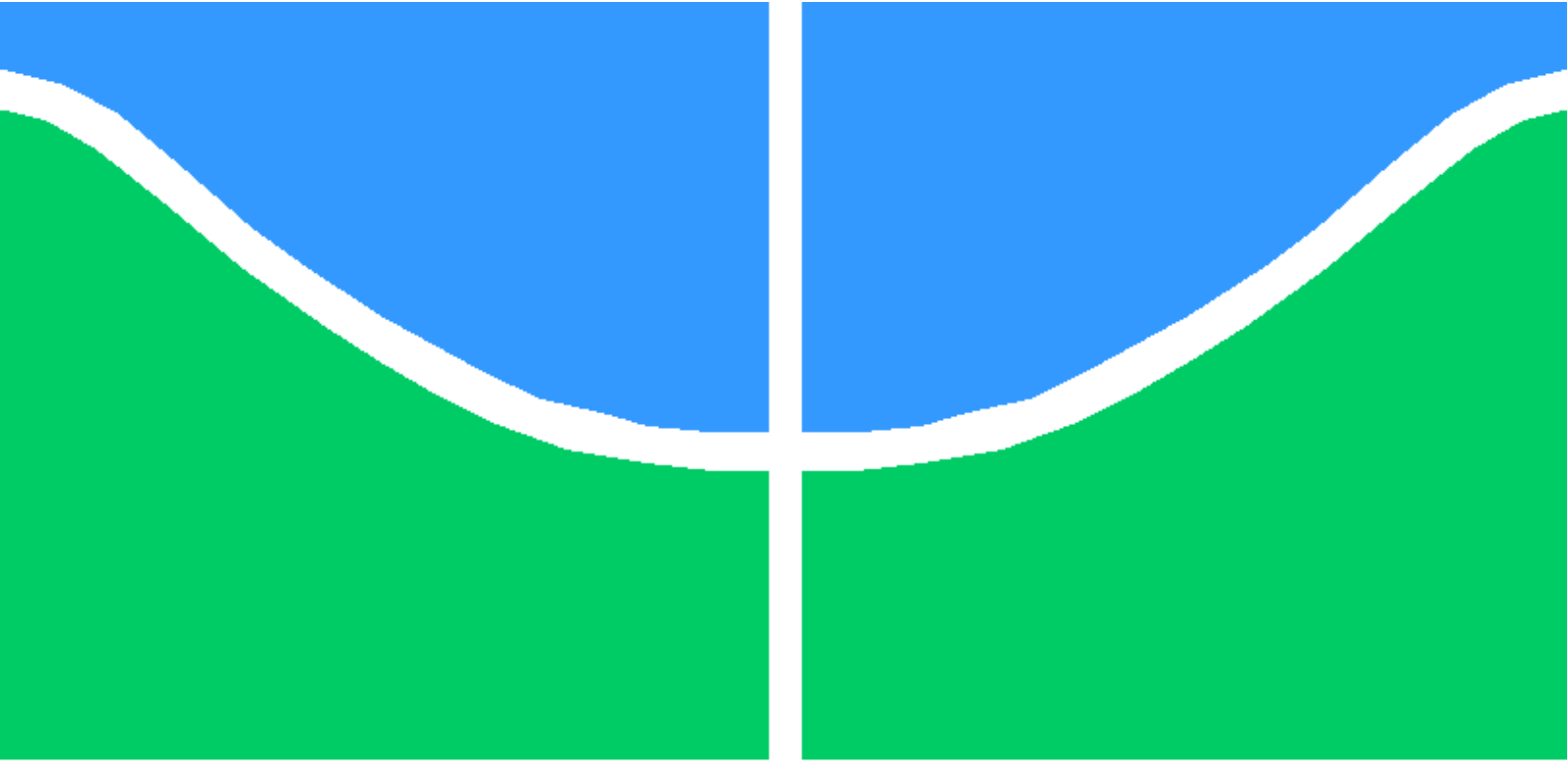

DISSERTAÇÃO DE MESTRADO EM
ENGENHARIA ELÉTRICA

COMPENSAÇÃO DE MOVIMENTO UTILIZANDO BLOCOS MULTI-ESCALA E FORMA VARIÁVEL EM UM CODEC DE VÍDEO HÍBRIDO

Edson Mintsu Hung

Brasília, julho de 2007

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

COMPENSAÇÃO DE MOVIMENTO UTILIZANDO BLOCOS MULTI-ESCALA E FORMA VARIÁVEL EM UM CODEC DE VÍDEO HÍBRIDO

Edson Mintsu Hung

Dissertação de mestrado 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.

Banca Examinadora

Prof. Ricardo Lopes de Queiroz, PhD. _____
UnB/ ENE (Orientador)

Prof. Francisco Assis de O. Nascimento, Dr. _____
UnB/ ENE (Examinador Interno)

Prof. Eduardo A. Barros da Silva, PhD. _____
UFRJ/ COPPE (Examinador Externo)

Data: Brasília, 24 de julho de 2007.

FICHA CATALOGRÁFICA

HUNG, EDSON MINTSU
Compensação de Movimento utilizando Módulos Multi-escala
e Forma Variável em um CODEC de Vídeo Híbrido. [Distrito Federal] 2007.
xiv, 73p., 297 mm (ENE/FT/UnB, Mestre, Telecomunicações
Processamento de Sinais, 2007). Dissertação de Mestrado.
Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Elétrica.

1. Compensação de movimento	2. Estimação de movimento
3. *quadtree*	4. *wedgelet*
5. H.264	6. MPEG
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

HUNG, E. M. (2007). Compensação de Movimento utilizando Módulos Multi-escala e Forma Variável em um CODEC de Vídeo Híbrido. Dissertação de Mestrado em Engenharia Elétrica com enfase em Telecomunicações, Publicação PPGENE.DM - 304/07, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 73p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Edson Mintsu Hung.

TÍTULO DA DISSERTAÇÃO DE MESTRADO: Compensação de Movimento utilizando Módulos Multi-escala e Forma Variável em um CODEC de Vídeo Híbrido.

GRAU / ANO: Mestre / 2007

Edson Mintsu Hung
QNL 11 Blobo E casa 06
72.151-115 Taguatinga - DF - Brasil.

Dedicatória

À minha família, meu pai Hung Chao-Shiung, à memória de minha mãe Yang Yu-Chu Hung, meus irmãos Alexandre Hung e Augusto Hung, e minha sobrinha Júlia Hung, por participarem da minha vida. Aos meus amigos, pela compreensão.

Edson Mintsu Hung

“Se você tem uma laranja e troca com outra pessoa que também tem uma laranja, cada um fica com uma laranja. Mas se você tem uma idéia e troca com outra pessoa que também tem uma idéia, cada um fica com duas.”

Confúcio

Agradecimentos

*Agradeço à Universidade de Brasília (**UnB**) por proporcionar uma ambiente favorável de aprendizado - em especial ao Departamento de Engenharia Elétrica (**ENE**), pela qualidade dos **docentes** e **funcionários** que me acompanharam diariamente. Em destaque os professores que me incentivaram Francisco Assis Oliveira Nascimento, Geovany Araújo Borges, Ricardo Zelenovsky, Anderson Nascimento e Adson Ferreira da Rocha.*

*Um agradecimento especial ao meu professor **orientador** Ricardo Lopes de Queiroz, pessoa pelo qual tenho muita admiração, por acreditar na minha capacidade. Sua "paciência", dedicação e competência foram fundamentais para que esta dissertação se concretizasse.*

*Aos integrantes do Grupo de Processamento Digital de Sinais - **GPDS**, pelo apoio, amizade e companheirismo. Principalmente ao Rafael dos Santos Ortis, que sem ele o laboratório seria um caos - obrigado pelo apoio nos projetos em que participamos. Ao Marcus Vinícius Chaffim e ao Diogo Caetano, pelas músicas.*

Ao pessoal da área de compressão daqui do GPDS: Alexandre Zaghetto, Bruno Machiavello, Eduardo Peixoto, Fernanda Brandi, Karen França, Rafael Galvão, Renan Utida e Tiago Alves. E agregados como Bárbara (Débora) Macedo, Frederico Nogueira e Rodrigo Balzan. Agradeço a todos pelos momentos de distração, amizade, companhia, conversas, discussões, reflexões, piadas, estórias, causos e mais um monte de coisas.

Agradecimentos especiais à Fernanda Brandi, minha companheira de madrugada, por me acompanhar nos hábitos noturnos do laboratório e pelas palavras de amizade - sua alegria e bom humor são contagiantes :-) ao Tiago Alves por me ajudar, incentivar e apoiar aqui no GPDS, valeu; e ao Eduardo Peixoto por ter revisado grande parte deste texto, seus comentários e sugestões foram "massa".

*Aos pesquisadores da **HP Labs** - Palo Alto doutor Debargha Mukherjee e doutora Yuxin Liu (Zoe), obrigado pelas sugestões (que foram fundamentais nos projetos que são realizados aqui no GPDS) e críticas (que foram construtivas e necessárias para um bom desenvolvimento). E ao pessoal da **HP Brasil** Ricardo Pianta, Paulo Sá e Marcelo Thielo por garantir um bom ambiente de trabalho, pela interação com os pesquisadores da HP Labs, por meio do convênio com a UnB e da doação de equipamentos.*

*Aos integrantes da **Mux Engenharia**: Wagner Popov, Tiago Alves e Luis Prata, que sempre contaram com a minha ausência - os meus sinceros muito obrigado pela compreensão e pela paciência.*

*Aos meus **amigos** de longa data um forte abraço. Peço ainda desculpa pela(s) ausência(s)... aos amigos que fiz ao longo da jornada aqui na UnB e aos amigos mais recentes agradeço pelo apoio, mesmo estando distantes, né Maíra?!? Que mesmo não fazendo parte deste universo acadêmico, vocês foram fundamentais para garantir minha lucidez.*

*Ao pessoal do **Só Alegria Moto Clube** pela receptividade e amizade, mesmo eu não tendo moto (...ainda).*

*Por fim, agradeço a receptividade e amizade das famílias Cunha, Silva, Santos, Oliveira, Carpaneda, Reis e Brandi. E à minha **família** Hung, que me apoiou durante toda vida. Agradeço por tudo, tanto pelos momentos bons, ruins e os mais difíceis.*

Edson Mintsu Hung

RESUMO

O padrão H.264 possui compensação de movimento entre quadros, na qual cada quadro é dividido em macroblocos. Cada macrobloco pode ser partido em submacroblocos na forma de quadrados ou retângulos em uma decomposição *quadtree*. Este trabalho estuda alguns métodos de compensação baseados em *wedges* (cunhas), ou seja, dividindo os macroblocos ou submacroblocos em duas regiões por um segmento de linha arbitrário. Esta técnica permite que o formato destas regiões acompanhe melhor as bordas dos objetos que se movem em uma cena. Todavia, existe um grande número de possibilidades de se particionar um macrobloco. Comparar qual delas seria a que oferece melhor relação distorção/taxa seria computacionalmente muito intenso, uma vez que para cada partição os cálculos de estimação de movimento devem ser refeitos. Portanto, um algoritmo mais rápido foi proposto. A idéia principal seria a de selecionar um conjunto de partições que possuem o segmento de reta com orientação similar às bordas da imagem contida no macrobloco. Finalmente, uma comparação entre alguns métodos de partição de macroblocos para compensação de movimento, indicando um desempenho melhor para as compensações de movimento baseado em partição com *wedges*.

ABSTRACT

In the H.264/AVC video coding standard, motion compensation can be performed by partitioning macroblocks into square or rectangular sub-macroblocks in a quadtree decomposition. This work studies a motion compensation method using wedges, i.e. partitioning macroblocks or sub-macroblocks into two regions by an arbitrary line segment. This technique allows the shapes of the divided regions to better match the boundaries between moving objects. However, there are a large number of ways to slice a block and searching exhaustively over all of them would be an extremely computer-intensive task. Thus, a faster algorithm which detects the predominant edge orientations within a block in order to pre-select candidate wedge lines was proposed. Finally a comparison among macroblock partition methods is performed, which points to the higher performance of the wedge partition method.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS, ABREVIAÇÕES E ACRÔNIMOS

Abreviações, Acrônimos e Siglas

H.264	Padrão internacional de compressão de vídeo, atualmente é o estado da arte
VCEG	*Video Coding Experts Group*
MPEG	*Motion Picture Experts Group*
JPEG	*Joint Photographic Experts Group*
JVT	*Joint Video Team*
DVD	*Digital Versatile Disk*
AVC	*Advanced Video Coding*
ITU	*International Telecommunication Union*
SAP	*Secondary Audio Program*
FRExt	*Fidelity Range Extensions*
FIR	*Finite Impulse Response*
CABAC	*Context-Based Adaptive Binary Arithmetic Coding*
CAVLC	*Context-Adaptive Variable Length Coding*
VLC	*Variable Length Code*
ASO	*Arbitrary Slice Order*
FMO	*Flexible Macroblock Order*
MBAFF	*Macroblock Adaptive Switching Between Frame and Field*
QP	*Quantization Parameter*
URQs	*Uniform-Reconstruction Quantizers*
MV	*Motion Vector*
SD	*Standard Definition*
HD	*High Definition*
SAP	*Secondary Audio Program*
SNR	*Signal to Noise Ratio*
PSNR	*Peak Signal to Noise Ratio*
MSE	*mean square error*
EBCOT	*Embedded Block Coding with Optimized Truncation*
EZW	*Embedded Zerotree Wavelet*
SPIHT	*Set Partitioning in Hierarchical Trees*
SDTV	*Standard Definition Television*
HDTV	*High Definition Television*
NAL	*Network Abstraction Layer*
Mbps	Mega bits por segundos (1048576 bits por segundos)
GB	Giga bytes (1073741824 bytes)
DMS	*discrete memoryless source*
IEC	*International Electrotechnical Commission*
SD	*Standard Definition*
SIF	*Source Input Format*
FIR	*Finite Impulse Response*
JBIG	*Joint Bi-level Image experts Group*
DCT	*Discrete Cossine Transform*
IDCT	*Inverse Discrete Cossine Transform*
JM	*Joint Model*
JSVM	*Joint Scalable Video Model*
ISO	*International Standards Organization*

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Compressão é o processo de compactar informação em um tamanho menor que a original. Portanto, a compressão de vídeo nada mais é que a compactação ou a condensação de uma seqüência de vídeo digital em um número menor de bits que o vídeo original. Um vídeo digitalizado e sem compressão (*raw*) com a qualidade típica de televisão analógica possui uma resolução de 720×576 pixels (*picture element*), ou seja, 414720 pixels (no caso de utilizar 24 bits por amostra obtém-se 9953280 bits por quadro) produziria aproximadamente uma taxa de 284,765 Mbps (mega bits por segundo, onde um mega é igual a 2^{20}) para uma taxa de 30 quadros por segundo, o que inviabilizaria sua transmissão [2]. Caso a intenção fosse armazenar digitalmente o vídeo, ele ocuparia, nessas condições de qualidade, 35,59 MBps (mega bytes por segundo, onde um byte é igual a 8 bits) de vídeo. Um filme de 2 horas de duração ocuparia 250,28 GB (no caso, um giga é igual a 2^{30}) de espaço em disco, desconsiderando o áudio, legendas, etc. Considerando que cada DVD (*Digital Versatile Disk*) dupla face possui capacidade de 8,5 GB, seriam necessários 30 DVD's para armazenar sem compressão um único filme. Essa desvantagem praticamente eliminaria as vantagens da digitalização do vídeo, apesar do sistema digitalizado ter novos serviços, melhor recepção, entre outras.

O processo de captura de uma cena envolve amostrar o espaço e o tempo. Atribuindo ao espaço duas dimensões que formam uma imagem, cuja resolução é medida de acordo com o número de elementos da imagem (*pixels*). E ao tempo é associado uma dimensão onde são relacionados as capturas das imagens em um intervalo de tempo. A figura 1.1 ilustra as três dimensões de uma seqüência de vídeo.

Regiões contíguas na imagem normalmente não apresentam mudanças bruscas, tampouco a mesma região em quadros diferentes (isto é, referentes a tempos diferentes). Com exceção de mudança de cena e variações abruptas no conteúdo de uma cena. Ou seja, tipicamente as amostras espaciais e temporais em um vídeo são bastante correlacionadas entre si. Essas características são

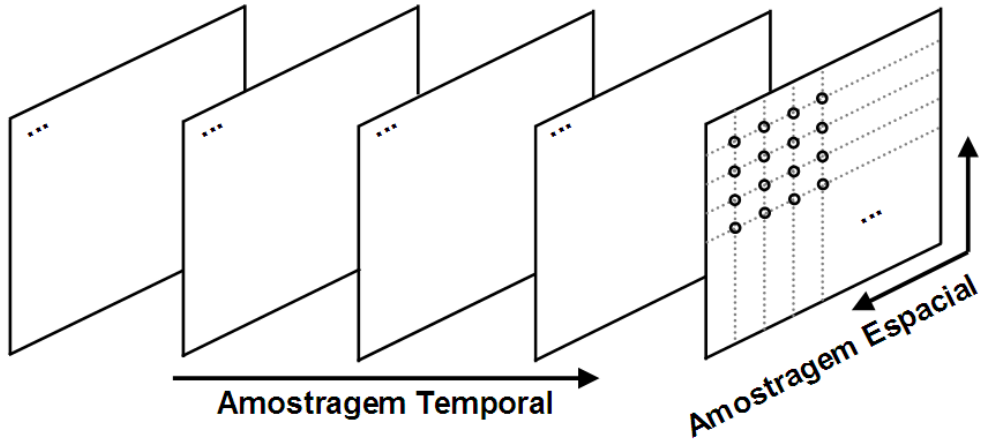

Figura 1.1: Seqüência de vídeo capturada e amostrada.

exploradas para que o vídeo digital seja representado e comprimido de maneira eficiente.

O processo inverso da compressão, ou seja, a descompressão, pode resultar em uma imagem idêntica à original, sendo esta compressão chamada de sem perdas (*lossless*). Se a descompressão gerar uma imagem semelhante à original, mas não idêntica, a compressão é dita com perdas (*lossy*). O primeiro tipo de compressão tem pouca utilização na área de transmissão de vídeo (exceto por estúdios onde se necessita de alta fidelidade), sendo o segundo processo utilizado por quase todos os algoritmos de compressão, por conseguir taxas de compressão muito maiores.

Observa-se um contínuo aumento da capacidade de transmissão de dados, seja pela Internet ou por outros meios de comunicação, além da alta capacidade de armazenamento em discos rígidos, memórias *flash* e meios ópticos. Em adição, verifica-se um custo de transmissão e armazenamento acessível e com valores decrescentes em relação ao tempo. Apesar de parecer um cenário paradoxal quanto à necessidade de compressão de vídeo, existem dois grandes benefícios. O primeiro é que na grande maioria dos canais de comunicação, os dados sem compressão (*raw*) não podem ser utilizados em tempo real, haja vista que o volume de dados a ser transmitido e recebido é tão grande que os dispositivos não suportariam dar vazão ao volume de informação. O segundo se refere à capacidade dos dispositivos de armazenamento. Por exemplo, um DVD é capaz de armazenar apenas alguns segundos de vídeo de alta definição e taxa de quadros (*frames*) no padrão da televisão, caso nenhuma técnica de compressão seja utilizada. Ou seja, a compressão de dados viabiliza a transmissão e o armazenamento dos dados de maneira mais rápida e eficiente, pois contém um número menor de dados e a “mesma” informação (se considerarmos que a

compressão de dados aplicada seja com perdas).

Para a televisão digital, tanto o sinal de vídeo quanto o de áudio são digitalizados e codificados utilizando técnicas eficientes de compressão de sinais. Os sinais codificados (codificação de fonte) e os serviços são multiplexados em um *stream* (fluxo de dados) que é recodificado (codificação de canal) e modulado para transmissão - vide Figura 1.2.

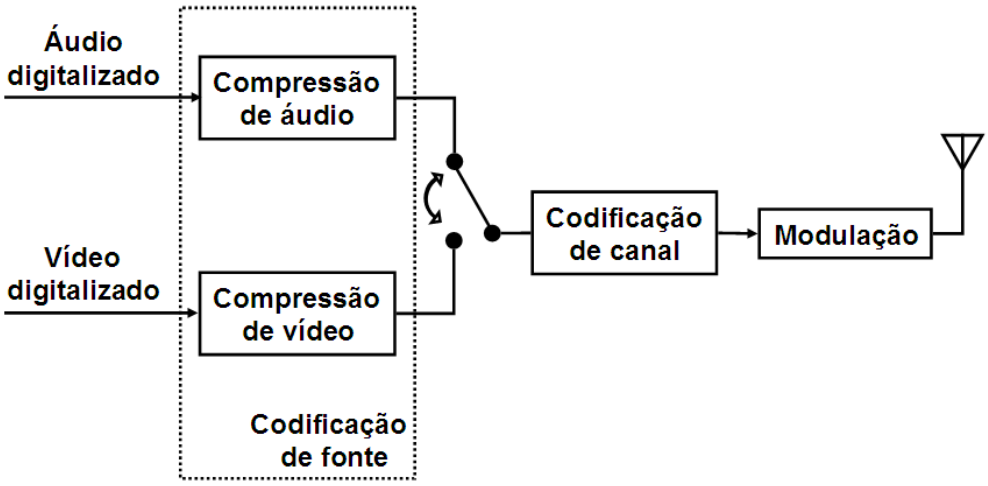

Figura 1.2: Diagrama de blocos de um emissor de sinais da televisão digital.

Já o esquemático da figura 1.3 ilustra um decodificador do sistema que seria um *set-top box* que decodificaria e demodularia o sinal de entrada, separaria áudio e vídeo dos serviços, descomprimiria os sinais de áudio e vídeo e, se for o caso, remodularia o sinal para ser enviado ao monitor. Tal monitor poderia ser um aparelho de televisão convencional ou um monitor de computador.

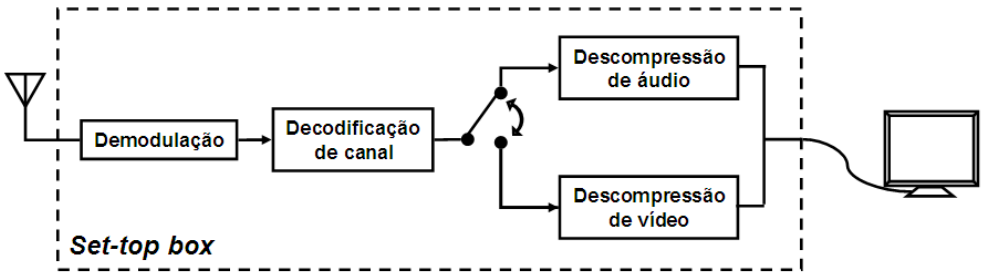

Figura 1.3: Diagrama de blocos de um receptor de sinais da televisão digital.

A codificação de vídeo é necessária para que se possa fazer compressão de dados que permita a transmissão da enorme quantidade de dados digitais de vídeo em canais com banda razoavelmente

limitada. Compressão de vídeo é uma necessidade e neste estudo nos concentraremos em dois padrões de compressão de vídeo: MPEG-2 [3] e H.264 [4]. O padrão MPEG-2 é o mais utilizado comercialmente, sendo ainda adotado por quase todos os provedores de televisão via satélite, por exemplo, SKY, DISH, DirecTV, e por várias TVs a cabo digitais no mundo inteiro. Além disto, o MPEG-2 é o algoritmo utilizado para comprimir vídeo em DVDs, garantindo ao codificador sua difusão e larga escala. O padrão H.264, também conhecido como *Advanced Video Coder* (AVC), ou como MPEG-4 *Part 10* é o estado da arte em compressão de vídeo - e é utilizado pela Apple Inc. nos softwares iTunes e QuickTime, além de ser utilizada como um dos padrões de compressão de vídeo no HD DVD e no *Blu-ray Disc*. É importante não confundir o H.264 ou AVC com o MPEG-4 em si (registrado como parte 2 no padrão). As inovações do H.264/AVC proporcionam um salto de eficiência de compressão comparado com MPEG-2 ou MPEG-4 [5]. As estatísticas apontam para o dobro de eficiência de compressão para uma mesma qualidade. Ou seja, o H.264 é capaz de comprimir duas vezes mais uma seqüência de vídeo do que o MPEG-2 ou do que MPEG-4. Entretanto, o AVC é bem mais complexo, em termos computacionais, que o MPEG-2. A tendência é que as operadoras de TV digital (cabo, satélite, *broadcast*, internet, celular, etc) migrem para a utilização do H.264.

1.2 DEFINIÇÃO DO PROBLEMA

Uma maneira de se comprimir um vídeo seria simplesmente comprimir um quadro (*frame*) por vez, como se fosse uma imagem utilizando técnicas específicas para tal propósito, como o JPEG [6] e o JPEG2000 [7] [8]. No caso do JPEG, a imagem é dividida em blocos de tamanho 8×8 e em seguida é aplicada a transformada de cossenos discreta (DCT), onde seus coeficientes são quantizados e transmitidos utilizando um código de Huffman. O JPEG 2000 segue uma abordagem um pouco diferente, onde para toda imagem a Transformada de *Wavelets* Discreta (DWT) é aplicada e seus coeficientes, inicialmente organizados em sub-banda, são quantizados por meio da técnica denominada na literatura como EBCOT - *Embedded Block Coding with Optimized Truncation* [9] (outras técnicas similares para codificação e quantização dos coeficientes podem ser exemplificadas pelo EZW - *Embedded Zerotree Wavelet* [10] e pelo SPIHT - *Set Partitioning in Hierarchical Trees* [11], entretanto não fazem parte do padrão

do JPEG2000). Em seguida os coeficientes são comprimidos entropicamente utilizando um codificador aritmético [8]. Os codificadores de vídeo totalmente baseados nos padrões de compressão de imagens JPEG e JPEG2000 são denominados como respectivamente como Motion-JPEG e Motion-JPEG2000. Note que esta "família" de codificadores não explora a redundância temporal contida em uma sequência de vídeo. Apesar de não possuir uma boa relação de qualidade de vídeo e taxa de compressão, esses compressores possuem esforço computacional reduzido, se compararmos com outros codecs.

O compressor de vídeo mais popular, utilizado atualmente nas TVs a cabo, nos DVDs e em *streaming* de vídeo na internet, se baseia no JPEG e foi desenvolvido por um grupo da ISO/IEC (*International Standards Organization/International Electrotechnical Commission*) denominado *Motion Picture Experts Group*, popularmente conhecido como MPEG. O MPEG [12] (MPEG-1 e MPEG-2) comprime a seqüência de vídeo explorando sua redundância temporal, diferentemente do que ocorre com o Motion JPEG, os quadros codificados são baseado em um GOP (*Group of Pictures*) do tipo IBBP isto significa que a primeira imagem é comprimida como um quadro "intra", ou seja, baseados apenas nas informações contidas no mesmo, em seguida é comprimido um novo quadro, baseando-se no quadro anteriormente comprimido, o que caracteriza um quadro do tipo P (Predito). O quadro P seria a predição do quadro seguinte ao estimar o movimento de blocos de tamanho 16×16 e em seguida comprimir a diferença entre o quadro estimado e o quadro original. Os quadros do tipo B (Bipreditos) são estimados e compensados a partir dos quadros do tipo I e do tipo P, por meio da busca de blocos de tamanhos fixos de 16×16. Observe a ilustração das figuras 1.4 e 1.5 note que a ordem de compressão dos quadros é diferente da sequência de exibição do mesmo. Isto permite a compensação de quadros baseados em quadros "futuros" tendo como parâmetro a seqüência de exibição (figura 1.4).

No caso do MPEG-1 e do MPEG-2 a predição é feita em blocos de 16×16 pixels. Para cada bloco a ser estimado, se procura em seu(s) quadro(s) de referência para cada bloco do tipo P ou B, um vetor de movimento associado ao mesmo, exceto nos casos de inferência (modo *skip*). Na Figura 1.6(a) se mostra um quadro de uma seqüência, e a Figura 1.6(b) se refere ao quadro seguinte. O movimento relativo dos objetos em cena é mostrado na Figura 1.6(c). Uma estimação de movimento, que seria o processo de procurar no quadro de referência o deslocamento bloco no quadro atual, gera os vetores de movimento que são ilustrados na Figura 1.6(d). Ao aplicar os

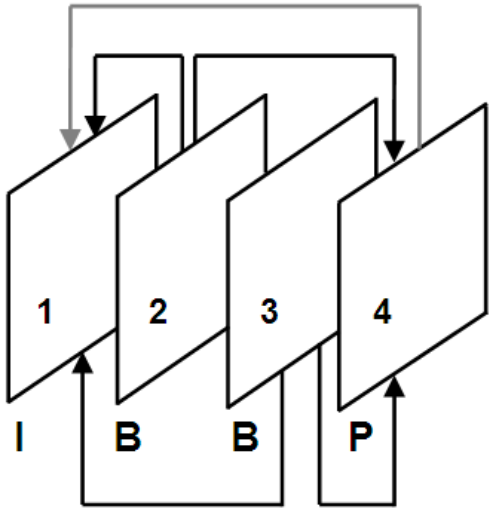

Figura 1.4: Quadros de compressão de vídeo ordenado de acordo com a seqüência de exibição.

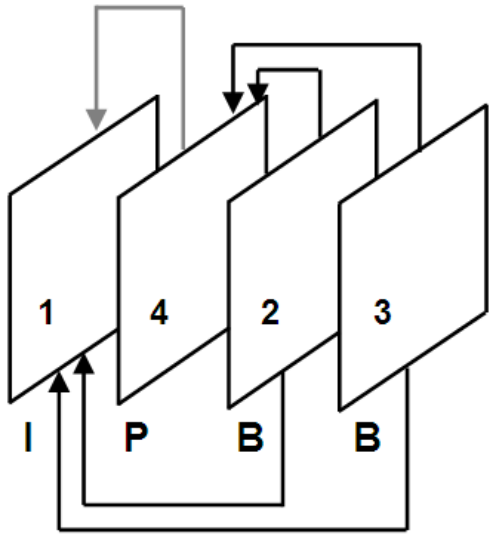

Figura 1.5: Quadros de compressão de vídeo ordenado de acordo com a seqüência de compressão.

vetores de movimento ao quadro anterior, teremos como resultado o quadro compensado (Figura 1.6(e)), que nem sempre é igual ao quadro atual. Ou seja, esse sistema de compensação pode resultar em um resíduo, ilustrado na Figura 1.6(f). Desta forma, a imagem da Figura 1.6(b) pode ser obtida caso sejam enviados apenas os vetores de movimento e o resíduo de compensação associado.

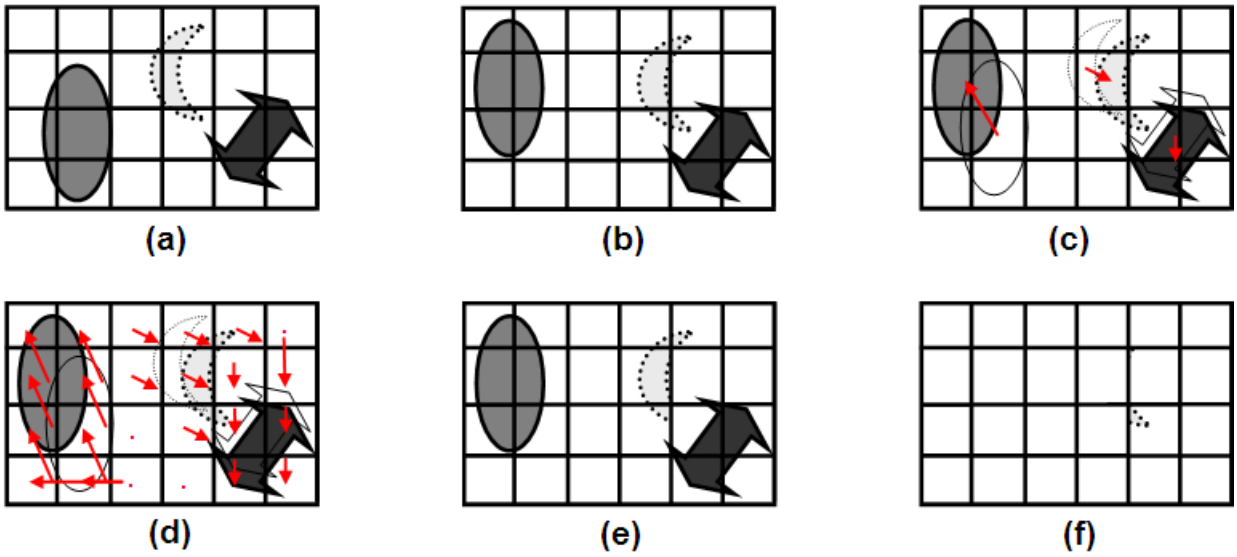

Figura 1.6: (a) Quadro n; (b) Quadro $n+1$; (c) Deslocamento dos objetos; (d) Vetores de movimento; (e) Quadro compensado; (f) Quadro residual.

Diferente do MPEG2, cujos blocos possuem tamanho fixo de 16×16 pixels para compensação de movimento, o H.264 [13] faz o uso de partições em seções retangulares de $N \times M$ pixels onde $(N, M) \in \{16, 8\}$ chegando até a particionar os sub-blocos de 8×8 em tamanhos de $N \times M$ onde $(N, M) \in \{8, 4\}$. Desta forma é permitida uma predição mais eficiente, pois quanto menor a região a ser estimada, menor a informação contida nela, e maior a chance de se obter um bloco no quadro de referência que se assemelhe com a dita região.

O resultado desta compensação, ou melhor, da predição entre quadros, é subtraída do bloco correspondente no quadro original, fornecendo um resíduo pelo qual se aplica uma transformada, cuja finalidade seria representar o bloco em valores numéricos (coeficientes) para que consigam ser transmitidos com maior eficiência. Utilizando o jargão de processamento de sinais podemos afirmar que a transformada compacta a energia do resíduo e também ajuda a descorrelacionar as amostras. Em seguida as amostras transformadas passam por uma etapa denominada quantização. A grosso modo seria a ferramenta matemática utilizada para se escolher

a quantidade de coeficientes significativos e a precisão da informação contida neles. A diferença entre os coeficientes originais e quantizados determina a qualidade da imagem ou do vídeo e, por conseqüência, da compressão do vídeo. Por fim, estes coeficientes são organizados e codificados entropicamente, ou melhor, são codificados de forma que os símbolos com maior ocorrência são representados com códigos menores e o contrário ocorre com símbolos de menor ocorrência. É importante que ressaltar que nessa etapa de codificação entrópica, a compressão é realizada de tal forma que nenhuma informação é perdida [14].

1.3 OBJETIVOS

O objetivo desta dissertação é propor técnicas de compressão de vídeo mais genéricas e mais eficientes comparadas às disponíveis no âmbito industrial e acadêmico. Para tanto foi desenvolvido um codec (codificador e decodificador) de vídeo cujos módulos são baseados nos padrões de vídeo MPEG-2 e H.264. O objetivo principal seria de testar e comparar as técnicas aqui propostas com as utilizadas classicamente nos codecs de vídeo dantes mencionado, e não propor um novo sistema de compressão que seja competitivo com o estado da arte em compressão de vídeo. Esta dissertação propõe métodos mais genéricos e eficientes de compensação de movimento, especificamente de partição de macroblocos. Ou seja, nesta dissertação apenas a correlação temporal (predição entre quadros) será explorada. Nos quadros comprimidos usando o modo “intra” far-se-á o uso do H.264, em modo “intra”.

Quando um macrobloco é partido, melhor e mais precisa é a compensação de movimento, comparado com compensação utilizando blocos maiores. Ou seja, obtém-se uma predição mais fidedigna com a informação original. Porém, um número maior de bits é gasto para informar ao decodificador qual partição foi escolhida além do número de bits utilizados para descrever o deslocamento relativo de cada região particionada. O resíduo resultante após a compensação é menor (em termos absolutos com relação à energia e a entropia do sinal). Sendo assim, ele pode ser comprimido utilizando um número menor de bits. Nesta dissertação estaremos discutindo a necessidade de partição dos macroblocos e quais tipos de partições de blocos são mais vantajosos.

1.4 APRESENTAÇÃO DO MANUSCRITO

No capítulo 2 é feita uma revisão bibliográfica sobre o tema de estudo, fundamentando de maneira sucinta todo o embasamento teórico necessário para o entendimento da dissertação. Como o tema da dissertação abrange uma gama enorme de assuntos, optou-se por utilizar uma descrição relativamente superficial dos padrões e das técnicas utilizadas para compressão de imagens e vídeos, sendo indicado ainda leituras complementares das referências. Em seguida, o capítulo 3 descreve com detalhes as técnicas padrões de partição de macroblocos para compensação de movimento juntamente com as técnicas propostas nesta dissertação. Resultados experimentais e são discutidos no capítulo 4, seguido das conclusões no capítulo 5.

2 COMPRESSÃO DE IMAGENS E VÍDEOS

2.1 INTRODUÇÃO

Os algoritmos de compressão de vídeo exploram algumas características inerentes a seqüências de imagens. Em particular a exploração da correlação espacial que se tem em um quadro (*frame*). Esse primeiro tratamento na informação é similar ao utilizado em compressão de imagens, no qual se aplica uma transformada (por sub-bandas ou em blocos - de modo a compactar a energia do sinal), uma quantização (esse parâmetro controla a qualidade e por conseguinte, a taxa da compressão), finalmente, aplica-se um codificador de entropia. Em seguida, explora-se a correlação temporal, uma vez que nos vídeos o conjunto de imagens em um pequeno intervalo de tempo geralmente possuem diferenças relativamente pequenas. A técnica mais comum seria mover um fragmento de um quadro anteriormente codificado, que pode ter tamanho ou formato fixos ou variáveis (maiores detalhes serão apresentados na Seção 3.2), utilizando o jargão de compressão de vídeo: compensar a partir da estimação de movimento. Esse processo resulta em um novo bloco ou quadro estimado a partir de um quadro ou bloco codificado anteriormente, onde para a reconstrução completa deste quadro são necessários: o(s) quadro(s) de referência, os vetores de movimento, a descrição dos particionamento dos blocos (se houver) e o resíduo - que seria a diferença entre o bloco ou quadro a ser comprimido e o estimado - caso exista.

2.2 ESPAÇO DE CORES YUV E AMOSTRAGEM 4:2:0

O sistema visual humano tende a perceber o conteúdo das cenas em termos de brilho e cores separadamente, e com maior sensibilidade aos detalhes de brilho comparados ao de cor [15]. De posse desta informação, os sistemas de transmissão de vídeo foram desenvolvidos de forma a tirar vantagem desta característica. Os codecs de vídeo mais populares (como o MPEG-1, MPEG-2, MPEG-4 e o H.264) utilizam o espaço de cor YCbCr (ou simplesmente YUV) juntamente

com uma redução da resolução (sub-amostragem) da informação de crominância Cb e Cr. A componente Y é denominada luminância, e representa o brilho. As outras duas componentes Cb e Cr representam o quanto a cor se distancia do eixo de luminância (escala de cinza) nos eixos azul-amarelo e vermelho-verde, respectivamente.

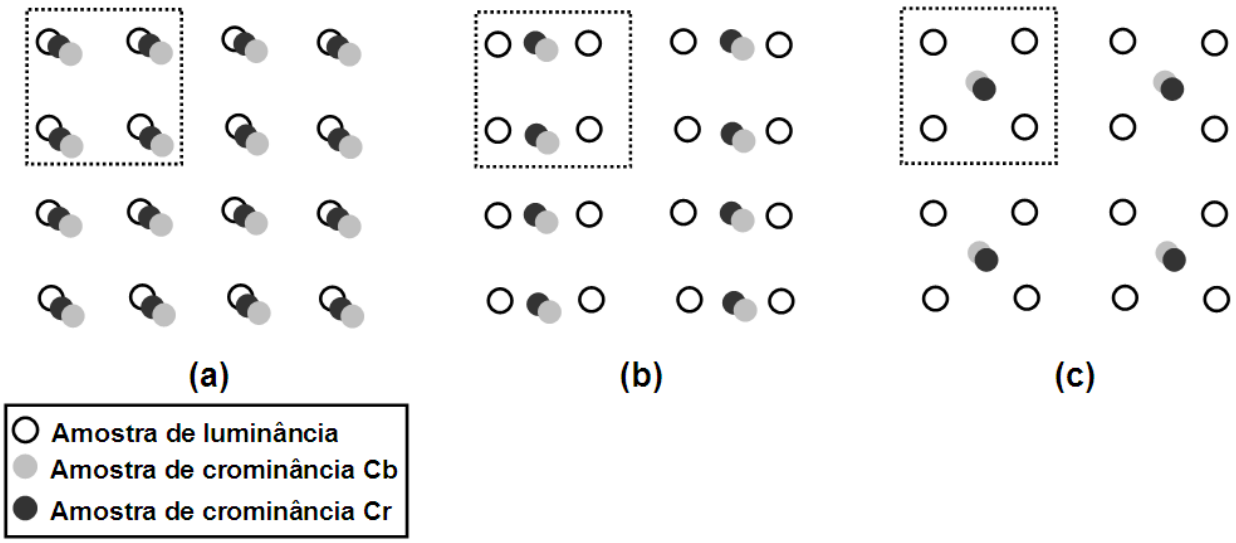

Figura 2.1: Amostras de luminância e crominância nos formatos (a) 4:4:4, (b) 4:2:2, (c) 4:2:0.

O sistema visual humano possui uma resolução maior à luminância do que à crominância. Por isso, o MPEG-1, MPEG-2 e H.264 na maioria dos perfis (que será detalhado na seção 2.9.8) se utiliza de uma estrutura de amostragem onde cada componente de crominância possui um quarto do número de amostras da componente de luminância (mais precisamente, metade do número de componentes tanto na horizontal quanto na vertical). Isso é denominado na literatura como um vídeo de amostragem 4:2:0 (Figura 2.1(c)). Tipicamente, são usados oito bits de precisão por amostra. Outras propostas de extensão do padrão do H.264 e vídeos de alta fidelidade permitem uma maior resolução de crominância (por exemplo 4:4:4 Figura 2.1(a) e 4:2:2 Figura 2.1(b)) e maior precisão de bits por amostra.

2.3 MÉTRICA DE QUALIDADE DE IMAGENS E VÍDEOS

A digitalização de um vídeo faz com que seja necessária uma compressão deste, que é usualmente feita aceitando perdas entre o vídeo comprimido e o vídeo original, a fim de se obter maiores taxas de compressão. Neste ponto, faz-se necessária a questão: até que ponto pode-se

aceitar essa perda na qualidade entre o vídeo comprimido e o vídeo original? Como medí-la? Em sistemas de comunicação a qualidade do sinal recebido é medida pela razão entre a potência do sinal e a potência do ruído, definindo a grandeza chamada SNR - *Signal to Noise Ratio*, ou relação sinal-ruído. Embora a SNR não seja perfeita, ela permite definir claramente se um determinado sinal recebido é melhor ou pior do que outro, além de permitir o conhecimeto (com precisão) da taxa de erro esperada em cada sinal. A SNR é uma medida objetiva, isto é, ela pode ser medida com precisão e não muda com a repetição do experimento, e amplamente utilizada em sistemas de comunicação, via rádio, cabo ou qualquer outro. Em sistemas de vídeo digital utiliza-se a medida chamada PSNR - *Peak Signal to Noise Ratio*, que poderia ser traduzida como relação sinal de pico ruído. A formulação da PSNR em decibéis é mostrada a seguir:

$$PSNR_{dB} = 10 \log_{10} \frac{(2^n - 1)^2}{MSE}, \tag{2.1}$$

onde MSE é o erro médio quadrático (*mean square error*) entre a imagem comprimida e a imagem original, e o numerador é o quadrado do valor máximo que um pixel pode assumir nessa imagem, dependente do número de bits por pixel. Por exemplo, se 8 bits são usados, o valor de pico seria 255.

Apesar de ser uma medida objetiva e de ser a métrica de qualidade mais utilizada, diferentemente da SNR em comunicações via rádio, a PSNR tem os seus opositores no campo de compressão de vídeo e imagens. Os argumentos mais comuns são: ela requer a imagem original para cálculo, o que nem sempre é possível, e às vezes contradiz a percepção subjetiva de qualidade. Apesar disso, em casos típicos, quanto maior a PSNR melhor é a percepção subjetiva da imagem, ou seja, a métrica objetiva da PSNR é correlacionada com a métrica subjetiva. A percepção subjetiva de qualidade se refere à como uma pessoa percebe a diferença entre as imagens. Isso pode depender muito de acordo com o uso que a pessoa fará da imagem, com a atenção que ela dá à imagem, com as experiências pessoais do observador, e muitas outras coisas, fazendo com que os resultados de uma medida subjetiva não possam ser repetidos nem aferidos com precisão.

A maior diferença de resultados entre a percepção subjetiva e a PSNR é que esta dá pesos

iguais a todos os pixels da imagem. Uma grande variação em um único pixel (ou região), em geral, não degrada muito a PSNR de uma imagem, enquanto o sistema visual humano é capaz de perceber essas variações. Exemplos comuns nesse tópico: se o valor de todos os pixels de uma imagem for subtraído por um valor unitário ou que tenham todos os pixels deslocados, a PSNR degrada muito, e um observador humano poderá afirmar que a imagem não foi deteriorada; no caso em que as áreas da imagem em que o observador geralmente foca ou observa com maior atenção possuírem alta qualidade, e as áreas de fundo (onde o observador normalmente não foca) estiverem degradadas, a PSNR diminui, mas o observador humano não vai mudar sua percepção; esta característica visual é denominada de foveação.

Apesar dos problemas encontrados na medida, a PSNR é ainda a medida mais utilizada de qualidade de imagem e vídeo encontrada, por ser um teste objetivo e de fácil medição, desde que se tenha a imagem (ou seqüência de imagens) original e a reconstruída.

2.4 COMPRESSÃO SEM PERDAS DE DE DADOS

O codificador de entropia é um tipo de compressor de informação sem perdas, extremamente dependente ou do modelo de probabilidades empregado ou da adaptabilidade do dicionário ou algoritmo, dependendo da técnica de compressão utilizada. Neste contexto converte uma série de símbolos que representam dados, pixels ou elementos da seqüência de vídeo (coeficientes de transformada quantizados, vetores de movimento, marcadores cuja função é indicar o ponto de ressincronização em uma seqüência, cabeçalhos e informações suplementares) em uma seqüência menor de bits, sendo mais adequada à transmissão ou ao armazenamento.

A codificação de entropia se utiliza de conceitos básicos da teoria de informação ou teoria estatística das comunicações [16], cuja fonte de informação pode ser vista como sendo um processo que gera uma seqüência de símbolos de um alfabeto finito. Portanto, os vídeos podem ser representados a partir de uma seqüência de pixels (*picture element*), e cada um deles são representados por um alfabeto finito, ou seja, os possíveis valores que podem ser atribuídos a cada pixel são valores inteiros entre 0 e 2^n, onde n é o número de bits que representa cada pixel. A ordem com que os pixels, amostras ou coeficientes referentes a uma imagem são percorridos

(reorganizados) podem resultar em uma fonte ou seqüência de símbolos que transforma a natureza estatística dos símbolos, reduzindo significativamente o conteúdo da informação espacial.

Os modelos de fontes de informação mais comuns em codificadores de vídeo: fontes discreta sem-memória (*discrete memoryless source*-DMS) e fontes de Markov. Os códigos de tamanho variável (*variable length coding*-VLC), com exceção dos códigos adaptativos, são baseados no modelo DMS e códigos preditivos são baseados no modelo Markoviano.

No caso dos DMS, cada símbolo da fonte é gerado independentemente, tornando-os estatisticamente independentes. A fonte é completamente definida pela relação entre símbolos e eventos e pela probabilidade de ocorrência de cada símbolo. $E = \{e_1, e_2, ..., e_n\}$ representaria os símbolos e o conjunto $\{p(e_1), p(e_2), ..., p(e_n)\}$ representaria a probabilidade de ocorrência de cada símbolo e n o número de símbolos do alfabeto.

Outro conceito importante é o da entropia, que é definido como sendo a média da informação contida na fonte. O conteúdo individual de cada evento ou símbolo é definido como

$$I(e_i) = \log \left\{ \frac{1}{p(e_i)} \right\}, \tag{2.2}$$

onde a base do logaritmo é determinada pelo número de estados utilizados para representar a informação da fonte. Ou seja, para fontes de informações digitais utiliza-se a base 2 para se obter o conteúdo da informação em bits por símbolo, ou taxa de bits. A entropia do sinal digital é definida como a média da informação contida na fonte. Duas técnicas de codificação de entropia amplamente usadas, são, a codificação de comprimento variável e a codificação aritmética. A entropia da fonte é definida por [16]

$$H(E) = \sum_{i=1}^{n} p(e_i) I(e_i) = -\sum_{i=1}^{n} p(e_i) \log_2 \{p(e_i)\} \left[bits\ por\ simbolo\right]. \tag{2.3}$$

A entropia quantifica a média de bits por símbolo necessária para representar a informação contida na fonte. O teorema de codificação de fontes sem ruído afirma que uma fonte pode ser codificada com uma média de bits por símbolo muito próximo, mas não menor que a da entropia da fonte. Logo, os codificadores de entropia procuram utilizar códigos cujo desempenho se aproxime da entropia da fonte.

A codificação de comprimento variável utiliza códigos instantâneos que consistem em mapear os símbolos que entram no codificador de entropia em uma série de palavras-código de comprimento variável (VLC's). Logo, cada uma delas deve conter um número inteiro de bits. Os símbolos com maior freqüência de ocorrência são representados com códigos menores e o contrário ocorre com símbolos de menor freqüência. Apesar de ter um desempenho satisfatório, a codificação de comprimento variável possui uma limitação de associar um valor inteiro de bits a um símbolo - impedindo a compressão de aproximar da entropia [14]. No caso do JPEG, o código de Huffman [17] é utilizado para sua compressão entrópica, que seria um caso clássico de códigos de tamanho variável. Por isso, a codificação aritmética torna-se uma alternativa à codificação de comprimento variável, dado que, com ela, é possível que se chegue mais perto das taxas de compressão teóricas [2, 18], visto que um codificador aritmético é capaz de converter uma seqüência de símbolos de dados em um único número fracionário.

Como a eficácia da codificação de entropia depende, em grande parte, do modelo de probabilidade de ocorrência de símbolos usados, a codificação aritmética baseada no contexto, que usa características locais para estimar a ocorrência (ou a probabilidade de ocorrência) de cada símbolo a ser codificado, se torna um dos codificadores de entropia de melhor desempenho a ser utilizado atualmente, sendo inclusive adotado em vários codificadores de imagens e vídeos como o JBIG2 [19], JPEG2000 [7] e o H.264 [4].

2.5 COMPRESSÃO ESPACIAL (COM PERDAS)

Para se comprimir cada quadro de um vídeo, utilizam-se técnicas similares às técnicas de compressão de imagens, como o JPEG. A compreensão de como funciona essa técnica ajudará no entendimento do funcionamento de codecs mais complexos. A técnica JPEG divide a imagem em blocos de 8×8 pixels. Se a imagem não possui linhas e colunas com dimensões múltiplas de 8 as últimas colunas ou linhas são repetidas para que se obtenha essa característica. Um exemplo dessa divisão pode ser visto na Figura 2.2:

A matriz B (2.4) mostra um bloco de pixels dessa imagem que será usada como exemplo. Em

cada bloco 8×8 são feitas uma série de operações, exemplificadas a seguir.

$$B = \begin{bmatrix} 104 & 108 & 107 & 101 & 94 & 95 & 98 & 102 \\ 96 & 100 & 103 & 100 & 96 & 74 & 75 & 73 \\ 77 & 69 & 70 & 87 & 84 & 64 & 64 & 67 \\ 71 & 60 & 52 & 59 & 64 & 56 & 54 & 57 \\ 58 & 53 & 51 & 54 & 52 & 51 & 52 & 52 \\ 53 & 50 & 53 & 52 & 52 & 58 & 51 & 47 \\ 48 & 53 & 53 & 51 & 53 & 55 & 51 & 53 \\ 47 & 48 & 48 & 47 & 55 & 47 & 51 & 48 \end{bmatrix} \tag{2.4}$$

Inicialmente, a imagem é subtraída de 128 (no caso de imagens de 8 bits), de forma que os pixels variem entre $[-128; 127]$. Para cada bloco é aplicada a transformada discreta de cosseno, DCT que seria uma decomposição em bases senoidais como ilustra a Figura 2.3. Em seguida, os valores são multiplicados por 8 gerando os coeficientes mostrados na matriz Θ (2.5) que são ponderações das bases mostradas na Figura 2.3. Note que os coeficientes mais próximos do topo, à esquerda, são os coeficientes das freqüências mais baixas.

$$\Theta = \begin{bmatrix} -495,5000 & 19,8185 & -7,8394 & 0,0189 & 10,0000 & -0,9774 & -3,2472 & 2,7423 \\ 134,5957 & 21,6732 & -2,7443 & -8,5780 & 7,3955 & 1,4963 & 3,0981 & 0,2423 \\ 58,8497 & 1,3480 & -0,8839 & -10,4132 & -9,1595 & -2,9341 & -0,6339 & 2,9325 \\ 17,4016 & -3,4000 & 8,5789 & -2,6026 & -13,8331 & 1,2787 & 6,4987 & -4,3676 \\ -4,5000 & -6,9291 & 14,3858 & 2,8536 & -1,5000 & -0,3074 & -0,5468 & -0,0110 \\ 2,1127 & -10,2568 & 7,1211 & 3,3039 & 0,4213 & -1,9182 & 2,3304 & -4,1175 \\ -1,6461 & -9,0033 & -1,1339 & 3,2009 & 2,7116 & 3,4405 & 0,8839 & -1,5484 \\ 0,6826 & -6,7326 & -0,4247 & -4,3075 & 2,4544 & 1,5464 & -1,1040 & -2,1524 \end{bmatrix} \tag{2.5}$$

Em seguida, os coeficientes da DCT são quantizados, segundo uma matriz de quantização. Uma das tabelas de quantização utilizada pelo JPEG é mostrada na matriz (2.7) que foi desenvolvida empiricamente, direcionada pelo sistema visual humano (HSV - *Human Visual System*). Os coeficientes são quantizados dividindo-os pela tabela de quantização, adicionando $0,5$ e truncados, conforme descreve a equação (2.6).

Figura 2.2: Imagem dividida em blocos de 8×8 pixels.

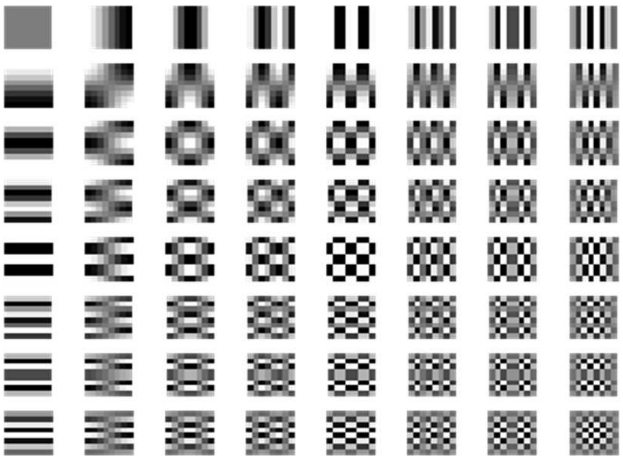

Figura 2.3: Bases da DCT.

$$l_{(i,j)} = \left\lfloor \frac{\theta_{(i,j)}}{q_{(i,j)}} + 0,5 \right\rfloor, \tag{2.6}$$

onde $l_{(i,j)}$ são os coeficientes transformados e quantizados, $\lfloor \cdot \rfloor$ é a operação de truncamento, $\theta_{(i,j)}$ são os coeficientes transformados, $q_{(i,j)}$ são os coeficientes da matriz de quantização e (i, j) são as posições dos elementos na matriz Q (equação (2.7)).

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{2.7}$$

Existem várias tabelas de quantização possíveis no formato JPEG. Cada uma delas gera uma relação diferente entre a compressão e a qualidade desejada. Por exemplo, se fosse utilizada uma matriz de quantização cujo denominador da Equação 2.6 for unitário, se obteria uma maior qualidade, porém uma menor compressão. Observe que na matriz de quantização utilizada, os coeficientes ficam maiores à medida que se afastam da posição do coeficiente DC (topo, à esquerda), indicando que as freqüências mais baixas são priorizadas.

$$L = \begin{bmatrix} -31 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 11 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.8}$$

Os coeficientes quantizados são mostrados na matriz L (equação (2.8)). Note que os coeficientes quantizados possuem uma grande quantidade de zeros. Percorrendo os coeficientes da matriz quantizada em ordem *zig-zag*, conforme mostra a Figura 2.4, obteremos no final desse processo um vetor. Isto implica na geração de sequência de valores iguais o que torna intuitivo de se comprimir utilizando a técnica de *zero-run*[14, 6], onde os coeficientes são descritos pela quantidade e o valor. Além da utilização de códigos especiais, como *End of Block*, para sinalizar que todos os coeficientes seguintes são nulos - o que implica em um grande aumento da eficiência desta técnica. Finalmente, o padrão JPEG utiliza o código de Huffman, com uma tabela, que associa o valor dos coeficientes a um código, específica conhecida tanto pelo codificador quanto pelo decodificador.

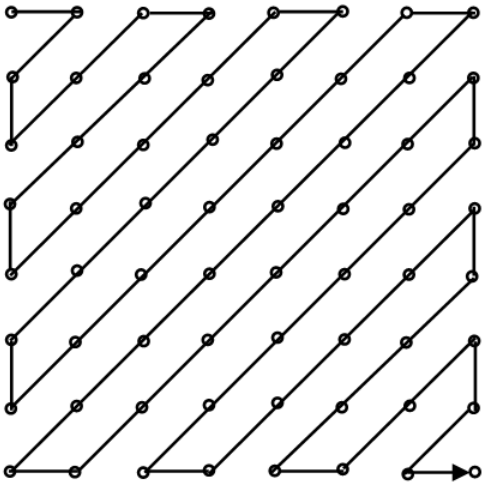

Figura 2.4: Reordenação dos coeficientes da DCT utilizando o *zig-zag*.

A descompressão é simples: decodifica-se o código de Huffman, gerando novamente a tabela dos coeficientes quantizados. Nesse ponto, basta aplicar a transformada discreta inversa do cosseno (IDCT) e somar 128. O resultado pode ser visto na matriz B' (2.9).

$$B' = \begin{bmatrix} 108 & 108 & 106 & 104 & 100 & 95 & 91 & 85 \\ 96 & 95 & 94 & 92 & 89 & 84 & 81 & 78 \\ 78 & 78 & 77 & 76 & 73 & 69 & 69 & 64 \\ 63 & 64 & 64 & 63 & 61 & 58 & 55 & 53 \\ 55 & 55 & 56 & 56 & 55 & 53 & 51 & 49 \\ 51 & 51 & 53 & 53 & 52 & 52 & 51 & 49 \\ 48 & 49 & 51 & 52 & 53 & 52 & 51 & 50 \\ 47 & 48 & 50 & 51 & 52 & 52 & 51 & 51 \end{bmatrix} \tag{2.9}$$

A diferença entre o bloco reconstruído e o bloco original é mostrada na matriz D (2.10). A percepção visual dessa diferença dos pixels é muitas vezes pequena (dependendo do quantizador), fazendo com que o algoritmo apresente bons resultados (neste exemplo do bloco, a PSNR foi 34,27 dB).

$$D = \begin{bmatrix} -4 & 0 & 1 & -3 & -6 & 0 & 7 & 13 \\ 0 & 5 & 9 & 8 & 7 & 10 & -6 & -5 \\ -1 & -9 & -7 & 11 & 11 & -5 & -2 & 3 \\ 8 & -4 & -12 & -4 & 3 & -2 & -1 & 4 \\ 3 & -2 & -5 & -2 & -3 & -2 & 1 & 3 \\ 2 & -1 & 0 & -1 & -1 & 6 & 0 & -2 \\ 0 & 4 & 2 & -1 & 0 & 3 & 0 & 3 \\ 0 & 0 & -2 & -4 & 3 & -5 & 0 & -3 \end{bmatrix} \tag{2.10}$$

Estendendo a compressão do bloco para toda figura, teremos na Figura 2.5 o resultado da codificação de uma imagem. À esquerda é mostrada a imagem original, e à direita a imagem codificada com a tabela de quantização apresentada. A PSNR da imagem reconstruída foi de 35,78 dB.

2.6 *MOTION* JPEG E *MOTION* JPEG 2000

O Motion JPEG é um codec de vídeo que faz o uso da técnica de compressão de imagens do JPEG (descrito na seção 2.5), mas não se utiliza das predições entre quadros. A ausência de predições "inter" quadros resulta em uma ineficiência na capacidade de compressão, mas facilita a edição do vídeo, uma vez que as edições podem ser realizadas em qualquer quadro, diferentemente do MPEG-2 e do H.264 cujo conteúdo depende de informação de quadros anteriores e posteriores - o que torna mais difícil a visualização completa do quadro para a edição. Outros compressores como o MPEG-2 e o H.264 quando operados de forma que utilizem apenas quadros "intra" (I) possui facilidades similares para edição. Quando os compressores de vídeo utilizam apenas quadros "intra", também existe uma perda na capacidade de compressão - independentemente da quantidade de movimento dos objetos em cena - pois a correlação temporal explorada pela predição "inter" não é utilizada.

O Motion JPEG é muito utilizado em circuito fechado de televisão ou sistema de monitoramento por câmeras, cuja aplicação faz uso de pouca resolução temporal (em torno de 5 quadros por segundo). O motivo da escolha é que com a amostragem temporal baixa, geralmente se diminui a correlação temporal. Além disso, como os quadros são totalmente independentes entre si, intuitivamente o sistema é mais robusto à falhas na gravação, transmissão e exibição haja vista que o erro de um quadro não propaga ou interfere nos demais. De maneira análoga, o JPEG2000 possui uma variante (parte 3) do padrão de codificação de imagens, que é um codec de vídeo, conhecido como Motion-JPEG2000 [20], cujo núcleo de compressão é baseado no JPEG2000 Parte 1. Atualmente é utilizado em cinemas digitais, gravação e edição de vídeos de alta qualidade baseados em quadros, armazenamento de vídeo em câmaras digitais, imagens médicas e imagens de satélites.

2.7 PREDIÇÃO ENTRE QUADROS (TEMPORAL)

Tipicamente os quadros vizinhos de uma seqüência de vídeo são muito correlacionados, pois foram capturados em instantes de tempo muito próximos entre si. Se a câmara permanecer estática, todo o fundo (*background*) da imagem se mantém, e os únicos pixels diferentes são

aqueles que representam objetos que se moveram (ou objetos que apareceram atrás dos objetos que se moveram).

A forma utilizada para se diminuir a redundância entre quadros (inteiros ou parte deles) vizinhos é prever o quadro atual a partir dos quadros anteriormente codificados e reconstituídos (localmente decodificados). O método mais simples seria comparar o quadro atual ao anterior, calcular a diferença entre eles (chamada de resíduo) e codificar apenas essa diferença. O resíduo é codificado utilizando-se alguma técnica de compressão espacial, mas, como tem muito menos energia (informação) do que o quadro completo, o resultado da deste tipo de compressão (predição e compressão do resíduo) tende a ser mais eficiente. A Figura 2.6 ilustra esse processo.

2.7.1 Estimação de Movimento

Estimação de movimento é o processo realizado para encontrar os movimentos resultantes que ocorrem entre pelo menos dois quadros. Esse processo consiste em procurar o lugar mais provável onde um objeto presente num quadro (geralmente chamado atual) se localizava em outro quadro (anteriormente codificado e reconstruído) que seria o quadro de referência.

Neste método, o quadro atual é dividido em regiões - que poderiam ser tão pequenas quanto um pixel, porém este não é normalmente o caso devido ao grande esforço computacional exigido e do grande número de bits para se descrever o deslocamento dessa região (vetores de movimento) que necessitam ser codificados. A estimativa do movimento de cada região da imagem é chamada de *optical flow* (fluxo óptico) ou simplesmente vetores de movimento. Canonicamente, estima-se o movimento de blocos da imagem, por isso a técnica é chamada *block-based motion estimation* ou estimação de movimentos baseada em blocos. A compensação de movimento seria a aplicação dos vetores de movimento nos quadros de referência de modo a gerar uma predição do quadro (ou *slice* ou bloco) atual. O decodificador deve usar os vetores de movimento para criar o quadro compensado, decodificar o resíduo, e utilizar os dois para formar o quadro final que será exibido. A Figura 2.7(b) mostra o fluxo óptico entre quadros subsequentes.

O uso de blocos retangulares é muito popular, mas tem algumas desvantagens. Objetos reais em geral têm bordas mais complexas, que não acompanham as bordas retangulares dos blocos usados para compensação de movimento. Além disso, movimentos mais complexos como:

Figura 2.5: Imagens original (esquerda) e reconstruída (direita).

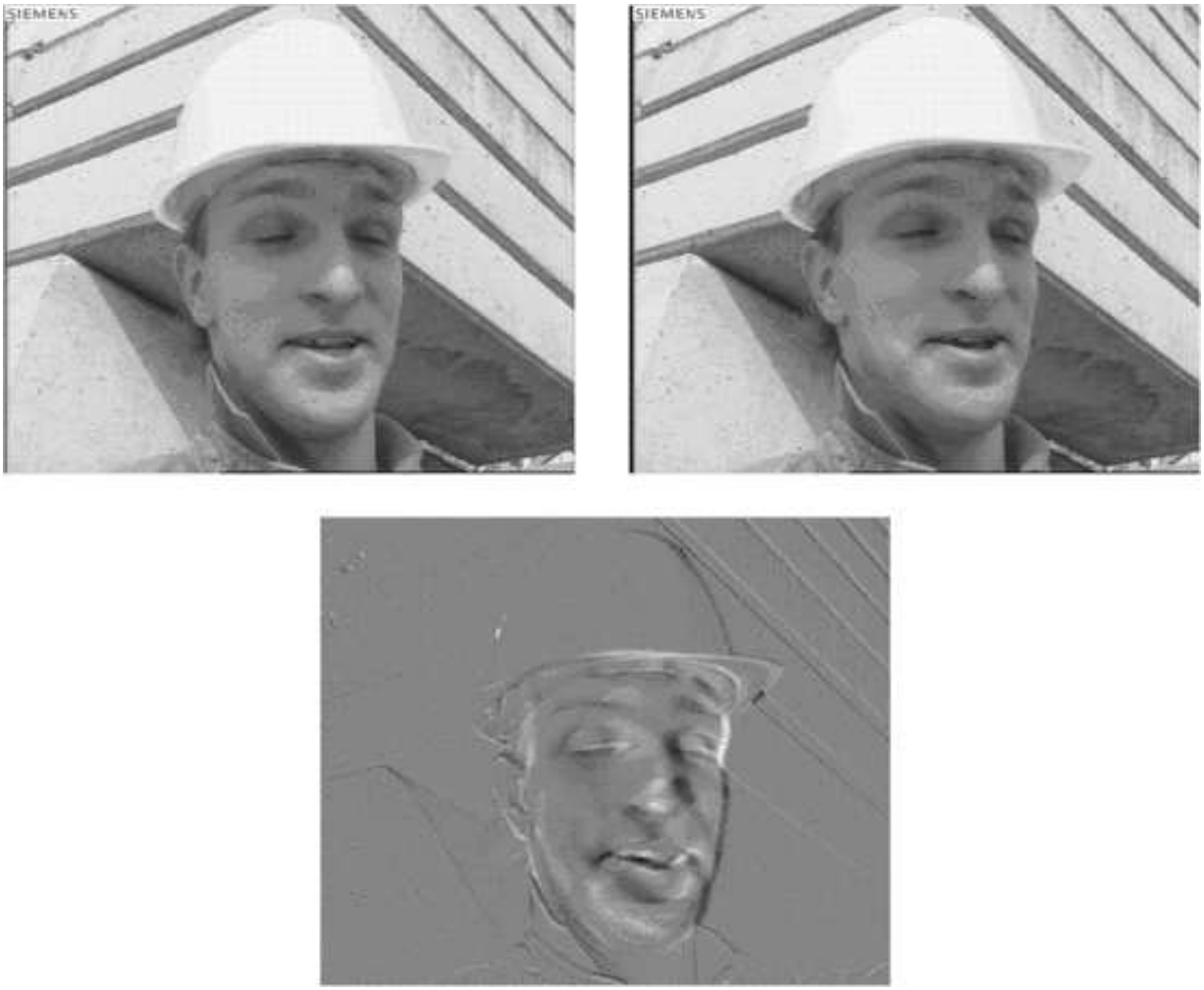

Figura 2.6: Quadros sucessivos e o resíduo entre eles.

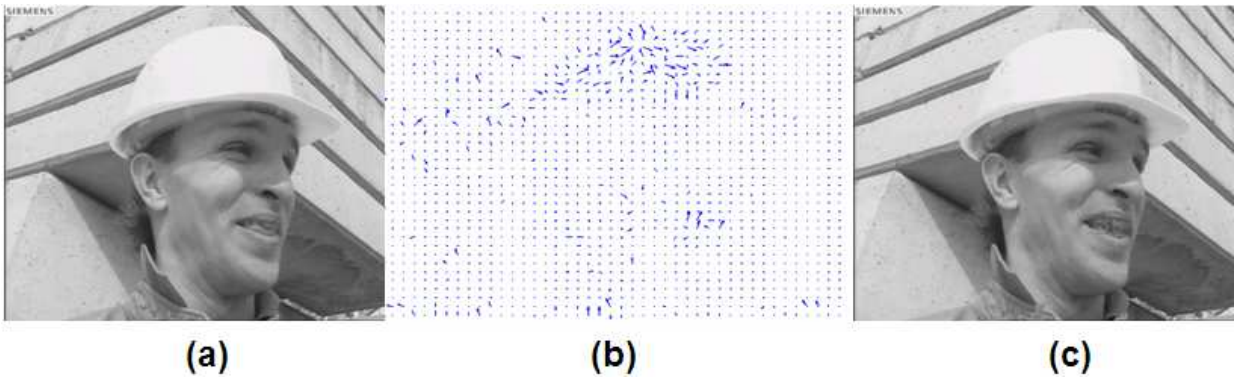

Figura 2.7: (a) Quadro de referência; (b) Fluxo óptico (c) Quadro compensado (estimado).

zoom, rotação deformações, fumaça, são difíceis de se estimar. Apesar disso, o fato de ser computacionalmente viável e ser compatível com transformadas baseadas em blocos, como a DCT, fez essa técnica ser utilizada por quase todos padrões de compressão de vídeo.

Na estimação de movimento (*Motion Estimation*) procura-se no quadro de referência (que pode estar no passado ou futuro, desde que já tenha sido codificado), a região que melhor represente o bloco atual. A área em que se procura o vetor de movimento no quadro de referência é limitada, pois normalmente os movimentos não são muito grandes na cena, e seria computacionalmente exaustivo procurar por todo o quadro.

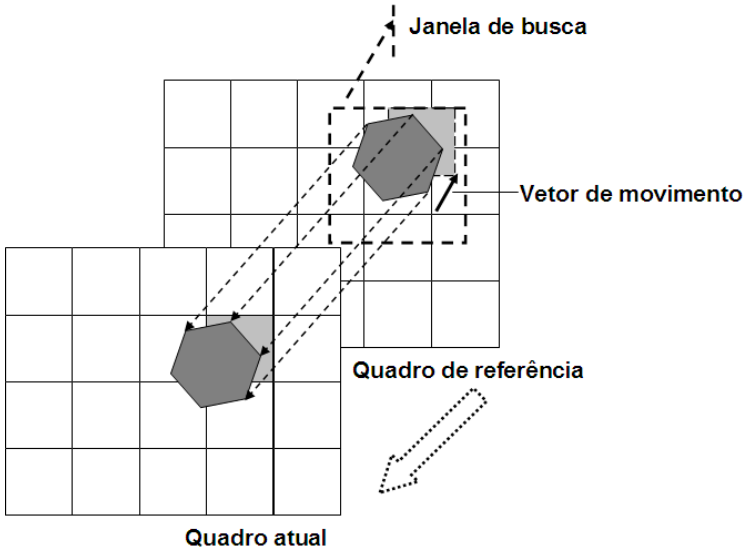

Figura 2.8: Estimação de movimento.

O critério de avaliação pode ser feito minimizando o MSE (*Mean Squared Error*, erro médio quadrático) ou minimizando a SAD (*Sum of Absolute Differences*, soma das diferenças absolutas)

ou minimizando a SSD (*Sum of Square Differences*, soma das diferenças quadráticas), ou até mesmo uma função de custo - que seria uma soma ponderada entre a distorção e a taxa de bits utilizada. A Figura 2.8, mostra dois quadros: o atual (a ser codificado) e o de referência (reconstruído) o deslocamento relativo de um bloco no quadro atual pode ser calculado pelas equações 2.11 e 2.12.

$$SAD = \sum_{i=1}^{N}\sum_{j=1}^{N} \left| pixel_{atual}(i,j) - pixel_{ref}(i+x, j+y) \right|, \tag{2.11}$$

$$SSD = \sum_{i=1}^{N}\sum_{j=1}^{N} \left(pixel_{atual}(i,j) - pixel_{ref}(i+x, j+y) \right)^2, \tag{2.12}$$

onde x e y representam os deslocamentos (vetores de movimento) da procura do bloco no quadro anteriormente codificado (referência). Para o cálculo de vetores de movimento varia-se os valores de (x, y) em torno de uma vizinhança ou no quadro inteiro de forma que minimize a SAD ou qualquer outro critério escolhido. No caso mostrado na Fig. 2.8, a procura é feita dentro de uma janela de busca (pois considera-se que os quadros capturados em um vídeo possui uma latência tão pequena que os deslocamentos dos objetos em cena são restritos às suas vizinhanças). O deslocamento da procura no quadro anterior pode ser definido de utilizando diversos algoritmos de buscas como: espiral, circular, hexagonal, telescópica, diamante, completa, etc [21, 22, 23, 24, 25].

2.7.2 Compensação de Movimento

Aplicação dos vetores de movimento oriundos da estimação, no(s) quadro(s) de referência de modo a obter um quadro ou região estimados (vide Figura 2.7(c)). A compensação de movimento é utilizada no codificador durante a reconstrução do quadro de referência, de forma a sincronizar as informações com o decodificador. Ou seja, como o sistema de compressão em questão gera perdas, se faz necessário que o codificador tenha as mesmas referências do decodificador para que o erro entre o quadro original e o com perdas não se propague. A compensação de movimento é um dos elementos chave no processo de decodificação de um vídeo e é responsável pela predição dos quadors temporais, já que os codecs geralmente operam baseados no DPCM (*Digital Pulse*

Code Modulation) onde se faz a predição do sinal a ser codificado e se codifica apenas sua diferença.

2.8 MPEG-1 E MPEG-2

Atualmente, o compressor de vídeo mais popular, utilizado por muitas TVs digital, nos DVDs e em *streaming* de vídeo na internet, se baseia no JPEG (o que pode ser facilmente observado no diagrama de blocos da Fig. 2.9) e foi desenvolvido por um grupo denominado *Motion Picture Experts Group* popularmente conhecido como MPEG.

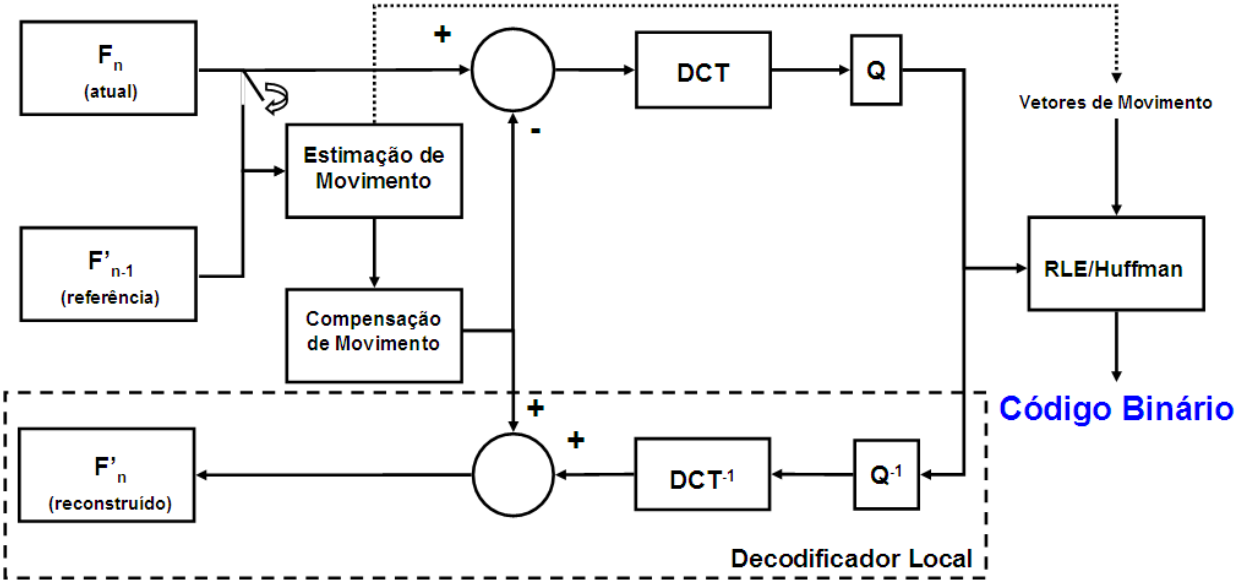

Figura 2.9: Diagrama de blocos simplificado do MPEG-1 e MPEG-2.

No caso do MPEG-1, o desenvolvimento foi otimizado para aplicações que utilizem aproximadamente 1,5 Mbps com um vídeo com resolução SIF (320×240 *pixels*) a trinta quadros por segundo. Já o MPEG-2 foi proposto para atender codificações de vídeo de alta qualidade com taxas entre 4 Mbps a 15 Mbps para compressão de vídeos com resolução SD (720×576 *pixels*). O MPEG (MPEG-1 [12] e MPEG-2) comprime as imagens se baseando em uma seqüência periódica de quadros de vários tipos denominado na literatura como GOP (*Group of Pictures*). Nos codecs MPEG-1 e MPEG-2 utiliza-se o GOP do tipo IBBP isto significa que a primeira imagem é comprimida como um quadro "intra", ou seja, baseados apenas nas informações contidas no mesmo, em seguida é comprimido um novo quadro, baseando-se no

quadro anteriormente comprimido, o que caracteriza um quadro do tipo P (Predito). O quadro P seria a predição do quadro seguinte ao estimar o movimento no quadro anterior (na ordem de exibição do vídeo) e em seguida comprimir a diferença entre o quadro estimado e o quadro original. Os quadros do tipo B (Biprediteos) são estimados e compensados a partir dos quadros do tipo I e do tipo P, previamente codificados e reconstruídos (localmente decodificado, de forma que o codificador tenha a mesma informação que o codificador para gerar as predições). Observe a ilustração das Figuras 2.10 e 2.11. Note que a ordem de compressão dos quadros é diferente da sequência de exibição do mesmo. Isto permite a compensação de quadros baseados em quadros "futuros" tendo como parâmetro a seqüência de exibição (Figura 2.10).

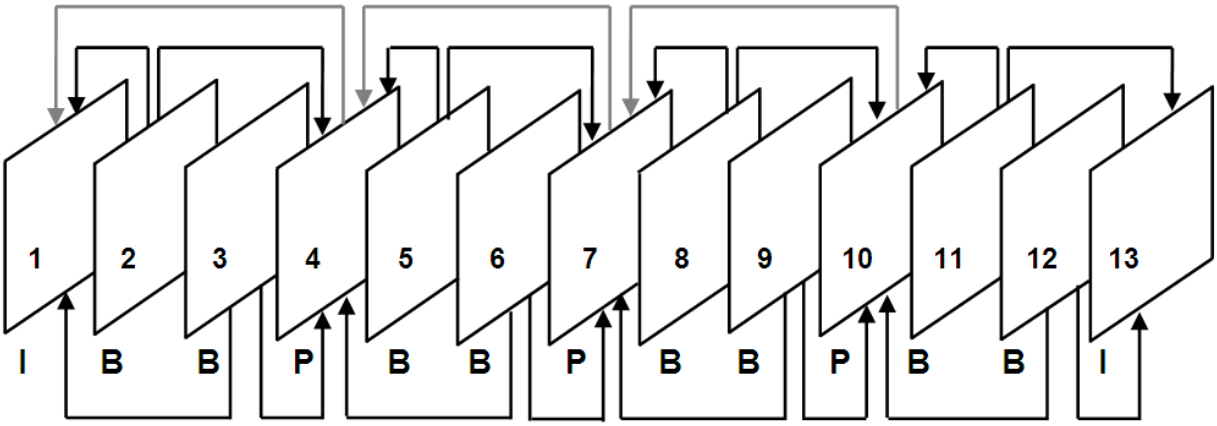

Figura 2.10: GOP típico de uma compressão MPEG-2, ordenado de acordo com a seqüência de exibição.

No MPEG-1 os quadros são constituídos de *slices*, que seria conjuntos de macroblocos contíguos em ordem lexicográfica (*raster scan*), exemplificada na Figura 2.12. Os *slices* são importantes para aumentar a robustez do sistema contra erros no canal. Por exemplo, se um *bitstream* contiver um erro em um bit, o erro causaria uma propagação devido à codificação de tamanho variável. No entanto, no próximo *slice* uma ressincronização do decodificador ocorre. Fazendo com que apenas o *slice* com erro de transmissão seja descartado ou retransmitido.

Um macrobloco no MPEG consiste em um bloco contendo 16×16 amostras de luminância e dois blocos de 8×8 da correspondente crominância, mostrado na Figura 2.1(c). Cada macrobloco codificado contém as informações da predição para compensação de movimento. Existem quatro tipos de macroblocos: "intra", "preditos por quadro anterior", "preditos por quadro posterior" e "preditos por uma média entre informações dos quadros anterior e posterior". A informação de

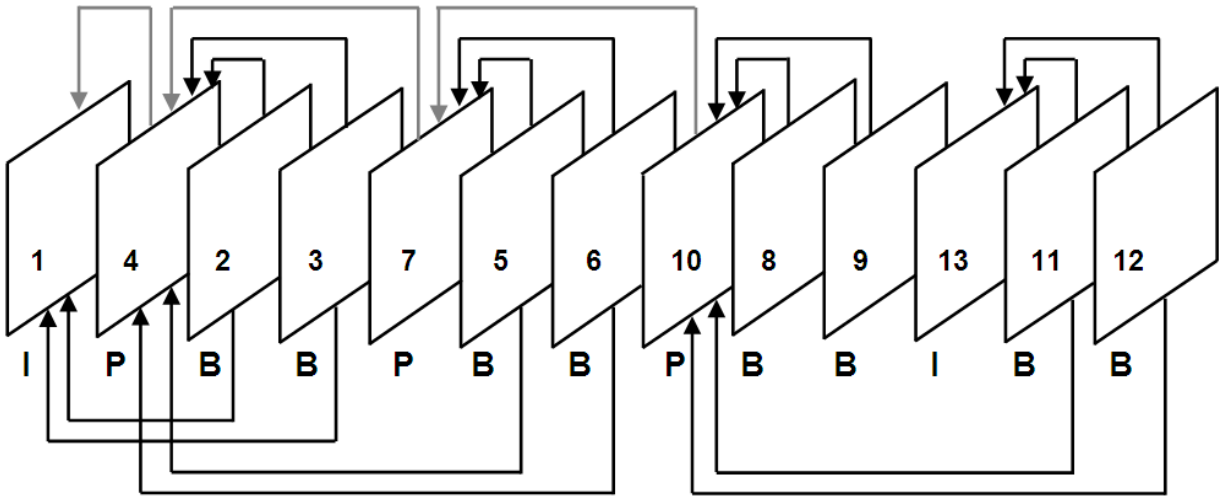

Figura 2.11: GOP típico de uma compressão MPEG-2, ordenado de acordo com a seqüência de compressão.

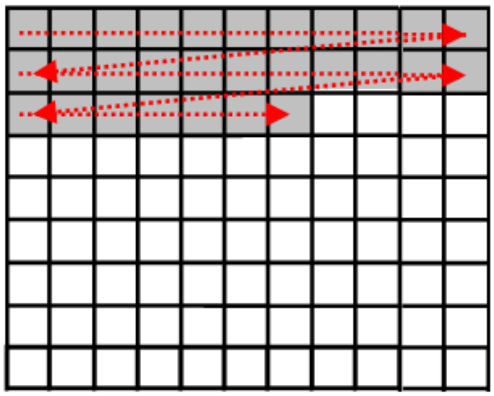

Figura 2.12: Exemplo de um *slice* em um quadro comprimido com MPEG.

movimento consiste em um vetor de movimento - com relação ao quadro anterior ou posterior - e dois vetores de movimento para o caso da informação ser predita bidirecionalmente. Quadros do tipo P podem ter macroblocos do tipo “intra” ou preditos por bloco anterior. Já os quadros B são mais genéricos e podem utilizar todos os quatro tipos de macrobloco. Existe um outro tipo de macrobloco onde a informação de movimento é inferida, juntamente com seu resíduo, denomindado *skip*, que é designado quando o vetor de movimento é nulo, bem como todos seus coeficientes da DCT. Entretanto, existe uma restrição de que o primeiro e o último macroblocos não podem ser *skip*. Para o decodificador, o *skip* seria apenas uma cópia do macrobloco de referência, o que diminui sua complexidade computacional.

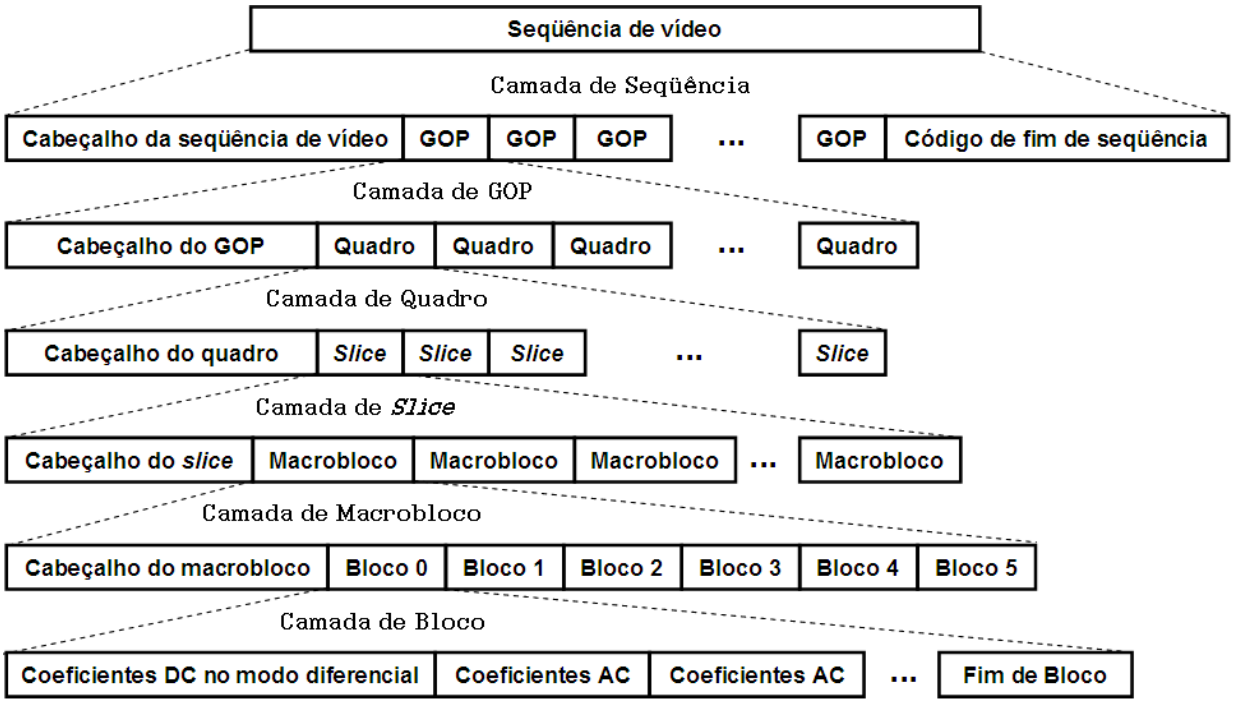

Figura 2.13: Camadas de um *bitstream* MPEG-1.

Como é mostrado na Figura 2.13, existem seis camadas de codificação (*bitstream*) no MPEG-1 e no MPEG-2: seqüência de vídeo, GOP, quadro, *slice*, macrobloco, e bloco.

O *bitstream* de uma seqüência de vídeo é basicamente constituída de um cabeçalho, um ou mais GOPs, e um código de fim de seqüência. O conteúdo dela contém uma série de parâmetros como tamanho da figura (dimensão horizontal e vertical em pixels), taxa de quadros por segundo, taxa de bits, tamanho mínimo de *buffer*, etc. A camada de bits referentes ao GOP consiste em um conjunto de quadros dispostos na ordem de exibição, e contém uma série de parâmetros: como o

código de tempo, que fornece as horas, minutos e segundos do intervalo de tempo desde o início da seqüência, e *flags* que indicam qual quadro de referência será utilizado pelo decodificador.

A camada de bits que representam os quadros atua como uma unidade primária de codificação, que contém uma série de parâmetros: a referência temporal, que identifica o número do quadro, ou seja, a seqüência para determinar a ordem de exibição; tipo de quadro (I/P/B/D) e a ocupação inicial do *buffer* de decodificação, evitando *overflow* e *underflow*; e ainda os vetores de movimento dos quadros P e B.

Já a camada de bits que representam o *slice* atua como uma unidade ressincronizadora, que contém a posição inicial do *slice* e o fator de quantização pelo qual o referido *slice* foi codificado.

A os bits referentes à descrição dos macroblocos age como uma unidade para compensação de movimentos. E contém os seguintes parâmetros: um enxerto de bits opcional, incremento do endereçamento do macrobloco, tipo de macrobloco, fator de quantização, vetores de movimento, e a forma de se codificar os seis blocos do macrobloco (são seis blocos de 8×8, onde quatro são referentes à luminância e dois referentes à crominância).

A camada de bits que se refere aos blocos seria a camada de nível mais baixo da seqüência de vídeo e consiste na codificação dos coeficientes da DCT de 8×8. Quando um macrobloco é codificado no modo "intra", os coeficientes DC de uma imagem são codificados de maneira similar ao JPEG, onde o coeficiente DC do macrobloco atual é predito a partir do coeficiente DC do macrobloco anterior. No início de cada *slice* é atribuído o valor de 1024 para a predição dos coeficientes DC para os todos os blocos de luminância e crominância. Os valores diferenciais de DC são codificados utilizando o VLC para representar a informação residual. Finalmente, os coeficientes AC são codificados utilizando VLC para representar os valores representados por *zero run length* (onde o valor nulo é representado apenas pela sua quantidade) e os coeficientes não-nulos.

O desenvolvimento do MPEG-2 foi fortemente baseado no MPEG-1, tanto que existem alguns modos em que são compatíveis entre si. Ou seja, o MPEG-2 consegue decodificar um *bitstream* de um vídeo comprimido com o MPEG-1, o que induz a afirmar que o MPEG-2 é uma adição de propriedades e características no MPEG-1, tal como: suporte tanto a vídeo progressivo (Fig. 2.14(a)) quanto entrelaçado (Fig. 2.14(b)), sendo que os quadros entrelaçados

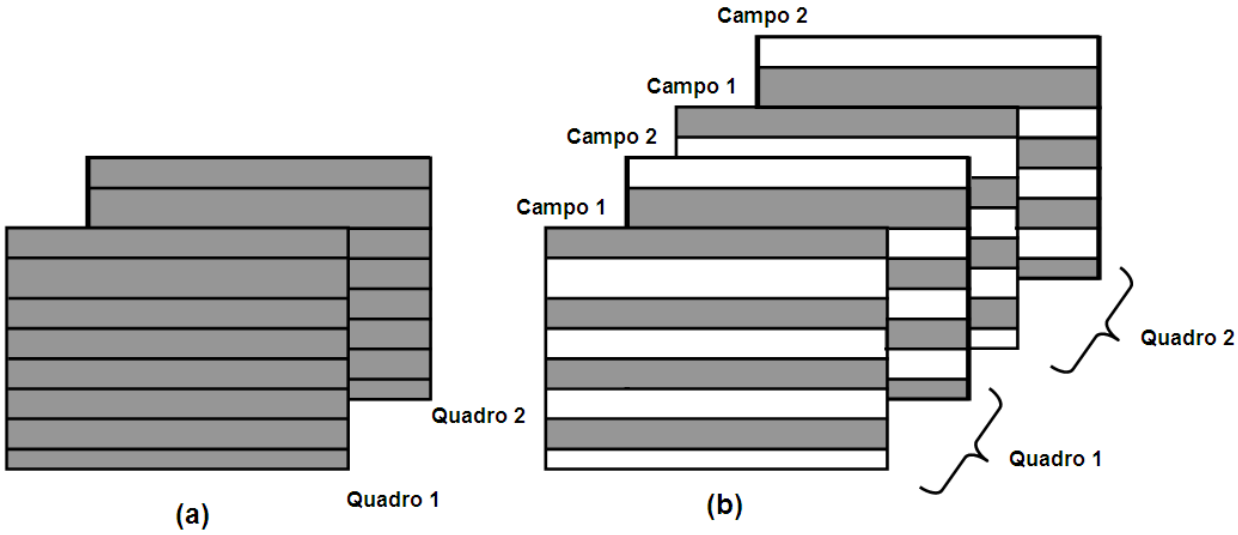

Figura 2.14: (a) Varredura de quadro progressiva; (b) Varredura de quadro entrelaçada.

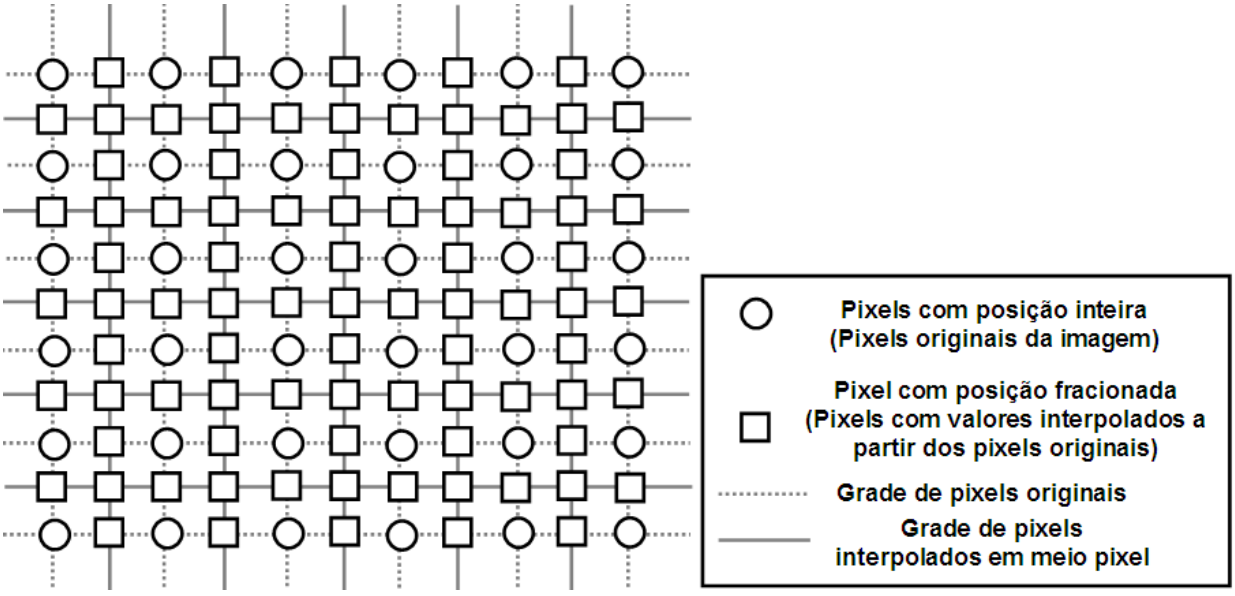

Figura 2.15: Ilustração de um bloco com precisão de meio pixel para estimação de movimento.

são muito utilizados para evitar cintilação (*flicker*) (onde a seqüência de vídeo “pisca” devido à falta de atualização da imagem); vetores de movimento com meio pixel de precisão (Fig. 2.15); escalabilidade de qualidade (SNR - *signal noise ratio*); escalabilidade espacial; escalabilidade temporal; permite ainda o particionamento de dados, que garante maior robustez contra erros e perdas de pacotes; e ocultamento de erros, que é um mecanismo de minimização de erros de transmissão no decodificador.

2.9 H.264: UMA VISÃO GERAL DESTE PADRÃO DE CODIFICAÇÃO DE VÍDEO

O H.264/MPEG-4 AVC (*Advanced Video Coder*) [4] é um padrão de compressão de vídeo relativamente novo, desenvolvido em esforço conjunto pelo MPEG (*Motion Picture Experts Group*), que é um grupo de estudos pertencente à ISO (*International Standards Organization*) e o VCEG (*Video Coding Experts Group*), que é um grupo de estudos pertencente à ITU (*International Telecommunication Union*). Este tem sido o padrão de compressão de vídeo mais aceito no mercado e na academia desde a adoção do MPEG-2.

Este padrão permite taxas de compressão bem maiores do que já se conseguiu com os padrões anteriores [5], permitindo a compressão de vídeos com ou sem entrelaçamento de forma bastante eficiente e mesmo usando altas taxas de compressão oferecendo ainda uma qualidade visual melhor do que os padrões anteriores. Além disso, esse padrão viabiliza que a codificação seja feita de forma mais flexível, bem como que se organize os dados codificados de forma que possam potencialmente aumentar a taxa de erros e minimizar as perdas da dados.

Era previsível, no entanto, que um aumento na eficiência e na flexibilidade de codificação implique em um aumento na complexidade, quando comparado aos padrões anteriores. O padrão consiste basicamente em uma gama de ferramentas projetadas para permitir que se codifique de forma eficiente uma grande variedade de material de vídeo.

O modelo, cujas etapas de compressão de vídeo que constituem numa interface de estimação e compensação de movimentos, uma fase de transformação e codificação de entropia é conhecido na literatura como sendo um modelo híbrido de codec de vídeo DPCM/DCT. Esse é, basicamente,

o modelo usado pela maior parte dos padrões de codificação de vídeo atualmente, apesar de haver várias diferenças em detalhes e implementação entre eles. Um detalhe interessante que deve ser mencionado é que em todos os padrões de vídeo não se define um codec (codificador e decodificador) específico, mas apenas sintaxe de uma seqüência de bits de vídeo codificado juntamente com um método de decodificação dessa seqüência de bits.

A Figura 2.16 e a Figura 2.17 mostram como seria o diagrama de blocos básico, tanto do codificador quanto do decodificador do H.264, respectivamente, e quais seriam os elementos funcionais a serem incluídos em ambos, a fim de que eles possam ser compatíveis e estarem de acordo com o que o padrão estabelece. Com exceção do filtro de *deblocking* (ou filtro de redução de efeito de blocos), a maior parte dos elementos funcionais básicos incluídos no H.264 (tais como predição, transformada, quantização e codificação) também estão presentes em padrões anteriores. As modificações mais importantes introduzidas no H.264 podem ser identificadas nos blocos funcionais.

No codificador, o quadro de vídeo é processado para ser reduzido a uma seqüência de bits codificada e no decodificador essa seqüência de bits comprimida é decodificada para que seja produzida uma versão reconstruída do quadro de vídeo original, que, em geral, não é idêntica àquele.

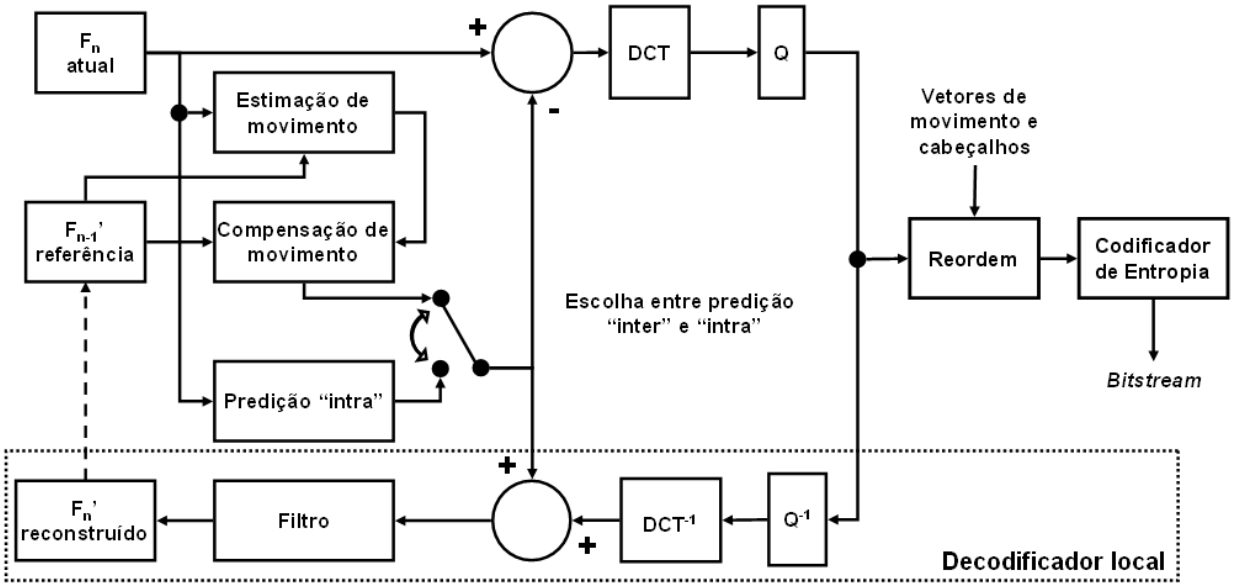

Figura 2.16: Blocos funcionais do codificador de vídeo do H.264.

O fluxo de dados no interior do codificador ocorre em dois sentidos: no da codificação e no

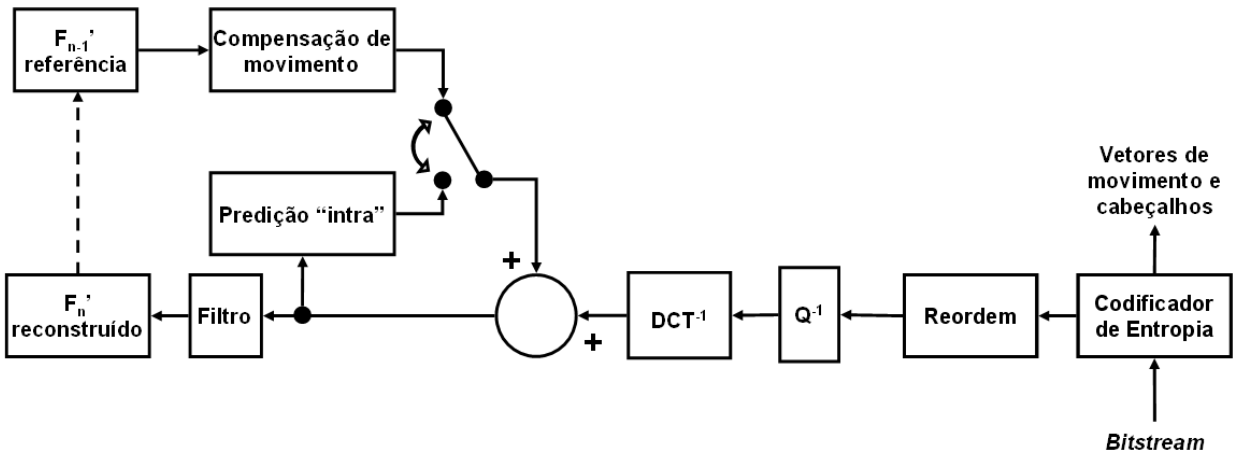

Figura 2.17: Blocos funcionais do decodificador de vídeo do H.264.

da reconstrução. No sentido da codificação, o quadro ou o *slice* de vídeo de entrada é dividido e processado em unidades de macroblocos. Cada um dos macroblocos é codificado em modo "intra" ou "inter" e, para cada macrobloco, uma predição é feita com base em amostras de quadros ou macroblocos previamente reconstruídos.

No modo "intra", a predição é feita com base em amostras do *slice* atual que tenham sido previamente codificadas e reconstruídas. Já no modo "inter", a predição formada é baseada em uma ou duas figuras de referência. Nas Figuras 2.16 e 2.17, essa figura (ou quadro, ou bloco) de referência é representada por F'_{n-1}, que denota a figura codificada imediatamente antes da atual. Foi adotada essa estratégia como uma medida de simplificação que viabilizasse a descrição do processo de forma ilustrativa; porém deve-se lembrar que podem ser escolhidas tanto figuras anteriores quanto posteriores (considerando-se a ordem de exibição) desde que tanto uma quanto a outra tenham sido previamente codificadas, decodificadas e reconstruídas.

O decodificador local dentro do codificador é necessário, pois tanto codificador quanto decodificador deve utilizar os mesmos quadros de referência para que seja feita a predição evitando assim um erro conhecido pela literatura como *drifting*, onde haveria uma propagação do erro entre o quadro original e o decodificado - na prática, este efeito geralmente deixa um rastro em torno dos objetos em movimento.

Ainda em se tratando do modo "inter", a função de estimação de movimento é responsável por encontrar uma região do mesmo tamanho do macrobloco que está sendo processado e que atenda aos critérios de correspondência entre uma e outra. O deslocamento entre a posição do

macrobloco atual e a região de referência é o vetor de movimento (MV). Depois, com base no vetor de movimento escolhido, uma predição baseada em compensação de movimentos é gerada.

Independentemente do tipo de predição gerada, a mesma é subtraída do macrobloco atual, gerando o macrobloco residual ou diferencial. Esse mesmo macrobloco residual é dividido em sub-blocos menores e todos são submetidos à transformada (que deve ser baseada em blocos) separadamente. Neste ponto, é importante mencionar que no H.264 é usada uma versão modificada da DCT.

Posteriormente à transformada, os blocos são quantizados e a saída do quantizador é um conjunto de coeficientes que são, em seguida, reordenados e, por fim, submetidos à codificação de entropia. Os coeficientes codificados, juntamente com informações suplementares necessárias à decodificação de cada bloco, formam a seqüência de bits (*bitstream*) de vídeo comprimido, que pode ser armazenado ou então disponibilizada à Camada de Abstração de Rede (NAL), responsável por prover meios de transportar dados de vídeo através de uma variedade de infra-estrutura de redes [26].

No sentido da reconstrução, os coeficientes de cada macrobloco quantizado são "reescalonados"(operação inversa à quantização) e inversamente transformados para produzir um bloco residual decodificado. É importante salientar que um certo nível de distorção é introduzido, visto que a quantização é um processo não-reversível. Então o bloco residual decodificado não é idêntico ao bloco residual produzido no sentido da codificação.

Finalmente, ao bloco residual é acrescentada a predição para produzir o macrobloco reconstruído que nada mais é do que a versão decodificada do bloco original. Em seguida, a versão reconstruída é submetida à filtragem, a fim de que se reduzam os efeitos de blocos. Ao final, todos os blocos reconstruídos são salvos para que juntos formem a versão reconstruída do quadro original.

Depois de realizada a codificação completa do quadro, o quadro reconstruído pode ser usado como quadro de referência para o próximo quadro a ser codificado.

Já no decodificador, a seqüência de bits comprimida é submetida à decodificação de entropia, ao reordenamento. Os coeficientes de cada bloco quantizado são “reescalonados”, inversamente transformados e, juntamente com as informações suplementares, formam um bloco residual

decodificado (que, por sinal, é idêntico àquele produzido no caminho da reconstrução, no codificador).

Então, o vetor de movimento decodificado é usado para localizar, no decodificador, uma região de referência e, uma vez localizada, para que seja feita a predição, que deve ser adicionada ao bloco residual a fim de que se produza a versão decodificada do mesmo. Em seguida, essa versão decodificada é filtrada. Por último, todos os macroblocos são salvos para que se produza a versão reconstruída do quadro.

2.9.1 Divisão de um quadro em macroblocos e *slices*

Todos os quadros são particionados em macroblocos de tamanho fixo de 16×16 amostras de componente de luminância. No caso do vídeo digitalizado no formato 4:2:0, amostras de 8×8 são utilizadas em cada componente de crominância. Todas as amostras (luminância e crominância) de um macrobloco são espacialmente ou temporalmente preditos, e o resíduo resultante (caso exista) é representado utilizando uma codificação por transformada. Os macroblocos são organizados em *slices*, que representam regiões de um dado quadro que podem ser decodificados entropicamente de maneira independentemente entre si. O H.264 suporta cinco tipo de *slices*. No mais simples: o *slice I* (onde *I* significa "intra"), todos os macroblocos contidos nele são codificados sem se referir a nenhum outro quadro da seqüência de vídeo. Quadros já comprimidos anteriormente podem ser utilizados para prever os macroblocos de *slices* do tipo *P* (preditivo) e *B* (bi-preditivo). Os outros dois tipos de *slices* são SP(*switching P*) e SI(*switching I*), que foram especificados para chavear eficientemente entre códigos com *bit streams* comprimidos em várias taxas. [13]

2.9.2 Predição Espacial Intra

Conforme mencionado anteriormente, cada macrobloco pode ser transmitido como sendo uma das várias possibilidades de codificação dependendo do tipo de *slice*. Em todos os tipos de *slice*, pelo menos dois tipos de codificação de macroblocos "intra" são suportados, cuja predição é implementada no domínio espacial, sendo ainda distinguíveis apenas pelas suas dimensões das amostras de luminância: 4×4, 8×8 (somente no perfil FREXt - *Fidelity Range Extensions*) e 16×16. Já os pixels de crominância são preditos de maneira análoga, mas com tamanhos

compatíveis ao de seu macrobloco. Neste tipo de predição os pixels vizinhos de blocos já codificados (eventualmente transmitidos) e decodificados são utilizados como referência para predição.

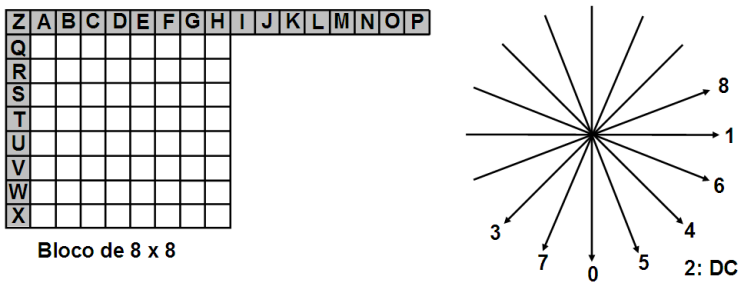

Figura 2.18: Predição “intra” para um bloco de 8×8.

Na Figura 2.18 exemplifica-se a predição “intra” de um bloco de 8×8 pixels, que são calculadas com base nas amostras A-P e Q-X, de acordo com as direções 0, 1, 3, 4, 5, 6, 7, 8. No caso do modo 2 (DC) todas as amostras são preditas com base em A-H e Q-X, como detalhado a seguir:

- *Modo 0*: Vertical As amostras A-H são extrapoladas verticalmente.
- *Modo 1*: Horizontal As amostras Q-X são extrapoladas horizontalmente.
- *Modo 2*: DC Todas as amostras são preditas a partir da média das amostras A-H e Q-X.
- *Modo 3*: Diagonal abaixo à esquerda As amostras são interpoladas em um ângulo de 45° a partir do canto superior direito (P).
- *Modo 4*: Diagonal abaixo à direita As amostras são interpoladas em um ângulo de 45° a partir do canto superior esquerdo (Z).
- *Modo 5*: Vertical direita) As amostras são interpoladas em um ângulo de 63,4° a partir do canto superior esquerdo (Z).
- *Modo 6*: Horizontal abaixo As amostras são interpoladas em um ângulo de 26,6° a partir do canto superior esquerdo (Z).
- *Modo 7*: Vertical esquerda As amostras são interpoladas em um ângulo de 63,4° a partir do canto superior direito (P).

- *Modo 8*: Horizontal acima As amostras são interpoladas em um ângulo de 26,6° a partir do canto inferior esquerdo (Z).

2.9.3 Compensação de Movimento Predição em *slices P*

Adicionados aos tipos de codificação "intra" dos macroblocos, várias predições ou compensações de movimento são permitidas em *slices P*. Cada macrobloco do tipo *P* pode ser particionado para descrever melhor o movimento em cena. Os macroblocos de tamanho 16×16 podem ser particionados em duas regiões nas seguintes formas: 16×8 ou 8×16; ou em quatro sub-macroblocos de 8×8 pixels, que por conseguinte, cada sub-macrobloco pode ser particionado em regiões de 8×4, 4×8 ou 4×4 pixels. De acordo com a ilustração na Figura 2.19.

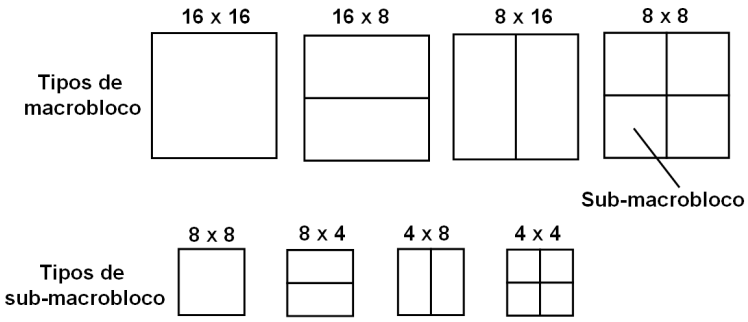

Figura 2.19: Tipos de macroblocos e sub-macroblocos.

O sinal predito de cada bloco de luminância é obtido por um deslocamento especificado por um vetor de movimento translacional e um índice que informa o quadro de referência. A precisão do vetor de movimento chega à granularidade de um quarto da distância entre pixels vizinhos. Se o vetor de movimento aponta para uma posição inteira, a predição do sinal corresponde às amostras do quadro de referência; caso contrário, a predição do sinal é obtida utilizando interpolação entre as posições inteiras. Os valores da predição em meio pixel são obtidos aplicando um filtro FIR unidimensional de seis *taps*, e valores de predição com posições com valores referentes a um quarto de pixel são gerados pela média das amostras entre as posições inteiras e de meio pixel (observe a Figura 2.20). Já a predição dos valores para as componentes de crominância são obtidos por interpolação bilinear.

As componentes dos vetores de movimento são codificadas diferencialmente utilizando a mediana da direção dos blocos vizinhos. A operação mediana ($Median(\cdot)$), neste caso, é definida

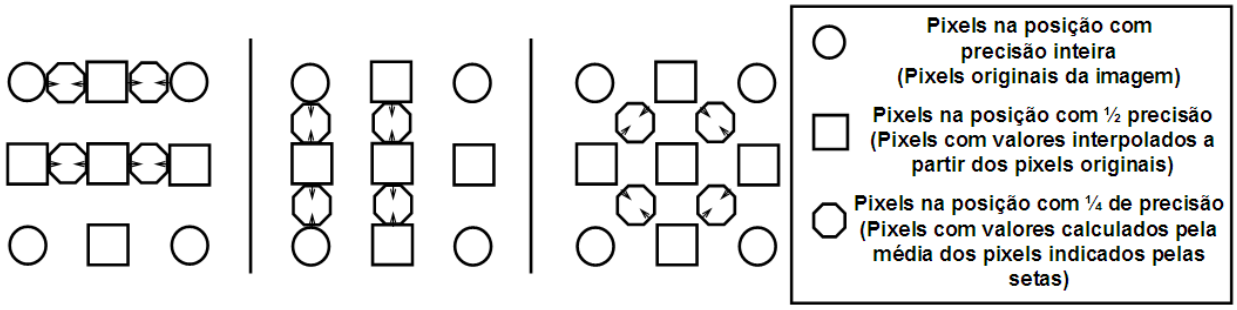

Figura 2.20: Ilustração das precisões (em pixel) de um quarto, meio e inteiro.

pela equação 2.13

$$Median(a,b,c) = a + b + c - Min(a, Min(b,c)) - Max(a, Max(b,c)), \tag{2.13}$$

e as operações $Min(\cdot)$ (equação 2.14) e $Max(\cdot)$ (equação 2.15) são definidos como

$$Min(a,b) = \begin{cases} a, & \text{se } a \leq b \\ b, & \text{se } a > b \end{cases} \tag{2.14}$$

$$Max(a,b) = \begin{cases} a, & \text{se } a \geq b \\ b, & \text{se } a < b \end{cases} \tag{2.15}$$

onde $a, b, c \in \mathbb{Z}$, apesar dos vetores de movimento serem representados vetorialmente, a mediana geométrica não é utilizada pelo padrão do H.264, mas sim a mediana dos coeficientes dos vetores de movimento.

Nenhuma predição das componentes dos vetores de movimento (ou outra forma de predição) ocorre entre a fronteira dos *slices*. O H.264/MPEG4-AVC suporta a predição da compensação de movimento entre múltiplos quadros previamente codificados. Este conceito é ilustrado na Figura 2.21.

Existe uma outra possibilidade associada aos *slices P* que podem ser codificados como *skip mode*. Para este modo, nenhum sinal de erro de predição quantizado (resíduo), tampouco os vetores de movimento ou parâmetros de referência são transmitidos. A reconstrução do sinal é computada de maneira similar à predição de um macrobloco de tamanho 16×16 e quadro de

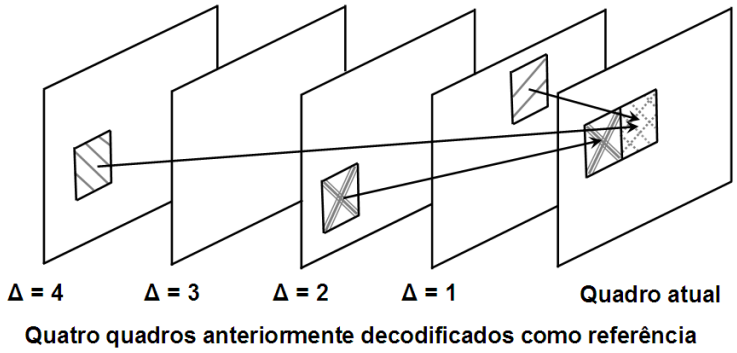

Figura 2.21: Compensação de movimento utilizando múltiplos quadros de referência.

referência com índice fixado em zero. Diferente dos padrões de vídeo antecessores, os vetores de movimento utilizados para reconstrução de um macrobloco do tipo *skip* são inferidos de acordo com o movimento dos macroblocos vizinhos previamente decodificados ao invés de assumí-las como zero (ou seja, sem movimento).

2.9.4 Compensação de Movimento Predição em *slices B*

Comparado com os padrões de vídeo anteriores, o conceito de *slices B* no H.264/MPEG4-AVC é generalizado de diversas maneiras [2]. Por exemplo, no MPEG2 os quadros do tipo *B* não podem ser utilizados como referência para predição de quadros subseqüentes. De fato, a única diferença substancial entre *slices P* e *B* no H.264 é que na última o bloco predito pode ser codificado como sendo uma média ponderada de duas predições (compensações de movimento) com valores distintos, o que não implica necessariamente que exista uma restrição para que se tenha predição com relação aos quadros posteriores ou anteriores à exibição do quadro de referência. Ou melhor, o conceito de *slices B* generalizados do H.264 permite quaisquer pares de quadros de referência para serem utilizados na predição de cada região. Para tanto foram utilizadas duas listas de forma a indexar múltiplos quadros de um *buffer* denominados *list 0* e *list 1*. Entretanto, dependendo de qual quadro de lista de referências for utilizado para predição do sinal, três tipos diferentes de predição entre quadros são distinguíveis nos *slices B*: *list 0*, *list 1*, e *bi-predictive*, onde o último utiliza uma superposição das predições referentes à *list 0* e *list 1*. De maneira similar às partições nos *slices P*, os três tipos de predição temporal existentes nos *slices B* podem ter seus macroblocos ou sub-macroblocos particionados (Figura 2.19). Em adição, os macroblocos ou sub-macroblocos de um *slice B* podem ser codificados em um “modo

direto", onde não existe a necessidade de se adicionar no *bitstream* nenhuma informação sobre o movimento. E se nenhum resíduo de predição for transmitido para este tipo de macrobloco, o mesmo é referido como sendo *skipped*, que é um modo muito eficiente tanto nos *slices B* quanto no *slice P*.

2.9.5 Transformada, Escalonamento e Quantização

Como mencionado anteriormente, o H.264/MPEG4-AVC também utiliza uma codificação por transformada no ruído de predição. Contudo, em contraste com padrões anteriores, como o MPEG2 ou H.263, que utilizam a transformada bidimensional de cossenos discreta (DCT-2D) de tamanho 8×8, o H.264 faz uso de um conjunto de transformadas inteiras de blocos de tamanhos diferentes. De modo geral, a transformada inteira de 4×4 é aplicada tanto para as componentes de luminância quanto para as de crominância do resíduo da compensação. Em adição outra transformada de $M \times N$ é aplicada para todos os coeficientes DC resultante de um macrobloco (16×16) que é codificado utilizando codificação "intra" com $M = 4$ e $N = 4$ assim como no caso das componentes de crominância que utilizam valores de $M, N \in \{2, 4\}$ dependendo do formato da luminância. Para esses estágios adicionais de transformadas, combinações separáveis da transformada Hadamard de quatro *taps* e transformadas Haar/Hadamard de dois *taps* são aplicadas. Apesar da importante propriedade de um sistema de baixa complexidade computacional, o uso de uma transformada de tamanho reduzido no H.264 tem ainda a vantagem de reduzir artefatos de *ringing* oriundos do fenômeno de Gibbs [27]. Todavia, para vídeo de alta fidelidade, a preservação da suavidade e da textura é geralmente beneficiada com representações com funções de bases maiores. Um bom custo benefício para esta situação ocorre com o uso da transformada de tamanho 8×8. Uma transformada inteira parecida com a DCT bidimensional de tamanho 8×8 foi incorporada ao FREXt, possibilitando implementações eficientes em sistemas com aritmética inteira. De fato, qualquer transformada inteira do H.264, assim como suas respectivas transformadas inversas, podem ser implementadas de maneira simples e eficiente, já que apenas as operações de deslocamento e adição em um processamento com $(8 + b)$ bits são necessários para comprimir e descomprimir um vídeo com b bits de profundidade.

Isso implica em mais um grau de liberdade no perfil de alta fidelidade, onde o codificador tem

a possibilidade de escolher o uso da transformada de 4×4 ou de 8×8 para que a representação dos resíduos consiga se adaptar às características intrínsecas de cada macrobloco. Esta adaptação é conjugada com outras partes relacionadas com o processo de decodificação; por exemplo, não permitir a utilização de transformadas de 8×8 caso o bloco de predição seja menor que 8×8.

Para quantizar os coeficientes transformados do H.264 utiliza-se um dos 52 possíveis valores de escalonamento dos quantizadores de reconstrução uniforme (URQs - *Uniform-Reconstruction Quantizers*), denominados de parâmetros de quantização ou simplesmente QP. A escala de operação é organizada de forma que o passo de quantização dobra a cada incremento de seis no valor de QP. Os coeficientes transformados de um bloco geralmente são percorridos via *zig-zag* e depois processados por um codificador de entropia, que será descrito a seguir.

2.9.6 Codificação de Entropia

No H.264 vários elementos sintáticos são codificados utilizando a mesma estrutura de código de tamanho variável (VLC - *Variable Length Code*) denominado código exponencial-Golomb de ordem zero. Alguns elementos sintáticos são codificados usando representação de códigos em tamanho fixo. Para os demais elementos sintáticos, duas possibilidades de codificação de entropia podem ser utilizadas. Quando utiliza-se a primeira configuração de codificação de entropia, que requer implementações de baixa complexidade computacional, o código exponencial-Golomb [2] é usado em quase todos os elementos sintáticos exceto para os coeficientes transformados e quantizados, que utiliza um método um pouco mais sofisticado denominado *CAVLC - context-adaptive variable length coding*. Quando o CAVLC é utilizado, o codificador chaveia entre diferentes tabelas de códigos de tamanho variável dependendo dos valores previamente transmitidos, adicionando assim uma característica de contexto adaptativo, uma vez que as tabelas de VLC foram desenvolvidas de forma que o contexto condicione a ocorrência de um símbolo. O desempenho do codificador de entropia aumenta sensivelmente ao utilizar a segunda configuração, referida na literatura como CABAC - *Context-Based Adaptive Binary Arithmetic Coding* [28, 29]. Como diagramado na Figura 2.22, o CABAC é baseado em três componentes: o binarizador, o modelador de contexo e o codificador aritmético binário.

A binarização permite uma codificação aritmética binária eficiente mapeando elementos

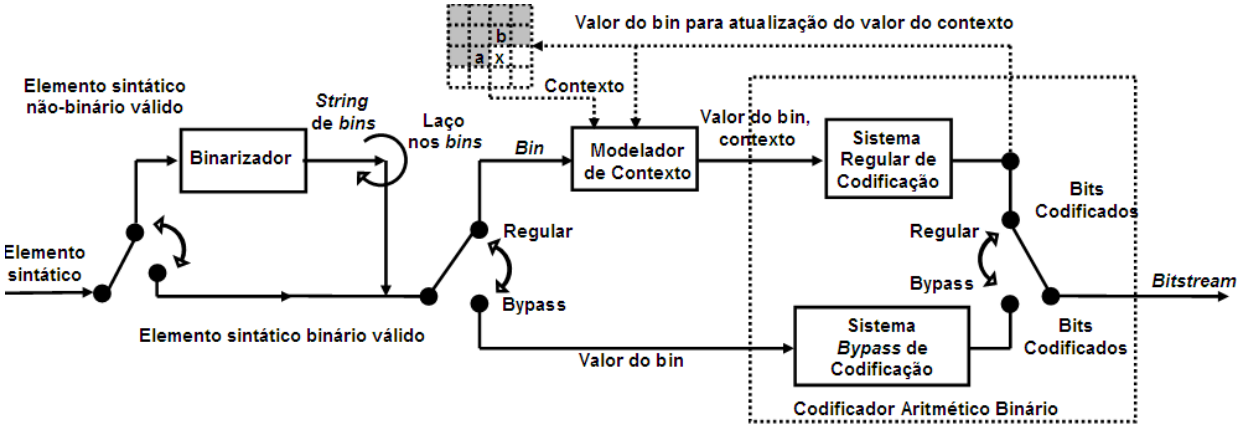

Figura 2.22: Diagrama de Blocos do CABAC.

sintáticos não-binários em sequências de bits referidas como "*string* de *bins*". Os *bins* de uma *string* podem ser processados tanto no modo de codificação aritmética, ou no modo de *bypass*, onde o último é um modo de codificação simplificado que é selecionado quando os *bins* contiverem conteúdos referentes à informação de sinais ou *bins* de menor importância de modo a tornar o processo de compressão e descompressão mais rápido. O modo de codificação aritmética provê o maior benefício de compressão, onde o *bin* pode ter seu contexto modelado e subsequentemente codificado aritmeticamente. Por uma decisão no desenvolvimento, na maioria dos casos, somente o *bin* mais provável de um elemento sintático é beneficiado pelo modelador de contexto externo, que é baseado em *bins* previamente codificados e decodificados. O desempenho da compressão aritmética dos *bins* é otimizada por uma estimação adaptativa da correspondente distribuição de probabilidades, que no caso é condicionada ao contexto. A estimação da probabilidade e a atual codificação aritmética binária são implementáveis utilizando métodos sem multiplicação, viabilizando eficientemente o desenvolvimento do mesmo em software e hardware. Comparado ao CALVC, o CABAC tipicamente reduz em torno de 10 a 20% a taxa de bits para a mesma qualidade objetiva de um sinal de vídeo SDTV/HDTV codificado.

2.9.7 Filtro Redutor de Efeito de Bloco

Uma das características particulares de codificadores baseados em blocos é a ocorrência de descontinuidades visualmente perceptíveis ao longo das bordas dos blocos, uma vez que os lados dos blocos são codificados com menos acurácia que o interior dos mesmos. Por esta razão, o

H.264 define um filtro de redução de efeitos de bloco adaptativo (igual ao do H.263) aplicável ao ciclo de codificação e reconstrução, e isso se constitui como uma componente necessária para o processo de decodificação. A adaptabilidade do filtro ocorre desde o nível de *slices*, passando pelas bordas até o nível de amostras. Os parâmetros do filtro são controlados pelos valores de vários elementos sintáticos. Para maiores detalhes vide a referência [30]. Como resultado, o efeito de bloco é reduzido sem afetar muito as altas freqüências do conteúdo da imagem. Consequentemente, a qualidade subjetiva aumenta significativamente, ao mesmo tempo em que o filtro reduz tipicamente entre 5 a 10 % de taxa de bits produzindo a mesma qualidade objetiva em comparação com um vídeo não filtrado.

2.9.8 Perfis do H.264

Os perfis e níveis especificam pontos de conformidade que facilitam a interoperabilidade entre várias aplicações que tenham requisitos funcionais similares. Um perfil define um conjunto de ferramentas de codificação ou algoritmos que podem ser usados para gerar um *bitstream*, já os níveis indicam restrições em certos parâmetros chave do *bitstream*. Todos os decodificadores de um determinado perfil deve ter a capacidade de suportar todas as características deste perfil. Já os codificadores não são obrigados à utilizar nenhuma característica específica de um perfil, mas comprimir gerando um *bitstream* compatível, ou seja, que algum decodificador H.264 em conformidade com o padrão consiga decodificar.

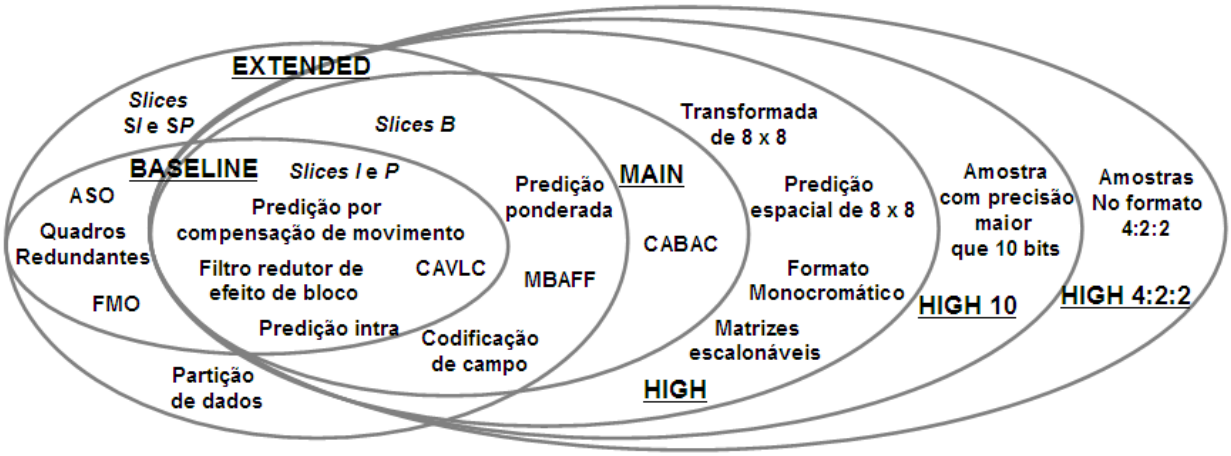

Figura 2.23: Ilustração dos perfis do H.264.

Na primeira versão do H.264 três perfis foram definidos: *Baseline*, *Extended* e *Main*. Onde

o perfil Baseline suporta todas as características do H.264/MPEG4-AVC, versão 1 (2003), exceto pelos conjuntos de características a seguir:

1. *Slices B*, codificação de campo, chaveamento adaptativo entre quadro e campo (MBAFF - *Macroblock Adaptive Switching Between Frame and Field*) e predição ponderada.
2. CABAC.
3. *Slices SI* e *SP*, e partição de dados com *slice*.

Os dois primeiros itens contêm um conjunto de características que são suportados pelo perfil *Main*, em adição às características suportadas pelo *Baseline* exceto para a FMO (*Flexible Macroblock Order*) e outras características de robustez a erros. [13]. O perfil *Extended* suporta todos as características do perfil *Baseline* adicionado aos itens um e três. A grosso modo, o perfil *Baseline* foi desenvolvido visando aplicações com o mínimo de complexidade computacional e o máximo de robustez a erro, já o perfil *Main* focava aplicações que necessitassem do máximo em eficiência de compressão. Finalmente, o perfil *Extended* foi desenvolvido para promover um compromisso entre os perfis *Baseline* e *Main* com um foco para necessidades específicas de aplicações com *streaming* de vídeo adicionado à robustez e erros e perda de pacotes.

Como mostra a Figura 2.23, existe um adendo ao padrão em sua terceira versão onde foi incluído o FRExt, que, em adição aos três perfis da primeira versão, outros três perfis foram desenvolvidos com base no perfil *Main*. Como suas interseções comuns, o perfil *High* contém as ferramentas mais relevantes do FRExt para aumentar a eficiência do codificador, e comparado ao perfil *Main* essas ferramentas adicionam apenas um aumento moderado, em alguns casos nenhum aumento, na complexidade em termos custos de implementação e esforço computacional (do decodificador). Entretanto, o perfil *High* faz uso, assim como no perfil *Main*,de uma precisão de 8 bits por amostra para seqüências no formato 4:2:0, em aplicações típicas de SD e HD. Outros dois perfis chamados *High 10* e *High 4:2:2* extendem a capacidade do padrão de incluir demandas que necessitem de amostras com maior precisão (maior que 10 bits por amostra) e maior formato de crominância (no caso, 4:2:2). Originalmente o FRExt ainda possuia a especificação do perfil *High 4:4:4*, todavia este foi removido da especificação.

3 MÉTODOS DE PARTIÇÃO DE MACROBLOCOS

3.1 INTRODUÇÃO

Os macroblocos são regiões em uma imagem (ou quadro) de tamanho $N \times N$, onde tipicamente $N = 16$, que são codificadas uma a uma, conforme descrito nas seções anteriores, são utilizados apenas em codificadores baseados em blocos como o MPEG-1, MPEG-2 e o H.264. Nestes codificadores cada macrobloco pode ser predito de três maneiras:

1. Predição com referência aos macroblocos previamente codificados do quadro atual (predição "intra" - existente no H.264).
2. Predição com referência a algum quadro anterior (predição "inter" P).
3. Predição com referência a mais de um quadro, seja anterior ou posterior (predição "inter" B).

O H.264, diferente dos padrões MPEG-1 e MPEG-2, permite a partição (divisão) de um macrobloco para a predição nos modos "intra" e "inter". No caso da predição intra, utilizada no H.264, os macroblocos de tamanho 16×16 podem ser particionados em sub-blocos de de 8×8 e 4×4 pixels, evidentemente gerando um número maior de sub-blocos.

De maneira similar, na predição entre quadros do H.264, os macroblocos também podem ser particionados em seções retangulares de 16×8, 8×16 e 8×8 pixels, o último pode particionar os sub-blocos de 8×8 em tamanhos de 8×4, 4×8 e 4×4 permitindo uma predição mais eficiente, já que quanto menor a região a ser estimada, maior a chance de se obter um bloco (ou parte dele) no quadro de referência que se assemelhe com a informação original. Pode-se assim aumentar a compressão da informação residual. Entretanto uma quantidade maior de bits é utilizada para descrever as partições deste bloco e dos vetores de movimento relativos a cada região particionada.

Ao se particionar um bloco uma função de custo deve ser utilizada como fator de critério para guiar a partição. A função de custo determinaria a relação entre custo (quantidade de bits gastos) e a qualidade resultante da compressão em um macrobloco.

3.2 PARTIÇÃO DE MACROBLOCOS PARA COMPENSAÇÃO DE MOVIMENTO

Tipicamente, a variação temporal em seqüências de vídeo ocorre devido aos movimentos da câmera ou dos objetos na cena. Ao compensarmos estes movimentos, uma predição mais eficaz do vídeo pode ser realizada utilizando uma informação relativamente compacta com respeito às regiões de movimentação. Isto possibilita aos algoritmos de compressão de vídeo reduzir significativamente a quantidade de bits necessários para transmitir a seqüência de vídeo com qualidade aceitável. Conclui-se então que a compensação de vídeo é crítica para obtermos bom desempenho em sistemas de compressão de vídeo [31].

O padrão internacional mais recente de compressão de vídeo [32], denominado H.264 ou MPEG-4 parte 10, codifica a informação de vídeo em macroblocos de 16×16 pixels. Quando a estimação de movimento é realizada na codificação entre quadros (*frames*) (*inter frame mode*), o macrobloco pode ser dividido em dois blocos de tamanho 16×8, 8×16 ou em quatro sub-macroblocos de tamanho 8×8 - que podem ser subdivididos em dois sub-blocos menores de tamanho 8×4, 4×8 ou quatro sub-blocos de 4×4 pixels. Forma-se assim uma estrutura hierárquica de decomposição denominada *quadtree* [1] [32], ilustrada pela Figura 3.1.

Nesta dissertação adicionaremos uma flexibilidade maior que a proposta pelo *quadtree*, partindo os blocos usando um segmento de reta arbitrário baseado na decomposição *wedgelet* [33]. Esta técnica mais geral será denominada compensação de movimento baseado em *wedges* (cunhas), e propõe melhorar o desempenho de compressores de vídeo híbridos como o H.264.

Em um trabalho paralelo ao realizado nesta dissertação Kondo et al. [34] propôs uma técnica

[1] também conhecido na literatura como *quad-tree like motion compensation* (compensação de movimento parecida com o *quad-tree*) ou *variable block size motion compensation* (compensação de movimento utilizando blocos de tamanho variável)

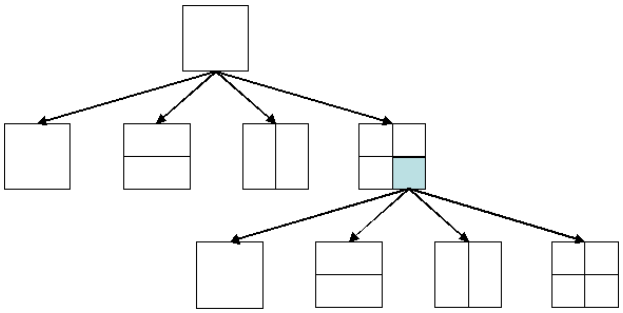

Figura 3.1: Decomposição *quadtree*.

de compensação de movimento partindo blocos utilizando um segmento de reta em um *codec* híbrido baseado no H.26L [35]. Obtendo como resultado um ganho de 5% com relação à taxa de bits na comparação com o H.26L convencional. Entretanto, os autores impuseram uma limitação para que o segmento de reta fosse descrito apenas a partir de dois pontos posicionados na extremidade do bloco.

Nesta dissertação, o segmento de reta, descrito em coordenadas polares, pode passar por qualquer posição do bloco. A vantagem de se utilizar uma notação (descrição matemática) seria a possibilidade de determinar a partição analiticamente. Além disso, propôs-se uma técnica de seleção de potenciais decomposições (partições) *wedges* baseado em: filtro de detecção de bordas, regressão linear e transformação na notação *wedgelet*, reduzindo drasticamente o esforço computacional.

Quando a partição dos macroblocos é feita, melhor e mais precisa é a compensação de movimento, mas um número maior de bits é gasto para informar ao decodificador qual partição foi escolhida. Entretanto, o resíduo resultante após a compensação é menor (em termos absolutos com relação à energia e a entropia do sinal), sendo assim comprimido utilizando um número menor de bits. Nesta dissertação estaremos discutindo a necessidade do particionamento dos macroblocos e quais tipos de partições de blocos são mais vantajosos.

Outros trabalhos mais recentes denominam as partições *wedge* por "*geometry-adaptive block partitioning*", cuja tradução literal seria partição de blocos com geometria adaptativa, que pode ser consultado em [36, 37, 38].

3.3 COMPENSAÇÃO DE MOVIMENTO BASEADO EM *WEDGES*

O *wedgelets* em multi-escala é uma ferramenta matemática [33] que explicitamente captura estruturas geométricas em imagens - que nesta dissertação é utilizada apenas sua notação em coordenadas polares. De forma a realizar a estimação e compensação de movimentos baseados nesta ferramenta, consideramos *wedges* obtidos de um bloco de tamanho $N \times N$ dividindo-o em duas regiões por meio de uma reta de distância r com relação ao centro do bloco e um ângulo θ , conforme apresenta a Figura 3.2.

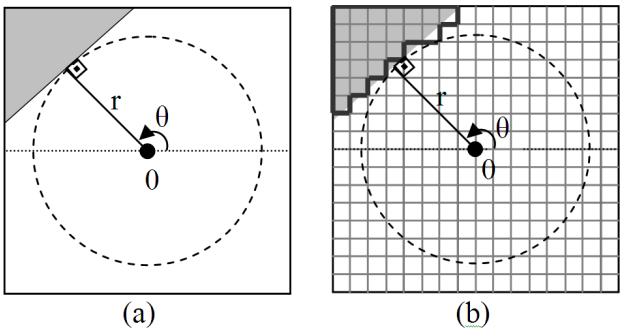

Figura 3.2: Notação em coordenadas polares de uma *wedgelet*: (a) contínuo, (b) discreto.

A estimação de movimento é calculada independentemente para cada região, gerando dois vetores de movimento. Diferentemente da partição proposta por Kondo et al. [34] (vide capítulo 3.2) a notação *wedgelet* gera um número maior de segmentos de reta possíveis na partição dos blocos para efetuar a compensação de movimento. Isto aumenta significativamente a complexidade computacional para a determinação dos vetores de movimento. Para solucionar este problema um método mais rápido será discutido posteriormente.

O conjunto de todas as partições de um bloco de tamanho $N \times N$ é gerado amostrando o subespaço determinado por r e θ em intervalos uniformes. Desta forma é gerado um dicionário de N_w elementos que depende da precisão dos incrementos de *r* e θ.

Além da partição *wedge* descrita anteriormente, um escalonamento das partições pode ser realizado de modo equivalente ao verificado na partição *quadtree*. A seguir, um pseudo algoritmo juntamente com a Figura 3.3 exemplificam com detalhe a implementação das *wedges* em multi-escala.

Seja um macrobloco de tamanho $N \times N$, onde tipicamente $N = 16$. Para cada bloco de

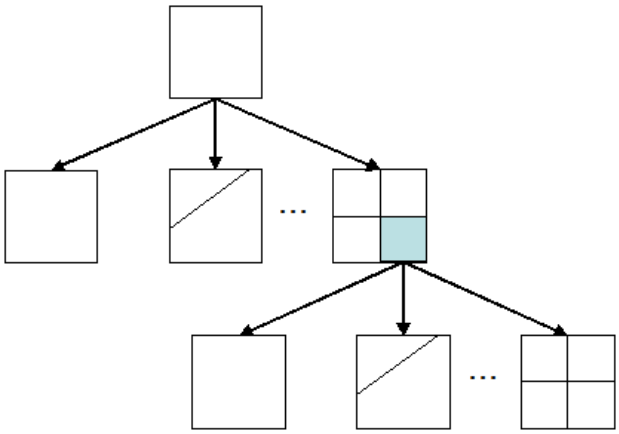

Figura 3.3: Estrutura hierárquica multi-escala das *wedges*.

$N \times N$ escolha uma das possibilidades:

- Utilize o bloco sem partição e com apenas um vetor de movimento. Termine a árvore. Caso, $N = N_{min}$, ou seja, para o bloco de menor tamanho (tipicamente 4) sempre escolha esta opção.
- Particione o bloco utilizando a técnica de *wedges*, de modo a obter duas regiões e dois vetores de movimento. Termine a árvore.
- Divida o bloco em quatro sub-blocos de tamanho $N/2 \times N/2$ e repita esta seleção de particionamento para cada sub-macrobloco.

Para cada macrobloco realizamos, idealmente, todas as possibilidades de partição e escolhemos a partição de acordo com o critério de taxa-distorção (*rate-distortion*) de acordo com a função de custo:

$$J(p) = D(p) + \lambda R(p), \tag{3.1}$$

cujo mínimo deve ser escolhido. De fato, esta função de custo pode ser calculada de diversas formas, já que este critério é utilizado apenas no codificador, tornando-o não normativo (não padronizado). Nesta equação, p se refere a todo conjunto possível de estratégia de partição de macrobloco, enquanto $J(p)$ é o resultado da função de custo, λ é um multiplicador de Lagrange [39], $R(p)$ é o número de bits necessários para a representação do macrobloco e $D(p)$ é a distorção

correspondente à estratégia de partição p, que geralmente são calculados com a SAD (Eq. 3.2) ou SSD (Eq. 3.3):

$$SAD = \sum_{i=1}^{N}\sum_{j=1}^{N} \left| pixel_{original}(i,j) - pixel_{reconstrudo}(i+x, j+y) \right|, \tag{3.2}$$

$$SSD = \sum_{i=1}^{N}\sum_{j=1}^{N} \left(pixel_{original}(i,j) - pixel_{reconstrudo}(i+x, j+y) \right)^2. \tag{3.3}$$

Para a compressão de vídeo, o compressor deve informar ao descompressor qual estratégia de partição de macrobloco foi utilizada. Seja $B(p)$ o número de bits necessários para codificar p. Ainda temos de informar os vetores de movimento associados à partição utilizada, gastando $MV(p)$ bits, mais o resíduo após a compensação de movimento, gastando $R_{res}(p)$ bits. Logo,

$$R(p) = MV(p) + B(p) + R_{res}(p). \tag{3.4}$$

Neste caso, para fins comparativos, os erros residuais após a compensação de movimento são comprimidos utilizando um algoritmo baseado em transformadas, um codificador JPEG (*Joint Photographic Experts Group*). Além disso, para uma para uma dada matriz com coeficientes de pequena ou média intensidade, a distorção $D(p)$ para um quantizador fixo não varia muito (em torno de 1% da variação). Logo, considerando $D(p)$ como sendo uma constante, o seguinte custo pode ser obtido:

$$J(p) = R(p), \tag{3.5}$$

ou seja, minimizando a taxa para uma distorção fixa.

3.4 ESTIMAÇÃO DE MOVIMENTO PARA O UTILIZAÇÃO DE PARTIÇÃO DE MACROBLOCOS BASEADOS EM *WEDGES*

Para viabilizar o esquema proposto, uma estimação de movimento (Seção 2.7.1) baseada em máscaras da partição em *wedges* é implementada, já que a representação matricial de uma *wedge*

seria relativamente complicado de se descrever. A máscara em questão é binária (como mostra a Figura 3.2) e multiplicada termo a termo pelo bloco atual que é deslocado dentro de uma janela de busca no quadro de referência, cuja posição de deslocamento onde possui distorção mínima seria relativo ao vetor de movimento da região cujo valor da máscara é ‘1’, a inversão da máscara deve ser feita de forma a achar o vetor de movimento relativo à outra região do macroloco. A Figura 3.4 mostra dois quadros subseqüentes e o movimento relativo dos objetos contidos nele, os vetores de movimento relativos ao deslocamento dos objetos que são buscados de maneira completa dentro de uma janela de busca.

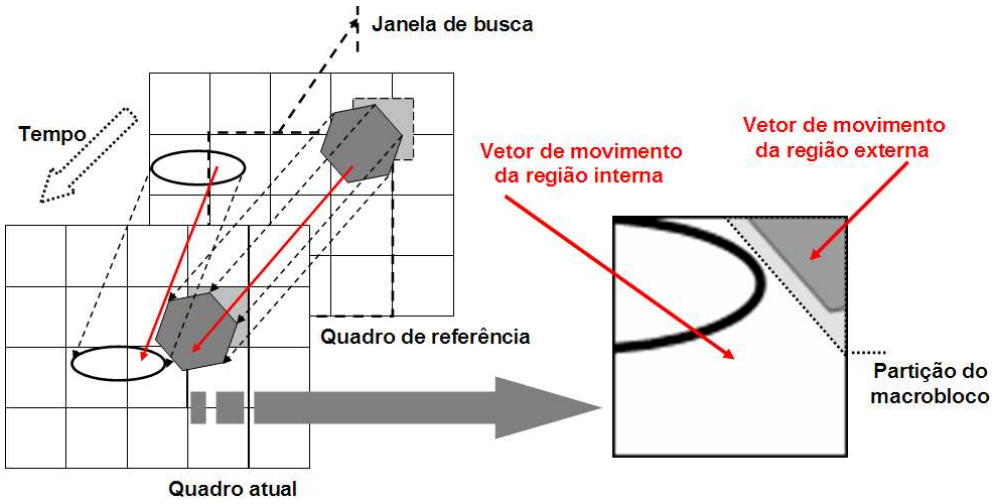

Figura 3.4: Estimação de movimento utilizando *wedges*.

No detalhe, dois objetos no macrobloco atual são mostrados, juntamente com uma máscara *wedge* que separam o macrobloco em duas regiões, finalmente, para cada região é associado um vetor de movimento. O decodificador necessita apenas da descrição das partições (se houver) do macrobloco e seus respectivos vetores de movimento para compensar o macrobloco a ser decodificado.

3.5 MÉTODO RÁPIDO PARA ESTIMAÇÃO E COMPENSAÇÃO DE MOVIMENTO BASEADO EM *WEDGES*

Como o conjunto de partição possível, em segmento de retas, de um macrobloco é muito grande, se faz essencial o uso de técnicas mais de processamento rápido para sua viabilização.

Para tanto, em qualquer que seja a escala do bloco, ao invés de se testar todas as partições possíveis de um macrobloco, propôs-se uma pré-seleção de um sub-conjunto de possíveis partições de bloco.

Já que o intuito de se compensar o movimento por partições (decomposições) *wedgelet* parte do princípio de que os blocos acompanhem os objetos em movimento. Então é razoável pré-selecionar as possíveis partições de um macrobloco por meio de bordas encontradas na mesma possa alcançar resultados satisfatórios.

Para um dado bloco de tamanho $N \times N$, aplica-se um algoritmo de detecção de borda, por exemplo, os filtros de Canny ou Sobel [40], seguido de uma operação de limiarização, obtendo assim uma matriz lógica bidimensional com o mesmo número de elementos igual ao número de pixels do macrobloco. Em seguida, os pixels com resposta positiva, ou seja, que caracterizam bordas, são mapeados em coordenadas Euclidianas de espaçamento em pixels resultando em pares (x, y). Por conseguinte, uma regressão linear (no caso, baseada em erro médio quadrático) é aplicada nesses pontos. O resultado é uma equação geral de uma reta: $y = ax + b$, ou $x = c$ (quando a regressão linear não for uma função). Esta linha é convertida em coordenadas polares, e associada a uma partição *wedge*. Segue-se a seguinte formulação. Seja (x_0, y_0) o centro do bloco, N o tamanho do bloco e θ_0 um offset que depende do sinal de $ax_0 + y_0 + b$, o que corresponde ao valor de $\theta_0 = 0°$ quando $ax_0 + y_0 + b$ é positivo e $\theta_0 = 180°$ caso contrário. Logo, temos como resultado os valores de r_{seed} e θ_{seed} que atuam como semente para a escolha do conjunto de *wedges*:

$$r_{seed} = \left| \frac{ax_0 + y_0 + b}{\sqrt{a^2 + 1}} \right| \tag{3.6}$$

$$\theta_{seed} = \arctan(a)\frac{180°}{\pi} + \theta_0 \tag{3.7}$$

No entanto, se a regressão resultar em $x = c$, utiliza-se a seguinte formulação matemática:

$$r_{seed} = \left| \frac{N}{2} - c \right| \tag{3.8}$$

$$\theta_{seed} = \theta_0 \tag{3.9}$$

onde θ_0 é um valor de *offset* que depende do sinal de $N/2 - c$, no qual associa-se o valor de $\theta_0 = 90°$ caso $N/2 - c > 0$, e $\theta_0 = 270°$ caso contrário. Selecionado as sementes r_{seed} e θ_{seed}, um conjunto de *wedges* é pré-selecionado variando suavemente os parâmetros de r e θ nos intervalos $[r_{seed} - \Delta r, r_{seed} + \Delta r]$ e $[\theta_{seed} - \Delta\theta, \theta_{seed} + \Delta\theta]$ com incrementos de δr e $\delta\theta$ respectivamente. Este processo é ilustrado na figura 3.5.

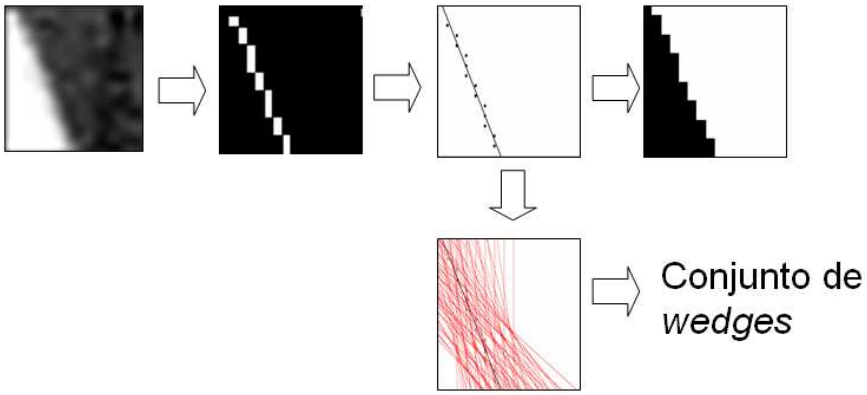

Figura 3.5: Selecionando (utilizando o algoritmo rápido) o conjunto de wedges para compensação de movimento.

Esta abordagem é eficiente para blocos que tenha em seu conteúdo bordas, singulares ou curvadas, bem definidas e pode vir a falhar caso o bloco contenha múltiplas bordas ou textura de alta freqüência, caso o método de multi-escala não for aplicado. Em seguida, a partição *quadtree* também é aplicada. Portanto, a estimação de movimento e a escolha de partição é restrita apenas aos conjuntos de *wedges* supracitados, diminuindo drasticamente a complexidade computacional.

3.6 PARTIÇÃO *WEDGE* PARA *SKIP* PARCIAL

Nos casos supracitados, a partição *wedge* associa a cada região particionada um vetor de movimento. Uma abordagem diferente será apresentada, uma suposição na qual um dos vetores de movimento é assumido nulo. O intuito desta abordagem seria inserir no codificador uma espécie de *"partial skip mode"* e, ao mesmo tempo, realizar apenas uma compensação de movimento em um dado macrobloco. Além de obtermos uma codificação com esforço

computacional menor do que as técnicas mencionadas anteriormente, uma vez que a escolha deste tipo de partição tende a ser analítica, este método traria vantagem na redução das partições de macrobloco, como veremos a seguir, a partição do macrobloco pode ser escolhida a partir da diferença entre o macrobloco no quadro de referência e o do quadro atual. Por meio de uma limiarização obtém-se um mapa binário onde são marcadas (mapeadas) as regiões onde existem movimentos significativos. Em uma das partições do macrobloco será feita a estimação de movimento como nas seções anteriores. Na outra será designado o movimento como "*skip*", ou seja, repete-se esta região da imagem do quadro anterior.

Para a determinação da partição do macrobloco, procura-se a partição *wedge* sobreposta (na mesma posição matricial) ao mapa binário que atenda os seguintes critérios:

1. A região nula da *wedge* não pode corresponder uma região não nula do mapa binário;
2. A região não nula da *wedge* pode corresponder uma qualquer informação do mapa binário;
3. A partição *wedge* deve possuir a maior região em que os zeros da *wedge* e do mapa binário coincidentes, dentre as partições que atendem os itens anteriores.

Um possível pseudo-código para implementar o *skip* parcial:

Seja um bloco (B_{atual}) de tamanho $N \times N$ em uma posição (x, y) de um quadro a ser codificado, um outro bloco ($B_{referencia}$) de mesmo tamanho $N \times N$ e mesma posição (x, y) e uma *wedge* de parâmetros r e θ.

- Faça a diferença entre os blocos $B_{referencia}$ e B_{atual} ($B_{diferena} = B_{referencia} - B_{atual}$);

  - Substitua cada elemento do bloco $B_{diferena}$ com o valor '1', caso seja diferente de zero. O resultado seria o que denominou-se de mapa binário.

- Atribua a uma *wedge* os valores de r e θ. Tipicamente os valores iniciais são nulos.

- Verificar se algum elemento nulo da *wedge* corresponde (na mesma posição) a um elemento não nulo do mapa binário.

  - Caso ocorra, descarte a *wedge*.

- Caso contrário, some o número de elementos nulos da *wedge*. Se o valor for maior que as anteriores escolha a $wedge_{candidata}$ r e θ como sendo candidata à partição.

- Repita a operação anterior incrementando que os valores de r e θ até que os valores finais estipulados sejam alcançados.

- A partição *wedge* escolhida é a $wedge_{candidata}$.

Note que a técnica de *skip* parcial pode ser modificada de forma a utilizar informações de predição de movimento, ao invés de movimento nulo. Modificando apenas o cálculo do mapa binário, ou seja, o bloco de referênica $B_{referencia}$ possui posição diferente do bloco atual B_{atual}, onde sua posição é deslocado do vetor de movimento predito.

Observe ainda que a informação residual deste tipo de macrobloco tende a ter duas regiões distintas: uma com resíduo ordinário e outra com resíduo nulo (ou quase nulo), cuja informação residual pode ou não ser descartada. No caso de optar por descartar tal informação, pode-se utilizar os métodos de preenchimento de dados (*data filling*), que são utilizados para garantir a compactação da energia de blocos que necessitem preenchimento [41].

4 EXPERIMENTOS

4.1 INTRODUÇÃO

Nesta dissertação serão apresentados dois codecs que comparam alguns métodos de partição de macrobloco, explorando a correlação temporal inerente nas seqüências de vídeo. O primeiro é implementado no MPEG-2 adicionado a técnicas de compensação de movimento, que serão comparadas experimentalmente.

O outro codec (codificador e decodificador) de vídeo foi baseado nas técnicas de compressão de vídeo canônicas: MPEG-2 e H.264, o primeiro o padrão de compressão mais popular, já o segundo é atualmente o mais eficiente em termos de qualidade/taxa de compressão. Apesar deste codificador ser baseado nestas duas técnicas o mesmo não tem o propósito de competir com ambos, haja vista que o objetivo deste codec seria de testar módulos e/ou técnicas de compressão de vídeo agregados ou inseridos no mesmo, possibilitando uma comparação entre técnicas.

4.2 IMPLEMENTAÇÃO DE UM MPEG-2 MODIFICADO

O codec nesta dissertação é resultado da implementação do MPEG-2 em seu perfil mais básico e da adição de técnicas para compensação de movimento, juntamente com algumas adaptações para que as técnicas de partição sejam comparadas. O codec faz uso do JPEG para a codificação do primeiro quadro. Os quadros subseqüentes são preditos a partir da compensação do quadro anterior até o final da seqüência. Ou seja, o grupo de figuras (GOP) é do tipo IPPP, sendo retratado na Figura 4.1. Nos quadros do tipo P a imagem é dividida em macroblocos, onde para cada macrobloco se faz uma estimação de movimento utilizando a busca completa em uma janela de 32×32 pixels baseada na minimização da SAD. De posse do vetor de movimento, o macrobloco em questão é compensado e reconstruído. A reconstrução é feita subtraindo o bloco atual (a informação original) pelo bloco compensado, gerando um resíduo de compensação que em seguida é comprimido com o JPEG. O sistema supracitado pode ser observado na Figura 4.2.

A partir dessas informações são calculados, dois parâmetros:

1. Distorção: métrica da diferença entre o bloco reconstruído e o bloco original. Existem basicamente duas métricas amplamente utilizadas na literatura, a SAD (soma das diferenças absolutas - Equação 4.1) ou a SSD (soma das diferenças quadráticas - Equação 4.2).

$$D_{SAD} = \sum_{i=1}^{N}\sum_{j=1}^{N} \left| pixel_{original}(i,j) - pixel_{reconstruido}(i,j) \right|, \tag{4.1}$$

$$D_{SSD} = \sum_{i=1}^{N}\sum_{j=1}^{N} \left(pixel_{original}(i,j) - pixel_{reconstruido}(i,j) \right)^2. \tag{4.2}$$

2. Taxa: quantificação dos bits utilizados de forma que o decodificador consiga reconstruir a imagem. Neste caso é a soma dos bits da partição (caso houver), do(s) vetor(es) de movimento e do resíduo comprimido.

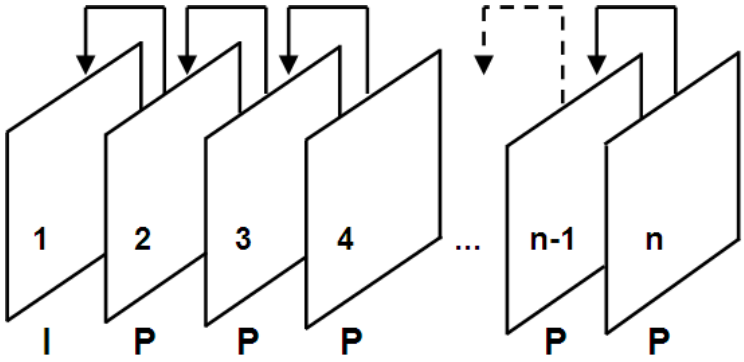

Figura 4.1: GOP utilizado.

Os vetores de movimento são codificados por um código de Huffman, e a descrição das partições são codificadas uniformemente. O processo de decodificação é bastante simplificado. Primeiramente, se faz uma decodificação no JPEG, que em seguida o quadro seguinte é compensado a partir do anterior, contudo a compensação neste caso é um pouco diferente do MPEG-2 padrão, pois a informação da partição e os vetores de movimento devem ser aplicados a cada região. Finalmente, o resíduo decodificado também com o JPEG é adicionado ao quadro compensado resultando em uma seqüência de vídeo reconstruída.

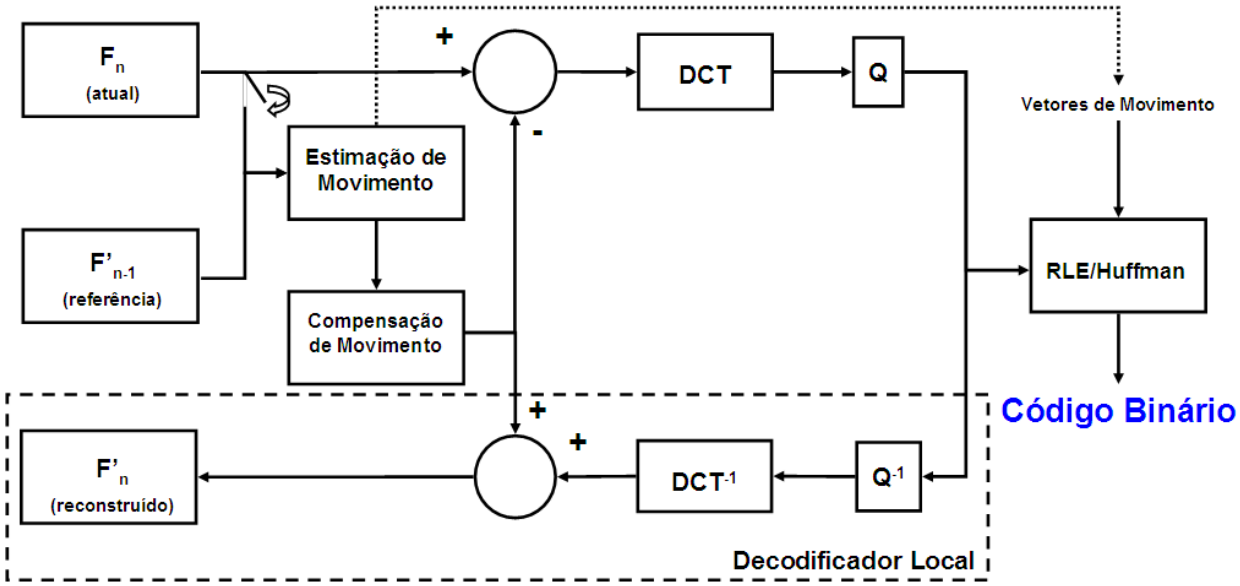

Figura 4.2: Diagrama de blocos simplificado do MPEG-1 e MPEG-2.

4.3 COMPARAÇÃO ENTRE OS MÉTODOS DE PARTIÇÃO DE MACROBLOCOS

Com o intuito de conhecer a efetividade e as limitações do método proposto (no caso as partições *wedges* e seu algoritmo sub-ótimo), comparou-se o mesmo com outras técnicas de compensação de movimento utilizando blocos de 16×16 pixels. A precisão usada nos vetores de movimento foi de um pixel, e foram feitas buscas completas para estimação de movimento dentro de uma janela de 32×32 pixels.

As técnicas utilizadas para a comparação foram as seguintes:

- Compensação de movimento sem partição de macroblocos.
- Compensação de movimento utilizando *quadtree* (vide figura 3.1).
- Compensação de movimento utilizando *wedges* (vide seção 3.3).
- Compensação de movimento utilizando o algoritmo rápido de *wedges* (vide seção 3.5).

Observe pela figura 4.3 que ao compensarmos o movimento, ou seja, estimamos o quadro seguinte baseado no anterior utilizando apenas um vetor de movimento para cada macrobloco.

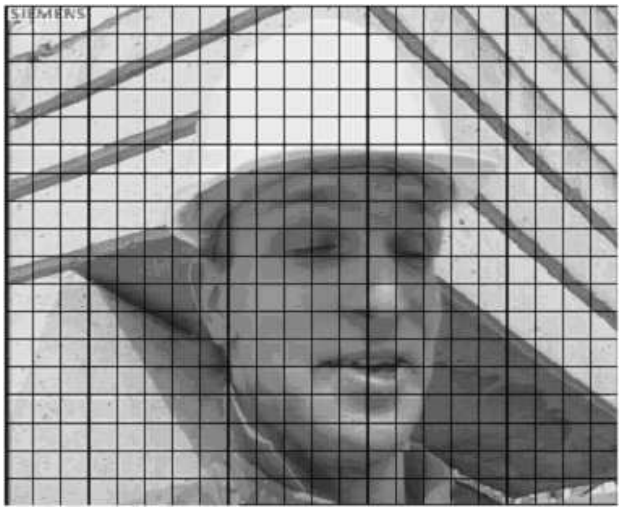

Figura 4.3: Imagem compensada sem partição de macroblocos.

Já utilizando a decomposição *quadtree*, ilustrada na figura 4.4, cada macrobloco pode ser compensado inteiro ou particionado em metades de 16×8, 8×16 e até mesmo em quatro sub-blocos de 8×8. E para cada sub-bloco, o mesmo pode ser compensado inteiro ou dividido em duas partes (8×4 ou 4×8) ou, por fim em blocos de 4×4. Para cada região particionada, se associa um vetor de movimento.

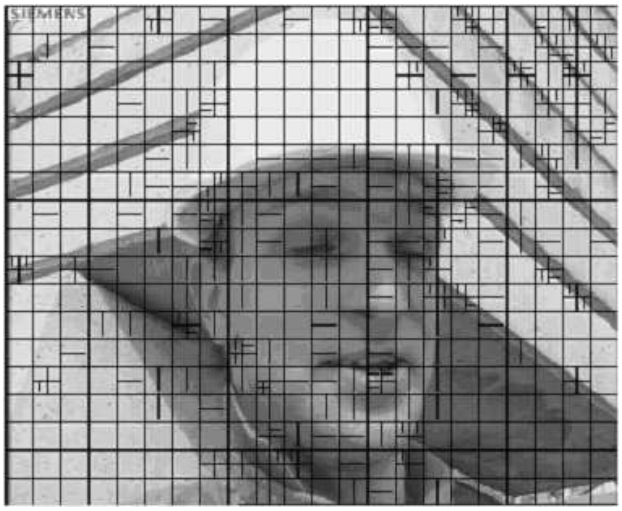

Figura 4.4: Imagem compensada utilizando a decomposição *quadtree*.

A técnica proposta nesta dissertação generaliza a decomposição *quadtree*, onde são adicionados ao conjunto de partições *wedges* de macroblocos e sub-macroblocos em partições em duas regiões. Isto implica em um número variável de vetores de movimento por macrobloco. Já as decisões de partição do macrobloco são feitas baseados no critério de minimização da função de

custo (*Rate Distortion Cost*). Nos experimentos, o dicionário de *wedges*, isto é, o conjunto das possíveis partições em uma escala de bloco de tamanho $N \times N$ *pixels*, foi obtido incrementando o valor de r e θ em passos unitários no intervalo de $0 \leq r < N/2$ e $0° \leq \theta < 360°$. Depois de removidas as partições redundantes, obtemos um dicionário contendo 2012 partições *wedges* para macroblocos de tamanho 16×16 *pixels* e 340 partições para os sub-macroblocos de 8×8 *pixels*. Neste caso em particular, um código prefixado é utilizado para indicar a escala da partição adicionando símbolos equiprováveis (exceto para as partições *quadtree*), resultando em um quadro compensado ilustrado pela figura 4.5.

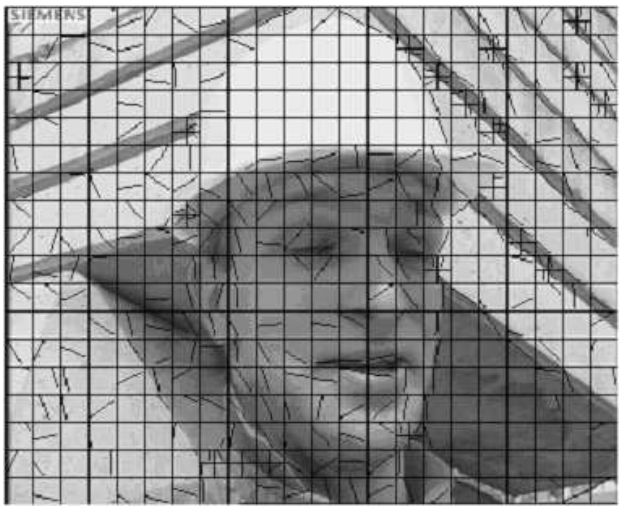

Figura 4.5: Quadro compensado utilizando *wedges*.

Note que a tendência das partições é acompanhar as bordas dos objetos em cena, assim como esperado. As partições que não acompanham as bordas possuem matematicamente o menor custo e são diretamente dependentes do quadro anterior e a referida partição resulta seguramente no menor resíduo (diferença entre o macrobloco real e o estimado).

Por conseguinte, foi proposta uma abordagem para reduzir a quantidade de *wedges* que possuem seus vetores de movimento estimados. Onde as partições *wedge* que terão a estimação de movimentos calculados são pré-selecionados de acordo com um critério de bordas (descrito na seção 3.5), resultado em um quadro compensado exemplificado pela Figura 4.6.

Comparando objetivamente as técnicas supracitadas por meio da curva entre a taxa e qualidade do vídeo após a compressão do mesmo. A Figura 4.7 mostra a curva de taxa $\times$ distorção comparando os quatro métodos, a redução da taxa de bits é de aproximadamente 4,7% ao

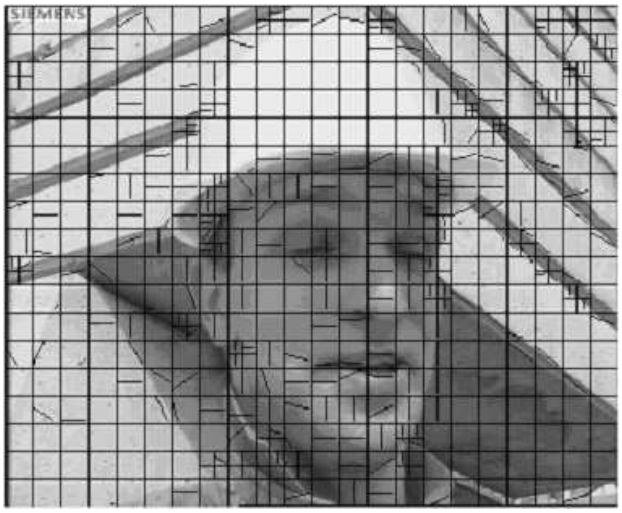

Figura 4.6: Quadro compensado utilizando uma técnica sub-ótima para *wedges*.

utilizarmos a técnica de compensação por *wedges* é de 3,1% a favor da técnica de sub-ótima de compensação por *wedges* ambas comparadas com a técnica de decomposição por *quadtree*. Os métodos baseados em *wedges* economizam mais de 20% comparando com o método tradicional sem partição de macroblocos. Em termos de PSNR (*Peak Signal-to-Noise Ratio*) obteve-se um ganho de 1 dB ao compararmos com a compensação tradicional (sem partição). Se compararmos com o *qudtree* verifica-se um ganho entre 0,3 dB e 0,2 dB para as técnicas utilizando o *wedge*.

4.4 ESTRUTURA DO CODEC DE VÍDEO PROPOSTO

O compressor de vídeo apresentado (figura 4.8) possui a estrutura baseada no DPCM, onde existe um estimador temporal, no caso a estimação e a compensação de movimentos. Os quadros da seqüência são comprimidos com GOP do tipo IPPP, onde o quadro “intra” é comprimido utilizando um JPEG modificado. Neste se utiliza a transformada e a quantização do H.264. Os quadros subsequentes são comprimidos com relação aos anteriores, e os macroblocos podem ser codificados nos modos “intra”, “inter” e *skip*. Implica-se assim na estimação de um novo quadro que possuirá seu resíduo (diferença entre o quadro estimado e o quadro a ser comprimido) codificado utilizando o H.264 no modo de quadros intra. A informação dos vetores de movimento e das partições dos macroblocos são comprimidos com um codificador de entropia denominado *Range Coder*, que é uma implementação do codificador aritmético em valores inteiros. Gera-se,

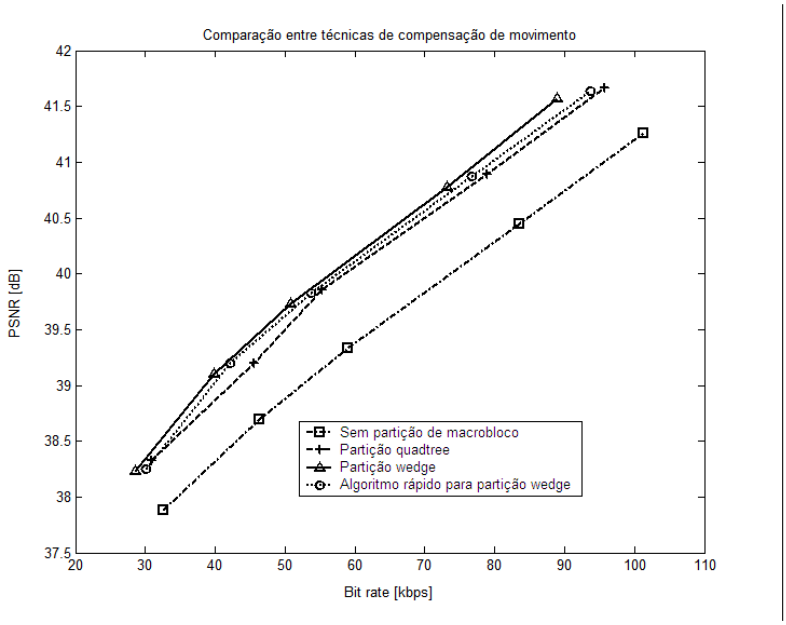

Figura 4.7: Curva de Taxa $\times$ Distorção comparando várias técnicas de compensação de movimento: sem partição de macroblocos, macroblocos particionados com *quadtree*, macroblocos particionados com *wedge* e macroblocos particionados com *wedge* utilizando uma algoritmo rápido [1].

por fim, um *stream* de bits na saída.

O *stream* de bits gerado pelo codificador é descomprimido utilizando um decodificador de entropia (*Range Coder*) resultando na informação dos quadros ainda comprimidos com o padrão H.264 intra, os vetores de movimento e as partições. Em seguida, o primeiro quadro é decodificado pelo H.264, e os quadros seguintes são compensados a partir das informações da partição e dos vetores de movimento. Estes quadros estimados são somados com seus resíduos correspondentes após uma decodificação apropriada, conforme ilustra a figura 4.9.

Para o cálculo dos vetores de movimento foi feita a busca completa dentro de uma janela de tamanho 32×32 pixels, utilizando uma precisão de um pixel. Apesar de ter sido implementado uma estimação de movimento de um quarto de pixel não foi utilizada por apresentar uma complexidade computacional intensa, mas gerando resultados similares, de acordo com testes realizados.

As técnicas de compensação de movimento comparadas por este codec foram as seguintes:

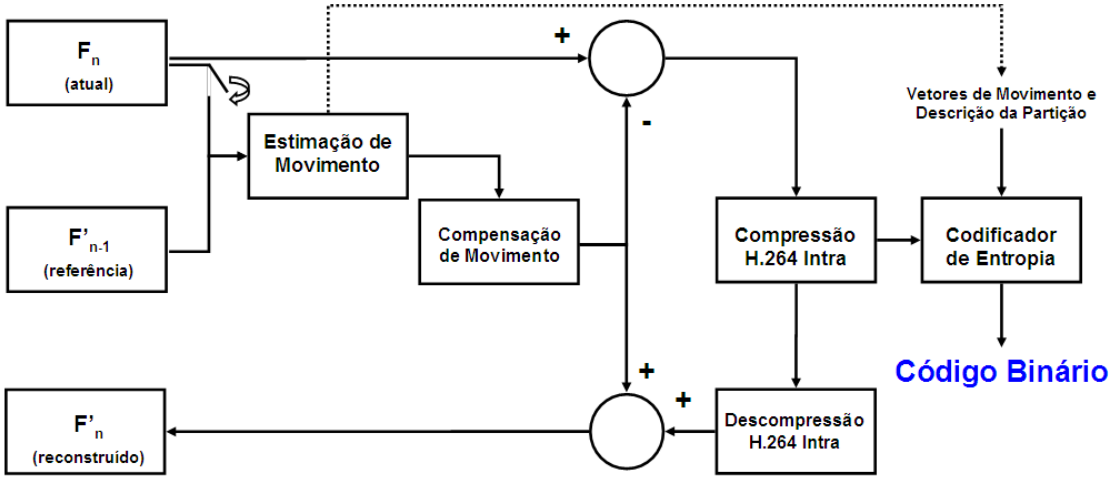

Figura 4.8: Esquemático do codificador de vídeo proposto

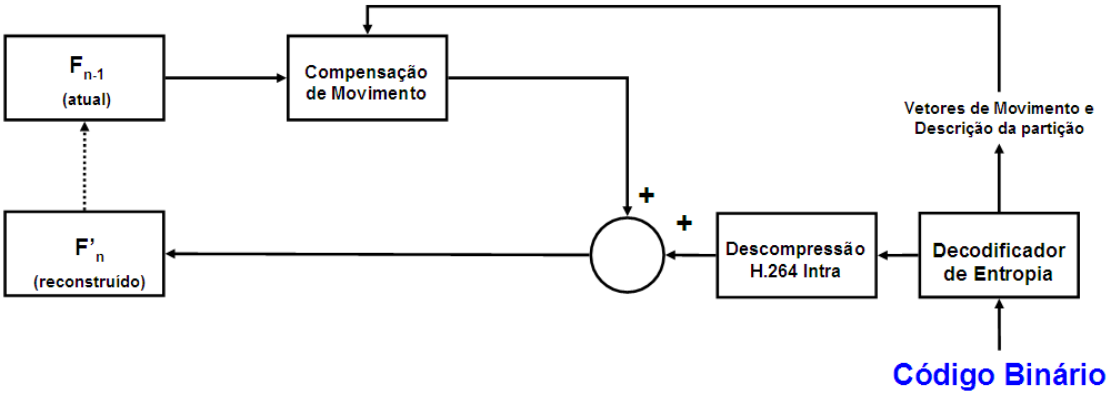

Figura 4.9: Esquemático do decodificador de vídeo proposto.

- Compensação de movimento utilizando *quadtree* (vide figura 3.1).
- Compensação de movimento utilizando o algoritmo rápido de *wedges* (vide seção 3.5).
- Compensação de movimento utilizando partições *wedge* para *skip* parcial (vide seção 3.6).

Os resultados visuais em termos de partição de macroblocos são similares ao do MPEG-2 modificado, tanto nas partições to tipo *quadtree* (vide Figura 4.4) quanto *wedge* (vide Figura 4.6). Neste codificador foi utilizado a partição *wedge* baseada em bordas (o método rápido) associando dois vetores de movimento. Este método será comparado com o *quadtree* e a compensação de movimento com *skip* parcial (*wedge* com um vetor de movimento).

A Figura 4.10(a) ilustra as regiões onde possui movimento (diferenças significativas entre o quadro de referência e o atual) e na Figura 4.10(b) a aplicação das máscaras *wedges* nas regiões

onde possui movimento distinguindo a região que possui (por inferência) vetor de movimento nulo e a região que possui um vetor de movimento associado.

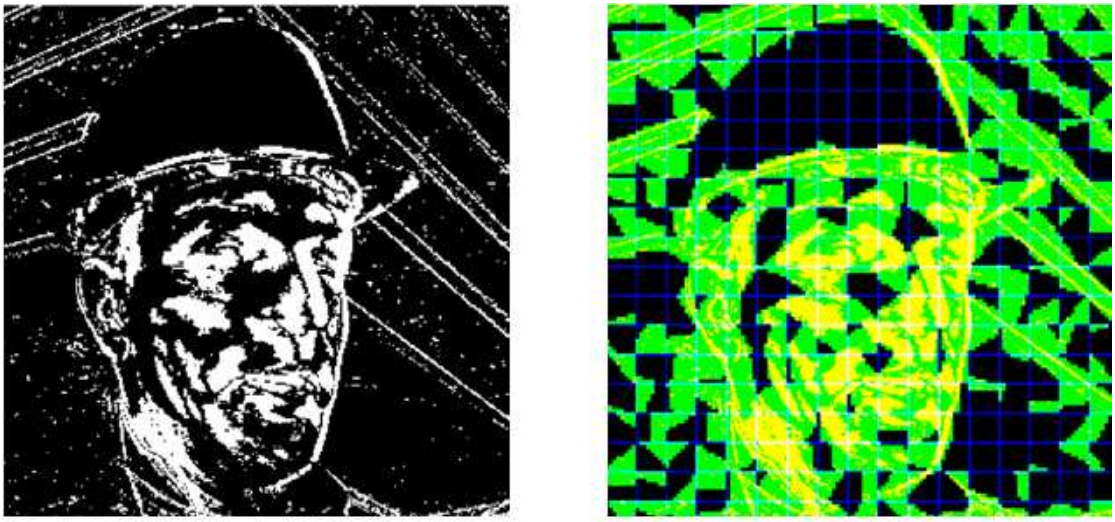

Figura 4.10: (a) Mapa binário onde se detecta pixels que possuem movimento. (b) Sobreposição de *wedges* no mapa binário.

A compressão foi realizada em três seqüências de vídeo populares: *Foreman*, *Coastguard* e *Mother and Daughter*. Os resultados da compressão podem ser observados nas Figuras 4.11, 4.12 e 4.13.

Observe que o comportamento esperados das curvas não foram alcançados, ou seja, os ganhos relativos às partições não foram atingidos diferente do que foi mostrado na Figura 4.7. Isso ocorre devido a uma implementação do codificador de entropia de uso geral, ou seja: sem otimização, sem contexto e tampouco uma estatística que retrate a ocorrência dos símbolos.

Outra técnica que pode contribuir para melhoria de desempenho é avaliar a distorção causada pela *wedge* que não possui "vetor de movimento", remetendo assim a utilização de técnicas de otimização do sistema de forma a obter a melhor relação taxa distorção neste tipo de sistema de compressão, como a estimação empírica dos multiplicadores de Lagrange [39].

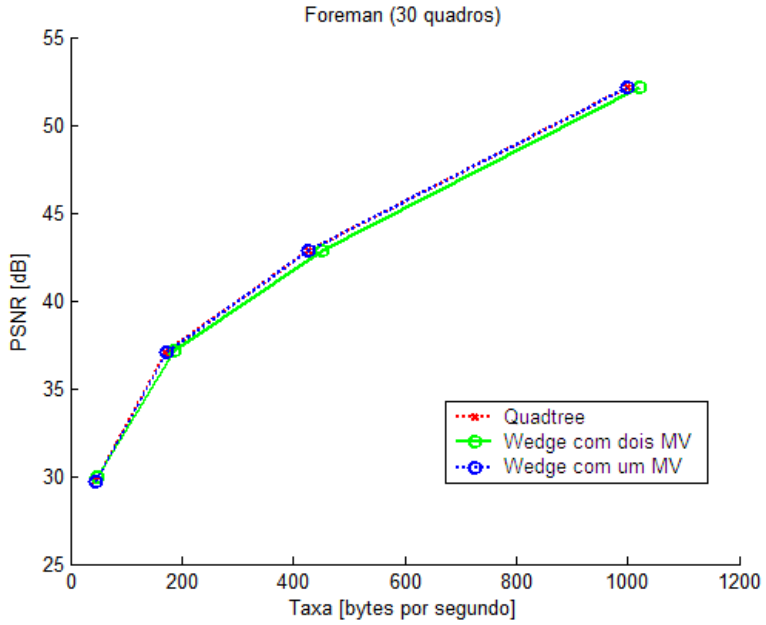

Figura 4.11: Curva de Taxa × Distorção comparando várias técnicas de compensação de movimento para a seqüência de vídeo *Foreman*.

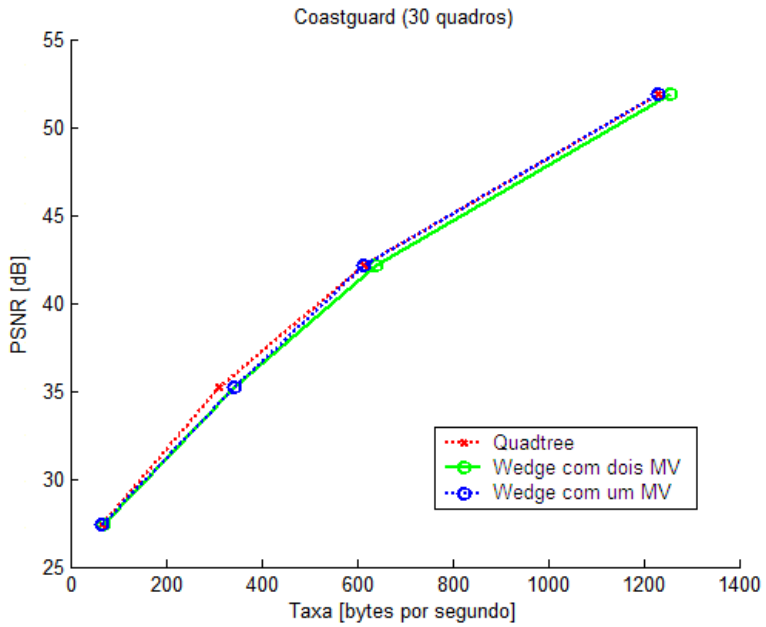

Figura 4.12: Curva de Taxa × Distorção comparando várias técnicas de compensação de movimento para a seqüência de vídeo *Coastguard*.

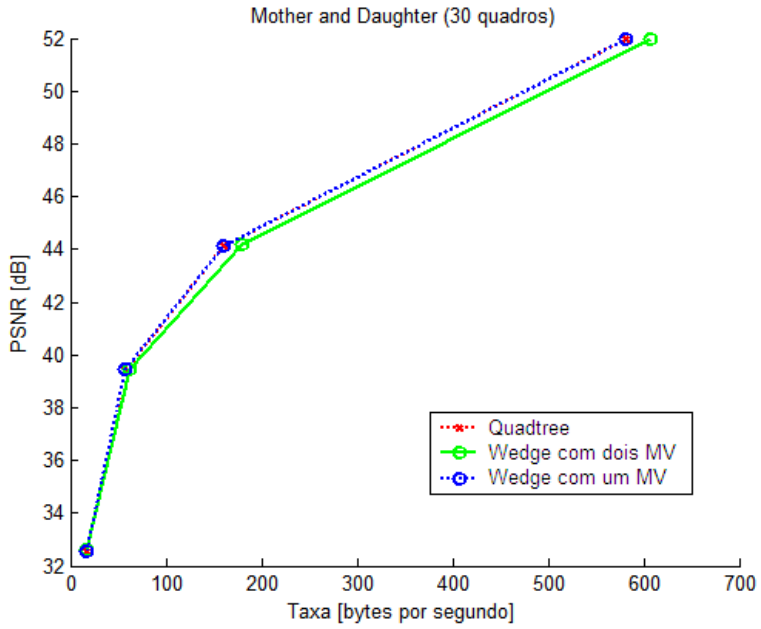

Figura 4.13: Curva de Taxa × Distorção comparando várias técnicas de compensação de movimento para a seqüência de vídeo *Mother and Daughter*.

5 CONCLUSÕES

Neste trabalho propôs-se técnicas de compressão de vídeo mais genéricas e mais eficientes comparadas às disponíveis no âmbito industrial e acadêmico. Para comparar as técnicas desenvolvidas e as atuais, foram desenvolvidos codecs (codificador e decodificador) de vídeo cujos módulos são baseados nos padrões de vídeo MPEG-2 e H.264. O objetivo principal seria de testar e comparar as técnicas aqui propostas com as utilizadas classicamente nos codecs de vídeo dantes mencionado, e não propor um novo sistema de compressão que seja competitivo com o estado da arte em compressão de vídeo. Esta dissertação propõe métodos mais genéricos e eficientes de compensação de movimento, especificamente de partição de macroblocos. Ou seja, nesta dissertação apenas a correlação temporal (predição entre quadros) foi explorada.

Comparamos as técnicas de compensação mais tradicionais, no caso, sem partição de macroblocos (utilizada no MPEG-2) e a partição *quadtree* (utilizada no H.264), além de propor três técnicas para compensação de movimento baseadas em *wedgelets*. Onde o tamanho do dicionário de *wedges* pode ser variado de acordo com a necessidade do codificador ou pela aplicação. Mas quanto maior o dicionário, maior o número de bits para codificar (representar) a partição e por conseguinte aumentam as opções de compensação e estimação de movimento (aumentando o esforço computacional). Para a viabilização desta técnica, propôs-se neste trabalho uma abordagem onde a partição do macrobloco é baseada nas bordas dos objetos encontrados no macrobloco. Resultando numa técnica sub-ótima, mas com uma redução drástica no esforço computacional sem influenciar significativamente na qualidade, uma vez que esta técnica reduz o conjunto de testes do dicionário completo. Por fim, a última proposta foi de utilizar as partições *wedge* para a implementação de um *skip* parcial.

Outros trabalhos mais recentes, denominados de "*geometry-adaptive block partitioning*" [36, 37, 38], que é idêntico à implementação das *wedges* no H.264, pode ser verificado que os resultados foram similares aos obtidos nesta dissertação.

5.1 TRABALHOS FUTUROS

Depois de observado o potencial da utilização de partições *wedge* por meio da experimentação em um codec mais simplificado, as técnicas supracitadas devem ser incluídas nos códigos de referência do H.264 AVC - JM(*Joint Model*) ou do H.264 SVC *Scalable Video Coding* - JSVM (*Joint Scalable Video Model*). O H.264 é o codec escolhido para esta implementação por ter melhor desempenho em termos de taxa distorção, além de ser academicamente e industrialmente difundido. De forma que a técnica de partição *wedge*, ou alguma outra baseada na mesma, possa eventualmente ser adicionada ao H.264 ou padrões de vídeo a serem desenvolvidos.

Outras contribuições que poderiam extender este trabalho seriam: (i) o desenvolvimento de técnicas menos complexas, mesmo que sub-ótimas para a escolha das partições *wedge* de um bloco; (ii) curvar a partição *wedge* utilizando parábolas ou funções polinomiais de ordem maior; (iii) fazendo *tree prunning* na descrição de uma *binary partition tree* correspondente a uma imagem, que aglutinaria os blocos em regiões maiores que um macrobloco, adaptando melhor a região compensada a um objeto de uma imagem.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] HUNG, E. M.; QUEIROZ, R. L. de; MUKHERJEE, D. On macroblock partition for motion compensation. *International Conference on Image Processing*, Atlanta, USA, p. 1697–1700, October 2006.

[2] RICHARDSON, I. E. G. *H.264 and MPEG-4 Video Compression*. [S.l.]: John Wiley & Sons Ltd, 2003.

[3] ITU-T and ISO/IEC JTC 1 - ISO/EC 13818-2 (MPEG-2). *Generic coding of moving pictures and associating audio information - Part 2: Video*. [S.l.], November 1994.

[4] ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4). *Advanced Video Coding for Generic Audiovisual Services*. [S.l.], Version 1, May 2003; Version 2, January 2004; Version 3, September 2004; Version 4, July 2005.

[5] QUEIROZ, R. L. de et al. Fringe Benefits of the H.264/AVC. In: *International Telecommunication Symposium*. [S.l.: s.n.], 2006. p. 208–212.

[6] ITU-T (formalmente CCITT) and ISO/IEC JTC1. *Digital Compression and Coding of Continuous-Tone Sitll Images*. [S.l.], September 1992.

[7] ITU-T Recommendation T.800 and ISO/IEC 15444-1. *Image Coding System: Core Coding System (JPEG2000 Part 1)*. [S.l.], September 2000.

[8] TAUBMAN, D. S.; MARCELLIN, M. W. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. [S.l.]: Kluwer Academic, 2002.

[9] TAUBMAN, D. *Directionality and Scalability in Image and Video Compression*. Tese (Doutorado) — University of California - Berkeley, 1994.

[10] SHAPIRO, J. M. Embedded image coding using zerotrees of wavelet coeficients. *IEEE Transactions on Signal Processing*, p. 3445–3462, December 1993.

[11] SAID, A.; PEARLMAN, W. A. A new fast and efficient coder based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, p. 243–250, June 1996.

[12] ITU-T. *Video Codec for Audiovisual Services at px64 kbit/s*. [S.l.], Version 1, November 1990; Version 2, March 1993.

[13] LUTHRA, A.; SULLIVAN, G. J.; WIEGAND, T. H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 13, n. 7, July 2003.

[14] SAYOOD, K. *Introduction to Data Compression*. [S.l.]: Morgan Kuffmann Publishers, 2000.

[15] SHARMA, G. *Digital Color Imaging Handbook*. [S.l.]: CRC Press, 2002.

[16] SHANNON, C. E. A mathematical theory of communication. *Bell Syst. Tech.*, n. J.27, p. 379–423, 623–656, 1948.

[17] HUFFMAN, D. A method for the construction of minimum redudancy codes. *Proc. IRE*, n. 40, p. 1098–1101, 1952.

[18] RAO, K. R.; HWANG, J. J. *Techniques and Standards for Image, Video and Audio Coding*. [S.l.]: Prentice Hall, 1997.

[19] ITU T.88 and ISO/IEC 14492. *JBIG2*. [S.l.], April 2001.

[20] ISO/IEC 15444-3:2002/Amd 2:2003. *Image Coding System: Motion JPEG 2000 (JPEG2000 Part 3)*. [S.l.], September 2003.

[21] KOGA, T. et al. Motion-compensated interframe coding for video conferencing. *Proc. NTC*, p. 961–965, November 1981.

[22] LI, R.; ZENG, B.; LIOU, M. L. A new three-step search algorithm for block estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 4, p. 438–443, August 1994.

[23] PO, L. M.; MA, W. C. A novel four-step search algorithm for fast block estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 6, p. 313–317, June 1996.

[24] THAM, J. Y.; RANGANATH, S.; KASSIM, A. A. A novel unrestricted center-biased diamond search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 8, n. 4, p. 369–377, August 1998.

[25] ZHU, S.; MA, K. A new diamond search algorithm for fast block matching motion estimation. *ICICS'97*, Sinagpore, p. 9–12, September 1997.

[26] SULLIVAN, G. J. et al. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 13, July 2003.

[27] ARCHIBALD, R.; GELB, A. A method to reduce the gibbs ringing artifact in mri scans while keeping tissue boundary integrity. *IEEE Transactions on Medical Imaging*, v. 21, p. 305–319, April 2002.

[28] MARPE, D. et al. Video compression using context-based adaptive arithmetic coding. *International Conference on Image Processing*, Thessaloniki (GR), p. 558–561, September 2001.

[29] MARPE, D.; SCHWARZ, H.; WIEGAND, T. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard,. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 13, n. 7, p. 620–636, July 2003.

[30] ITU-T. *ITU-T Recommendation H.263, Video coding for low bit rate communication*. [S.l.], November 2000.

[31] TEKALP, A. M. *Digital Video Processing*. New Jersey, USA: Prentice-Hall, 1995.

[32] JVT of ISO/IEC MPEG and ITU-T VCEG. *Advanced Video Coding for Generic Audiovisual Services*. [S.l.], March 2005.

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

[34] KONDO, S.; SASAI, H. A motion compensation technique using sliced blocks in hybrid video coding. *IEEE International Conference on Image Processing*, Genova, Italy, September 2005.

[35] ITU-T. *Joint Model Number 1 (JM-1) - JVT-A003*. [S.l.], January 2005.

[36] DIVORRA, . et al. Geometry-adaptive block partitioning for video coding. *International Conference on Acoustics, Speech, and Signal Processing*, April 2007.

[37] DIVORRA, .; YIN, P.; GOMILA, C. *Geometry-Adaptive Block Partitioning*. VCEG 32nd Meeting, April 2007.

[38] DAI, C. et al. Geometry-adaptive block partitioning for intra prediction in image and video coding. *International Conference on Image Processing*, September 2007.

[39] WIEGAND, T.; GIROD, B. Lagrange multiplier selection in hybrid video coder control. *International Conference on Image Processing*, p. 558–561, September 2001.

[40] PRATT, W. K. *Digital Image Processing: PIKS Inside*. California, USA: Wiley-Interscience, 2001.

[41] QUEIROZ, R. L. de. On data-filling algorithms for MRC layers. *International Conference on Image Processing*, II, p. 586–589, September 2000.