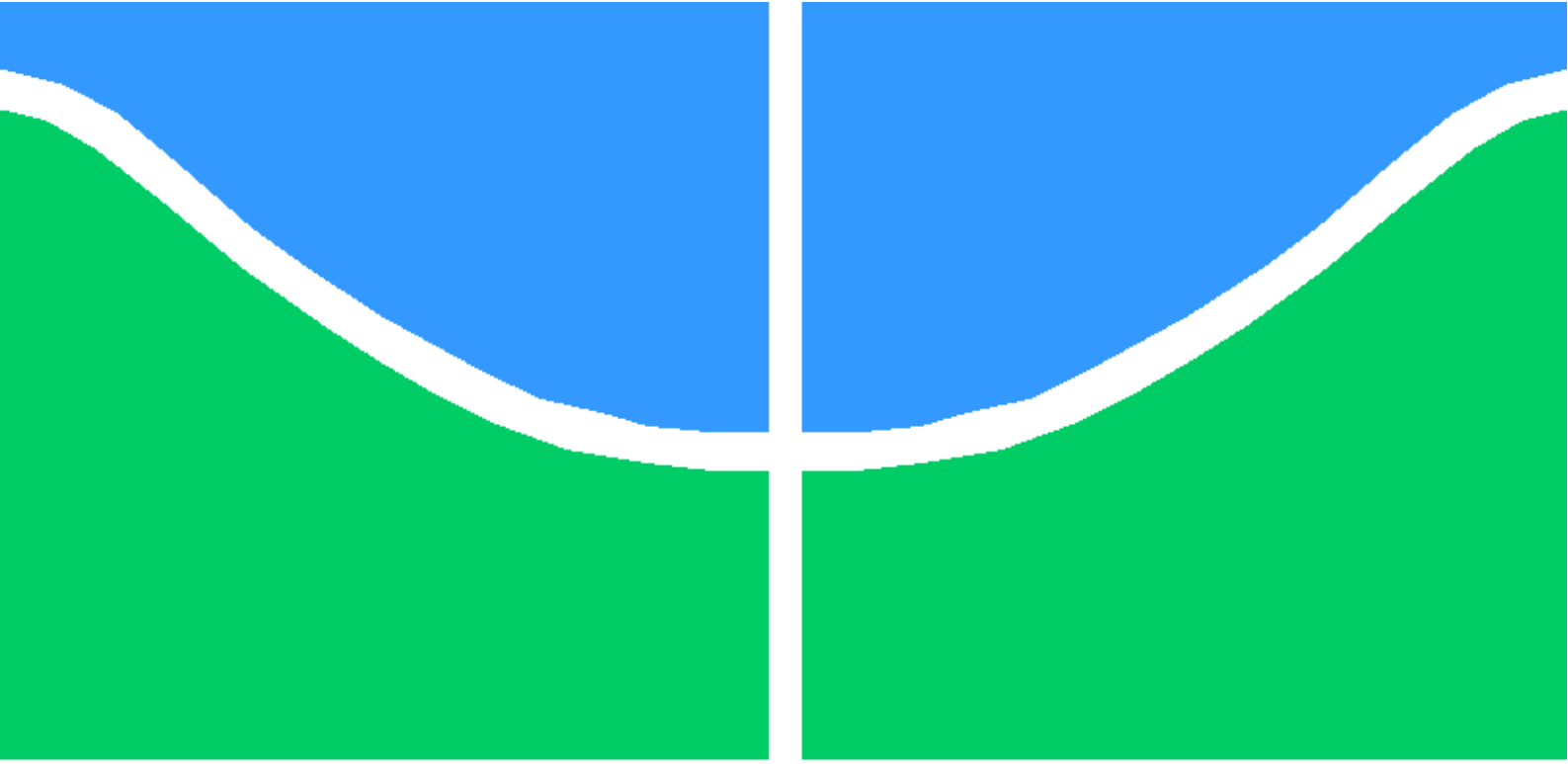


DISSERTAÇÃO DE MESTRADO EM
ENGENHARIA ELÉTRICA

TRANSCODIFICADOR DE VÍDEO WYNER-ZIV/H.263 PARA COMUNICAÇÃO ENTRE DISPOSITIVOS MÓVEIS

Eduardo Peixoto Fernandes da Silva

Brasília, fevereiro de 2008

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

TRANSCODIFICADOR DE VÍDEO WYNER-ZIV/H.263 PARA COMUNICAÇÃO ENTRE DISPOSITIVOS MÓVEIS

Eduardo Peixoto Fernandes da Silva

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)

FICHA CATALOGRÁFICA

SILVA, EDUARDO PEIXOTO
Transcodificador de Vídeo Wyner-Ziv/H.263 para
Comunicação entre Dispositivos Móveis. [Distrito Federal] 2008.
xiv, 99p., 297 mm (ENE/FT/UnB, Mestre, Telecomunicações
Processamento de Sinais, 2008). Dissertação de Mestrado.
Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Elétrica.

1. Transcodificação	2. Wyner-Ziv
3. Codificação de Vídeo Distribuída	4. Codificação de Baixa Complexidade
5. H.263	6. Vídeo para Dispositivos Móveis
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

PEIXOTO, E. F. (2008). Transcodificador de Vídeo Wyner-Ziv/H.263 para Comunicação entre Dispositivos Móveis. Dissertação de Mestrado em Engenharia Elétrica com enfase em Telecomunicações, Publicação MTARH.DM - 326 A/08, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 99p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Eduardo Peixoto Fernandes da Silva.

TÍTULO DA DISSERTAÇÃO DE MESTRADO: Transcodificador de Vídeo Wyner-Ziv/H.263 para Comunicação entre Dispositivos Móveis.

GRAU / ANO: Mestre / 2008

Eduardo Peixoto Fernandes da Silva
CCSW 2 Lote 5 Bloco A Apto 406
70680-250 Sudoeste, Brasília - DF - Brasil.

Dedicatória

À minha esposa, Ana Carolina, e à minha família, em especial minha mãe, que me ajudaram durante todo o mestrado.

Eduardo Peixoto Fernandes da Silva

Agradecimentos

Agradeço à minha família, minha esposa Ana Carolina, minha mãe, Eliane, meu pai, Sinval, meus irmãos André, Danielle, Luísa e Arthur, meus sobrinhos Alana, Aline, Gabriel, Gabriela e Thiago, minha madrasta Ana, meus cunhados Rodrigo e Sandra, e minha sogra, Geisa, por fazerem parte da minha vida.

Aos professores e funcionários da Universidade de Brasília, em especial os professores Paulo Henrique Portela, Leonardo R.A.X., Arthur Vicentini, Nigel Pitt, Geovany Borges, Ricardo Zelenovsky e Adson Rocha, que sempre me incentivaram e me ajudaram na minha caminhada até aqui.

Ao meu orientador, Ricardo Queiroz, por sua ajuda, críticas e sugestões ao trabalho. O trabalho não teria ficado pronto sem sua conhecida exigência.

Aos pesquisadores do ***HP Labs*** *em Palo Alto, Doutora Yuxin Liu (Zoe) e, em especial, ao Doutor Debargha Mukherjee, que contribuiu com idéias, apoio e até mesmo apresentando este trabalho em conferência. Ao pessoal da HP Brasil, Ricardo Pianta, Paulo Sá e Marcelo Thielo, sem os quais o convênio da UnB (e esta dissertação) não existiriam.*

Aos companheiros de trabalho no GPDS, Rafael "Capim", Rafael Ortis, Fernanda, Zaghetto, Bruno, Mintsu, Tiago, Karen, Renan, Maria do Carmo, Frederico, Chaffim, Diogo "Buraco", Patrick e Alberto, que formam um excelente ambiente de trabalho, compartilhando conhecimento, risadas e até mesmo uma cafeteira expresso.

Aos meus amigos da UnB, cujos nomes são muitos mas que estão sempre juntos, particulamente àqueles que, como eu, ingressaram no curso de mestrado, Maisa, Rafael "Astronauta", Fernando, Sauro e Renato. Aos amigos do Só Alegria Moto Clube, companheiros de viagens (próximo destino: Ushuaia, viu?) e reuniões na "segunda sexta-feira do mês". Aos amigos de longa data, Otávio, Márcio, Mariana, Marcela, Luís "Nando", Joana, Natália, Paulo, Antônio, Patrícia, e muitos outros, por simplesmente "estarem lá".

Por fim, um agradecimento especial ao Bruno, Mintsu e Zaghetto, que por muitas vezes eu importunei para tirar dúvidas e revisar a dissertação.

Eduardo Peixoto Fernandes da Silva

RESUMO

Em comunicações de vídeo entre dispositivos móveis, tanto o terminal transmissor quanto o terminal receptor podem não ter os recursos computacionais necessários para realizar tarefas complexas de compressão e descompressão de vídeo. Codificadores de vídeo tradicionais apresentam maior complexidade na operação de codificação do que na operação de decodificação. No entanto, a codificação Wyner-Ziv permite que se construa um codificador de vídeo onde a codificação é menos complexa, ao custo de um decodificador mais complexo. Neste trabalho é proposto um sistema de comunicação de vídeo onde o transmissor utiliza um codificador Wyner-Ziv (de complexidade reversa), enquanto o receptor utiliza um decodificador tradicional, de forma que a complexidade seja minimizada em ambos os terminais. Para que este sistema funcione, é necessário inserir um transcodificador na rede para converter a sequência de vídeo. É apresentado um transcodificador eficiente, que recebe uma sequência codificada com um codificador Wyner-Ziv simples e transcodifica para o padrão H.263. A abordagem utilizada diminui a complexidade do sistema ao re-utilizar a estimação de movimento realizada na decodificação Wyner-Ziv, entre outras coisas.

Foi implementado um codificador Wyner-Ziv no domínio dos *pixels* para o desenvolvimento do transcodificador. Além de reutilizar os vetores de movimento calculados na decodificação Wyner-Ziv, o transcodificador também apresenta várias opções, podendo alterar o GOP da sequência transcodificada e refinar os vetores de movimento. Foram realizados testes extensivos para avaliar o transcodificador proposto e seus modos opcionais, utilizando sequências de vídeo populares como *Foreman, Salesman, CarPhone* e *Coastguard*.

ABSTRACT

In mobile to mobile video communications, both the transmitting and receiving ends may not have the necessary computing power to perform complex video compression and decompression tasks. Traditional video codecs tipycally have highly complex encoders and less complex decoders. However, Wyner-Ziv coding allows for a low complexity encoder at the price of a more complex decoder. It is proposed a video communication system where the transmitter uses a Wyner-Ziv (reverse complexity) encoder, while the receiver uses a traditional decoder, hence minimizing complexity at both ends. For that to work it becomes necessary to insert a transcoder in the network to convert the video stream. It is presented an efficient transcoder from a simple Wyner-Ziv approach to the H.263 standard. This approach saves a large amount of computation by reusing the motion estimation performed at the Wyner-Ziv decoder stage, among other things.

A pixel-domain Wyner-Ziv codec was implemented for the transcoder. Along with reusing the motion estimation done in the Wyner-Ziv decoding process, the transcoder also allows one to change the GOP length of the transcoded sequence and to refine the motions vectors. Extensive tests were carried to evaluate the proposed transcoder performance using popular video sequences such as Foreman, Salesman, Carphone and Coastguard.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS, ABREVIAÇÕES E ACRÔNIMOS

Abreviações, Acrônimos e Siglas

CD	*Compact Disc*
CODEC	*Codificador e Decodificador*
CRC	*Cyclic Redundancy Check*
DCT	*Discrete Cosine Transform*
DSC	*Distributed Source Coding*
DVC	*Distributed Video Coding*
DVD	*Digital Versatile Disc*
GOP	*Group of Pictures*
H.263	Padrão internacional de compressão de vídeo
H.264	Padrão internacional de compressão de vídeo
HD-DVD	*High-Definition Digital Versatile Disc*
IDCT	*Inverse Discrete Cosine Transform*
ITU	*International Telecommunication Union*
LDPC	*Low Density Parity Check Codes*
LDPCA	*Low Density Parity Check Accumulate Codes*
MPEG	*Motion Picture Experts Group*
PAL	*Phase Alternating Line*
PRISM	*Power Efficient, Robust, High Compression, Syndrome Based Multimedia Coding*
PSNR	*Peak Signal to Noise Ratio*
QCIF	*Quarter Common Intermediate Format*
RCPT	*Rate Compatible Punctured Turbo Codes*
SAD	*Sum of Absolute Differences*
SNR	*Signal to Noise Ratio*
VLC	*Variable Length Codes*

1 INTRODUÇÃO

1.1 INTRODUÇÃO

Com a evolução da tecnologia digital, os últimos anos mostraram uma tendência para a digitalização de mídias. Vantagens como facilidade de transmissão, fidelidade de cópia e melhores relações sinal/ruído são algumas das razões para essa transição. Uma das primeiras substituições desse gênero foi a transição do disco de vinil para o CD (*compact disc*). Embora alguns aficcionados prefiram seus discos de vinil até hoje, é inegável que o CD sucedeu o disco de vinil no mercado fonográfico.

Com a mídia de vídeo não foi diferente. A especificação para armazenar um vídeo digital em um CD foi criada em 1993 pela Sony, Philips, Matsushita e JVC. Em 1995, foi lançado o DVD (*digital versatile disc*), mídia óptica similar ao CD, usado inicialmente para reprodução de vídeos em formato digital. Atualmente, os formatos HD-DVD (*High-Definition DVD*) e Blu-Ray disputam o mercado como sucessores do DVD.

Um vídeo digital é uma sequência de imagens (denominados quadros ou *frames*) representando cenas em movimento. Uma cena de vídeo natural é espacial e temporalmente contínua. A digitalização de um vídeo requer a amostragem temporal, que divide o vídeo em imagens estáticas (os quadros), e a amostragem espacial (que divide cada imagem em pontos, denominados *picture elements* ou simplesmente *pixels*). A digitalização é exemplificada na Figura 1.1. Cada *pixel* é representado como um número ou conjunto de números que descreve o brilho (luminância) e a cor (crominância) dessa amostra. A quantidade de *pixels* em cada imagem define a resolução espacial do vídeo, enquanto a quantidade de quadros por segundo define a resolução temporal.

A amostragem típica para um DVD em formato PAL (*Phase Alternating Line*) é de 720×576 *pixels* para cada imagem, e 25 quadros por segundo. Nessas condições um filme de 2 horas ocuparia $1.791.590.400.000$ bits, ou $223,95$ *gigabytes* (cada *gigabyte*, ou GB, é considerado aqui como $1.000.000.000$ bytes). Como a mídia de DVD suporta, no máximo, $17,08$ GB, seriam necessários 14 mídias para armazenar um vídeo de 2 horas nesse formato, sem qualquer tipo de

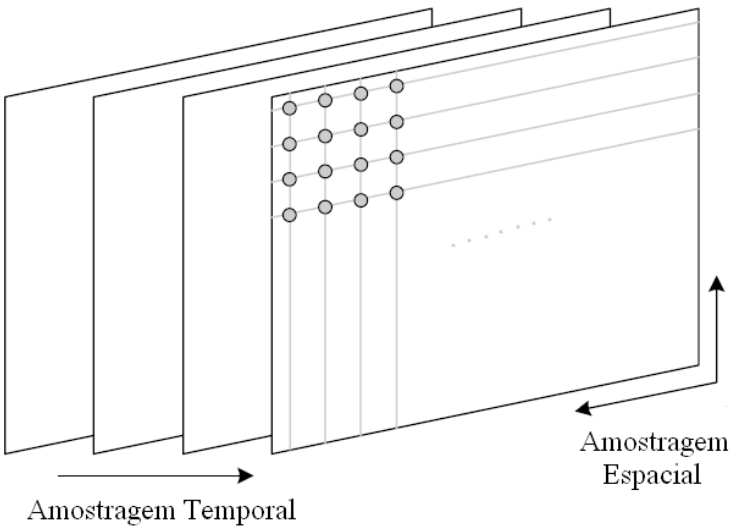


Figura 1.1: Amostragem espacial e temporal de uma sequência de vídeo.

compressão.

Essa desvantagem eliminaria o uso prático de vídeo digital. Para utilizar vídeos em formato digital, foi necessário utilizar técnicas de compressão de sinais, o que significa representar uma informação em um menor número de bits. O processo inverso da compressão, ou seja, a descompressão, pode resultar em uma imagem idêntica à original, sendo a compressão, nesse caso, sem perdas (*lossless*), ou em uma imagem semelhante à original, sendo a compressão com perdas (*lossy*). O primeiro tipo de compressão citado tem pouca utilização na área de vídeo, já o segundo processo é utilizado por quase todos os padrões de compressão de vídeo, como H.26x [1, 2] e MPEG-x [3, 4], por conseguir taxas de compressão muito maiores.

Na compressão de um vídeo, explora-se a correlação espacial (a semelhança entre pixels vizinhos em cada imagem) e a correlação temporal (a semelhança entre quadros adjacentes) a fim de se obter um melhor desempenho de compressão, ou seja, a melhor relação taxa/distorção. Existem diversos padrões para codificação de vídeo. Dentre eles, destacam-se o MPEG-2 [3], o H.263 [1] e o H.264/AVC [2]. Embora tenham desempenhos diferentes, todos tem uma estrutura muito similar: utilizam transformadas como a transformada de cossenos discreta (DCT) para explorar a redundância espacial, técnicas de estimação e compensação de movimento para explorar a redundância temporal e técnicas de codificação de entropia para codificar os resíduos da melhor maneira possível. Todos esses procedimentos são aplicados no codificador, restando

ao decodificador apenas a operação inversa de cada um desses procedimentos. Ao conjunto codificador e decodificador é dado o nome de "codec". De maneira geral, o codificador é, tipicamente, 5 a 10 vezes mais complexo que o decodificador [5, 6, 7].

À medida que o uso e transmissão de vídeo digital se expandiu, vários outros dispositivos além dos dispositivos específicos para isso passaram a incluir entre suas funcionalidades a gravação e transmissão de vídeo digital, como, por exemplo, telefones celulares, máquinas fotográficas digitais, palm-tops, entre outros. Estes dispositivos compartilham algumas características: a restrição de consumo de energia, pois todos funcionam à bateria, e a limitação de recursos computacionais, como memória escassa e capacidade de processamento. Os codificadores de vídeo tradicionais, como o H.263 e o H.264/AVC, não estão preparados para lidar com as restrições impostas por estes dispositivos, pois sua melhor eficiência é obtida através de algoritmos caracterizados por buscas exaustivas. Assim, quando utilizados em dispositivos móveis, estes codificadores trabalham em modos mais simples, perdendo em desempenho.

Um novo paradigma de codificação de vídeo vem recebendo cada vez mais atenção da comunidade: a Codificação de Vídeo Distribuída [8] (*Distributed Video Coding*, ou DVC). Esse paradigma é baseado nos teoremas de Slepian-Wolf [9] e Wyner-Ziv [10] que permitem que se explorem as semelhanças do vídeo no decodificador, resultando em codificadores menos complexos, porém decodificadores mais complexos, eventualmente podendo atingir o mesmo desempenho em relação taxa/distorção de um codec de vídeo tradicional, como o H.264/AVC, que explora as estatísticas do vídeo no codificador. Codecs de vídeo com perdas baseados neste paradigma são comumente conhecidos como codecs Wyner-Ziv. Os pesquisadores já perceberam a potencialidade de se construir, para dispostivos móveis, um codec de vídeo baseado nestas técnicas, já que possibilitam a construção de codificadores menos complexos [11, 12, 13].

Embora um codec Wyner-Ziv seja apropriado para a codificação (e a subsequente transmissão) de um vídeo em um dispositivo móvel, esse tipo de codec é totalmente inapropriado para a decodificação desse vídeo no mesmo dispositivo, devido à alta complexidade de seu decodificador. Se considerarmos um cenário de comunicação de vídeo entre dois dispositivos móveis, de forma que a codificação e a decodificação tenha que ser feita em um ambiente com restrições de recursos, nem um codec Wyner-Ziv, nem um codec tradicional, se mostram apropriados para o problema.

É nesse contexto que este trabalho se insere, propondo um transcodificador de vídeo de uma sequência Wyner-Ziv para um codec tradicional, como mostra a Figura 1.2.

Figura 1.2: Cenário do Transcodificador.

Nesse cenário, o primeiro dispositivo móvel captura e codifica a cena utilizando um codificador Wyner-Ziv. Este dispositivo transmite, então, os dados para um servidor, que irá atuar como um transcodificador. Este servidor, por sua vez, transcodifica os dados codificados com o codificador Wyner-Ziv para outro formato de vídeo tradicional, mais apropriado para a decodificação em ambientes com recursos escassos. O servidor, então, envia os dados transcodificados para o outro dispositivo móvel.

A tarefa executada pelo servidor da Figura 1.2 poderia ser realizada utilizando-se um decodificador Wyner-Ziv em cascata com um codificador de vídeo tradicional, formando o que se chama de “transcodificador trivial”. No entanto, este transcodificador seria deveras complexo, o que dificultaria o seu uso prático.

O propósito de todo transcodificador é apresentar alguma melhoria em relação ao transcodificador trivial, seja uma menor complexidade ou uma melhor qualidade. Neste trabalho, procura-se diminuir a complexidade do transcodificador, uma vez que a grande complexidade do transcodificador trivial é o maior empecilho para seu uso prático.

1.2 COMPLEXIDADE

Neste trabalho, o termo complexidade computacional se refere à quantidade de operações que um algoritmo executa para finalizar seu trabalho. Um maior número de operações indica uma maior complexidade computacional e, portanto, uma maior alocação de recursos para a execução

do algoritmo.

A análise de um algoritmo significa predizer os recursos requeridos na execução deste algoritmo [14]. Às vezes, recursos como memória e portas lógicas são o objetivo principal, porém na maioria das vezes o que se quer medir é o tempo de execução do algoritmo.

O tempo de execução de um algoritmo particular é o número de operações primitivas, ou "passos", executados. A definição da operação primitiva pode variar de acordo com o algoritmo que se quer analisar. Para um algoritmo de busca por vetores de movimento em dois quadros de uma sequência de vídeo, por exemplo, um passo pode ser definido como o teste de um candidato particular. O algoritmo que analisar menos candidatos será considerado menos complexo e será executado em um menor tempo. Como outro exemplo, para um algoritmo de cálculo de transformada de blocos, poder-se-ia contar o número de operações aritméticas que se fazem necessárias para computar a saída de cada bloco.

Embora uma mesma operação necessite de diferentes tipos de recursos dependendo da arquitetura do computador onde o algoritmo é executado, essa arquitetura não é considerada aqui. Da mesma forma, diferentes implementações de um mesmo algoritmo podem levar à diferentes complexidades - um algoritmo pode ser mais otimizado do que outro, por exemplo. A análise da complexidade levando em conta todos esses fatores está fora do escopo do trabalho, que utiliza o termo complexidade computacional apenas como a quantidade de passos que o algoritmo deve executar, desconsiderando sua implementação ou execução em um processador específico.

1.3 ORGANIZAÇÃO

O presente trabalho é organizado da seguinte maneira: o capítulo 2 contém a revisão bibliográfica sobre os teoremas de Slepian-Wolf e Wyner-Ziv, além de uma visão geral da aplicação destes teoremas na codificação de vídeo. O capítulo 3 é uma revisão do padrão de codificação H.263, usado neste trabalho como codificador de vídeo tradicional. O capítulo 4 detalha o funcionamento do codec Wyner-Ziv implementado, uma vez que ainda não existe um padrão que utilize este paradigma.

O capítulo 5 detalha o transcodificador proposto. No capítulo 6 o transcodificador é avaliado,

mostrando os testes realizados. O capítulo 7 apresenta as conclusões e propostas para trabalhos futuros.

2 CODIFICAÇÃO DISTRIBUÍDA DE FONTE

2.1 INTRODUÇÃO

A codificação distribuída de fonte (*Distributed Source Coding* ou DSC) é baseada nos trabalhos de David Slepian e Jack Keil Wolf [9], para o caso sem perdas, e Aaron Wyner e Jacob Ziv [10], para o caso com perdas, na década de 70.

O resultado principal da teoria de DSC prova que é possível codificar separadamente múltiplas fontes que sejam estatisticamente dependentes e realizar a decodificação conjunta dessas fontes, atingindo o mesmo desempenho para o caso em que as fontes são codificadas e decodificadas em conjunto. A teoria de DSC é muito próxima das teorias de codificação de canal, sendo muitas vezes estudadas em assuntos relacionados [15].

Embora o embasamento matemático exista há 30 anos, apenas recentemente essas ferramentas foram reunidas na área de codificação de vídeo, sob o codinome de *Codificação Distribuída de Vídeo* (*Distributed Video Coding* - DVC) [8]. A consequência principal do uso de codificação distribuída para codificação de vídeo é que ela permite a reversão da complexidade do codificador de vídeo, ou seja, transferir parte da complexidade do codificador para o decodificador, teoricamente sem perda de desempenho com relação à técnicas tradicionais.

2.2 O TEOREMA DE SLEPIAN-WOLF

A entropia de uma fonte diz respeito à quantidade de informação da mesma [16]. A definição matemática de entropia é:

$$H\left(X\right) = -\sum_{x \in \chi} p\left(x\right) \cdot log_2 p\left(x\right) . \tag{2.1}$$

onde x representa cada termo da fonte X e $p\left(x\right)$ representa sua probabilidade. Desde 1948, por meio do trabalho de Shannon [17], é conhecido que uma fonte X pode ser codificada (ou

comprimida) a uma taxa R_X não menor que a entropia de X, $H(X)$, para que a decodificação perfeita (sem perdas) seja possível. Isso é mostrado na Figura 2.1. A codificação de duas fontes X e Y separadamente é a expansão trivial da solução anterior e pode ser vista na Figura 2.2.

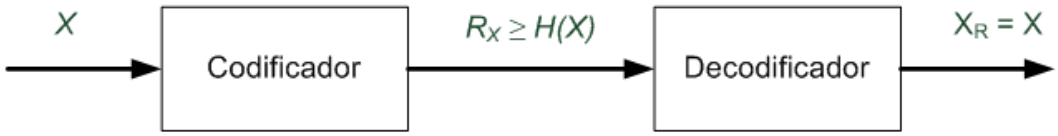


Figura 2.1: Codificação de uma fonte X. R_X denota a taxa na qual a fonte X será codificada e X_R representa a informação decodificada.

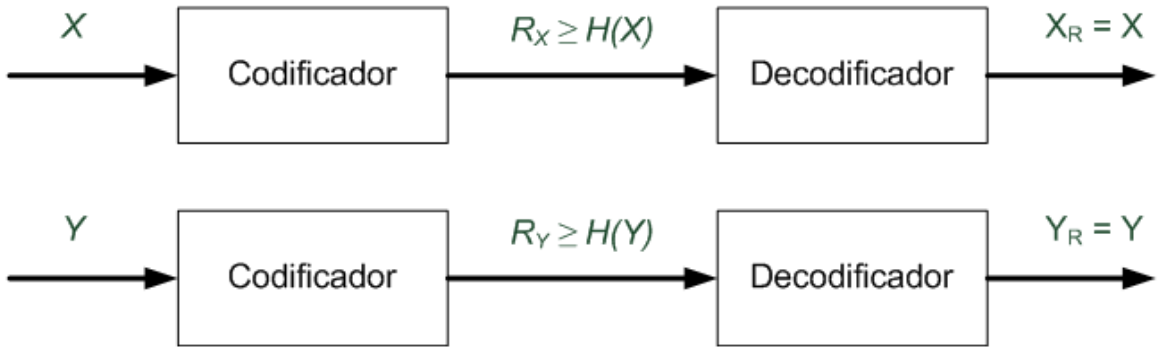


Figura 2.2: Codificação de duas fontes X e Y separadamente.

Ainda pelo trabalho de Shannon, sabemos como codificar duas fontes X, Y por um mesmo codificador, exemplificado na Figura 2.3. A taxa, neste caso, deveria ser igual ou maior que a entropia conjunta de X e Y, cuja relação com as estatísticas de X e Y é:

$$H(X,Y) = -\sum_{x \in \chi} \sum_{y \in \gamma} p(x,y) \cdot log_2 p(x,y) \,. \tag{2.2}$$

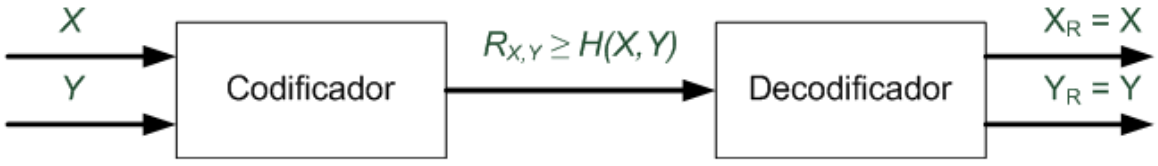


Figura 2.3: Codificação conjunta de duas fontes X e Y.

onde $H(X,Y)$ denota a entropia conjunta de X e Y e $p(x,y)$ a probabilidade conjunta de X e Y. Outra forma de se entender a codificação conjunta de duas fontes X e Y é entender esta

como sendo a codificação da fonte Y em conjunto com a codificação da fonte X condicionada ao conhecimento prévio Y. A Figura 2.4 ilustra essa situação.

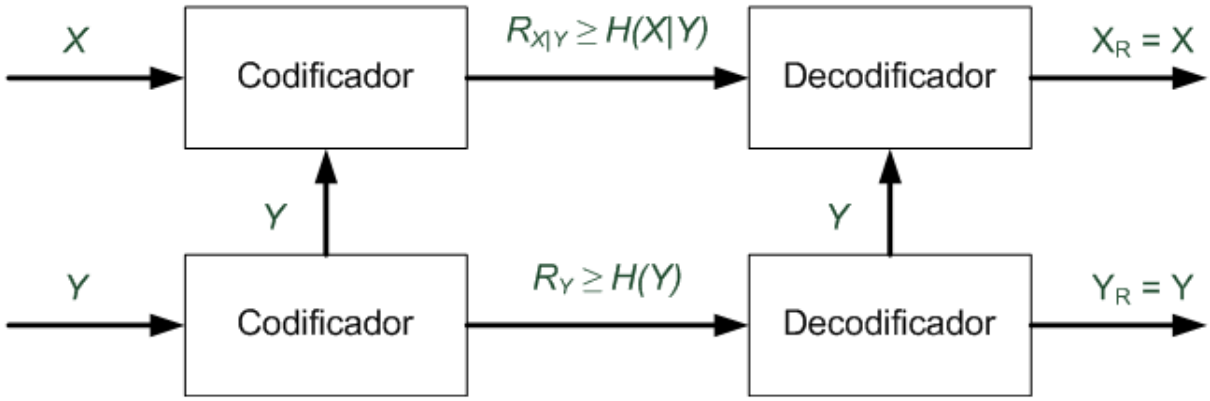


Figura 2.4: Codificação de $X|Y$.

As Figuras 2.3 e 2.4 são equivalentes. É importante notar que no esquema da Figura 2.4 a fonte Y ainda teria que ser transmitida, a uma taxa não menor que $H(Y)$. Assim, pode-se entender que a codificação de duas fontes X e Y pode ser feita por um único codificador, a uma taxa não menor que $H(X,Y)$, ou por dois codificadores, um que codifica a fonte Y a uma taxa não menor que $H(Y)$ e outro que codifica a fonte X condicionada ao conhecimento prévio Y a uma taxa não menor que $H(X|Y)$. As duas quantidades são idênticas, como mostra a Equação 2.3.

$$H(X,Y) = H(Y) + H(X|Y) \ . \tag{2.3}$$

onde $H(X|Y)$ denota a entropia de X dado o conhecimento prévio Y. O que foi provado em 1973 [9] e que se convencionou chamar de *codificação distribuída de fontes* é o esquemático da Figura 2.5, ou seja, se as estatísticas conjuntas das fontes X e Y são conhecidas, então pode-se codificar a fonte X a uma taxa $R_{X|Y} \geq H(X|Y)$ tendo acesso à informação Y somente no decodificador. Isso é formalizado no teorema de Slepian-Wolf [9, 18]:

TEOREMA 1 *Teorema de Slepian-Wolf: para o problema de codificação distribuída das fontes (X,Y) independentes e identicamente distribuídas (i.i.d.) com probabilidade $p(x,y)$ mostrado na Figura 2.6, a região possível onde a decodificação é perfeita é mostrada na Figura 2.7 e é dada por:*

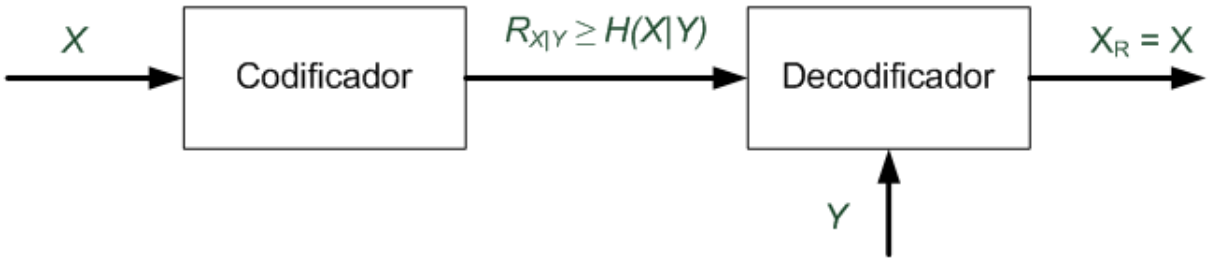


Figura 2.5: Codificação de X dado que o decodificador conhece Y.

$$
\begin{gathered}
R_1 \geq H\left(X|Y\right) \\
\cap \\
R_2 \geq H\left(Y|X\right) \\
\cap \\
R_1 + R_2 \geq H\left(X,Y\right)
\end{gathered} \tag{2.4}
$$

Na Figura 2.7, considerando $R_X = R_1$ e $R_Y = R_2$, pode-se observar que a região que apresenta probabilidade de erro nula é a região onde $R_X \geq H\left(X\right)$ e $R_Y \geq H\left(Y\right)$, resultado decorrente da teoria de Shannon. Embora as regiões $H\left(X|Y\right) \leq R_X \leq H\left(X\right) \cap R_Y \geq H\left(Y\right)$ e $H\left(Y|X\right) \leq R_Y \leq H\left(Y\right) \cap R_X \geq H\left(X\right)$ também sejam decorrentes da teoria de Shannon, o Teorema 1 mostra que a região com probabilidade de erro decrescente para sequências longas mostrada na Figura 2.7 também pode ser atingida mesmo que as fontes sejam codificadas por codificadores separados.

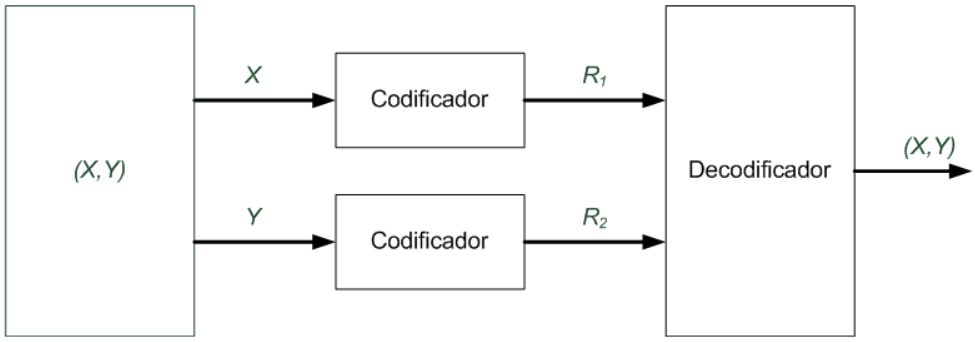


Figura 2.6: Codificação Slepian-Wolf.

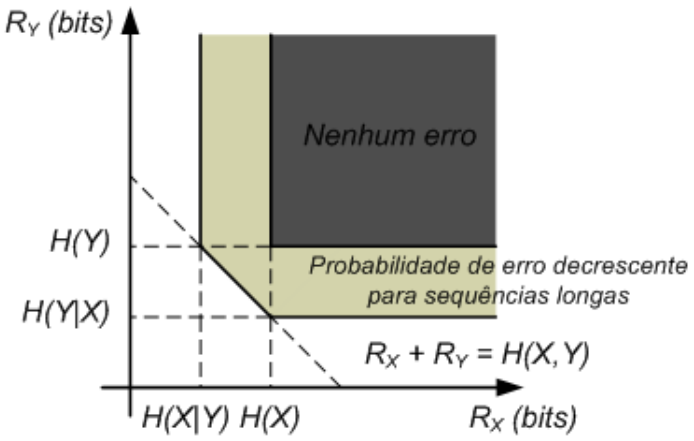


Figura 2.7: Teorema de Slepian-Wolf: região de taxa possível para a compressão distribuída de duas fontes i.i.d. estatisticamente dependentes X e Y.

2.3 O TEOREMA DE WYNER-ZIV

O trabalho de Aaron Wyner e Jacob Ziv em 1976 [10] generaliza o teorema de Slepian-Wolf para o caso de codificação com perdas. Considere duas sequências aleatórias X e Y independentes e identicamente distribuídas (i.i.d.), representando a fonte e a informação lateral, respectivamente. Os valores de X são codificados sem acesso à informação lateral Y, como mostra a Figura 2.8. O decodificador, no entanto, decodifica a informação utilizando a informação lateral Y. Dado que uma distorção $D = E\left[d\left(X, \hat{X}\right)\right]$ é permitida, a função taxa-distorção de Wyner-Ziv $R_{X|Y}^{WZ}(D)$ é o limite inferior da taxa para obter uma máxima distorção D. A função taxa-distorção $R_{X|Y}(D)$ é a taxa requerida se a informação lateral Y estivesse presente no codificador.

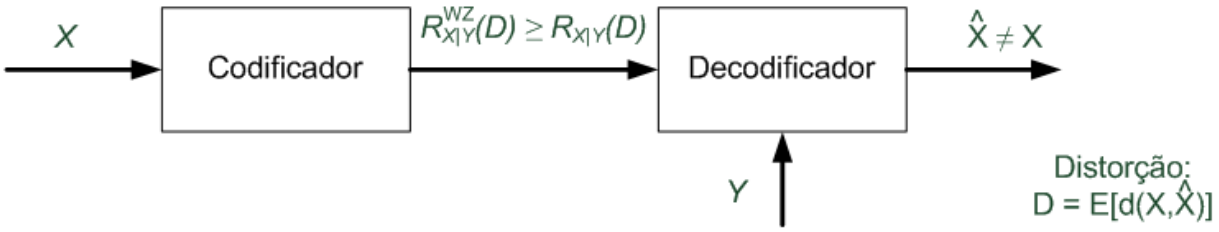


Figura 2.8: Codificação Wyner-Ziv.

O que Wyner e Ziv provaram foi que uma perda de taxa $R_{X|Y}^{WZ}(D) - R_{X|Y}(D) \geq 0$ ocorre quando o codificador não tem acesso à informação lateral. No entanto, eles também provaram que

$R_{X|Y}^{WZ}(D) - R_{X|Y}(D) = 0$ para o caso de fontes gaussianas sem memória e distorção por erro médio quadrático. A igualdade também vale para o caso de X ser composto pela soma de uma informação lateral Y com qualquer distribuição com ruído gaussiano. Para estatísticas gerais e distorção do tipo erro médio quadrático, Zamir provou em um trabalho posterior [19] que a perda em taxa é menor que 0.5 bits/amostra. Isso é formalizado no Teorema de Wyner-Ziv [10, 18]:

TEOREMA 2 *Teorema de Wyner-Ziv: Sejam duas fontes i.i.d X, Y com probabilidade $p(x,y)$, e seja $d(x^n, \hat{x}^n) = \frac{1}{n}\sum_{i=1}^{n} d(x_i, \hat{x}_i)$. A função taxa/distorção com informação lateral é dada por:*

$$R_Y(D) = \min_{p(w|x)} \min_{f} \left(I(X;W) - I(Y;W)\right) \tag{2.5}$$

onde $I(X;W)$ representa a informação mútua de X e W e a minimização é sobre todas as funções $f : Y \times W \rightarrow \hat{X}$ e funções de probabilidade condicional $p(w|x), |W| \leq |X| + 1$, tal que:

$$\sum_{x}\sum_{w}\sum_{y} p(x,y)\, p(w|x)\, d(x, f(y,w)) \leq D \tag{2.6}$$

2.4 INTERPRETAÇÕES DA CODIFICAÇÃO DISTRIBUÍDA DE FONTES

A codificação distribuída de fontes é uma área muito próxima da codificação de canal. Uma das maneiras de se aplicar o teorema de Slepian-Wolf é por meio do seguinte problema: considere duas sequências binárias i.i.d. X e Y. Se X e Y tem uma relação entre si, uma "sequência de erro" $\Delta = X \oplus Y$ (onde $\oplus$ denota a operação de "ou exclusivo") seria constituída de zeros, exceto pelas poucas partes onde X e Y seriam diferentes. Para "proteger" a sequência X destes erros pode-se aplicar um código corretor de erros sistemático (isto é, a palavra de saída é constituída da palavra de entrada concatenada com os bits de paridade) a X. Desta forma, gera-se a sequência $[X|Z]$, e realiza-se um perfuramento [20] para transmitir apenas os bits de paridade Z, como mostra a Figura 2.9. O perfuramento consiste em enviar menos bits do que os bits gerados pela matriz geradora do código corretor de erros e, portanto, resulta em um código com menor capacidade de

correção de erros. No exemplo, realiza-se o perfuramento apenas nos bits da parte sistemática, enviando todos os bits de paridade.

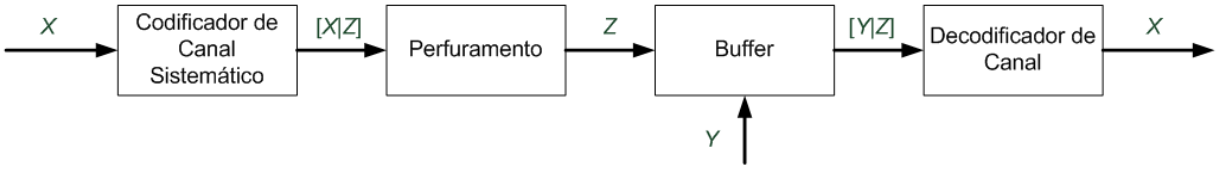


Figura 2.9: Implementação do Teorema de Slepian-Wolf.

No decodificador, a sequência de informação lateral Y seria concatenada com os bits de paridade de X, Z, formando a sequência $[Y|Z]$. Esta sequência $[Y|Z]$ pode ser pensada como sendo a sequência $[X|Z]$ adicionada de erros, podendo ser decodificada com um decodificador de canal. Se X e Y forem realmente similares, poucos erros seriam encontrados entre as sequências $[X|Z]$ e $[Y|Z]$, e a decodificação perfeita seria possível. Neste caso, poucos bits de paridade seriam necessários, e portanto a compressão seria obtida. É importante notar que este exemplo não usa o codificador de canal para corrigir erros no canal de comunicação entre o codificador e o decodificador, mas forma um "canal de correlação" que captura a dependência estatística entre X e sua informação lateral Y.

Outra forma de abordar esse problema, dado que se conheçam as estatísticas entre X e Y, é dividir o alfabeto de X em *cosets*. O codificador envia apenas o *coset* a qual X pertence, e não a própria informação X. O decodificador então usaria o *coset* recebido em conjunto com a informação lateral Y para decidir qual a informação mais provável que teria gerado o *coset* enviado pelo codificador. Ou seja, o decodificador usa Y para tirar a ambiguidade do *coset* recebido, declarando como palavra decodificada o valor de X pertencente ao *coset* mais próximo da informação lateral Y.

As duas interpretações dadas acima são equivalentes [8]: na primeira, é enviado um vetor binário $X_p = XP$, onde $G = [I|P]$ é a matriz geradora de um código de blocos linear sistemático C_p (ou seja, são enviados apenas os bits de paridade gerados por um código corretor de erros linear). Na interpretação por *cosets*, é enviada a síndrome $S = XH$, onde H é a matriz de paridade de um outro código de blocos linear C_s (ou seja, são enviados os bits referentes à síndrome gerada pela matriz de paridade). Se as matrizes que geram os bits de correção de

erros são iguais (ou seja, se a matriz P do código C_p for igual à matriz H do código C_s), os bits enviados pelos dois códigos são idênticos.

Os exemplos acima se referem a codificação distribuída de fontes sem perdas, aplicando o teorema de Slepian-Wolf. Embora alguns trabalhos foquem em implementar um codificador Wyner-Ziv através de projetos de quantizadores específicos [21, 22], a maneira mais usual de se encontrar um codificador Wyner-Ziv é mostrada na Figura 2.10, onde Q denota um quantizador.

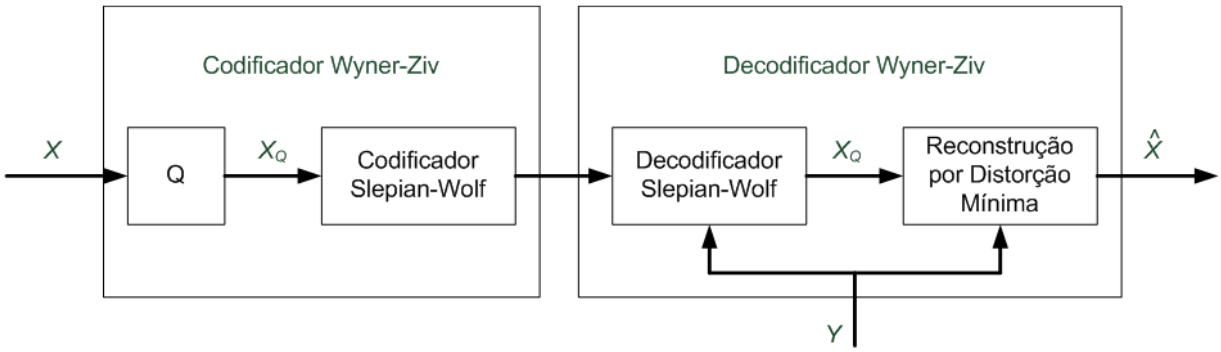


Figura 2.10: Codificação Wyner-Ziv. Q denota um quantizador.

Um codificador Wyner-Ziv pode ser obtido por meio de um quantizador seguido por um codificador Slepian-Wolf. Um decodificador Wyner-Ziv, por sua vez, pode ser obtido através de um decodificador Slepian-Wolf seguido por uma última etapa de reconstrução com critério de distorção mínima, realizada para gerar a versão final da informação decodificada. Essa etapa utiliza as duas informações Q (a saída do codificador Slepian-Wolf e, portanto, idêntica à fonte quantizada) e Y (a informação lateral utilizada pelo codificador Slepian-Wolf) para gerar uma saída $\hat{X}$ mais próxima a X.

2.5 APLICAÇÕES DE DSC PARA COMPRESSÃO DE VÍDEO

As implementações dos padrões atuais de compressão de vídeo, como o ISO MPEG [3] e as recomendações do ITU-T H.263 [1] e H.264 [2] requerem muito mais complexidade para o codificador do que para o decodificador - tipicamente, o codificador é de 5 a 10 vezes mais complexo que o decodificador [5, 6, 7]. Esta assimetria é apropriada para aplicações como *broadcasting* ou DVD‘s, onde o vídeo é codificado uma vez e decodificado várias vezes. No

entanto, algumas aplicações requerem uma menor complexidade no codificador, por exemplo sensores de vídeo sem fio para vigilância, celulares, entre outros. Nestes casos, a codificação deve ser feita na câmera, onde os recursos computacionais são escassos.

A teoria de Slepian-Wolf e Wyner-Ziv sugere que um codificador de vídeo não convencional que codifique os quadros independentemente e os decodifique condicionalmente é viável. É possível, inclusive, que um codificador desse tipo possa alcançar o mesmo desempenho de um codificador comum, que codifica os quadros condicionalmente. Diferentemente dos codificadores tradicionais, que utilizam quadros obtidos através de estimação e compensação de movimento como informação adicional para codificar o quadro atual, os codecs DVC usam a informação temporal apenas no decodificador. Assim, em muitas implementações, o processo de estimação de movimento é deslocado do codificador para o decodificador, levando consigo grande parte da complexidade do codificador, uma vez que este processo é responsável por grande parte do esforço computacional [5, 6, 7].

Alguns codecs *Wyner-Ziv* foram propostos na literatura: o codificador PRISM [23, 12, 24], os codificadores no domínio dos *pixels* [13, 25] e da transformada [26] e o codificador baseado na redução da resolução espacial [11]. Outros codificadores também foram propostos [27, 28].

2.5.1 PRISM

O codec PRISM [24] (*Power-Efficient, Robust, High-Compression, Syndrome-Based Multimedia Coding*) divide a imagem em macroblocos e aplica a cada um deles uma codificação distribuída baseada em *cosets*.

O codificador PRISM encara o problema de codificação de um macrobloco da seguinte forma: seja X o macrobloco original a ser codificado (o tamanho do macrobloco pode variar, usualmente sendo 16×16 ou 8×8) e seja Y sua melhor estimativa (obtida através de estimação de movimento). Podemos modelar a relação entre X e Y como sendo $Y = X + N$, onde N é um ruído e X e N são variáveis aleatórias independentes com distribuição Laplaciana. Inicialmente, codifica-se X em modo *intra*, obtendo uma palavra-código para X. Assumindo que o decodificador terá acesso à informação Y, o codificador procura por um codificador de canal para casar com o "ruído de correlação" N existente entre X e sua predição Y.

O codificador PRISM segue as seguintes etapas:

1. **Classificação**: Neste passo, o codificador tenta classificar o ruído de correlação entre X e seu preditor Y para que seja escolhido o codificador de canal apropriado. Esta classificação é baseada em modelagem prévia do tipo de ruído entre os macroblocos. A cada macrobloco é aplicada uma DCT bidimensional para reduzir a correlação espacial.

2. **Quantização Escalar**: Os coeficientes DCT são quantizados com um passo proporcional ao desvio padrão do ruído de correlação N.

3. **Codificação com Síndromes**: Os coeficientes quantizados são codificados com um código treliça de taxa $1/2$ e memória 7 [29].

4. **Refinamento da Quantização**: Quando os coeficientes codificados por síndrome foram quantizados, a escolha do passo foi decidida em função do ruído de correlação N. Para alcançar a qualidade desejada, esses coeficientes são quantizados novamente neste estágio.

5. ***Cyclic Redundancy Check (CRC)***: Um código de verificação de redundância é enviado para auxiliar o decodificador a encontrar qual é o melhor preditor Y para o macrobloco atual X.

6. **Codificação de Fonte Pura**: Uma parte dos coeficientes da DCT é quantizada e codificada com um código de comprimento variável (VLC).

A decodificação desses coeficientes é realizada da seguinte forma:

1. **Estimação de Movimento e Decodificação de Síndrome**: É tarefa do decodificador realizar a estimação de movimento, auxiliada pelo CRC enviado. Para decodificar a síndrome é utilizado o algoritmo de Viterbi [20].

2. **Estimação e Reconstrução**: Uma vez que a palavra código é recuperada, ela é usada como preditor para obter a melhor estimação para a fonte.

No codec PRISM, não ocorre estimação de movimento no codificador, reduzindo assim sua complexidade. No entanto, o codificador auxilia o decodificador na tarefa de estimação de

movimento, ao enviar o CRC do macrobloco. A performance do codec PRISM é superior a de um codificador *intra*, porém ainda inferior à de um codificador *inter*.

2.5.2 Os Codificadores Wyner-Ziv no Domínio Espacial e da Transformada

Girod et. al. publicou vários artigos a respeito de codificadores Wyner-Ziv [25, 26, 8]. Este codecs podem ser sub-divididos em dois grandes grupos: codificadores no domínio espacial (*pixels*) e codificadores no domínio da transformada. Uma breve explicação desses codecs é dada a seguir, enquanto uma discussão mais detalhada pode ser encontrada no Capítulo 4.

O codec no domínio dos *pixels* [25] é um codec de vídeo Wyner-Ziv simples, que combina um codificador *intra* com um decodificador *inter*. Um subconjunto de quadros, igualmente espaçados pela sequência, é codificado e decodificado utilizando um codec de vídeo comum, funcionando no modo *intra* (isto é, cada quadro é codificado sem utilizar outros quadros como referência). Estes quadros são chamados de quadros chave (*key-frames*). Os demais quadros da sequência são chamados de quadros Wyner-Ziv. O codec é mostrado na Figura 2.11.

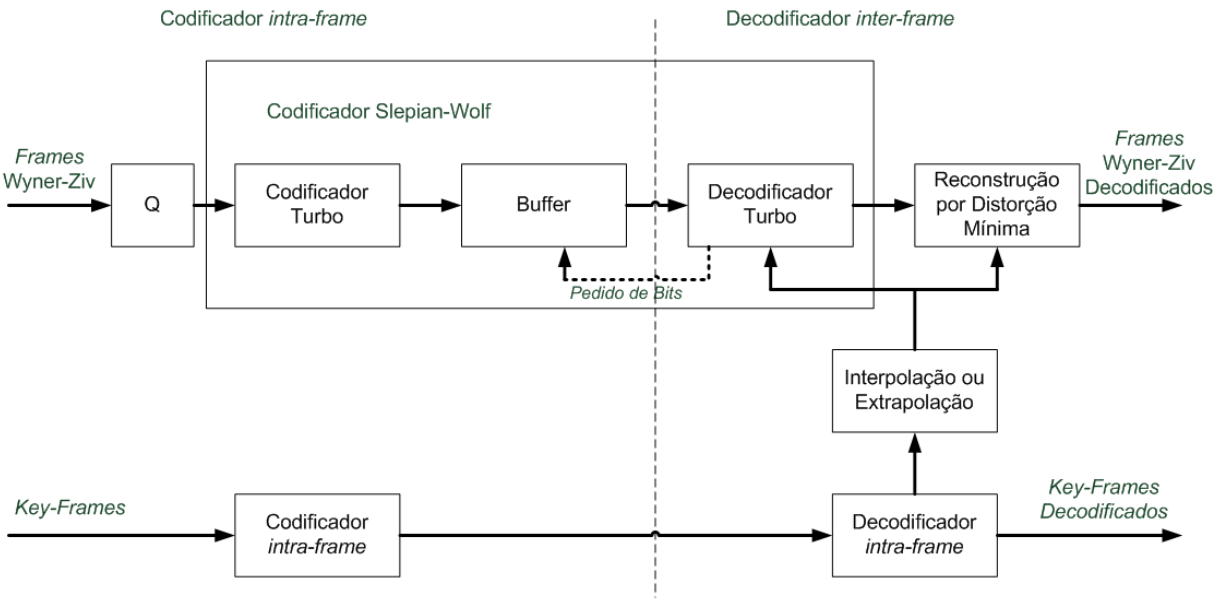


Figura 2.11: Codificador Wyner-Ziv no Domínio Espacial.

Para um quadro Wyner-Ziv X, cada *pixel* é primeiro quantizado utilizando 2^M intervalos. Os valores quantizados são agrupados em blocos suficientemente grandes e então entregues ao codificador Slepian-Wolf. Este codificador pode ser implementado utilizando *Rate-Compatible*

Punctured Turbo Code - RCPT [25, 30], *Low Density Parity Check Codes* - LDPC [31, 32] ou mesmo codificadores turbo comuns [33].

Para este mesmo quadro Wyner-Ziv, o decodificador gera uma informação lateral Y por interpolação ou extrapolação de quadros chave previamente decodificados. O decodificador turbo combina a informação lateral Y com os bits de paridade recebidos para formar a sequência de símbolos $\hat{q}$. Se o decodificador não puder decodificar a informação de maneira confiável, ele pede ao codificador, através de um canal de retorno, que envie mais bits de paridade para este bloco. Este processo é repetido até que uma probabilidade de erro de símbolo aceitável seja atingida. Utilizando a informação lateral, o decodificador muitas vezes "acerta" a informação q e, portanto, precisa de um número $k \leq M$ bits para descobrir qual dos 2^M símbolos foi transmitido, portanto comprimindo a sequência.

Após decodificar o índice de quantização $\hat{q}$, o decodificador realiza uma reconstrução por máxima verossimilhança $\hat{S} = E[S|\hat{q}, Y]$. Se a informação lateral estiver dentro do intervalo de quantização denotado pelo índice, o valor final do *pixel* irá receber um valor próximo ao da informação lateral. Se, no entanto, o valor da informação lateral estiver fora deste intervalo, o processo de reconstrução irá forçá-lo a se encontrar neste intervalo, limitando a magnitude do erro de reconstrução.

O codec Wyner-Ziv no domínio espacial é consideravelmente menos complexo do que um codec de vídeo tradicional. Não é usado nenhum processo de estimação e compensação de movimento no codificador, nem mesmo DCT e IDCT (transformada de cossenos discreta inversa) para os quadros Wyner-Ziv. Este codec é mais eficaz que um codificador simples *intra-frame*, porém menos eficaz que um codificador *inter-frame*.

O codec Wyner-Ziv no domínio da transformada é similar ao codificador no domínio dos pixels. A diferença é que, ao invés de codificar com um codificador Slepian-Wolf os *pixels* do quadro Wyner-Ziv, codifica-se coeficientes da DCT. Inicialmente, aplica-se a transformada DCT bloco a bloco na imagem. Em seguida, os coeficientes da transformada são quantizados independentemente e agrupados em sub-bandas, e só então são codificados com o codificador Slepian-Wolf. O decodificador gera a informação lateral de maneira similar ao codificador no domínio espacial, e então aplica-se a DCT bloco a bloco nesta imagem. Os coeficientes são

separados em sub-bandas, e um banco de decodificacores turbo decodifica os bits de paridade enviados pelo codificador, de maneira similar ao codificador no domínio espacial. Cada sub-banda é então reconstruída como sendo a melhor estimativa dados os símbolos reconstruídos e a informação lateral gerada.

Este codec é um pouco mais complexo do que o codificador no domínio espacial, porém atinge um melhor desempenho. O nível de complexidade é ainda muito menor que o de um codificador tradicional, em virtude de não existir o processo de estimação e compensação de movimento no codificador.

2.5.3 Sistemas Baseados em Redução da Resolução Espacial

Outro tipo de codec proposto [11] funciona da seguinte forma: alguns quadros selecionados são codificados por um codificador de vídeo regular, como o H.263, como quadros *I, P* ou *B* e são chamados de quadros chave (*key frames*). Os demais quadros que não serão usados como referência (chamados de quadros Wyner-Ziv) são dizimados por um fator múltiplo de 2 e então codificados pelo mesmo codificador de vídeo regular usado para os quadros chave. Os quadros de referência também passam pelo processo de dizimação para que possam ser usados como referência para os quadros Wyner-Ziv.

O *bitstream* gerado por essa codificação gera a camada base, composta por quadros chave em resolução completa e por quadros Wyner-Ziv em resolução reduzida, e pode ser decodificado a uma baixa complexidade ignorando a camada de realce. Para formar a camada de realce, o codificador utiliza o quadro original X e sua versão reconstruída pela camada base, porém interpolado de volta à resolução original. Denominamos o quadro reconstruído e interpolado de Z (este quadro está disponível no decodificador, de modo a não ocorrer *drift*). O codificador então envia ao decodificador o *coset* ao qual pertence os coeficientes da transformada de $X - Z$.

O decodificador decodifica primeiro a camada base. Para decodificar a camada de realce, ele utiliza um processo iterativo baseado em estimação de movimento para gerar a informação lateral Y, que será usada em conjunto com o quadro Z para decodificar o *coset* enviado pelo codificador.

Neste esquema de codificação Wyner-Ziv a redução de complexidade é atingida codificando os quadros Wyner-Ziv a uma menor resolução, diminuindo a complexidade das operações

de estimação e compensação de movimento, as maiores responsáveis pela complexidade do codificador. Embora a complexidade geral do codificador seja maior do que a complexidade de outros esquemas Wyner-Ziv, sua melhora em performance é significativa, atingindo e até ultrapassando o desempenho de codificadores tradicionais em algumas sequências de baixo movimento.

3 O PADRÃO H.263

3.1 INTRODUÇÃO

O codec H.263 foi oficialmente adotado como padrão do ITU-T em março de 1996 [1]. Em janeiro de 1998 foi lançada a especificação da segunda versão do H.263, também conhecida como H.263+ [34]. No ano 2000, foi lançada a especificação da terceira versão, o H.263++ [35]. Ele foi desenvolvido para ser um codificador de vídeo a baixas taxas, obtendo melhor desempenho que seus predecessores a um custo computacional relativamente baixo [36]. O H.263 é baseado em técnicas comuns a vários padrões de codificação de vídeo, como estimação de movimento e transformada DCT.

O menor custo computacional do codec H.263 foi a maior razão para sua escolha neste trabalho. O novo padrão, H.264/AVC [2], tem uma performance muito superior [37], porém apresenta uma complexidade muito grande para o codificador [38]. As próximas seções detalham o funcionamento do H.263, dando ênfase para as partes que serão utilizadas pelo transcodificador.

3.2 ESTRUTURA DO CODIFICADOR H.263

A estrutura deste codificador é mostrada como um diagrama de blocos na Figura 3.1.

Os quadros codificados sem dependência com os demais quadros do vídeo são chamados de quadros *intra* ou quadros do tipo *I*. A esses quadros é aplicada a DCT seguida da re-ordenação, quantização e codificação de entropia de seus coeficientes.

Para os demais quadros aplicam-se técnicas de estimação e compensação de movimento para gerar uma predição deste quadro. A estimação de movimento pode utilizar como referência apenas o quadro anterior, gerando um quadro tipo *P*, ou o quadro anterior e o posterior (desde que este já tenha sido codificado), gerando um quadro tipo *B*. Neste caso, apenas o resíduo, ou seja, a diferença entre o quadro original e sua predição, será codificada.

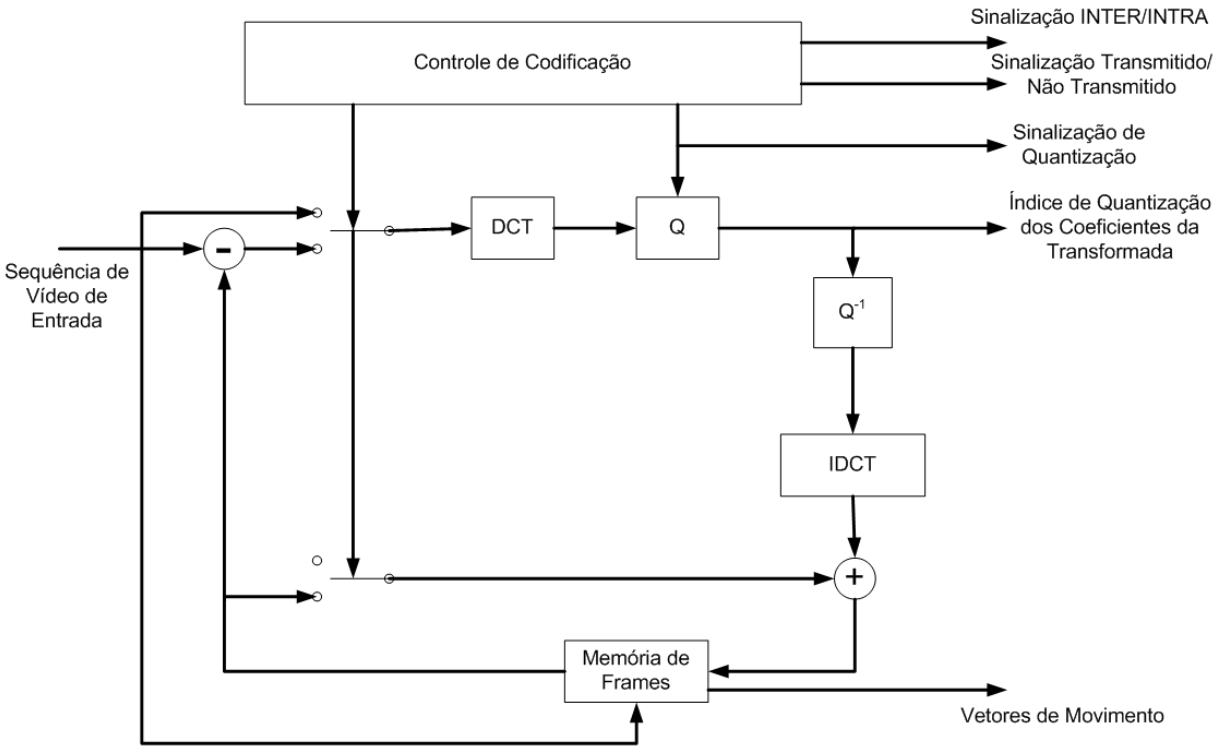


Figura 3.1: Estrutura de um codificador H.263. Q denota quantização e Q^{-1} denota sua operação inversa.

O tipo de codificação de cada quadro é definido como uma opção no codificador, e define o *Group of Pictures*, ou GOP, do vídeo.

A seguir, serão tratados cada uma das técnicas usadas pelo codificador H.263.

3.3 ESTRUTURA DOS FRAMES

O H.263 aceita as resoluções de vídeo mais utilizadas na época de sua padronização: sub-QCIF (128×96), QCIF (176×144), CIF (352×288), 4CIF (704×576) e 16-CIF(1408×1152). No H.263+, qualquer tamanho de imagem pode ser utilizado.

O H.263 *baseline* utiliza a estrutura conhecida como YC_bC_r $4 : 2 : 0$ [7], na qual os componentes de crominância C_b e C_r são amostrados a uma taxa 4 vezes menor (metade na horizontal e metade na vertical) que o componente de luminância *Y*.

A estrutura de um frame no tamanho QCIF é mostrado na Fig 3.2. A primeira divisão do

quadro é em blocos. Um grupo de blocos (*Group of Blocks* - GOB) é definido como sendo um número inteiro de linhas de macroblocos. Este número depende da resolução do quadro que será codificado. Cada macrobloco consiste em 4 quadrados 8×8 de luminância Y, um quadrado 8×8 de crominância C_b e um quadrado 8×8 de crominância C_r.

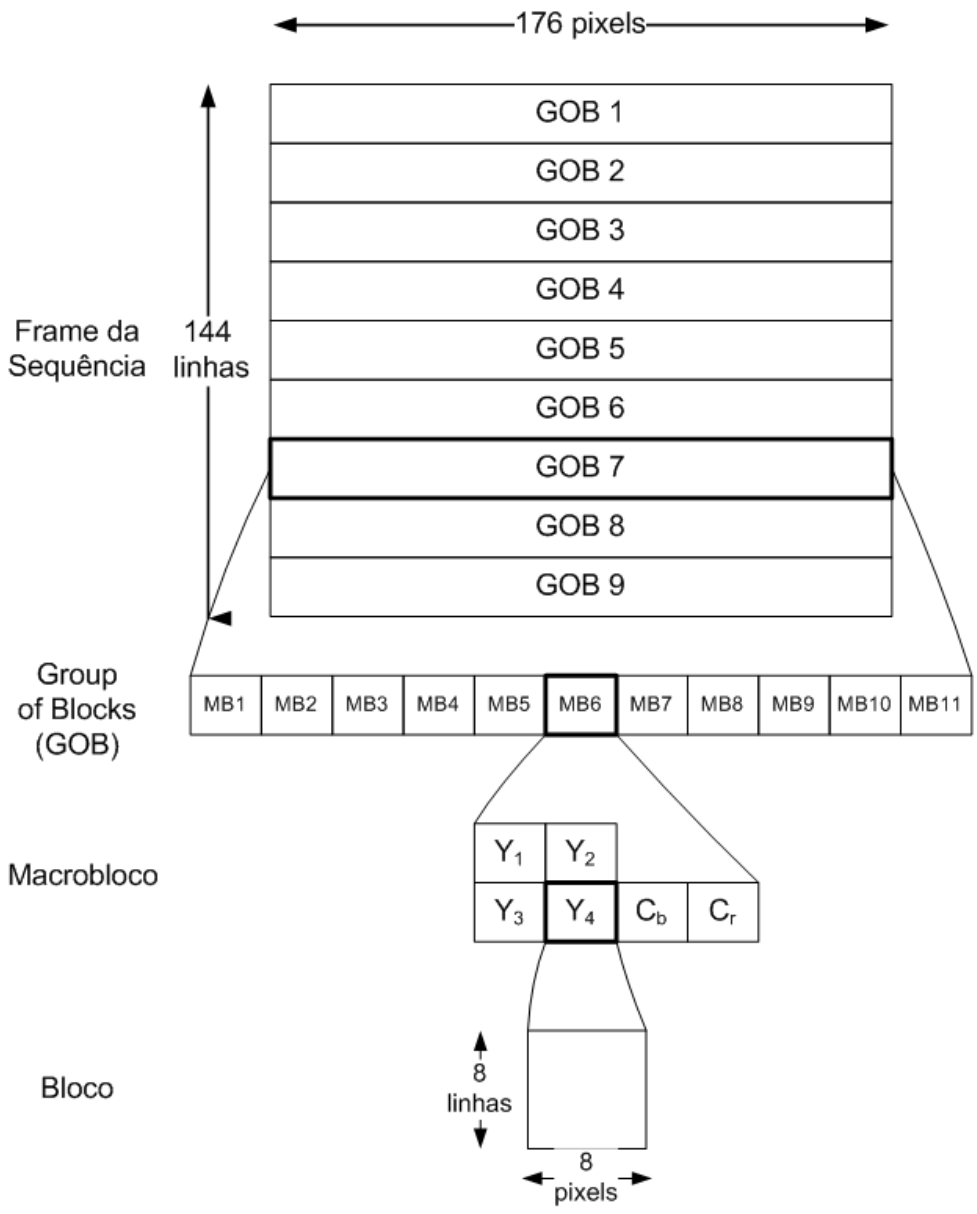


Figura 3.2: Estrutura de um frame em resolução QCIF.

3.4 ESTIMAÇÃO E COMPENSAÇÃO DE MOVIMENTO

O processo de estimação e compensação de movimento parte do princípio de que um quadro pode ser modelado como a translação de quadros anteriores [39]. O processo de estimação de movimento consiste basicamente na procura em quadros já codificados de um macrobloco que melhor aproxime o macrobloco que está sendo codificado. Embora estes macroblocos possam, teoricamente, ter formatos arbitrários [40], é comum que ele seja sempre quadrado ou retangular [38]. No H.263 *baseline* o formato do macrobloco é sempre um quadrado de 16×16. O Anexo F da recomendação especifica formato de 8×8 para a estimação de movimento.

A informação de movimento é representada por um vetor de deslocamento bidimensional chamado *vetor de movimento*. No H.263, como apresentado na seção 3.3, cada macrobloco tem tamanho de 16×16 *pixels* no componente de luminância. É uma prática comum fazer o processo de procura de vetores de movimento apenas para o quadro de luminância, enquanto os quadros de crominância são compensados usando os mesmos (ou versões escalonadas destes, para o caso dos frames de crominância possuírem menor resolução) vetores de movimento.

Devido a representação do quadro em macroblocos, o processo de estimação de movimento utiliza técnicas de casamento de blocos, onde o vetor de movimento é encontrado minimizando uma função que calcula o custo da escolha de um determinado vetor de movimento. Muitas medidas de custo foram introduzidas na literatura [41], algumas incluindo até mesmo uma ponderação entre a taxa (número de bits gasto para codificar o bloco) e a qualidade (medida entre o bloco codificado e o bloco original) que resultam da escolha de determinado vetor de movimento [42]. No entanto, para o H.263, a função de custo mais utilizada [36] é a soma das diferenças absolutas ou *SAD* (*sum of absolute differences*) definida por:

$$SAD = \sum_{k=1}^{16} \sum_{l=1}^{16} |B_{i,j}(k,l) - B_{i-u,l-v}(k,l)| \tag{3.1}$$

onde $B_{i,j}(k,l)$ representa o *pixel* (k,l) de um macrobloco cujo primeiro *pixel* se encontra na posição (i,j), e $B_{i-u,l-v}(k,l)$ representa o *pixel* (k,l) de um macrobloco candidato de um quadro de referência na posição (k,l) transladado por um vetor de movimento (u,v).

Para encontrar o vetor de movimento que produz menor erro é necessário calcular a SAD em

várias posições. O jeito mais seguro de se encontrar o melhor vetor de movimento é procurar todas as posições possíveis dentro de uma janela de procura, em um processo de busca completa (*Full-Search*). Essa técnica encontra o menor erro, porém a um custo computacional grande. Existem outras técnicas que reduzem a busca diminuindo o número de macroblocos procurados, porém estas técnicas apresentam pior desempenho com relação à busca completa [43, 44]. Algumas técnicas de procura serão apresentadas em seções posteriores.

Outra maneira de reduzir a complexidade é reduzir a janela de busca dos vetores de movimento. A redução da janela de busca assume que o movimento entre dois *frames* é pequeno, estando dentro da mesma. No H.263 *baseline* é permitido um vetor de movimento por macrobloco, com precisão de meio-pixel por meio de interpolação, como será apresentado na seção 3.4.1.

Embora a procura por vetores de movimento possa ser extendida a vários quadros desde que uma informação adicional indique a qual quadro se refere determinado vetor de movimento, no H.263 *baseline* a estimação de movimento para um quadro tipo *P* é feita *sempre* usando como referência o quadro tipo *I* ou *P* imediatamente anterior. Para quadros tipo *B*, são usados os quadros tipo *I* ou *P* imediatamente anteriores e os quadros tipo *I* ou *P* imediatamente posteriores, nunca outro quadro *B*. Neste caso, a ordem de codificação dos quadros é diferente da ordem de exibição dos mesmos, como mostra a Figura 3.3.

Para cada macrobloco em um quadro tipo *P* ou *B* existe uma sinalização para indicar se ele será codificado usando compensação de movimento ou se ele será codificado sem utilizar informação temporal. No primeiro caso diz-se que ele foi codificado no modo *inter* e, no segundo caso, em modo *intra*.

3.4.1 Estimação de Movimento com precisão de Meio-Pixel

A qualidade da estimação de movimento pode ser melhorada interpolando os quadros de referência para aumentar sua resolução, gerando um vetor de movimento mais preciso [39]. No caso do H.263, os valores de meio-pixel são encontrados através de interpolação bilinear, como mostra a Figura 3.4, onde:

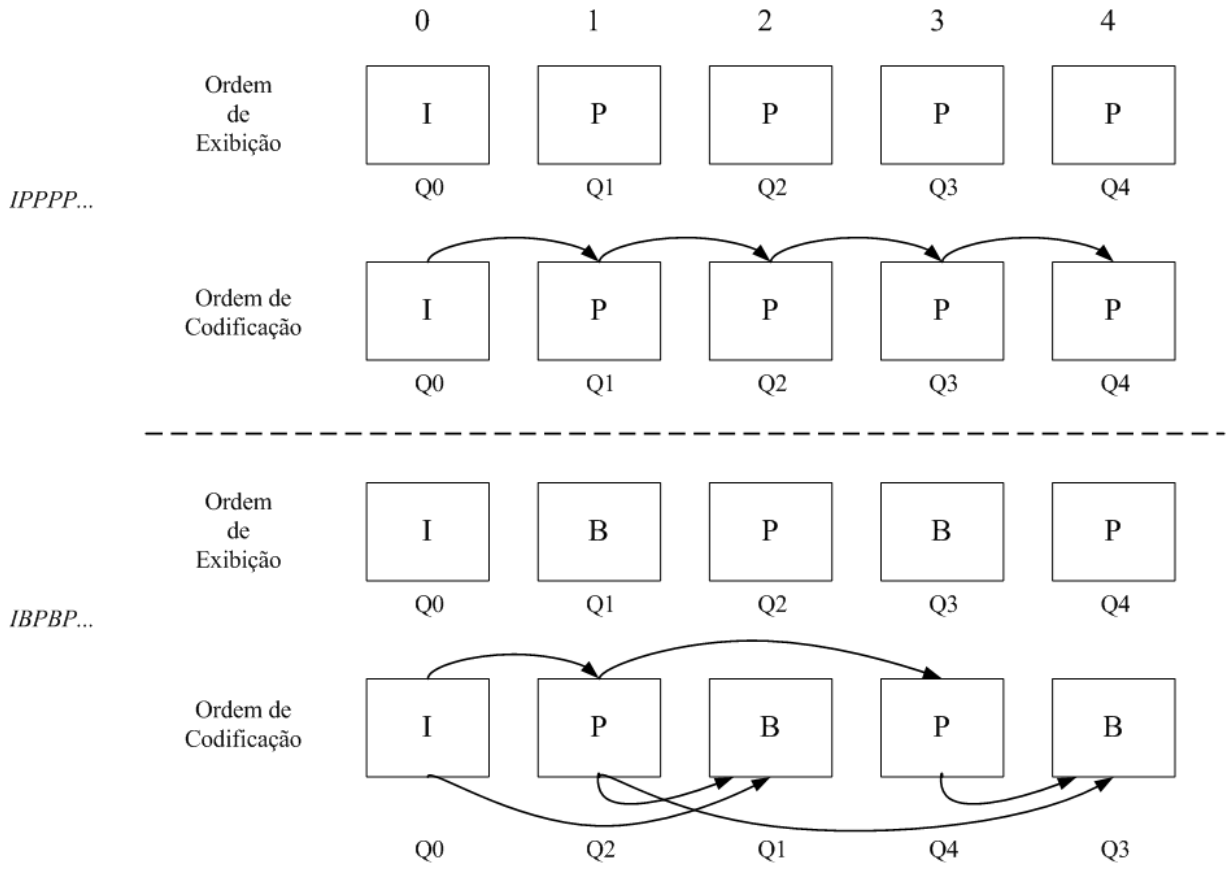


Figura 3.3: Exemplo da ordem de codificação e exibição dos quadros. As setas indicam quais quadros são usados como referência na estimação do quadro atual.

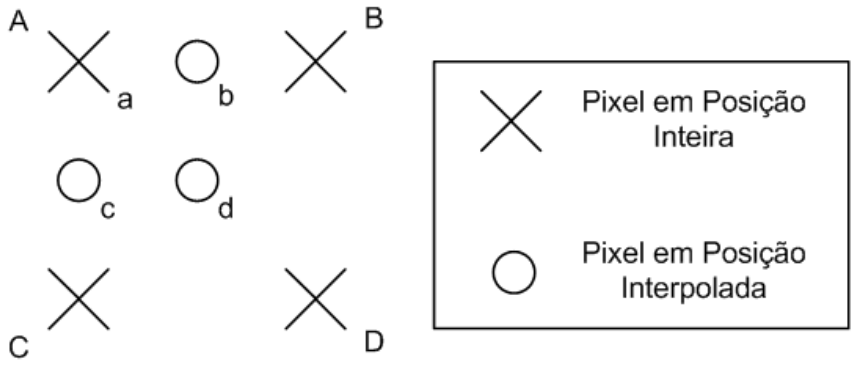


Figura 3.4: Predição de Meio-Pixel por Interpolação Bilinear.

- $a = A$
- $b = \frac{A+B+1-RCONTROL}{2}$
- $c = \frac{A+C+1-RCONTROL}{2}$
- $d = \frac{A+B+C+D+2-RCONTROL}{4}$

Na Figura 3.4 as letras maiúsculas se referem aos *pixels* antes da interpolação e as letras minúsculas se referem aos *pixels* após a interpolação. RCONTROL assume valores 0 ou 1, e pode ser usado para modificar o arredondamento do cálculo (o valor 0 força o arredondamento para baixo, enquanto o valor 1 força o arredondamento para cima).

3.4.2 Estimação de Movimento com busca em losangos

A busca em losangos (*diamond search*) [45] é um método de procura rápido, que diminui a complexidade computacional quando comparada com a busca completa. A busca em losangos tem o seguinte algoritmo:

1. Iniciar o algoritmo com o centro no vetor de movimento $(0, 0)$.
2. Calcular a SAD nos macroblocos acima, abaixo, à esquerda e à direita do centro.
3. Se a menor SAD não for o centro, modificar o centro para o ponto de menor SAD e voltar ao passo 2.
4. Parar o algoritmo, considerando o centro atual como sendo o vetor de movimento.

Em cada nova iteração são calculadas somente mais duas ou três SAD‘s, como mostra a Figura 3.5.

3.4.3 Estimação de Movimento com Busca nos Vizinhos

A busca nos vizinhos é muito similar à busca diamante, com a única diferença que a cada nova iteração são calculadas as SAD‘s dos macroblocos com conectividade 8 com o macrobloco central (ao invés da conectividade 4 procurados na busca em losangos), ou seja, os macroblocos

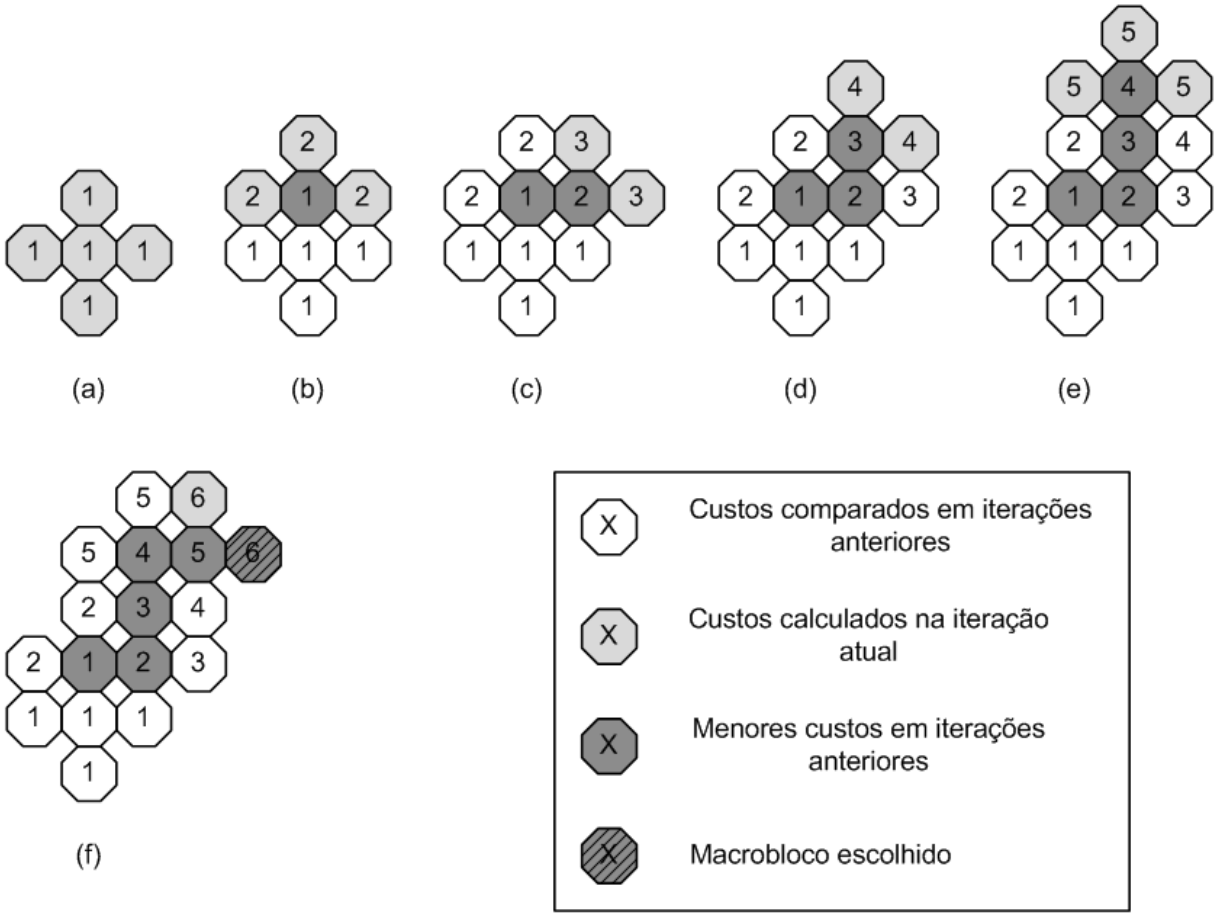


Figura 3.5: Exemplo de busca em losangos. As letras indicam os passos, enquanto os números indicam quais SADs foram calculadas nessa iteração.

acima à esquerda, acima, acima à direita, à esquerda, à direita, abaixo à esquerda, abaixo e abaixo à direita. Em cada nova iteração são calculadas somente mais três, quatro ou cinco SAD‘s, como mostra a Figura 3.6.

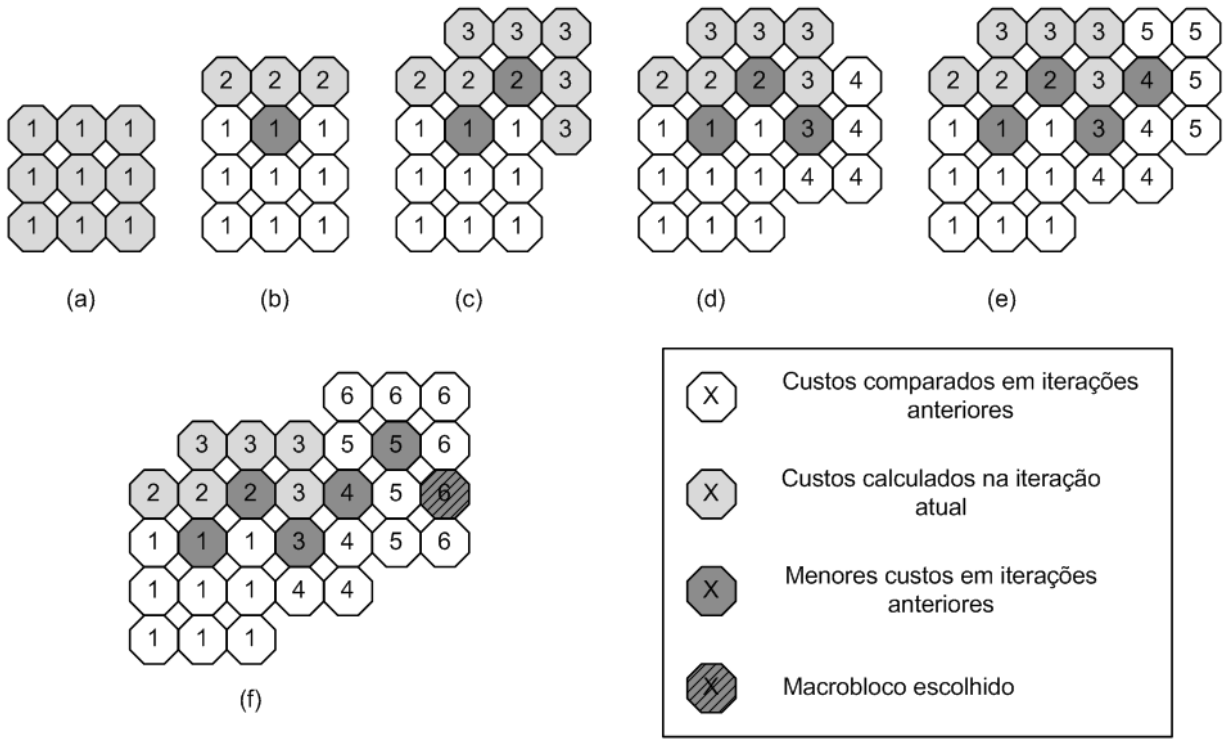


Figura 3.6: Exemplo de busca nos vizinhos. As letras indicam os passos, enquanto os números indicam quais SADs foram calculadas nessa iteração.

3.4.4 Estimação de Movimento com Busca Hexagonal/Losangos

A busca hexagonal [46] também segue a mesma idéia, com a diferença que agora procuram-se SADs mais distantes do centro, ao invés de apenas SADs vizinhas. A busca se inicia calculando a SAD no centro e em mais 6 vizinhos, formando um hexágono ao redor do centro. A cada nova iteração são calculadas somente mais três SADs, como mostra a Figura 3.7. Ao se chegar na situação onde a SAD no centro é a menor, procura-se mais quatro vizinhos (acima, à esquerda, à direita e abaixo) e escolhe-se como vetor de movimento o de menor SAD.

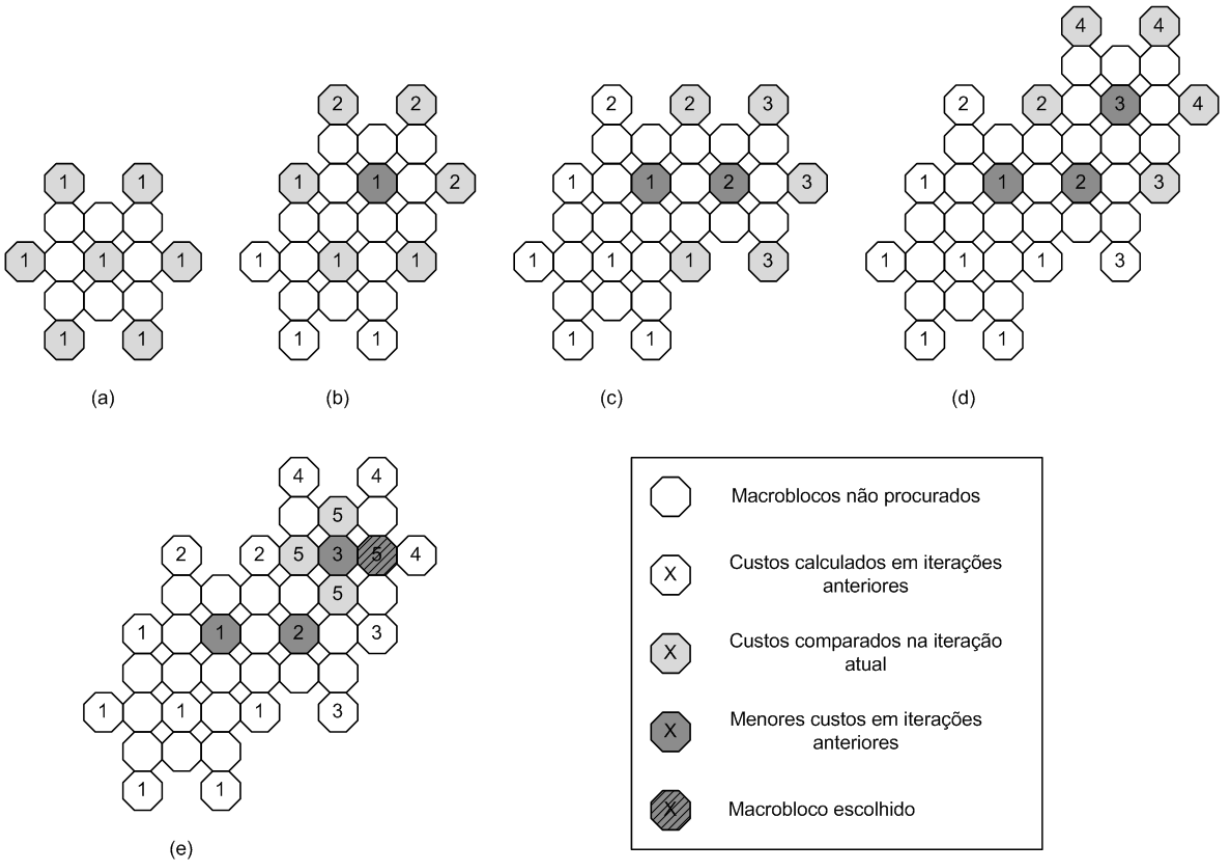


Figura 3.7: Exemplo de busca hexagonal/losangos. As letras indicam os passos, enquanto os números indicam quais SADs foram calculadas nessa iteração.

3.4.5 Estimação de Movimento com Busca Hexagonal/Vizinhos

A busca hexagonal/vizinhos [46] segue o mesmo princípio da busca hexagonal/losangos, com a única diferença que na iteração final da busca hexagonal (quando o macrobloco central tiver o menor custo) são buscados todos os vizinhos do centro. A Figura 3.8 exemplifica o processo.

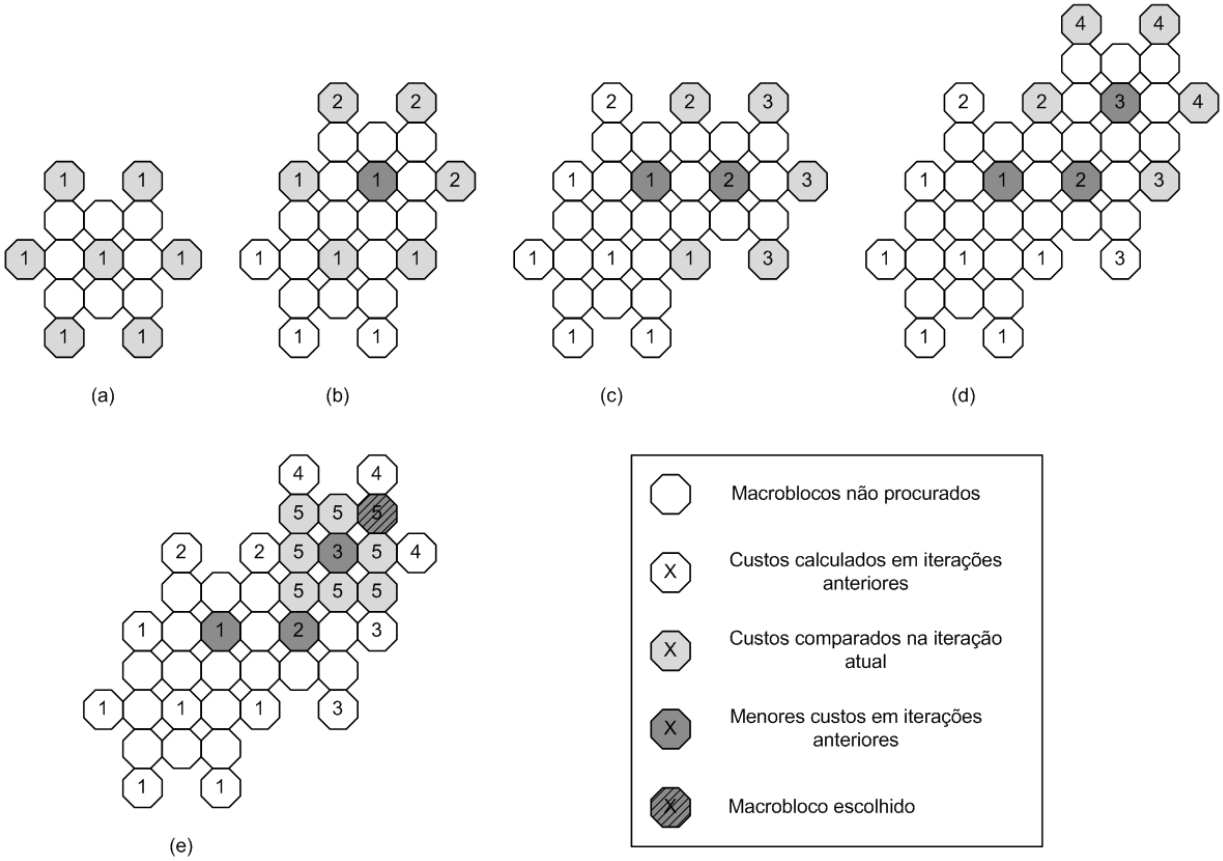


Figura 3.8: Exemplo de busca hexagonal/vizinhos. As letras indicam os passos, enquanto os números indicam quais SADs foram calculadas nessa iteração.

3.5 TRANSFORMADA

O H.263 especifica uma Transformada de Cossenos Discreta (DCT) [47] de tamanho 8×8, com o propósito de descorrelacionar os *pixels* do bloco original, compactando sua energia em um menor número de coeficientes. A DCT 8×8 é definida por:

$$C_{m,n} = \alpha(m)\,\beta(n) \sum_{i=1}^{8} \sum_{j=1}^{8} B_{i,j} \cdot cos\left(\frac{\pi\,(2i+1)\,m}{16}\right) \cdot cos\left(\frac{\pi\,(2j+1)\,n}{16}\right) \tag{3.2}$$

onde: $0 \leq m,n \leq 7$, $\alpha(0) = \beta(0) = \sqrt{\frac{1}{8}}$, e $\alpha(m) = \beta(n) = \sqrt{\frac{1}{4}}$ para $1 \leq m,n \leq 7$. $B_{i,j}$ denota o pixel (i,j) do bloco original e $C_{(m,n)}$ são os coeficientes da DCT do bloco transformado. A transformada inversa definida pelo padrão é dada por:

$$B_{i,j} = \sum_{m=1}^{8} \sum_{n=1}^{8} C_{(m,n)}\alpha(m) \cdot cos\left(\frac{\pi\,(2m+1)\,i}{16}\right)\beta(n) \cdot cos\left(\frac{\pi\,(2n+1)\,j}{16}\right) \tag{3.3}$$

onde $0 \leq i,j \leq 7$. No H.263 a transformada é implementada em apenas uma dimensão: primeiro são transformadas as linhas e, em seguida, as colunas.

3.6 QUANTIZAÇÃO

O H.263 é um codificador de vídeo com perdas, isto é, a sequência original é diferente da sequência decodificada. O H.263 controla as perdas na codificação (e, por consequência, a taxa de bits) por meio da quantização dos coeficientes da DCT de cada macrobloco.

O primeiro coeficiente de um macrobloco (o coeficiente DC) do tipo *intra* é *sempre* quantizado uniformemente com um passo de 8. Para todos os demais coeficientes de blocos *intra* e *inter*, um parâmetro de quantização é definido pelo usuário no intervalo $1...31$. Cada um dos coeficientes é então quantizado uniformemente com um passo no intervalo $2...62$ (2 vezes o parâmetro de quantização). Os quantizadores consistem em níveis de reconstrução igualmente espaçados com uma zona morta centrada em zero, como mostra a equação:

$$C^{q}_{m,n} = \frac{C_{m,n}}{Q_{m,n}} \tag{3.4}$$

onde $0 \leq m,n \leq 7$. Os valores reais resultantes são arredondados para o valor inteiro mais próximo.

3.7 CODIFICAÇÃO DE ENTROPIA

O H.263 *baseline* faz a codificação de entropia utilizando um código de comprimento variável (*variable lenght codes*, ou VLC). No Anexo E da recomendação do ITU que define o H.263 [1] é apresentado também um modo de codificação aritmética baseado em sintaxe (*syntax-based arithmetic coding mode*) para substituir o VLC. Como ambos os codificadores são codificadores sem perda, a qualidade da imagem não é modificada pelo uso do codificador de entropia aritmético, ocorrendo apenas uma redução na taxa.

Para os vetores de movimento, utiliza-se como predição a mediana dos vetores de movimento de macroblocos vizinhos já codificados (os macroblocos à esquerda, acima e acima à direita). A diferença entre o vetor de movimento e sua predição então alimenta o codificador de entropia.

Os coeficientes da DCT são re-arranjados em ordem zigue-zague em um vetor unidimensional, como mostra a Fig. 3.9. Este arranjo coloca o coeficiente DC primeiro, seguido pelos demais coeficientes na ordem de frequências mais baixas para as mais altas. Em seguida, este vetor unidimensional é codificado utilizando o VLC ou o codificador aritmético.

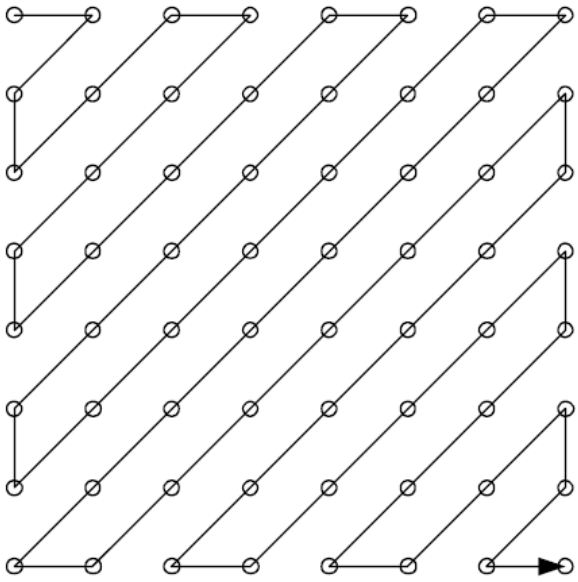


Figura 3.9: Padrão de rearranjo dos coeficientes da DCT.

3.8 CRITÉRIOS DE QUALIDADE

A codificação de um vídeo é usualmente realizada aceitando-se perdas entre o vídeo codificado e o vídeo original a fim de se obter maiores taxas de compressão. Dessa forma, faz-se necessária a questão: até que ponto pode-se aceitar a perda de qualidade entre o vídeo comprimido e o vídeo original? Como medir essa distorção?

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*, relação sinal ruído). Embora essa medida não seja perfeita, ela permite definir claramente se um determinado sinal recebido é melhor ou pior do que outro, além de permitir com precisão qual a taxa de erro esperada em cada sinal e até mesmo indicar se a demodulação é possível. A SNR é uma medida objetiva, isto é, ela pode ser medida com precisão e não varia com a repetição do experimento, e é amplamente utilizada em sistemas de comunicação via rádio, cabo ou qualquer outro.

Em sistemas de vídeo digital utiliza-se comumente uma medida chamada PSNR [7] (*peak signal to noise ratio*, relação sinal ruído de pico). A formulação da PSNR em decibéis é dada por:

$$PSNR_{dB} = 10 \cdot log_{10} \frac{(2^n - 1)^2}{MSE} \,. \tag{3.5}$$

$$MSE = \frac{1}{N} \cdot \sum_{i=1}^{N} (x\,(i) - y\,(i))^2. \tag{3.6}$$

onde MSE indica o erro médio quadrático entre a imagem comprimida (y) e a imagem original (x), e o numerador é o valor máximo que um *pixel* pode assumir nessa imagem, dependente do número de bits por *pixel* (por exemplo, se são usados $n = 8$ bits por *pixel* o valor de pico do numerador é 255).

Apesar de ser uma medida objetiva e de ser a métrica de qualidade mais utilizada no campo de vídeo digital, diferentemente da SNR em comunicações via rádio, a PSNR tem 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 que ela às vezes contradiz a percepção subjetiva de qualidade. Apesar disso, tipicamente, quanto maior a PSNR melhor é a

percepção subjetiva da imagem.

A percepção subjetiva de qualidade se refere à como uma pessoa percebe a diferença entre as imagens. Isso pode depender muito de acordo com a aplicação (se é para trabalho, ou apenas entretenimento), com a atenção que se dá à imagem, com as experiências pessoais do observador, e muitos outros fatores, fazendo com que os resultados de uma medida subjetiva não possam ser repetidos nem medidos com precisão.

Apesar dos problemas citados, em geral uma melhor PSNR significa uma melhor percepção, salvo casos onde a distorção é centralizada em certas áreas. A PSNR é ainda a medida mais utilizada de qualidade de imagem e vídeo encontrada, devido principalmente à facilidade na medição (dado que a imagem de referência esteja disponível), à possibilidade de repetição do experimento e a possibilidade de comparação com outros trabalhos. Por essa razão, o critério de qualidade utilizado neste trabalho é a PSNR.

4 O CODIFICADOR WYNER-ZIV UTILIZADO

4.1 INTRODUÇÃO

O codec de vídeo Wyner-Ziv utilizado foi proposto originalmente por Girod et. al. [31], porém foram realizadas pequenas modificações em alguns módulos. Este codec é um codificador Wyner-Ziv no domínio dos *pixels*, que apresenta baixa complexidade do lado do codificador pois não há estimação de movimento e nem mesmo transformada DCT.

Este codec foi escolhido por já ter sido publicado previamente, por um grupo reconhecido nesta área, e devido à sua baixa complexidade do lado do codificador.

4.2 ESTRUTURA DO CODIFICADOR/DECODIFICADOR

A estrutura do codec Wyner-Ziv é mostrada na Figura 4.1. Os quadros da sequência de vídeo são divididos em quadros chave (*key frames*) e quadros Wyner-Ziv. A forma como os quadros são divididos define o GOP (*group of pictures*) da sequência codificada. Um GOP de 2 indica que será codificado um quadro Wyner-Ziv para cada quadro chave (na forma *IWIW...*, onde *I* denota um quadro chave e *W* um quadro Wyner-Ziv. O codificador foi implementado para aceitar um GOP de 2 ou 3 (*IWWIWW...*), mas pode ser expandido para utilizar um GOP ainda maior. Quanto maior o GOP da sequência, maior a redução de complexidade no codificador. Deve-se notar, no entanto, que a qualidade da informação lateral no decodificador diminui quando o GOP da sequência aumenta, como será visto na seção 4.4.1, pois serão utilizados quadros chave mais distantes para gerá-la, e o desempenho do codificador piora quando se aumenta muito o número de quadros Wyner-Ziv entre dois quadros chave.

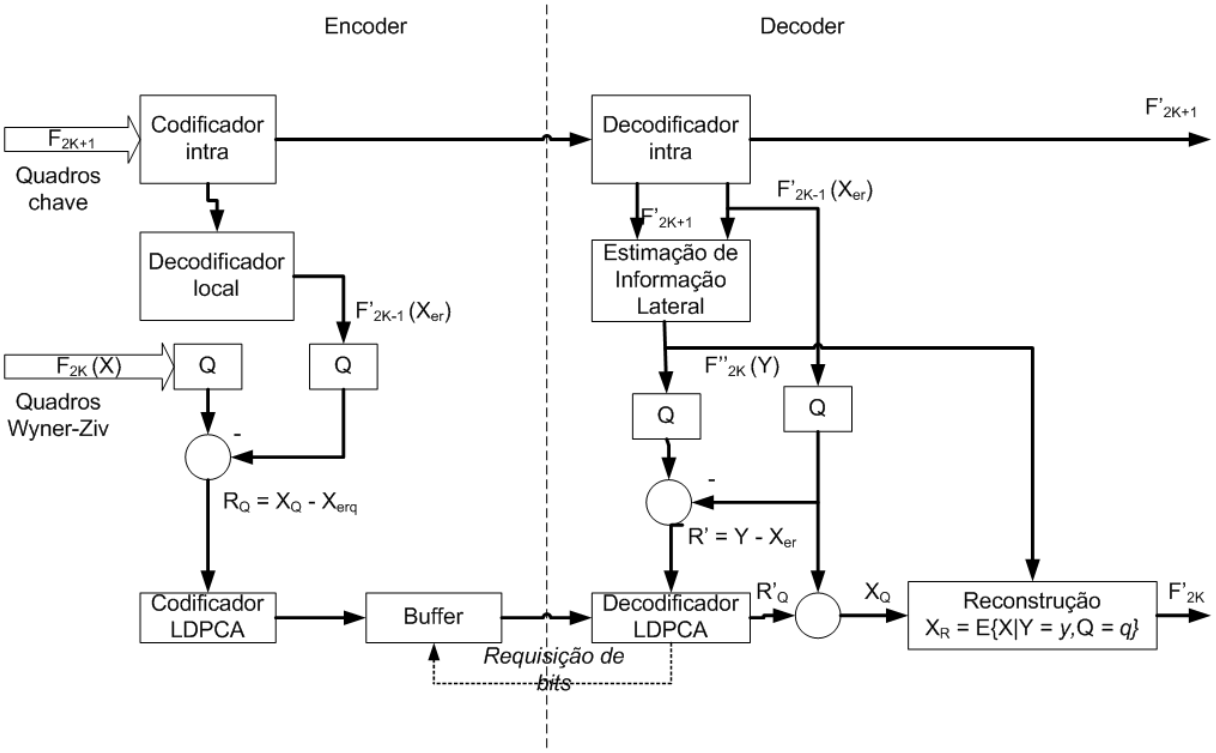


Figura 4.1: Estrutura do codificador Wyner-Ziv utilizado.

4.3 O CODIFICADOR

Os quadros chave são codificados utilizando um codificador de vídeo tradicional trabalhando no modo *intra*. O codificador utilizado foi o H.263 *baseline* [1] mostrado no capítulo 3, utilizando como codificador de entropia o codificador aritmético definido no Anexo E da especificação do H.263 [1]. A única forma de controle definida pelo usuário sobre a qualidade/taxa destes quadros é o QP (passo de quantização), que modifica o quantizador dos coeficientes da DCT. Um QP maior implica em uma quantização maior e, portanto, em um aumento da distorção. Entretanto, com passos de quantização maiores, mais coeficientes da DCT assumem o valor 0, diminuindo a taxa da sequência codificada. A Figura 4.2 mostra o resultado da codificação do primeiro quadro da sequência *Foreman* com $QP = 4$. Todos os exemplos deste capítulo serão baseados nos primeiros 3 quadros dessa sequência, a não ser quando especificado de outra forma.

Os quadros Wyner-Ziv são codificados de maneira diferente. A primeira operação é quantizar o quadro atual X e o quadro de referência do codificador X_{er}. Esse quadro de referência é tomado como sendo a reconstrução do último quadro chave codificado. Essa informação está disponível

(a) Quadro Original

(b) Quadro Codificado - PSNR = $39.4925dB$.

(c) Detalhe do Quadro Original

(d) Detalhe do Quadro Codificado

Figura 4.2: Exemplo da codificação de um quadro chave da sequência *Foreman* com $QP = 4$.

no decodificador, de forma a não ocorrer erros de *drift*. A Figura 4.3 mostra o quadro Wyner-Ziv que será codificado X, o quadro de referência do codificador X_{er}, bem como o resultado da quantização de cada um.

(a) Quadro de Referência Original (X_{er}) (b) Quadro Wyner-Ziv Original (X).

(c) Quadro de Referência Quantizado (X_{erq}) (d) Quadro Wyner-Ziv Quantizado (X_q)

Figura 4.3: Exemplo da quantização de um quadro Wyner-Ziv e de um quadro de referência.

O passo do quantizador para os quadros Wyner-Ziv varia de acordo com o QP utilizado para codificar os quadros chave. A Tabela 4.1 mostra a relação entre o QP (passo de quantização para os quadros chave) e o passo de quantização para os quadros Wyner-Ziv, chamado de $QPWZ$. Esta relação foi obtida empiricamente, testando vários $QPWZ$ para cada QP, e escolhendo o que apresentava melhor desempenho em relação taxa/distorção. Neste teste, foram utilizados os primeiros 31 quadros da sequência *Foreman*, com $QPWZ$ assumindo valores de $\{64, 48, 32, 28, 24, 20, 16, 12, 8\}$ para cada QP. Em seguida, os pares $(QP, QPWZ)$ selecionados (mostrados na Tabela 4.1) foram validados para as demais sequências de teste (*Salesman, Coastguard, Carphone* e *Akiyo*. A fórmula da quantização utilizada é mostrada em 4.3 em dois passos: o primeiro que obtém o índice de quantização q e o segundo que mostra a

Tabela 4.1: Relação entre o passo de quantização dos quadros Wyner-Ziv e o QP utilizado para os quadros-chave

QP	7	6	5	4	3
$QPWZ$	64	48	32	16	8

reconstrução deste índice.

$$q = \left\lfloor \frac{x}{QPWZ} \right\rfloor \tag{4.1}$$

$$X_q = \left(q + \frac{1}{2} \right) \cdot QPWZ \tag{4.2}$$

onde o operador $\lfloor x \rfloor$ indica o arredondamento para baixo do valor x (ou seja, o maior inteiro menor que x).

Em seguida, calculamos o resíduo $R = X_q - X_{erq}$ como a diferença entre os dois quadros quantizados, onde X_q é o quadro original quantizado com passo $QPWZ$ e X_{erq} é o quadro chave decodificado localmente e quantizado com o mesmo passo $QPWZ$. O quadro R contém o índice de quantização do resíduo de cada *pixel* do quadro atual. A estes índices deve ser associado um código binário para que possa ser usado como entrada pelo codificador de canal. Antes, porém, será apresentado como se gera a informação lateral que será utilizada para decodificar o resíduo R.

4.4 O DECODIFICADOR

A decodificação dos quadros chave é feita utilizando um decodificador H.263 *intra*. São decodificados dois quadros chave antes de se iniciar a decodificação de um quadro Wyner-Ziv, e em seguida é decodificado um quadro chave para cada quadro Wyner-Ziv, de modo que se tenha sempre uma referência de um quadro chave anterior e posterior para gerar a informação lateral de um quadro Wyner-Ziv. As três etapas da decodificação do quadro Wyner-Ziv são apresentadas a seguir: Informação Lateral, Decodificação Slepian-Wolf e Reconstrução por Máxima Verossimilhança.

4.4.1 Informação Lateral

O processo de geração da informação lateral consiste em gerar, no decodificador, uma estimativa para o quadro Wyner-Ziv enviado pelo codificador (X) para ser utilizado pelo decodificador de canal para decodificar a informação transmitida. O processo utilizado foi baseado no processo proposto por Delp et. al. [48] e utiliza os quadros chave anterior e posterior ao quadro Wyner-Ziv atual.

O método de geração da informação lateral pode ser visto na Figura 4.4. Os quadros chave reconstruídos F'_{2K} e F'_{2K+2} estão disponíveis no decodificador e já foram decodificados no momento da decodificação do quadro Wyner-Ziv atual F_{2K+1}. O processo se inicia usando o quadro anterior F'_{2K} como referência e o quadro posterior F'_{2K+2} como fonte para uma estimação de movimento com macroblocos 16×16 e busca completa em uma janela de 15 *pixels*. Após encontrar os vetores de movimento MV_F (do inglês *motion vectors*) resultantes dessa busca, utiliza-se $\frac{MV_F}{2}$ no quadro de referência F'_{2K} para gerar o quadro compensado P_F. Em seguida, utiliza-se o quadro posterior F'_{2K+2} como referência e o quadro anterior F'_{2K} como fonte para uma nova busca por vetores de movimento. Após encontrados os vetores de movimento MV_B resultantes dessa busca, utiliza-se $\frac{MV_B}{2}$ no quadro de referência F'_{2K+2} para gerar o quadro compensado P_B. A informação lateral final Y é tomada como sendo a média aritmética entre P_F e P_B.

A Figura 4.5 mostra os resultados final e intermediários do processo de geração da informação lateral, bem como suas avaliações se comparadas ao quadro original. As medidas de PSNR são comparadas com o quadro Wyner-Ziv original.

Este processo pode ser expandido de forma a gerar mais quadros entre dois quadros chave, combinando técnicas propostas anteriormente [48, 25]. O processo para gerar 2 quadros de informação lateral entre 2 quadros chave F'_{3K} e F'_{3K+3} pode ser visto na Figura 4.6. Inicialmente calculam-se os vetores de movimento MV_F e MV_B. Em seguida, utilizam-se os vetores de movimento $\frac{MV_F}{3}$ sobre o quadro de referência F'_{3K} para gerar o quadro P_{F1} e os vetores de movimento $\frac{2 \times MV_B}{3}$ sobre o quadro de referência F'_{3K+3} para gerar o quadro P_{B1}. A informação lateral para o quadro F_{3K+1} é a média aritmética entre os quadros P_{F1} e P_{B1}. De forma similar, utiliza-se $\frac{2 \times MV_F}{3}$ sobre o quadro de referência F'_{3K} para gerar o quadro P_{F2} e $\frac{MV_B}{3}$ sobre o quadro

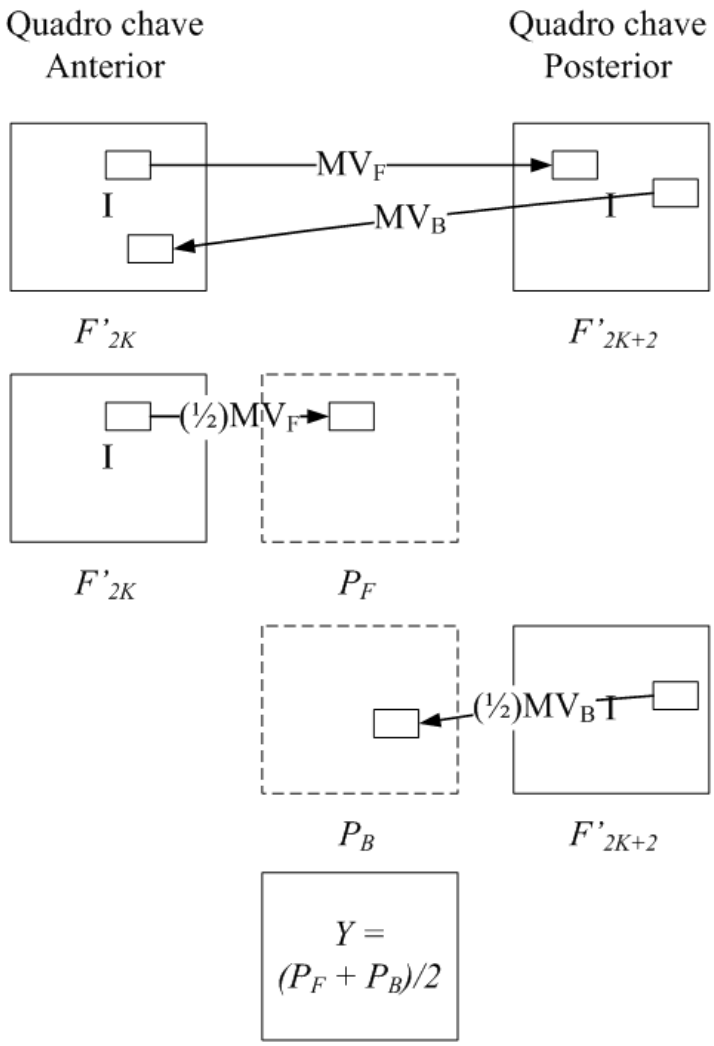


Figura 4.4: Processo de geração da informação lateral. Os vetores de movimento MV_F e MV_B são calculados utilizando os quadros chave. Em seguida, utiliza-se $\frac{MV_F}{2}$ para gerar o quadro P_F e $\frac{MV_B}{2}$ para gerar o quadro P_B. A informação lateral final é tomada como $Y = \frac{P_F+P_B}{2}$.

(a) Quadro chave anterior (F'_{2K}), PSNR = 28.4853 dB.

(b) Quadro chave posterior (F'_{2K+2}), PSNR = 27.8781 dB .

(c) Etapa intermediária (P_F), PSNR = 31.0644 dB

(d) Etapa intermediária (P_B), PSNR = 31.4093 dB.

(e) Quadro Wyner-Ziv Original (F_{2K+1})

(f) Informação Lateral (Y), PSNR = 33.2856 dB.

Figura 4.5: Processo de geração da informação lateral. A informação lateral final é mais próxima do quadro original do que o quadro chave anterior, posterior ou os quadros gerados nas etapas da geração da informação lateral.

de referência F'_{3K+3} para gerar o quadro P_{B2}. A informação lateral para o quadro F_{3K+2} é a média aritmética entre os quadros P_{F2} e P_{B2}.

A Figura 4.7 mostra os resultados final e intermediários do processo de geração da informação lateral, bem como sua avaliação se comparada ao quadro original .

Como pode ser visto nas Figuras 4.5 e 4.7, a qualidade da informação lateral cai com a distância entre os quadros chave, pois a correlação entre o quadro chave e o quadro Wyner-Ziv diminui. A qualidade da informação lateral afeta diretamente a qualidade da codificação Wyner-Ziv. Com isso, existe uma região em que não é mais vantajoso adicionar quadros Wyner-Ziv entre quadros chave. No codificador implementado, foram utilizados 1 ou 2 quadros Wyner-Ziv entre cada quadro chave.

4.4.2 Decodificação Slepian-Wolf

O codificador de canal utilizado foi um LDPCA (*Low Density Parity Check Accumulate Codes*) com taxa adaptativa proposto por Girod et. al. [32]. Como o foco do trabalho não foi o desenvolvimento de um codificador de vídeo Wyner-Ziv, utilizou-se por conveniência o código desenvolvido pelo grupo da Universidade de Stanford, disponível em [49]. Este codificador Slepian-Wolf faz uma modificação em um código LDPC, adaptando um canal de retorno para realizar codificação de fontes distribuída à uma taxa próxima ao limite de Slepian-Wolf.

O código LDPCA funciona da seguinte maneira: na codificação de uma fonte X os bits de síndrome são acumulados em um *buffer* e uma parte desses bits são enviados ao decodificador. Ao decodificador é apresentada uma informação Y, que atua como uma versão degradada de X, e o decodificador utiliza as duas informações (Y e os bits de síndrome recebidos) para tentar corrigir os erros de Y. Se os bits de síndrome transmitidos não forem suficientes, o decodificador faz a requisição de mais bits de síndrome ao codificador. Esse processo continua até que todos os erros tenham sido corrigidos. Este decodificador funciona como um codificador Slepian-Wolf (sem perdas), onde a informação X é a fonte e Y é sua informação lateral. Como os códigos LDPC são formados por matrizes esparsas, os bits de síndrome não sofrem exatamente uma perfuração (*puncturing*) [20]: ao invés disso, eles são combinados (acumulados utilizando soma módulo 2).

A vantagem de se ter um canal de retorno é que, caso a correlação entre X e Y seja alta,

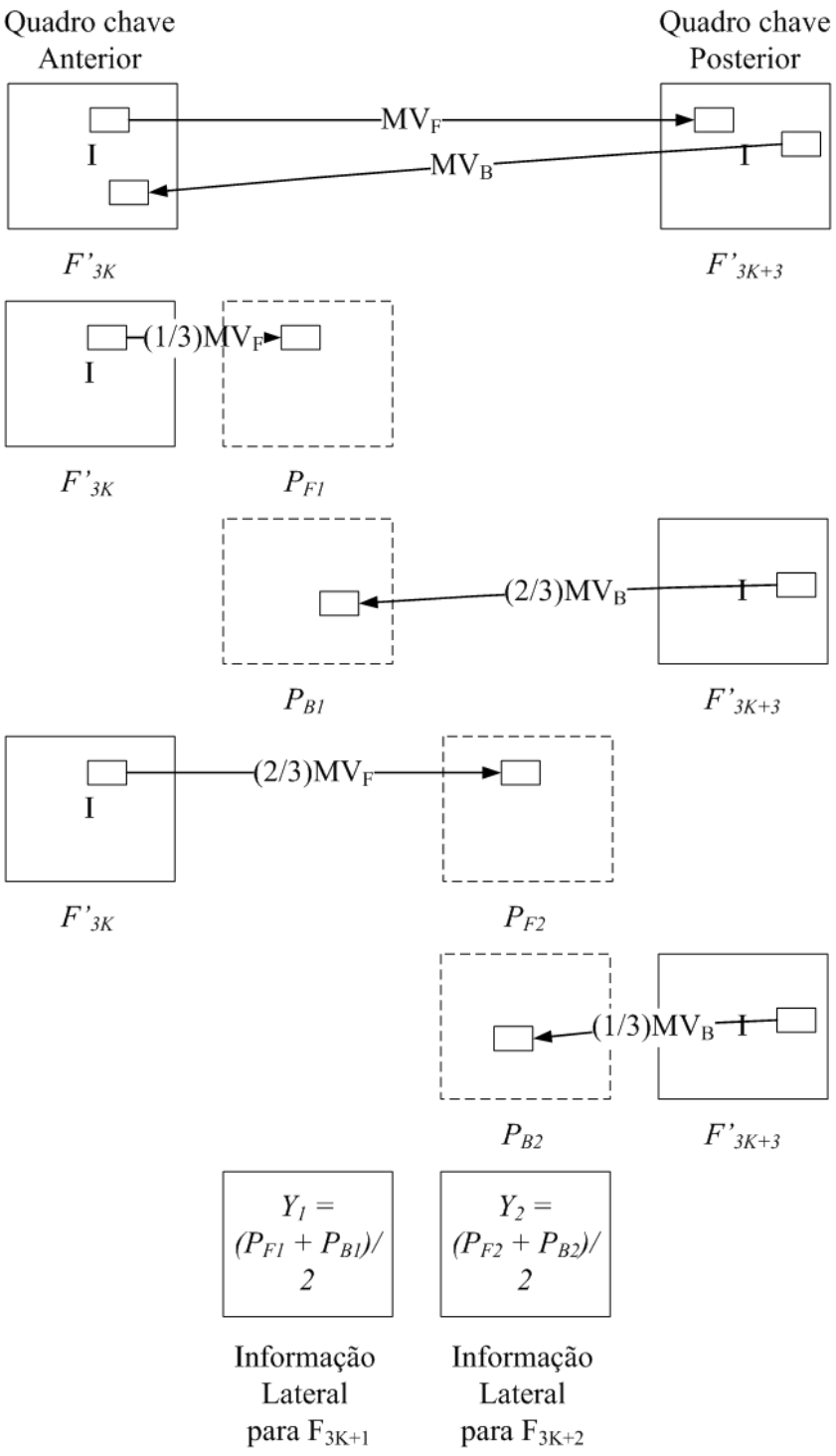


Figura 4.6: Processo de geração da informação lateral para 2 quadros entre os quadros chave. Os vetores de movimento MV_F e MV_B são calculados utilizando os quadros chave. Em seguida, utiliza-se $\frac{MV_F}{3}$ para gerar o quadro P_{F1} e $\frac{2 \cdot MV_B}{3}$ para gerar o quadro P_{B1}. A informação lateral final para o quadro F'_{3K+1} é tomada como $Y_1 = \frac{P_{F1}+P_{B1}}{2}$. Depois, utiliza-se $\frac{2 \cdot MV_F}{3}$ para gerar o quadro P_{F2} e $\frac{MV_B}{3}$ para gerar o quadro P_{B2}. A informação lateral final para o quadro F'_{3K+2} é tomada como $Y_2 = \frac{P_{F2}+P_{B2}}{2}$.

(a) Quadro chave Anterior (F'_{3K})

(b) Quadro chave Posterior (F'_{3K+3})

(c) Quadro Wyner-Ziv Original (F_{3K+1})

(d) Informação Lateral (Y_1), PSNR = 30.0927 dB.

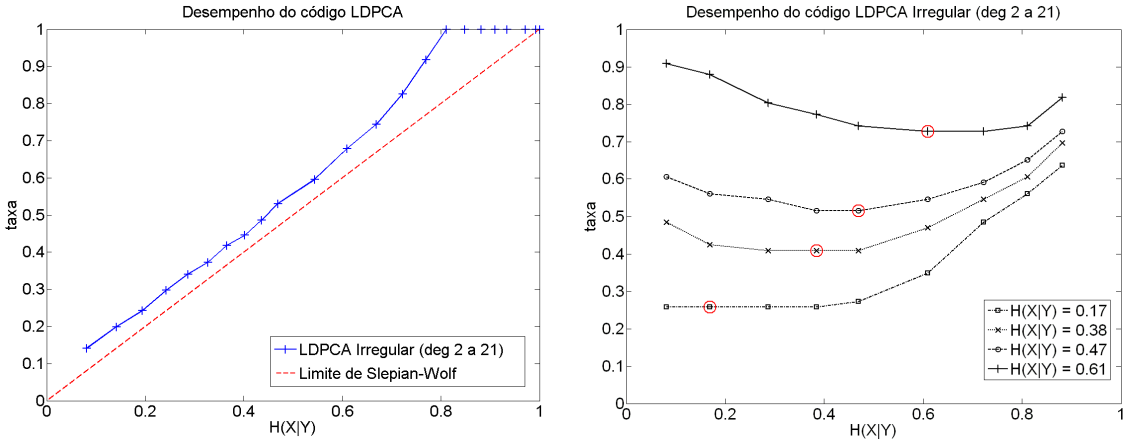
(e) Quadro Wyner-Ziv Original (F_{3K+2})

(f) Informação Lateral (Y_2), PSNR = 30.1099 dB.

Figura 4.7: Processo de geração da informação lateral. A geração de 2 quadros é visivelmente inferior à geração de um único quadro de informação lateral (Figura 4.5).

poucos bits seriam transmitidos. No entanto, caso Y não seja tão fiel à informação original, mais bits são transmitidos. A taxa deste código é, portanto, muito dependente da qualidade da informação lateral Y com respeito à taxa, uma vez que a decodificação sempre continua até que não haja mais distorção. Por esse motivo este codificador é chamado de um codificador com taxa adaptativa. A desvantagem, por outro lado, é que a decodificação deve ser feita *online*, ou seja, à medida em que o codificador codifica os quadros o decodificador deve decodificá-los.

O desempenho do codificador LDPCA utilizado é mostrado na Figura 4.8. O limite teórico é o teorema de Slepian-Wolf, no qual a taxa mínima de transmissão é $H\left(X|Y\right)$. Foi utilizada a matriz *6336_irregDeg2to21*, também disponível em [49]. Na Figura 4.8(a), as sequências são decodificadas considerando que o decodificador conhece as estatísticas de $X|Y$. Na Figura 4.8(b) são mostradas algumas curvas quando o decodificador não conhece esse dado (isto é, ele tenta decodificar a sequência recebida considerando que a estatística é diferente do que realmente é).



(a) Desempenho quando a estatística é conhecida. (b) Desempenho quando a estatística não é conhecida.

Figura 4.8: Desempenho do codificador LDPCA.

As curvas da Figura 4.8(b) consideram que o decodificador irá utilizar diferentes $H\left(X|Y\right)$ para decodificar um mesmo par X e Y fixo (isto é, $H\left(X|Y\right)$ é fixo, mostrado na legenda). Podemos observar, sem surpresas, que o ponto mínimo (denotado por um círculo) de cada curva da Figura 4.8(b) ocorre quando o decodificador utiliza a estatística real de $X|Y$. No entanto, esta não é uma premissa razoável. Pode-se observar que caso o decodificador use uma estatística próxima à estatística real, a perda em taxa é pequena e, portanto, um bom estimador dessa estatística aumentaria a eficiência do codificador LDPCA.

4.4.3 Decodificação de um Quadro Wyner-Ziv

Para utilizar o código LDPCA no codificador de vídeo, utilizamos a informação $R = X_q - X_{erq}$ como entrada para o codificador. No decodificador, utilizamos a informação $R' = Y_q - X_{erq}$ como informação lateral. A informação X_{erq} é igual no codificador e no decodificador, de modo que não ocorre *drift*.

Para cada elemento de R deve ser associado um código binário, para que este possa ser codificado com o codificador de canal binário utilizado. Lembrando que o resíduo entre quadros vizinhos em uma sequência de vídeo pode ser modelado com uma distribuição laplaciana [13], como é mostrado na Figura 4.9, os índices mais próximos de 0 tem maior probabilidade de ocorrência. Foi feita uma tentativa de assinalar códigos binários próximos (isto é, com uma distância de Hamming pequena) para números próximos, de modo que um erro pequeno no número decimal não acarrete um erro grande em sua representação binária. Logo, os índices de quantização R são mapeados para números decimais (D_e) que geram números binários (B) com a característica desejada. A Tabela 4.2 mostra como são assinalados os números binários para um $QPWZ$ de 32. Para outros valores de $QPWZ$, o mapeamento dos resíduos é feito de forma similar.

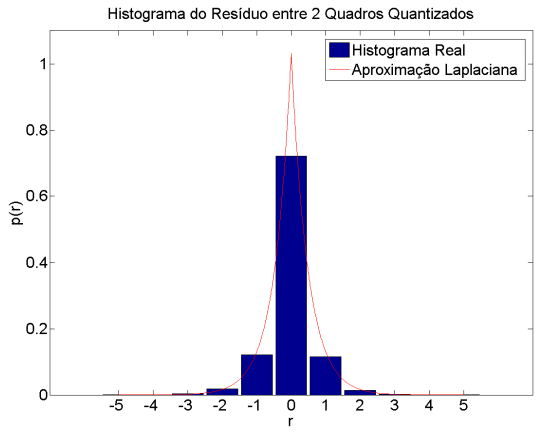


Figura 4.9: Distribuição de Probabilidade do Resíduo entre Quadros Quantizados. A Distribuição Laplaciana tem desvio padrão $\sigma = 0.68$.

O resíduo $R = X - X_{er}$ calculado no codificador é binarizado utilizando a codificação *signed* Exp-Golomb encontrada na especificação do padrão H.264, e é mostrada na Tabela 4.2. Em seguida, os bits são divididos em planos de bits, do mais significativo ao menos significativo.

Tabela 4.2: Assinalando números binários de 5 dígitos ao resíduo.

R	D_e	B
0	0	00000
1	1	00001
-1	2	00010
2	3	00011
-2	4	00100
3	5	00101
-3	6	00110
$(-1)^{k+1} \cdot \lfloor \frac{k}{2} \rfloor$	k	$b(k)$

Tabela 4.3: Relação entre o passo de quantização dos quadros Wyner-Ziv, QP utilizado para os quadros-chave e o Número de Planos de Bits codificados

QP	7	6	5	4	3
$QPWZ$	64	48	32	16	8
$PlanosdeBits$	4	4	4	5	6

O número de planos de bits depende do $QPWZ$ do quadro codificado, como pode ser visto na Tabela 4.3, devido à quantidade de planos de bits requeridos por cada passo de quantização.

Foi utilizado um código de 6336 bits. Para codificar um quadro em resolução QCIF (176×144 *pixels*, totalizando 25344 *pixels*), após a divisão em plano de bits, cada um deles é reordenado em uma sequência de 25344 bits e dividida em 4 blocos de 6336 bits cada um. O agrupamento é feito da esquerda para a direita, de cima para baixo. O decodificador precisa da probabilidade de erro entre a fonte X e sua informação lateral Y. Como esta informação não está disponível no decodificador, foi feito um teste para avaliar a robustez do decodificador com relação ao uso de uma probabilidade de erro diferente da probabilidade real, como mostra a Figura 4.8(b). Nesta figura, é possível ver que, caso a probabilidade de erro seja próxima da probabilidade real, o desempenho da decodificação não é seriamente afetado. Na implementação feita, o primeiro bloco codificado de cada quadro utiliza uma probabilidade de erro entre a fonte X e sua informação lateral Y de 0.05. Para os demais blocos, calcula-se o erro entre a informação lateral e a informação decodificada do bloco anterior (que é, teoricamente, igual à fonte X, uma

vez que o canal de retorno irá realizar iterações até que a decodificação seja perfeita) e utiliza-se essa probabilidade como probabilidade de erro do bloco atual.

(a) Quadro Original

(b) Quadro quantizado no codificador. PSNR = 35.14 dB.

(c) Quadro quantizado no decodificador. PSNR = 35.11 dB.

Figura 4.10: Exemplo da decodificação de um quadro Wyner-Ziv. A codificação do quadro quantizado ocorre praticamente sem perdas.

A Figura 4.10 mostra o quadro Wyner-Ziv original (Figura 4.10(a)), o quadro quantizado X_q que será transmitido (Figura 4.10(b)) e o quadro decodificado $\hat{X}_q$ (Figura 4.10(c)). Embora tenha sido transmitido apenas o resíduo $R = X_q - X_{erq}$, a Figura 4.10 mostra os quadros já somados ao quadro de referência X_{erq} para melhor visualização. A PSNR entre o quadro enviado pelo codificador e o quadro decodificado é 57.29 dB, o que mostra que a decodificação deste quadro específico não foi perfeita, havendo um pequeno erro na decodificação. Ainda assim, no entanto, a transmissão será considerada sem perdas para efeito da reconstrução por máxima verossimilhança.

A Tabela 4.5 mostra o número de bits enviado para cada plano de bits da imagem, bem como

Tabela 4.4: Assinalando números binários ao resíduo.

R	D_e	B
0	0	00000
1	1	00001
-1	17	10001
2	2	00010
-2	18	10010
3	3	00011
-3	19	10011

Tabela 4.5: Quantidade de Bits Requerido para codificar um Quadro Wyner-Ziv com $QP = 4$

Plano de Bits	Mapeamento da Tabela 4.2	Mapeamento da Tabela 4.4
1	768	11232
2	1344	768
3	4800	1440
4	14208	6433
5	10848	20256
Total	31968	40129

o total de bits enviado. O ganho obtido com o uso do mapeamento da Tabela 4.2 pode ser visto na Tabela 4.5, que mostra o número de bits enviados utilizando o mapeamento Exp-Golomb utilizado e o número de bits enviados utilizando um mapeamento mais simples, no qual o primeiro bit representa o sinal (0 para positivo e 1 para negativo) e os demais bits representam o número decimal convertido para binário. Esse segundo mapeamento é mostrado na Tabela 4.4.

4.4.4 Reconstrução por Máxima Verossimilhança

Após a decodificação de canal, o decodificador tem duas versões para o quadro Wyner-Ziv transmitido X: o resultado da decodificação de canal Q, que é uma versão quantizada do quadro Wyner-Ziv original, e a informação lateral Y, que apesar de não ser quantizada, apresenta distorções de blocos características de imagens compensadas. As três imagens são mostradas na Figura 4.11.

(a) Quadro Original X

(b) Quadro quantizado Q. PSNR = 35.11 dB.

(c) Informação lateral Y. PSNR = 33.28 dB.

Figura 4.11: Exemplos de quadros que serão utilizados na reconstrução por máxima verossimilhança. O quadro quantizado e a informação lateral se referem ao mesmo quadro Wyner-Ziv. O processo de reconstrução consiste em combinar as duas informações para gerar uma melhor aproximação do quadro original.

O processo de reconstrução por Máxima Verossimilhança consiste em combinar as duas informações Q e Y para gerar a melhor aproximação $\hat{X}$ para X.

O quadro X é quantizado para Q utilizando um quantizador Φ utilizando o parâmetro $QPWZ$ como mostra a equação:

$$Q = \Phi\left(X, QPWZ\right) = \left\lfloor \frac{X}{QPWZ} \right\rfloor \tag{4.3}$$

Se um dado bin q corresponde ao intervalo $\left[x_l\left(q\right), x_h\left(q\right)\right]$, então sua probabilidade de ocorrência é dada por:

$$p_Q\left(q\right) = \int_{x_l(q)}^{x_h(q)} f_x\left(x\right) dx \tag{4.4}$$

onde $f_x\left(x\right)$ é a função de densidade de probabilidade (fdp) de X, $x_l\left(q\right)$ é o limite inferior do setor de quantização q e $x_h\left(q\right)$ é o limite superior do setor de quantização q.

Para a decodificação, a função de reconstrução de máxima verossimilhança é dada por [50]:

$$\hat{X}_{YQ}\left(y,q\right) = E\left\{X|Y=y, Q=q\right\} = E\left\{X|Y=y, \Phi\left(X, QPWZ\right)=q\right\} \tag{4.5}$$

$$\hat{X}_{YQ}\left(y,q\right) = \frac{\int_{x_l(q)}^{x_h(q)} x \cdot f_{x|y}\left(x,y\right)}{\int_{x_l(q)}^{x_h(q)} f_{x|y}\left(x,y\right)} \tag{4.6}$$

Lembrando o teorema de Bayes, a probabilidade condicional $f_{x|y}\left(x,y\right)$ pode ser derivada como:

$$f_{x|y}\left(x,y\right) = \frac{f_{xy}\left(x,y\right)}{f_y\left(y\right)} \tag{4.7}$$

onde $f_{xy}\left(x,y\right)$ é a fdp conjunta de X e Y e $f_y\left(y\right)$ é a fdp de Y. Como a informação lateral Y está disponível no codificador, e os valores dos *pixels* estão no intervalo $\left[0...255\right]$, a função $f_y\left(y\right)$ é computacionalmente tratável no decodificador, que foi considerado como uma máquina sem restrição de recursos.

A fdp conjunta de X e Y, por outro lado, não está disponível no decodificador, e deve ser estimada de outro modo. Além disso, não podemos considerar que X e Y são independentes.

Definimos, então, uma variável aleatória Z com a seguinte construção:

$$Z = X - Y \tag{4.8}$$

Com essa construção, a variável Z tem uma distribuição que pode ser aproximada por uma distribuição laplaciana com média zero, como mostra a Figura 4.12. Essa distribuição é definida apenas pelo parâmetro λ, como mostra a equação:

$$f_z(z) = \frac{\lambda}{2} \cdot e^{-\lambda|z|} \tag{4.9}$$

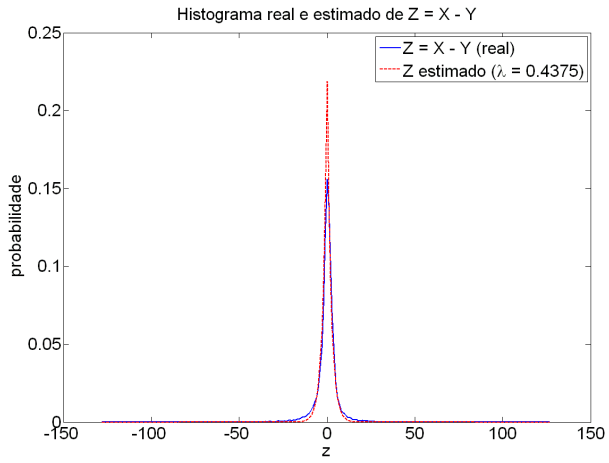


Figura 4.12: Distribuição de probabilidade de Z.

Com a variável Z definida pela equação 4.8, temos que:

$$X = Y + Z \tag{4.10}$$

Além disso, definimos $f_y(y)$ como sendo a fdp de Y, $f_z(z)$ como sendo a fdp de Z e, considerando Z e Y duas variáveis aleatórias independentes, a fdp conjunta de Y e Z seria dada por:

$$f_{yz}(y, z) = f_y(y) \cdot f_z(z) \tag{4.11}$$

Faremos, agora, uma mudança de variáveis das variáveis Y e Z, ambas disponíveis no decodificador se considerarmos que o decodificador tem o parâmetro da variância que define

Z, definidas como:

$$X = g_1(y, z) = Y + Z$$
$$W = g_2(y, z) = Y \tag{4.12}$$

onde $g_1(y, z)$ é a função que mapeia Y e Z para a variável X (note que a equação segue da definição de Z) e $g_2(y, z)$ é a função que mapeia Y e Z para a variável W. Da mesma forma:

$$Y = h_1(w, x) = W$$
$$Z = h_2(w, x) = X - W \tag{4.13}$$

Essa transformação é especificamente eficiente por causa do Jacobiano resultante [51]:

$$J_{h_1 h_2}(w, x) = \begin{vmatrix} \frac{\partial h_1}{\partial w} & \frac{\partial h_1}{\partial x} \\ \frac{\partial h_2}{\partial w} & \frac{\partial h_2}{\partial x} \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix} = 1 \tag{4.14}$$

o que nos leva a:

$$f_{xw}(x, w) = f_{yz}(y = h_1(w, x) = w, z = h_2(w, x) = x - w) = f_{yz}(w, x - w) \tag{4.15}$$

Lembrando agora que, na definição da equação 4.12 $W = Y$ temos:

$$f_{xy}(x, y) = f_{yz}(y, x - y) = f_y(y) \cdot f_z(x - y) \tag{4.16}$$

E, portanto:

$$f_{x|y}(x, y) = \frac{f_y(y) \cdot f_z(x - y)}{f_y(y)} = f_z(x - y) \tag{4.17}$$

Este resultado poderia ser obtido intuitivamente, uma vez que se $X = Y + Z$, então dado $Y = y$, temos que a variável aleatória $X|Y = y$ é dada pela soma de uma variável aleatória

Tabela 4.6: Relação entre o QP utilizado para os quadros-chave e o parâmetro λ usado para gerar a variável Z

QP	7	6	5	4	3
λ_{QP}	0.3347	0.3655	0.3854	0.4375	0.4669

Z com uma constante dada $Y = y$. Como a distribuição de Z é laplaciana com média 0, a distribuição de $X|Y = y$ é também laplaciana com média $Y = y$.

A variável Z não está disponível no decodificador, que conhece apenas a variável Y. No entanto, sabe-se que Z pode ser aproximada por uma distribuição laplaciana com média 0, que pode ser totalmente descrita com um único parâmetro λ (Equação 4.9). Como a informação lateral Y é muito dependente da qualidade dos quadros chaves, que é controlada pelo parâmetro QP, é natural que a variável Z também seja dependente do QP utilizado, pois uma melhor qualidade na informação lateral Y leva a uma menor variância na variável $Z = X - Y$, enquanto uma qualidade pior da informação lateral Y leva a uma maior variância para Z. Dessa forma, utilizamos o valor λ_{QP} para indicar que o parâmetro λ é dependente do QP utilizado.

Foi utilizado para o parâmetro λ_{QP} o valor $\lambda_{QP} = \alpha \cdot \bar{\lambda}$, onde $\bar{\lambda}$ denota a média do parâmetro λ real (isto é, aproximando a variável aleatória $Z = X - Y$ através da Equação 4.9) calculado para os primeiros 51 quadros de 4 sequências de vídeo populares: *Foreman*, *Salesman*, *CarPhone* e *Akiyo*. O parâmetro λ_{QP} reflete o quanto a reconstrução confia na informação lateral Y. Experimentalmente, foi notado que uma maior confiança (isto é, um menor valor de λ_{QP}) na informação lateral Y leva a resultados piores na reconstrução do que uma menor confiança. Dessa forma, foi obtido empiricamente o valor de $\alpha = 1,25$. O valor utilizado pelo decodificador para estimar a variável aleatória Z pode ser visto na Tabela 4.6.

A Figura 4.13 mostra a distribuição de X e Y para o primeiro quadro Wyner-Ziv da sequência *Foreman*. Note que a distribuição de X não está disponível no decodificador, mas foi mostrada para que se possa ver a relação com Y. Por outro lado, a distribuição de Y pode ser calculada a partir do quadro Y.

Combinando a fdp de Y mostrada na Figura 4.13 e Z mostrada na Figura 4.12, obtemos $f_{zy}(z, y)$, mostrada na Figura 4.14 vista por dois ângulos diferentes.

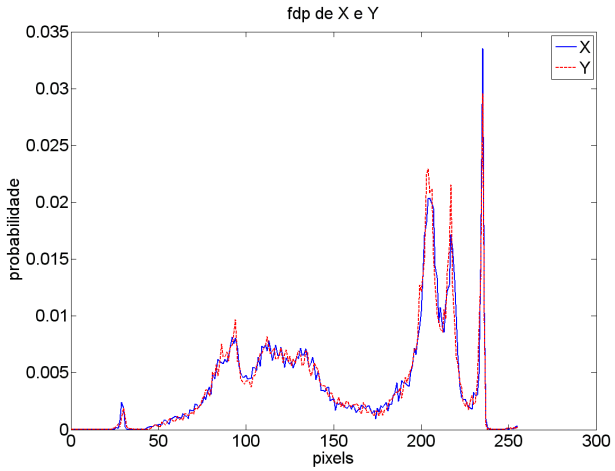
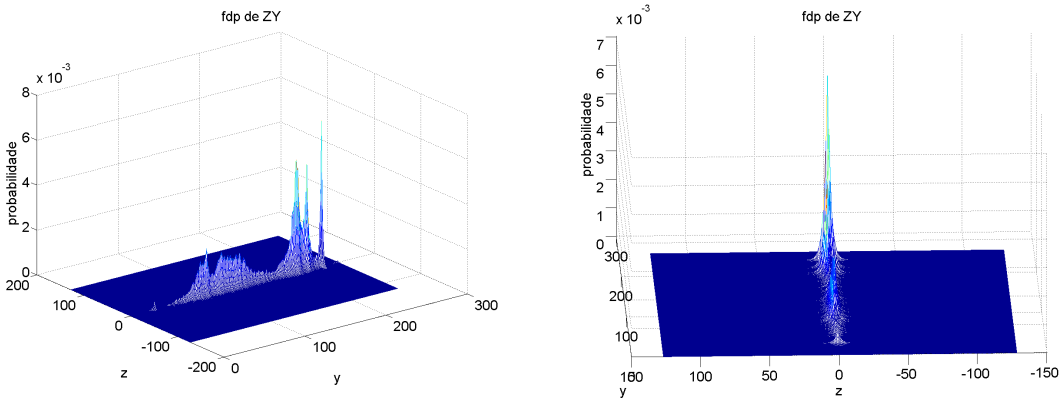


Figura 4.13: Distribuição de probabilidade de X e Y.



(a) Vista de $f_{zy}(z, y)$.

(b) Vista de $f_{zy}(z, y)$.

Figura 4.14: Distribuição de probabilidade de ZY.

A Figura 4.15 mostra $f_{xy}(x, y)$ obtida através de $f_{zy}(z, y)$.

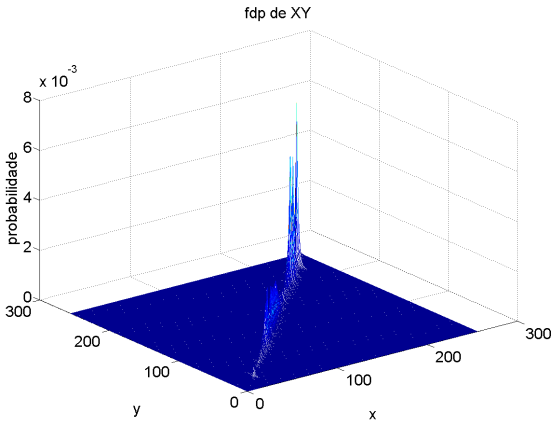


Figura 4.15: Distribuição de probabilidade de XY.

Por fim, estimamos $f_{x|y}(x, y)$, como mostra a Figura 4.16.

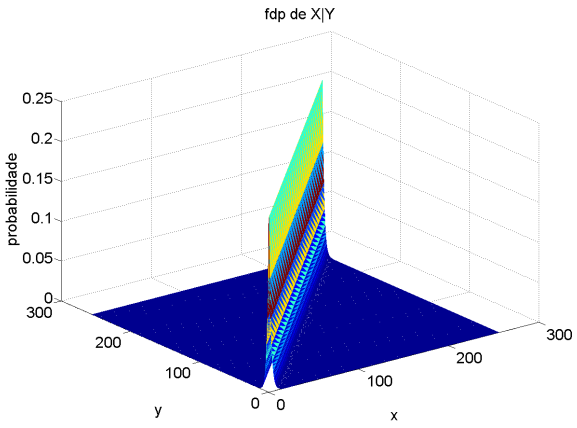


Figura 4.16: Distribuição de probabilidade de $X|Y$.

A Figura 4.17 ilustra o processo para um *pixel* específico $(109, 110)$ do primeiro quadro Wyner-Ziv da sequência *Foreman*. O decodificador dispõe de duas informações: $Q = 152$ e $Y = 141$. O gráfico mostra $f_{x|y}(x|y = 141)$. Podemos ver que, partindo apenas da informação lateral Y, a maior probabilidade é de que X seja igual à informação lateral, e as probabilidades vão diminuindo para os valores próximos. Porém, como mostrado na Figura 4.17, a informação Q, confiável, desloca a área possível para X para a área demarcada (ou seja, o intervalo $[x_l(q)\, x_h(q)]$. Calculando, então, $\hat{X} = E\{X|Y = 141, Q = 152\} = 147$. Neste caso específico, $X(109, 110) = 147$, de forma que a reconstrução combinou as duas informações da maneira correta.

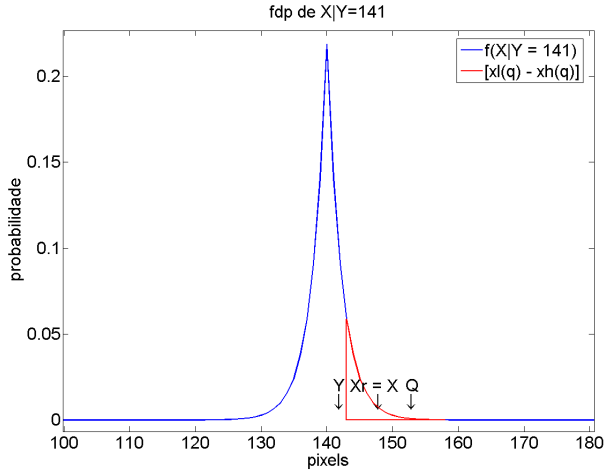


Figura 4.17: Ilustração do processo de reconstrução para um dado *pixel* da sequência *Foreman*. A informação lateral $Y = 141$ gera uma curva de probabilidade para a informação original X. O *pixel* quantizado $Q = 152$ desloca a região possível onde o valor original de X se encontrava. O cálculo $\hat{X} = E\{X|Y = 141, Q = 152\} = 147$ gera uma melhor aproximação do valor original de X.

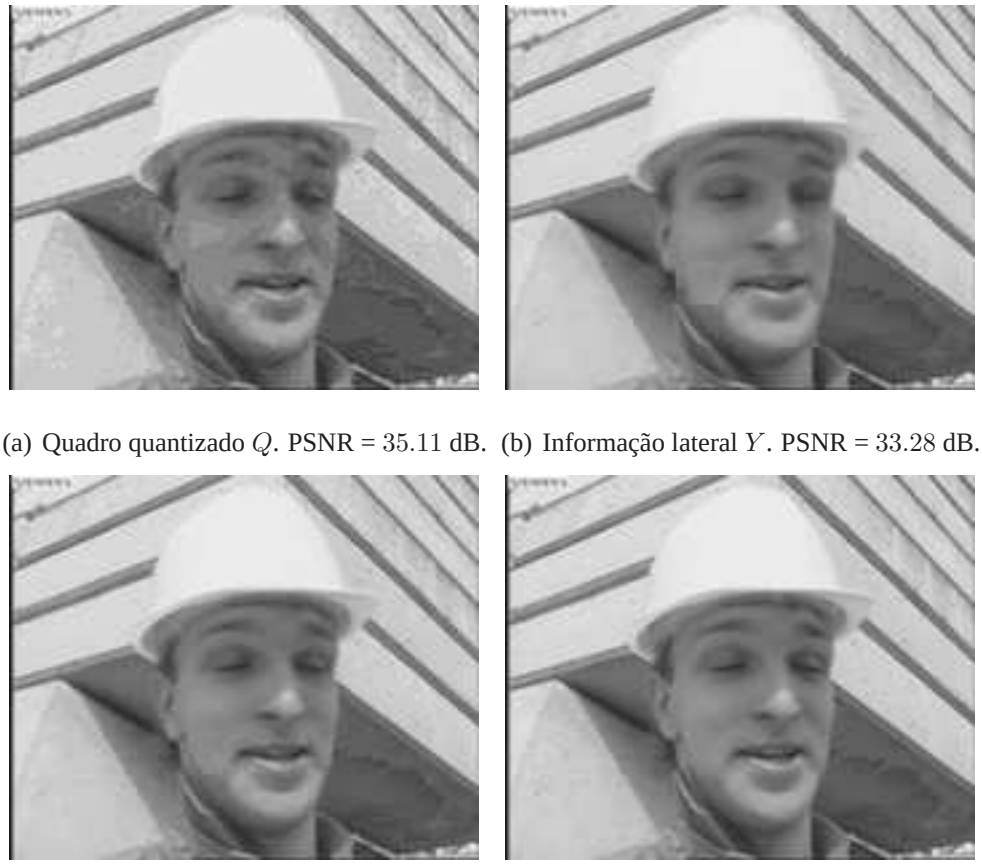
Tabela 4.7: Codificação dos quadros Wyner-Ziv de Crominância

	$bits$	PSNR	$bits$	PSNR (dB)
C_b	0	42.57	3744	42.58
C_r	0	43.57	3840	43.57

Por fim, a Figura 4.18 mostra o resultado da reconstrução. O quadro final $\hat{X}$ é muito mais próximo de X do que Q ou Y.

4.5 CODIFICAÇÃO DOS QUADROS DE CROMINÂNCIA

Para os quadros de crominância, o sistema visual humano é menos rigoroso [39]. Foi verificado que utilizar o processo de geração da informação lateral gera bons resultados, e que utilizar o codificador LDPCA para codificar os quadros de crominância aumentava bastante a taxa, porém não melhorava tanto o resultado. Isto pode ser visto na Tabela 4.7.



(a) Quadro quantizado Q. PSNR = 35.11 dB.

(b) Informação lateral Y. PSNR = 33.28 dB.

(c) Quadro reconstruído $\hat{X}$. PSNR = 38.44 dB.

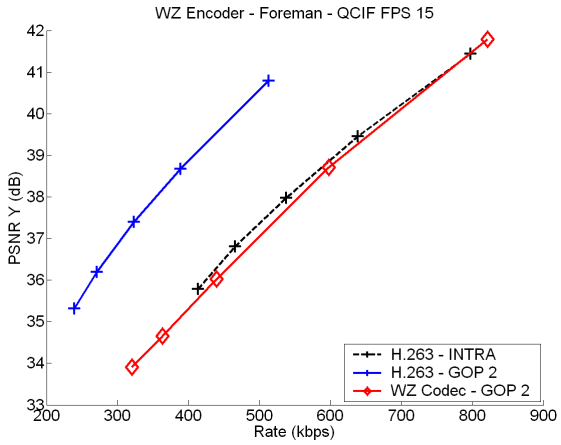
(d) Quadro Original X.

Figura 4.18: Resultados da reconstrução por máxima verossimilhança. O quadro reconstruído da Figura 4.18(c) é muito mais próximo do quadro original da Figura 4.18(d) do que sua informação lateral (Figura 4.18(b)) ou quantizada (Figura 4.18(a)).

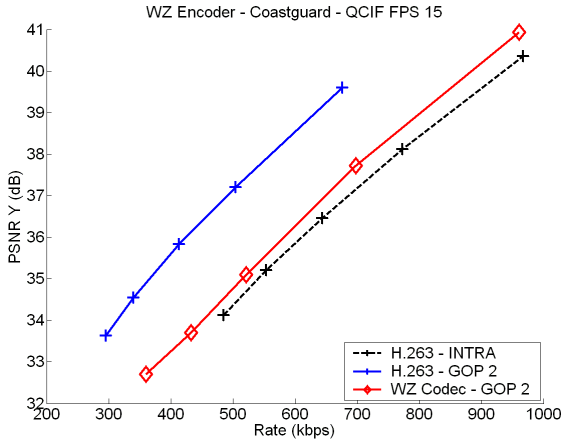
4.6 RESULTADOS DO CODIFICADOR WYNER-ZIV

Os resultados do codificador Wyner-Ziv podem ser vistos na Figura 4.19, para as sequências *Foreman*, *Coastguard*, *Carphone* e *Salesman* com GOP de comprimento 2. Na Figura 4.20 são mostrados os resultados para as sequências *Foreman* e *Salesman* com o GOP de comprimento 3. O resultado do codificador é mostrado comparado com o H.263 *intra* e com o H.263 com o GOP igual ao da sequência Wyner-Ziv.

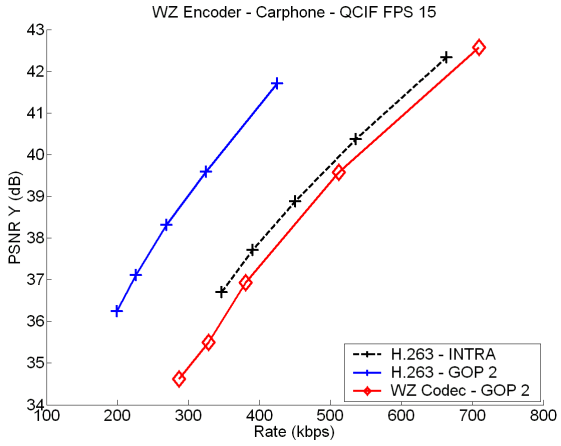
Como esperado, o codificador apresenta melhor desempenho nas sequências com menor movimento (como a sequência *Salesman*), pois a qualidade da informação lateral aumenta nessas sequências. Isto pode ser visto, em especial, na Figura 4.20, que utiliza um GOP de comprimento 3, o que deteriora ainda mais a qualidade da informação lateral.



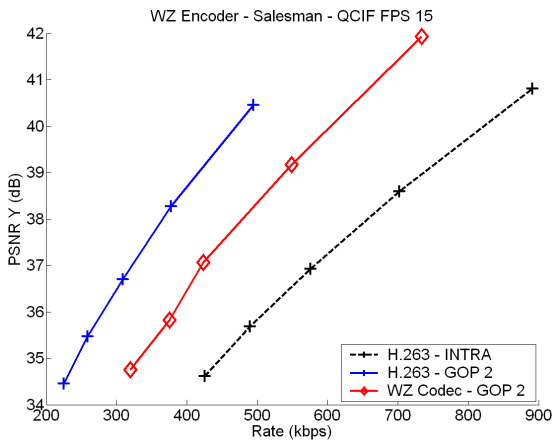
(a) Sequência Foreman.



(b) Sequência Coastguard.

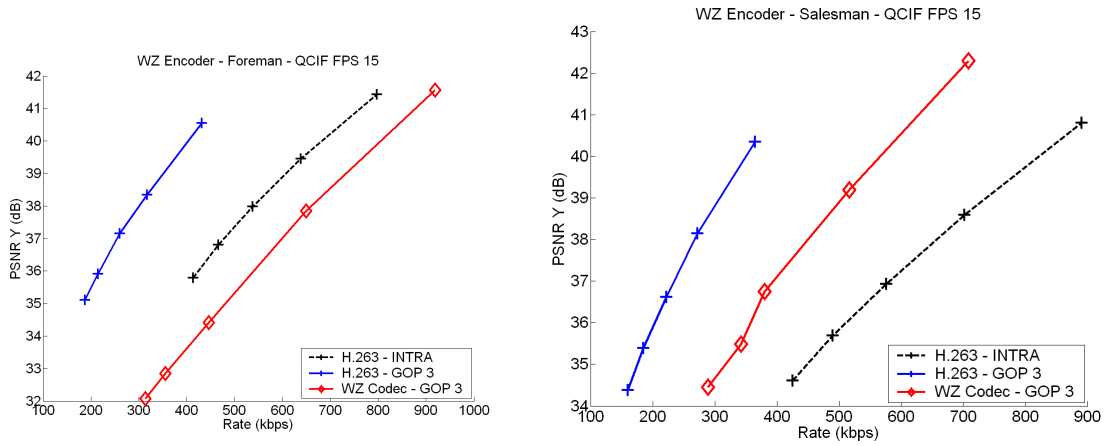


(c) Sequência Carphone.



(d) Sequência Salesman.

Figura 4.19: Resultado do codificador para várias sequências utilizando um GOP de comprimento 2.



(a) Sequência Foreman.

(b) Sequência Salesman.

Figura 4.20: Resultado do codificador para várias sequências utilizando um GOP de comprimento 3.

5 TRANSCODIFICADOR WYNER-ZIV/H.263

5.1 INTRODUÇÃO

O codec de vídeo mostrado no capítulo 4 apresenta uma redução de complexidade do lado do codificador, se comparado com um codificador tradicional, pois é baseado em conceitos de DVC. No entanto, ele apenas transfere a complexidade para o decodificador, fazendo deste uma máquina muito mais complexa que um decodificador de vídeo tradicional.

Quando movemos para um ambiente onde o codificador e o decodificador tem restrições de recursos, nem o codec tradicional nem o codec Wyner-Ziv podem trabalhar em seus melhores modos. Dessa forma, propomos um transcodificador de vídeo que receba uma sequência codificada com o codificador Wyner-Ziv e transcodifique para um codec tradicional. Neste trabalho, o codificador de vídeo tradicional foi escolhido como o H.263.

Nesse esquema, a sequência seria codificada por um aparelho com restrição de recursos utilizando o codificador Wyner-Ziv e transmitida para um servidor, que trabalharia como um transcodificador. Esse servidor transcodificaria a sequência para que possa ser decodificada com um decodificador tradicional, no caso um decodificador H.263. A complexidade deste esquema está concentrada no servidor, que é considerado aqui uma máquina mais poderosa, sem limitações de energia ou processamento. Este sistema pode ser visto na Figura 5.1.

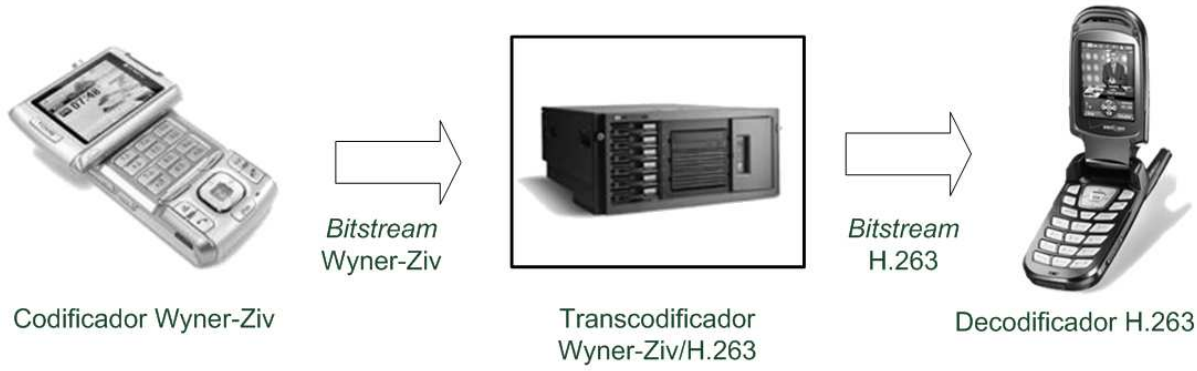


Figura 5.1: Esquema do transcodificador Wyner-Ziv/H.263.

A primeira abordagem na construção de um transcodificador é simplesmente concatenar

um decodificador Wyner-Ziv com um codificador H.263. Essa abordagem é chamada de transcodificador trivial e seu diagrama de blocos pode ser observado na Figura 5.2. A desvantagem dessa abordagem é o grande acúmulo de complexidade no servidor, pois tanto o decodificador Wyner-Ziv quanto o codificador H.263 requerem alto esforço computacional. O objetivo deste trabalho é propor um transcodificador menos complexo do que o transcodificador trivial, mais apropriado para uma implementação em um ambiente real.

A tarefa do transcodificador é modificar o *bitstream* recebido, que foi codificado com um codificador Wyner-Ziv, em um *bitstream* compatível com um decodificador H.263. Podemos dividir o *bitstream* gerado pelo codificador Wyner-Ziv em dois: um contendo os quadros chave, e o outro contendo os quadros Wyner-Ziv. O *bitstream* referente aos quadros chave já é compatível com um decodificador H.263 - ele consiste apenas dos quadros originais codificados com um codificador H.263 *intra*. Porém, o *bitstream* dos quadros Wyner-Ziv é muito diferente do *bitstream* esperado por um decodificador H.263, uma vez que ele consiste do quadro Wyner-Ziv codificado com um codificador de canal LDPC.

Uma vez que a decodificação de um quadro Wyner-Ziv é fundamentalmente diferente da decodificação de um quadro chave (realizada com um decodificador H.263 regular), esses quadros devem ser decodificados e re-codificados no transcodificador, para que o *bitstream* gerado seja compatível com um decodificador H.263. No entanto, analisando a Figura 5.2, pode-se perceber que tanto a decodificação Wyner-Ziv quanto a codificação H.263 realizam processoes de estimação de movimento. Como esta é a tarefa mais complexa do codificador H.263, o transcodificador proposto irá utilizar informações geradas na decodificação dos quadros Wyner-Ziv para evitar realizar mais uma vez a estimação de movimento com busca completa.

A seção 5.2 propõe o transcodificador, apresentando suas principais idéias. A seção 5.3 discute o uso de estimação de movimento no transcodificador. As seções 5.4 e 5.5 mostram as opções do transcodificador proposto.

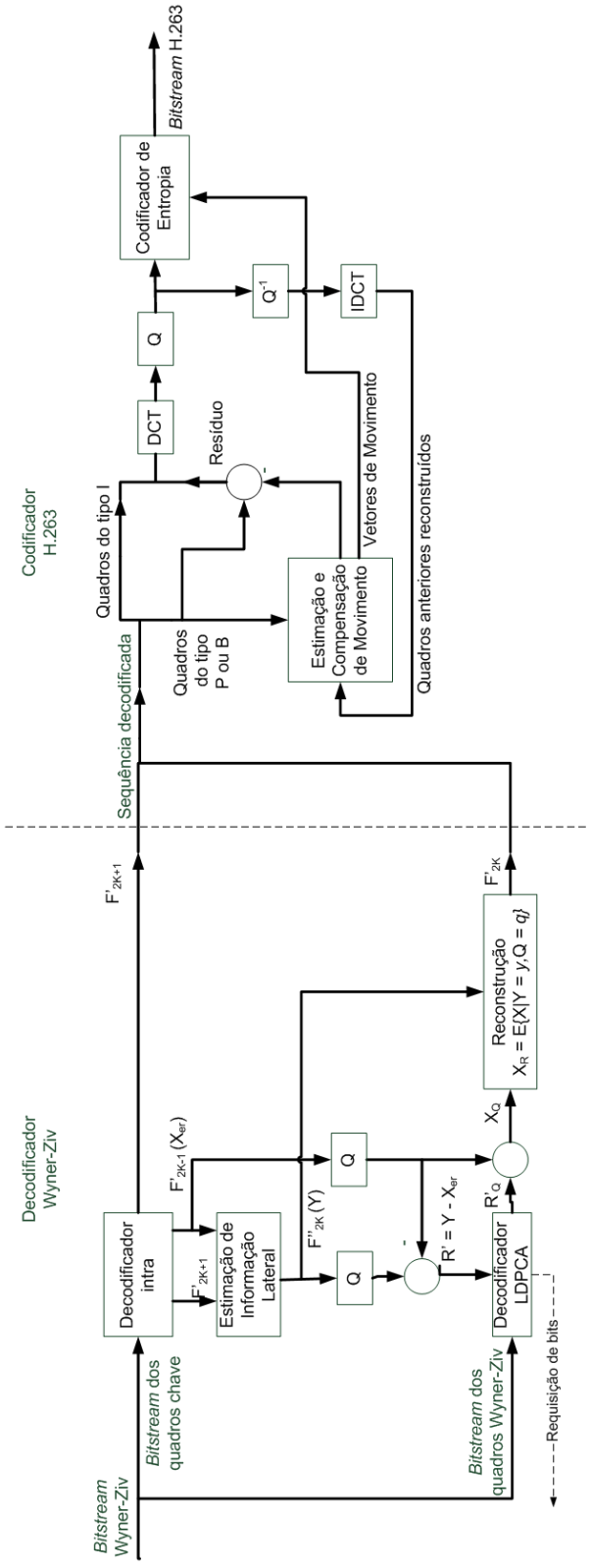


Figura 5.2: Esquemático do transcodificador trivial: um decodificador Wyner-Ziv em cascata com um codificador H.263.

5.2 O TRANSCODIFICADOR

O diagrama do transcodificador proposto [52] é mostrado na Figura 5.3. O decodificador Wyner-Ziv é parte do transcodificador. Ao invés de decodificar e re-codificar a sequência inteira, o transcodificador utiliza informações obtidas no processo de decodificação Wyner-Ziv, como os vetores de movimento calculados na geração da informação lateral, para acelerar a transcodificação. As idéias principais do transcodificador são:

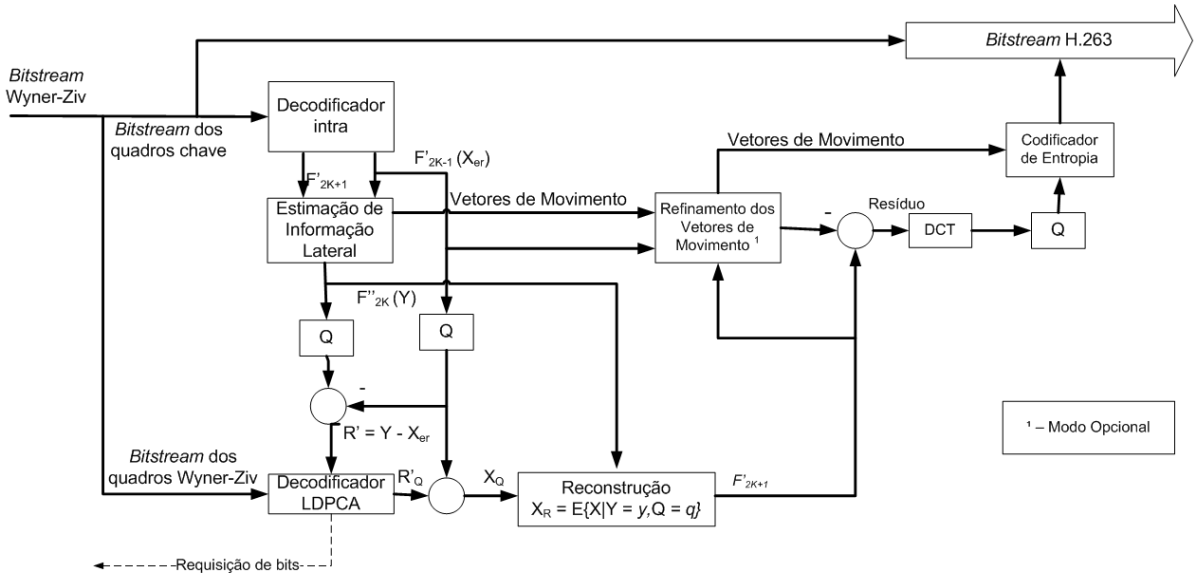


Figura 5.3: Diagrama do transcodificador.

- Copiar os *bitstreams* dos quadros que não terão seu tipo modificado (quadros chave).
- Reutilizar os vetores de movimento calculados no processo de geração da informação lateral, ao invés de re-calcular estes vetores.
- Opcionalmente, refinar estes vetores de movimento.

Existem várias opções para o transcodificador. O comprimento do GOP da sequência transcodificada não precisa necessariamente ser igual ao comprimento do GOP da sequência Wyner-Ziv. É possível selecionar, através de parâmetros de entrada, além de quadros *I* e *P*, quadros do tipo *B* na transcodificação. Além disso, é possível selecionar algumas opções de refinamento dos vetores de movimento.

5.3 ESTIMAÇÃO DE MOVIMENTO NO TRANSCODIFICADOR

O codificador H.263 utiliza os vetores de movimento para gerar uma estimativa do quadro que será codificado. Essa estimativa é chamada de quadro compensado. Quanto mais próximo o quadro compensado estiver do quadro original, menor será a energia do resíduo entre eles, e maior será o ganho de codificação. A busca completa por vetores de movimento garante que o melhor quadro compensado será encontrado (segundo a minimização da função de custo usada para realizar a busca, no caso do H.263, a SAD). Outros tipos de buscas podem ser realizados, gerando outras aproximações do quadro que será codificado.

O uso de vetores de movimento diferentes dos ótimos implica no aumento da energia do resíduo, piorando a relação taxa/distorção do codificador. No entanto, o uso de outros vetores de movimento não implica em erro de codificação - o codificador funciona mesmo com vetores de movimento sub-ótimos.

No transcodificador não é realizado um novo processo de busca por vetores de movimento. Ao invés disso, são utilizados os vetores de movimento calculados na geração da informação lateral da decodificação Wyner-Ziv, como visto na seção 4.4.1. O objetivo é aproveitar essa informação de movimento para codificar o quadro Wyner-Ziv F'_{2K+1} como um quadro do tipo *P*.

Naquela seção, para o cálculo dos vetores de movimento MV_F foi utilizado o quadro chave F'_{2K} como referência e o quadro chave F'_{2K+2} como fonte. Da mesma forma que esses vetores foram escalonados para gerar o quadro compensado P_F na geração da informação lateral, utiliza-se os vetores de movimento escalonados para estimar o quadro Wyner-Ziv F'_{2K+1}. Embora esses vetores de movimento não sejam ótimos, eles oferecem uma solução de compromisso para não refazer a busca completa.

Foi codificado o segundo quadro da sequência *Foreman* com $QP = 4$ utilizando três conjuntos de vetores de movimento distintos: os vetores de movimento (MVs) ótimos (gerados por busca completa), os vetores de movimento $\frac{MV_F}{2}$ (calculados na geração da informação lateral da decodificação Wyner-Ziv) e vetores de movimento nulos $(0, 0)$ para todos os macroblocos. A Tabela 5.1 e a Figura 5.4 fazem a comparação entre estes três conjuntos de vetores de movimento. Contrariando as expectativas, para este quadro específico, o uso dos vetores de movimento nulos

Tabela 5.1: Comparação entre conjuntos de vetores de movimento distintos na codificação do segundo quadro da sequência *Foreman* como um quadro tipo *P* com $QP = 4$.

Vetor de Movimento	PSNR (dB)	Taxa (bits)
MVs ótimos	36.06	10024
MVs do transcodificador	35.76	10936
MVs nulos	36.46	15272

gerou um quadro com PSNR maior do que os demais conjuntos de vetores de movimento, embora a uma taxa muito maior, resultando em um pior desempenho em relação a taxa/distorção.

5.4 MUDANÇA DE GOP

O GOP da sequência Wyner-Ziv recebida pelo transcodificador limita quais são as opções para o GOP da sequência transcodificada. Na sequência Wyner-Ziv, existem dois tipos de quadro: os quadros chave e os quadros Wyner-Ziv. No codificador Wyner-Ziv utilizado, os quadros chave são codificados com H.263 *intra*, de modo que seu *bitstream* é idêntico ao *bitstream* caso ele fosse codificado como um quadro *I* no codificador H.263. Já para o quadro Wyner-Ziv, o *bitstream* é totalmente diferente do que pode ser reconhecido por um decodificador H.263. Por esta razão, a decodificação dos quadros Wyner-Ziv é inevitável. No entanto, o trabalho de re-codificar a sequência para H.263 pode ser simplificado.

A primeira idéia é re-aproveitar os *bitstreams* dos quadros chave da sequência Wyner-Ziv como quadros do tipo *I* para a sequência H.263 final. Como os *bitstreams* já estão codificados, eles são apenas copiados para a sequência final, sem nenhuma modificação a não ser os cabeçalhos.

A segunda idéia é nunca transformar um quadro Wyner-Ziv em um quadro *intra*. Como já temos alguma informação de movimento (calculada pelo processo de geração da informação lateral) para este quadro, e não temos *bitstream* algum, é melhor reutilizar os vetores de movimento já calculados e codificar este quadro como um quadro tipo *P* ou *B* do que re-codificá-lo como um quadro tipo *I*.

Dessa forma, se a sequência recebida tem um GOP de comprimento 2, isto é, $IWIWI...$,

(a) Quadro Original

(b) Quadro codificado com vetores de movimento ótimos - PSNR = 36.06 dB.

(c) Quadro codificado com vetores de movimento do transcodificador - PSNR = 35.76 dB.

(d) Quadro codificado com vetores de movimento nulos - PSNR = 36.46 dB.

Figura 5.4: Exemplo da codificação do segundo quadro da sequência *Foreman* como um quadro tipo *P* com $QP = 4$ utilizando conjuntos de vetores de movimento distintos.

a primeira opção de transcodificação é utilizar um GOP de comprimento 2 para a sequência transcodificada, isto é, $IPIPI...$. No entanto, podemos utilizar um GOP maior para a sequência transcodificada. Na verdade, qualquer comprimento de GOP que seja um múltiplo inteiro do GOP original pode ser utilizado (esta restrição ocorre para que quadros Wyner-Ziv não sejam codificados como quadros *intra*). É possível transcodificar a sequência Wyner-Ziv com GOP de comprimento 2 para uma sequência com GOP de comprimento 4 ($IPPPI...$). A forma como são re-utilizados os vetores de movimento é mostrada na Figura 5.5.

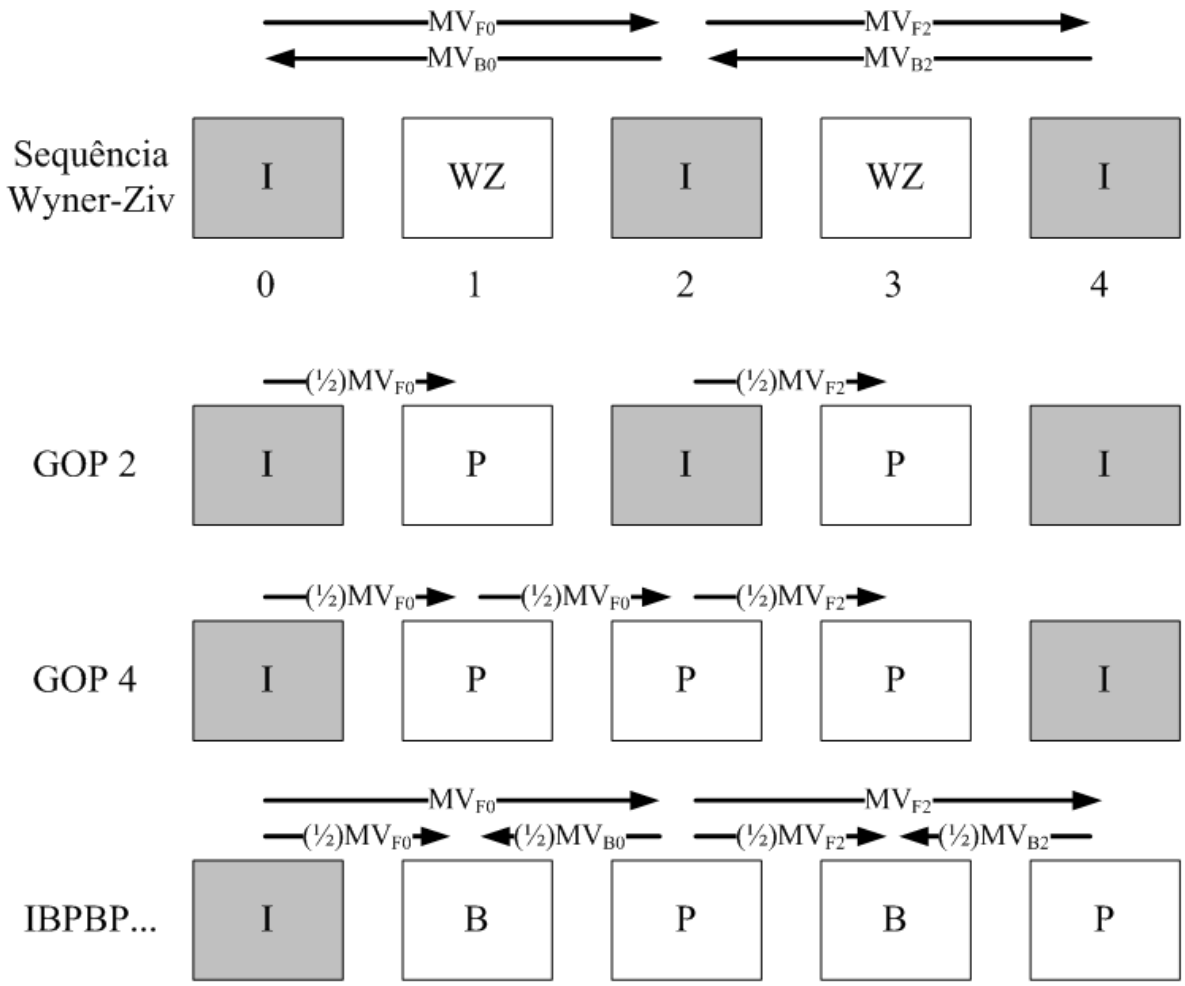


Figura 5.5: Opções do GOP para a sequência transcodificada para o caso onde a sequência Wyner-Ziv original tinha GOP de comprimento 2. MV_{Fk} e MV_{Bk} são os vetores de movimento calculados no processo de geração da informação lateral para o quadro k. As setas indicam como os vetores de movimento são utilizados na codificação de cada quadro *P* ou *B* para cada estrutura de GOP mostrada.

Na Figura 5.5, são mostrados algumas possibilidades de transcodificação quando a sequência Wyner-Ziv recebida possui GOP comprimento 2.

A primeira possibilidade é manter o comprimento do GOP, apenas transformando os quadros Wyner-Ziv em quadros tipo *P*. Neste caso, utilizamos apenas os vetores de movimento MV_F calculados pelo processo de geração da informação lateral, ignorando os vetores de movimento MV_B. Como mostrado na seção 4.4.1, os vetores de movimento MV_F foram calculados utilizando o quadro chave anterior F_{2K} como referência e o quadro chave posterior F_{2K+2} como fonte. Assim, como queremos uma predição para o quadro F_{2K+1}, utilizamos $\frac{MV_F}{2}$.

A segunda possibilidade é expandir o GOP, por exemplo, para 4. Neste caso, alguns quadros chave, originalmente codificados como quadros tipo *I*, serão transformados em quadros tipo *P*. Note que, embora os vetores de movimento MV_{F0} foram calculados utilizando o quadro chave F_{2K+2} como fonte, estes vetores não podem ser utilizados diretamente, pois foram calculados utilizando o quadro F_{2K} como referência, e o H.263 só aceita como referência o quadro imediatamente anterior. Assim, também foram utilizados os vetores $\frac{MV_F}{2}$ para este quadro. Esta maneira pode ser expandida para qualquer outro múltiplo de 2.

Outra possibilidade é transformar o quadro Wyner-Ziv em um quadro tipo *B*. Dessa maneira, a sequência seria transcodificada para $IBPBP...$. Neste caso, os quadros chave seria transformados em quadros *P*, e utilizariam os vetores de movimento MV_F para sua predição. Note que estes vetores foram calculados utilizando o quadro chave anterior. Logo, como os quadros B não podem ser utilizados como referência no H.263 *baseline*, estes vetores de movimento estão de acordo com o que o H.263 espera. Além disso, como o processo de geração da informação lateral utiliza estimação de movimento bidirecional, os vetores de movimento MV_F e MV_B já foram calculados, podendo ser utilizados conforme a Figura 5.5.

Para o caso da sequência Wyner-Zir ter GOP 3, modos similares podem ser utilizados. A Figura 5.6 ilustra algumas opções de transcodificação.

De maneira geral, generalizando para qualquer comprimento de GOP da sequência Wyner-Ziv recebida, caso o processo de geração da informação lateral do decodificador Wyner-Ziv seja similar ao descrito na seção 4.4.1, o transcodificador pode utilizar qualquer GOP de comprimento $k \cdot l$, onde k é um número inteiro e l o comprimento do GOP da sequência recebida.

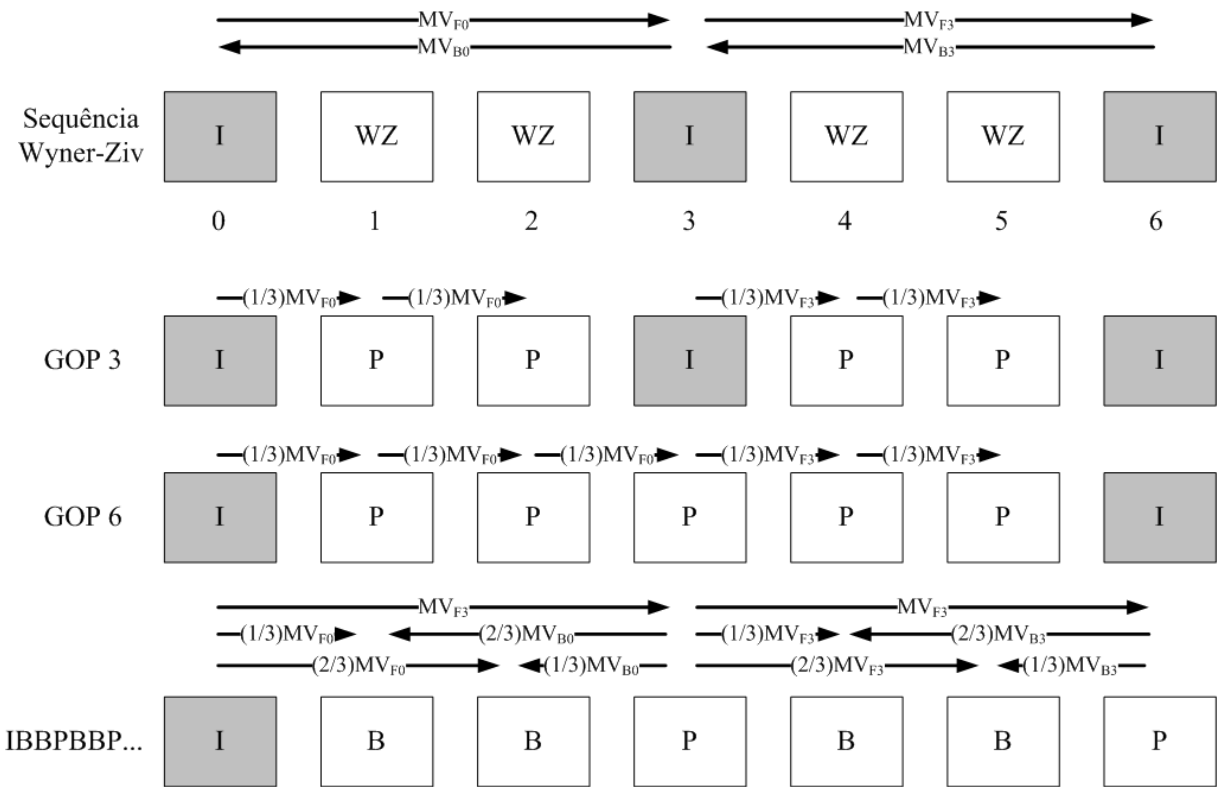


Figura 5.6: Opções do GOP para a sequência transcodificada para o caso onde a sequência Wyner-Ziv original tinha GOP de comprimento 3. MV_{Fk} e MV_{Bk} são os vetores de movimento calculados no processo de geração da informação lateral para o quadro k. As setas indicam como os vetores de movimento são utilizados na codificação de cada quadro *P* ou *B* para cada estrutura de GOP mostrada.

Alternativamente, $l-1$ quadros tipo *B* podem ser inseridos, gerando uma sequência do tipo $I\underbrace{BB...B}_{l-1}P\underbrace{BB...B}_{l-1}P$.

5.5 REFINAMENTO DOS VETORES DE MOVIMENTO

Como visto na seção 5.4, o transcodificador re-utiliza vetores de movimento calculados na decodificação Wyner-Ziv. No entanto, os quadros usados como referência e fonte para calcular os vetores de movimento na decodificação Wyner-Ziv nem sempre são os mesmos usados como referência e fonte para gerar a predição final na transcodificação, razão pela qual os vetores de movimento são escalonados. Isto, porém, não é suficiente, e os vetores de movimento utilizados tem uma defasagem em desempenho se comparados com vetores calculados novamente com busca completa (como visto na Tabela 5.1). A Figura 5.7 mostra o quadro Wyner-Ziv decodificado (Figura 5.7(a)), o quadro compensado utilizando um escalonamento dos vetores de movimento ($\frac{MV_F}{2}$) calculados na decodificação Wyner-Ziv (Figura 5.7(b)) e o quadro compensado re-calculando os vetores de movimento utilizando busca completa (Figura 5.7(c)). Note que na Figura 5.7 as medidas de PSNR tem como referência o quadro Wyner-Ziv decodificado (Figura 5.7(a)) e não o quadro original, pois o quadro que será codificado no transcodificador é o quadro da Figura 5.7(a).

Para não se re-calcular os vetores de movimento utilizando busca completa, sugerimos utilizar métodos de estimação rápida, como a busca em losango [45] ou hexagonal [46]. No entanto, ao invés de iniciar a busca do vetor de movimento na posição $(0,0)$, utilizamos o vetor de movimento que já temos como uma estimativa inicial. Como essa estimativa pode não ser boa, fazemos um teste para verificar se essa estimativa é melhor que o vetor $(0,0)$, e iniciamos a procura no melhor dos dois. Este processo de refinamento é muito menos complexo que a busca completa, e leva a resultados similares, como pode ser observado na Tabela 5.2. Além disso, como a busca é iniciada na posição dada pelo decodificador Wyner-Ziv, se comparada diretamente com a busca em losango iniciada na posição central (ou seja, na mesma posição do macrobloco atual, no vetor de movimento $(0,0)$), o refinamento ainda é menos complexo e oferece um resultado melhor.

A Figura 5.8 ilustra o processo de refinamento com a busca em losango. Na Figura 5.8(a),

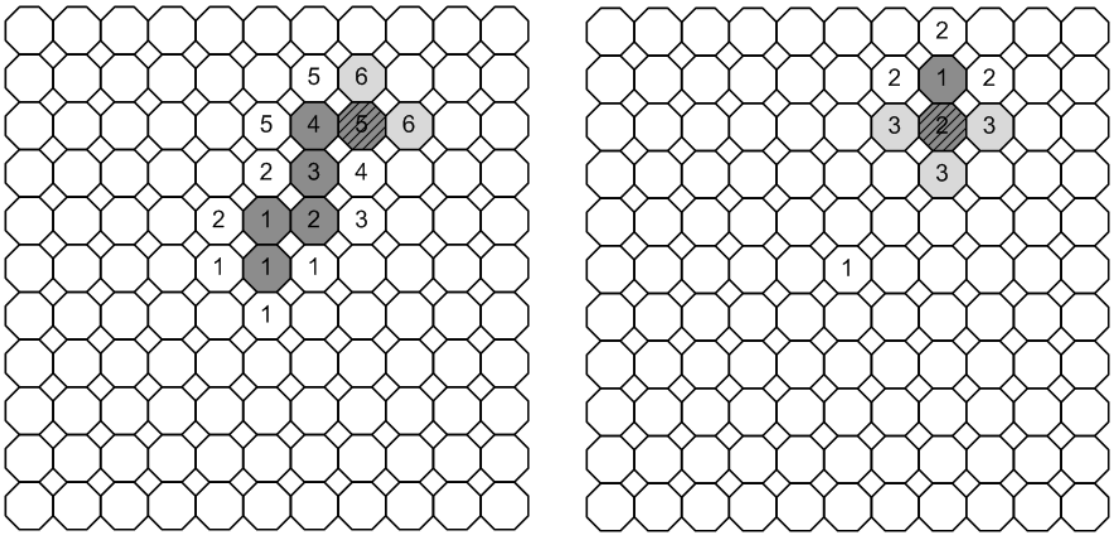
(a) Quadro Wyner-Ziv decodificado

(b) Quadro compensado utilizando os vetores do transcodificador. PSNR = 32.00 dB.

(c) Quadro compensado utilizando vetores calculados com busca completa. PSNR = 33.17 dB.

Figura 5.7: Exemplo da decodificação de um quadro Wyner-Ziv.

é mostrado o processo normal de busca em losango, partindo da posição $(0, 0)$ e finalizando na posição $(2, -3)$. Na Figura 5.8(b), é mostrado o processo de refinamento, partindo da posição $(2, -4)$ e também terminando na posição $(2, -3)$.



(a) Exemplo de busca em losango.

(b) Exemplo de refinamento.

Figura 5.8: Exemplo de Refinamento.

A Tabela 5.2 mostra o número de buscas e a PSNR do quadro compensado para 4 conjuntos de vetores de movimento: os vetores $\frac{MV_F}{2}$ calculados na decodificação Wyner-Ziv, os vetores resultantes de busca completa, os vetores resultantes de busca em diamante e os vetores resultantes do refinamento dos vetores $\frac{MV_F}{2}$. A complexidade é medida em número de blocos procurados (ou seja, em que se calculou a SAD). Como esperado, a busca completa oferece o melhor resultado às custas do maior custo computacional. O uso dos vetores $\frac{MV_F}{2}$ oferece o pior resultado, porém não oferece nenhum custo computacional adicional. Entretanto, o resultado do refinamento de $\frac{MV_F}{2}$ é menos complexo e melhor que o resultado da busca em losango.

Outros tipos de buscas foram implementadas para a opção de refinamento dos vetores de movimento. As buscas em losango, vizinhos, hexagonal/losango e hexagonal/vizinhos foram testadas para realizar o refinamento.

Tabela 5.2: Comparação entre os tipos de estimação de movimento do transcodificador para o segundo quadro da sequência *Foreman* codificada com $QP = 4$.

Tipo de Busca	Complexidade	PSNR (dB)
$\frac{MV_F}{2}$	0	32.00
Busca Completa	77538	33.17
Busca em Losango	710	32.85
Refinamento de $\frac{MV_F}{2}$	512	33.00

5.5.1 Meio-Pixel

Uma outra opção disponível no transcodificador é o uso da estimação de movimento de meio-pixel. No H.263, essa técnica já é prevista e implementada no código de referência da seguinte maneira: o melhor vetor de movimento de precisão inteira é encontrado. Em seguida, os quadros são interpolados e a vizinhança de 8 [53] daquele vetor de movimento (ou seja, apenas mais 8 buscas) são considerados como candidatos, e é escolhido o melhor deles. Esta técnica aumenta pouco a complexidade, mas pode levar a um ganho de desempenho superior a 1 dB.

No transcodificador, somente é permitida a opção de meio-*pixel* se foi realizado o refinamento dos vetores de movimento.

6 EXPERIMENTOS

6.1 INTRODUÇÃO

Neste capítulo serão mostrados resultados de todos os experimentos realizados para avaliar o desempenho do transcodificador. Os experimentos são realizados da seguinte forma: a sequência original é codificada com o codificador Wyner-Ziv. O transcodificador recebe estes dados e transcodifica a mesma sequência utilizando as diferentes opções. Por fim, as sequências transcodificadas são decodificadas usando um decodificador H.263 comum.

O primeiro teste foi feito para decidir qual tipo de refinamento será utilizado. Em seguida, os modos do transcodificador são testados, mostrando os resultados da transcodificação de várias sequências. A qualidade da transcodificação é sempre medida entre a sequência original e a sequência transcodificada (e não entre a sequência transcodificada e a sequência Wyner-Ziv, que de fato serve como entrada para o transcodificador).

6.2 REFINAMENTO DOS VETORES DE MOVIMENTO

Foram testados 4 tipos de refinamento de vetores de movimento: busca em losango, busca nos vizinhos, busca hexagonal/losango (*hexagonal 1*) e busca hexagonal/vizinhos (*hexagonal 2*). Embora a literatura sugira que a busca hexagonal é melhor do que uma busca simples em losango, o teste foi reproduzido pois as condições são diferentes - já temos um vetor de movimento inicial, o objetivo é apenas refiná-lo. O resultado para a sequência *Foreman*, resolução QCIF (176×144), e 15 quadros por segundo pode ser visto na Tabela 6.1. A complexidade é medida com relação à busca completa, contando o número de cálculos de SAD realizados por cada busca (N_{SADs}) e dividindo pelo número de cálculos da busca completa (N_{SADsFS}), segundo a equação 6.1, onde C indica a complexidade que se deseja calcular.

$$C = \frac{N_{SADs}}{N_{SADsFS}}\% \tag{6.1}$$

Como mostrado na Tabela 6.1, os resultados dos quatro tipos de refinamento são muito semelhantes com relação à taxa/distorção. Dessa forma, o critério utilizado para escolher o tipo de busca foi o critério de complexidade. Nesse quesito, a busca em losango é menos complexa, e por isso foi escolhida para todos os demais testes.

6.3 RESULTADOS DO TRANSCODIFICADOR

Os resultados da transcodificação foram agrupados por opções do transcodificador. São mostrados os resultados para as sequências *Foreman*, *Salesman*, *Coastguard* e *CarPhone* para cada opção do transcodificador.

6.3.1 GOP de comprimento 2

A Figura 6.1 mostra o resultado para a primeira opção do transcodificador, quando o comprimento do GOP da sequência Wyner-Ziv é 2 e o comprimento do GOP do transcodificador também é 2. São mostradas 5 curvas nas figuras: o resultado do transcodificador sem refinamento, o resultado do transcodificador trivial (decodificar a sequência Wyner-Ziv e re-codificá-la com um codificador H.263) utilizando busca completa, o resultado do transcodificador com refinamento em losango, o resultado do transcodificador trivial com busca completa e procura em meio-pixel (apenas nos 8 vizinhos do resultado da busca inteira) e o resultado do transcodificador com refinamento em losango mais procura em meio-pixel. Os números ao lado das curvas representam a complexidade do refinamento com relação a busca completa, calculado em número de SADs computadas.

É possível perceber que nas sequências com mais movimento (*Foreman* e *Carphone*), o transcodificador sem refinamento apresenta um desempenho inferior ao transcodificador trivial (sem uso de meio-pixel). Já nas sequências com menos movimento (*Salesman* e *Coastguard*), o desempenho do transcodificador sem refinamento é mais próximo do transcodificador trivial (sem uso de meio-pixel). No entanto, em todos os casos o uso do refinamento aproxima o transcodificador proposto do transcodificador trivial, e ainda apresentando uma complexidade muito menor.

Tabela 6.1: Teste do Refinamento de Vetores de Movimento. Foram codificados 121 quadros da sequência *Foreman* usando diferentes tipos de refinamento partindo dos mesmo vetores de movimento (calculados na decodificação Wyner-Ziv).

Codificação com $QP = 7$.

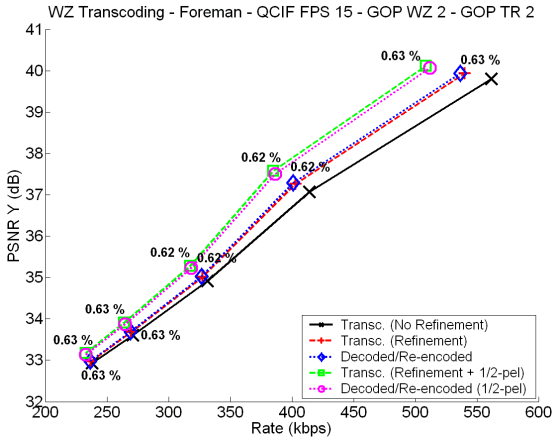
Tipo	Taxa (kbps)	PSNR (dB)	Complexidade (%)
Losango	237.01	32.97	0.63
Vizinhos	236.79	32.97	1.03
Hexagonal 1	237.00	32.97	1.25
Hexagonal 2	236.86	32.97	1.66

Codificação com $QP = 5$.

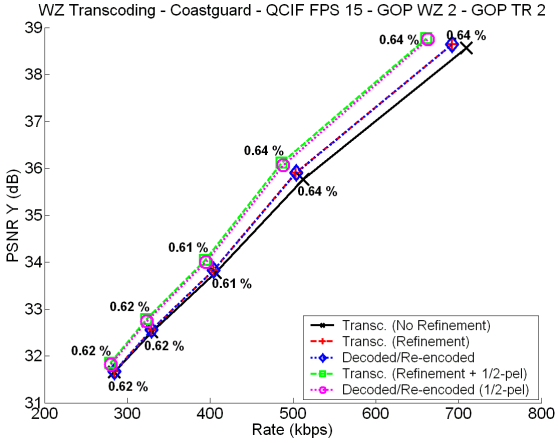
Tipo	Taxa (kbps)	PSNR (dB)	Complexidade (%)
Losango	327.75	35.01	0.62
Vizinhos	327.31	35.01	1.03
Hexagonal 1	327.78	35.01	1.24
Hexagonal 2	327.48	35.01	1.65

Codificação com $QP = 3$.

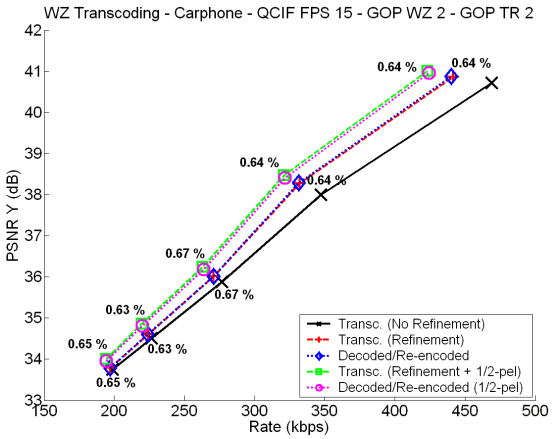
Tipo	Taxa (kbps)	PSNR (dB)	Complexidade (%)
Losango	540.02	39.93	0.63
Vizinhos	537.61	39.93	1.04
Hexagonal 1	540.92	39.93	1.24
Hexagonal 2	538.92	39.93	1.65



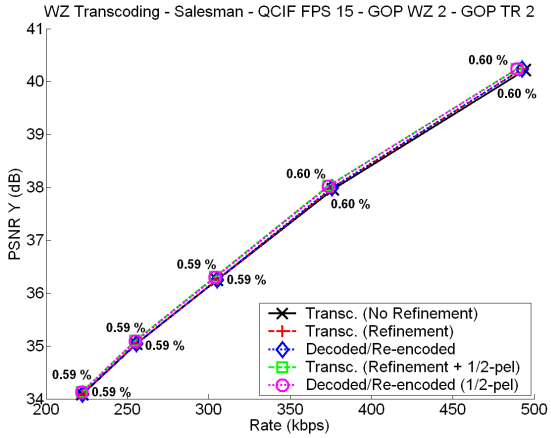
(a) Sequência Foreman.



(b) Sequência Coastguard.



(c) Sequência Carphone.

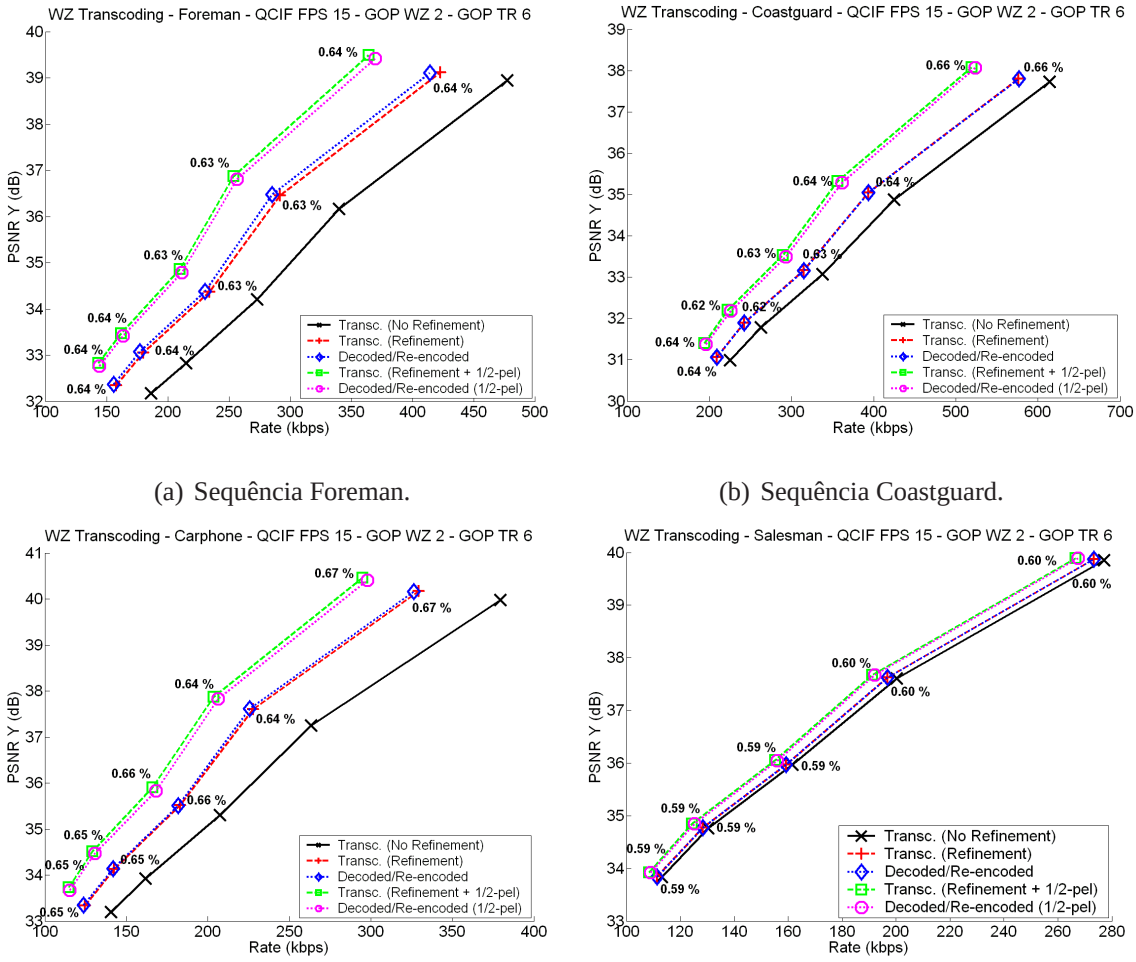


(d) Sequência Salesman.

Figura 6.1: Resultado da transcodificação para um GOP de comprimento 2.

6.3.2 GOP de comprimento 6

Como dito, o transcodificador pode optar por mudar o comprimento do GOP da sequência transcodificada. A única restrição é que o novo comprimento de GOP seja um múltiplo do comprimento do GOP original (essa restrição existe para que nenhum quadro Wyner-Ziv seja transcodificada para um quadro *intra*). Os mesmos tipos de curvas mostrados na seção 6.3.1 são mostrados aqui, porém dessa vez mudando o comprimento do GOP da sequência transcodificada de 2 para 6. Os resultados são mostrados na Figura 6.2.



(a) Sequência Foreman.

(b) Sequência Coastguard.

(c) Sequência Carphone.

(d) Sequência Salesman.

Figura 6.2: Resultado da transcodificação para um GOP de comprimento 6.

Neste caso, o desempenho do transcodificador proposto sem refinamento dos vetores de movimento fica aquém do desempenho do transcodificador trivial (sem estimação em meio-pixel). Esse desempenho é mais evidente nas sequências com mais movimento (*Foreman* e *CarPhone*) do que nas sequências com menos movimento (*Salesman* e *Coastguard*). No entanto, em todas

as sequências o uso do refinamento dos vetores de movimento faz com que o transcodificador proposto tenha desempenho semelhante ao transcodificador trivial (usando ou não estimação em meio-pixel), porém deve ser notado que a complexidade do transcodificador proposto é menor que a do transcodificador trivial.

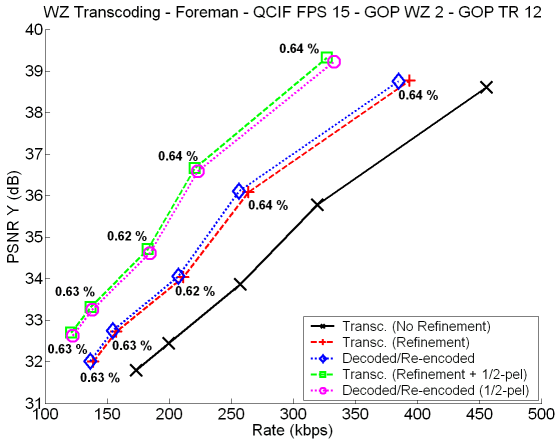
6.3.3 GOP de comprimento 12

Aumentando ainda mais o comprimento do GOP da sequência transcodificada, podemos notar uma tendência no desempenho do transcodificador. A figura 6.3 mostra os resultados quando a sequência original tinha um comprimento de GOP de 2 e a sequência transcodificada tem um comprimento de GOP de 12. Ao aumentar a distância entre os quadros tipo *I*, o transcodificador proposto sem refinamento apresenta um desempenho inferior com relação ao transcodificador trivial (sem estimação de meio-*pixel*). No entanto, podemos ver que o uso do refinamento faz com que o transcodificador proposto apresente resultado semelhante ao transcodificador trivial, tanto no caso com estimação em meio-*pixel* quanto no caso sem essa estimação. Em todos os casos, o refinamento é muito menos custoso computacionalmente do que uma nova busca completa.

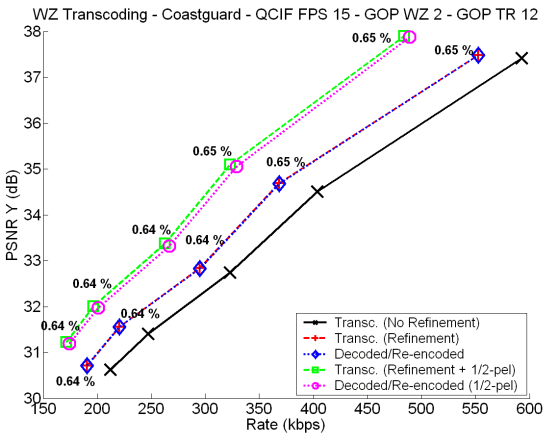
6.3.4 Utilizando quadros tipo *B*

Uma outra opção do transcodificador é utilizar quadros do tipo *B*. Como o processo de geração da informação lateral utiliza estimação de movimento bi-direcional, é natural que queiramos utilizar todos os vetores de movimento na busca do melhor deles. Este modo do transcodificador permite isso, e ainda oferece uma vantagem adicional: a estimação de movimento para os quadros do tipo *P* é melhor aproveitada, pois utiliza na codificação os mesmos quadros de fonte e referência que foram utilizados na busca dos vetores de movimento na decodificação Wyner-Ziv.

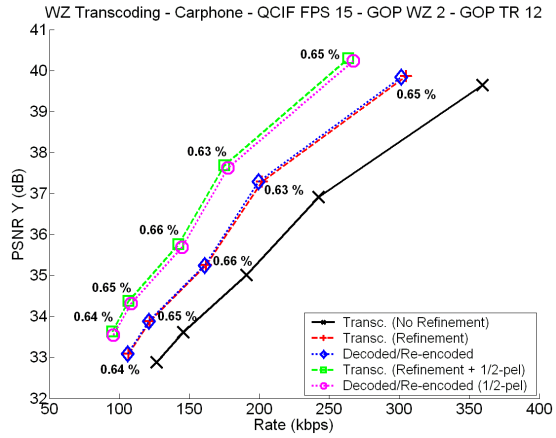
Na Figura 6.4 são mostrados os resultados quando a sequência original tinha um GOP de comprimento 2 e a sequência transcodificada tem um GOP tipo *IBPBP...*. Esta opção do transcodificador apresenta resultados interessantes: o desempenho do transcodificador proposto sem refinamento é muito próximo ao desempenho do transcodificador trivial (sem estimação de meio-*pixel*) para todas as sequências, mesmo as que apresentam mais movimento (*Foreman* e *CarPhone*). Embora o transcodificador proposto sem refinamento não re-calcule nenhum vetor



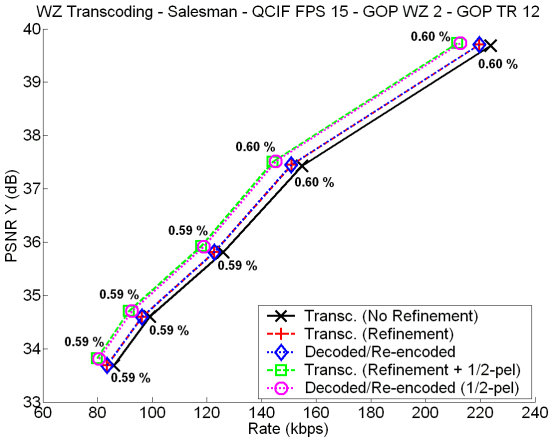
(a) Sequência Foreman.



(b) Sequência Coastguard.

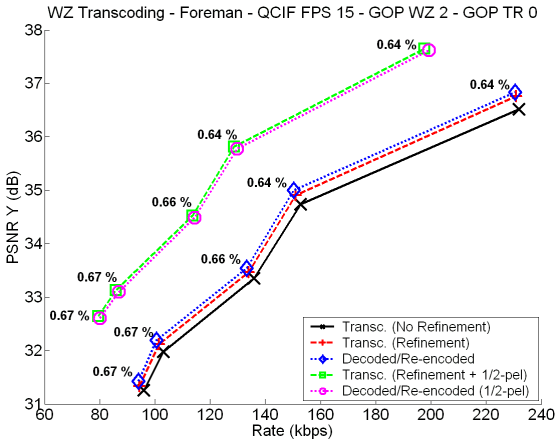


(c) Sequência Carphone.

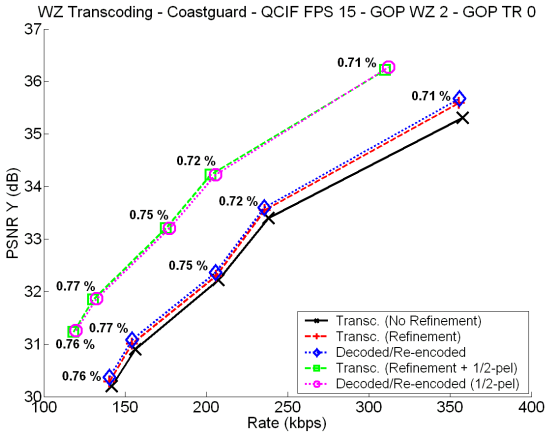


(d) Sequência Salesman.

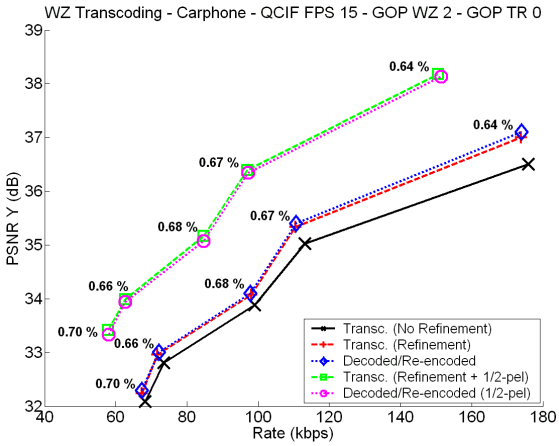
Figura 6.3: Resultado da transcodificação para um GOP de comprimento 12.



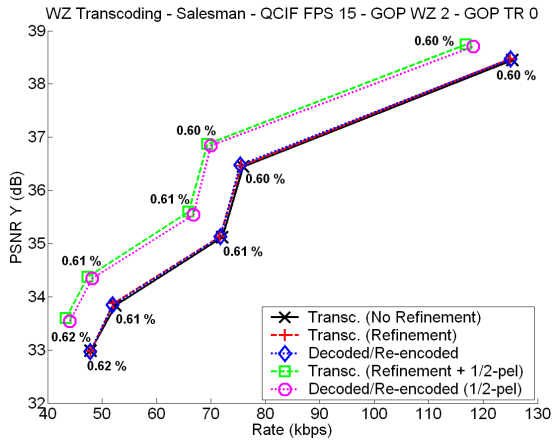
(a) Sequência Foreman.



(b) Sequência Coastguard.



(c) Sequência Carphone.



(d) Sequência Salesman.

Figura 6.4: Resultado da transcodificação para um GOP tipo *IBPBP*....

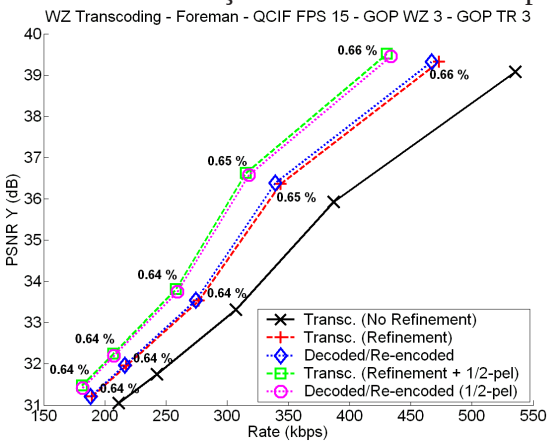
de movimento, ele sempre tem 2 opções para cada macrobloco em um quadro Wyner-Ziv, que será codificado como um quadro tipo *B*. Embora o uso da estimação de meio-*pixel* apresente um desempenho superior, o uso do refinamento com este tipo de estimação apresenta desempenho semelhante ao transcodificador trivial utilizando também esta técnica, com a vantagem de apresentar menor complexidade.

6.3.5 Resultados para um GOP original de comprimento 3

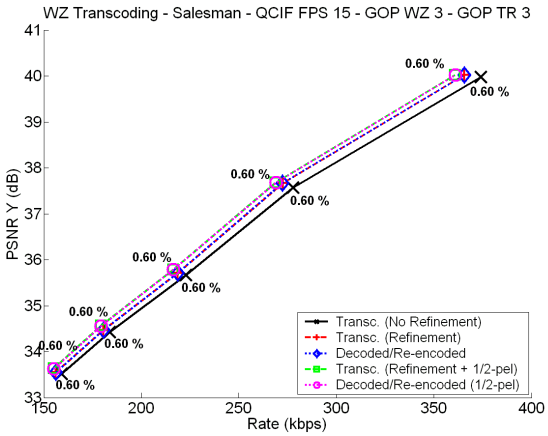
O codificador Wyner-Ziv utilizado também pode utilizar um GOP de comprimento 3 (*IWWIWW...*), embora o desempenho não melhore tanto quanto um codificador tradicional quando se aumenta o GOP (o fato dos quadros chave estarem mais distantes entre si compromete a qualidade da informaçã lateral, o que influencia diretamente no desempenho do codificador Wyner-Ziv). Neste caso, a restrição para o comprimento do GOP da sequência transcodificada permanece o mesmo: ele deve ser um múltiplo do comprimento do GOP da sequência original. Assim, podemos utilizar na transcodificação um GOP de 3, 6, etc.. O desempenho do transcodificador quando a sequência original possui um GOP de comprimento 3 é mostrado na Figura 6.5 para várias opções de transcodificação.

Os resultados mostrados na Figura 6.5 se mostram consistentes com os resultados mostrados nas seções anteriores. Na transcodificação para um GOP similar ao original, o transcodificador proposto sem refinamento apresenta um desempenho próximo ao do transcodificador trivial (sem estimação de meio-*pixel*). À medida que o comprimento do GOP da sequência transcodificada aumenta, aumenta também a distância entre o transcodificador proposto sem refinamento e o transcodificador trivial. em todos os casos, porém, o uso do refinamento aproxima o desempenho do transcodificador proposto com relação ao transcodificador trivial, tanto para o caso em que se usa a estimação de meio-*pixel* quanto para o caso em que não se usa essa estimação. Novamente, porém, o melhor resultado para o transcodificador proposto sem refinamento é quando se utilizam quadros do tipo *B*, formando uma sequência com estrutura *IBBPBBP...*.

Transcodificação de um GOP de comprimento 3 para um GOP de comprimento 3.

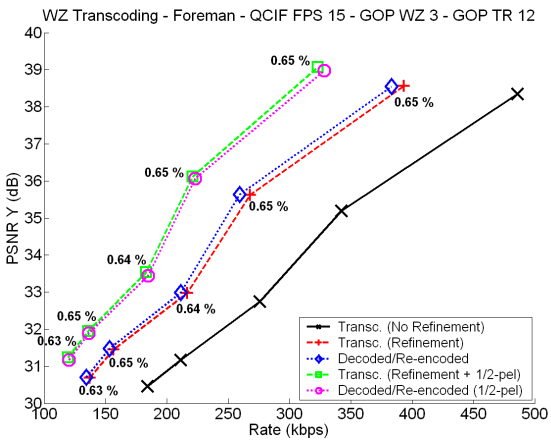


(a) Sequência Foreman.

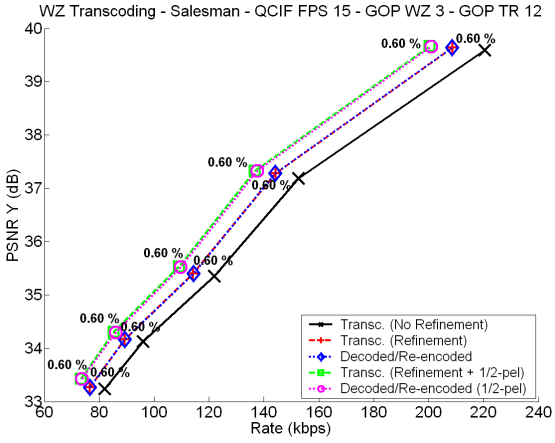


(b) Sequência Salesman.

Transcodificação de um GOP de comprimento 3 para um GOP de comprimento 12.

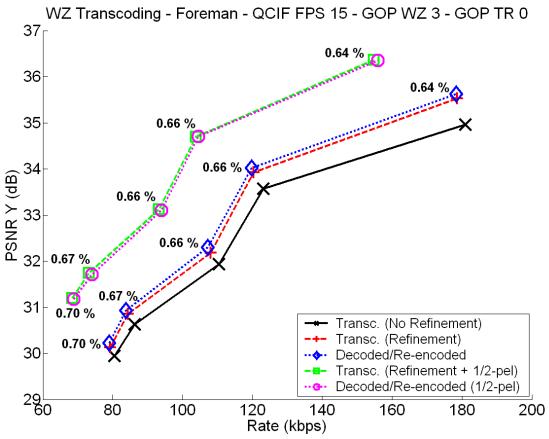


(c) Sequência Foreman.

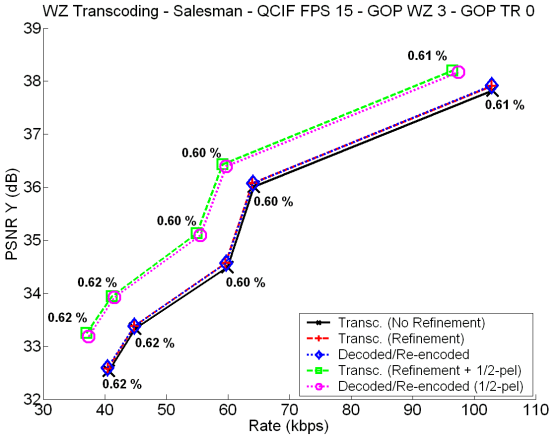


(d) Sequência Salesman.

Transcodificação de um GOP de comprimento 3 para um GOP tipo *IBBPBBP*....



(e) Sequência Foreman.



(f) Sequência Salesman.

Figura 6.5: Resultado da transcodificação para um GOP original de comprimento 3.

6.4 ANÁLISE DE COMPLEXIDADE

O transcodificador possui flexibilidade em sua complexidade, podendo ser adaptado às diversas situações. A complexidade do transcodificador trivial é a complexidade do decodificador Wyner-Ziv somada à complexidade de um codificador H.263 (que realiza estimação de movimento através de busca completa). Como o decodificador Wyner-Ziv é parte do transcodificador proposto (todas as operações do decodificador Wyner-Ziv estão presentes no transcodificador), a redução de complexidade é obtida minimizando as operações feitas pelo codificador H.263. Na re-codificação, utiliza-se informações obtidas na decodificação Wyner-Ziv, como os vetores de movimento, além dos *bitstreams* dos quadros que já haviam sido codificados com um codificador H.263 (os quadros chave da sequência Wyner-Ziv foram codificados utilizando um codificador H.263 *intra*).

No caso mais simples do transcodificador, onde a sequência Wyner-Ziv recebida tem um GOP de comprimento 2 e a sequência transcodificada também tem um GOP de 2, temos a seguinte situação para a transcodificação: os *bitstreams* dos quadros do tipo *I* são simplesmente copiados para a saída, não havendo nenhuma outra operação computacional para estes quadros. Caso se opte por não utilizar o refinamento dos vetores de movimento, a transcodificação dos quadros Wyner-Ziv irá envolver apenas a compensação de movimento, geração do resíduo, transformada DCT e codificação de entropia. Ou seja, o transcodificador terá uma complexidade similar à complexidade de um quadro *intra*. Assim, a complexidade para esta opção é a complexidade do decodificador Wyner-Ziv somada à complexidade de se codificar aproximadamente 1 quadro *intra* a cada 2 quadros. Como a maior parte da complexidade computacional de um codificador de vídeo se encontra na codificação de quadros com predição temporal [5, 7] (isto é, quadros do tipo *P* ou *B*), conclui-se que a complexidade neste modo é mínima. O uso do refinamento implica em refazer o processo de estimação de movimento, porém, como pode ser visto nas Figuras de resultado, a complexidade do refinamento é $0.6-0.75\%$ da complexidade de uma busca completa.

Para as outras opções do transcodificador, onde a sequência Wyner-Ziv tem um GOP de comprimento 2 e a sequência transcodificada tem comprimentos maiores de GOP, a complexidade do transcodificador é diferente. A decodificação Wyner-Ziv ainda estará presente no transcodificador. No entanto, apenas 1 quadro intra a cada GOP é re-aproveitado. Os demais quadros deste GOP

tem complexidade semelhante à complexidade de um quadro *intra*, para o caso onde não há refinamento dos vetores de movimento. Dessa forma, para o caso da transcodificação para um GOP de comprimento n, a complexidade do transcodificador é a complexidade do decodificador Wyner-Ziv somada à complexidade de se codificar $n-1$ quadros *intra* a cada n quadros. Deve-se notar que, embora o desempenho do transcodificador proposto nestes modos quando não é utilizado refinamento dos vetores de movimento seja inferior ao desempenho do transcodificador trivial, a complexidade diminui ainda mais devido à mudança de GOP, pois mais quadros devem ser preditos utilizando estimação de movimento.

Na última opção do transcodificador, onde são utilizados quadros do tipo B, a análise é também semelhante. Apenas a *bitstream* do primeiro quadro é re-aproveitada, pois os demais quadros do tipo *I* serão transformados para quadros do tipo *P*. Assim, a complexidade do transcodificador neste modo, sem refinamento, seria a complexidade do decodificador Wyner-Ziv somada à complexidade de se codificar os demais quadros como quadros tipo *I*. É importante notar, no entanto, que o desempenho do transcodificador proposto neste modo é muito similar ao desempenho do transcodificador trivial, mesmo quando não é utilizado refinamento dos vetores de movimento.

7 CONCLUSÕES

7.1 SUMÁRIO DO TRABALHO

O objetivo deste trabalho é propor um esquema de comunicação de vídeo que apresentasse baixa complexidade nos dois terminais, isto é, no remetente e no destinatário. A solução encontrada foi aliar dois codificadores diferentes, um que apresenta baixa complexidade do lado do codificador e outro que apresenta baixa complexidade do lado do decodificador, e construir um transcodificador entre eles que seria inserido na rede. O transcodificador concentra a complexidade da comunicação em um ambiente controlado e preparado para isso.

Para atingir esse objetivo, foi implementado um codificador de vídeo baseado no paradigma de codificação de vídeo distribuída, a partir de trabalhos publicados por outros pesquisadores. O codificador implementado é um codificador Wyner-Ziv no domínio dos *pixels*, que apresenta baixa complexidade de codificação pois não utiliza estimação de movimento na codificação, e especificamente para os quadros Wyner-Ziv não utiliza transformada DCT.

Utilizando o codificador Wyner-Ziv implementado e a versão modificada do código de referência do padrão H.263, foi desenvolvido um transcodificador que recebe uma sequência codificada com o codificador Wyner-Ziv e a transforma em uma sequência compatível com o padrão H.263. O transcodificador proposto deveria ser melhor que o transcodificador trivial em termos de complexidade, e ao mesmo tempo apresentar um desempenho semelhante. Outro requerimento era obter certa escalonabilidade em termos de complexidade e desempenho, tornando o transcodificador mais flexível.

7.2 APRESENTAÇÃO DA CONTRIBUIÇÃO DO MESTRADO

Neste trabalho foi implementado um codificador Wyner-Ziv baseado no trabalho de outros pesquisadores. Com exceção do algoritmo para codificação LDPCA, todos os demais algoritmos foram implementados, validados e aperfeiçoados neste trabalho, utilizando diversas metodologias

apresentas nos capítulos anteriores.

Também foi modificado o código de referência do H.263 de forma que ele pudesse ser utilizado no transcodificador. Foram adicionados módulos para ler vetores de movimento advindos da decodificação Wyner-Ziv e módulos de estimação de movimento rápida usados para refinar estes vetores de movimento. Também foram adicionadoas funções para calcular o número de operações realizadas por estes algoritmos, de forma a estimar sua complexidade. Além disso, foi habilitada uma opção para ligar e desligar a estimação de movimento com precisão de meio-*pixel*.

A codificação distribuída de vídeo é uma área nova, baseada em um paradigma radicalmente diferente da codificação de vídeo tradicional. A construção de codificadores baseados nesse paradigma está ainda em desenvolvimento, não existindo nenhum padrão de codificação de vídeo baseado nesse paradigma ainda. Embora a idéia do transcodificador não seja nova [24, 8], não foi encontrada na literatura nenhuma implementação de um transcodificador que desempenhasse um papel similar ao transcodificador proposto [52]. A contribuição principal do trabalho se insere nessa nova área.

7.3 CONCLUSÕES DO TRABALHO

O transcodificador proposto apresenta um desempenho comparável ao transcodificador trivial, porém com uma menor complexidade, alcançando o objetivo estipulado. Além disso, o transcodificador proposto possui modos ajustáveis a diferentes requerimentos de complexidade, sendo capaz de reduzir a taxa da sequência transcodificada modificando a estrutura do GOP dessa sequência, além de apresentar uma opção de refinamento dos vetores de movimento, adquirindo assim a capacidade de ajuste de complexidade e desempenho requerida.

7.4 PERSPECTIVAS PARA TRABALHOS FUTUROS

Existem algumas alternativas que podem ser exploradas para aprofundar o trabalho. O desenvolvimento de um transcodificador Wyner-Ziv para o estado da arte em codificação de vídeo

H.264/AVC é uma das opções. Alguns modos do H.264/AVC podem melhorar o desempenho do transcodificador, como sua habilidade de definir qualquer quadro como referência para estimação de movimento. Além disso, pode-se implementar um transcodificador que utiliza como codificador Wyner-Ziv outro tipo de codificador ao invés do codificador no domínio dos *pixels* utilizado, como o codificador baseado em redução da resolução espacial [11].

Embora não tenha sido o foco deste trabalho, foi verificado que o aperfeiçoamento do estágio de geração de informação lateral melhora significativamente o desempenho do codificador Wyner-Ziv utilizado. O melhor entendimento do canal de correlação que conecta a informação que se deseja transmitir à sua informação lateral disponível no decodificador também pode melhorar o desempenho do codificador Wyner-Ziv. Utilizar um codificador de canal que não necessite de canal de retorno (utilizando algum estimador da correlação entre a fonte e sua informação lateral no codificador, por exemplo) pode diminuir a complexidade da implementação da solução proposta.

Uma vez que o codificador H.263 utilizado é um padrão da indústria e o *bitstream* da sequência transcodificada deve ser compatível com qualquer decodificador do padrão H.263, as opções na geração deste *bitstream* ficam limitadas. Uma outra alternativa seria o desenvolvimento de um codificador Wyner-Ziv mais apropriado à transcodificação. Como nenhum codificador Wyner-Ziv foi padronizado até o momento, a escolha do codificador Wyner-Ziv ainda é livre, fazendo desta uma área promissora para melhorar o desempenho do transcodificador.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ITU-T. *ITU-T Recommendation H.263, Video coding for low bit rate communication*. [S.l.], March 1996.

[2] ITU-T. *ITU-T Recommendation H.264, Advanced Video Coding for Generic Audiovisual Services*. [S.l.], May 2003.

[3] ISO/IEC. *ISO/IEC 13818 - Generic Coding of Moving Pictures and Associated Audio Information*. [S.l.], 1995.

[4] ISO/IEC. *ISO/IEC 14496-2, Coding of Audio-Visual Objects - Part 2:Visual*. [S.l.], 2001.

[5] RAO, K. R.; HWANG, J. J. *Techniques and Standards for Image, Video and Audio Coding*. [S.l.]: Prentice Hall PTR, 1996.

[6] GIBSON, J. D. et al. *Digital Compression for Multimedia: Principles and Standards*. [S.l.]: Morgan Kaufmann Publishers, 2006.

[7] RICHARDSON, I. E. G. *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*. [S.l.]: Wiley, 2003.

[8] GIROD, B. et al. Distributed video coding. In: *PROC. IEEE SPECIAL ISSUE ON ADVANCES IN VIDEO CODING AND DELIVERY*. [S.l.: s.n.], 2005. p. 1–12.

[9] SLEPIAN, J. D.; WOLF, J. K. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, IT-19, p. 471–480, July 1973.

[10] WYNER, A. D.; ZIV, J. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, IT-22, n. 1, p. 1–10, January 1976.

[11] MUKHERJEE, D.; MACCHIAVELLO, B.; QUEIROZ, R. L. de. A simple reversed-complexity Wyner-Ziv video coding mode based on a spatial reduction framework. *Proc. SPIE Visual Communication and Image Processing*, v. 6508, p. 65081Y1–65081Y12, January 2007.

[12] PURI, R.; RAMCHANDRAN, K. Prism: An uplink-friendly multimedia coding paradigm. *Proc. International Conference on Acoustics, Speech and Signal Processing*, v. 4, p. 856–859, April 2003.

[13] AARON, A.; ZHANG, R.; GIROD, B. Wyner-Ziv coding of motion video. *Proc. Asilomar Conference on Signals and Systems*, November 2002.

[14] H, C. T.; LEISERSON, C. E.; RIVEST, R. L. *Introduction to Algorithms*. [S.l.]: McGraw-Hill, 1999.

[15] WYNER, A. D. Recent results in the shannon theory. *IEEE Transactions on Information Theory*, v. 20, n. 1, p. 2–10, January 1974.

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

[17] SHANNON, C. E. A mathematical theory of communication. *Bell System Technical Journal*, v. 27, n. 17, p. 379–423,623–656, July and October 1948.

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

[19] ZAMIR, R. The rate loss in the Wyner-Ziv problem. *IEEE Transactions on Information Theory*, v. 42, n. 6, p. 2073–2084, November 1996.

[20] LIN, S.; COSTELLO, D. J. *Error Control Coding*. [S.l.]: Prentice Hall, 2004.

[21] ZAMIR, R.; SHAMAI, S. Nested linear/lattice codes for Wyner-Ziv encoding. *Proc. Information Theory Workshop*, p. 92–93, June 1998.

[22] ZAMIR, R.; SHAMAI, S.; EREZ, U. Nested linear/lattice codes for structured multiterminal binning. *IEEE Transactions on Information Theory*, v. 48, n. 6, p. 1250–1276, June 2002.

[23] PURI, R.; RAMCHANDRAN, K. Prism: A new robust video coding architecture based on distributed compression principles. *Proc. Allerton Conference on Communication, Control and Computing*, October 2002.

[24] PURI, R.; RAMCHANDRAN, K. Prism: A ‘reversed’ multimedia coding paradigm. *Proc. IEEE International Conference on Image Processing*, v. 1, p. 617–620, September 2003.

[25] AARON, A.; SETTON, E.; GIROD, B. Towards practical Wyner-Ziv coding of video. *Proc. IEEE International Conference on Image Processing*, v. 3, p. 869–872, September 2003.

[26] AARON, A. et al. Transform-domain Wyner-Ziv codec for video. *Proc. SPIE Visual Communication and Image Processing*, v. 5308, p. 520–528, January 2004.

[27] XU, Q.; XIONG, Z. Layered Wyner-Ziv video coding. *Proc. SPIE Visual Communication and Image Processing*, v. 5308, p. 83–91, January 2004.

[28] WANG, H.; CHEUNG, N.-M.; ORTEGA, A. A framework for adaptive scalable video coding using Wyner-Ziv techniques. *EURASIP Journal of Applied Signal Processing*, n. 1, p. 1–18, January 2006.

[29] SCHLEGEL, C. B.; PéREZ, L. C. *Trellis and Turbo Coding*. [S.l.]: Wiley Interscience, 2004.

[30] ROWITCH, D.; MILSTEIN, L. On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo codes. *IEEE Transactions on Communications*, v. 48, n. 6, p. 948–959, June 2000.

[31] AARON, A.; VARODAYAN, D.; GIROD, B. Wyner-Ziv residual coding of video. *Proc. International Picture Coding Symposium*, April 2006.

[32] AARON, A.; VARODAYAN, D.; GIROD, B. Rate-adaptive distributed source coding using low-density parity-check codes. *Proc. Asilomar Conference on Signals, Systems and Computing*, November 2005.

[33] AARON, A.; GIROD, B. Compression with side information using turbo codes. *Proc. IEEE Data Compression Conference*, p. 252–261, April 2002.

[34] ITU-T. *ITU-T Recommendation H.263, Video coding for low bit rate communication*. [S.l.], February 1998.

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

[36] COTé, G. et al. H.263+: Video coding at low bit rates. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 8, n. 7, p. 849–866, November 1998.

[37] 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.

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

[39] RICHARDSON, I. E. G. *Video Codec Design: Developing Image and Video Compression Systems*. [S.l.]: Wiley, 2002.

[40] 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.

[41] JAIN, J. R.; JAIN, A. K. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications*, COM-29, p. 1799–1808, December 1981.

[42] SULLIVAN, G. J.; WIEGAND, T. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, p. 74–90, November 1998.

[43] LI, R.; ZENG, B.; LIOU, M. L. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 4, p. 438–442, August 1994.

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

[45] GALANT, M.; CôTé, G.; KOSSENTINI, F. An efficient computation-constrained block-based motion estimation algorithm for low bit rate video coding. *IEEE Transactions on Image Processing*, v. 8, n. 12, p. 1816–1823, December 1999.

[46] ZHU, C. et al. A novel hexagon-based search algorithm for fast block motion estimation. *Proc. IEEE Internetional Conference on Acoustics, Speech and Signal Processing*, p. 1593–1596, May 2001.

[47] AHMED, N.; NATARAJAN, T.; RAO, K. R. Discrete cosine transform. *IEEE Transactions on Computers*, p. 90–93, January 1974.

[48] LI, Z.; DELP, E. J. Wyner-Ziv side estimator: Conventional motion search methods revisited. *International Conference on Image Processing*, v. 1, n. 1, p. 825–828, September 2005.

[49] VARODAYAN, D.; GIROD, B. *Rate-Adaptive LDPC Accumulate Codes for Distributed Source Coding*. http://www.stanford.edu/~divad/ldpca.html, Acessado em 23/01/2008 2006.

[50] MUKHERJEE, D. *Optimal parameter choice for Wyner-Ziv coding of laplacian sources with decoder side-information*. [S.l.], 2007.

[51] KOMO, J. J. *Random Signal Analysis in Engineering Systems*. [S.l.]: Academic Press, 1987.

[52] PEIXOTO, E.; QUEIROZ, R. L.; MUKHERJEE, D. Mobile video communications using a Wyner-Ziv transcoder. *Proc. SPIE Visual Communication and Image Processing*, v. 2008, January.

[53] GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. [S.l.]: Addison-Wesley Publishing, 1992.