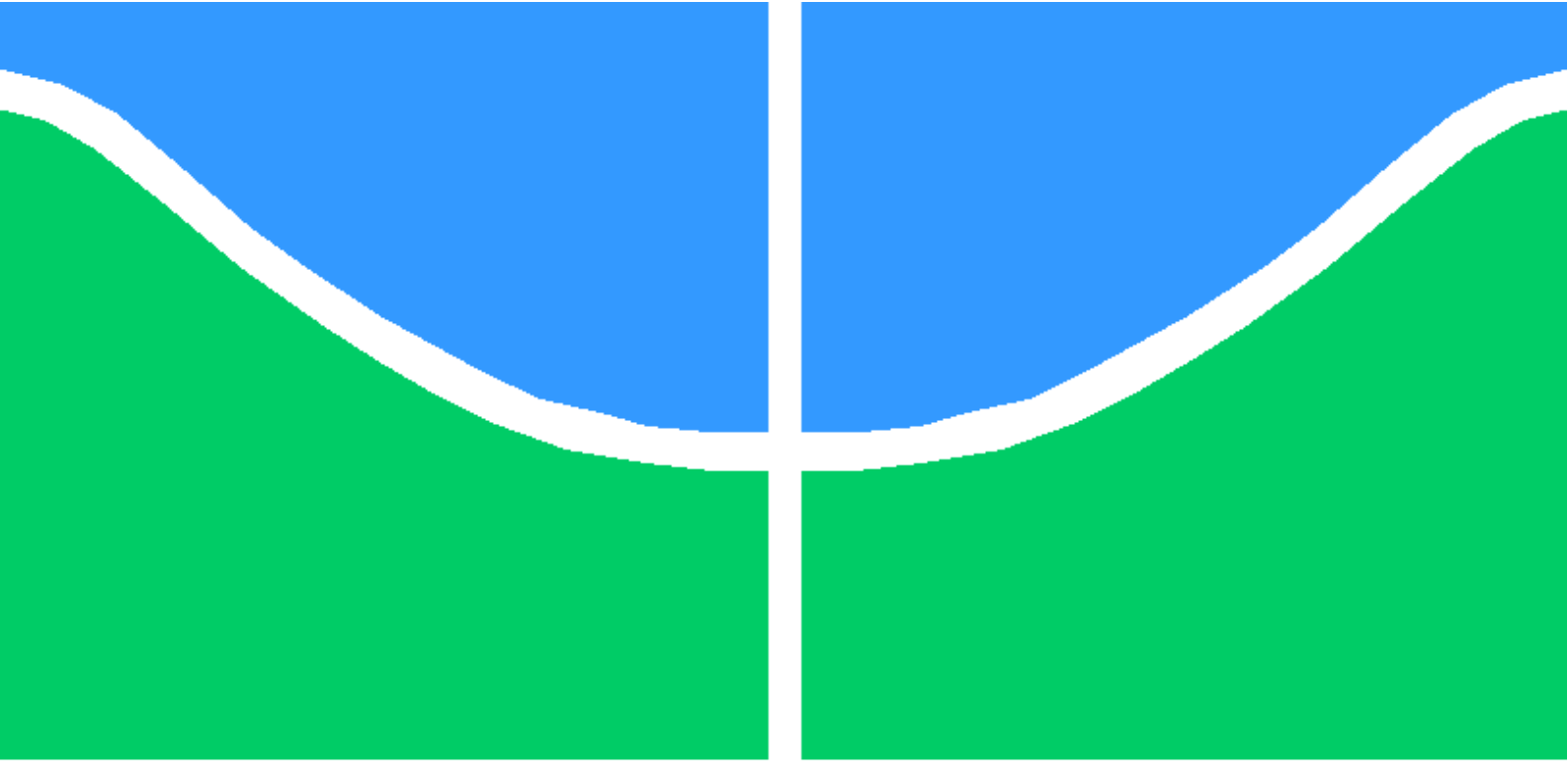


DISSERTAÇÃO DE MESTRADO EM
ENGENHARIA ELÉTRICA

CODIFICADOR H.264/AVC COM COMPENSAÇÃO DE MOVIMENTO BASEADA EM PARTIÇÕES ALTERNATIVAS DE MACROBLOCO

Renan Utida Ferreira

Brasília, Janeiro de 2009

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CODIFICADOR H.264/AVC COM COMPENSAÇÃO DE MOVIMENTO BASEADA EM PARTIÇÕES ALTERNATIVAS DE MACROBLOCO

RENAN UTIDA FERREIRA

ORIENTADOR: RICARDO LOPES DE QUEIROZ

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM - 364A/09
BRASÍLIA/DF: JANEIRO – 2009

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CODIFICADOR H.264/AVC COM COMPENSAÇÃO DE MOVIMENTO BASEADA EM PARTIÇÕES ALTERNATIVAS DE MACROBLOCOS

RENAN UTIDA FERREIRA

DISSERTAÇÃO DE MESTRADO ACADÊMICO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

RICARDO LOPES DE QUEIROZ, Dr, ENE/UNB
(ORIENTADOR)

ADOLFO BAUCHSPIESS, Dr., ENE/UNB
(EXAMINADOR INTERNO)

EDUARDO ANTÔNIO BARROS DA SILVA, Dr., COPPE/UFRJ
(EXAMINADOR EXTERNO)

BRASÍLIA, 30 DE JANEIRO DE 2009.

FICHA CATALOGRÁFICA

FERREIRA, RENAN UTIDA
Codificador H.264/AVC com Compensação
de Movimento Baseada em Partições
Alternativas de Macrobloco. [Distrito Federal] 2009.
xii, 83p., 210x297 mm (ENE/FT/UnB, Mestre, Telecomunicações
Processamento de Sinais, 2009). Dissertação de Mestrado.
Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Elétrica.

1. Codificação de Vídeo	2. H.264/AVC
3. Estimação de Movimento	4. Compensação de Movimento
5. Partição de Macrobloco	6. Partições *Wedge*
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

FERREIRA, R. U. (2009). Codificador H.264/AVC com Compensação de Movimento Baseada em Partições Alternativas de Macrobloco. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM - 364A/09, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 83p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Renan Utida Ferreira.

TÍTULO DA DISSERTAÇÃO DE MESTRADO: Codificador H.264/AVC com Compensação de Movimento Baseada em Partições Alternativas de Macrobloco.

GRAU / ANO: Mestre / 2009

_____

Renan Utida Ferreira
SQS 105 Bloco C Apto 201
70344-030 Brasília - DF - Brasil.

Dedicatória

Ao meu avô, Francisco Bonifácio Ferreira

Renan Utida Ferreira

Agradecimentos

Agradeço aos meus pais, Clarice e Helber, por todo o incentivo e apoio ao meu desenvolvimento acadêmico. Se não fosse por eles, eu nunca teria chegado aqui. Agradeço também aos meus irmãos, cunhados e sobrinhos por todos os momentos de alegria que me proporcionaram. Agradeço aos colegas e amigos do GPDS pelo companheirismo, por comartilharem seus sábios conhecimentos e também pelos divertidos momentos de descontração. Agradeço a todos os professores que contribuíram para a amplicação dos meus conhecimento, em especial ao meu orientador, Professor Ricardo Lopes de Queiroz, por me fazer gostar cada vez mais da pesquisa e da vida acadêmica. Agradeço à HP, pelo apoio financeiro. Agradeço à Vívian, por tudo.

Renan Utida Ferreira

RESUMO

O mais recente padrão de codificação de vídeo H.264/AVC representa um grande avanço em comparação com os padrões anteriores. Uma de suas vantagens é a possibilidade de uma melhor codificação preditiva baseada numa maior segmentação dos quadros do vídeo por meio do uso de partições de macroblocos. Entretanto, esta segmentação é limitada às direções vertical e horizontal. Este trabalho consiste na implementação, neste padrão, de partições alternativas dos macroblocos. Um tipo dessas partições alternativas é determinado por segmentos de retas de direções arbitrárias chamadas *wedges* (cunhas). Uma contribuição deste trabalho é a apresentação das partições *One-Wedge* e partições por Máscara Binária Arbitrária. Visto que o uso de partições *wedges* tem um elevado custo computacional, um outra contribuição é a proposição de métodos de redução da complexidade computacional. Os resultados mostram que o uso de partições alternativas produzem um ganho de compressão e que a redução de complexidade é viável.

ABSTRACT

The most recent video coding standard H.264/AVC represents a great advance in comparison with previous standards. One of its advantages is the possibility of a better predictive coding based on an improved video frame segmentation through the use of macroblock partitions. However, this segmentation is limited to the vertical and horizontal directions. This work consists of the implementation, in this standard, of alternative macroblock partitions. One type of these alternative partitions is determined by a straight line segment of arbitrary direction called a "wedge". A contribution of this work is the use of "One-Wedge" partitions and Arbitrary Binary Mask based partition. Given that the use of wedge partitions has a high computational cost, another contribution is the proposition of methods for the reduction of the computation complexity. The results show that the use of alternative partitions yield a compression gain and that the complexity reduction is viable.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS, ABREVIAÇÕES E ACRÔNIMOS

Abreviações, Acrônimos e Siglas

AVC	*Advanced Video Coding*
BD	*Blu-Ray Disc*
CIF	*Common Intermediate Format*
CODEC	Codificador e Decodificador
DCT	*Discrete Cosine Transform*
DPCM	*Differential Pulse Code Modulation*
DVC	*Distributed Video Coding*
DVD	*Digital Versatile Disc*
EM	Estimação de Movimento
GB	*Gigabyte*
H.263	Padrão de Compressão de vídeo
H.263+	Padrão de Compressão de vídeo (versão melhorada do H.263)
H.264/AVC	Padrão de Compressão de vídeo (Advanced Video Coding)
H.264/SVC	Padrão de Compressão de vídeo (Scalable Video Coding)
H.26L	Codificador implementado entre os padrões H.263+ e H.264/AVC
IDCT	*Inverse Discrete Cosine Transform*
IEC	*International Electrotechnical Commission*
ISO	*International Organisation for Standardization*
ITU	*International Telecommunication Union*
JBIG	*Joint Bi-level Image Compression Group*
JPEG	*Joint Photographic Experts Group*
JM	*Joint Model*
JSMV	*Joint Scalable Video Model*
kbps	kilo *bits* por segundo
Mbps	Mega *bits* por segundo
MPEG	*Motion Picture Experts Group*
MPEG-1	Padrão de Compressão de vídeo
MPEG-2	Padrão de Compressão de vídeo
MPEG-4	Padrão de Compressão de vídeo
MSE	*Mean Square Error*
NTSC	*National Television System Committee*
PSNR	*Peak Signal to Noise Ratio*
QCIF	*Quarter Common Intermediate Format*
SAD	*Sum of Absolute Differences*
SSD	*Sum of Squared Differences*
SVC	*Scalable Video Coder*
TV	Televisão
VCEG	*Video Coding Experts Group*
VM	Vetor de Movimento
VMd	Vetor de Movimento Diferencial
VMest	Vetor de Movimento Estimado
VMpred	Vetor de Movimento Predito

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O vídeo digital é um sinal tridimensional, discretizado em duas dimensões espaciais e no tempo. Para cada instante de tempo, tem-se um quadro bidimensional composto por *pixels*. Um *pixel* (contração de elemento de imagem, ou *picture element* do inglês) é o menor elemento discreto de uma imagem digital. A Figura 1.1 mostra isso.

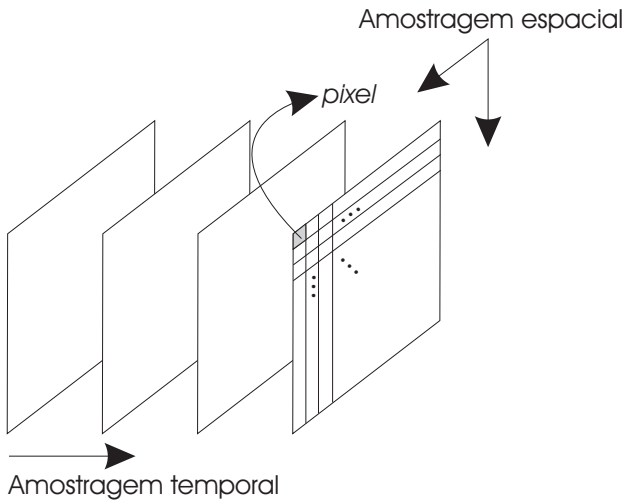


Figura 1.1: Amostragem temporal e espacial em vídeo digital.

A quantidade de informação armazenada em um *pixel* depende da escala de variabilidade que se deseja. Uma imagem em preto e branco, por exemplo, necessita de apenas um *bit* por *pixel*. Por outro lado, uma imagem com 256 níveis em escala de cinza necessita de 8 *bits* para cada nível. Imagens coloridas precisam de mais informação ainda, pois as cores agregam informação extra à imagem. A informação de níveis de cinza é chamada de luminância, enquanto que a informação de cores é chamada crominância. Como os seres humanos são mais sensíveis às alterações de luminância que de crominância, a quantidade destas informações por *pixel* pode ser reduzida [4].

Outra importante informação a respeito da imagem digital do vídeo, ou quadro, é a sua resolução. A resolução é a quantidade de linhas e colunas de um quadro. Existem vários tamanhos padronizados, como o CIF (*Common Intermediate Format*), que tem tamanho 352x288 *pixels*, SD

(*Standard Definition*), com tamanho 720x480 *pixels* e *Full* HD (*High Definition*), com tamanho 1920x1080 *pixels*. Na outra dimensão de amostragem do vídeo, o domínio temporal, um vídeo tem a sua taxa de quadros, que é a quantidade de quadros existentes em um segundo, ou seja, a frequência de apresentação desses quadros, dada em Hz.

A codificação de vídeo digital (também chamada de compressão de vídeo) é o processo pelo qual um vídeo passa para ter seu tamanho reduzido pela necessidade que se tem de armazenar ou transmitir o seu conteúdo. Este tamanho se refere à quantidade de *bits* e não necessariamente à dimensão espacial. Vídeos digitais não-comprimidos ou "*raw*" (crus) geralmente são representados por uma taxa de *bits* extremamente elevada, quando comparada às capacidades de transmissão de dados comerciais atuais. A taxa de *bits* é a quantidade de *bits* por segundo (bps).

Como exemplo, cita-se uma sequência de vídeo com qualidade de TV NTSC (*National Television System Committee*), que possui taxa aproximada de 216 Mbps (resolução 858x525, 16 *bits* por *pixel* e taxa de quadros de 30 Hz) [5], o que é bastante alto em comparação às taxas de transmissão de banda larga atualmente disponíveis comercialmente para usuários domésticos no Brasil (algo em torno de 4 Mbps). Para o caso de armazenagem de um conteúdo de vídeo, se a informação for não-comprimida, o espaço necessário para um filme de duas horas, por exemplo, no formato SD a 30 Hz (também com 16 *bits* por *pixel*) seria de, aproximadamente, 104,28 GB (*gigabytes*, ou 2^{30} *bytes*), valor esse muito maior que a capacidade máxima de um DVD (*Digital Versatile Disc*), que é de 17,08 GB (dupla face e dupla camada). Existe então uma necessidade muito grande de melhor codificar sinais de vídeo.

Os padrões de codificação, incluindo os de vídeo, são desenvolvidos por órgãos internacionais visando impor restrições, de forma a se manter uma certa uniformidade. Alguns dos padrões de codificação mais conhecidos são o codificador de imagem JPEG (*Joint Photographic Experts Group*) [6] e o codificador de vídeo MPEG-2 (*Motion Picture Experts Group*, também conhecido por H.262) [7], usado nos filmes em DVDs e no padrão de TV digital europeu [8]. Contudo, o aumento na quantidade de informação tornou obrigatório o desenvolvimento de novos padrões, como é o caso do padrão H.264/AVC (*Advanced Video Coding*) [2] para codificação de vídeo. Um exemplo de uso deste novo padrão é na mídia digital *Blu-ray Disc* (BD).

1.2 DEFINIÇÃO DO PROBLEMA

Apesar de o padrão H.264/AVC já ter melhorado bastante a codificação de vídeos, acredita-se que ainda possam ser feitas outras coisas para torná-lo mais eficiente. Quando este padrão foi proposto, uma série de técnicas (novas ou não) foi usada em conjunto para melhorar as várias etapas do processo de codificação, de forma que a melhoria global fosse bem elevada. Entretanto, algumas dessas técnicas não vislumbram toda uma possibilidade de abordagens de forma que ainda podem ser exploradas.

A maneira como os codificadores de vídeo trabalham com cada quadro se dá pela delimitação de regiões do quadro. No caso do H.264, isso acontece em blocos de 16x16 *pixels*, chamados macroblocos. Para aproveitar melhor certas características do vídeo, esses macroblocos podem ser divididos em cortes nas direções vertical e horizontal, chamadas partições do macrobloco. Todavia, essas divisões do macrobloco são soluções um tanto limitadas, o que leva a crer que novas formas de partição sejam mais adequadas.

Outro problema do H.264, assim como praticamente todos os codificadores de vídeo (exceto aqueles baseados em DVC - *Distributed Video Coding* [9]), é a grande complexidade computacional do lado do codificador (enquanto que o DVC transfere a complexidade para o lado do decodificador). Com isso, qualquer inserção de técnicas ou ferramentas em um codificador de vídeo, torna-o ainda mais complexo. Para alguns casos, esse aumento de complexidade é mais aparente que em outros.

A etapa mais custosa do codificador do H.264 é a estimação de movimento (responsável pela busca de descolamentos entre quadros), com um gasto de aproximadamente 90% do tempo total de codificação [10]. Por causa disso, qualquer alteração feita no codificador que implique aumento no tempo gasto para a estimação de movimento terá um impacto final considerável no tempo de execução da codificação. Este é um caso de aumento da complexidade mais perceptível que pode ser provocado pelo aumento no número de partições ou formas de particionar o macrobloco.

1.3 OBJETIVOS

Este trabalho tem por objetivos a implementação de métodos alternativos de partição de macroblocos dentro do padrão de codificação de vídeo H.264, bem como um estudo sobre a redução da complexidade desta implementação. É importante ressaltar que esta redução de complexidade deve evitar o mínimo possível de redução sobre os ganhos obtidos pelo uso das partições.

Por meio desses métodos alternativos de partição, é possível mostrar que as etapas estimação e compensação de movimento (explicadas no Capítulo 2) podem ser melhoradas. Isto permite um aumento na compressão, que pode ser medido pela redução da taxa de *bits* do arquivo comprimido.

Como a inserção desses métodos alternativos é feita na estimação e compensação de movimento, isto provoca um aumento considerável na complexidade computacional do codificador. Por isso, neste trabalho buscamos formas de promover melhorias na codificação, mas que causem um impacto menor no seu custo computacional. Neste contexto, são propostas abordagens de se fazer essas partições alternativas de maneira que exija ao codificador um menor esforço computacional.

1.4 ORGANIZAÇÃO DO MANUSCRITO

O Capítulo 2 apresenta uma descrição das ideias gerais por trás dos codificadores de vídeo, bem como o detalhamento de alguns aspectos do padrão H.264/AVC. No Capítulo 3 são apresentadas as técnicas de partições alternativas de macroblocos, seguidas dos métodos propostos para a redução de complexidade dessas técnicas. Em seguida, o Capítulo 4 traz detalhes sobre a implementação dessas técnicas e os resultados obtidos. Por fim, o Capítulo 5 traz as conclusões deste trabalho.

2 CODIFICAÇÃO DE VÍDEO - O PADRÃO H.264/AVC

A codificação implica a existência de um par de sistemas complementares: o codificador e o decodificador. O codificador é o sistema responsável pela redução da quantidade de *bits* do vídeo para transmissão ou armazenagem. Porém, para que o vídeo seja reproduzido, é necessário que esses *bits* sejam decodificados e este é o papel do decodificador. O par de sistemas codificador-decodificador é comumente chamado de CODEC.

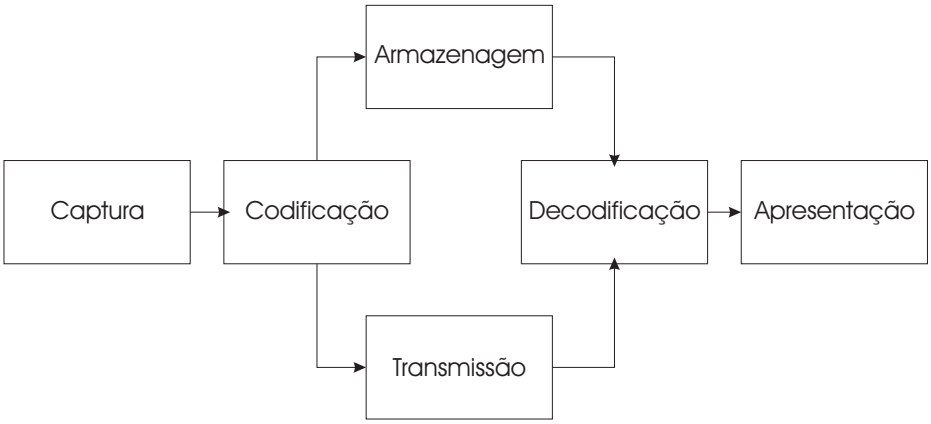


Figura 2.1: CODEC contextualizado.

O padrão H.264/AVC é um padrão de codificação de vídeo desenvolvido pelo *Joint Video Team* (JVT), formado conjuntamente pelo *Video Coding Experts Group* (VCEG), grupo da ITU-T (grupo ITU-T SG16 Q.6) e o MPEG, grupo da ISO/IEC (grupo ISO/IEC JTC 1/SC 29/WG 11).

A União Internacional de Telecomunicações (ITU) é uma agência da ONU especializada no campo das telecomunicações. O Setor de Padronização em Telecomunicações da ITU (ITU-T) é um órgão permanente da ITU responsável por estudos de questões técnicas, operacionais e de tarifação e por pedidos de Recomendações nesses tópicos visando a padronização das telecomunicações ao redor do mundo [11]. A Organização Internacional para Padronização (ISO) é uma organização não governamental formada pelos institutos nacionais de padronização de 157 países e funciona como uma ponte entre os setores público e privado [12].

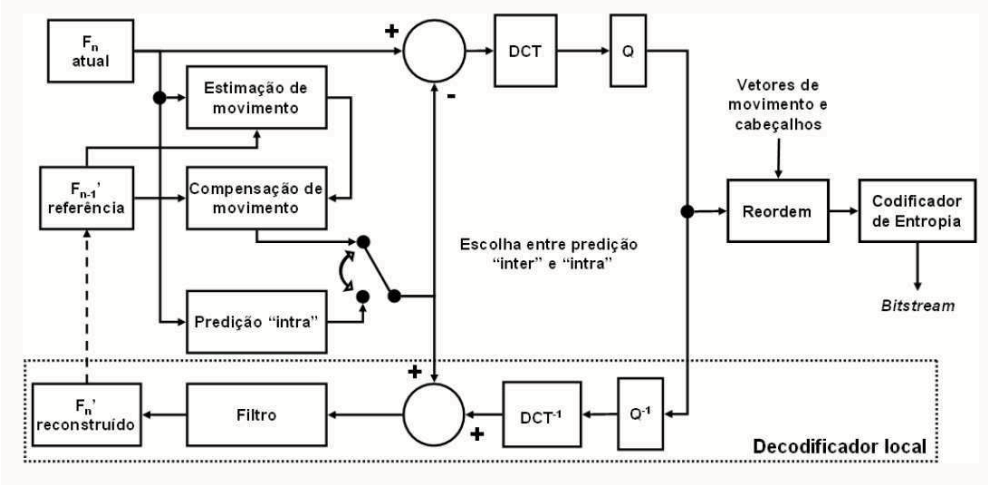
O padrão H.264/AVC, também chamado de MPEG-4 PART 10, é uma evolução dos padrões de codificação de vídeo ITU-T H.261 [13], H.262 (MPEG-2) [7], H.263 [14] e seus sucessores H.263+ e H.263++. Seu desenvolvimento foi motivado pela crescente necessidade por compressão de vídeo mais eficiente para várias aplicações, como vídeo-conferência, armazenagem em meio digital, radio-difusão de sinal de TV, vigilância remota por vídeo, serviços de vídeo-sob-demanda, mensagens multimídia em redes de banda-larga etc. Ele surgiu inicialmente com uma chamada por propostas em 1998 pelo VCEG para o projeto chamado H.26L, que visava duplicar a eficiência de codificação, ou seja, reduzir pela metade a taxa de *bits* necessária para um dado nível de fidelidade, em comparação com qualquer outro padrão existente e para uma grande variedade de aplicações. A primeira versão foi apresentada em Outubro de 1999. Em Dezembro de 2001, foi formado o JVT, que veio a submeter a versão para aprovação em Março de 2003.

O processo de compressão, para todos os tipos de sinais, busca tirar proveito da redundância de informação. Para o caso de vídeos não é diferente. Entretanto, nos vídeos existem dois tipos de redundância: temporal e espacial. A redundância espacial se dá pela correlação que os *pixels* têm com seus vizinhos dentro de um mesmo quadro. Já a redundância temporal ocorre pela semelhança entre quadros subsequentes. É claro que em regiões de bordas bem definidas de objetos e em mudanças de cenas não existe tanta correlação, mas os codificadores se valem de certas técnicas para superar essas dificuldades.

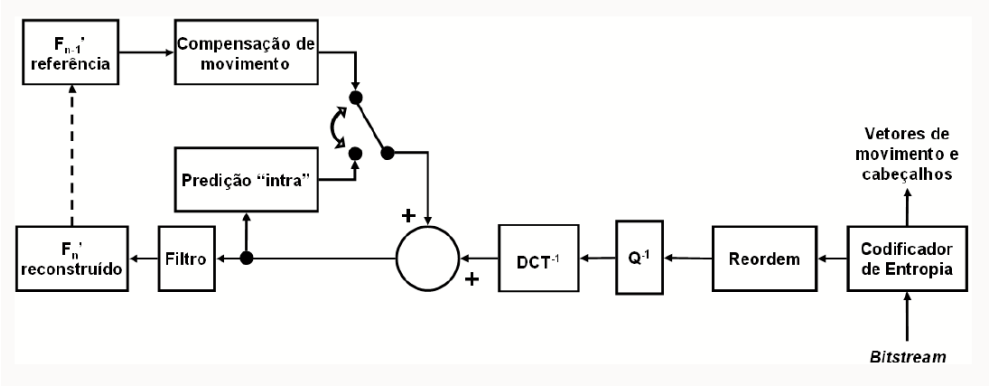
Mas afinal, como é feita a codificação de vídeo? Atualmente, os padrões de codificação de vídeo vigentes nos sistemas de TV digital baseiam-se no modelo híbrido DPCM/DCT de codificação. Ele consiste basicamente em três estágios: um processo diferencial, transformação de domínio e quantização e por fim um codificador de entropia. O termo DPCM (modulação diferencial de código de pulso) vem do modelo de codificação usado em sistemas de comunicação que codifica amostras PCM diferencialmente. Na prática, isto significa que o codificador explora as redundâncias espacial e temporal de maneira diferencial. O termo DCT vem da transformada de cossenos discreta [15] que busca, por meio de transformação de domínios, a concentração de energia em uma menor quantidade de amostras.

A Figura 2.2 mostra os diagramas de blocos simplificados do codificador e do decodificador usados no padrão H.264. Os blocos mostrando a letra "F" (*Frame*), F_n, F'_n e F'_{n-1} representam os

quadros atual, atual reconstruído e reconstruído anterior usado como referência, respectivamente. O bloco "Q" representa a etapa de quantização, sendo que o expoente -1 (tanto para o bloco Q^{-1} quanto DCT^{-1}) indica as etapas inversas. O bloco "Filtro" representa a etapa de filtragem para remoção de efeito de blocos, mas essa parte não será comentada neste trabalho. Os blocos de "Estimação de Movimento", "Compensação de Movimento" e "Predição intra" serão explicados mais a frente.



(a)



(b)

Figura 2.2: Diagramas de bloco do H.264: (a) Codificador; (b) Decodificador [1].

2.1 MODELO TEMPORAL

O modelo temporal tem por objetivo explorar a redundância entre quadros. Neste modelo, cada quadro é codificado de cada vez e a melhor maneira de se explorar a redundância é pela

diferença entre o quadro atual e uma versão predita dele. O quadro atual é aquele sendo codificado no momento. Esta diferença é chamada resíduo ou erro de predição. A predição do quadro atual consiste em buscar em outros quadros previamente codificados e reconstruídos (decodificados localmente) informações que permitam reduzir a energia do resíduo. Assim, quanto melhor a predição, menos energia terá o resíduo. Os quadros usados para a geração do quadro predito são chamados quadros de referência. Do lado do decodificador, o quadro predito é recriado e somado ao resíduo para gerar o quadro atual decodificado. Vale lembrar que a ordem de codificação dos quadros não precisa ocorrer na mesma ordem em que eles são apresentados.

A forma mais simples de predição é usar o quadro imediatamente anterior como quadro predito do quadro atual. A diferença entre eles é então o próprio resíduo. A Figura 2.3 mostra os dois primeiros quadros da sequência "*Foreman*" e a diferença entre esses dois quadros. Entretanto, mesmo que os dois quadros pertençam à mesma cena do vídeo (sem nenhuma mudança abrupta de cena), o resíduo apresentará energia elevada nas regiões de movimento, seja movimento de câmera ou de objetos. Uma boa forma de predição pode então ser feita compensando-se pelo movimento existente na cena. O quadro predito não precisa ser necessariamente um dos quadros anteriores por inteiro, mas sim uma composição deles. Esta composição é chamada justamente de Compensação de Movimento. Porém, é necessário que se determine os movimentos no quadro atual e isto é feito pela Estimação de Movimento.

(a) (b) (c)

Figura 2.3: Sequência *Foreman*: (a) Quadro 1; (b) Quadro 2; (c) Diferença entre Quadros 1 e 2.

2.1.1 Estimação e compensação de movimento

Basicamente, a estimação de movimento consiste em buscar o deslocamento de uma determinada região do quadro atual referenciada por outro quadro. Esse deslocamento pode

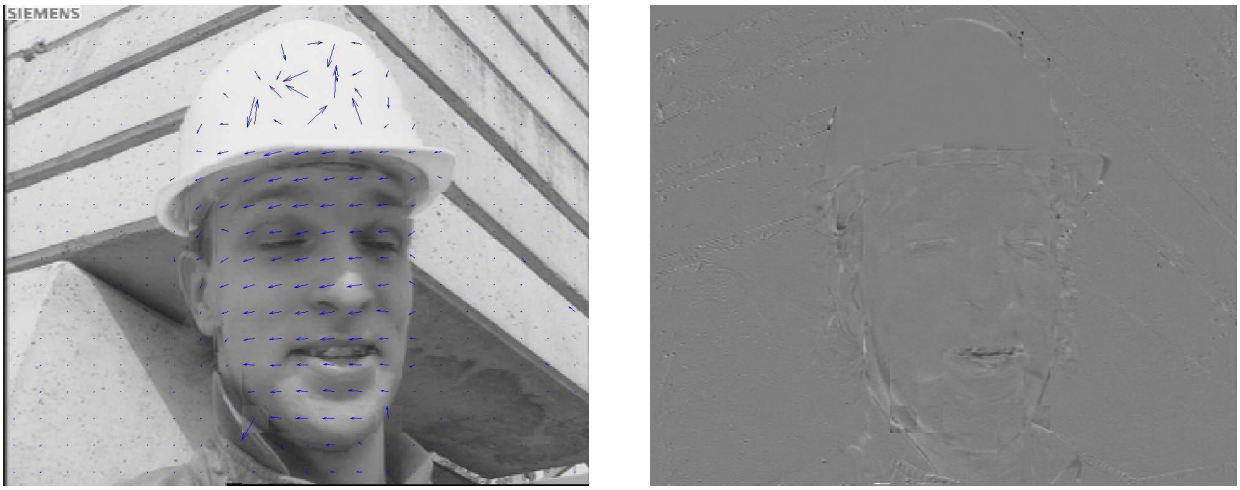
ocorrer pelo deslocamento de algum objeto ou por deslocamento da câmera em relação à cena observada. Desta forma, no caso anterior de dois quadros subsequentes, uma determinada região de um quadro, digamos um objeto qualquer, está necessariamente no outro. Contudo, existe uma grande possibilidade de que a percepção do objeto nos dois quadros seja diferente, seja por mudança na iluminação do objeto, deslocamento do próprio objeto, deslocamento da câmera ou até mesmo desclocamento de um outro objeto sobre o outro. De qualquer maneira, se tomarmos o primeiro quadro como referência, podemos fazer uma busca do objeto presente no segundo quadro e determinar o quanto ele se deslocou. A diferença entre as posições do objeto nos dois quadros é chamada de vetor de movimento, ou VM, e o conjunto dos vetores de movimento de um quadro é chamado de fluxo óptico.

Ao longo do desenvolvimento de padrões de codificação de vídeo, notou-se que a estimação de movimento de objetos inteiros pode ser bastante complexa, principalmente porque pode ser bastante difícil se determinar qual parte do quadro pertence a um objeto qualquer. Outra ideia seria fazer a busca de cada *pixel* individualmente. Isto permitiria um VM exato (ou quase) para cada *pixel*, porém o custo computacional de fazer buscas para todos os *pixels* é extremamente alto. Decidiu-se então por fazer a busca para blocos de *pixels*. Normalmente, esses blocos são de tamanho 16 por 16 *pixels* e são chamados de macroblocos.

A estimação de movimento será então feita bloco a bloco, buscando nos quadros de referência o bloco mais parecido possível com o bloco atual. Para achar aquele bloco de referência que tem um melhor "casamento" com o atual, usam-se técnicas baseadas na diferença entre os blocos. As técnicas mais comuns são a Soma das Diferenças Absolutas (SAD - *Sum of Absolute Differences*) e a Soma da Diferença Quadrática (SSD - *Sum of Squared Differences*). Essa busca pode ser feita de diversas formas diferentes, desde uma busca completa dentre todos os blocos de uma janela de busca dos quadros de referência (janelas que podem ser do tamanho do quadro inteiro) até buscas otimizadas [16].

Depois de feita a estimação de movimento e determinação dos VMs, é possível fazer a compensação de movimento. Assim, constrói-se o quadro predito a partir dos diversos blocos pertencentes aos quadros de referência encontrados como os melhores casamentos para os blocos do quadro atual.

A Figura 2.4(a) mostra uma predição do segundo quadro da sequência "*Foreman*" feita a partir do quadro anterior (com a partição do quadro atual em macroblocos) e o fluxo óptico gerado no processo de estimação de movimento. Oberservando-se as Figuras 2.3(a) e 2.3(b), percebe-se que o rosto vira em direção à esquerda. Portanto, o fluxo óptico nesta região do quadro aponta na direção oposta, pois ele aponta para a posição original do macrobloco predito no quadro de referência. Já a Figura 2.4(b) mostra o resíduo (diferença) entre o quadro 2 e o quadro predito.

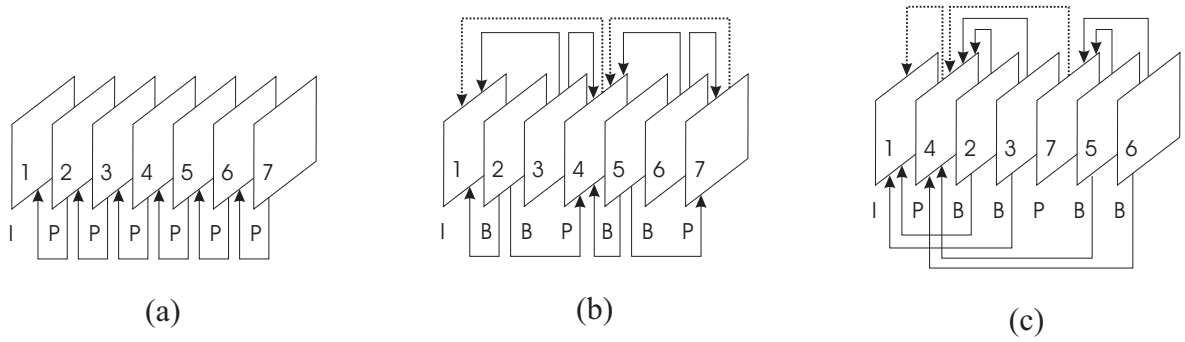


(a) (b)

Figura 2.4: Estimação de Movimento: (a) Quadro Predito e Fluxo Óptico; (b) Resíduo.

No padrão H.264 existem dois tipos de predição temporal (predições Inter-quadros): predição do tipo P e do tipo B (bipredição). Na predição do tipo P, o macrobloco predito provém de apenas um quadro de referência. Já a predição do tipo B permite que o macrobloco predito seja composto por até dois macroblocos. Neste tipo de predição existem duas listas de quadros de referência e ambas podem conter tanto quadros passados quanto futuros, na ordem de apresentação. A primeira, chamada Lista 0, é formada primeiramente com quadros passados e depois futuros. Já a segunda, chamada Lista 1, é formada primeiramente com quadros futuros e depois quadros passados. O macrobloco pode ser escolhido dentre os quadros da Lista 0, da Lista 1 ou de ambas. No último caso, o macrobloco predito resultante será uma média de cada um dos blocos escolhidos de cada lista. As Figuras 2.5(a) e 2.5(b) mostram exemplos com quadros com predições dos tipos P e B, respectivamente, com os quadros sequenciados na ordem de apresentação, enquanto a Figura 2.5 (c) mostra a sequência de codificação. As setas indicam de qual quadro é feita a predição. Existe também a predição do tipo I (Intra-quadro), em que o

macrobloco predito provém do próprio quadro atual, ou seja, para este tipo de predição não existe estimação e compensação de movimento.



(a) (b) (c)

Figura 2.5: Tipos de Predição: (a) Tipo P; (b) Tipos P e B - Ordem de apresentação; (c) Tipos P e B - Ordem de codifição.

Os tipos de predição apresentados acima não restringem o quadro por inteiro. Isso quer dizer que em um mesmo quadro pode haver macroblocos preditos pelos diferentes tipos. Isto porque o quadro pode ser separado em *slices* (ou fatias), que são agrupamentos de macroblocos em ordem *raster* (da esqueda para a direita, de cima para baixo), mas não necessariamente contíguos. As *slices* também são classificadas como I, P e B. Entretanto, as *slices* dos tipos P e B também podem ter macroblocos do tipo I. Isso aumenta a flexibilidade da predição. Além disso, diferentes *slices* podem ser aglutinadas em *Slice Groups*, ou grupos de *slices*.

O padrão H.264 se destaca por alguns avanços no que diz respeito à estimação e compensação de movimento [17]. Aquele que mais interessa a este trabalho é a possibilidade de subdividir os macroblocos em partições menores objetivando a diminuição da energia dos resíduos. Este avanço será detalhado mais adiante, porém é interessante mencionar os demais. Em alguns casos, o avanço foi a melhora de técnicas já existentes em outros padrões ou o agrupamento delas.

A maioria dos padrões de codificação de vídeo anteriores aos H.246 usava estimação de movimento com precisão de metade de *pixel*. Isto decorre do fato de que muitas vezes o deslocamento de objetos não coincide com um número inteiro de *pixels*, principalmente para vídeos de baixa resolução. Desta forma, o que se faz é interpolar os *pixels* existentes para gerar *pixels* fictícios entre eles. Isso parece algo estranho, mas melhora consideravelmente a codificação. A inovação no H.264 é o uso da resolução de um quarto de *pixel* para os vetores de movimento. Apesar de isto já estar presente no padrão MPEG-4 Visual (ou MPEG-4 PART 2), a complexidade do processo de interpolação foi reduzida bastante.

Já foi mencionado anteriormente o uso de mais de um quadro de referência. Isto é uma extensão da técnica já encontrada no padrão H.263++. Na maioria dos padrões anteriores, para cada quadro, era possível haver apenas um quadro de referência. Agora, o codificador permite fazer buscas em uma série de quadros previamente codificados e armazenados nas Listas 0 e/ou 1.

Quando se fala de biprédição, já se assume a possibilidade de codificação de quadros em ordem distinta da ordem de apresentação, ou seja, o codificador pode "saltar" quadros durante a codificação para codificá-los posteriormente. No H.264, existe uma grande flexibilidade para a escolha da ordem de codificação, mesmo para predição do tipo P. Isto permite, por exemplo, que macroblocos vizinhos do quadro atual sejam codificados com predições distintas, de modo que um macrobloco tenha predição de um quadro passado e o outro seja predito de um quadro futuro.

Uma restrição não mais existente no H.264 é o uso de quadros codificados com biprédição como quadros de referência. Sem essa proibição, este padrão aumenta a possibilidade de casamento entre o macrobloco atual e seu predito pelo maior número de variações de macroblocos, e essa flexibilidade permite inclusive que um quadro formado por médias de macroblocos usado como referência seja uma melhor aproximação do quadro atual que os demais.

Outra inovação bastante interessante é a predição de movimento ponderada. Caso a ponderação seja "explícita", o codificador determina um valor de atenuação para os *pixels* do macrobloco predito. Isto é vantajoso para cenas em transição suave ou que escurecem lentamente. Este caso é possível tanto para predição dos tipos P e B. Por outro lado, se for escolhido o uso da ponderação "implícita", possível apenas na biprédição, os pesos dos macroblocos dos quadros de referência das Listas 0 e 1 são determinados pela distância temporal destes em relação ao quadro atual, sendo maior o peso quanto mais próximo for o quadro de referência do atual e menor caso contrário.

2.2 PARTIÇÃO DE MACROBLOCO

Conforme mencionado, um dos avanços do padrão H.264 é a possibilidade de subdividir o macrobloco em blocos menores de *pixels*. Essa partição é chamada Compensação de Movimento

em Estrutura de Árvore e guarda certa semelhança com a estrutura de *quad-tree* (mostrada na Figura 2.6(a)). Nas áreas de processamento de imagens e vídeos e computação gráfica, a estrutura *quad-tree* consiste na subdivisão de blocos de *pixels* (que podem ser pequenos ou até a imagem inteira) em quatro quadrantes. Cada quadrante pode ser novamente dividido em quadrantes e assim por diante, daí o nome que se refere a uma árvore de quadrantes. Esta estrutura pode ser usada para uma série de diferentes fins, incluindo compressão e decomposição de imagens para classificação de regiões.

A principal diferença entre a estrutura de *quad-tree* e aquela usada no H.264 é que, apesar de esta também consistir em uma estrutura de árvore, os blocos não precisam ser divididos em quadrantes, mas podem ser quebrados em pares de blocos retangulares, o que permite maior flexibilidade. Desta forma, um macrobloco pode ser dividido em dois blocos de 16x8 *pixels*, dois blocos de 8x16 *pixels* ou quatro blocos de 8x8 *pixels* (que seria a divisão igual à da *quadtree*). Os blocos de tamanho 8x8 *pixels* podem também ser divididos em sub-blocos de tamanho 8x4 *pixels*, 4x8 *pixels* ou 4x4 *pixels*, como pode ser visto na Figura 2.6(b).

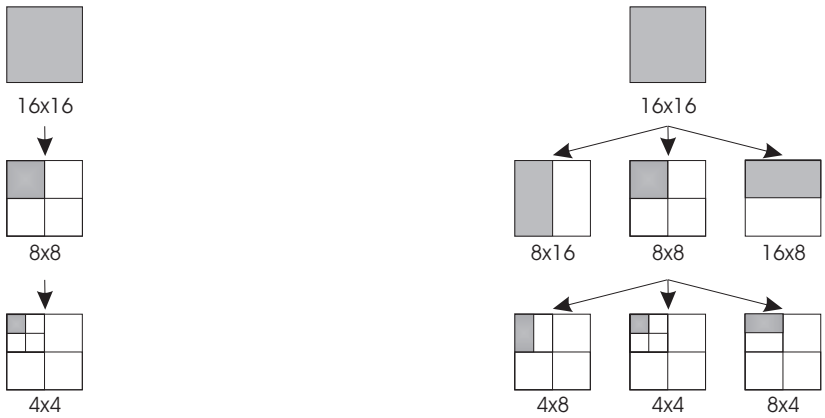


Figura 2.6: Estruturas em Árvore para Partição de Macrobloco: (a) *Quad-Tree*; (b) H.264.

Quando é feita a partição do macrobloco, cada região pode ter um VM diferente, seguindo o mesmo princípio que levou o quadro a ser dividido em macroblocos: dentro de um mesmo macrobloco, diferentes regiões podem guardar semelhanças a partes do quadro de referência que não pertençam a um mesmo macrobloco. Apesar de serem necessários mais *bits* para esses vetores de movimento, diferentemente dos poucos usados para apenas um vetor no caso do macrobloco completo, o ganho que se obtém pela redução do resíduo de cada partição acaba por ser vantajoso.

A tomada de decisão quanto a se há vantagem em dividir o macrobloco (e os blocos por conseguinte) depende em parte da quantidade de *bits* necessários para armazenar tanto os resíduos quanto os vetores de movimento. Somando-se esses *bits* aos demais resultantes da codificação, tem-se a taxa referente ao macrobloco. A outra métrica para essa decisão é a distorção (SAD, por exemplo). O custo de um macrobloco é determinado em função dessas duas métricas.

Para determinar a melhor partição, o macrobloco é classificado em modos, de acordo com o tipo de partição: 16x16, 16x8, 8x16, 8x8. Para cada modo, o codificador verifica o custo. Apenas se o modo 8x8 for o melhor (menor custo), os blocos são divididos e classificados em sub-modos: 8x4, 4x8, 4x4.

Como o processo de estimação de movimento pode ser feito para diferentes quadros de referência, cada partição deve também conter essa informação. Uma maneira encontrada para reduzir tanto a variabilidade dessa informação quanto a quantidade de buscas foi restringir as buscas feitas pelas sub-partições ao quadro de referência de melhor casamento para o bloco 8x8 do qual a partição é oriunda.

2.2.1 Predição do vetor de movimento

Considere uma sequência de quadros em que uma região se desloca por inteiro em uma direção constante. Dado que esta região é grande o suficiente para ser preenchida por vários macroblocos, pode-se assumir que seus vetores de movimento serão semelhantes. O fluxo óptico mostrado na Figura 2.4 é um bom exemplo desta situação. Com base nisso, o padrão H.264 utiliza uma técnica chamada predição do vetor de movimento. Assim como os quadros de um vídeo tendem a ser parecidos entre si quanto mais próximos temporalmente, os vetores de movimento de macroblocos vizinhos muitas vezes também guardam uma certa "redundância de movimento".

Durante o processo de estimação de movimento, o VM encontrado é chamado de vetor de movimento estimado, ou VMest. O vetor de movimento predito, ou VMpred, é justamente o VMest (ou a mediana de três VMest) de macroblocos, blocos ou sub-blocos previamente codificados, vizinhos ao macrobloco, bloco ou sub-bloco atual. Para aproveitar a "redundância de movimento" mencionada, o que se codifica realmente não é exatamente o VMest, mas sim o

vetor de movimento diferencial, ou VMd, que é a diferença entre o VMpred e o VMest.

$$VMd = VMpred - VMest$$

Quando um macrobloco é particionado, cada partição terá um VMest, conforme explicado anteriormente. Claramente, cada partição também deve estar associada a um VMpred. Assim, determinar o VMpred depende do modo de partição do macrobloco e da partição em si. Como a sequência de codificação de macroblocos é *raster*, os VMs usados na predição serão dos blocos à esquerda (A), acima (B) e acima e à direita (C) em relação ao atual (X), como na Figura 2.7(a).

Caso haja mais de uma partição imediatamente à esquerda da partição atual, a escolhida será aquela mais acima. Para o caso de mais de uma partição acima, a usada será aquela mais à esquerda. O terceiro VM (do bloco acima e à direita) será sempre da partição ou sub-partição mais inferior e mais à esquerda. Isso vale tanto para um bloco com seus vizinhos pertencentes a macroblocos diferentes quanto para partições dentro de um mesmo macrobloco. A Figura 2.7(b) mostra um exemplo. Caso não haja um macrobloco vizinho ou ele não tenha um VM associado, o VMpred será igual a zero. Para os modos 16x16 e 8x8, o VMpred é calculado pelo valor mediano em cada direção (horizontal e vertical) dos três VMpred adquiridos de A, B e C. A Figura 2.8 mostra de onde vêm os vetores de movimento preditos para os modos 8x16 e 16x8.

(a) (b)

Figura 2.7: Predição do Vetor de Movimento: (a) Modelo geral; (b) Exemplo.

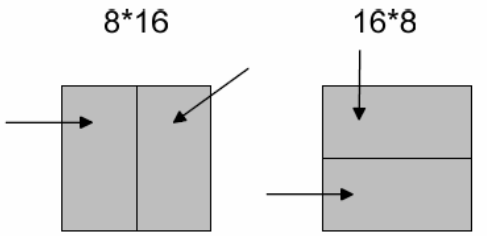


Figura 2.8: Predição no padrão H.264 (informativo) [2].

2.3 MODELO ESPACIAL

O modelo espacial tem por objetivo explorar a redundância espacial dentro de um quadro da sequência de vídeo, descorrelacionando o resíduo para convertê-lo em uma forma que possa ser ainda mais comprimido por um codificador de entropia. Modelos espaciais comumente possuem três componentes principais: transformação (descorrelaciona e compacta informação), quantização (reduz a precisão da informação transformada) e reordenamento (rearranja a informação para agrupar valores significativos).

2.3.1 Codificação por transformada: transformada de cossenos discreta - DCT

O propósito do passo de transformada em um codificador de vídeo é converter o resíduo para um outro domínio. A ideia da transformada depende basicamente de três critérios, todos atendidos pela DCT.

- A informação no domínio transformado deve ser descorrelacionada e compacta.
- O processo de transformada deve ser reversível.
- A transformada deve ser computacionalmente viável.

O cálculo dos coeficientes transformados de uma imagem (um quadro de resíduos, por exemplo) se dá pela operação matricial mostrada na equação 2.1 (sendo que todas as matrizes têm dimensão NxN, em que Θ é a matriz dos coeficientes transformados, A é a matriz de transformação e X é a imagem original a ser transformada). Para se recuperar a imagem original, deve-se fazer o processo de transformação inversa, mostrado na equação 2.2 (assumindo que a transformada seja ortonormal).

$$\Theta = AXA^T \tag{2.1}$$

$$X = A^T\Theta A \tag{2.2}$$

A transformada de cossenos discreta recebe este nome pelo fato de as linhas da matriz de transformação **C** serem obtidas como funções de cossenos, como mostrado pela expressão 2.3. Os produtos externos das linhas da matriz **C** consistem nas bases da transformada, ou seja, o produto interno entre a imagem e a base resulta em um coeficiente transformado. A Figura 2.9 mostra as bases da DCT de tamanho 8x8.

$$[C]_{i,j} = \begin{cases} \sqrt{\frac{1}{N}} \cos \frac{(2j+1)i\pi}{2N} & \text{i=0,j=0,1,...,N-1} \\ \sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N} & \text{i=1,2,...,N-1,j=0,1,...,N-1} \end{cases} \tag{2.3}$$

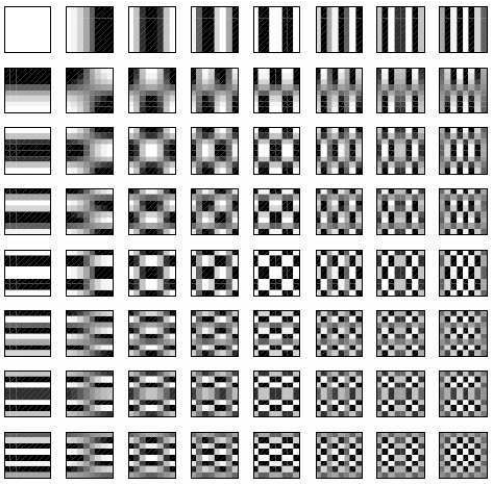


Figura 2.9: Bases da DCT 8x8.

2.3.2 Quantização

O processo de quantização mapeia um sinal com um intervalo de valores X para um sinal quantizado com um intervalo reduzido de valores X_q. Desta forma, é possível representar um sinal quantizado com uma menor quantidade de *bits* que o sinal original. A quantização escalar mapeia um valor do sinal de entrada em um valor quantizado. Já a quantização vetorial mapeia um grupo de amostras do sinal de entrada (vetor) em um grupo de valores quantizados, porém não se falará deste tipo de quantização doravante.

Possivelmente, a maneira mais simples de se realizar a quantização escalar de valores reais é por arredondamento para o valor inteiro mais próximo, ou seja, um mapeamento de $\mathcal{R}$ para $\mathcal{Z}$. Este processo é dito ser com perdas, uma vez que é impossível recuperar o valor original a partir do valor quantizado. Outro exemplo mais geral de quantização é a chamada quantização

uniforme, representado por 2.4, em que Q_p é o passo de quantização, X é o valor a ser quantizado, X_q é o valor quantizado, $\hat{X}$ é o valor recuperado. O valor recuperado é comumente, porém erroneamente, chamado de valor desquantizado. Como o processo de quantização não permite a recuperação do valor original exato para qualquer dos casos, ele é irreversível. Desta forma, um termo mais adequado seria valor reescalonado [18].

$$X_q = round\left(\frac{X}{Q_p}\right) \tag{2.4}$$

$$\hat{X} = X_q.Q_p \tag{2.5}$$

Este processo normalmente é aplicado após o processo de transformada, eliminando coeficientes de valores insignificantes próximos a zero e mantendo apenas um número reduzido de valores significativos. O resultado do processo de quantização dos coeficientes transformados é uma matriz esparsa, composta em sua maioria por entradas iguais a zero. Codificadores de imagem e vídeo, como o JPEG por exemplo, muitas vezes usam uma tabela de quantização, que permite o uso de passos de quantização distintos para cada coeficiente.

Como exemplo, usaremos um bloco de tamanho 8x8 extraído do resíduo apresentado na Figura 2.4(b) e a tabela de quantização do padrão JPEG mostrada na Tabela 2.2. Os valores dos *pixels* deste bloco são apresentado na Tabela 2.1. Os valores dos blocos transformado (pela DCT) e quantizado são mostrados nas Tabelas 2.3 e 2.4, respectivamente.

Tabela 2.1: Bloco 8x8 de resíduos.

-2	-2	-1	6	18	12	-1	-2
-1	0	-1	-1	1	6	5	-1
0	-1	0	2	-7	-6	1	-3
-1	1	-2	-3	-1	2	3	2
-1	1	2	-1	2	1	-2	-1
0	2	1	0	0	0	1	-1
-1	1	2	1	2	2	1	1
1	1	1	1	-1	-3	-2	-2

Tabela 2.2: Tabela de quantização do JPEG.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Tabela 2.3: Bloco transformado.

57,8750	34,0422	-30,8251	-1,8822	3,6250	-8,4951	-5,4972	4,2405
-38,2364	1,1645	-41,5549	22,1580	-27,7356	8,9246	-7,5870	-1,4612
-9,2204	28,3195	-41,8711	33,5663	-34,4742	6,5632	-0,8436	-5,8853
-1,8892	41,9590	-7,5757	25,4611	-1,9353	-2,6480	0,7227	-3,4208
-31,8750	52,5784	1,1200	-11,4830	24,6250	-7,0802	2,3773	2,3093
-4,9106	6,4469	2,1081	-9,7191	4,1768	4,6382	2,3321	-1,9163
-0,3751	-9,9092	-5,3436	7,4830	-7,7741	-1,0881	4,6211	-2,4698
11,2908	-8,5322	-8,7200	6,9507	-5,5660	-4,8681	7,8368	-0,7637

Tabela 2.4: Bloco transformado e quantizado.

-4	3	-3	0	0	0	0	0
-3	0	-3	1	-1	0	0	0
-1	2	-3	1	-1	0	0	0
0	2	0	1	0	0	0	0
-2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

2.3.3 Reordenamento

Os coeficientes transformados e quantizados devem ser codificados de forma mais agrupada possível. Para tal, faz-se uso do reordenamento desses coeficientes para que os valores significativos sejam agrupados e de uma representação mais eficiente dos coeficientes de valor zero. Os coeficientes mais significativos da DCT geralmente se concentram na região de baixa frequência (canto superior esquerdo da matriz), como pode ser observado na Tabela 2.4, e diminuem de energia conforme se afastam para as regiões de alta frequência. O reordenamento se dá então pela varredura em *zig-zag*, ou *zig-zag scan*, mostrado na Figura 2.10.

O resultado do reordenamento é um vetor composto de alguns coeficientes não-zeros e vários coeficientes de valor zero espalhados. Uma maneira de se compactar essa informação diminuindo

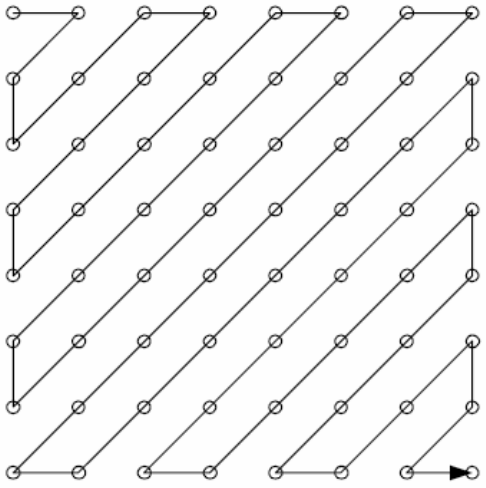


Figura 2.10: Varredura em zig-zag.

o tamanho do vetor é usar a técnica de *run-zero*. Esta técnica consiste em representar o vetor por uma série de pares (*run*,*level*), em que *run* indica a quantidade de zeros precedendo um coeficiente não-zero e *level* indica a magnitude do coeficiente não-zero. Ao final da sequência, deve ser apresentado também um código especial separado que determina o último dos coeficientes não-zeros. Como exemplo, podemos fazer a varredura em zig-zag no bloco apresentado na Tabela 2.4 e o agrupamento em pares (*run*,*level*) e teremos os resultados a seguir:

$$-4, 3, -3, -1, 0, -3, 0, -3, 2, 0, -2, 2, -3, 1, 0, 0, -1, 1, 0, 2, 0, 0, 0, 0, 1, -1, 0 \ldots$$

$$(0, -4), (0, 3), (0, -3), (0, -1), (1, -3), (1, -3), (1, 2), (1, -2), (0, 2), (0, -3), (0, 1), (2, -1)$$

$$(0, 1), (1, 2), (4, 1), (0, -1), (38, 0)$$

onde o valor de $level = 0$ indica o fim do bloco. Nota-se claramente a redução da quantidade de valores a serem codificados de 64 para 34 (17 pares (*run*,*level*)).

2.3.4 O modelo espacial no padrão H.264

O padrão H.264 utiliza três tipos de transformadas, de acordo com o tipo de resíduo a ser codificado. Como o foco deste trabalho está relacionado com o resíduo de predição temporal, apenas este tipo será abordado. A transformada usada opera em blocos de tamanho 4x4 dentro dos macroblocos residuais e é baseada na DCT, mas possui algumas alterações importantes.

As alterações da transformada baseada em DCT visam principalmente à redução da complexidade computacional. A primeira alteração é ter-se criado uma transformada inteira. Por causa disso, todas as operações podem ser executadas por aritmética inteira, sem perda de precisão durante a decodificação. Isso garante que entre o codificador e a transformada inversa no decodificador não haja qualquer descasamento. Outra mudança é a parte central da transformada poder ser implementada usando-se apenas adições e deslocamento de *bits* (multiplicações e divisões por 2). Por fim, parte das operações da transformada (multiplicação de escala) é integrada ao processo de quantização, visando à redução do número total de multiplicações. Os procedimentos de reescala e transformada inversa podem ser feitos usando-se aritmética de 16 *(*bits) sem qualquer perda de precisão, exceto por alguns padrões anômalos de resíduos [18].

A DCT 4x4 é dada pela equação 2.6:

$$\mathbf{\Theta = CXC^T} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \cdot \begin{bmatrix} \\ \mathbf{X} \\ \\ \end{bmatrix} \cdot \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix} \tag{2.6}$$

em que:

$$a = \tfrac{1}{2}, \quad b = \sqrt{\tfrac{1}{2}}\cos\left(\tfrac{\pi}{8}\right), \quad c = \sqrt{\tfrac{1}{2}}\cos\left(\tfrac{3\pi}{8}\right)$$

Todavia, por uma série de operações de fatoração de matrizes, escalonamento e aproximações, chega-se à expressão 2.7, em que C_d é a matriz de transformação direta modificada, $C_d X C_d^T$ é o núcleo da transformada, E_d é a matriz de fatores de escala direta e o símbolo $\otimes$ representa multiplicação escalar entre as entradas de $C_d X C_d^T$ e E_d na mesma posição.

$$\mathbf{\Theta = C_d X C_d^T \otimes E_d} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \cdot \begin{bmatrix} \\ \mathbf{X} \\ \\ \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix} \tag{2.7}$$

sendo:

$$a = \frac{1}{2}, \quad b = \sqrt{\frac{2}{5}}$$

No padrão H.264 existem 52 possíveis valores para o passo de quantização, indexados por um Parâmetro de Quantização, ou QP, sendo que o passo de quantização dobra a cada incremento de 6 no QP, como mostrado na Tabela 2.5. Uma vez calculados os coeficientes transformados não-escalonados $\mathbf{W} = \mathbf{C_d X C_d^T}$, para o cálculo dos coeficientes quantizados Z_{mn}, o passo de quantização é usado juntamente com a matriz de escalonamento $\mathbf{E_d}$, como mostrado na expressão 2.8.

Tabela 2.5: Parâmentros e Passos de Quantização no padrão H.264.

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	...
Q_p	0.625	0.6875	0.8125	0.875	1	1.125	1.25	1.375	1.625	1.75	2	2.25	2.5	...
QP	...	18	...	24	...	30	...	36	...	42	...	48	...	51
Q_p	...	5	...	10	...	20	...	40	...	80	...	160	...	224

$$Z_{mn} = round\left(W_{mn} \cdot \frac{E_{mn}}{Q_p}\right). \tag{2.8}$$

Entretanto, visando simplificar a aritmética, o fator (E_{mn}/Q_p) é implementado como uma multiplicação por um fator MF e um deslocamento de *bits* à direita, evitando assim qualquer operação de divisão. Os valores de MF são apresentados na Tabela 2.6 de acordo com o valor do QP e as posições da matriz $\mathbf{E_d}$. Com isso, sendo $qbits = 15\,\lfloor QP/6 \rfloor$, calcula-se Z_{mn} por:

$$Z_{mn} = round\left(W_{mn} \cdot \frac{MF}{2^{qbits}}\right). \tag{2.9}$$

O processo de reconstrução do bloco é relativamente simples. A primeira parte seria a etapa de reescala, ou "quantização inversa" que, para evitar erros de arredondamento, tem um fator multiplicativo de 64 no cálculo dos coeficientes reescalonados W'_{mn}, como mostrado na equação 2.10, com os fatores multiplicativos PF sendo as entradas da matriz $\mathbf{E_i}$ (matriz de escala inversa).

$$W'_{mn} = Z_{mn} \cdot Q_p \cdot PF \cdot 64 \tag{2.10}$$

Em sequida, esses coeficientes passam pelo núcleo da transformada inversa $C_i^T W C_i$. Ao final,

Tabela 2.6: Multiplicador MF.

	Posições	Posições	Outras
QP	(0,0),(2,0),(2,2),(0,2)	(1,1),(1,3),(3,1),(3,3)	Posições
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

os valores devem ser divididos por 64 para remover o fator de escala. Tanto a multiplicação quanto a divisão por 64 são calculados por deslocamentos de *bits*, o que evita cálculos mais complexos. As matrizes E_i e C_i são dadas por:

$$E_i = \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \quad C_i = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & 1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix}$$

onde:

$$a = \tfrac{1}{2}, \quad b = \sqrt{\tfrac{2}{5}}$$

2.4 CODIFICADOR DE ENTROPIA

Entropia é a medida de incerteza de uma variável aleatória. Para uma variável aleatória X com função massa de probabilidade $p(x)$, a entropia é calculada pela expressão 2.11 e, neste caso (pelo uso do log_2), é dada em *bits*. Assim, quanto maior a entropia, mais a distribuição de X se aproxima da distribuição unifome.

$$H(X) = \sum_{x \in X} p(x) log_2 p(x) \tag{2.11}$$

A codificação de entropia é uma codificação sem perdas que se dá pela atribuição de códigos menores para as saídas mais frequentes de uma fonte de dados, enquanto que às saídas menos frequentes são atribuídos descritores necessariamente mais longos. Um exemplo disso é o código Morse. O que ocorre de fato nos codificadores de vídeo é a conversão de uma série de símbolos representantes da sequência de vídeo em uma *bitstream* comprimida adequada para transmissão ou armazenagem. Os símbolos de entrada do codificador de entropia podem incluir coeficientes transformados quantizados, vetores de movimento, marcadores (códigos que indicam pontos de ressincronização da sequência), cabeçalhos e informações suplementares. A codificação de comprimento variável é um tipo de codificação de entropia em que o número de *bits* de cada palavra-código é inteiro, ao contrário da codificação aritmética, que codifica grupos de símbolos por um número fracionário [19].

Uma vez criados os códigos $C(X)$ para a variável X, pode-se calcular o comprimento esperado $L(C)$ dos códigos $l(x)$, dado pela equação 2.12:

$$L(C) = \sum_{x \in X} p(x) l(x) \tag{2.12}$$

Este comprimento mantém uma relação muito importante com a entropia. Esta relação é dada pela inequação 2.13 e diz que o comprimento esperado de um código binário para uma variável aleatória X é sempre maior ou igual à entropia de X com igualdade se, e somente se, $2^{-l(x)} = p(x)$. Os códigos que minimizam o comprimento esperado são chamados códigos ótimos.

$$L \geq H(X) \tag{2.13}$$

Para exemplificar, tomemos a variável aleatória X, com as seguintes distribuições e atribuições de códigos:

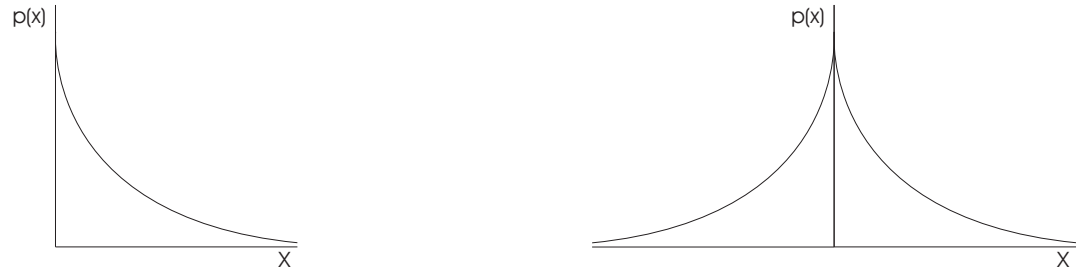
$$\begin{aligned} Pr(X = 1) &= \tfrac{1}{2}, \quad C(1) = 0 \\ Pr(X = 2) &= \tfrac{1}{4}, \quad C(2) = 10 \\ Pr(X = 3) &= \tfrac{1}{8}, \quad C(3) = 110 \\ Pr(X = 4) &= \tfrac{1}{8}, \quad C(4) = 111 \end{aligned}$$

Neste exemplo, tanto a entropia quanto o comprimento esperado têm valor de 1,75 *bits*. Agora,

se tomarmos uma variável aleatória Y com quatro símbolos assim como X, mas com distribuição uniforme e atribuição de códigos ótimos, os códigos serão todos de comprimento $C(y) = 2, \forall y \in Y$, ou seja, não existe nenhuma compressão efetiva.

2.4.1 Codificador Goulomb Exponencial

Os códigos de Goulomb Exponencial, ou Exp-Goulomb, usados no padrão H.264, são códigos de comprimento variável com uma construção regular. Ele é especialmente útil para codificar símbolos de um alfabeto com distribuições próximas da exponencial ou laplaciana (esboço na Figura 2.11), por causa da maior concentração próxima ao valor zero.



(a) (b)

Figura 2.11: Esboços de distribuições: (a) Exponencial; (b) Laplaciana.

No padrão H.264, símbolos provenintes de codificação preditiva (aquela usada para o cálculo dos vetores de movimento diferenciais) são codificados por Exp-Goulomb, devido à característica laplaciana. A forma geral desses códigos tem a seguinte estrutura:

$$[Mzeros][1][INFO]$$

O conjunto de *bits* formado pela sequência de M *bits* de valor 0 e um *bit* de valor 1 é chamado de prefixo. A sequência de *bits* INFO é chamada sufixo. O valor de M e a sequência INFO são dados pelas expressões abaixo. Note que M é nada menos que o comprimento de INFO.

$$M = \lfloor \log_2(simbolo + 1) \rfloor$$

$$INFO = simbolo + 1 - 2^M$$

A Tabela 2.7 mostra a forma geral dos códigos para diferentes intervalos de símbolos, enquanto que a Tabela 2.8 mostra sequências de *bits* para alguns símbolos. Para esta codificação,

é necessário que o símbolo seja maior ou igual a zero, ou seja, um número natural. Desta maneira, devem-se mapear os valores inteiros para valores naturais. A Tabela 2.9 mostra como isso é feito.

Tabela 2.7: Forma geral de códigos Exp-Goulomb.

Sequência de bits	Intervalos de símbolos
1	0
0 1 x_0	1-2
0 0 1 x_1 x_0	3-6
0 0 0 1 x_2 x_1 x_0	7-14
0 0 0 0 1 x_3 x_2 x_1 x_0	15-36
...	...

Tabela 2.8: Exemplos de códigos.

Sequência de bits	Símbolos
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8

2.4.2 Codificação de comprimento variável adaptativa baseada em contexto

Este método de codificação é usado para codificar coeficientes transformados de resíduos após a varredura em zig-zag. O CAVLC é projetado para tirar vantagem de diversas características de blocos 4x4 quantizados. A primeira vantagem é o uso de codificação por *run-level*. Após a varredura em zig-zag, em geral, os coeficientes não-zeros de maior magnitude são ± 1 e o CAVLC é capaz sinalizar a quantidade de ± 1 de maneira compacta. Outra característica dos

Tabela 2.9: Mapeamento para códigos de valores inteiros.

símbolo	valor a ser codificado
0	0
1	1
2	-1
3	2
4	-2
5	3
6	-4
k	$(-1)^{k+1} \lceil k \div 2 \rceil$

blocos é a correlação entre o número de coeficientes não-zeros de blocos vizinhos. O número de coeficientes não-zeros é codificado usando-se uma tabela de referência e a escolha da tabela depende da quantidade de coeficientes não-zeros dos blocos vizinhos. O CAVLC também se aproveita da magnitude dos coeficientes de blocos vizinhos para codificar os coeficientes do bloco atual.

2.5 QUALIDADE DA COMPRESSÃO

O processo de compressão pode ser de dois tipos: com ou sem perdas. Em uma codificação sem perdas, quando um vídeo passa pelo processo de codificação e decodificação, o vídeo original e o vídeo decodificado são exatamente iguais. Por outro lado, no caso da codificação com perdas, o vídeo decodificado tem sua qualidade deteriorada em relação ao original. Qual seria, então, o interesse em se ter uma codificação com perda de qualidade? Isto se dá pelo fato de que, quando se usa a codificação com perdas, geralmente se obtém uma maior redução da quantidade de *bits* representativos do vídeo e a perda de qualidade pode ser de um nível visualmente aceitável. Existe portanto uma solução de compromisso entre a qualidade, que pode ser tanto objetiva quanto subjetiva, e a taxa de compressão.

Os dois tipos de qualidade mencionados anteriormente se diferem pela forma de percepção. Determinar a qualidade subjetiva de um vídeo decodificado depende de diversos fatores, mas é

principalmente feita com o auxílio de sujeitos experimentais, pela maneira como os espectadores percebem o vídeo. Enquanto uma pessoa pode considerar a qualidade de um vídeo boa, outra pode considerar ruim. Os fatores que podem interferir na percepção incluem a luminosidade ambiente, níveis de cores, resolução, entre outros.

No caso da determinação da qualidade objetiva, usa-se uma métrica matemática comparativa entre os dois vídeos (original e decodificado). Essas métricas não medem exatamente a qualidade do vídeo decodificado, mas sim a distorção existente ente este e o vídeo original. Desta maneira, pode-se assumir que quanto menor a distorção, maior a qualidade objetiva do vídeo decodificado. A métrica mais utilizada e mencionada na literatura é a Relação Sinal-Ruído de Pico (PSNR, da sigla em inglês). Esta métrica se dá em escala logarítmica e depende do erro médio quadrático (MSE - *Mean Square Error*) entre duas sequências de vídeo com relação ao quadrado do maior valor de sinal possível do quadro ($(2^n - 1)^2$), sendo expressa pela equação 2.14, em que n é o número de *bits* por amostra do quadro.

$$PSNR_{db} = 10 \log_{10} \frac{(2^n - 1)^2}{MSE} \tag{2.14}$$

Normalmente, valores elevados de PSNR indicam qualidade elevada (baixa distorção), enquanto valores baixos indicam qualidade reduzida (alta distorção). Dois vídeos idênticos, por exemplo, teriam valor PSNR infinita, ou seja, não existe qualquer distorção entre eles. A Figura 2.12 mostra exemplos de curvas de PSNR para 300 quadros de quatro sequências no tamanho CIF. Graficamente, quanto mais à esquerda e mais acima está a curva, melhor. Quando se comparam diferentes condições de codificação de um vídeo (seja pela alteração de parâmetros do codificador ou pelo uso de codificadores distintos), são traçadas as curvas de PSNR para a mesma sequência de vídeo. Várias curvas comparativas serão apresentadas no Capítulo 4.

Apesar de ser a mais usada (basicamente por ser de fácil implementação), a PSNR é também bastante questionada, visto que depende de duas sequências representativas de um mesmo vídeo e não tem uma boa correlação com medidas subjetivas. Esta disparidade entre a PSNR e as medições subjetivas muitas vezes ocorre por uma questão de região de interesse dentro do vídeo. É possível que, em uma determinada região espacial de um trecho do vídeo, a codificação tenha tido algum erro, o que resultará em um baixo valor de PSNR. Entretanto, se essa região não for

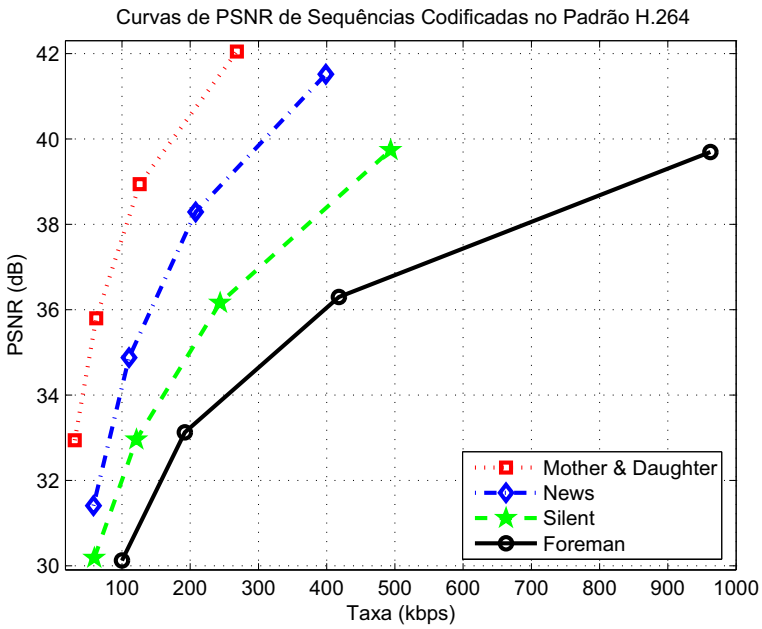


Figura 2.12: Exemplos de curvas de PSNR.

considerada como o foco da atenção do espectador, provavelmente aquele pedaço do vídeo pode ter sua qualidade subjetiva avaliada como boa.

Outro problema é o fato de que esta medição pode ser feita para toda a sequencia de vídeo ou quadro-a-quadro. Certas recomendações [20] sugerem que o cálculo da PSNR seja feito para cada quadro separadamente para então ser calculada a média total. Esta média é dada como a PSNR final da sequência. Ora, e se por um acaso, dentro de uma sequência inteira, um único quadro tiver reconstrução perfeita (distorção inexistente)? Como para este quadro a PSNR teria valor infinito, a média também seria infinita, o que não mostra a realidade sobre a qualidade da sequência por inteiro. Em outras aplicações [21], a qualidade de alguns quadros é bastante deteriorada em relação a outros. Neste caso, o cálculo da PSNR final por média mascara os quadros com maior distorção. Felizmente, para a maioria dos casos, a prática recomendada comentada acima serve bem o seu propósito.

Além da distorção, a solução de compromisso também deve levar em consideração a taxa de compressão, que indica a redução no tamanho representativo da informação. Esta medida é estritamente objetiva. Para se determinar a taxa de compressão de um processo de codificação, geralmente apontada em termos percentuais, é necessário se ter a taxa efetiva de *bits* do arquivo a ser codificado.

3 PARTIÇÕES ALTERNATIVAS DE MACROBLOCOS

Um dos avanços do Padrão H.264 foi a forma de particionar os macroblocos para a estimação e compensação de movimento. Entretanto, partições nas direções horizontal e vertical não compreendem todas as possibilidades de formas de contorno, mesmo com o uso de subpartições. Visando melhorar ainda mais essa característica do padrão, alguns estudos [22],[23] mostraram que é possível melhorar a codificação (diminuindo tanto taxa quanto distorção) a partir da partição de macroblocos por segmentos de retas com inclinação diferente de $0°$ (horizontal, modo 16x8) e $90°$ (vertical, modo 8x16).

Como este trabalho é continuidade de [1], estas partições serão chamadas partições *wedge* (tradução para cunha), em vez de partições GEO, usada em [23]. O termo *wedge* provém do termo *wedgelet* [24], como uma ferramenta matemática para representação de objetos. A partição *wedge* é uma restrição das *wedgelets* quando estas são descritas por coordenadas polares.

Assim como a predição dos vetores de movimento já existente no padrão depende das partições, tanto do macrobloco atual quanto dos seus vizinhos, para as partições *wedge* também devem ser obedecidas determinadas regras buscando o melhor aproveitamento da redundância da informação de movimento. Serão então apresentadas a geração dessas partições bem como as regras de implementação da predição de VM. Aproveitando as partições *wedge* e as regras de predição do VM, são apresentadas também as partições *one-wedge*. Em seguida, são propostos métodos de redução de complexidade, que incluem um levantamento estatístico de submodos *wedge*, definição do submodo *one-wedge* a partir de um mapa de diferenças e, por fim, partição por máscara binária arbitrária.

3.1 PARTIÇÕES WEDGE

A partição *wedge* divide o macrobloco por um segmento de reta de direção arbitrária. Este segmento tem sua posição e orientação determinados por uma distância r (dada em *pixels*) do centro do macrobloco e por um ângulo θ (dado em graus). Estes raio e ângulo são chamados parâmetros da *wedge*. Basicamente, trata-se de um segmento tangente à circunferência de raio r, com posição determinada pelo ângulo θ.

A Figura 3.1 ilustra a partição de forma contínua e discreta. O segmento de reta cheio mostra onde deveria ser dividido o macrobloco para o caso contínuo. Para o caso discreto, observam-se as duas partições I (Interna) e E (Externa), com preenchimento claro e escuro, respectivamente, classificadas de acordo com o centro do macrobloco. Note que o termo "partição" é usado para definir tanto a divisão do macrobloco como cada uma das partes resultantes da divisão.

Assim como no padrão H.264 as partições 8x4, 4x8 e 4x4 são chamadas submodos, este mesmo termo é usado para fazer referência a qualquer uma das partições geradas por um par de parâmetros: raio e ângulo. Por exemplo, o submodo $(2; 45°)$ é a partição *wedge* gerada pelo raio $r = 2$ *pixels* e ângulo $\theta = 45°$.

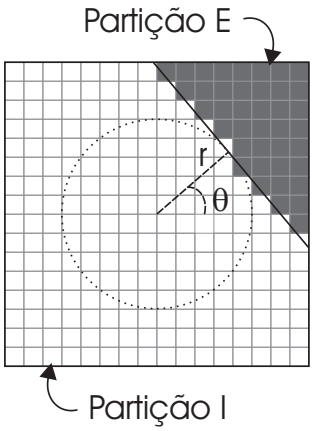


Figura 3.1: Partições *wedge*.

Para determinar o melhor modo para o macrobloco, o padrão H.264 testa todos os modos. Para o modo *wedge*, devem ser feitos testes para várias partições possíveis. Uma vez geradas as partições *wedge*, a estimação de movimento é feita de maneira semelhante àquelas dos modos 16x8 e 8x16, ou seja, a busca é feita para cada partição independentemente da outra. Isto é feito por meio de uma máscara binária de tamanho 16x16 *pixels* para cada partição (sendo uma máscara

inversa da outra). Em seguida, a cada partição será relacionado um VM e um índice de quadro de referência. A Figura 3.2 mostra a extensão da árvore de compensação do padrão H.264 com a inclusão do modo *wedge*.

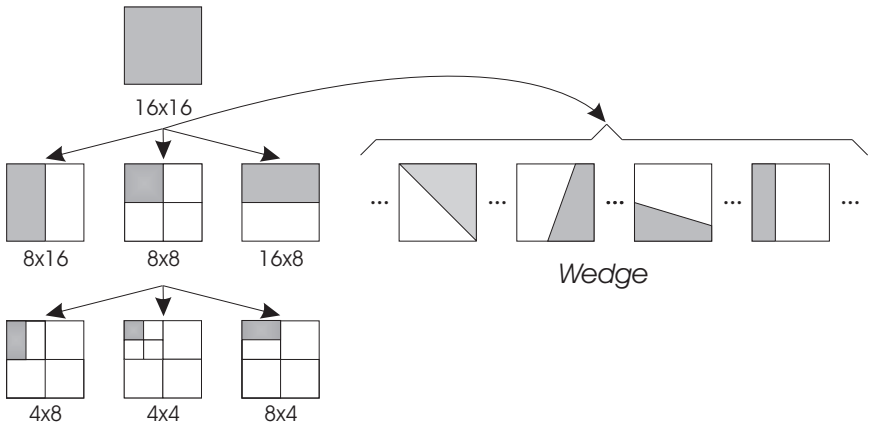


Figura 3.2: Estrutura em Árvore do H.264 extendida.

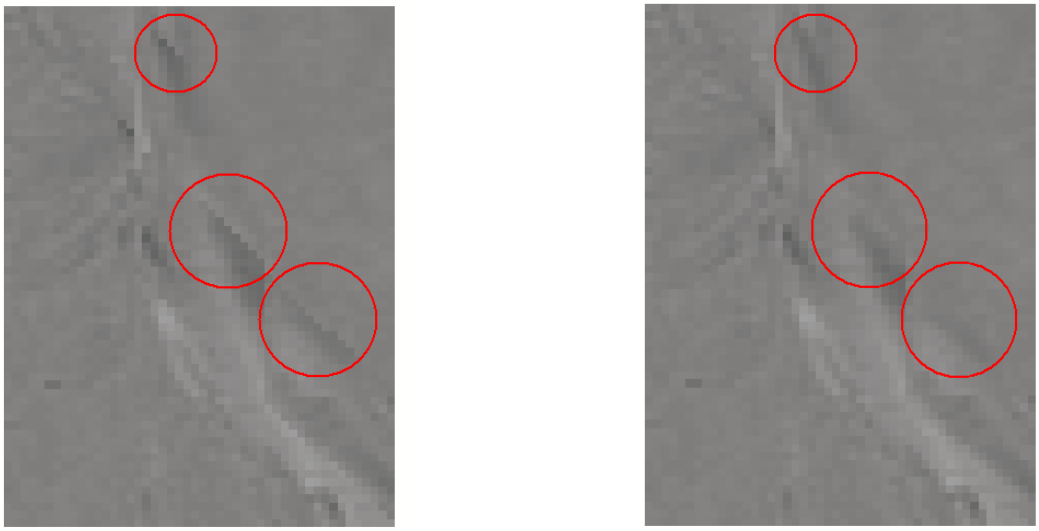
É importante ressaltar que existe uma diferença entre o processo de estimação de movimento para os modos bi-particionados do padrão e o modo *wedge*. Para os modos do padrão, trabalha-se com cada metade do MB de forma que a quantidade de operações gastas é a mesma para um MB no modo 16x16. Já para o modo *wedge*, os cálculos são feitos para o MB inteiro e só então a máscara binária é aplicada. Como isto é feito para cada uma das partições *wedge*, tal procedimento equivale ao processamento próximo àquele gasto por dois MBs no modo 16x16. Isto quer dizer que a quantidade de operações realizadas para cada partição é a soma da quantidade de operações de um MB no modo 16x16 mais as operações de multiplicação pela máscara binária. A estimação de movimento para o modo *wedge* pode ser mais bem entendida pelo seguinte pseudo algoritmo, sendo que alguns passos serão explicados e detalhados mais adiante:

Para cada possível par de parâmetros (que define o submodo) ocorrem os seguintes passos:

- Geração da máscara binária a partir dos parâmetros da *wedge*.
- A partir dos macroblocos vizinhos, é encontrado o VMpred.
- O VMpred define a região onde devem ocorrer as buscas.
- Para a partição Interna, usa-se a máscara gerada; caso contrário (partição Externa), é usada a máscara é invertida.

- Durante a busca, quando é feito o cálculo da distorção, a máscara binária é aplicada ao MB;

Depois da estimação de movimento, deve ser feita a compensação de movimento. Semelhantemente ao que ocorre nos demais modos, existe uma quebra na borda das partições, porém isto não é exatamente um problema para os modos já existentes no padrão. Como a transformada DCT usada é de tamanho 4x4 ou 8x8, a alta frequência espacial criada pela quebra dos blocos não aparece no domínio transformado. Infelizmente, a mesma coisa não ocorre para as partições *wedge*, como pode ser visto no resíduo mostrado na Figura 3.3(a). Para tal, faz-se necessário uma suavização na borda das partições durante o processo de compensação de movimento, mostrado na Figura 3.3(b). Percebe-se claramente esta diferença nas regiões destacadas.



(a) (b)

Figura 3.3: Exemplo de resíduo da sequência *Foreman*: (a) normal; (b) suavizado.

O que ocorre de fato é uma ponderação das duas máscaras binárias ao longo da reta de partição. Os valores da máscara suavizada variam de 0 a 1 e esta ponderação deve ocorrer de forma que a soma das duas máscaras resulte em uma máscara de 16x16 *pixels* preenchida por 1's. Essa ponderação se dá por uma filtragem espacial do tipo passa-baixas em que a amostra atual da máscara é a média das suas 8 vizinhas e a máscara M do filtro é dada por 3.1. Esta filtragem permite inclusive que a implementação seja feita apenas por somas e deslocamentos de *bits*. As Figuras 3.4(a) e 3.4(b) mostram exemplos de máscaras binária e suavizada, respectivamente.

$$M = \frac{1}{8}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.1}$$

(a) (b)

Figura 3.4: Exemplo de máscara: (a) binária; (b) suavizada.

Após a estimação e compensação de movimento, é feita a decisão sobre a escolha do modo de partição. Caso o modo *wedge* seja o escolhido por ter o menor custo, os resíduos dos macroblocos são codificados exatamente da mesma maneira que é feita para os demais modos.

3.2 PREDIÇÃO DO VETOR DE MOVIMENTO

Uma vez criada a partição, é possível fazer as predições dos vetores de movimento. Visto que a predição do vetor de movimento parte do pressuposto de que os vetores de movimento de blocos próximos tendem a ser semelhantes (seguindo movimento de objetos ou da câmera), foram determinadas regras sobre esta predição para sete casos, de acordo com o ângulo θ da *wedge*. Entretanto, buscou-se manter, na medida do possível, a predição coerente àquela do padrão H.264 (para as partições verticais e horizontais), mostrada no Capítulo 2. Tomou-se por base também a predição para modos alternativos propostos para o padrão H.26L [3], mostrados na Figura 3.5, em que cada cor representa uma partição distinta. Observa-se nesta figura os blocos de tamanho 4x4 (cada bloco é mostrado como um quadrado sólido) de onde o vetor de movimento é obtido, externos ao macrobloco atual, que está representado em cor rosa ou rosa e verde. Isto se dá por uma questão de organização, uma vez que todos os sub-blocos 4x4 pertencentes a uma

mesma partição têm o mesmo vetor de movimento. Os casos que mostram 3 sub-blocos para uma mesma partição (mesma cor) significam que a predição é feita por mediana. Nos exemplos em que os blocos são sobrepostos, quer dizer que para uma partição é usada mediana e para a outra o vetor de movimento do próprio bloco.

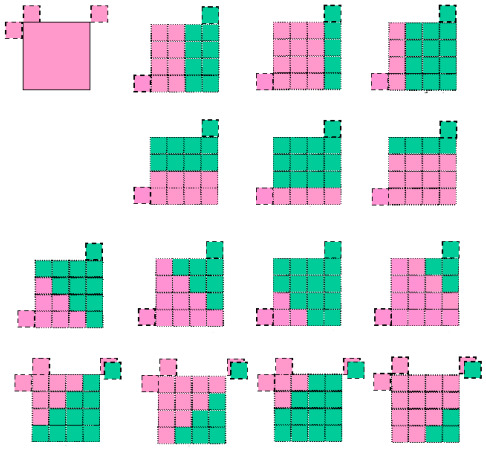


Figura 3.5: Modos propostos para o padrão H.26L (adaptado de [3]).

Essas regras foram definidas para intervalos de ângulos de forma que se mantivesse a contiguidade da partição com o sub-bloco 4x4 de onde o vetor de movimento é adquirido. Isto implica que, em alguns casos, se a predição for feita do bloco à esquerda (bloco A), deverá ser usada a partição mais inferior ou a partição mais à esquerda, no caso de o bloco usado ser o superior (bloco B), ao contrário da regra do padrão H.264. A Tabela 3.1 apresenta de qual bloco (A, B, C ou Mediana) deve ser predito o VM para as partições I e E, de acordo com um intervalo do ângulo θ. A Figura 3.6 mostra exemplos dos macroblocos atuais e os sub-blocos 4x4 vizinhos para cada faixa de ângulo. Outra questão que deve ser levada em consideração é o fato de que os vetores de movimento podem ser preditos de macroblocos *wedge*. Para esses casos, o vetor de movimento deve ser proveniente da partição que tem maior sobreposição com o sub-bloco 4x4 vizinho.

Um fato importante de se ressaltar a respeito das partições *wedge* é o aumento da quantidade de informação a ser codificada. Por meio desse modo alternativo de partição, busca-se melhorar a codificação pela redução da energia dos resíduos. Entretanto, para que a decodificação ocorra corretamente, os parâmetros das *wedges* escolhidos para cada macrobloco (caso o modo *wedge* seja o melhor) devem ser enviados de alguma forma ao decodificador. No campo de estudo de

Tabela 3.1: Regras para Predição do VM.

Casos	Faixa de Ângulo	Predição para Partição I	Predição para Partição E
Caso 1	$0° \leq \theta < 45°$	B (esquerda)	A (baixo)
Caso 2	$45° \leq \theta < 90°$	C	A (baixo)
Caso 3	$90° \leq \theta < 135°$	C	A (topo)
Caso 4	$135° \leq \theta < 180°$	A (baixo)	B (esquerda)
Caso 5	$180° \leq \theta < 225°$	A (baixo)	B
Caso 6	$225° \leq \theta < 270°$	A (topo)	C
Caso 7	$270° \leq \theta < 360°$	A (topo)	Mediana

codificação, essa informação extra é comumente chamada de *overhead*. Claramente, espera-se que a redução do resíduo seja suficientemente boa para compensar pela inserção desse *overhead*. Assim como toda informação que se deseja enviar ao decodificador, é interessante que os parâmetros da *wedge* também seja codificados de alguma forma. Mais detalhes sobre isso serão apresentados no Capítulo 4.

3.3 PARTIÇÕES *ONE-WEDGE*

A ideia principal do modo *one-wedge* é que a estimação de movimento seja feita apenas para uma das partições *wedge* e, desta forma, seria encontrado apenas um VMest e um VMd para o macrobloco inteiro. A motivação para o uso deste método de partição foi buscar aproveitar a característica de alguns tipos de vídeos que têm tanto a câmera quanto uma região de fundo estáticas. Assim, a divisão do macrobloco deve ocorrer nas bordas de objetos que se movem na cena.

A partição para a qual é feita a busca é chamada dinâmica, enquanto a outra é chamada estática. Exceto pelo VMpred, que é o mesmo para ambas as partições, quase tudo relacionado à partição dinâmica se comporta exatamente como para as partições no modo *wedge* original. O VM predito segue a regra de predição para as partições *wedge*. Como para a partição estática não é feita busca, associa-se a ela VMd de valor zero. Quanto ao quadro de referência, como o VMpred é o mesmo para ambas as partições, ele deve ser também o mesmo para ambas.

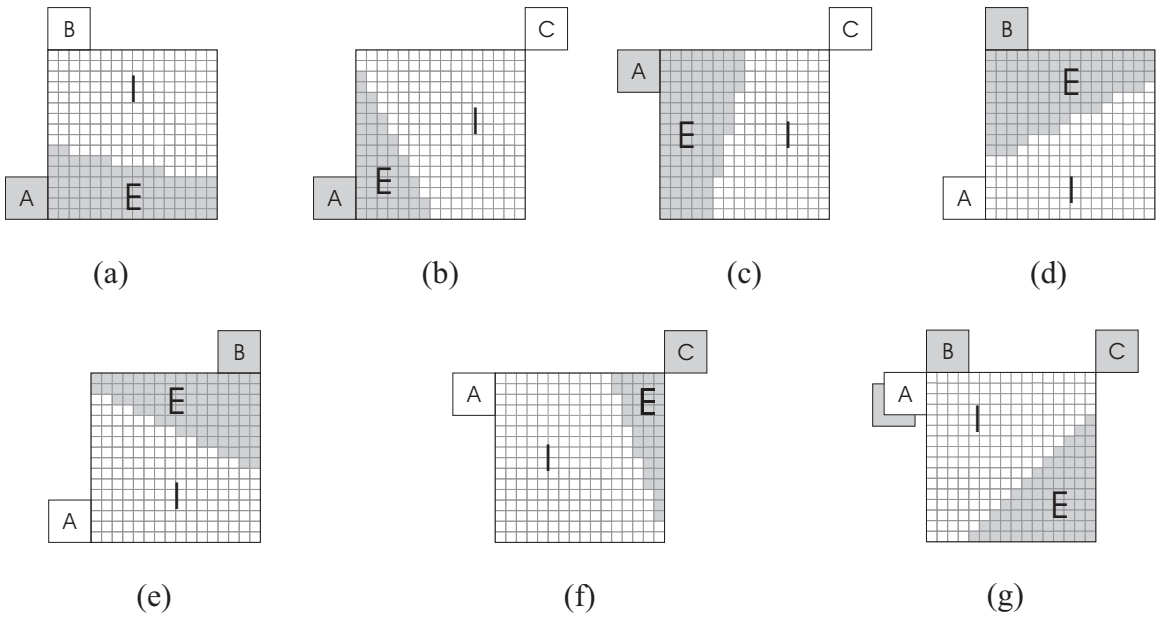


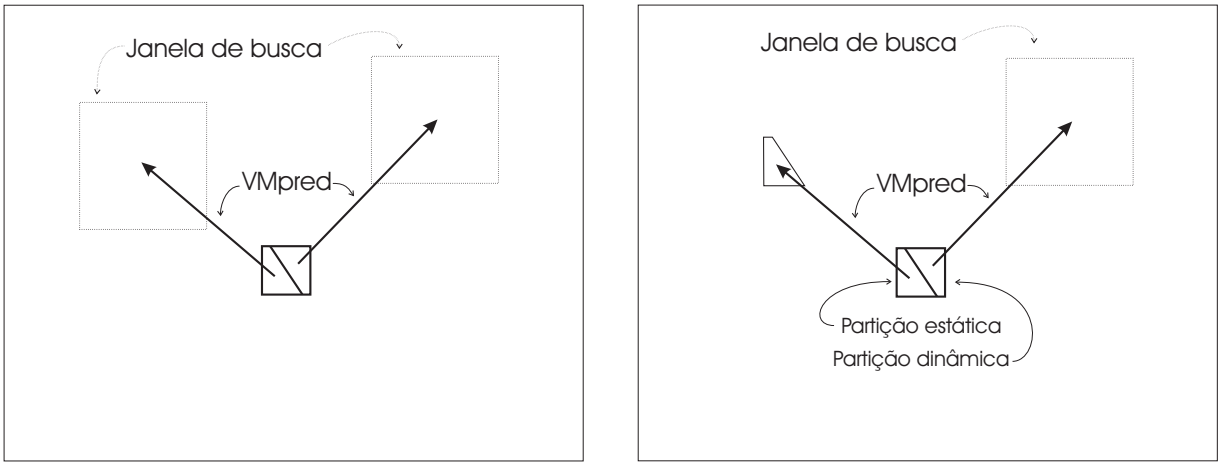
Figura 3.6: Exemplos de predição: (a) Caso 1; (b) Caso 2; (c) Caso 3; (d) Caso 4; (e) Caso 5; (f) Caso 6; (g) Caso 7.

Em resumo, as etapas de estimação e compensação de movimento para o modo *one-wedge* seguem os seguintes passos para cada submodo da *wedge*:

- A partir dos macroblocos vizinhos, é encontrado o VMpred seguindo as regras de predição para o modo *wedge*.
- No passo de estimação de movimento, é feita a busca apenas para a partição dinâmica e com isso é encontrado o seu VMd associado. Tanto a partição interna quanto a externa podem ser a partição dinâmica.
- Para a partição estática, tem-se $VMd = 0$.
- No passo de compensação de movimento, o bloco compensado possui partições pertencentes ao mesmo quadro de referência obrigatoriamente, sendo montado com o VMpred para a partição estática e o VMest para a partição dinâmica.

Por fim, o que se tem é a diminuição nas informações a serem codificadas. Como o VMd associado à partição estática é zero, ele não precisa ser codificado. Além disso, também há dispensa de codificar dois índices de quadros de referência para as duas partições. As Figuras 3.3(a) e 3.3(b) mostram a ideia geral para as buscas nos modos *wedge* e *one-wedge*,

respectivamente. As figuras mostram um macrobloco particionado com seus VMpreds associados apontando para a região onde é feita a estimação de movimento. Claramente, esta região pertence a um quadro de referência, porém estamos apenas ilustrando a questão espacial.



(a) (b)

Figura 3.7: Buscas nos modos: (a)*wedge*; (b)*one-wedge*.

3.4 REDUÇÃO DE COMPLEXIDADE

Os trabalhos anteriores [22],[23] mostraram que a principal desvantagem das partições *wedge* é o custo computacional pelo elevado número de buscas na etapa de estimação de movimento para todos os parâmetros da *wedge*. Visando reduzir a quantidade de buscas a serem feitas, e consequentemente a complexidade computacional, são propostos métodos aplicados tanto às partições *wedge*, quanto às *one-wedge*.

3.4.1 Análise estatística de submodos *wedge*

Durante a estimação de movimento do modo *wedge*, a escolha dos melhores parâmetros se dá baseada naqueles que geram um menor resíduo dentre todos os parâmetros possíveis. Uma maneira de se reduzir a complexidade seria diminuir a quantidade de buscas a partir da diminuição na quantidade de partições testadas. Contudo, esta redução pode ser feita de diversas maneiras. Uma delas seria simplesmente, a partir de um número total de partições possíveis, escolher os parâmetros aleatoriamente. Outra forma seria escolher parâmetros pré-definidos.

Decidiu-se por fazer uma análise estatística dos modos mais prováveis. A Figura 3.8 mostra o histograma conjunto dos parâmetros raio e ângulo para os 300 quadros da sequência *Mobile*, com 256 possíveis pares de parâmetros. Como as sequências de vídeos têm características diferentes, é de se esperar que essa estatística varie de vídeo para vídeo. Então, uma partição qualquer que esteja entre as mais escolhidas para um vídeo, não necessariamente é boa o suficiente para outro. Porém, esse levantamento é feito para os melhores parâmetros para cada macrobloco, o que não nos dá qualquer informação sobre o segundo (ou terceiro etc.) melhor conjunto de parâmetros. A ideia é que, mesmo que os parâmetros usados não sejam os melhores para uma determinada sequência, se eles forem bons o suficiente, teremos ganhos sobre os modos do padrão, inclusive pela redução do *overhead* por macrobloco.

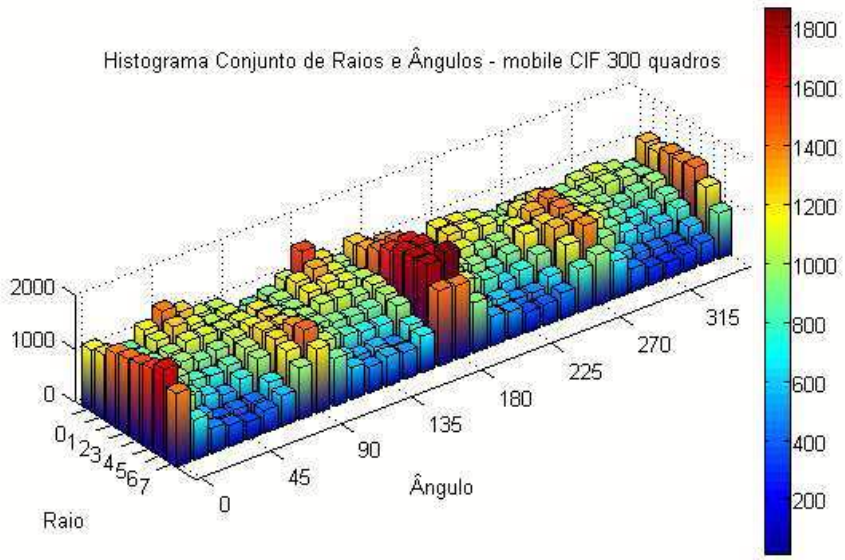
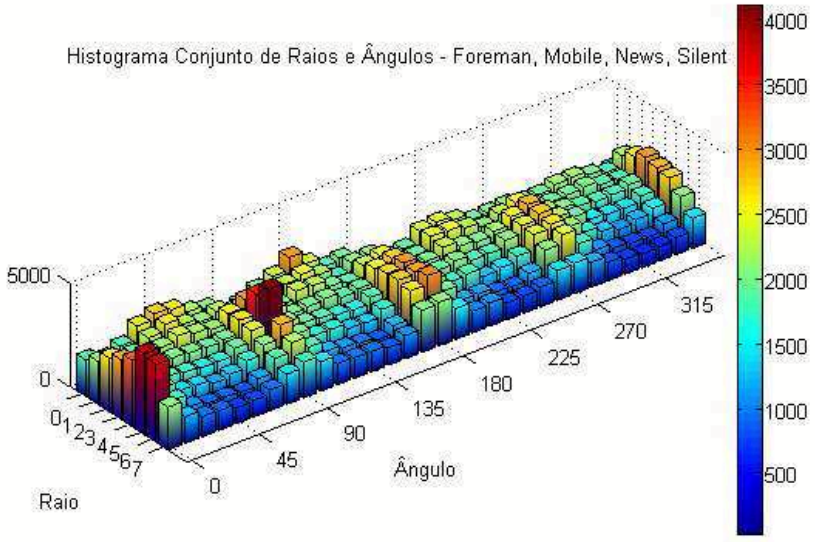
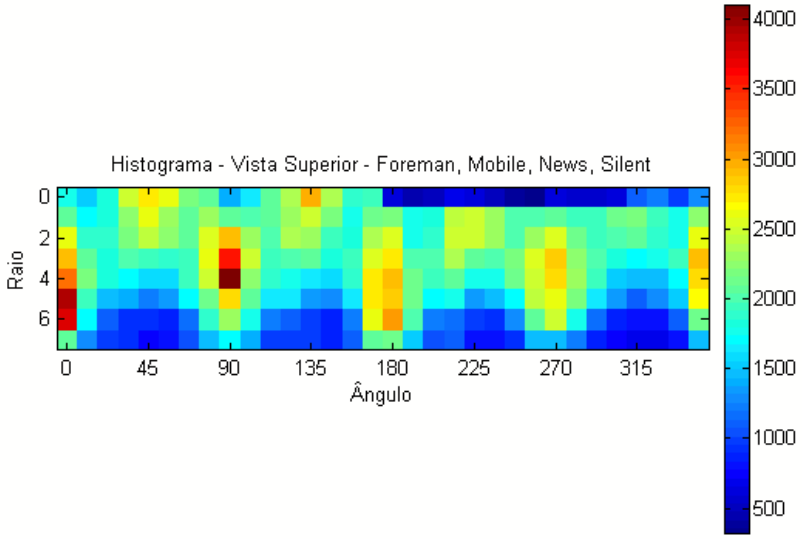


Figura 3.8: Histograma 3D de parâmetros da *wedge* da sequência *Mobile*.

É claro que seria vantajoso para ganhos em taxa e distorção se fizéssemos a codificação de um determinado vídeo uma vez e, a partir das estatísticas adquiridas, o codificássemos novamente. Contudo, isso iria contra tudo aquilo que buscamos do ponto de vista de redução de complexidade. O que se escolheu fazer foi testar os ganhos de codificação para parâmetros adquiridos de um grupo de vídeos, usado como base de dados. Somaram-se então os histogramas conjuntos de quatro sequências (*Foreman, Mobile, News, Silent*), mostrado na Figura 3.9. A tabela 3.4.1 mostra a quantidade total de todos os submodos para as quatro sequências (*Foreman, Mobile, News* e *Silent*) escolhidas para a base dados.



(a)



(b)

Figura 3.9: Histograma conjunto da base de dados para análise estatística: (a) Vista em perspectiva; (b) Vista superior.

A partir dos dados do histograma conjunto usado como base de dados, são tomados aqueles pares de parâmetros mais prováveis. Não existe garantias de que os submodos mais prováveis para a base de dados e para uma sequência qualquer sejam os mesmos, devido às diferentes características de cada vídeo, como já foi comentado. Entretanto, nota-se que a intersecção é

Tabela 3.2: Estatística dos submodos da base de dados.

Parâmetros r e θ	0	1	2	3	4	5	6	7
$0°$	1771	2084	2592	2887	3221	3937	3775	2075
$11,25°$	1519	1690	1876	2072	2195	1946	1727	1289
$22,5°$	1806	1804	1865	1822	1798	1475	1102	976
$33,75°$	2473	2271	2162	2022	1682	1422	933	951
$45°$	2749	2615	2411	1945	1595	1211	909	769
$56,25°$	2613	2292	2219	1895	1580	1338	846	810
$67,5°$	2212	2090	2085	1966	1925	1680	1189	1019
$78,75°$	2052	2288	2550	2599	2471	2127	1863	1464
$90°$	1397	2053	2921	3544	4137	2793	2310	1731
$101,25°$	1666	1997	2318	2469	2223	2052	1782	1429
$112,5°$	2048	2070	1999	1944	1882	1645	1248	1015
$123,75°$	2361	2328	2376	1994	1787	1631	1102	983
$135°$	2962	2500	2321	2068	1629	1341	1004	981
$146,25°$	2345	2167	1973	1817	1564	1295	856	844
$157,5°$	1861	1737	1824	1918	1864	1655	1240	1117
$168,75°$	1966	2124	2365	2576	2708	2756	2609	2037
$180°$	601	2171	2524	2751	2912	2848	3025	2107
$191,25°$	470	1768	1760	1995	2147	2128	1984	1496
$202,5°$	491	1848	1889	1986	1924	1709	1213	1044
$213,75°$	713	2410	2511	2353	2053	1782	1193	1126
$225°$	639	2508	2465	2091	1768	1355	965	815
$236,25°$	483	2302	2278	1978	1776	1455	910	824
$247,5°$	376	1978	2017	2173	2119	1711	1256	952
$258,75°$	295	2019	2321	2568	2516	2446	2070	1444
$270°$	610	2080	2557	2842	2773	2625	2498	1478
$281,25°$	563	1941	2112	2222	2281	2178	1745	1204
$292,5°$	568	1952	1944	2013	1975	1630	1136	891
$303,75°$	610	2134	1995	1815	1549	1339	885	729
$315°$	1120	2168	2063	1715	1471	1107	804	669
$326,25°$	1213	1968	1857	1739	1458	1266	784	707
$337,5°$	994	1763	1781	1917	1734	1514	1127	827
$348,75°$	1304	2252	2603	2923	2788	2673	2133	1455

suficientemente alta. Por exemplo, sejam o conjunto A os 128 submodos mais prováveis (metade do total) da base de dados e o conjunto B os 128 mais prováveis da sequência *Mobile*. A intersecção entre esses dois conjuntos é de aproximadamente 88% de B. A Figura 3.10 mostra em branco justamente essa intersecção, enquanto que em cinza vêem-se os demais submodos de A. O que se espera é que mesmo esses modos mostrados em cinza sejam capazes de produzir algum ganho.

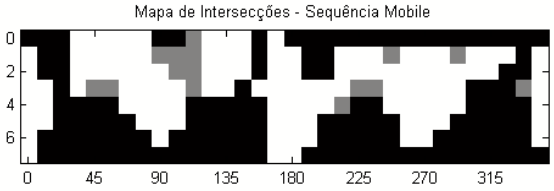


Figura 3.10: Submodos mais prováveis.

Além da redução pela metade, pode-se também diminuir ainda mais o conjunto de submodos usados na busca. É esperado que a redução na quantidade de submodos diminua também os ganhos sobre o padrão H.264. Contudo, o que se busca verificar é de quanto é essa redução, pois acredita-se que uma redução pela metade na quantidade de modos não implica uma redução de ganho pela mesma razão. Em termos gráficos, espera-se que as curvas de PSNR para as buscas reduzidas sejam bastante próximas da curva referente à busca com todos os modos.

3.4.2 Método rápido baseado em mapa de diferenças para modo *One-Wedge*

Visando melhorar o processo de busca do modo *one-wedge*, pensou-se em uma maneira de acelerá-lo. O questionamento levantado foi se haveria alguma maneira de, para cada macrobloco, não ser preciso testar todos os submodos *one-wedge* possíveis. Seria bastante vantajoso se fosse possível determinar o melhor submodo *one-wedge* (ou pelo menos bom o suficiente de forma que o modo *One-Wedge* seja escolhido) a partir de algum tipo de informação extraída do macrobloco. A maneira escolhida para a determinação do submodo foi o uso de um mapa de diferença entre o macrobloco atual e o macrobloco apontado pelo VMpred.

A diferença entre os dois macroblocos pode ser limiarizada de forma a se criar um mapa de diferenças. Para as regiões em que a diferença é pequena (critério arbitrário que pode ir até mais

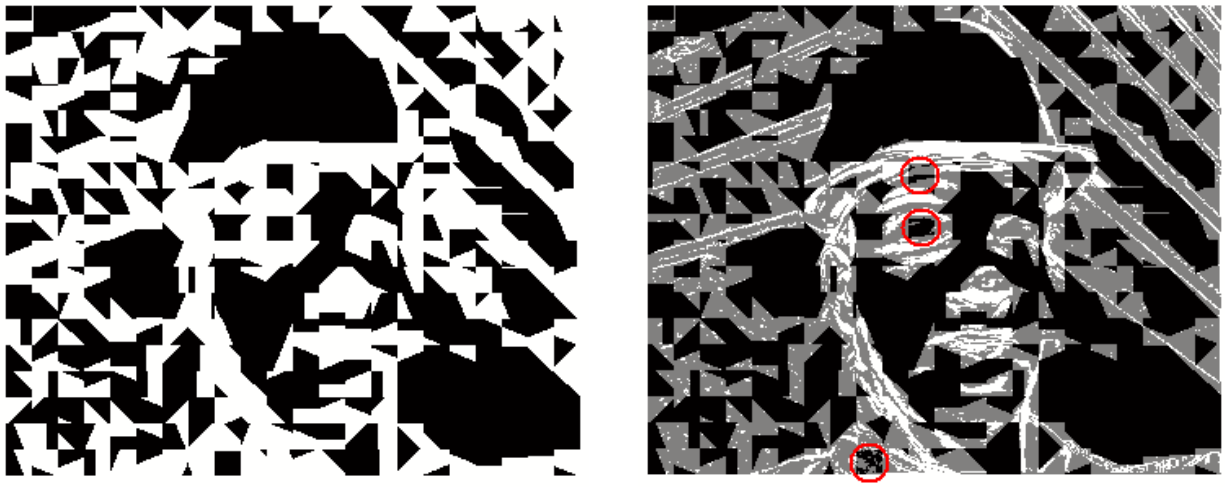
ou menos 10 níveis de cinza), pode-se considerar que não existe movimento e arbitrar diferença 0. É importante se levar em consideração a influência da quantização já aplicada ao quadro de referência, o que acaba por resultar em um pequeno resíduo quando se calcula a diferença entre os MBs. Já na região onde a diferença é maior, considera-se necessário buscar uma melhor referência para o cálculo do resíduo final e arbitra-se diferença 1. Por fim, o mapa de diferenças nada mais é que uma máscara binária. A Figura 3.11 mostra um mapa de diferenças criado entre os dois primeiros quadros da sequência *Foreman*.

Figura 3.11: Mapa de diferenças.

Com o mapa de diferenças, pode-se gerar a partição *one-wedge*. Visando ao melhor aproveitamento do mapa, dentre todas as partições possíveis, optou-se por estabelecer a partição estática como aquela que tem maior área pertencente à região de diferença zero. A Figura 3.12(a) mostra as partições *one-wedge* geradas a partir do mapa de diferenças, sendo as partições dinâmicas mostradas em cor branca. Já a Figura 3.12(b) mostra a sobreposição do mapa de diferenças com as partições. Observa-se que para os três macroblocos destacados não existe partição *one-wedge* que satisfaça a condição de sobreposição apresentada acima.

3.4.3 Partição por máscara binária arbitrária

Outra forma alternativa de partição de macroblocos é possível. A partir dos mapas de diferenças criados com a finalidade de determinar partições *one-wedge* rapidamente, surgiu a ideia de se usar máscaras binárias arbitrárias, em vez das formas bem definidas das *wedge*. As



(a) (b)

Figura 3.12: Partições *one-wedge*: (a) partições rápidas; (b) sobreposição com mapa de diferenças.

máscaras binárias usadas na estimação e compensação de movimento seriam os próprios mapas de diferença.

Assim como para o caso das partições *wedge*, para as partições arbitrárias também existe a necessidade de se conhecer as máscaras do lado do decodificador. Como as máscaras são dependentes de cada macrobloco e das referências, em princípio todas as 2^{256} máscaras possíveis podem vir a ser utilizadas. Este valor torna inviável qualquer indexação para todas as máscaras dentro de uma mesma lista. Neste caso, a melhor opção seria codificar a própria máscara e o codificador escolhido para a codificação das máscaras foi o JBIG, um padrão de codificação sem perdas de imagens binárias [15]. Para aproveitar melhor o codificador JBIG, as máscaras para cada macrobloco devem ser agrupadas de forma a gerar uma única máscara binária, pois o JBIG aproveita as vizinhanças de cada *pixel*. Por fim, um único arquivo comprimido é enviado ao decodificador separadamente da *bitstream*.

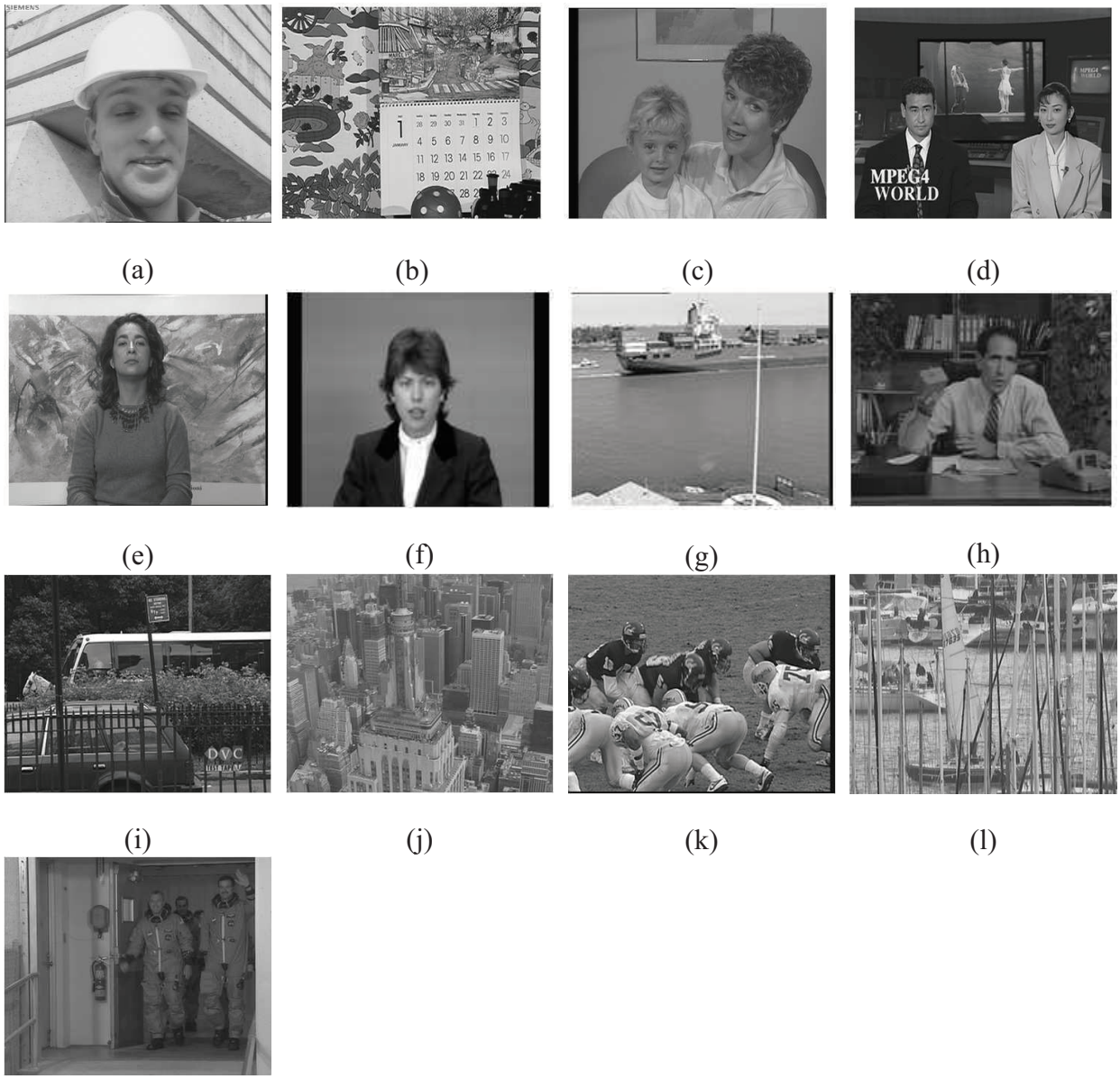
Um dos principais problemas dessa ideia é o fato de que, mesmo codificada, a máscara binária deve representar um *overhead* bastante elevado. Desta forma, assim como para as partições *wedge*, o ganho deve ser suficientemente alto para compensar este acréscimo de informação. Outra dificuldade é a estimação da taxa de cada macrobloco devido à maneira como as máscaras são codificadas em conjunto.

4 IMPLEMENTAÇÕES E RESULTADOS

A implementação das partições alternativas no padrão H.264 se deu por meio do código JSVM (*Joint Scalable Video Model*) versão 9.2, escrito em linguagem C++ [25]. O JSVM é um código de referência do padrão H.264/SVC, uma extensão do padrão H.264/AVC para codificação de vídeo escalonável. Apesar da existência do código JM para o padrão H.264/AVC, optou-se por usar o JSVM pela sua organização, pelo uso de orientação a objetos e pela possibilidade de futuras implementações para codificação escalonável. Devido à semelhança nos trabalhos, as informações apresentadas por [23], [26] foram usadas como base para nossas implementações.

Durante a implementação buscou-se o mínimo de intervenção possível no código, visando ao melhor aproveitamento de funções e classes preexistentes. Com isso, o novo modo foi implementado mantendo-se a estrutura do código usada para a implementação dos demais modos. Assim como nos modos 16x8 e 8x16, o modo *wedge* possui duas partições: interna e externa. Isso permitiu que se fizesse apenas algumas adaptações nas funções ligadas à estimação e compensação de movimento desses modos.

Para a validação do trabalho, foram necessários exaustivos testes. Optou-se por usar como sequências de teste as sequências *Foreman*, *Mobile*, *Mother & Daughter*, *News* e *Silent* de tamanho CIF (resolução 352x288 *pixels*), *Claire*, *Container* e *Salesman* de tamanho QCIF (resolução 176x144 *pixels*). Para a validação do método rápido pela análise estatística, foram usadas também as sequências *Bus*, *City*, *Football* e *Harbour*, de tamanho CIF e *Crew* de tamanho 4CIF (resolução 704x576 *pixels*). Todas essas sequências têm taxa de quadros de 30 Hz e quase todas tinham 300 quadros, exceto pelas sequências *Bus* (possui 150 quadros), *Football* (possui 260 quadros) e *Crew* (foram usados apenas 60 dos 300 quadros). A Figura 4.1 mostra o primeiro quadro de cada uma dessas sequências.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m)

Figura 4.1: Sequências de Teste: (a) *Foreman*; (b) *Mobile*; (c) *Mother & Daughter*; (d) *News*; (e) *Silent*; (f) *Claire*; (g) *Container*; (h) *Salesman*; (i) *Bus*; (j) *City* (k) *Football*; (l) *Harbour*; (m) *Crew*.

Todos os testes foram executados com todos os quadros (exceto o primeiro) sendo preditos por predição do tipo P. Esta estrutura de predição é conhecida por IPPP. Seguindo a recomendação VGEG-AE010 [20], foram usados quatro quadros de referência para as predições; os valores de QP usados foram 22, 27, 32 e 37 para o primeiro quadro (tipo I) e 23, 28, 33 e 38 para os demais quadros (tipo P); a janela de busca da estimação de movimento é de 32x32 *pixels*.

Os resultados são mostrados por curvas de PSNR e tabelas, e esses resultados são todos obtidos pelas inserções no código JSVM. Desta forma, quando são apresentados gráficos para o modo *wedge*, por exemplo, quer dizer que o modo *wedge* foi acrescido aos modos existentes no padrão H.264 mas não houve uma substituição. As diferenças objetivas entre as curvas, tanto em taxa quanto em distorção (medida por PSNR), são calculadas pelo método de medidas de Bjøntegaard [27].

4.1 PARÂMETRO *MB_TYPE* PARA O MODO *WEDGE*

Quando é feita a escolha por um determinado modo de partição, essa informação deve ser codificada e enviada ao decodificador. Essa informação é chamada de *mb_type*, ou seja, o tipo (ou modo) do macrobloco. A Tabela 4.1 mostra os *mb_types* para os modos para predição do tipo P. Uma das primeiras alterações e também uma das mais importantes foi definir um valor para o *mb_type* relativo ao modo *wedge*. Esta decisão se dá pelo fato de que este parâmetro também deve ser codificado por um codificador de entropia.

Tabela 4.1: *mb_types*.

mb_type	Modo
0	16x16
1	16x8
2	8x16
3	8x8

Inicialmente, optou-se por associar ao modo *wedge* o valor 3, posicionando-o entre os modos 8x16 e 8x8 [23]. Porém, verificou-se que atribuir o valor 1 (posicionado logo após o modo 16x16) é mais vantajoso. Isto se deve ao fato de que este parâmetro é codificado com codificador Exp-

Goulomb, ou seja, quanto mais frequente o modo, menor deve ser o valor do parâmetro. A partir de 60 quadros (equivalente a 2 segundos) das sequências de testes de tamanho CIF, foi feita a comparação para os dois casos. A Figura 4.2 mostra as curvas referentes aos diferentes valores de *mb_type* para o modo *wedge* em comparação com padrão H.264 para a sequência *Mother & Daugther*.

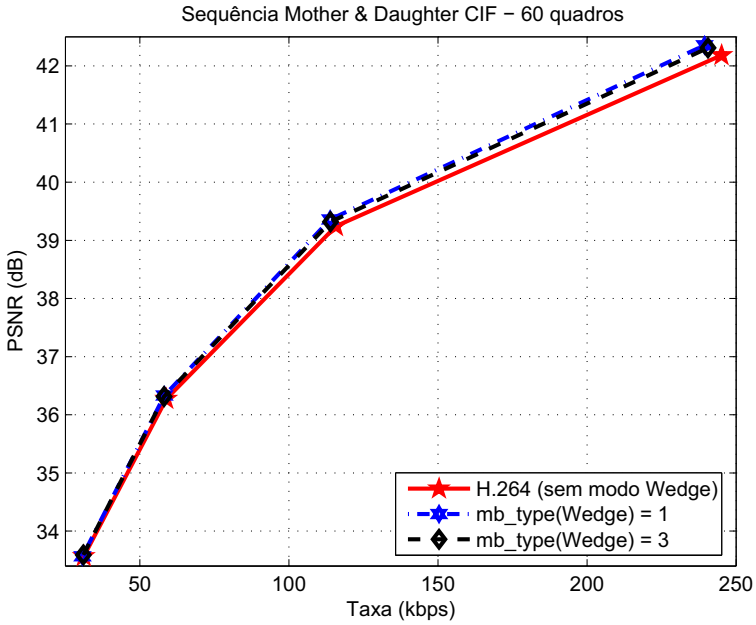


Figura 4.2: Curvas de PSNR para diferentes valores de *mb_type*.

Em média, a atribuição do valor *mb_type* $= 1$ teve uma redução de taxa de $0,776\%$ sobre o outro caso. A Tabela 4.2 mostra a redução percentual de taxa para cada sequência.

Tabela 4.2: Ganhos percentuais comparativos para valores do *mb_type* do modo *wedge*.

Sequências:	*Foreman*	*Mobile*	*Mother & Daughter*	*News*	*Silent*
Ganho comparativo(%)	0,844	0,973	0,728	0,566	0,766

4.2 SUAVIZAÇÃO DE BORDA

A ideia de se usar a suavização de borda da máscara binária surgiu de [23]. Decidiu-se então verificar a real necessidade desta aplicação. A Tabela 4.3 mostra os ganhos do uso da suavização para cada sequência CIF e a Figura 4.3 mostra as curvas para a sequência *News*. Em média, o

uso da suavização da máscara teve uma redução de taxa de $1,032\%$ sobre o outro caso. Este teste também foi feito com 60 quadros de cada sequência.

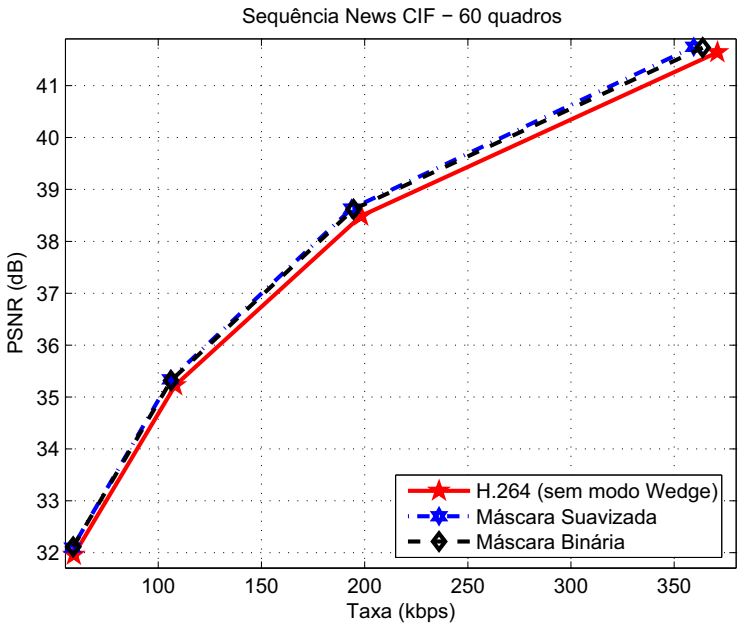


Figura 4.3: Curvas de PSNR para máscaras binária e suavizada.

Tabela 4.3: Ganhos percentuais comparativos para suavização da máscara binária.

Sequências:	*Foreman*	*Mobile*	*Mother & Daughter*	*News*	*Silent*
Ganho comparativo(%)	0,36	1,71	1,04	0,77	1,28

Um questionamento que surgiu foi se seria mais interessante usar essa suavização na máscara já no processo de estimação de movimento. Chegou-se a fazer as implementações necessárias no código, porém notou-se uma elevação absurda no tempo de execução. A explicação para tal fato foi a alteração no tipo e quantidade de operações para o cálculo dos resíduos. Uma das vantagens do padrão H.264 são os cálculos pelo uso de aritmética inteira, e o uso das máscaras binárias não muda muito isso, visto que se trata apenas de multiplicações por zero ou um. Por outro lado, quando se faz a aplicação das máscaras suavizadas, são necessárias multiplicações por aritmética de ponto flutuante. Como aproximadamente 90% do processamento do codificador H.264 é gasto em estimação de movimento, o uso de aritmética de ponto flutuante para cada busca de cada submodo *wedge* torna a suavização da máscara na estimação de movimento simplesmente inviável em nossa implementação.

4.3 CODIFICAÇÃO DOS PARÂMETROS DA *WEDGE*

A implementação do modo *wedge*, além do aumento considerável no tempo de execução do codificador, implica também a necessidade de se codificar uma informação extra: os parâmetros da *wedge*. Inicialmente, tinha-se por objetivo fazer todas as alterações necessárias de modo que esses parâmetros fossem inseridos na *bitstream* codificada juntamente com todas as outras informações. Para tal, optou-se por usar o codificador Exp-Goulomb como codificador de entropia. Porém, sem um conhecimento prévio da estatística desses parâmetros, assumiu-se as distribuições exponencial para raios e uniforme para ângulos, conforme [23].

A quantidade de raios possíveis é igual a 8 por ser metade da largura ou altura do macrobloco, ou seja, tem-se uma variação no tamanho do raio Δr de 1 *pixel*. Com esse valor, poderíamos determinar a quantidade de ângulos a serem usados a partir da menor variação possível que gerasse máscaras distintas. Contudo, isso possivelmente nos daria uma quantidade muito elevada de submodos *wedge* e dificultaria a comparação de resultados com os trabalhos anteriores. Optou-se então por um total de 32 possíveis ângulos, com variação $\Delta\theta = 11,25°$, assim como em [23].

Como o total de *wedges* possíveis era de 256 (produto entre 8 valores de raios e 32 valores de ângulos), seria necessário um *byte* extra para cada macrobloco. A maneira pensada para melhor aproveitar o codificador Exp-Goulomb foi agrupar os 3 *bits* referentes aos raios aos 5 *bits* referentes aos ângulos, de forma que os 3 *bits* dos raios fossem os mais significativos, como mostrado na Figura 4.4.

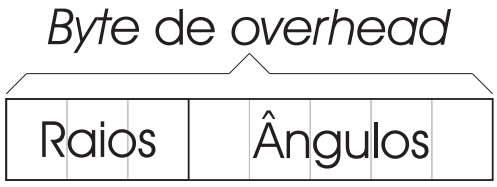


Figura 4.4: *Byte* de *overhead* com *bits* de raios e ângulos.

Infelizmente, verificou-se que os ganhos não chegavam muito próximos daqueles esperados. Por exemplo, para 300 quadros da sequência *Foreman* tamanho CIF, era esperado um ganho de aproximadamente $6,12\%$ em taxa (segundo [26]), mas nossos resultados não chegavam sequer aos 5%. Tivemos então que ceder à intenção de manter toda a informação codificada na *bitstream* e enviar a informação de *overhead* sem codificação para o decodificador por meio de informação

lateral, ou seja, usando um arquivo ou *bitstream* separados. O uso de um arquivo separado se deu apenas por questão de conveniência.

De maneira muito interessante, observamos que os resultados esperados foram atingidos usando-se a informação lateral não codificada. O passo seguinte foi verificar, da nossa parte, qual a estatística dos parâmetros da *wedge* e determinar se era realmente possível algum tipo de codificação. Para tal, fizemos testes calculando a entropia tanto dos raios e ângulos separadamente, quanto do *byte* inteiro e o comprimento médio dos códigos de Huffman (códigos ótimos [19]) para esses casos. As Tabelas 4.4 4.5 mostram os resultados para estes testes, com valores dados em *bits*.

Tabela 4.4: Entropia de raios e ângulos.

Sequências:	*Foreman*	*Mobile*	*Mother & Daughter*	*News*	*Silent*
Overhead	7,829	7,8987	7,8622	7,8542	7,887
Raio	2,9477	2,9761	2,9248	2,9612	2,9562
Ângulo	4,9457	4,9665	4,9841	4,9493	4,9734

Tabela 4.5: Comprimentos médios dos códigos de Huffman para raios e ângulos.

Sequências:	*Foreman*	*Mobile*	*Mother & Daughter*	*News*	*Silent*
Overhead	7,8541	7,9255	7,8928	7,8838	7,9194
Raio	3	3	2,969	3	3
Ângulo	4,9682	4,995	5	4,9802	4,9965

Esses resultados mostram que não é prático comprimir a informação referente aos parâmetros da *wedge*, pois como mostrado na Tabela 4.4, as distribuições tanto dos raios quanto dos ângulos são praticamente uniformes. Além disso, os comprimentos de códigos gerados são todos maiores que os valores das entropias, para cada caso.

4.4 MODOS *WEDGE* E *ONE-WEDGE*

Como mencionado anteriormente, os modos *wedge* e *one-wedge* foram inseridos como modos extras no padrão H.264. Devido ao elevado gasto de tempo pela estimação de movimento desses

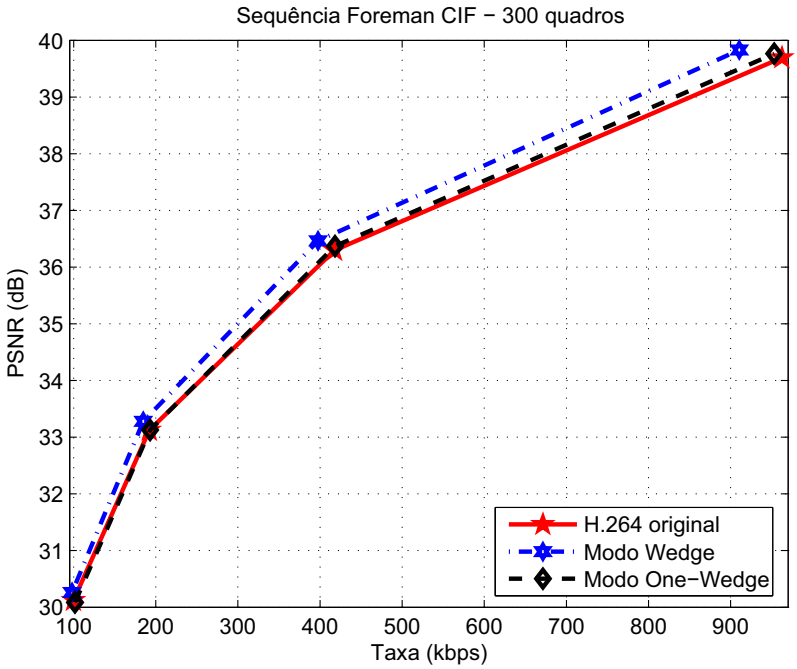
modos, decidiu-se por usar apenas para esses modos o método de busca rápida já implementado no JSVM. Isso também foi descrito em [23], [26]. São apresentados então os resultados dos testes com esses dois modos para todas as sequências. Na Tabela 4.6, são mostrados os ganhos comparativos em taxa e distorção, enquanto que nas Figuras 4.5, 4.6, 4.7 e 4.8 podem ser vistas as curvas de PSNR para os dois modos *wedge* e *one-wedge*. Nesses testes são mostradas também as curvas de PSNR para a codificação dos vídeos pelo padrão H.264 para fins de comparação.

Para as partições *one-wedge*, esperava-se que ocorressem ganhos importantes apenas em determinadas sequências com fundo estático. Do grupo de sequências usadas para os testes, aquelas que apresentam tal características são *Mother & Daughter*, *Silent*, *Claire* e *Salesman*. Curiosamente, apenas as sequências de tamanho QCIF, mesmo aquela que não tem a característica para a qual o modo foi proposto (sequência *Container*), mostraram algum ganho competitivo.

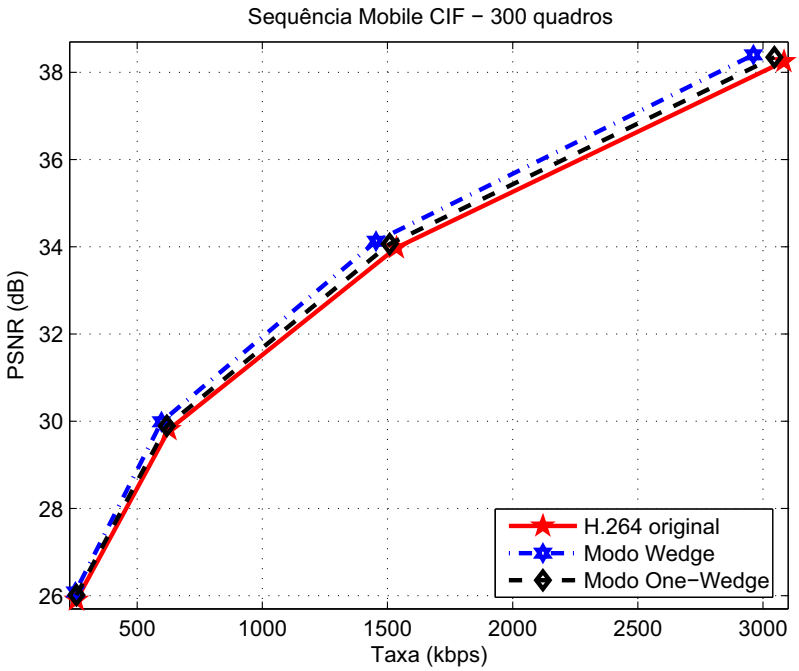
Para o modo *wedge*, foram testados todos os 256 submodos, como explicado na Seção 4.3. Com a intenção de se manter o mesmo *overhead* por macrobloco, os testes no modo *one-wedge* foram feitos com 128 submodos. Isto ocorreu porque, para este modo, é preciso informar ao decodificador qual das partições, Interna ou Externa, é a partição estática, o que exige um *bit* a mais no *overhead*. Desta forma, optou-se por elevar a variação de ângulo para $\Delta\theta = 22,5°$.

Tabela 4.6: Ganhos Médios em Taxa (%) e Distorção Medida por PSNR (dB) - Modos *Wedge* e *One-Wedge*.

Sequência		Modos	
		Wedge	*One-Wedge*
Foreman	Taxa	7,67	0,56
	PSNR	0,338	0,024
Mobile	Taxa	7,65	3,16
	PSNR	0,394	0,159
Mother &	Taxa	4,69	0,159
Daughter	PSNR	0,207	0,091
News	Taxa	5,88	0,29
	PSNR	0,319	0,017
Silent	Taxa	6,39	0,54
	PSNR	0,299	0,025
Claire	Taxa	7,56	5,94
	PSNR	0,414	0,321
Container	Taxa	3,75	2,61
	PSNR	0,172	0,115
Salesman	Taxa	7,06	4,96
	PSNR	0,380	0,264
Média	Taxa	6,456	2,277
	PSNR	0,3114	0,127

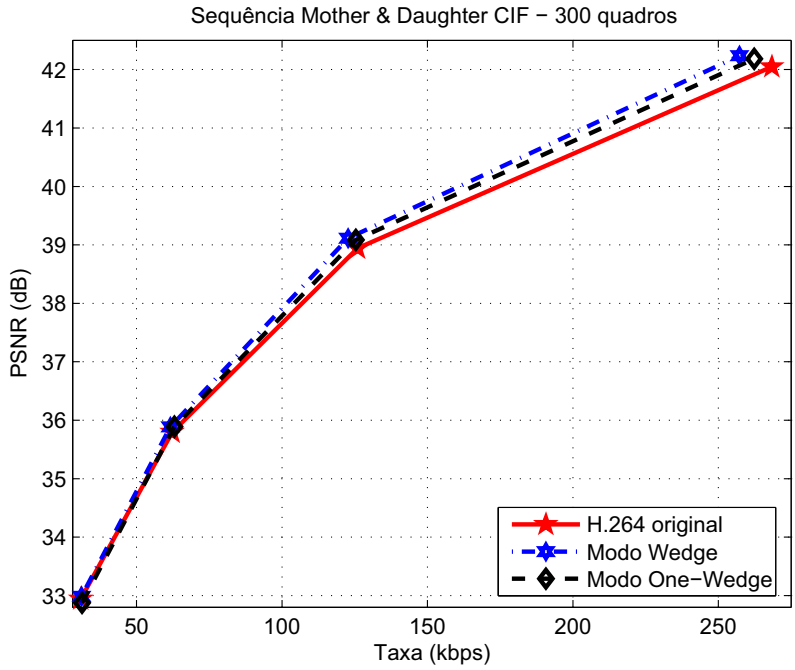


(a)

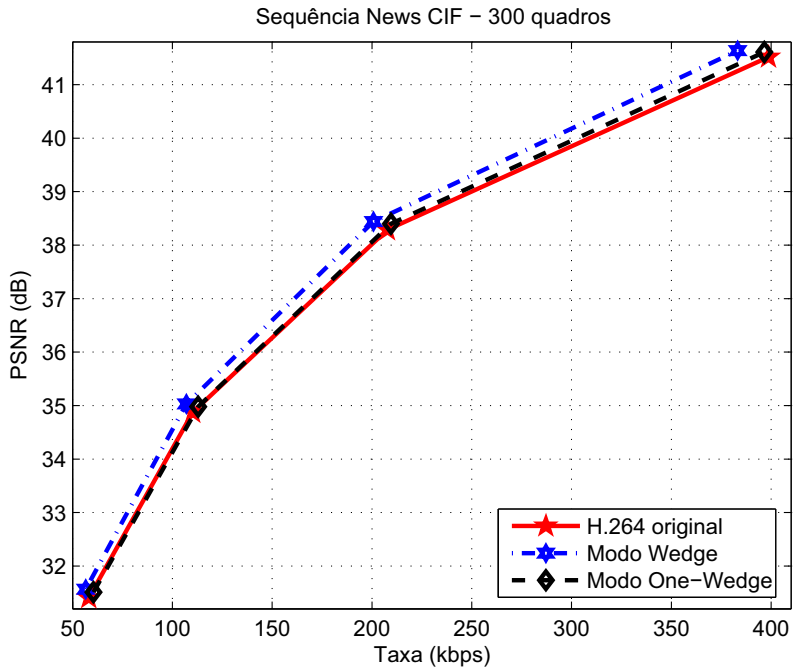


(b)

Figura 4.5: Curvas de PSNR para os Modos *Wedge* e *One-Wedge*: (a) *Foreman*; (b) *Mobile*.

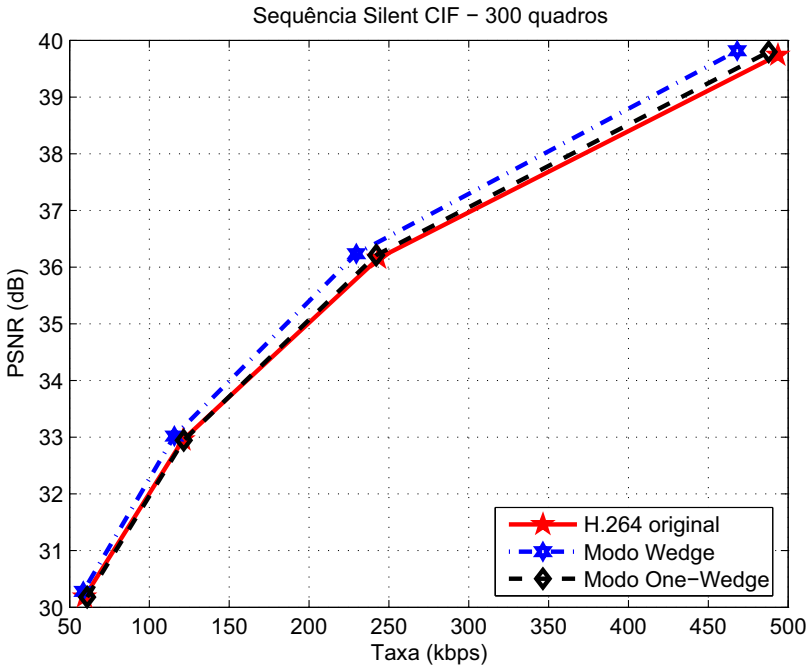


(a)

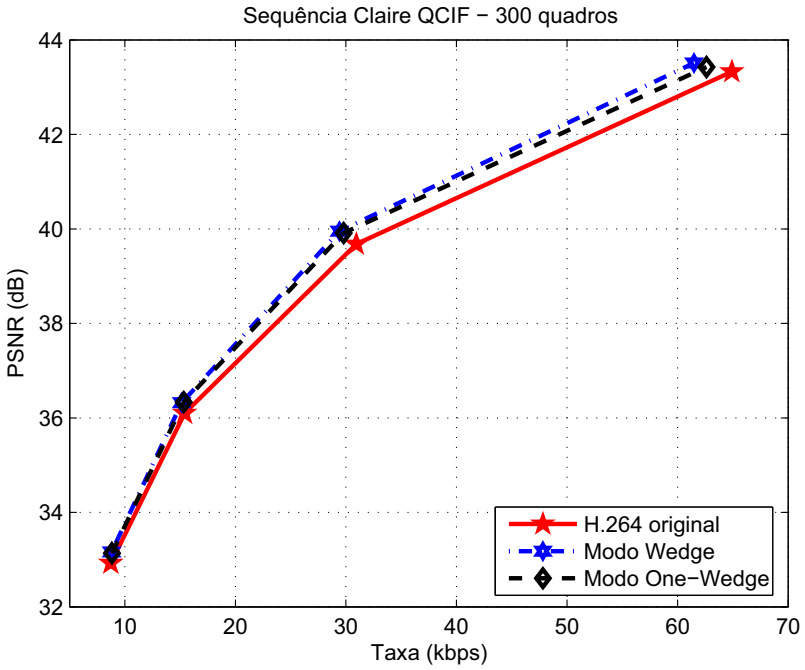


(b)

Figura 4.6: Curvas de PSNR para os Modos *Wedge* e *One-Wedge*: (a) *Mother & Daughter*; (b) *News*.

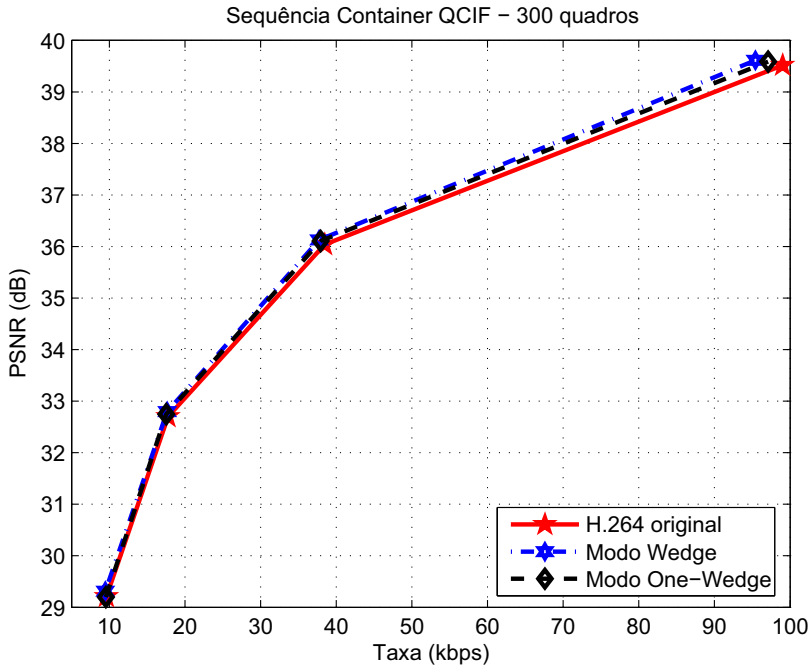


(a)

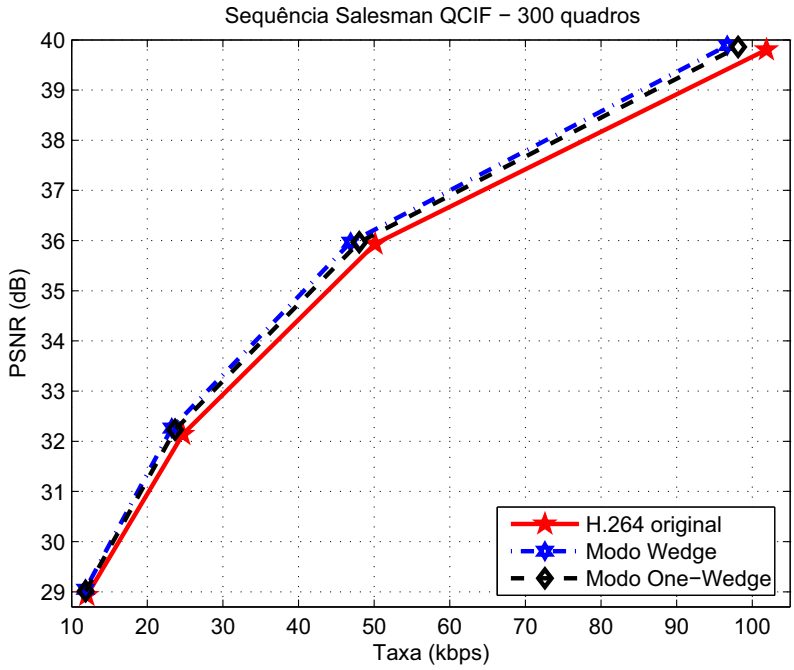


(b)

Figura 4.7: Curvas de PSNR para os Modos *Wedge* e *One-Wedge*: (a) *Silent*; (b) *Claire*.



(a)



(b)

Figura 4.8: Curvas de PSNR para os Modos *Wedge* e *One-Wedge*: (a) *Container*; (b) *Salesman*.

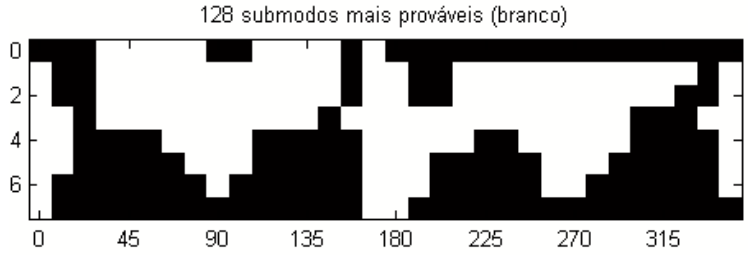
4.5 REDUÇÃO DE COMPLEXIDADE

Como apresentado no Capítulo 3, foram propostas algumas técnicas para a redução de complexidade no uso de partições alternativas de macroblocos. Duas das três técnicas não trouxeram grandes benefícios, diferentemente do que era esperado. Felizmente, um dos métodos (análise estatística) trouxe resultados bastante satisfatórios.

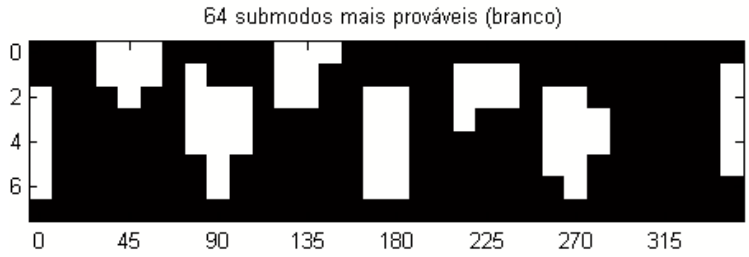
4.5.1 Análise estatística

Este método de redução de complexidade foi o que mostrou os resultados mais interessantes. Foram feitos testes com vários níveis de redução de complexidade e consequentemente de redução de *overhead*. A escolha dos submodos usados depende não só da estatística dos parâmetros mas também da quantidade de modos que se deseja usar. Como a quantidade de modos está diretamente ligada ao *overhead* por macrobloco, é interessante que a quantidade de modos seja um número igual a uma potência de 2. Então, partindo do máximo inicial de partições possíveis, cada vez que se restringe a quantidade, essa quantidade cai pela metade e consequentemente reduz o *overhead* em 1 *bit*. Por isso, foram feitos testes com os 128, 64 e 32 submodos mais prováveis da base de dados, que resultam cada um em *overheads* de 7, 6 e 5 *bits* por macrobloco. As Figuras 4.9 (a), (b) e (c) mostram graficamente esses submodos mais prováveis. Já a Figura 4.9(d) representa a sobrepodição das três outras figuras.

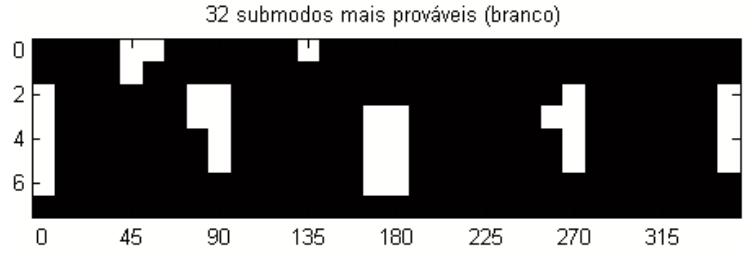
As Figuras 4.10-4.22 mostram as curvas de PSNR para cada uma das sequências de vídeo testadas. Como já era esperado, observa-se que a redução de complexidade implica redução de ganho sobre o padrão H.264. Contudo, os resultados mostram que de fato, apesar da diminuição de submodos *wedge* usados, os ganhos ainda são bem próximos do ganho máximo (que usa todos os submodos). A Tabela 4.7 mostra os ganhos percentuais para taxa e diminuição da distorção para cada caso de redução na quantidade de submodos para todas as sequências testadas.



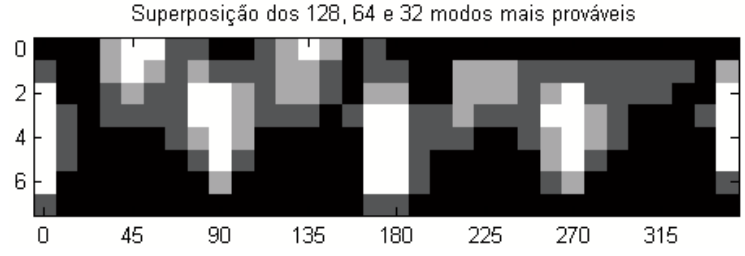
(a)



(b)



(c)

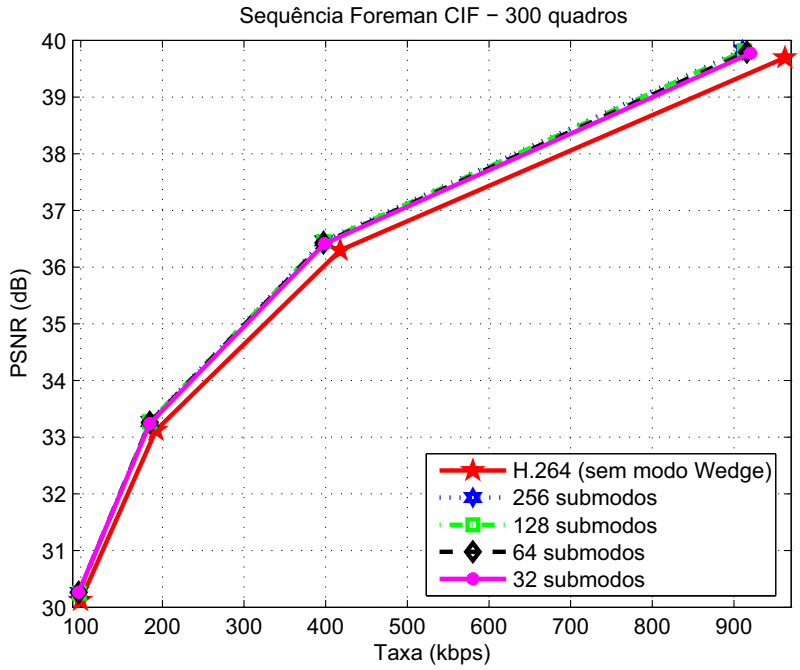


(d)

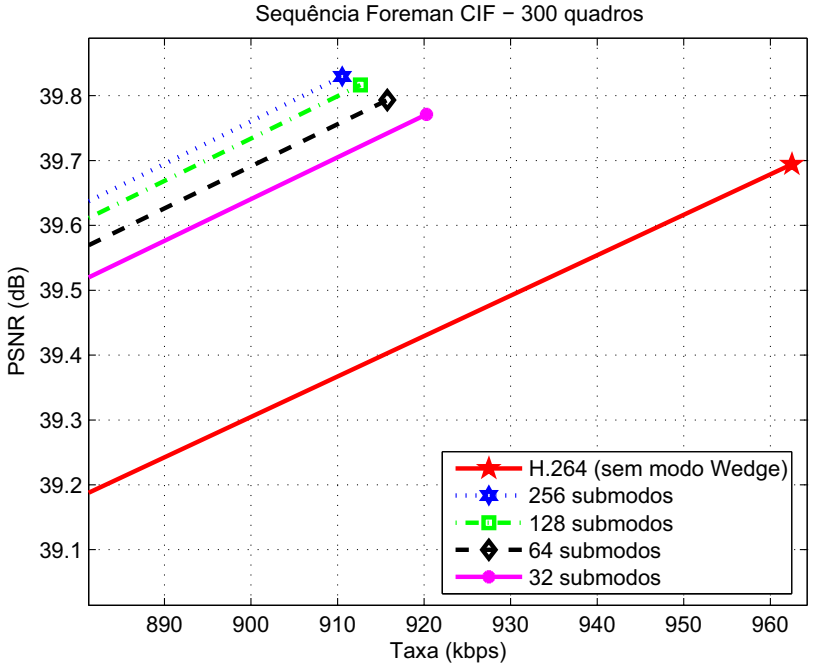
Figura 4.9: Submodos mais prováveis: (a) 128; (b) 64; (c) 32; (d) Sobreposição dos três casos.

Tabela 4.7: Ganhos Médios em Taxa (%) e Distorção Medida por PSNR (dB) - Redução de Complexidade.

Sequência		*Overhead* (em *bits*)			
		8	7	6	5
Foreman	Taxa	7,67	7,51	7,15	6,64
	PSNR	0,338	0,330	0,314	0,290
Mobile	Taxa	7,65	7,33	6,80	6,05
	PSNR	0,394	0,377	0,348	0,309
News	Taxa	5,88	5,88	5,35	4,80
	PSNR	0,319	0,318	0,289	0,258
Silent	Taxa	6,39	6,03	5,37	4,46
	PSNR	0,299	0,283	0,251	0,207
Claire	Taxa	7,56	7,27	6,52	6,44
	PSNR	0,414	0,397	0,354	0,349
Container	Taxa	3,75	3,22	3,63	3,23
	PSNR	0,172	0,146	0,164	0,148
Salesman	Taxa	7,06	6,46	6,08	5,18
	PSNR	0,380	0,347	0,325	0,277
Bus	Rate	8,12	8,14	7,30	6,56
	PSNR	0,435	0,435	0,389	0,347
City	Rate	7,46	7,33	6,79	6,10
	PSNR	0,348	0,341	0,315	0,281
Football	Rate	4,37	4,10	3,64	3,10
	PSNR	0,244	0,228	0,203	0,172
Harbour	Rate	10,81	10,78	10,39	9,79
	PSNR	0,514	0,511	0,492	0,460
Mother &	Taxa	4,69	4,75	4,74	3,88
Daughter	PSNR	0,207	0,210	0,209	0,170
Crew	Rate	8,50	8,05	7,01	5,80
	PSNR	0,285	0,269	0,233	0,191
Média	Taxa	6,92	6,68	6,21	5,54
	PSNR	0,335	0,322	0,299	0,266

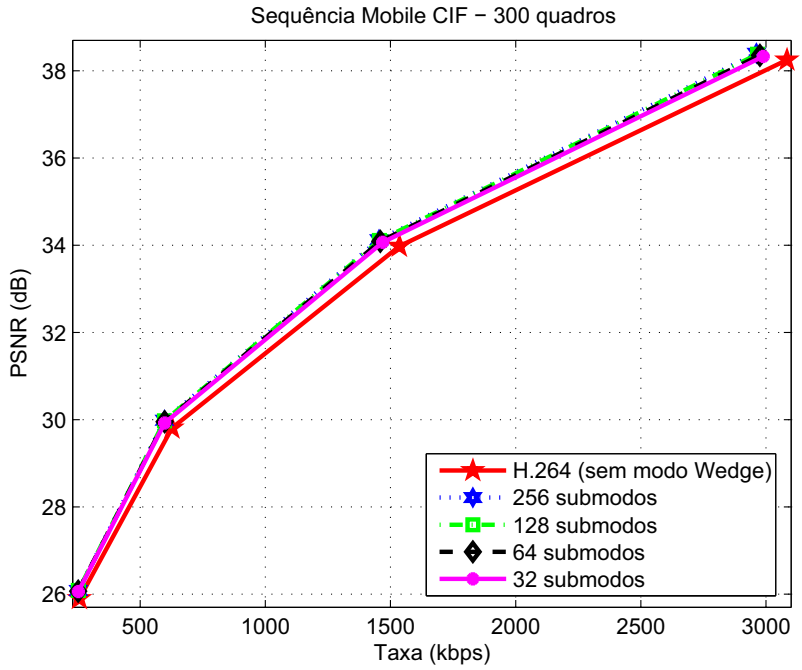


(a)

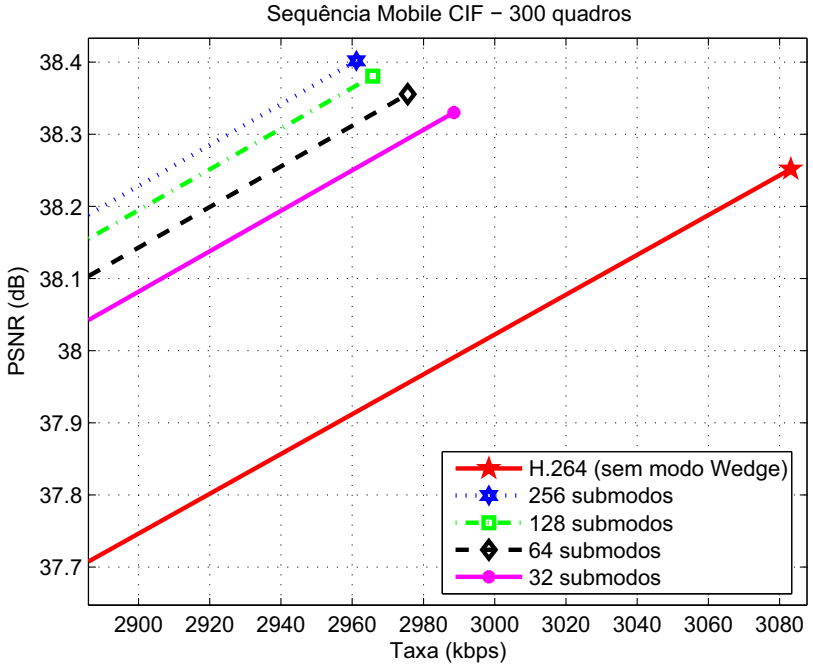


(b)

Figura 4.10: Curvas de PSNR para Redução de Complexidade - Sequência *Foreman*: (a) Curvas completas; (b) Zoom.

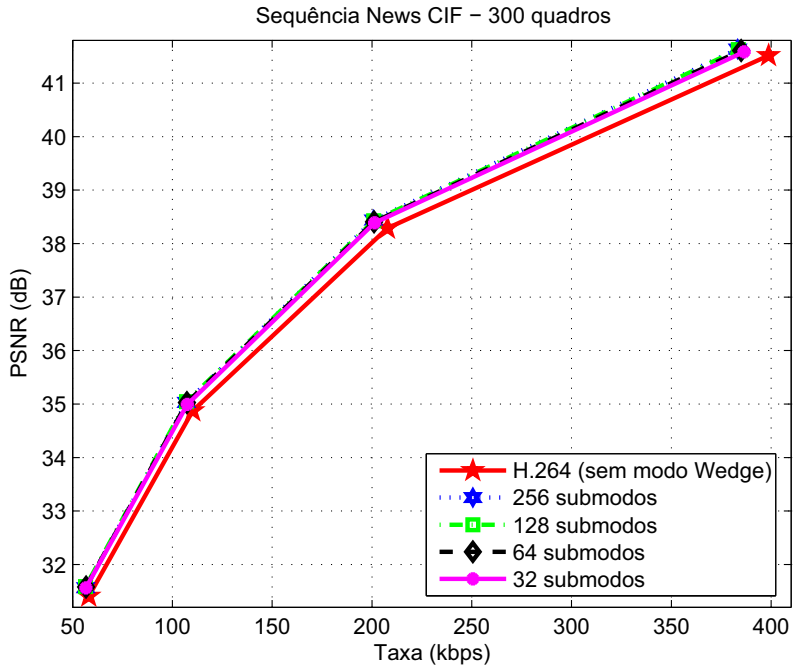


(a)

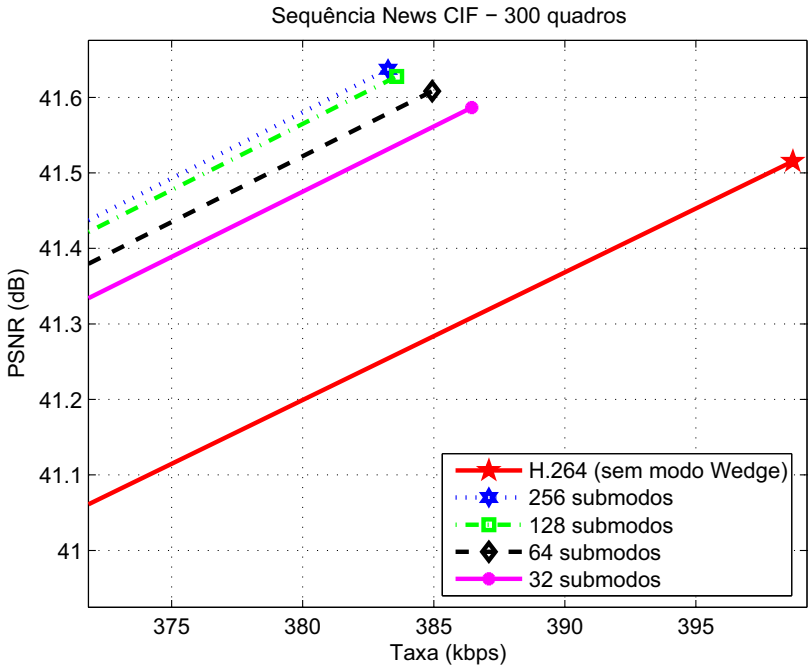


(b)

Figura 4.11: Curvas de PSNR para Redução de Complexidade - Sequência *Mobile*: (a) Curvas completas; (b) Zoom.

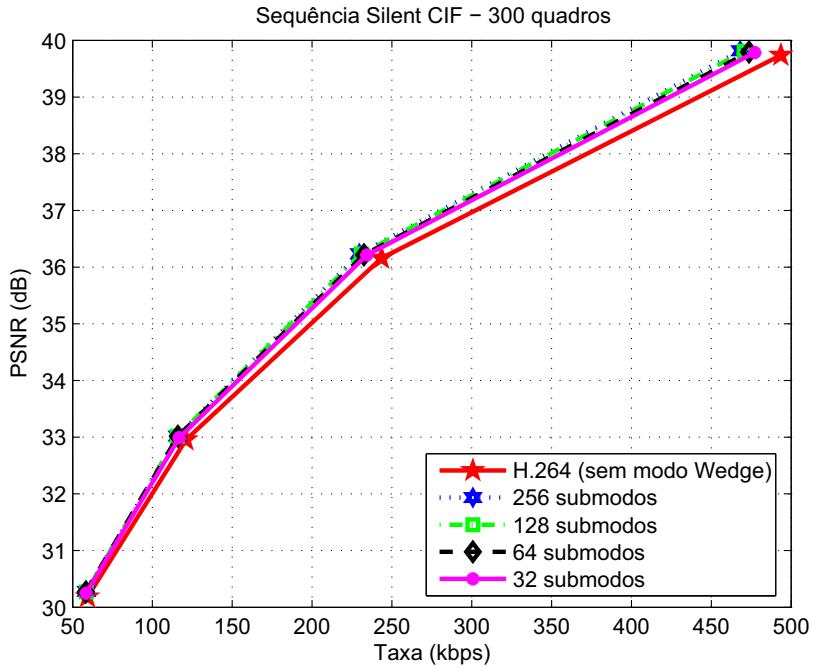


(a)

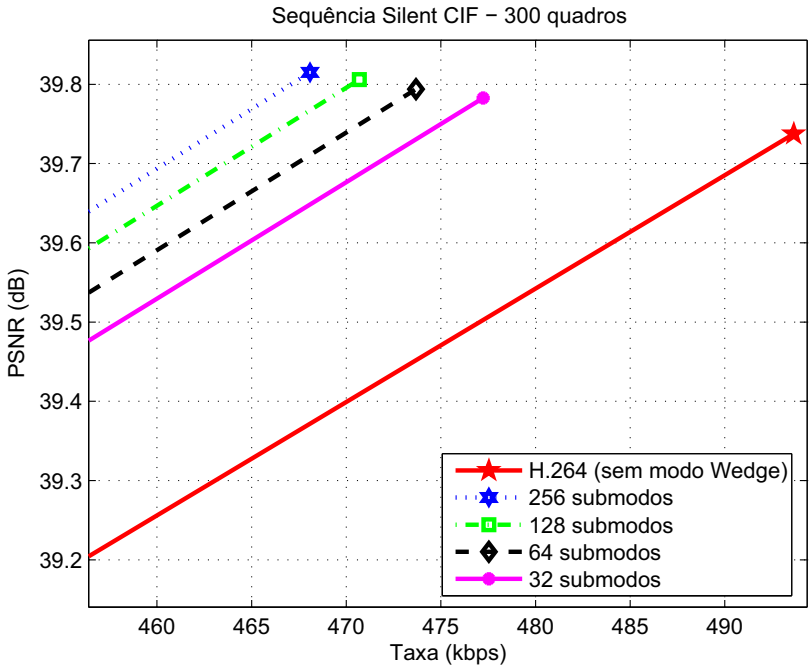


(b)

Figura 4.12: Curvas de PSNR para Redução de Complexidade - Sequência *News*: (a) Curvas completas; (b) Zoom.

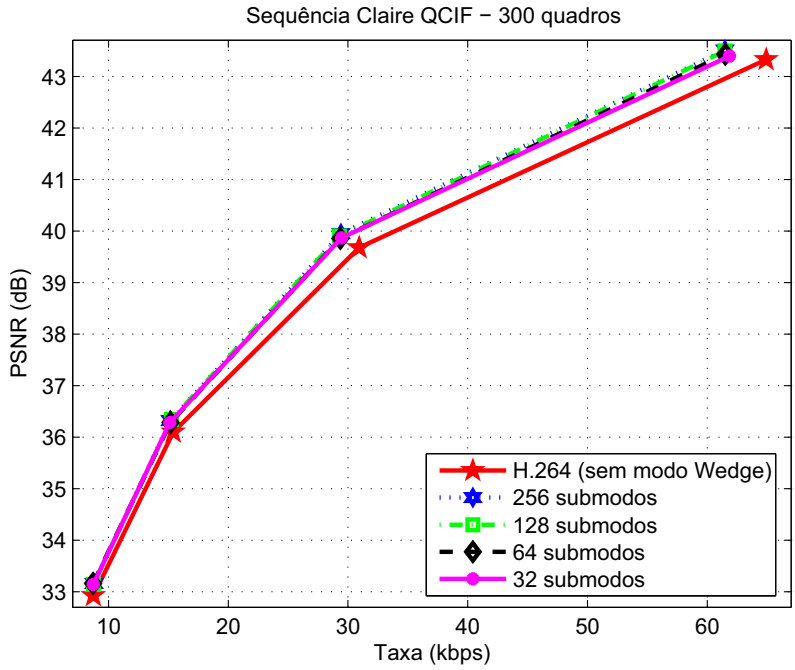


(a)

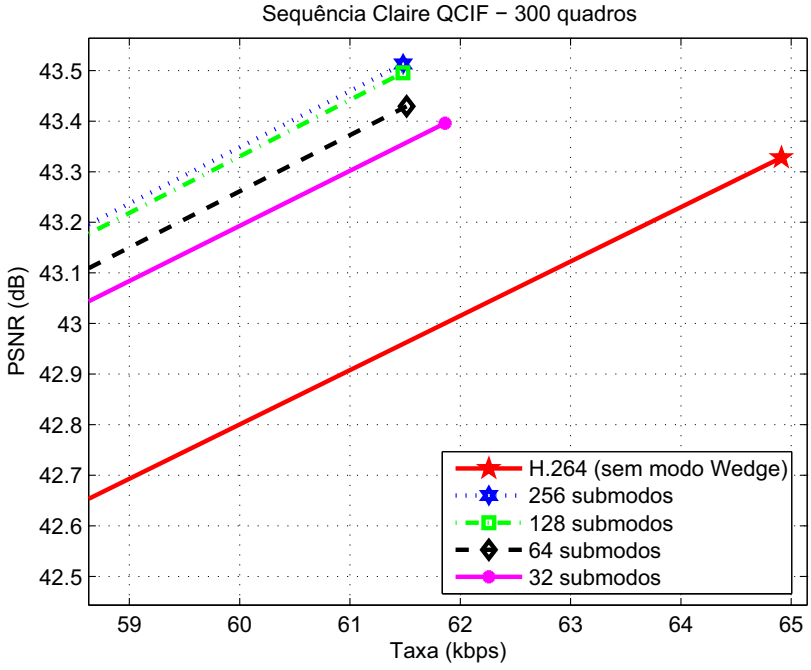


(b)

Figura 4.13: Curvas de PSNR para Redução de Complexidade - Sequência *Silent*: (a) Curvas completas; (b) Zoom.

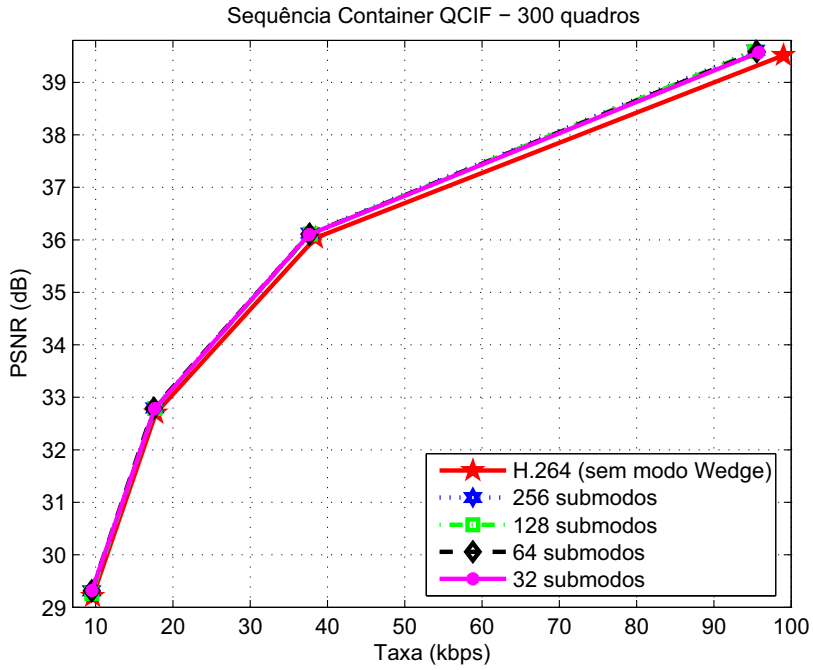


(a)

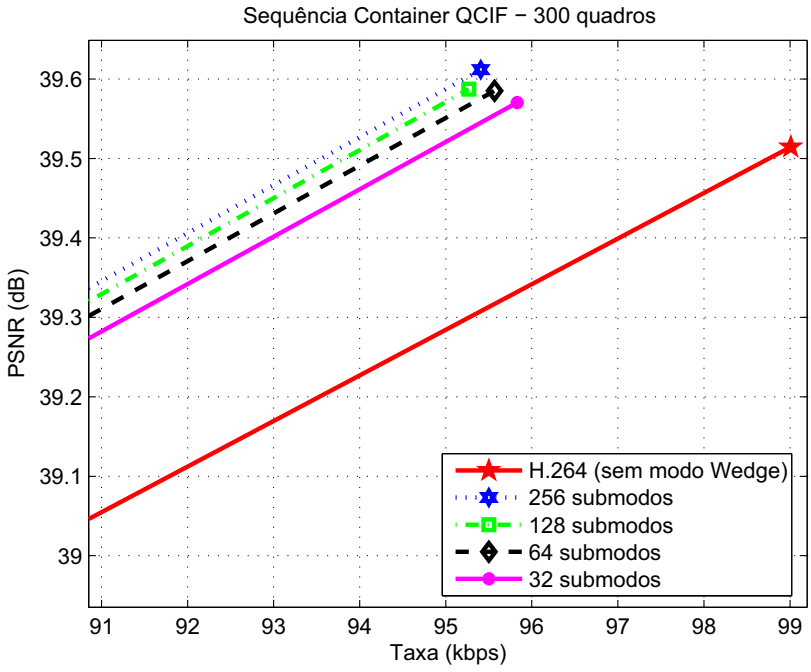


(b)

Figura 4.14: Curvas de PSNR para Redução de Complexidade - Sequência *Claire*: (a) Curvas completas; (b) Zoom.

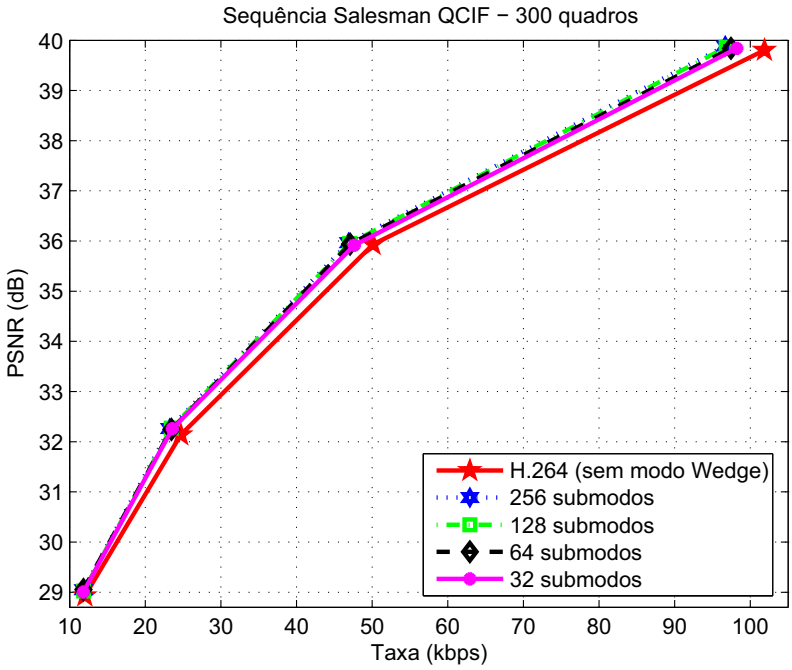


(a)

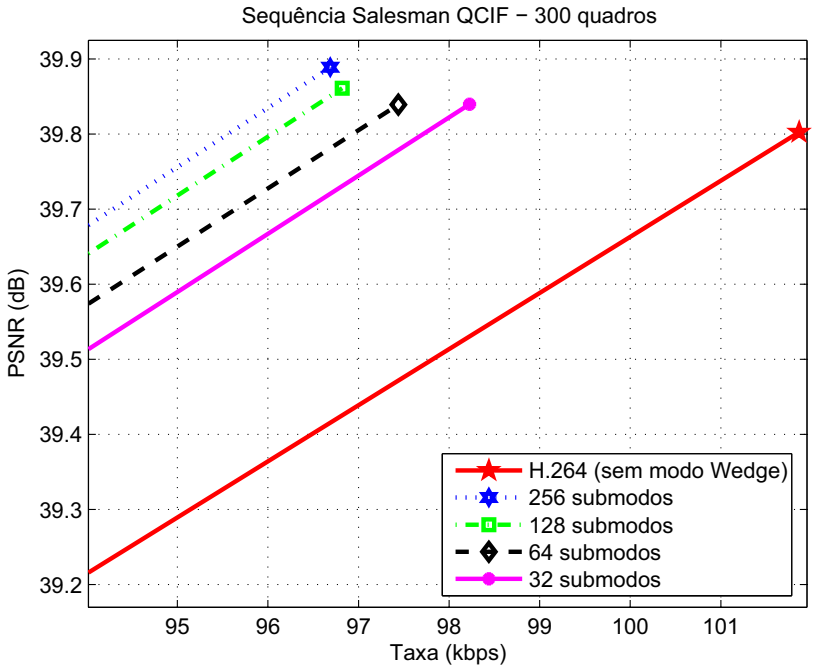


(b)

Figura 4.15: Curvas de PSNR para Redução de Complexidade - Sequência *Container*: (a) Curvas completas; (b) Zoom.

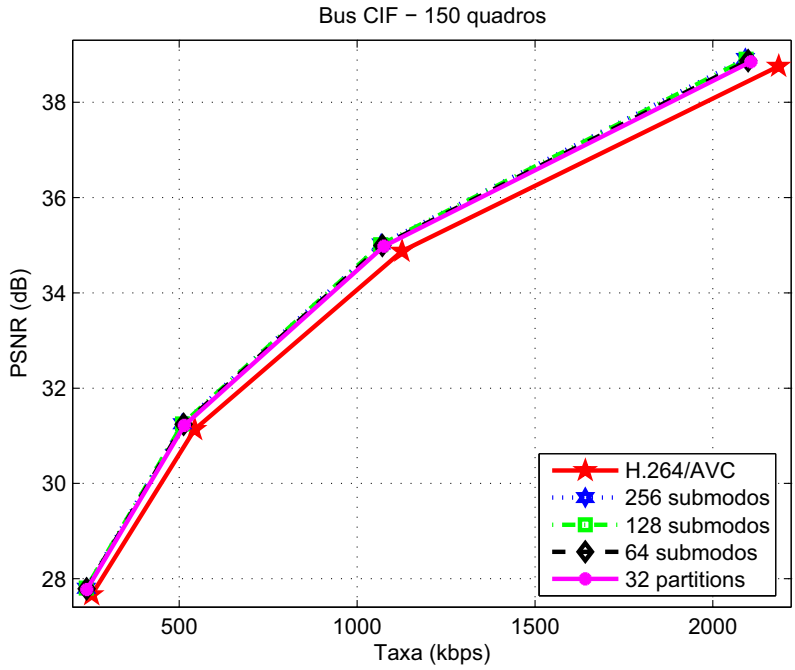


(a)

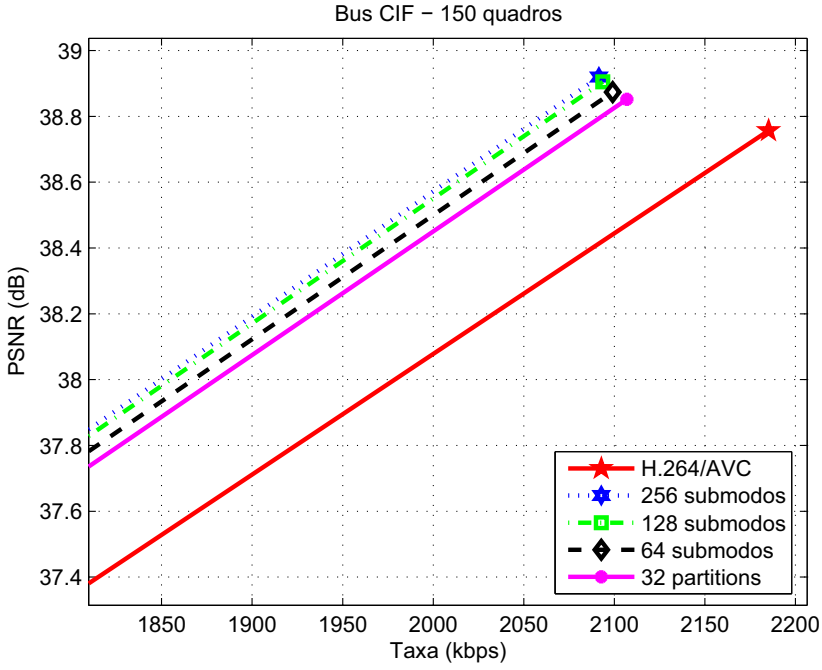


(b)

Figura 4.16: Curvas de PSNR para Redução de Complexidade - Sequência *Salesman*: (a) Curvas completas; (b) Zoom.

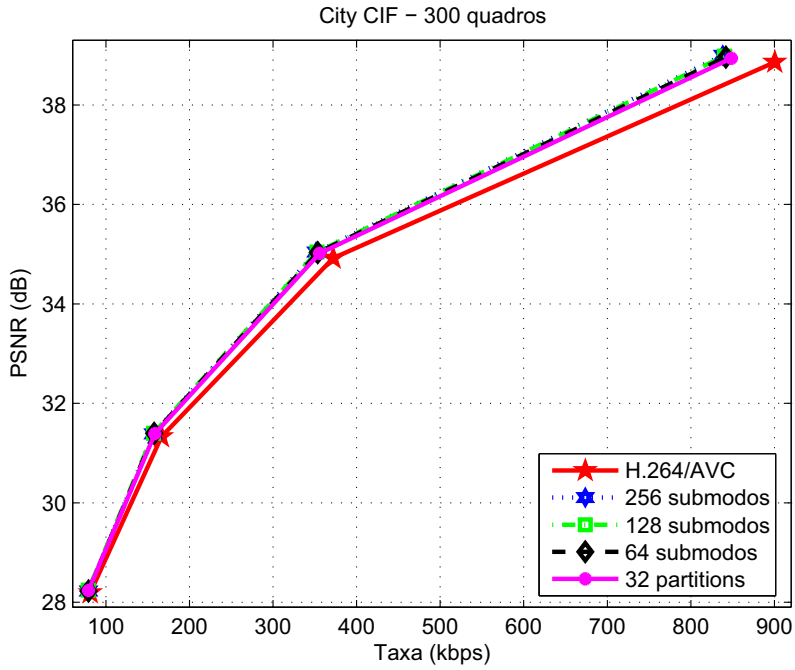


(a)

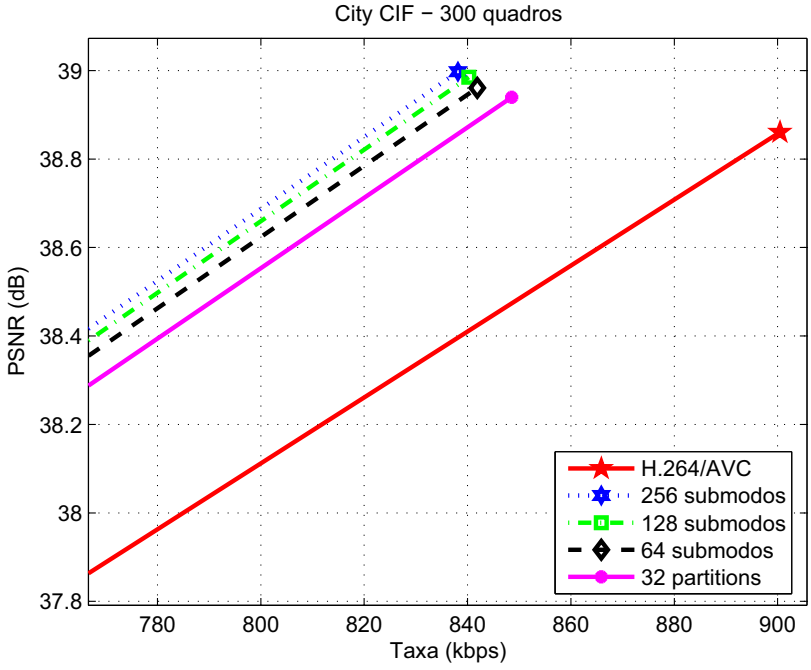


(b)

Figura 4.17: Curvas de PSNR para Redução de Complexidade - Sequência *Bus*: (a) Curvas completas; (b) Zoom.

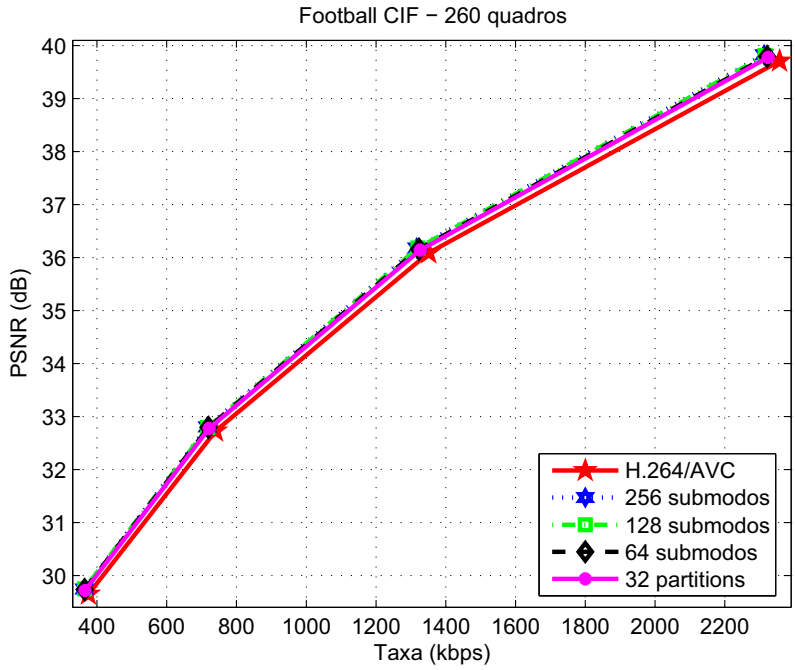


(a)

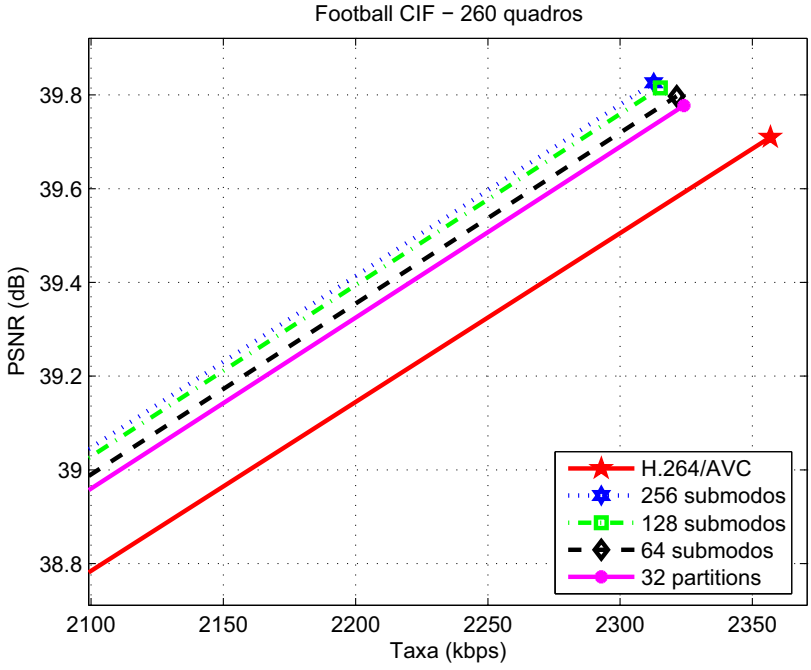


(b)

Figura 4.18: Curvas de PSNR para Redução de Complexidade - Sequência *City*: (a) Curvas completas; (b) Zoom.

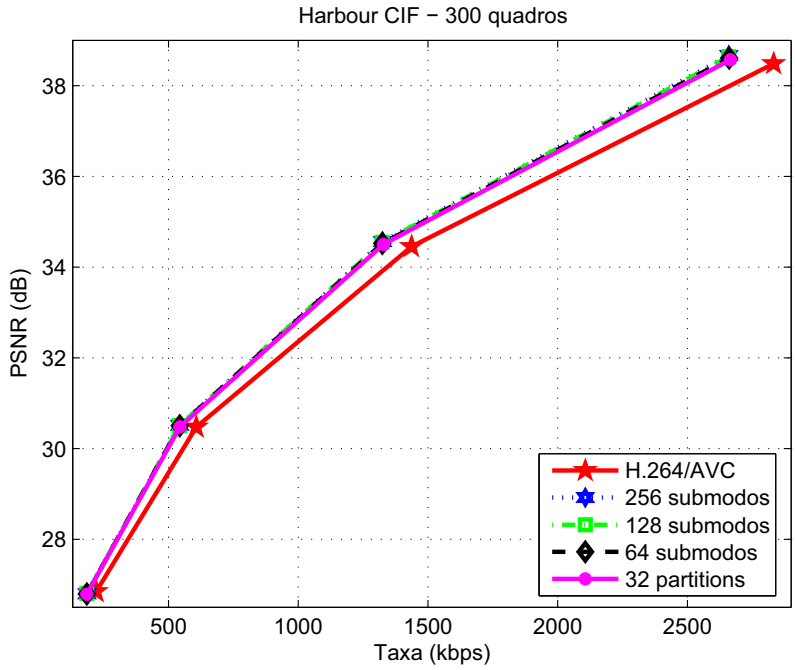


(a)

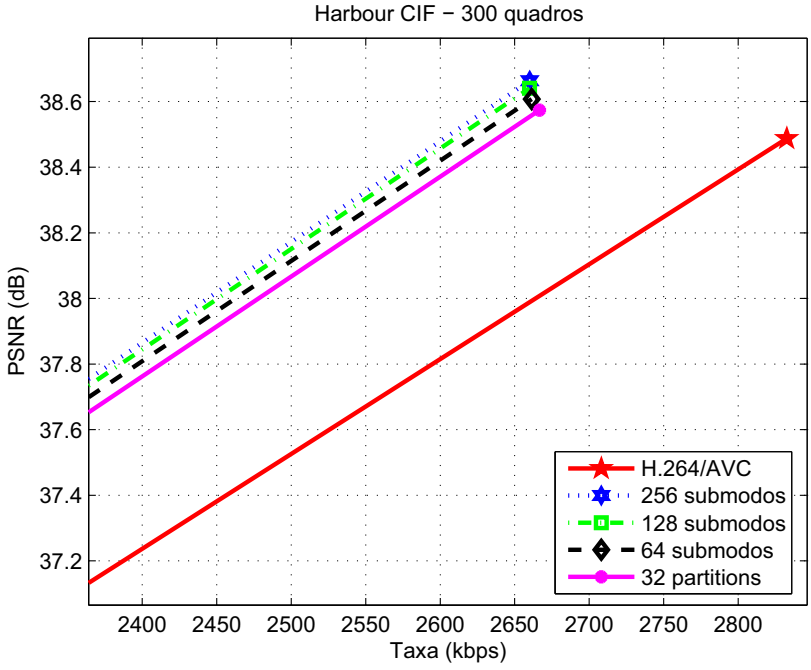


(b)

Figura 4.19: Curvas de PSNR para Redução de Complexidade - Sequência *Football*: (a) Curvas completas; (b) Zoom.

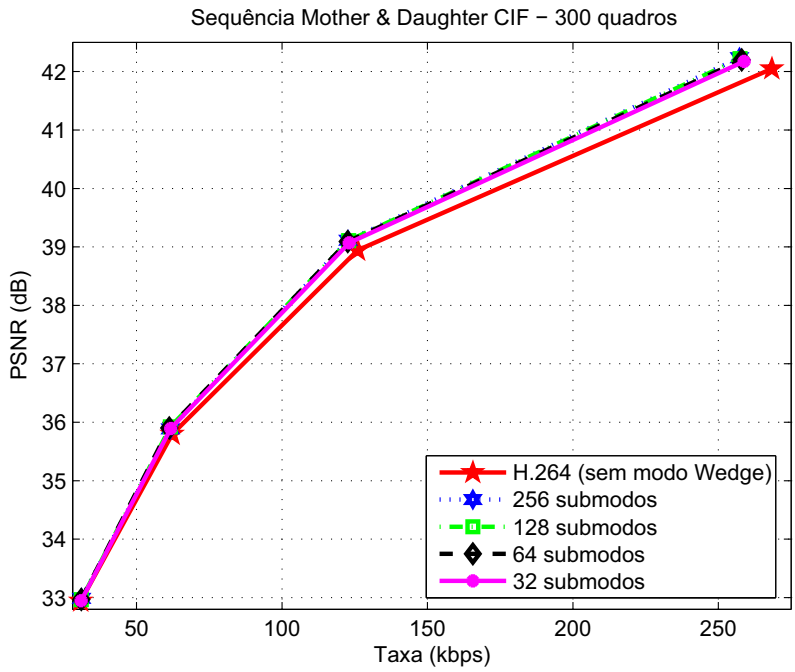


(a)

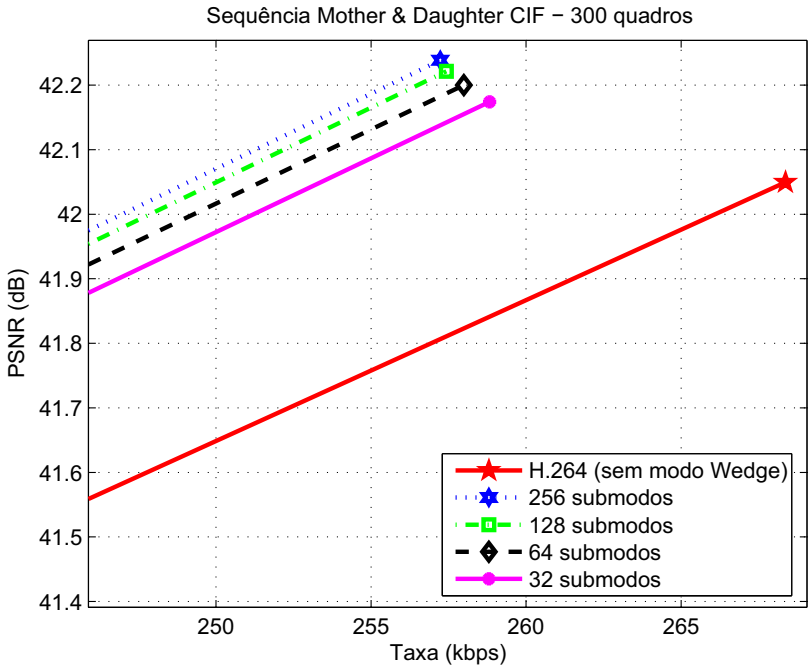


(b)

Figura 4.20: Curvas de PSNR para Redução de Complexidade - Sequência *Harbour*: (a) Curvas completas; (b) Zoom.

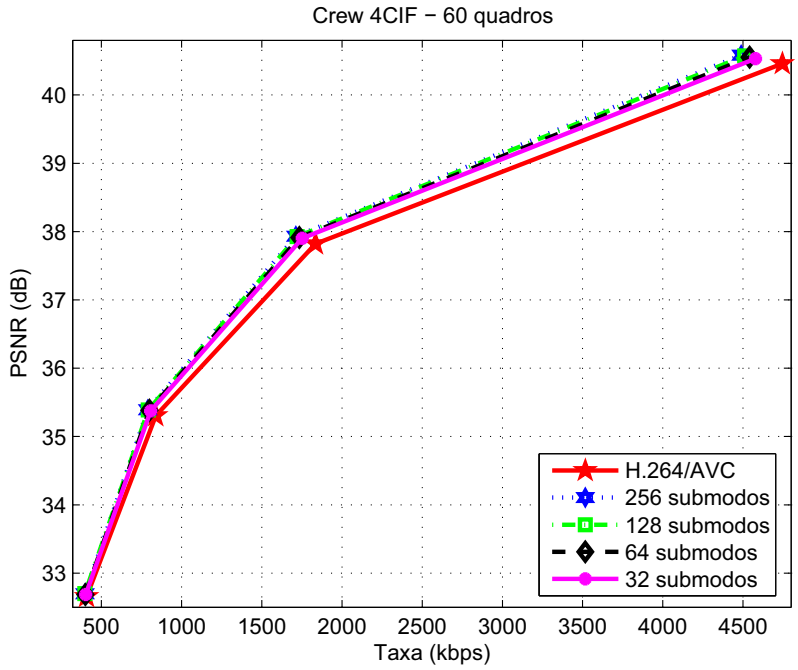


(a)

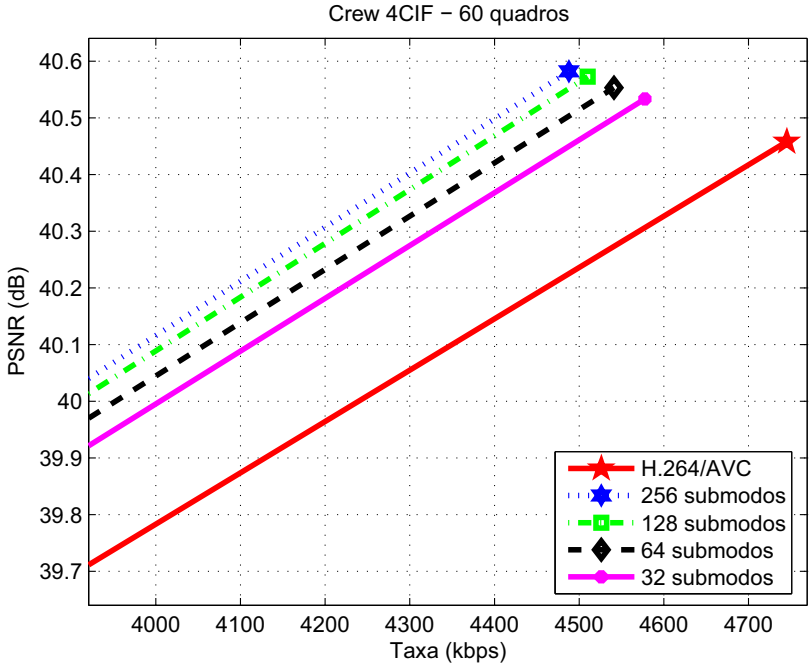


(b)

Figura 4.21: Curvas de PSNR para Redução de Complexidade - Sequência *Mother & Daughter*: (a) Curvas completas; (b) Zoom.



(a)



(b)

Figura 4.22: Curvas de PSNR para Redução de Complexidade - Sequência *Crew*: (a) Curvas completas; (b) Zoom.

4.5.2 Método rápido baseado em mapa de diferenças para modo *One-Wedge*

Este método mostrou uma vantagem quase imperceptível sobre a codificação feita unicamente pelo padrão. Em termos percentuais médios, o ganho para as sequências CIF foi de apenas $0,04\%$. Já para as sequências QCIF, foi de $0,6\%$. Interessantemente, assim como ocorrido para o modo *one-wedge* completo, os resultados para as sequências de tamanho QCIF foram melhores que para as demais sequências.

4.5.3 Partição por máscara binária arbitrária

Este método foi o que mostrou os piores resultados. Mesmo sem contar o *overhead* constituído pelas máscaras binárias, não foi observado qualquer ganho. Assim como no caso do método anterior, este método foi escolhido para pouquíssimos macroblocos e é possível que esta escolha só tenha ocorrido por não se ter considerado os *bits* referentes às máscaras binárias no momento em que é calculado o custo do uso deste tipo de partição.

Como foi comentado quando o método foi apresentado, um dos principais problemas desse tipo de partição é o fato de ser difícil estimar a taxa do macrobloco levando em consideração o *overhead*. Por causa disso, durante o cálculo do custo, considerou-se que o custo extra seria zero. A ideia era primeiramente verificar a viabilidade do método para, em seguida, determinar melhores maneiras de se organizar as máscaras para serem codificadas pelo codificador JBIG e, finalmente, buscar possibilidades de se estimar o custo do macrobloco durante a escolha de modos. Como os primeiros resultados já foram insatisfatórios, os demais passos foram abandonados.

5 CONCLUSÕES

5.1 SUMÁRIO DO TRABALHO

O objetivo principal deste trabalho foi a implementação de partições alternativas de macroblocos na estimação e compensação de movimento do padrão de codificação de vídeo H.264/AVC. Em segundo plano, buscou-se um estudo e proposição de métodos de redução de complexidade das implementações feitas.

As partições alternativas estão fundamentadas em divisões do macrobloco por segmentos de reta de direções arbitrárias, nomeadas de *wedges*. Em um caso, denominado modo *wedge*, as duas regiões derivadas da segmentação são usadas na estimação de movimento. Para o outro caso, denominado modo *one-wedge* (que é uma restrição do modo *wedge*), a busca foi feita somente para uma das regiões.

Para reduzir o custo computacional extra resultante da inserção desses novos modos de partição, foram abordadas três frentes: uma para cada modo (*wedge* e *one-wedge*) e outra independente. Para o modo *wedge*, foi feito um levantamento estatístico dos submodos mais recorrentes. Já para o modo *one-wedge*, buscou-se a definição *a priori* do submodo mais apropriado a partir de um mapa de diferenças. A terceira frente foi o uso de partições completamente arbitrárias, sem nenhuma forma pré-definida.

5.2 AVALIAÇÃO DOS RESULTADOS

Os resultados mostraram que o uso do modo *wedge* de maneira complementar aos demais modos do padrão H.264 resultou em uma redução média de $6,5\%$ em taxa para as oito sequências testadas. Em comparação com resultados apresentados anteriormente [26], as sequências *Foreman*, *Mobile*, *Silent* mostraram ganhos superiores aos anteriores. Como naquele trabalho as demais sequências de tamanho CIF não foram testadas e as sequências de tamanho QCIF tinham taxa de quadros de 15 Hz e foram testadas apenas 150 quadros, é inviável fazer uma comparação

direta. Já o uso do modo *one-wedge* resultou em uma redução média de $2,3\%$ em taxa, sendo que, se forem levadas em consideração apenas as sequências de tamanho QCIF, este valor sobe para $4,5\%$.

Os ganhos obtidos aqui, maiores que aqueles previamente apresentados, podem ser explicados por dois motivos principais. O primeiro é o uso do valor de *mb_type* $= 1$ para o modo *wedge*. O segundo é a predição do vetor de movimento mais abrangente, o que implica inclusive diferentes resultados para o submodo *wedge* $(0; 0°)$ e quando comparados ao modo 16x8, apesar de as formas de partição serem idênticas.

Comparando os modos *wedge* e *one-wedge*, nota-se que apesar de o segundo modo ser relativamente melhor para sequências de tamanho QCIF, seus ganhos ainda são inferiores àqueles do outro modo. É possível que parte da diferença entre os ganhos dos dois modos seja decorrente da diferença entre os valores de $\Delta\theta$ para cada modo. Entretanto, mesmo o modo *one-wedge* operando com 256 submodos e 9 bits de *overhead*, é bem improvável que os dois modos tivessem ganhos tão próximos quanto desejável.

Os resultados referentes ao levantamento estatístico dos submodos *wedge* mais recorrentes foram bastantes satisfatórios. Claramente, para as sequências usadas na base de dados (*Foreman*, *Mobile*, *News* e *Silent*), já era esperado que a redução no número de submodos não diminuiria tanto os ganhos. Contudo, os resultados mais surpreendentes foram observados para as demais sequências (*Claire*, *Container Salesman*, *Bus*, *City*, *Football*, *Mother & Daughter* e *Crew*). Isto comprova a ideia de que é possível explorar a concentração de submodos e consequente redução do custo extra do modo *wedge* (causado pelo *overhead*) e que as sequências de vídeo podem ter uma distribuição de submodos semelhante. Entretanto, ainda é prematuro afirmar categoricamente que todas as sequências de vídeo apresentariam resultado semelhante.

Os outros resultados ligados à redução de complexidade, referentes aos métodos que usam mapas de diferenças, foram insatisfatórios. Apesar de não serem esperados resultados tão bons quanto os anteriores, foi muito importante fazer esses testes para fins de validação dos métodos. Verificou-se que, ao mesmo tempo em que bastante rápidos, esse métodos são também muito limitados, o que é comprovado pelo reduzidíssimo número de macroblocos codificados sob suas partições.

5.3 TRABALHOS FUTUROS

Este trabalho mostrou que mesmo em novas técnicas já inseridas em padrões de codificação de vídeo, há espaço para melhorias. Além disso, mostrou também a possibilidade de se ter bons ganhos, sem ser necessário aumentar tanto a complexidade da estimação de movimento. Por isso, vários tópicos podem ser sugeridos para trabalhos subsequentes a este.

- Como as partições *wedge* já foram implementadas para blocos de tamanho 8x8 [26], existe a possibilidade de melhorias pela implementação, para blocos menores, da predição de vetor de movimento mais abrangente aqui proposta.
- Pode ser feito um estudo mais aprofundado sobre a possível correlação entre determinadas características de macroblocos e submodos *wedge*, visando a eliminação da necessidade de busca entre todos os submodos disponíveis.
- Cabe também um estudo sobre melhores formas de codificar os resíduos para as partições *wedge* pelo uso de transformadas direcionais ou transformadas adaptável a formas (ou, em inglês, *shape-adaptive transform*).

REFERÊNCIAS BIBLIOGRÁFICAS

[1] HUNG, E. M. *Compensação de Movimento Utilizando Blocos Multi-Escala e Forma Variável Em Um Codec de Vídeo Híbrido*. Dissertação (Mestrado) — Universidade de Brasília, 2007.

[2] INTERNATIONAL TELECOMMUNICATIONS UNION. *ITU-T Recommendation H.264 : Advanced video coding for generic audiovisual services*. Maio 2003.

[3] ADACHI, S. et al. Refined results on the low-overhead prediction modes, ITU-T Q.6/SG16 VCEG, VCEG-O22. Dezembro 2001.

[4] GONZALEZ R. E. WOODS, S. L. E. R. C. *Digital Image Processing using Matlab*. [S.l.]: Prentice-Hall, 2004.

[5] INTERNATIONAL TELECOMMUNICATIONS UNION. *ITU-R Recommendation BT.601-5: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*. Outubro 1995.

[6] INTERNATIONAL TELECOMMUNICATIONS UNION. *ITU-T T.81 - Information Technology - Digital Compression and Coding of Continuous-Tone Still Images - Requirements and Guidelines*. Setembro 1992.

[7] INTERNATIONAL TELECOMMUNICATIONS UNION. *ITU-T Recommendation H.262 : Information technology - Generic coding of moving pictures and associated audio information: Video*. Julho 1995.

[8] EUROPEAN TELECOMMUNICATIONS STANDARDS INSTITUTE. *Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in satellite, cable and terrestrial broadcasting applications*. Janeiro 1996.

[9] GIROD, B. et al. Distributed video coding. *Proceedings of the IEEE*, v. 93, n. 1, p. 71–83, Janeiro 2005.

[10] HUANG, Y.-Y. et al. Analysis and complexity reduction of multiple reference frames motion estimation in H.264/AVC. *IEEE Trans. on Circuits and Systems for Video Technology*, v. 1, n. 16, p. 507–522, Abril 2006.

[11] INTERNATIONAL Telecommunication Union. Último acesso em 18/12/2008. `http://www.itu.int/net/home/index.aspx`.

[12] INTERNATIONAL Organization for Standardization. Último acesso em 18/12/2008. `http://www.iso.org/iso/home.htm`.

[13] INTERNATIONAL TELECOMMUNICATIONS UNION. *ITU-T Recommendation H.261 : Video codec for audiovisual services at p x 64 kbit/s*. Novembro 1988.

[14] INTERNATIONAL TELECOMMUNICATIONS UNION. *ITU-T Recommendation H.263 : Video coding for low bit rate communication*. Março 1996.

[15] SAYOOD, K. *Introduction to Data Compression*. [S.l.]: Morgan Kaufmann, 2000.

[16] PEIXOTO, E. *Transcodificador de vídeo Wyner-Ziv/H.263 para comunicação entre dispositivos móveis*. Dissertação (Mestrado) — Universidade de Brasília, 2008.

[17] WIEGAND, T. et al. Overview of the H.264/AVC video coding standard. *IEEE Trans. on Circuits and Systems for Video Technology*, v. 13, n. 7, p. 560–576, July 2003.

[18] RICHARDSON, I. E. G. *H.264 and MPEG-4 video compression*. [S.l.]: Wiley, 2003.

[19] COVER, T. M.; THOMAS, J. A. *Elements of Information Theory*. [S.l.]: Wiley, 2006.

[20] TAN, T. K.; SULLIVAN, G.; WEDI, T. Recommended simulation common conditions for coding efficiency experiments revision 1, ITU-T Q.6/SG16 VCEG, VCEG-AE010. Janeiro 2007.

[21] MACCHIAVELO, B.; QUEIROZ, R. L. de; MUKHERJEE, D. Parameter estimation for an H.264-based distributed video coder. *Proc. IEEE Intl. Conf. on Image Processing, ICIP*, San Diego, USA, Outubro 2008.

[22] HUNG, E. M.; QUEIROZ, R. L. de; MUKHERJEE, D. On macroblock partition for motion compensation. *In Proc of IEEE International Conf on Image Processing*, Atlanta, USA, Outubro 2006.

[23] DIVORRA Ò. et al. Geometry-adaptive block partitioning for video coding. *In Proc of IEEE International Conference on Acoustics, Speech and Signal Processing*, p. I–657,I–660, 2007.

[24] DONOHO, D. L. Wedgelets: Nearly-minimax estimation of edges. *Annals of Stat.*, v. 27, p. 859–897, 1999.

[25] ISO/IEC JTC1/SC29/WG11 e ITU-T SG16 Q.6. *JSVM 9.2 Software*. Abril 2006.

[26] DIVORRA Ò.; YIN, P.; GOMILA, C. Geometry-adaptive block partioning, ITU-T Q.6/SG16 VCEG, VCEG-AF10. Abril 2007.

[27] BJONTEGAARD, G. Calculation of average PSNR differences between RD-curves, ITU-T Q.6/SG16 VCEG, VCEG-M33. Abril 2001.