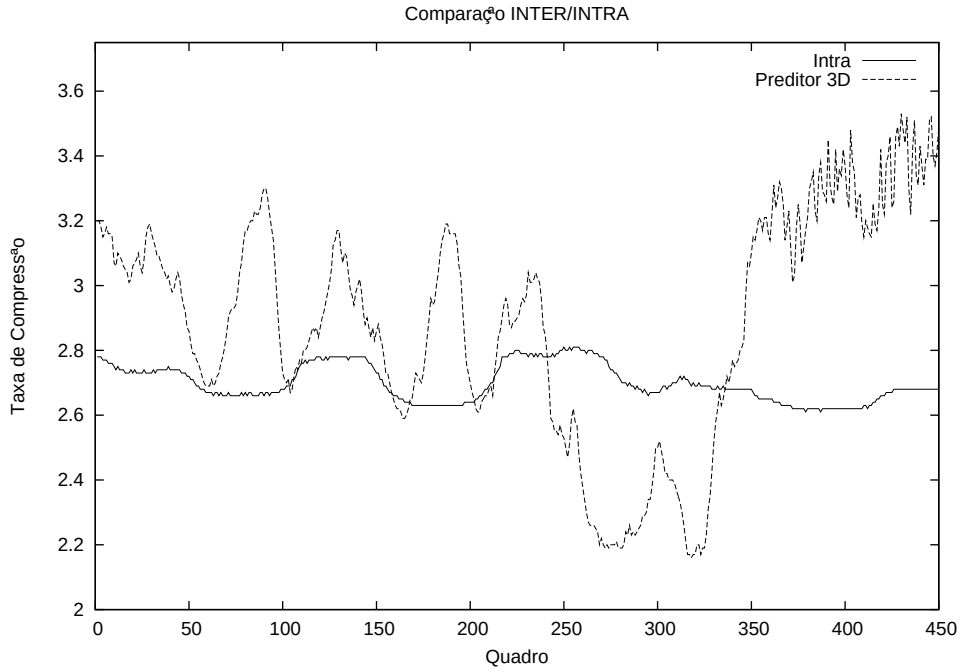


Figura 3.7: *Cognac and Fruit* - Taxa de Codificação para os modos INTRA e INTER

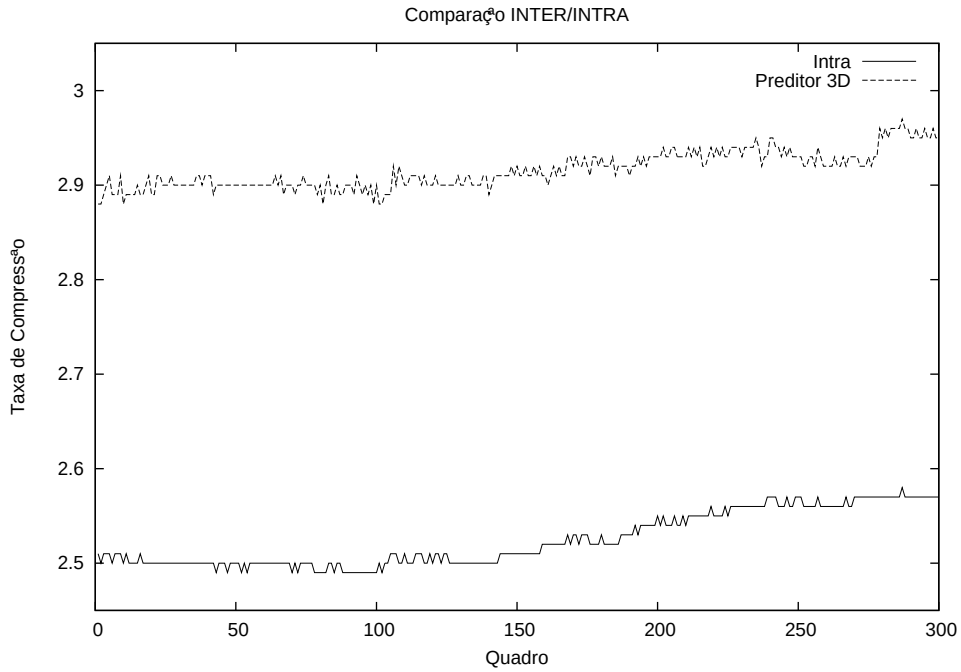


Figura 3.8: *Container* - Taxa de Codificação para os modos INTRA e INTER

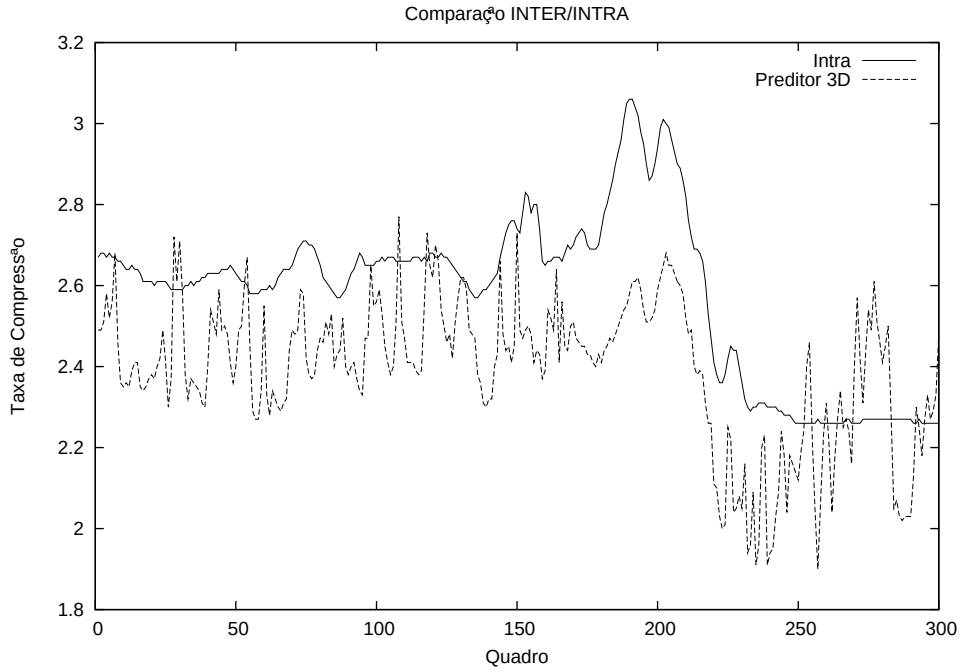


Figura 3.9: *Foreman* - Taxa de Codificação para os modos INTRA e INTER

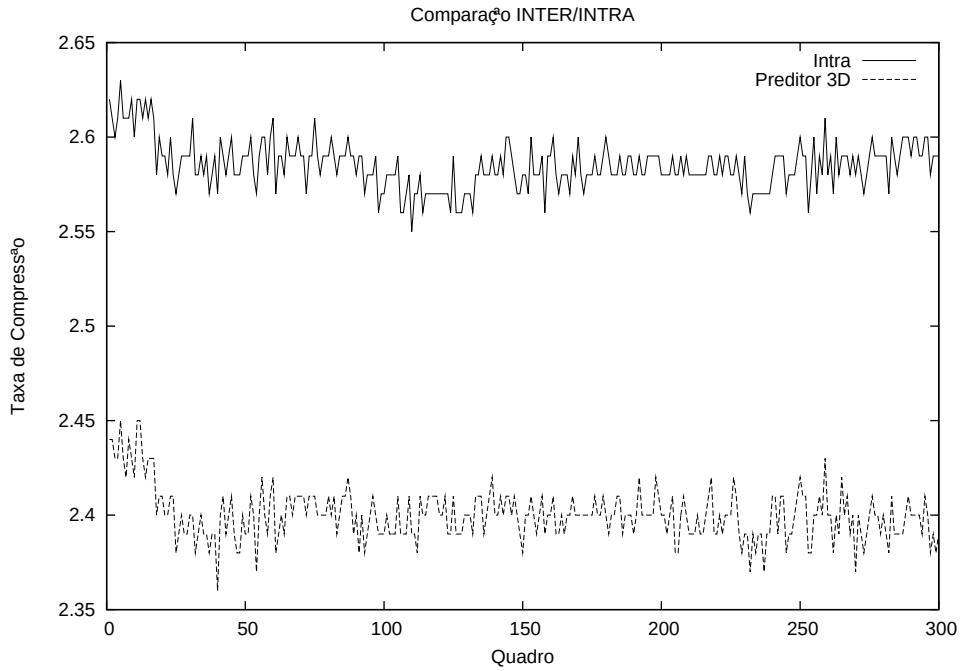


Figura 3.10: *Hall Monitor* - Taxa de Codificação para os modos INTRA e INTER

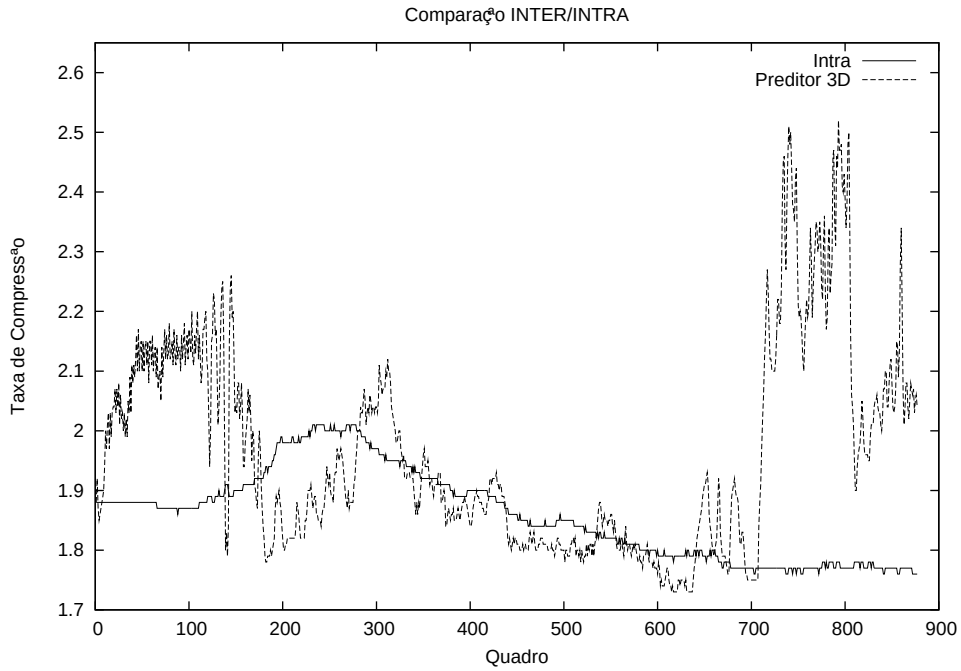


Figura 3.11: *Mobile and Calendar* - Taxa de Codificação para os modos INTRA e INTER

Tabela 3.2: Ganho de codificação INTER/INTRA - Preditor 3D

Sequência	Min (%)	Med (%)	Max (%)
Boy and Toys	-22.62	17.21	38.66
Cognac and Fruit	-20.94	6.99	32.82
Container	13.62	15.50	16.87
Foreman	-16.95	-7.08	14.98
Hall Monitor	-8.17	-7.15	-6.20
Mobile & Calendar	-9.09	5.74	41.81
Tennis	-11.43	6.18	24.26

forma, em outras regiões a predição temporal seria mais eficiente do que a espacial.

Dessa forma, esse método define um conjunto de preditores que são selecionados pelo codificador para representar cada bloco de um quadro de vídeo. Essa informação da escolha do preditor é armazenada no *bitstream* como informação lateral.

A escolha do tamanho do bloco no qual a imagem será segmentada se dá através de um compromisso entre o *overhead* gerado pela informação lateral, e uma predição mais eficiente devido a uma segmentação com maior resolução. Ou seja, blocos muito grandes geram pequeno *overhead*, mas não conseguem uma boa adaptação à textura da image. Por outro lado blocos pequenos que são ideais para uma boa predição geram um *overhead* que pode superar os bits economizados por essa boa predição.

A predição espacial é baseada no preditor LOCO-I, apresentado na seção 2.5.3. Baseado nesse preditor, definimos um preditor LOCO-I modificado que chamamos de LOCO-I Reverso (seção 3.2.1), que visa seguir bordas "não causais"(à direita e abaixo) utilizando-se da imagem anterior. A predição puramente temporal é feita através da cópia do bloco do quadro anterior, porém já com movimento estimado.

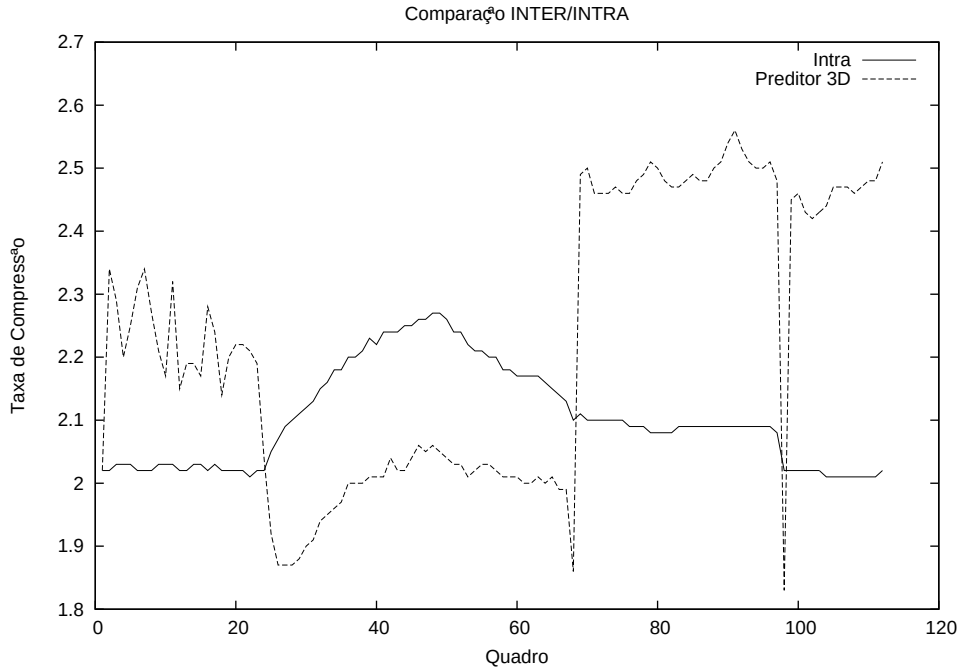


Figura 3.12: *Tennis* - Taxa de Codificação para os modos INTRA e INTER

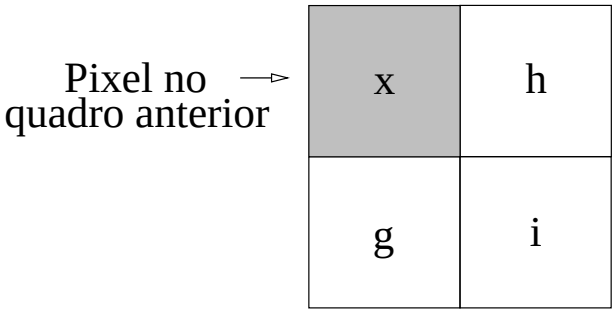


Figura 3.13: Grade de predição do algoritmo LOCO-I Reverso

3.2.1 Preditor LOCO-I Reverso

Como o algoritmo LOCO-I é capaz de detectar bordas seguindo os pixels causais da imagem, se um quadro é semelhante ao anterior podemos utilizar os pixels das posições não-causais do quadro anterior, já que esses pixels são causais para o quadro atual. Assim podemos definir o preditor LOCO-I Reverso, que é capaz de detectar arestas nas direções baixo-cima e direita-esquerda (Fig. 3.13 e Eq. 3.10).

$$\hat{x} = \begin{cases} \min(g,h) & \text{se } i \geq \max(g,h) \\ \max(g,h) & \text{se } i \leq \min(g,h) \\ g + h - i & \text{outros casos} \end{cases} \tag{3.10}$$

3.2.2 Codificador

Após realizar estimação de movimento no quadro anterior, particionamos a imagem em blocos de 4x4 pixels. Para cada um desses blocos o codificador escolhe a predição mais adequada, que acarretaria em uma maior extração das redundâncias da imagem. Nesse trabalho a maior extração de redundâncias foi considerada a que dá o menor valor de SAD (soma das diferenças absolutas). Assim como os vetores de movimento, o identificador do preditor escolhido para cada bloco é incluído no *bitstream* como informação lateral. A escolha do tamanho do bloco se deu através do cálculo do *overhead* aproximado. Nesse cálculo consideramos 2 bits para o armazenamento do tipo de predição para cada bloco e um fator de compressão da ordem de 3:1. A tabela 3.3 mostra os resultados para alguns tamanhos de bloco. O tamanho 4x4 representa uma boa adaptação à textura da imagem com um *overhead* relativamente pequeno quando comparado ao ganho de codificação do método.

Tabela 3.3: *Overhead* para transmissão do tipo de preditor em função do tamanho do bloco

Tamanho do bloco	Overhead (%)
1x1	25
2x2	6.25
4x4	1.56
8x8	0.39
16x16	0.10
32x32	0.02

Assim, definimos os seguintes preditores que podem ser utilizados por cada bloco da imagem:

P1. LOCO-I

P2. LOCO-I Reverso

P3. Bloco do quadro anterior (já com movimento compensado)

P4. $MEDIANA(P1, P2, P3)$

Nas simulações aqui apresentadas, a escolha do preditor se deu através da medida de SAD dos pixels no bloco.

Após a escolha do preditor de um bloco, os resíduos de predição de cada pixel são codificados utilizando um codificador aritmético adaptativo. Esse processo é realizado para todos os pixels do bloco, assim como para todos os blocos da imagem e todos os pixels de cada bloco. A varredura dos pixels e dos blocos seguem a direção convencional (da esquerda para a direite e de cima para baixo). O processo é repetido para as componentes de cor Y, Co e Cg.

Os contextos utilizados pelo codificador aritmético são descritos na tabela 3.4.

Tabela 3.4: Contextos do codificador aritmético

Contexto	Utilização
MV_X	Componente Horizontal do Vetor de Movimento
MV_Y	Componente Vertical do Vetor de Movimento
PREDITOR	Preditor utilizado pelo bloco (mesmo para Y, Co e Cg)
PIX_Y	Resíduo de predição do canal Y
PIX_CO	Resíduo de predição do canal Co
PIX_CG	Resíduo de predição do canal Cg

3.2.3 Resultados

Foram realizadas simulações do codificador com as sequências *Boy and Toys, Cognac and Fruit, Container, Foreman, Hall Monitor, Mobile and Calendar* e *Tennis*. Nessas simulações foram observados o ganho de codificação dos quadros INTER comparado com ganho para o mesmo quadro codificado no modo INTRA. Para o

modo INTER, também foram coletadas informações sobre o percentual de utilização de cada um dos quatro modos de predição, indicando o modo considerado mais eficiente para a maior parte dos blocos de um quadro. Os ganhos mínimo, médio e máximo de codificação do modo INTER em relação ao INTRA são reportados na tabela 3.5.

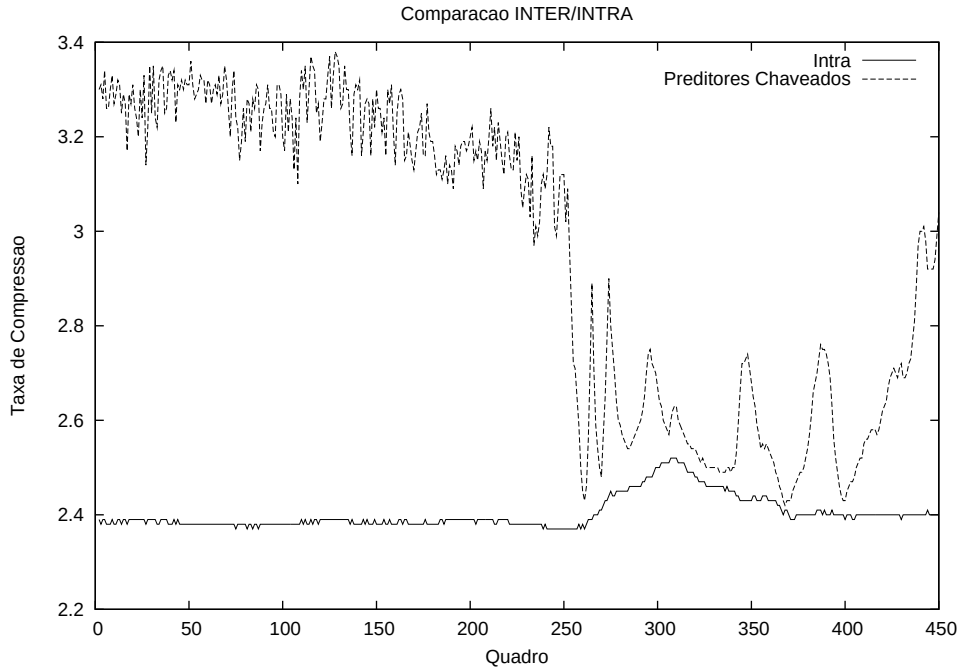
Tabela 3.5: Ganho de codificação INTER/INTRA - Preditores Chaveados

Sequência	Mínimo (%)	Médio (%)	Máximo (%)
Boy and Toys	0.41	23.24	41.42
Cognac and Fruit	1.77	17.66	35.45
Container	21.01	22.41	23.29
Foreman	0.02	5.47	14.09
Hall Monitor	3.08	3.91	4.96
Mobile and Calendar	1.67	9.62	45.76
Tennis	-0.50	13.73	28.36

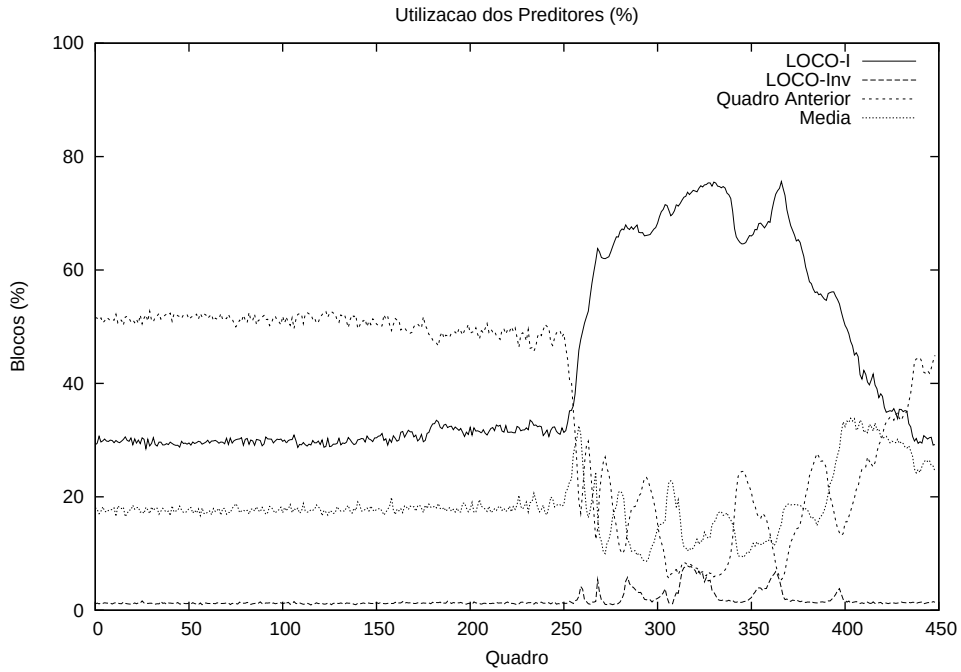
Nas sequências *Boy and Toys* e *Cognac and Fruit*, observamos que nos trechos onde não há movimentação intensa de câmera, mas sim pequenas movimentações de objetos que compõe a imagem, o codificador utiliza bastante a predição temporal (Figuras 3.14 e 3.15). A predição espacial é bastante utilizada nos trechos onde há intensa movimentação de câmera.

As sequências *Container* e *Hall Monitor* (Figuras 3.16 e 3.17), apesar de serem captadas por uma câmera estática, onde não há movimentação do plano de fundo da cena, há movimentação de pequenos objetos na cena. *Container* contém variações complexas na textura da água, mas o codificador ainda assim alcançou uma boa eficiência (da ordem de 20%) de codificação INTER. Entretanto, a sequência *Hall Monitor* não apresentou um bom ganho de codificação INTER (da ordem de 4%), provavelmente ao intenso ruído de captação, que prejudica a codificação temporal lossless.

A sequência *Foreman* (Figura 3.18) apresentou ganhos variáveis de codificação, desde 0% até 14% , e a falta de estabilidade da câmera causou deficiência nas predições.

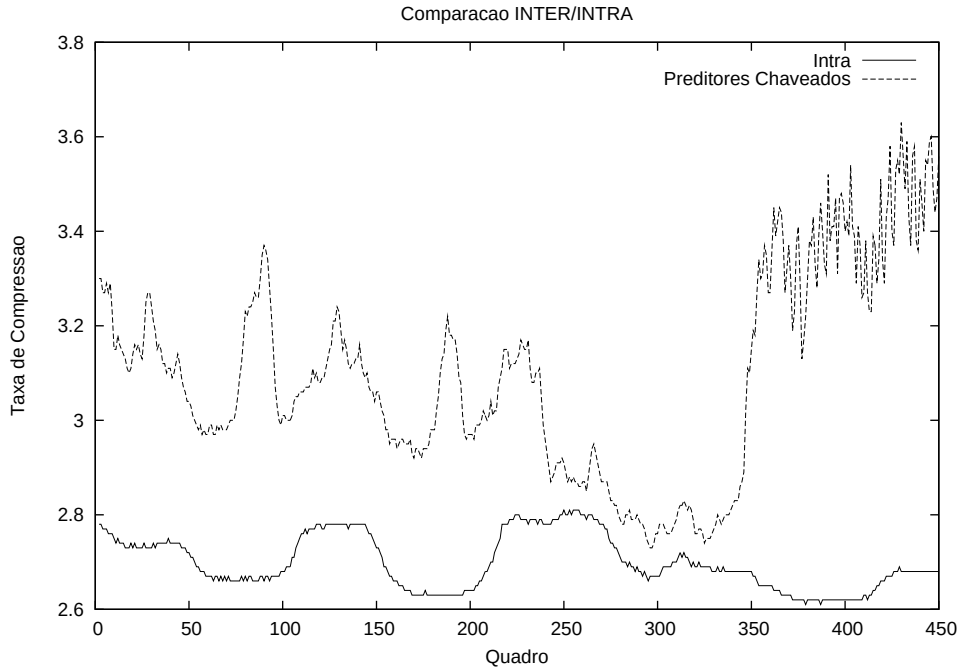


(a)

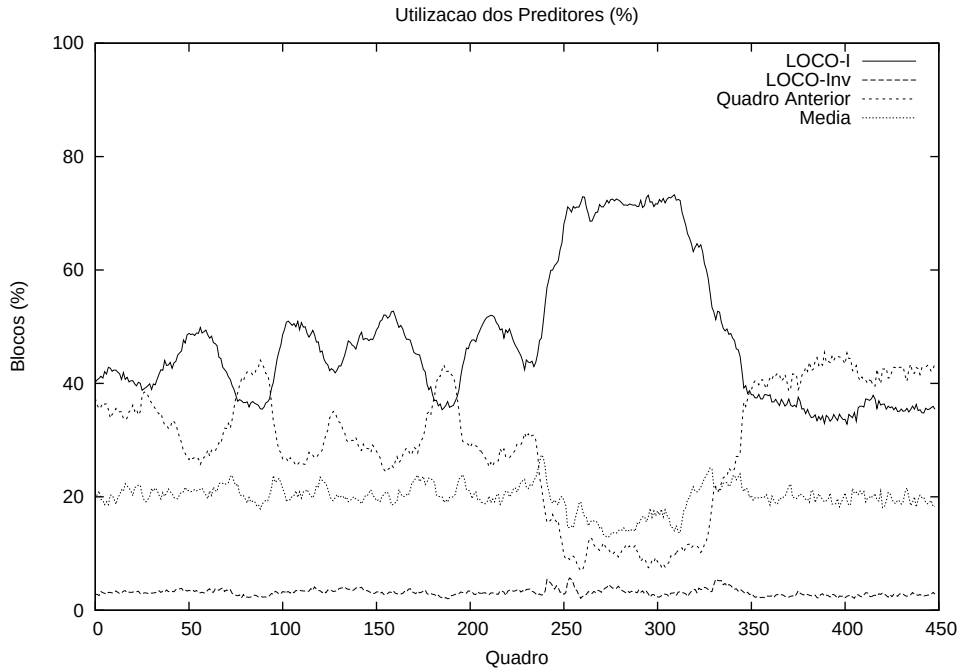


(b)

Figura 3.14: *Boy and Toys* - (a) Taxa de Codificação para os modos INTRA e INTER e (b) Percentual de utilização de cada um dos 4 preditores nos blocos

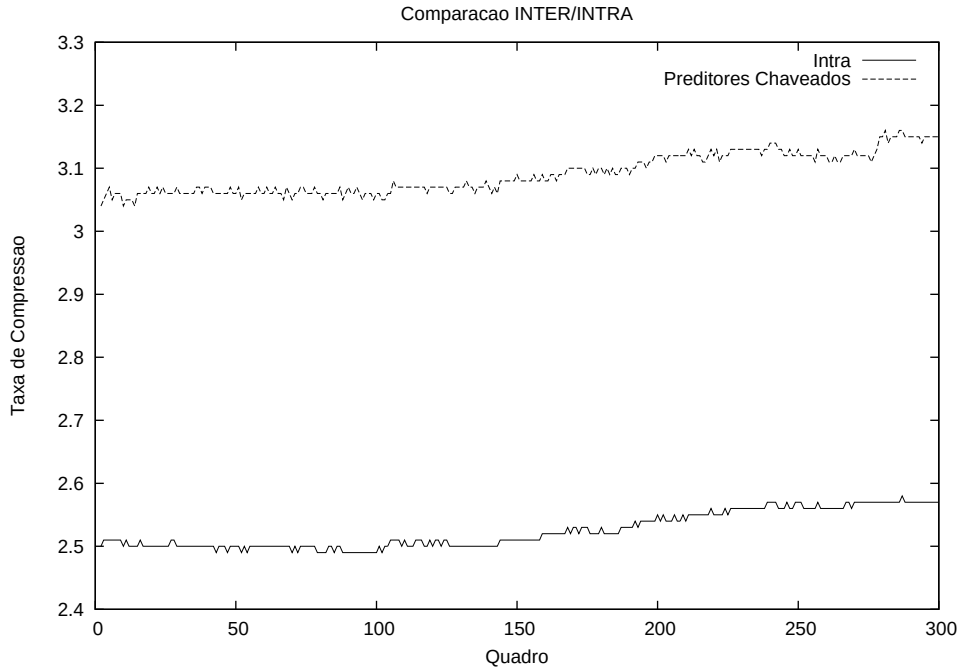


(a)

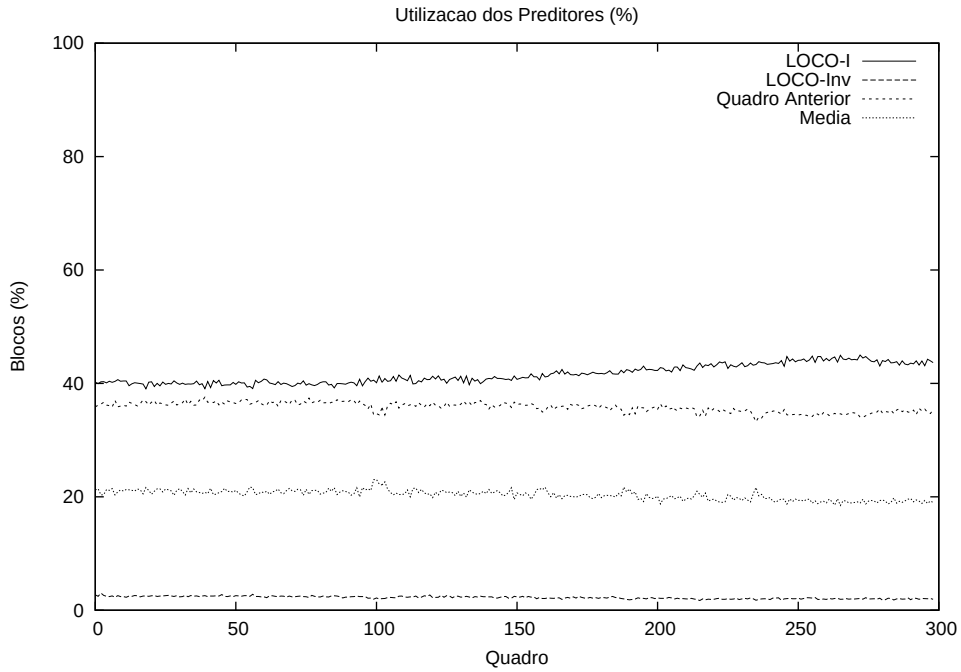


(b)

Figura 3.15: *Cognac and Fruit* - (a) Taxa de Codificação para os modos INTRA e INTER e (b) Percentual de utilização de cada um dos 4 preditores nos blocos

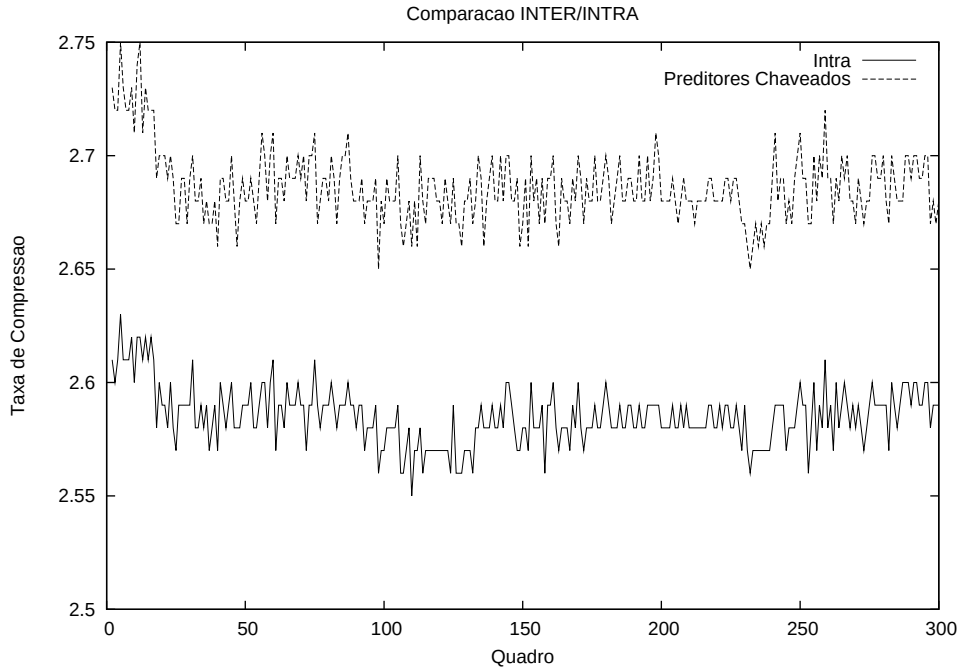


(a)

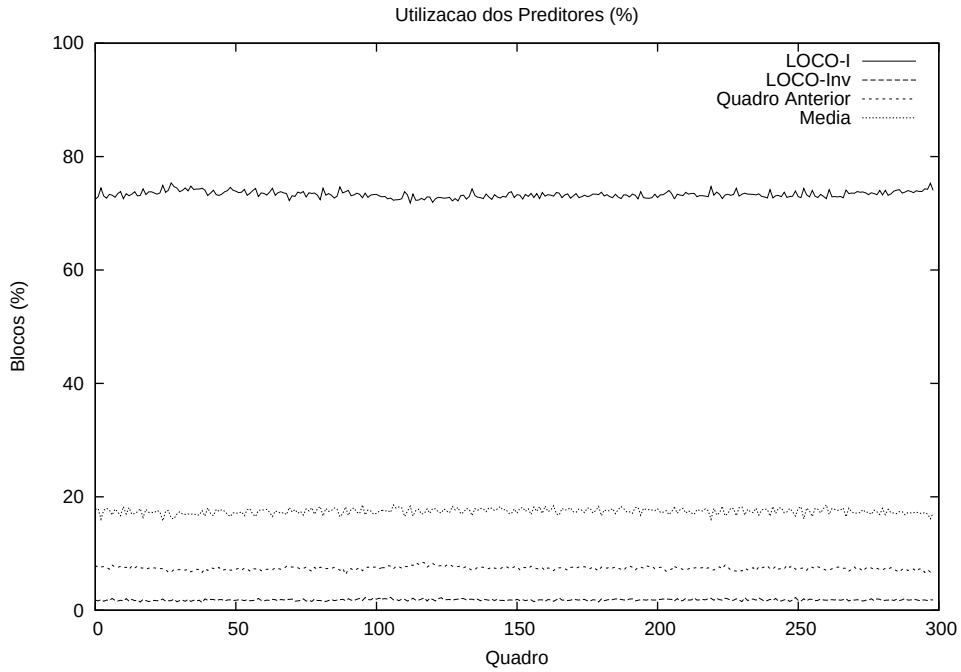


(b)

Figura 3.16: *Container* - (a) Taxa de Codificação para os modos INTRA e INTER e (b) Percentual de utilização de cada um dos 4 preditores nos blocos

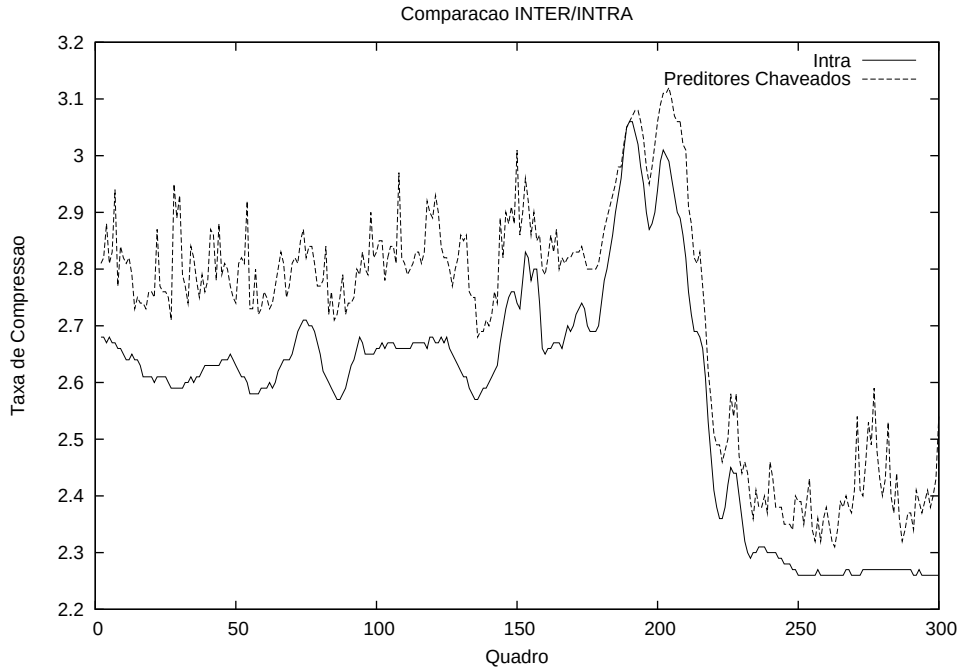


(a)

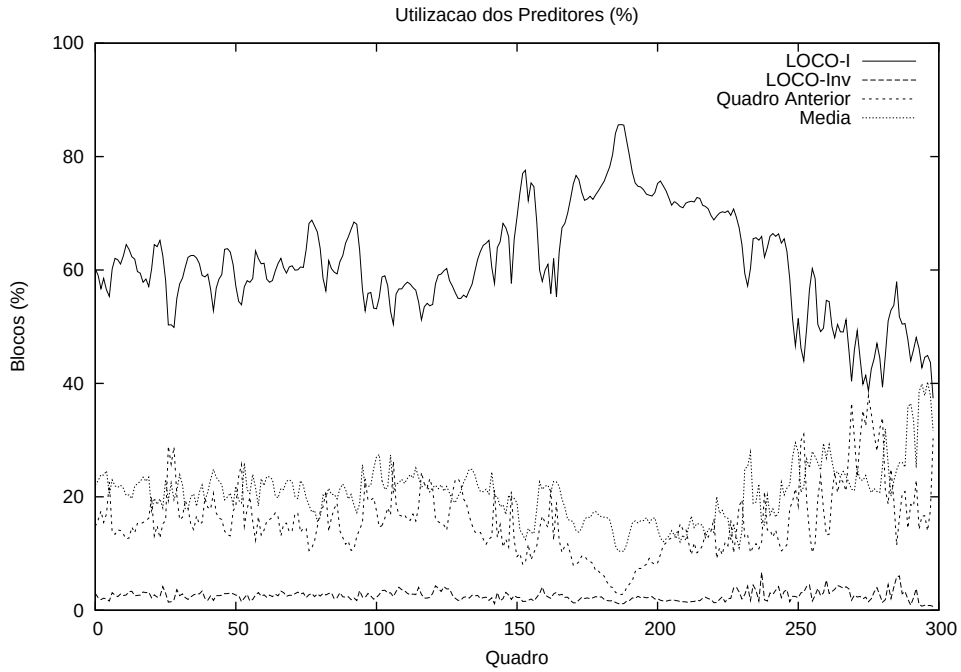


(b)

Figura 3.17: *Hall Monitor* - (a) Taxa de Codificação para os modos INTRA e INTER e (b) Percentual de utilização de cada um dos 4 preditores nos blocos



(a)



(b)

Figura 3.18: *Foreman* - (a) Taxa de Codificação para os modos INTRA e INTER e (b) Percentual de utilização de cada um dos 4 preditores nos blocos

Mobile and Calendar (Figura 3.19), que é caracterizada por intensa movimentação conseguiu obter ganhos de até 45% em trechos de maior estabilidade no vídeo. Mesmo nas partes mais complexas ganhos razoáveis de codificação são observados, ganhos estes impulsionados pela estimação de movimento realizada no vídeo.

Na sequência *Tennis* (Figura 3.20) nota-se claramente a existência de dois cortes de cena, nos quais a codificação INTER não é apropriada e resulta em um ganho de codificação negativo (taxa INTER < taxa INTRA). Entretanto, são observados consideráveis ganhos de codificação (chegando até 28%) nos trechos mais estáticos do vídeo.

De forma geral, os resultados comprovam que o método tem boa eficiência, mostrando ganhos de codificação mesmo em regiões de intensa movimentação no vídeo. Apenas nos cortes de cena da sequência *Tennis* a taxa INTER ficou inferior à taxa INTRA, porém com uma diferença muito pequena (-0.5%).

3.3 Análise comparativa entre o Método do Chaveamento de Preditores e o Método do Preditor Tridimensional

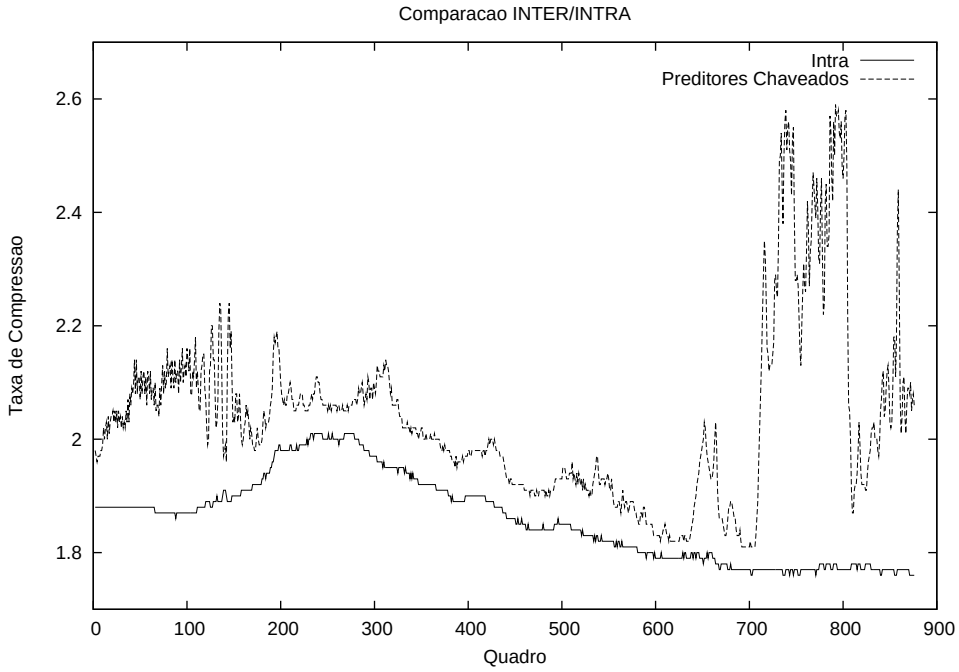
Nessa seção, comparamos o peso da transmissão da informação lateral necessária pelo método dos Preditores Chaveados é com o método do Preditor 3D, que não utiliza qualquer informação lateral. A comparação é feita tomando-se um quadro INTRA como referência.

Considerando-se uma imagem com dimensões W x H, um quadro INTER do método dos Preditores Chaveados teria como informação lateral:

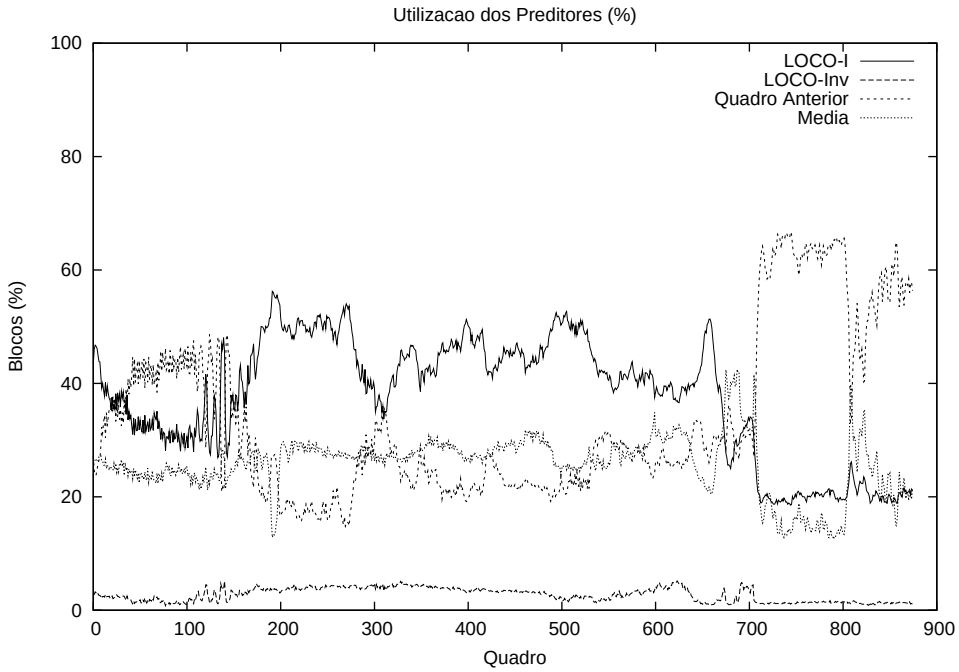
- Vetores de Movimento: 10 bits para cada um dos $\frac{W}{16} \times \frac{H}{16}$ macrobolocos.
- Tipo de Preditor: 2 bits para cada um dos $\frac{W}{4} \times \frac{H}{4}$ blocos

Os valores acima não consideram a compressão do codificador por entropia. Estamos tratando de um caso extremo para efeito comparativo.

Teríamos então o acréscimo de $10 \times \frac{W \times H}{256} + 2 \times \frac{W \times H}{16} = 42 \times \frac{W \times H}{256}$ bits de informação lateral. Tendo a imagem original 24 bits por pixel e os codificadores com

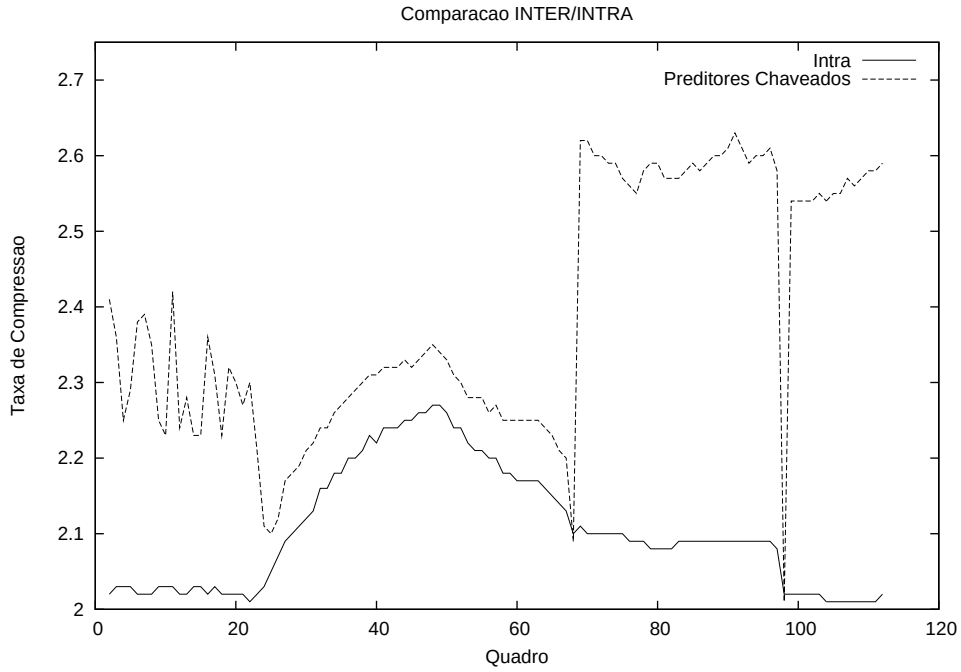


(a)

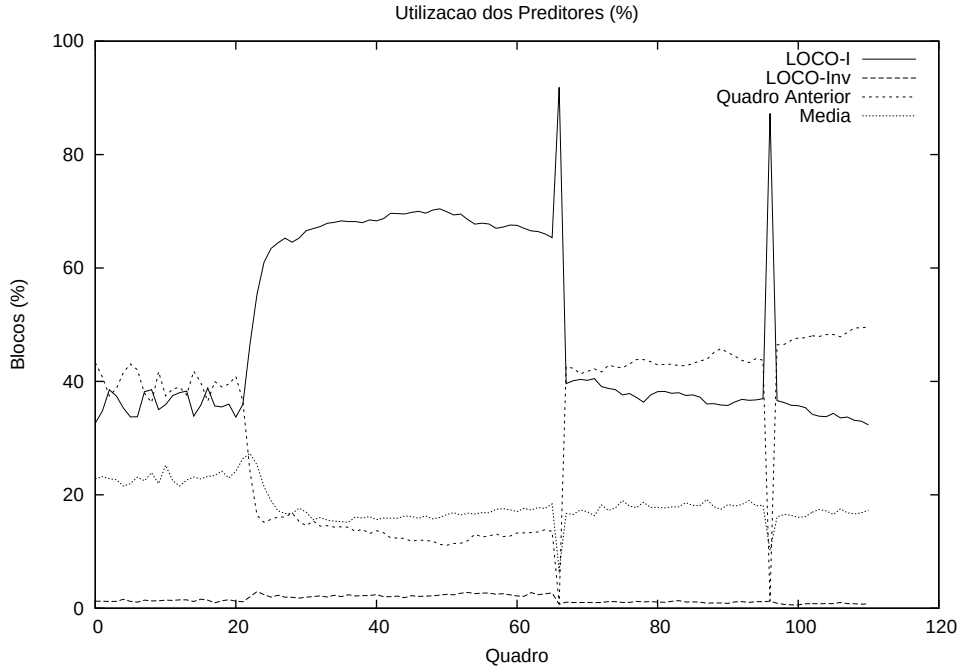


(b)

Figura 3.19: *Mobile and Calendar* - (a) Taxa de Codificação para os modos INTRA e INTER e (b) Percentual de utilização de cada um dos 4 preditores nos blocos



(a)



(b)

Figura 3.20: *Tennis* - (a) Taxa de Codificação para os modos INTRA e INTER e (b) Percentual de utilização de cada um dos 4 preditores nos blocos

desempenho da ordem de 3:1, cada quadro teria um *bitstream* com 8 bits por pixel, ou $8 \times W \times H$ bits para a imagem inteira.

Assim, os Preditores Chaveados teriam uma sobrecarga de $\frac{42 \times \frac{W \times H}{256}}{8 \times W \times H}$, ou 2.05% de informação lateral. Esse resultado significa que no pior caso de comparação INTER/INTRA, onde para todos os blocos escolhemos a predição LOCO-I que é a predição de um quadro INTRA, o código seria apenas 2.05% menos eficiente do que o modo INTRA. Porém, como o codificador pode escolher outros modos para os blocos e em alguns casos outros preditores podem ser mais eficientes do que o LOCO-I, esse fator de 2% é facilmente compensado e vemos nos gráficos que dificilmente o quadro INTER tem desempenho pior do que o INTRA.

Já para o Preditor 3D não temos nada que garanta os limiares inferiores, pois o preditor pode assumir qualquer comportamento. Assim vemos várias situações em que o desempenho INTER é muito pior do que o INTRA.

Capítulo 4

Codificação de Imagens Estéreo utilizando o H.264

4.1 Método

Um par estéreo de imagens tem como característica a diferença de ângulo de captação de uma imagem para outra. Esses efeitos não conseguem ser bem modelados por sistemas baseados em deslocamento de macroblocos na estimação de movimento, que é capaz de compensar somente translação de objetos no plano 2D. As distorções em perspectiva, bastante comuns em pares estéreo, são difíceis de serem tratadas por estimadores de movimento projetados para trabalhar com sequências monoculares.

Podemos codificar o par estéreo considerando as duas vistas como images estáticas, onde a imagem direita não utiliza a esquerda para eliminação de redundâncias. Outra opção é considerá-las como dois quadros consecutivos de uma seqüência de vídeo. Como já foi dito, os efeitos de deslocamentos angulares não conseguem ser bem modelados, mas é esperado um bom desempenho em regiões da imagem que podem parecer estáticas de uma imagem para outra. Como o H.264 usa métodos taxa-distorção para definir se um macrobloco vai ser INTRA ou INTER, ele terá a tendência de escolher com eficiência em quais regiões irá valer ou não a pena utilizar a estimação de movimento (mais rigorosamente, no caso de imagens estéreo, estimação de disparidade). Este último aspecto fica bem claro quando analisamos os resultados obtidos para os diferentes pares estéreo. Além disso, a estimação de

movimento (no caso, disparidade) usando blocos de tamanho variável faz com que nos casos de macroblocos INTER a precisão na estimação seja bem maior, o que tenderá a fornecer um aumento no desempenho.

Quando se deseja codificar uma seqüência visando atingir uma determinada taxa de bits por pixel, devemos realizar um controle de banda no codificador. Entretanto, os algoritmos de controle de banda aplicados em vídeo atuam com uma resposta mais lenta, objetivando atingir a banda configurada ao longo do tempo em uma média de vários quadros. Esse tipo de algoritmo de controle de banda não funciona para o caso do par estéreo, pois a taxa desejada deve ser atingida em apenas dois quadros, correspondendo às imagens direita e esquerda que compõem o par estéreo. Assim, neste artigo, realizamos o controle manualmente, acertando o valor ideal para o passo de quantização em cada um dos casos para uma dada taxa e mantendo a relação sinal ruído. Com isto, tivemos o objetivo de verificar qual o melhor desempenho possível de se obter usando o padrão H.264 para a codificação de pares estéreo. No caso da codificação dos dois quadros como INTRA, o passo de quantização é ajustado para atingir metade da taxa desejada em cada um dos dois quadros. Como as duas imagens que compõem o par estéreo são semelhantes, é esperado para um mesmo passo de quantização a mesma taxa de bits seja atingida. Porém, para o caso de uma imagem INTRA e a outra INTER, a taxa de codificação para a imagem INTER é muito menor do que a taxa para a imagem INTRA. Assim, podemos reduzir o passo de quantização do quadro INTRA, concedendo a ele maior qualidade, e ajustando o passo de quantização do quadro INTER para que o valor da relação sinal-ruído seja semelhante nos dois quadros. O software de referência do H.264 permite ajustes do passo de quantização quadro a quadro, e assim não é possível atingir uma taxa exata de codificação, mas sim um valor aproximado. Para um ajuste fino na taxa de codificação seria necessária a implementação de um ajuste automático no passo de quantização macrobloco a macrobloco, recurso que é previsto na norma mas inexistente no software de referência.

4.2 Resultados

Nesta seção apresentamos os resultados obtidos codificando-se os pares estéreo como duas imagens estáticas independentes e como uma sequência composta somente por dois quadros, com o primeiro quadro sendo uma das vistas (e.g. esquerda) e a outra vista sendo o segundo quadro da sequência.

Utilizamos os pares estéreo FRUIT [1] e CORRIDOR [2] (também conhecido como ROOM), e os primeiros quadros das sequências estéreo naturais [3] AQUA, MAN, SAXO e TRAIN. O par estéreo FRUIT (256x256) é uma cena natural com pequenas e médias regiões de oclusão. O par sintético CORRIDOR (256x256) apresenta pequenas diferenças de disparidade e possui áreas de baixa textura. Os quadros da sequência natural AQUA (352x288), com muitas áreas de textura, apresentam áreas de oclusão moderada. O par SAXO (720x576) possui áreas homogêneas dentro de alguns objetos, e áreas de textura como a camisa do saxofonista e apresenta regiões de média oclusão. Em TRAIN (720x576) tem-se algumas áreas de oclusão moderada e regiões de baixa, média e alta textura. O par MAN (384x384) é uma cena típica de videconferência (Figura A.2), com muitas áreas de baixíssima textura e com grandes áreas de oclusão (partes do rosto, como as orelhas, o nariz, o cabelo e laterais da face, são vistas por uma vista e ficam oclusas para a outra vista e vice-versa). Este par em particular é um excelente exemplo para se ilustrar a distorção perspectiva.

Para a realização dos testes utilizamos o modelo teste JM-7.6 [20]. O conjunto de pares estéreo abrange vários níveis de oclusão e diferentes escalas de textura. Somente o componente de luminância foi usado para o cálculo do PSNR. Nas simulações com o H.264, diferentes valores de parâmetros de quantização, variando de 13 a 45, foram utilizados (Tabela 4.1). É importante ressaltar que como não estamos codificando sequências, não foi possível estabelecer-se taxas de bits. O objetivo deste trabalho é avaliar os resultados do codificador H.264 [21] quando aplicado à imagens

[1] http://vasc.ri.cmu.edu/idb/html/stereo/

[2] Computer Vision and Pattern Recognition Group, University of Bonn.

[3] CCETT: Centre Commun d'Etudes de Télédiffusion et Télécommunications (sequências de teste captadas e distribuídas pelos Projetos Europeus RACE-DISTIMA e ACTS-PANORAMA) , França.

estéreo. Na verdade, quando codificamos as vistas separadamente, isto equivale a usar o H.264 somente no modo INTRA (Tabela 4.1). No modo INTRA os campos PSNR1 e PSNR2 correspondem aos valores obtidos para as vistas da esquerda e da direita respectivamente. E, quando assumimos que as duas vistas compõem uma sequência estéreo de dois quadros, o codificador chaveia do modo INTRA/INTER procurando o melhor compromisso entre taxa de compressão e distorção.

Observando-se os resultados na Tabela 4.1, nota-se que a PSNR das duas vistas consideradas em conjunto é quase sempre superior quando o codificador está operando no modo INTRA/INTER, indicando que o preditor utilizado no H.264 é efetivo ao explorar a redundância existente entre as duas vistas de cada par. A exceção ocorre com o par MAN, para o qual o desempenho global no modo INTRA foi ligeiramente superior em algumas taxas (0.645 dB na PSNR2 o que euivale a cerca de 0.33 dB na PSNR total, na taxa de 0.8 bits/pixel) ao do modo INTRA/INTER. Isso pode ser explicado pela grande diferença de disparidades entre pontos homólogos nas duas vistas deste par em particular, o que dificulta o processo de predição do H.264 que não utiliza nenhum tipo de compensação explícita para a distorção geométrica.

As figuras 4.1 e 4.2 ilustram os desempenhos do H.264 e de alguns outros codificadores da literatura (HQBM [16], Frajka [18] e stereo-MMP [19]) para as imagens AQUA e CORRIDOR. O desempenho do H.264 é superior ao dos outros codificadores exceto para o par AQUA na regiao abaixo de 0.5 bits/pixel onde o codificador descrito em [18] é superior. Nestes gráficos, o PSNR é obtido a partir do erro médio quadrático das duas vistas em conjunto.

Assim, conclui-se que o codificador H.264 apresentou boa eficiência na codificação de pares estéreo, o que mostra que o processo de estimação de movimento baseado em múltiplas partições de blocos juntamente com as ferramentas de codificação do H.264 conseguiram modelar bem as disparidades entre as duas vistas.

		INTRA				INTRA/INTER			
par	taxa	QP1	PSNR1	QP2	PSNR2	QP1	PSNR1	QP2	PSNR2
AQUA	0.2	45	22.748	45	22.195	42	24.267	42	22.887
AQUA	0.6	38	26.645	38	26.329	35	28.663	34	27.706
AQUA	1.0	34	29.524	34	29.307	33	30.268	30	30.835
AQUA	1.6	30	32.710	30	32.592	28	34.643	27	33.560
AQUA	2.0	28	34.643	28	34.586	26	36.510	24	36.165
AQUA	2.4	25	37.675	25	37.606	24	38.482	22	38.207
CORRIDOR	0.2	42	28.774	42	28.824	37	33.073	34	33.882
CORRIDOR	0.4	33	36.153	33	36.274	28	40.062	26	40.622
CORRIDOR	0.6	28	40.062	28	40.118	23	43.713	21	44.658
CORRIDOR	0.8	23	43.713	23	43.705	19	46.529	16	48.263
CORRIDOR	1.0	20	45.634	20	46.050	15	50.124	13	50.469
CORRIDOR	1.2	17	48.080	17	48.233	13	51.734	10	52.532
FRUIT	0.2	36	31.669	36	32.111	35	32.261	34	32.783
FRUIT	0.6	28	36.363	28	37.044	26	37.597	26	37.410
FRUIT	1.0	24	39.017	24	39.673	23	39.730	22	39.932
FRUIT	1.6	20	42.273	20	42.642	19	43.251	18	42.856
FRUIT	2.0	17	45.159	17	45.252	17	45.159	15	45.602
FRUIT	2.4	15	47.166	15	47.257	15	47.166	13	47.330
MAN	0.1	30	42.054	30	41.924	30	42.054	29	41.694
MAN	0.2	23	44.396	23	44.325	23	44.396	23	43.820
MAN	0.4	19	45.751	19	45.774	19	45.751	17	46.197
MAN	0.6	16	47.398	16	47.395	16	47.398	15	47.465
MAN	0.8	14	48.740	14	48.722	14	48.740	14	48.077
SAXO	0.2	39	31.019	39	31.089	36	32.910	35	32.591
SAXO	0.6	28	38.445	28	38.512	27	39.064	25	39.354
SAXO	0.8	25	40.609	25	40.640	24	41.184	22	41.290
SAXO	1.0	22	42.613	22	42.565	21	43.306	20	42.606
SAXO	1.4	19	44.813	19	44.709	18	45.534	16	45.740
SAXO	1.8	16	47.260	16	47.179	14	48.862	14	47.378
TRAIN	0.2	35	34.434	35	34.365	33	35.647	31	35.722
TRAIN	0.6	25	40.664	25	40.530	24	41.150	22	41.183
TRAIN	0.8	22	42.509	22	42.348	21	43.130	20	42.450
TRAIN	1.0	20	43.828	20	43.695	20	43.828	18	43.962
TRAIN	1.4	17	46.162	17	46.062	17	46.162	15	46.552
TRAIN	1.8	14	48.779	14	48.717	14	48.779	12	48.836

Tabela 4.1: PSNR de codificação dos modos INTRA e INTRA/INTER em diversas taxas

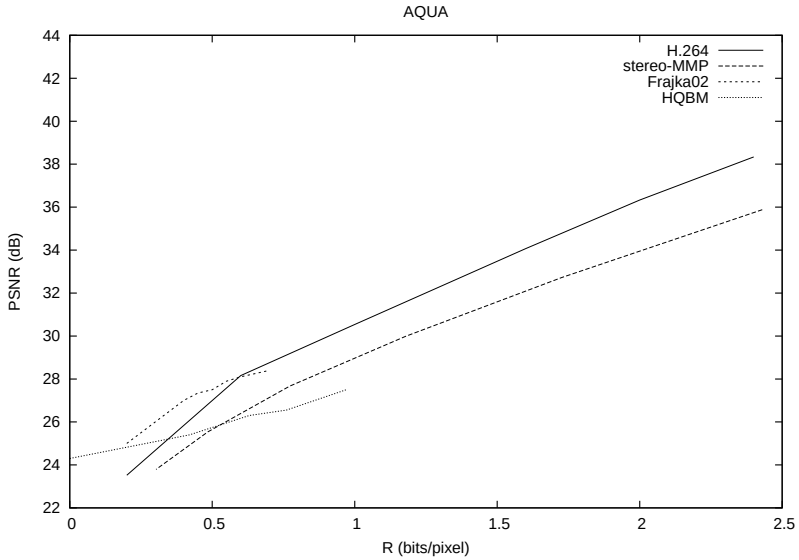


Figura 4.1: Desempenho para AQUA

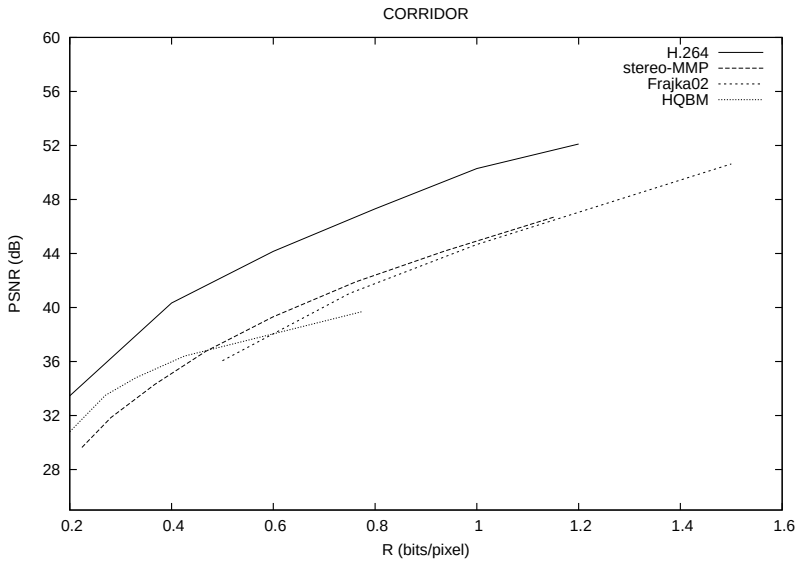


Figura 4.2: Desempenho para CORRIDOR

Capítulo 5

Algoritmo Anti-Flickering para o H.264

5.1 Introdução

O codificador H.264 é capaz de alcançar taxas elevadas de codificação de quadros INTRA em relação a padrões de compressão para imagens estáticas, como o JPEG, MPEG INTRA ou mesmo JPEG2000 [24]. Isso se deve ao método de predição espacial empregado no codificador, que reduz entropia do sinal baseando-se nos pixels da vizinhança causal.

A codificação de vídeo utilizando somente quadros INTRA tem importante utilização em aplicações onde a decodificação de um quadro deve ser independente de outros quadros. Nesse cenário se incluem aplicações de edição não-linear de vídeo, acelerando o acesso randômico aos quadros ou implementação de codificadores e decodificadores com reduzida complexidade, já que não seriam empregados métodos de estimação de movimento e quadros anteriores não seriam armazenados como estado do codificador e decodificador.

Entretanto, experimentos realizados com o codificador H.264 mostraram que sequências codificadas somente com quadros INTRA apresentam excessivo efeito de *flickering* [23]. Esse efeito é observado em análises perceptuais do vídeo, já que cada quadro individualmente é codificado sob otimização da relação sinal-ruído em vista da taxa de codificação. Como é um artefato temporal, causado por diferenças de nível que se oscilam entre dois quadros consecutivos, esse efeito não é refletido nas

medidas de PSNR reportadas pelo codificador.

O efeito de *flickering* consiste de oscilações de grande amplitude no nível de luminância do sinal entre os quadros de vídeo. Esse efeito é visível nas regiões estáticas da imagem, como cenário de fundo e objetos que não se movem e portanto o nível do sinal nessas regiões deveria permanecer constante. Os dispositivos de captação de imagem, principalmente os sensores de baixa precisão, podem apresentar pequenas variações no nível de um pixel em quadros consecutivos. Até os sensores mais precisos apresentam tais variações ao longo do tempo, mas por serem de baixas amplitudes, essas oscilações não são detectadas pelo sistema visual humano. Entretanto, o efeito do processo de quantização do codificador pode transformar essa oscilação de pequena amplitude em uma oscilação de amplitude perceptível. A figura 5.1 mostra no sinal (a) uma pequena oscilação em torno do *threshold* de quantização. Ao ser quantizada, essa oscilação de pequena amplitude passa a possuir em (b) a amplitude correspondente ao passo de quantização. Isso acontece porque o *threshold* é o ponto de decisão entre dois níveis quantizados.

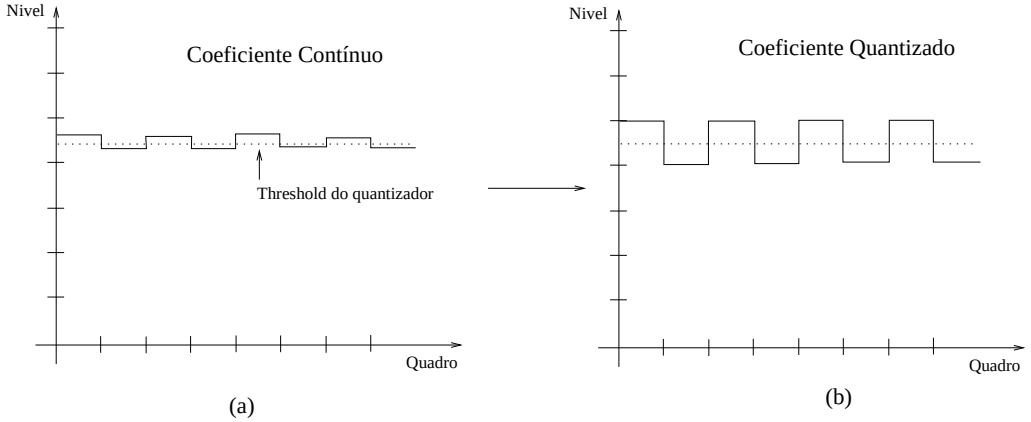


Figura 5.1: Quantizador acentuando efeito de *flickering*. Na figura (a) temos uma oscilação de baixa amplitude, causada por imprecisões do dispositivo de captação da imagem. Na figura (b) vemos a mesma oscilação, porém com sua amplitude acentuada pelo passo de quantização

Esse efeito foi também observado por um grupo de pesquisadores e uma solução foi proposta por eles ao grupo JVT, responsável pela definição do padrão H.264 [23]. Essa proposta incluía também um método para medição da quantidade de existente entre dois quadros, e essa medida será utilizada nas análises aqui realizadas.

Entretanto, o algoritmo proposto por tal contribuição, apesar de causar uma boa redução no *flickering*, não era suficiente para sua eliminação. Esse capítulo apresenta um algorítmo que, associado ao método contido na contribuição JVT-E070, é capaz de eliminar quase que por completo esse artefato de codificação.

Primeiramente será apresentado o método proposto na contribuição JVT-E070, bem como a medida de *flickering* que será utilizada como parâmetro de comparação. Em seguida será exposto o algorítmo desenvolvido para codificação *Anti-Flickering*, bom como seus resultados.

5.2 Contribuição JVT-E070

A sensação de *flickering* é causada por oscilações em regiões da imagem ao longo do tempo. Essas oscilações são mais perceptíveis em partes originalmente estáticas da imagem, ou seja, onde não há movimentação de objetos.

Foi introduzida a medida de *flickering S*. Essa medida é basicamente uma média das diferenças dos valores dos pixels de um quadro para o quadro anterior, calculada apenas nos blocos onde não há intensa alteração de conteúdo em relação ao quadro original. Seja O(i,j) o macrobloco j do quadro i original, e R(i,j) o macrobloco correspondente do quadro a ser analisado. A operação SSD representa a soma dos quadrados das diferenças (*Sum of Squared Differences*) dentro de um bloco.

$$S = \underset{SSD(O(i,j)-O(i+1,j))<\varepsilon}{\text{AVG}}(SSD(R(i+i,j)-R(i,j), O(i+1,j)-O(i,j))) \quad (5.1)$$

A subtração $O(i+1,j) - O(i,j)$ denota a diferença de um macrobloco entre dois quadros consecutivos do vídeo original. Caso essa diferença seja menor do que um valor ε, há pequena variação no vídeo e a imagem reconstruída está sujeita ao efeito de *flickering*. A medida S então se constitui da média quadrática das diferenças entre a variação do vídeo na sequência original e na sequência a ser medida.

O método de codificação INTRA do H.264 é feito através da codificação dos resíduos da imagem original em relação a uma predição que pode ser escolhida entre 8 modos disponíveis [21] [22]. Esses modos são formados por repetições de pixels

adjacentes na vizinhança causal ou mesmo por uma combinação linear entre esses pixels, gerando assim padrões diagonais e planos. Esse resíduo passa então pelo processo de transformação e quantização.

Seja T o processo de transformação, T^{-1} a transformada inversa, Q o processo de quantização e Q^{-1} a quantização inversa. Seja $W(O) = T^{-1}Q^{-1}QT(O)$, então caso não existisse o processo de predição, o sinal reconstruído R seria dado por $R(O) = W(O)$. Porém, considerando-se que o sinal original é subtraído da predição, o sinal reconstruído é $R(O) = W(O - P) + P$.

A utilização da predição acarreta na subtração de um valor com faixa contínua da imagem original. Como esse valor é calculado por expressões lineares e não é quantizado, a probabilidade de que a predição apresente um mesmo valor de um quadro para outro é pequena. Logo, mesmo que o macrobloco apresente pequenas variações de vídeo de um quadro para outro, como o valor da predição é provavelmente diferente nos dois quadros, essa pequena diferença entre os macroblocos originais poderá se transformar em uma diferença muito maior e mais perceptível na imagem reconstruída. Daí a sensação de *flickering*.

A proposta contida no documento JVT-E070 introduz um processo de quantização na predição (Figura 5.3), não permitindo que o valor da predição seja contínuo em toda sua faixa dinâmica. Isso faz com que as pequenas diferenças entre os pixels de um quadro para outro se encaixem nos degraus de quantização e esses valores possam ser replicados exatamente de um quadro para o quadro seguinte.

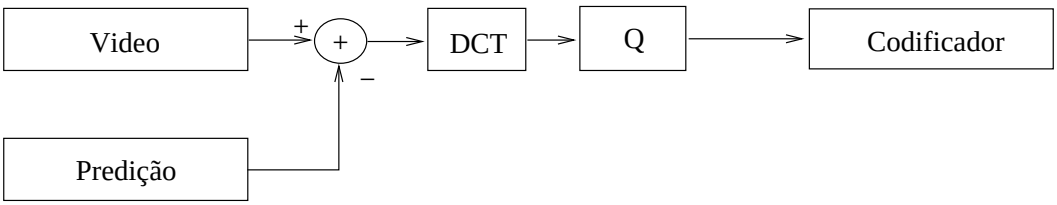


Figura 5.2: Diagrama em blocos da codificação INTRA do H.264

5.3 Método proposto

Apesar da quantização da predição proposta pelo documento JVT-E070 permitir com que os níveis dos pixels reconstruídos de um quadro se repitam no quadro

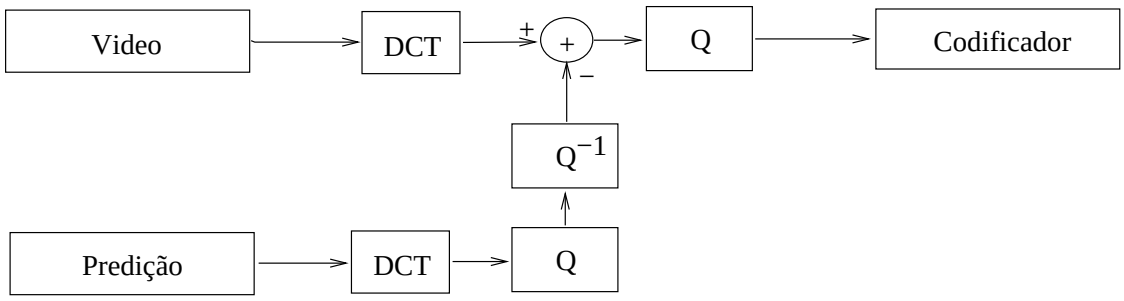


Figura 5.3: Diagrama em blocos da modificação proposta para a codificação INTRA do H.264

seguinte, caso o codificador não tenha o necessário cuidado ao quantizar o resíduo, valores muito semelhantes de pixel de um quadro para outro podem cair em diferentes níveis de quantização, agravando essa diferença. Se além disso o valor do pixel de um quadro para outro apresentar uma pequena e imperceptível oscilação, essa oscilação pode se refletir em uma oscilação no nível de quantização, ocasionando em um efeito de *flickering*.

O método consiste em definir uma *Zona de Perigo* na curva de quantização. Essa região consiste em uma faixa em torno do ponto crítico (*threshold*) de mudança do valor quantizado (Figura 5.4).

No codificador, todos os coeficientes da transformada de cada bloco são armazenados em duas matrizes: uma correspondente ao nível contínuo do coeficiente e a segunda contendo o valor quantizado. No quadro seguinte, ao quantizar um determinado coeficiente, observamos o coeficiente contínuo do mesmo bloco no quadro anterior. Se ambos estiverem na mesma Zona de Perigo, repetimos o coeficiente quantizado do bloco no quadro anterior que está armazenado em uma das matrizes mencionadas acima. Caso o coeficiente não se encontre em uma Zona de Perigo, ou se a Zona de Perigo não for a mesma do quadro anterior, realizamos a quantização original para esse coeficiente.

A figura 5.5 mostra um exemplo teórico que ilustra o funcionamento do algoritmo *Anti-Flickering*. O sinal original apresenta níveis com pequenas variações de nível dentro da Zona de Perigo. Essas variações são bastante comuns em um sistema de captação de imagens e são praticamente imperceptíveis ao sistema visual humano, por possuir baixíssima amplitude. Entretanto, por estarem oscilando ao

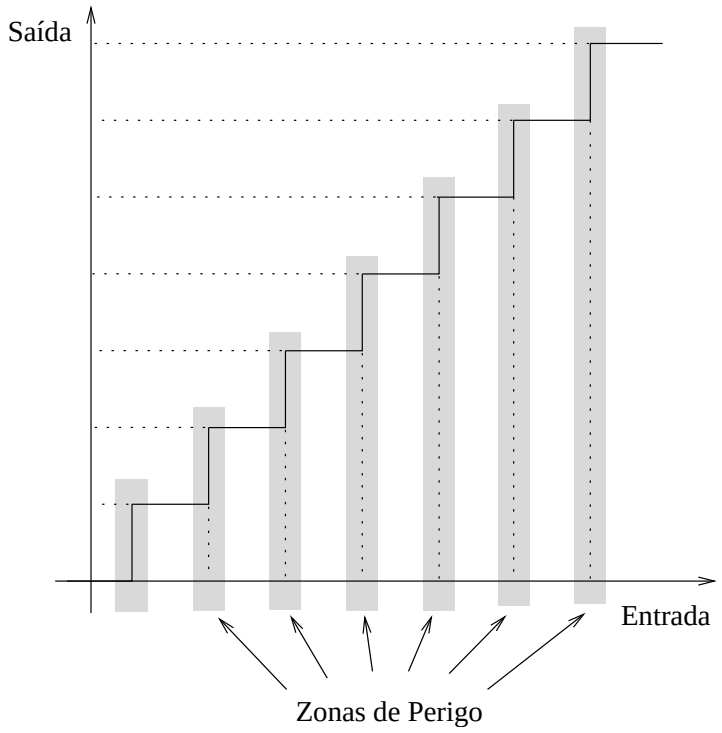


Figura 5.4: Curva do quantizador com Zonas de Perigo destacadas

redor do *threshold* do quantizador, este apresentará na saída um sinal oscilando nos níveis quantizados. A quantização fez com que essa pequena e imperceptível oscilação se transformasse em uma oscilação de grande amplitude, que agora é bastante perceptível ao sistema visual humano e produz o desagradável efeito de *flickering*. O algoritmo *Anti-Flickering* proposto prevê essa possibilidade de *flickering* por essa oscilação se encontrar dentro da Zona de Perigo definida e assim repete o nível de quantização do quadro anterior, mantendo um nível constante na saída. Mantendo o nível de saída constante, temos o efeito de *flickering* totalmente cancelado.

As figuras 5.6, 5.7, 5.8 e 5.9 ilustram um exemplo prático onde é possível visualizar o efeito de *flickering* [1]. A figura 5.6 corresponde ao quadro número 10 da sequência *Cognac and Fruit*. Nesse quadro foi demarcada uma região onde o efeito de *flickering* será demonstrado. A figura 5.7 mostra a imagem correspondente ao bloco demarcado para os quadros 10 a 15 da sequência. Pode-se perceber que não há variações visíveis nos níveis do pixels desse bloco da imagem ao longo dos quadros

[1] As imagens dos blocos são exibidas com acentuação de contraste, para que o efeito se torne mais visível ao leitor

apresentados. A figura 5.8 mostra o mesmo bloco porém codificado com o algorítmo H.264. É notável a grande variação de nível nos pixels do bloco ao longo dos quadros. Essa variação no tempo é a causa do efeito desagradável de *flickering*. Na figura 5.9 vemos o resultado da codificação H.264 com o algorítmo *Anti-Flickering* proposto embutido. O resultado mostra que não há variações visíveis no nível dos pixels ao longo dos quadros, ocasionando na eliminação do efeito de *flickering*.

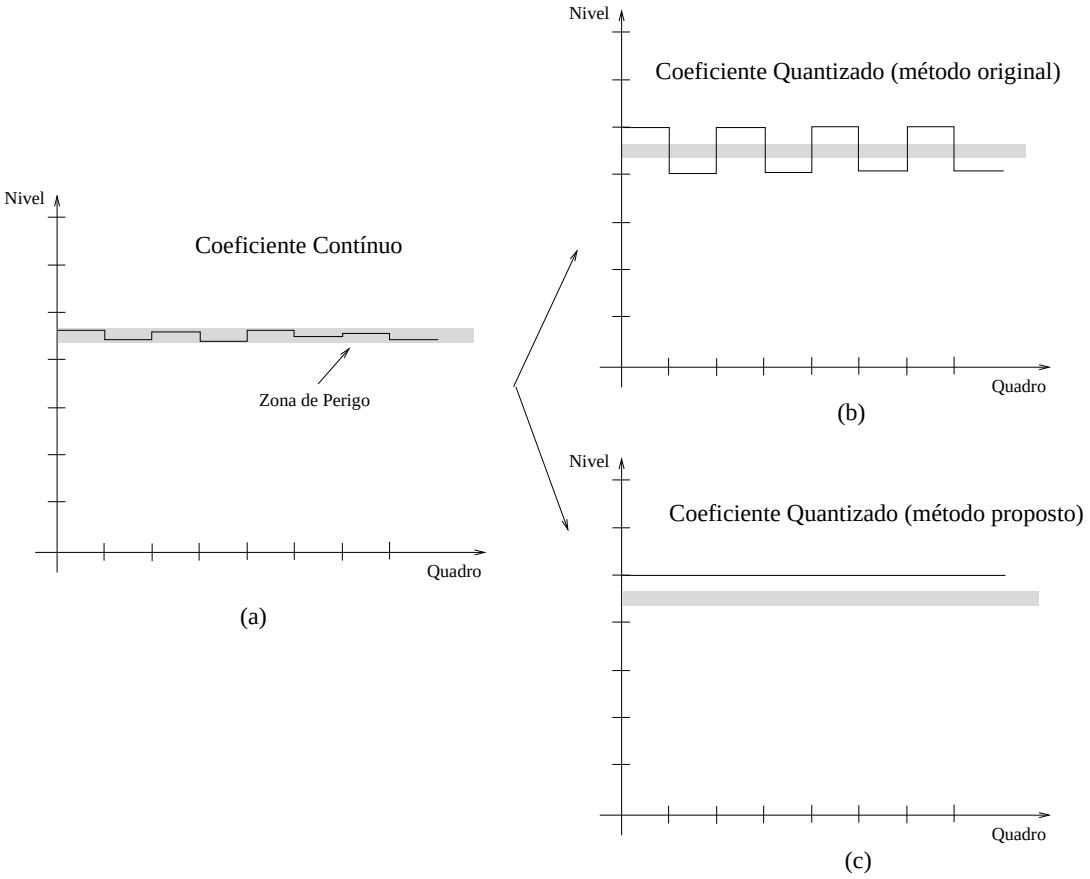
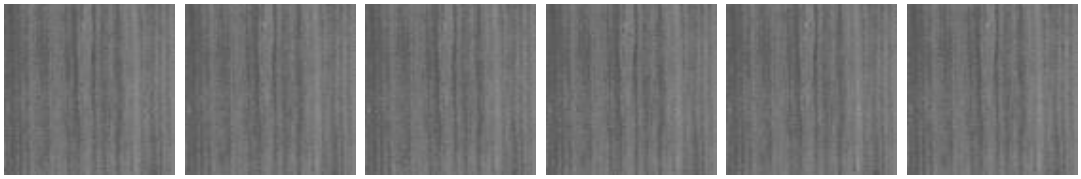


Figura 5.5: Exemplo de funcionamento do algoritmo: (a) indica um sinal na entrada do quantizador. (b) representa a saída do quantizador original do H.264 e (c) mostra o método de quantização proposto

5.4 Resultados

O método *Anti-Flickering* foi aplicado nas sequências *Boy and Toys, Cognac and Fruit, Container, Foreman, Hall Monitor, Mobile and Calendar* e *Tennis*. Utilizamos como parâmetro de comparação a medida S de *flickering* introduzida

Figura 5.6: Quadro número 10 da sequência *Cognac and Fruit* com região de análise demarcada



(a) (b) (c) (d) (e) (f)

Figura 5.7: Quadros 10 a 15 da sequência original

(a) (b) (c) (d) (e) (f)

Figura 5.8: Quadros 10 a 15 da sequência codificada com H.264 original

(a) (b) (c) (d) (e) (f)

Figura 5.9: Quadros 10 a 15 da sequência codificada com o algorítmo *Anti-Flickering*

nessa seção. Foram realizadas medidas de *flickering* do codificador H.264 original, somente com a contribuição JVT-E070 e o terceiro caso contendo não somente a contribuição JVT-E070 mas também o algorítmo *Anti-Flickering* aqui apresentado. A taxa média de *flickering* para as sequências é apresentada na tabela 5.1.

Tabela 5.1: Taxa média de *Flickering* para os algoritmos

Sequência	JM	JVT-E070	Proposta
Boy and Toys	4107	1851	539
Cognac and Fruit	3385	1634	484
Container	4034	2034	679
Foreman	2545	1003	605
Hall Monitor	3251	1500	539
Mobile and Calendar	3986	2906	866
Tennis	2620	1235	750

As figuras 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 e 5.16 mostram o comportamento dos algoritmos ao longo dos quadros.

Os resultados mostram que o a contribuição JVT-E070 sozinha consegue reduzir o nível médio de *flickering* em até 2.5x, no caso da sequência *Foreman*. No pior caso observa-se uma redução de 1.4x na sequência *Hall Monitor*. Já a combinação JVT-E070 com o algorítmo *Anti-Flickering* aqui apresentado reduz no melhor caso o nível de *Flickering* em 7.6x (*Boy and Toys*) e 3.5x no pior caso (*Tennis*).

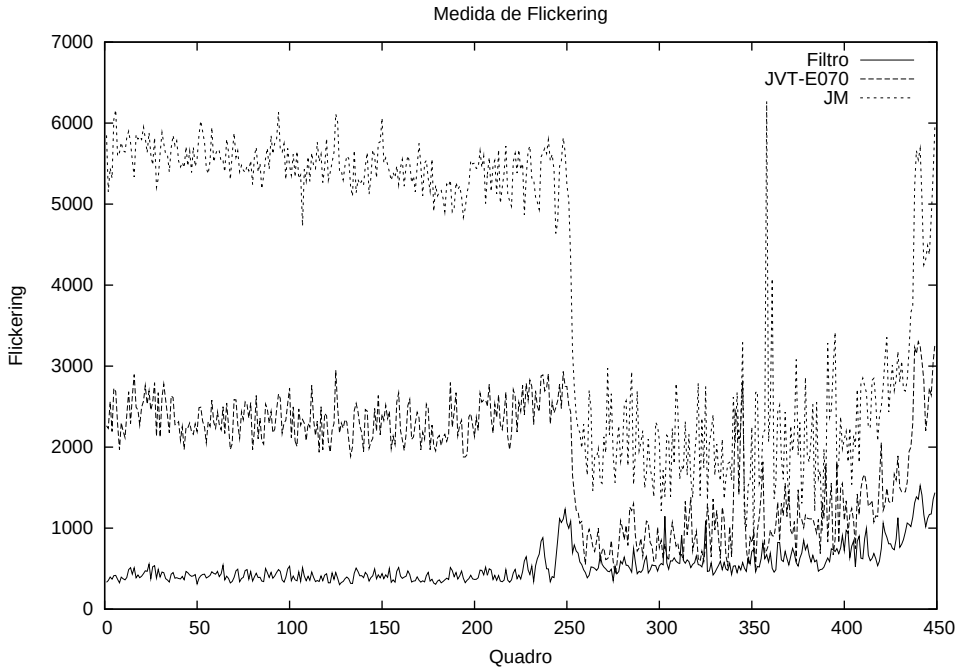


Figura 5.10: Taxa de *Flickering* da sequência *Boy and Toys*

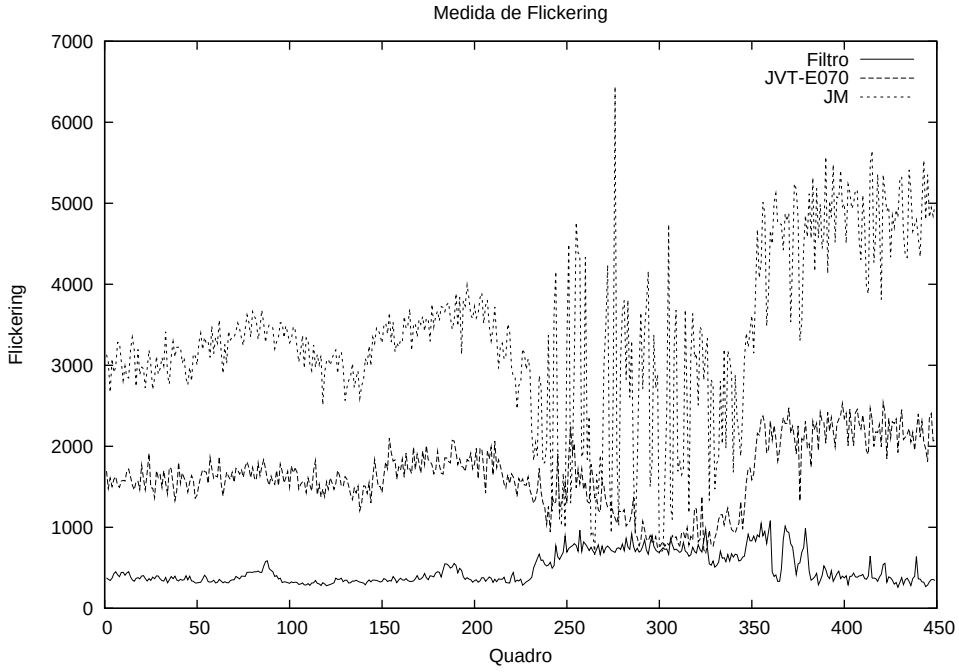


Figura 5.11: Taxa de *Flickering* da sequência *Cognac and Fruit*

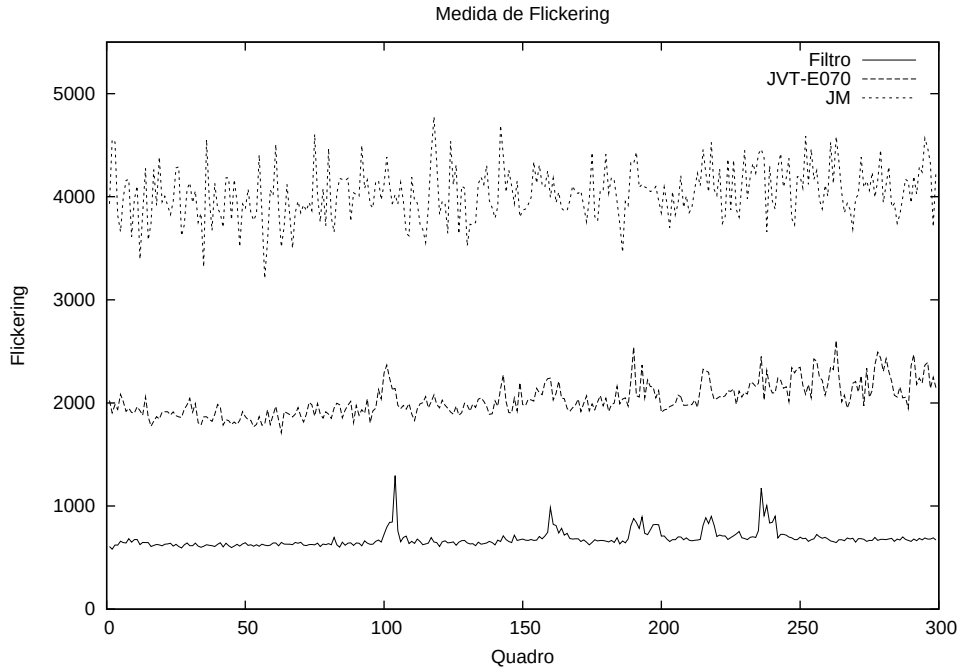


Figura 5.12: Taxa de *Flickering* da sequência *Container*

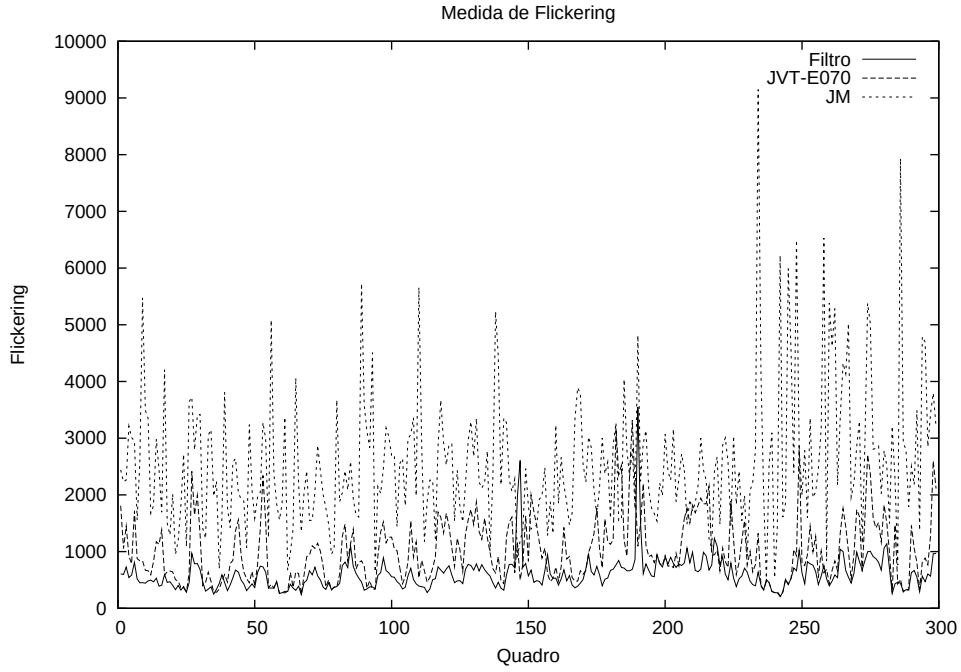


Figura 5.13: Taxa de *Flickering* da sequência *Foreman*

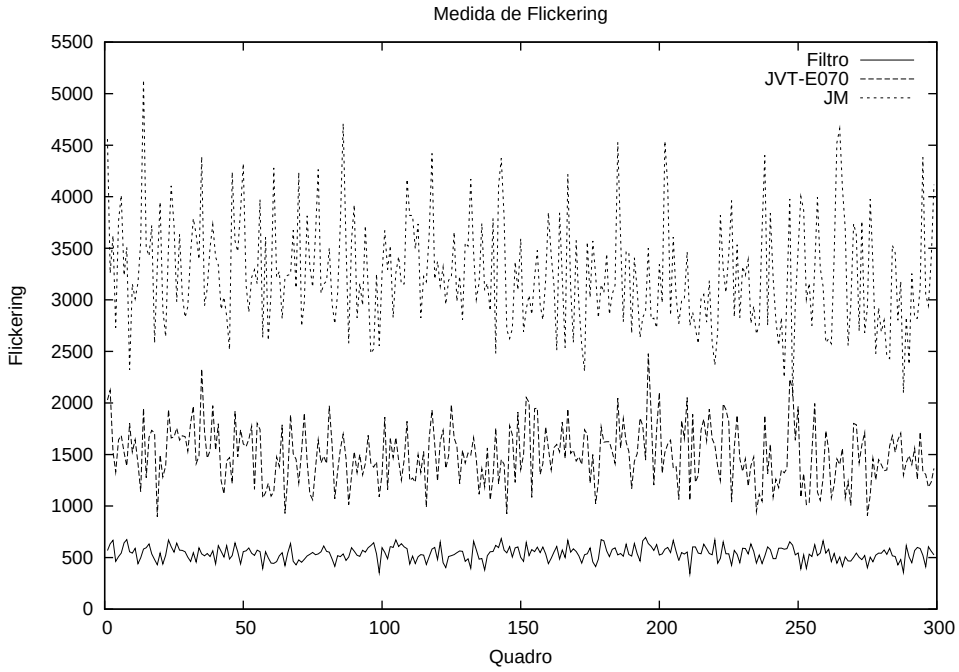


Figura 5.14: Taxa de *Flickering* da sequência *Hall Monitor*

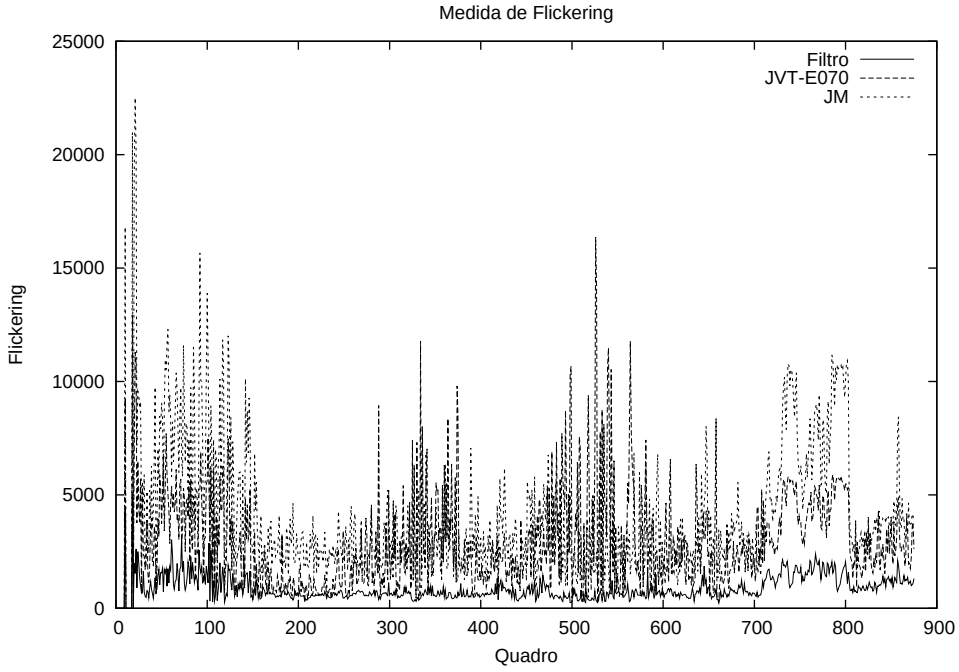


Figura 5.15: Taxa de *Flickering* da sequência *Mobile and Calendar*

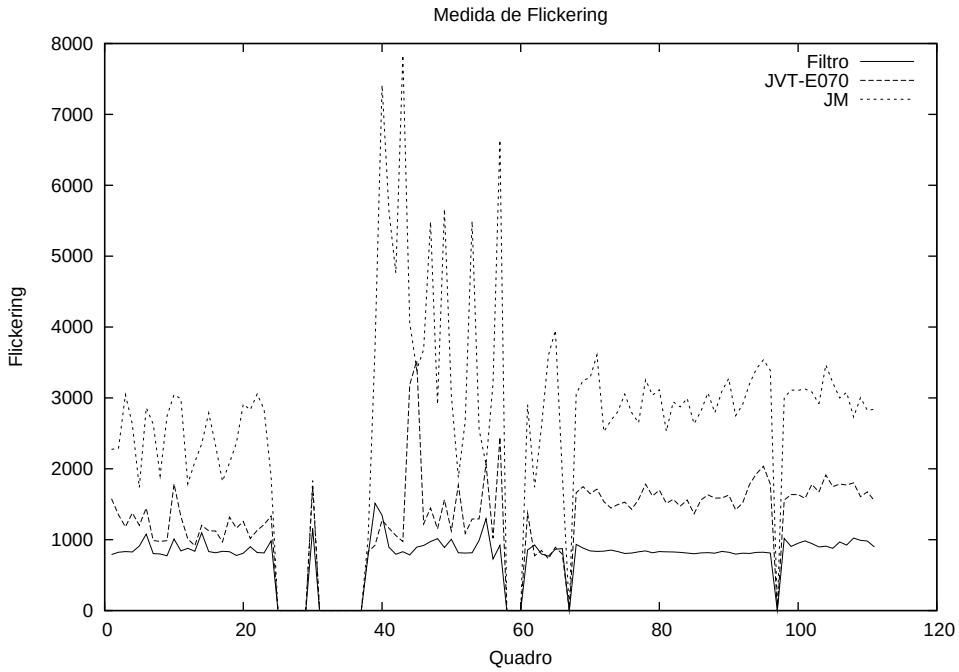


Figura 5.16: Taxa de *Flickering* da sequência *Tennis*

Capítulo 6

Conclusões

Nessa tese foram estudados diferentes métodos de compressão de vídeo.

Nos capítulos 3, foram propostos dois novos algoritmos para compressão de vídeo sem perdas. Nesses algorítmos foram utilizadas extensões espaço-temporais do preditor espacial LOCO-I e os resultados mostraram que as redundâncias temporais são importantes para a redução da taxa de *bits* desde que sejam exploradas de forma eficiente. A análise dos dois codificadores propostos traz consigo um questionamento sobre a utilização de informação lateral no codificador, que apesar de gerar um *overhead* no código pode conduzir a uma melhor predição, melhorando a eficiência do codificador. Essa mesma eficiência do método dos Preditores Chaveados não conseguiu ser alcançada com o codificador 3D, que não utiliza informação lateral. Isto acontece porque o melhor preditor para o pixel não conseguiu ser um bom estimador. Sugerimos como trabalho futuro uma tentativa de se melhorar a adaptação do preditor tridimensional, alternando entre as predições espacial e temporal, para que o algoritmo atinja eficiência compatível com o método do Chaveamento de Preditores.

O utilização do H.264 para a codificação de pares de imagens estéreo provou mostrou que as disparidades geométricas conseguiram ser bem modeladas pela eficiência do esquema de estimação de movimento, com blocos de tamanho variável e algoritmos de busca com otimização da relação sinal-ruído em vista da taxa de *bits* necessária. Essa verificação mostra que o H.264 é um codificador de grande potencial e que pode ser utilizado em uma vasta gama de aplicações, inclusive as não abordadas durante a sua especificação.

No último algorítmo proposto, o problema de *flickering* foi praticamente eliminado através da utilização do quadro anterior para a detecção da possibilidade da ocorrência desse efeito. Os resultados mostraram a eficácia do algorítmo, que resolve um problema crítico dos codificadores baseados apenas em quadros INTRA. Trabalhos futuros podem estudar o efeito em outros codificadores e buscar formas de contornar o problema nas regiões onde há movimentação de objetos, pois nessas áreas o algoritmo proposto não apresentou a mesma eficácia das regiões estáticas.

Referências Bibliográficas

[1] Al Bovik, *Handbook of Image and Video Processing*. Academic Press, 2000.

[2] A. Tekalp, *Digital Video Processing*. Prentice Hall, 1995.

[3] P. Symes, *Video Compression Demystified*. McGraw Hill, 2001.

[4] K. Sayood, *Introduction to Data Compression*. Academic Press, 2000.

[5] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[6] I. Richardson, *Video Codec Design - Developing Image and Video Compression Systems*. Wiley, 2002.

[7] Nassir D. Memon e Khalid Sayood,"Lossless Compression of Video Sequences," *IEEE Transactions on Communications*, v. 44, n. 10, p. 1340-1345, Outubro 1996.

[8] Marcelo J. Weinberger, Gadiel Seroussi e Guillermo Sapiro,"LOCO-I: A Low Complexity, Context Based, Lossless Image Compression Algorithm," *IEEE Data Compression Conference*, Março-Abril 1996.

[9] Sahng-Gyu Park, Edward J. Delp e Haoping Yu,"Adaptive Lossless Video Compression using an Integer Wavelet Transform," *IEEE International Conference on Image Processing*, Outubro 2004.

[10] Huiyuan Wang e David Zhang,"A Linear Edge Model and its Application in Lossless Image Coding," *Signal Processing - Image Communication*, v. 19, p. 955-958, 2004.

[11] I. Witten, R. Neal e J.G. Cleary,"Arithmetic Coding for Data Compression," *Communications of the ACM*, v. 30, p. 520-540, 1987.

[12] Henrique Malvar e Gary Sullivan,"YCoCg-R: A Color Space with RGB Reversibility and Low Dynamic Range," *Proposta JVT-I014r3 para JVT-Joint Video Team. http://ftp3.itu.ch/av-arch/jvt-site/2003_09_SanDiego/JVT-I014r3.doc*, Julho 2003.

[13] D. Tzovaras and N. Grammalidis and M. G. Strintzis,"Object-based coding of stereo image sequences using joint 3-D motion/disparity compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, v. 7, n. 2, p. 312-328, April 1997.

[14] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, 1993.

[15] B. Tseng and D. Anastassiou,"Multi-viewpoint video coding with MPEG-2 compatibility," *IEEE Transactions on Circuits and Systems for Video Technolog*, v. 6, n. 4, p. 414-488, 1996.

[16] W. Woo and A. Ortega,"Overlapped Block Disparity Compensation with Adaptive Windows for Stereo Image Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, v. 10, n. 2, p. 194-200, March 2000.

[17] N. V. Boulgouris and M. G. Strintzis,"Embedded coding of stereo images," *2002 IEEE International Conference on Image Processing*, September 2002.

[18] T. Frajka and K. Zeger,"Residual image coding for stereo image compression," *2002 IEEE International Conference on Image Processing*, September 2002.

[19] H. Duarte and M. Carvalho and E. Silva and C. Pagliari and G. Mendonça,"Stereo Image Coding Using Multiscale Recurrent Patterns," *2002 IEEE International Conference on Image Processing*, September 2002.

[20] Joint Video team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Joint Model Reference Software Version 7.6.

[21] "Draft ITU-T recommendation and final draft internation standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14496-10 AVC", Joint Video Team (JVT) da ISO/IEC MPEG e ITU-T VCEG, JVT-G050, 2003.

[22] Tomas Wiegand, Gary Sullivan, Gisle Bjontegaard e Ajay Luthra,"Overview of the H.264/AVC Video Coding Standard," *IEEE Transactions on Cirtuits and Systems for Video Technology*, v. 13, n.7, p. 560-576, Julho 2003.

[23] Xiaopeng Fan, Wen Gao, Yan Lu e Debin Zhao,"Flicking Reduction in All Intra Frame Coding," *Proposta JVT-E070 para JVT-Joint Video Team. http://ftp3.itu.ch/av-arch/jvt-site/2002_10_Geneva/JVT-E070r2.doc*, Outubro 2002.

[24] T. Halback,"Performance Comparison: H.264 Intra Coding vs JPEG2000," *Proposta JVT-D039 para JVT-Joint Video Team. http://ftp3.itu.ch/av-arch/jvt-site/2002_07_Klagenfurt/JVT-D039.doc*, Julho 2002.

Apêndice A

Sequências e imagens de Teste

A.1 Sequências

Tabela A.1: Sequencias utilizadas para coleta de resultados experimentais

Sequência	Dimensões	Frequência	Quadros
Boy and Toys	352x240	29.97 Hz	450
Cognac and Fruit	352x240	29.97 Hz	450
Container	352x288	29.97 Hz	300
Foreman	352x288	29.97 Hz	300
Hall Monitor	352x288	29.97 Hz	300
Mobile and Calendar	352x288	29.97 Hz	876
Tennis	352x240	29.97 Hz	112

Descrição do conteúdo das sequências:

- *Boy and Toys* - Pequena movimentação de objetos. Pan de câmera no final da sequência.
- *Cognac and Fruit* - Pequena movimentação. Pan de câmera no meio da sequência.
- *Container* - Câmera estática. Navio se movimentando. Alterações na textura da água.

- *Foreman* - Câmera fechada e com trepidações. Pan no final.
- *Hall Monitor* - Câmera estática, pessoas se movimentando. Imagem ruidosa.
- *Mobile and Calendar* - Zoom, pan e intensa movimentação de objetos.
- *Tennis* - Pequenas movimentações de objetos, zoom e cortes de cena.

Figura A.1: Quadro número 50 da sequência *Boy and Toys*

Figura A.2: Quadro número 50 da sequência *Cognac and Fruit*

Figura A.3: Quadro número 50 da sequência *Container*

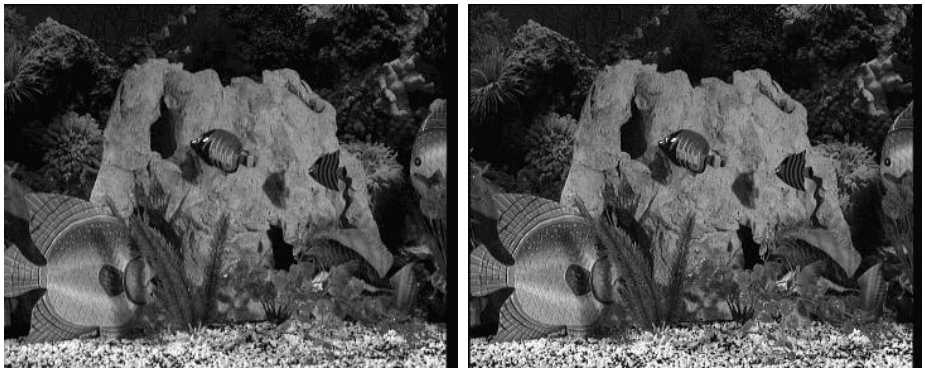
Figura A.4: Quadro número 50 da sequência *Foreman*

Figura A.5: Quadro número 50 da sequência *Hall Monitor*

Figura A.6: Quadro número 50 da sequência *Mobile and Calendar*

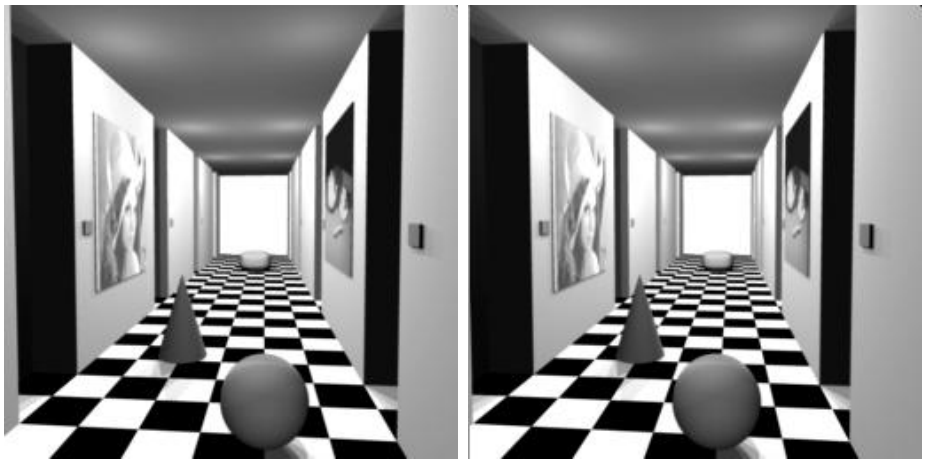
Figura A.7: Quadro número 50 da sequência *Tennis*

A.2 Pares Estéreo



(a) (b)

Figura A.8: Par estéreo *Aqua* - (a) esquerdo e (b) direito



(a) (b)

Figura A.9: Par estéreo *Corridor* - (a) esquerdo e (b) direito

(a) (b)

Figura A.10: Par estéreo *Fruit* - (a) esquerdo e (b) direito

(a) (b)

Figura A.11: Par estéreo *Man* - (a) esquerdo e (b) direito

(a) (b)

Figura A.12: Par estéreo *Saxo* - (a) esquerdo e (b) direito

(a) (b)

Figura A.13: Par estéreo *Train* - (a) esquerdo e (b) direito