CODIFICAÇÃO DISTRIBUÍDA DE VÍDEO

Danillo Bracco Graziosi

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

Aprovada por:

_____

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

_____

Prof. Gelson Vieira de Mendonça, Ph.D.

_____

Prof. Weiler Alves Finamore, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2006

GRAZIOSI, DANILLO BRACCO

Codificação distribuída de vídeo [Rio de Janeiro] 2006

xi, 91 pp 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2006)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1.Compressão de Vídeo

2.Wyner-Ziv

3.Quantização por aproximações sucessivas

I.COPPE/UFRJ II.Título (série)

Agradecimentos

Agradeço a todas as pessoas que contribuíram diretamente e indiretamente para a conclusão desta tese. Um agradecimento especial merecem algumas pessoas, que sem a contribuição delas esta tese não teria se tornado realidade, e por isso dedico a minha tese a essas pessoas.

Ao pessoal do LPS (Michel, Tadeu, Leonardo, Ana, e todos os demais), que me ajudaram com valiosas dicas de LaTeX, fornecendo exemplos e sempre dispostos a sanar qualquer dúvida que eu tivesse.

Ao colega José Fernando, que foi peça fundamental desta tese, me passando o código turbo e auxiliando-me em diversas outras ocasiões.

Ao meu amigo Daniel, que me ensinou a programar. Seu companheirismo e apoio foram sempre constantes, e por tudo o que ele fez para me ajudar, serei eternamente grato.

À minha família, que esteve sempre do meu lado, me apoiando em todos as minhas decisões.

Ao amor da minha vida, minha esposa Flávia, que soube suportar os maus momentos, e tentou de várias maneiras me auxiliar, estando sempre do meu lado.

Ao meu orientador Eduardo, que foi mais do que a definição de orientador poderia explicar, sobrepondo grandes distâncias para sempre me guiar nessa jornada. Obrigado pelas horas de conversa, pelas dicas valiosas e pelas críticas que vc soube fazer.

A Deus.

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

CODIFICAÇÃO DISTRIBUÍDA DE VÍDEO

Danillo Bracco Graziosi

Março/2006

Orientador: Eduardo Antônio Barros da Silva

Programa de Engenharia Elétrica

Ultimamente a comunidade científica tem demonstrado um crescente interesse em técnicas de codificação distribuídas. Remetendo a resultados publicados por Slepian e Wolf [8] e Wyner e Ziv [10], grupos independentes publicaram resultados práticos que se aproximam bastante dos declarados na teoria, e com implicações interessantes, como a robustez de transmissão e a simplificação do codificador.

Nesta tese, iremos analisar este cenário, esclarecendo algumas implementações. Ao final, iremos propor uma técnica alternativa de codificação distribuída baseada no estado-da-arte em codificação de imagens, usando quantização vetorial por aproximações sucessivas.

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

DISTRIBUTED VIDEO CODING

Danillo Bracco Graziosi

March/2006

Advisor: Eduardo Antônio Barros da Silva

Department: Electrical Engineering

Lately there has been a grown interest of the scientific community in distributed coding techniques. Based on results published by Slepian and Wolf [8] and Wyner and Ziv [10], independent groups achieved practical results, that come very close to the theory, with very interesting applications, like robust transmission and simplification of the encoder.

In this thesis, we will analyze this scenario, clarifying some implementations. At the end, we will propose an alternative distributed coding technique based on the state-of-art image coding, using successive approximation vector quantization.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivações

As técnicas de codificação de vídeo dos dias de hoje exploram a redundância das fontes de sinal para conseguirem atingir taxas de compressão, que a cada dia que passam, surpreendem pelo seu ganho e desempenho. Padrões de codificação de vídeo como o bem conhecido MPEG, assim como padrões novos como o H.264 utilizam técnicas rebuscadas para diminuir o número de bits necessários para transmitir uma seqüência de imagens. Porém estes codificadores funcionam bem num ambiente onde o codificador tem recursos e uma grande capacidade computacional, no qual é necessário apenas codificar uma vez, para depois decodificar várias vezes. Exemplos de situações como a descrita anteriormente são o uso dos codificadores de vídeo nos DVDs, ou uma transmissão de vídeo em radio-difusão.

Porém, com o surgimento da TV Digital, fica cada dia mais evidente que a transmissão de imagens não se limitará as telas de cinema ou a televisão da sala. A evolução dos aparelhos celulares e PDAs mostra que as imagens de hoje tem um novo destino : o aparelho móvel.

A mobilidade e a evolução dos aparelhos celulares e PDAs trazem consigo também uma mudança de paradigma. Hoje em dia não teremos mais somente a transmissão de um para muitos, mas também a transmissão de um para um, ou de muitos pra muitos. Poderemos ter, num futuro não tão distante assim, o telespectador fazendo parte da programação, interagindo não somente através de ligações gratuitas ou de voto via internet, mas quem sabe até mandando seu próprio depoi-

mento, ou simplesmente mostrando o lugar onde passou as suas férias, utilizando apenas a câmera embutida no seu celular.

Para que isso se torne realidade, temos que compreender este novo ambiente no qual estaremos inseridos, para que assim consigamos adaptar os futuros codificadores para este novo paradigma. Com estas novas condições, algumas técnicas reconhecidas de compressão de vídeo passam a não ser mais adequadas, como por exemplo a estimação de movimento do lado do codificador. A necessidade de uma grande capacidade computacional e de energia para realizarmos essa tarefa inviabiliza sua utilização em aparelhos com capacidade limitada, onde a utilização da bateria deve ser feita de uma maneira econômica e racional.

Com isso em mente, algumas pessoas se voltaram para resultados já conhecidos da teoria de informação desde a década de 70, com Slepian e Wolf [8] e com Wyner e Ziv [10], e desenvolveram técnicas de codificação onde quem explora as estatísticas da fonte é o decodificador, e não o codificador, o que permite usar um codificador menos complexo, ao custo de ter um decodificador mais complexo. Algoritmos baseados em tais paradigmas são também conhecidos como algoritmos de codificação distribuídos, e devido as suas características são uma grande promessa para serem usados no meio móvel.

1.2 Organização da Tese

Aqui iremos decorrer sobre algumas técnicas implementadas para esse novo ambiente, analisá-las, e ao final propor uma maneira nova de encarar este problema.

No capítulo 2, iremos mostrar os resultados que chegaram Slepian e Wolf [8], assim como Wyner e Ziv [10], para sedimentar a base teórica na qual se fundamentam os algoritmos apresentados nesta tese.

No capítulo 3 faremos uma revisão dos trabalhos publicados sobre este tema até o presente momento, e faremos uma breve análise dos diversos métodos apresentados.

Dois dos métodos que serão mostrados no capítulo 3 foram escolhidos para serem implementados, de modo que os resultados possam ser comparados, e as técnicas de codificação neste novo ambiente serem esclarecidas. No capítulo 4 apresentare-

mos um codificador usando o paradigma de Wyner-Ziv no domínio dos pixels, já no capítulo 5 mostraremos um codificador no domínio das transformadas, e também assim com no capítulo anterior, iremos apresentar tanto a teoria quanto o resultado de nossa implementação, comparando com o resultado publicados.

Após apresentar alguns métodos de codificação, iremos propor no capítulo 6 uma técnica alternativa de codificação distribuída de uma seqüência de vídeo, e ao final do capítulo faremos uma comparação entre os diversos métodos apresentados.

Na conclusão, capítulo 7, além do resultado, algumas propostas para a melhoria do codificador e novas idéias de implementações futuras serão apresentadas.

Para auxiliar o entendimento de alguns items da tese, três apêndices também se encontram ao final : o apêndice A contém algumas definições e demonstrações de teoremas de teoria da informação, para sedimentar a base teórica desta tese. No apêndice B, as equações do código turbo para codificação distribuída são derivadas, e no apêndice C, algumas técnicas para a obtenção da informação lateral (informação usada para decodificar uma imagem) serão demonstradas.

Capítulo 2

O Paradigma de Wyner-Ziv

2.1 Introdução

Neste capítulo iremos mostrar a teoria na qual se basearam diferentes grupos para desenvolverem seus codificadores distribuídos. Os fundamentos teóricos da codificação de fontes distribuídas serão revisados e os teoremas de maior importância serão postulados. Veremos também uma interpretação geométrica da solução do problema e como construir códigos que implementam o paradigma de Wyner-Ziv. Para complementar este capítulo, separamos no apêndice A alguns conceitos chaves e demonstrações dos teoremas. Para o leitor que já é familiarizado com os fundamentos da teoria da informação para codificação distribuída, os capítulos seguintes despertarão um maior interessante, pois nele demonstraremos alguns resultados que chegaram pesquisadores, implementando na prática o que Wyner e Ziv declararam na teoria.

2.2 Codificação de fontes correlacionadas : o teorema de Slepian Wolf

Da teoria da informação, sabemos que a taxa necessária para codificar uma fonte $\mathscr{X}$ sem erros tem que ser maior do que a entropia da fonte ($X > H(X)$). Agora supomos ter duas fontes $(\mathscr{X}, \mathscr{Y}) \sim p(x,y)$. Seguindo o mesmo raciocínio, uma taxa $H(X,Y)$ é suficiente para codificar ambas as fontes juntas. Mas como fazer se desejamos descrever $\mathscr{X}$ e Y separadamente, porém reconstruí-los conjuntamente ?

A figura 2.1 ilustra essa situação.

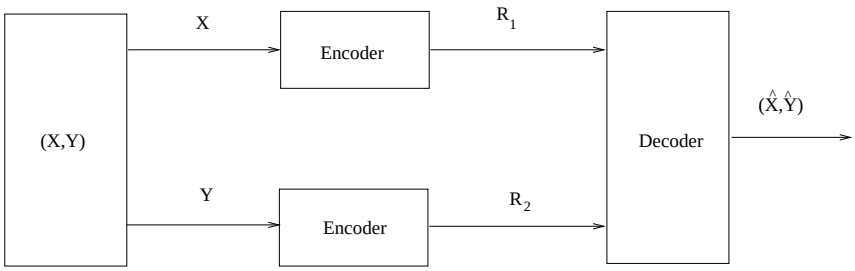


Figura 2.1: Estrutura de codificação de fontes correlacionadas

Claramente, se desejamos codificar as fontes separadamente, a taxa $R = R_x + R_y > H(X) + H(Y)$ é suficiente. Porém, Slepian e Wolf [8] mostraram em seu artigo de 1973 que uma taxa total de $R = H(X,Y)$ é suficiente para codificar separadamente fontes correlacionadas.

Teorema 2.2.1 *(Teorema de **Slepian-Wolf**) Seja uma fonte $\mathscr{X}$ codificada a uma taxa R_1, e uma fonte $\mathscr{Y}$, correlacionada com $\mathscr{X}$, porém codificada independentemente com a taxa R_2. Podemos mostrar que a região das taxas é dada por*

$$\begin{aligned} R_1 &\geq H(X) \\ R_2 &\geq H(Y) \\ R_1 + R_2 &\geq H(X,Y) \end{aligned} \tag{2.1}$$

2.2.1 Interpretação do teorema de Slepian Wolf

Uma outra interpretação do teorema de Slepian Wolf pode ser exemplificada no seguinte problema. Iremos transmitir o vetor X^n com uma taxa $R_1 = nH(X)$, e assim decodificá-lo com uma probabilidade de erro arbitrariamente baixa. Como podemos decodificar Y^n, para que a taxa de transmissão de Y^n, R_2, seja igual a $nH(Y|X)$? A solução para este problema pode ser visualizada na figura 2.2.

Se o codificador de Y^n conhecer $\mathscr{X}^n$, basta mandar o índice dentro do subconjunto criado por X^n, porém este não conhece $\mathscr{X}^n$. Ao invés de determinar o subconjunto típico, o codificador de Y^n colore arbitrariamente $\mathscr{Y}^n$ com 2^{nR_2} cores, e manda a cor do elemento a ser transmitido. Se o número de cores é alto o suficiente, então com uma alta probabilidade teremos cores diferentes dentro de um

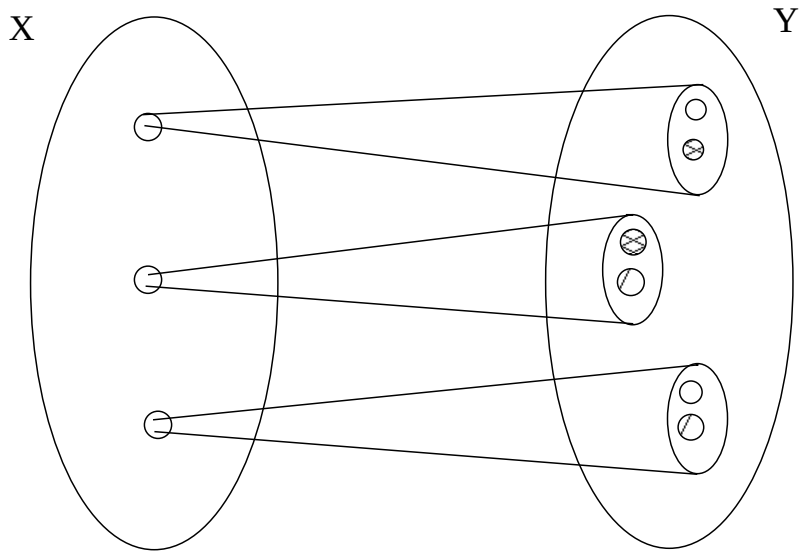


Figura 2.2: Interpretação do teorema de Slepian Wolf

subconjunto associado a $\mathscr{X}^n$, e $\mathscr{Y}^n$ poderá ser unicamente decodificado, dado $\mathscr{X}^n$. Se $R_2 > nH(Y|X)$, o número de cores é exponencialmente maior do que o número de elementos dentro de um subconjunto, e teremos uma probabilidade de erro exponencialmente pequena. Podemos ver outra aplicação do teorema de Slepian-Wolf no exemplo a seguir.

Exemplo

$\mathscr{X}$ e $\mathscr{Y}$ são duas fontes correlacionadas de 3 bits equiprováveis. A distância de Hamming entre seus elementos é de 1.

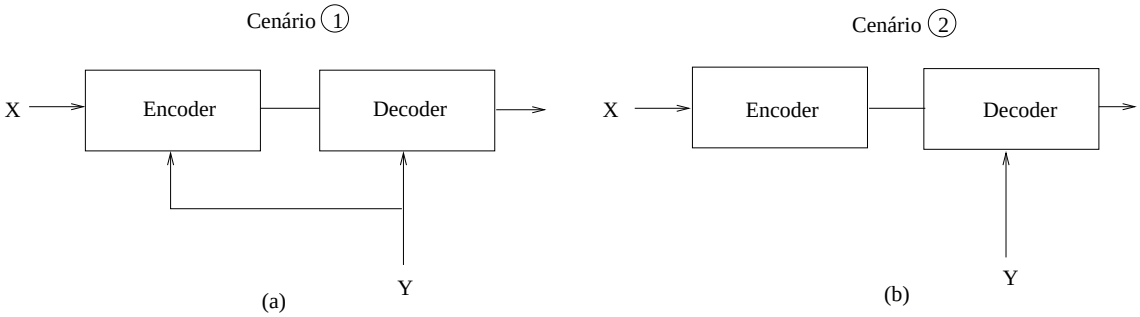


Figura 2.3: Exemplo de codificação com informação lateral (a) ou com informação lateral apenas no decodificador (b)

Cenário 1 Transmitimos $X \otimes Y$, que pode assumir 4 valores distintos $(000, 001, 010, 100)$, logo com 2 bits apenas.

Cenário 2 O codificador não conhece Y, porém conhece a estrutura da correlação entre as duas variáveis.

O espaço de $\mathscr{X}$ é particionado em 4 *cosets* .

***Coset* 1** ([000] e [111])

***Coset* 2** ([001] e [110])

***Coset* 3** ([010] e [101])

***Coset* 4** ([100] e [011])

O codificador identifica o *coset* de $\mathscr{X}$ e manda seu índice com apenas 2 bits. O decodificador, por sua vez, usa $\mathscr{Y}$ para tirar a ambigüidade de $\mathscr{X}$ (declarando a palavra-código do *coset* mais próximo à ela). Por exemplo, se X for [010](pertence ao coset 3), todos os seguintes valores de Y([010],[110],[000] e [011]) irão decodificar X corretamente, provendo robustez em relação à Y. Vemos que:

1. O mapeamento de $\mathscr{X}$ pode ser feito de maneira computacionalmente eficiente através dos *coset codes*, resultando numa baixa complexidade de codificação.

2. Foi usado um *coset* que é casado com a distância entre $\mathscr{X}$ e $\mathscr{Y}$ para particionar o espaço, resultando num alto desempenho em termos de compressão.

3. A partição de $\mathscr{X}$ é universal. A mesma partição serve, independentemente do valor de $\mathscr{Y}$, desde que eles mantenham a estrutura de correlação.

2.3 Codificação de fonte com informação lateral

Iremos considerar agora duas variáveis aleatórias X e Y, que serão codificadas separadamente, porém apenas X será recuperada. Quantos bits R_1 são necessários para descrever X, se temos apenas R_2 bits para descrever Y?

$\rightarrow$ Se $R_2 > H(Y)$, então de acordo com Slepian-Wolf $R_1 = H(X|Y)$

$\rightarrow$ Se $R_2 = 0$, não temos ajuda alguma, portanto $R_1 = H(X)$

→ Geralmente iremos usar $R_2 = I(Y;\widehat{Y})$ bits para descrever uma versão aproximada de Y. Isso nos permite descrever X usando $H(X|\widehat{Y})$ bits na presença da informação lateral $\widehat{Y}$

Teorema 2.3.1 *[7] Seja $(X,Y) \sim p(x,y)$. Se Y é codificado a uma taxa R_2 (e com isso recebemos U no lado do decodificador) e X a uma taxa R_1, então podemos recuperar X com uma probabilidade pequena de erro se*

$$\begin{aligned} R_1 &\geq H(X|U) \\ R_2 &\geq I(Y;U) \end{aligned}$$

para uma função de probabilidade de massa conjunta $p(x,y)p(u|y)$, onde $|\mathscr{U}| \leq |\mathscr{Y}| + 2$, onde $|\cdot|$ designa o número de elementos de um conjunto.

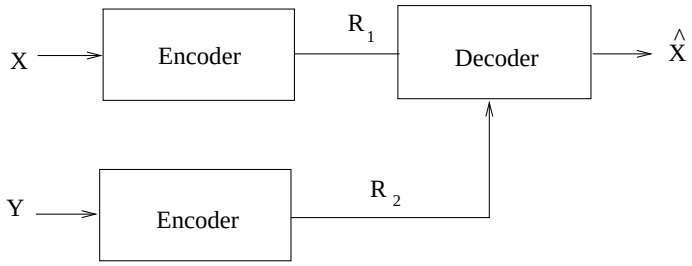


Figura 2.4: Codificação de fonte com informação lateral

2.4 Taxa-Distorção com informação lateral

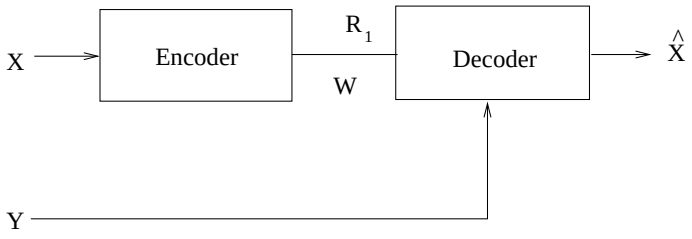


Figura 2.5: Codificação com informação lateral, para o cálculo da taxa-distorção

A função taxa-distorção com informação lateral ($R_y(D)$) é definida como a taxa mínima requerida para se alcançar uma distorção D, se a informação lateral

Y está disponível no lado do decodificador. Como a informação lateral só poderá ajudar, teremos sempre $R_y(D) \leq R(D)$ (o caso específico de distorção nula equivale ao teorema de Slepian-Wolf, ou seja $R_y(0) = H(X|Y)$)).

O que Wyner e Ziv declaram no seu artigo de 1976 [9] é repetido aqui para efeito de esclarecimento, assim como sua dedução, reproduzida no apêndice A.

Teorema 2.4.1 *(Teorema de **Wyner-Ziv**) Seja (X, Y) com distribuição i.i.d. dada por $\backsim p(x, y)$, e considere a distorção dada por $d(x^n, \hat{x}^n) = \sum_{i=1}^{n} d(x_i, \hat{x}_i)$. A função taxa-distorção com informação lateral é dada por:*

$$R_y(D) = \min_{p(w|x)} \min_{f} (I(X; W) - I(Y; W))$$

onde a minimização é sobre todas as funções decodificadoras $f : \mathscr{Y} \times \mathscr{W} \rightarrow \widehat{\mathscr{X}}$ e a função de probabilidade de massa condicional $p(w|x)$, $|W| \leq |\mathscr{X}| + 1$, tal que

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

O teorema de Wyner-Ziv mostra a curva taxa-distorção que a codificação de fontes correlacionadas pode atingir. Quando a distorção é zero, temos o teorema de Slepian-Wolf.

2.5 Estimação de movimento no âmbito Wyner-Ziv

Um dos argumentos mais fortemente usados pela comunidade e que tem atraído o interesse para este assunto é o fato de podermos usar o paradigma de Wyner-Ziv para levar a tarefa de estimação de movimento do codificador para o decodificador. Algumas implementações podem ser vistas nos capítulos 3, 4 e 5. Porém existem várias técnicas de se estimar o movimento, podendo assim o codificador gerar diversos vetores de movimento, dependendo da taxa alvo. No artigo de P. Ishwar, V. M. Prabhakaran e K. Ramchandran [11], a forma como o codificador Wyner-Ziv trata as diversas estimações de movimento é demonstrada, o que segue aqui para esclarecimento.

Os autores consideram a estimação de movimento como uma ligação entre a fonte e a informação lateral, dada pelo estado do sistema. Eles chamaram a codificação que leva em conta a estimação de movimento como informação lateral de *Source Encoding with side-information under Ambiguous State-Of-Nature* (**SEASON**).

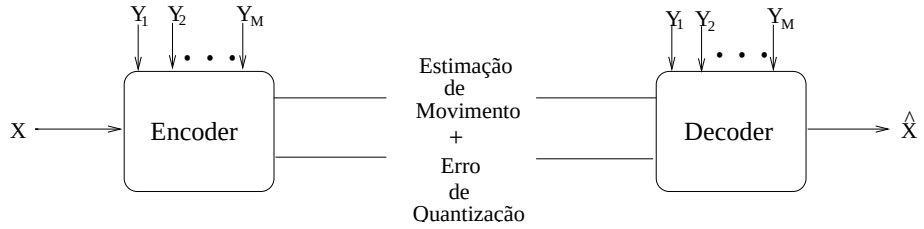


Figura 2.6: Estrutura de um codificador convencional

Nos codificadores de vídeo convencionais, como mostrado na figura 2.5, a taxa usualmente gasta para transmitir X é de $\frac{1}{n}\log M$ bits para especificar a estimação de movimento e adicionado da taxa $R(D) = \min(0, \frac{1}{2}\log(\frac{\sigma_Z^2}{D}))$ para transmitir os pixels.

Em [11], os autores demonstram que poderemos transmitir com a mesma taxa, mesmo quando não temos as imagens estimadas do lado do codificador. Para superar a incerteza adicionada pela escolha dos diversos vetores de movimento, devemos diminuir o tamanho dos conjuntos onde são divididas as palavras-código, equivalentemente aumentando o número de conjuntos (com uma adição de $\frac{1}{n}\log(M)$ bits). A prova deste postulado segue a mesma idéia apresentada anteriormente de dividir o espaço em conjuntos usando códigos aleatórios, baseada em teoria de informação.

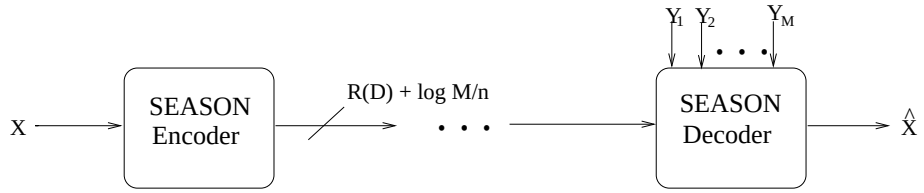


Figura 2.7: Codificador *SEASON*

Os mesmos resultados aqui demonstrados são realizáveis usando quantização escalar em coeficientes e códigos coset baseados em treliça, como mostrou o mesmo grupo em [14], e outros grupos em [12] e [15]. O entusiasmo com os resultados vem do fato de podermos usar ferramentas de codificação intra e obtermos uma taxa de compressão como se estivéssemos codificando quadros inter. Outra vantagem deste esquema de codificação é a robustez inerente do processo, que também foi estudada em [16], [17] e em [18].

Iremos no próximo capítulo apresentar algumas implementações publicadas até o momento. Destas implementações escolhemos duas e as implementamos, para podermos comparar com os resultados publicados, e aprender um pouco mais sobre codificação com informação lateral. No capítulo 6 implementamos uma maneira alternativa de codificar imagens com informação lateral baseado em quantizadores vetoriais por aproximações sucessivas.

Capítulo 3

Implementações do teorema de Wyner-Ziv

Neste capítulo iremos apresentar alguns trabalhos recentemente publicados, acentuando algumas características em comum e algumas particularidades de cada trabalho. Iremos aqui analisar como a comunidade científica está tratando o problema de codificação distribuída, para podermos assim identificar alguns quesitos essenciais para a codificação com informação lateral. Dos trabalhos aqui apresentados escolhemos dois para implementar. As implementações serão discutidas nos capítulos 4 e 5. Com o conhecimento adquirido durante a implementação, desenvolvemos no capítulo 6 uma maneira alternativa de codificar, usando quantizadores vetoriais por aproximações sucessivas. Os resultados serão discutidos posteriormente.

3.1 O codificador PRISM

Em [14] foi apresentado o codificador **PRISM** (*power-efficient, robust, high-compression, Syndrome-based multimedia coding*). A implementação deste codificador foi motivada por resultados obtidos pelo mesmo grupo sobre o tema de codificação distribuída, como em [19], ou em [21] e [20].

Este codificador herda a baixa complexidade e a robustez de codificadores de vídeo intra *motion*-JPEG, enquanto que atinge a eficiência de codificadores de vídeo interframe com completa estimação de movimento. Como usa algumas ferramentas usuais de padrões de codificação de vídeo, um cenário onde ele estaria inserido

seria com transmissores e receptores simples, conversando através de um "proxy transcodificador". Este poderia ser uma estação central onde o *bitstream* gerado pelo codificador **PRISM** seria convertido em um *bitstream* de um padrão conhecido de codificação de vídeo, como por exemplo o MPEG ou o H.264.

O processo de codificação segue a idéia mostrada no capítulo anterior de partição do espaço da fonte em subconjuntos. Seja X o macrobloco a ser codificado, Y a sua melhor predição (com estimação de movimento) usando quadros passados. Podemos interpretar Y como sendo $X+N$, onde N será o ruído de correlação (X e N são modelados como vetores laplacianos aleatórios independentes). O macrobloco X será codificado como uma bloco intra, e com isso obteremos uma versão quantizada de X. Devemos achar um código de canal casado com o ruído de correlação N, e então usaremos isso para particionar o espaço de X em subconjuntos[1].

3.1.1 Implementação do codificador PRISM

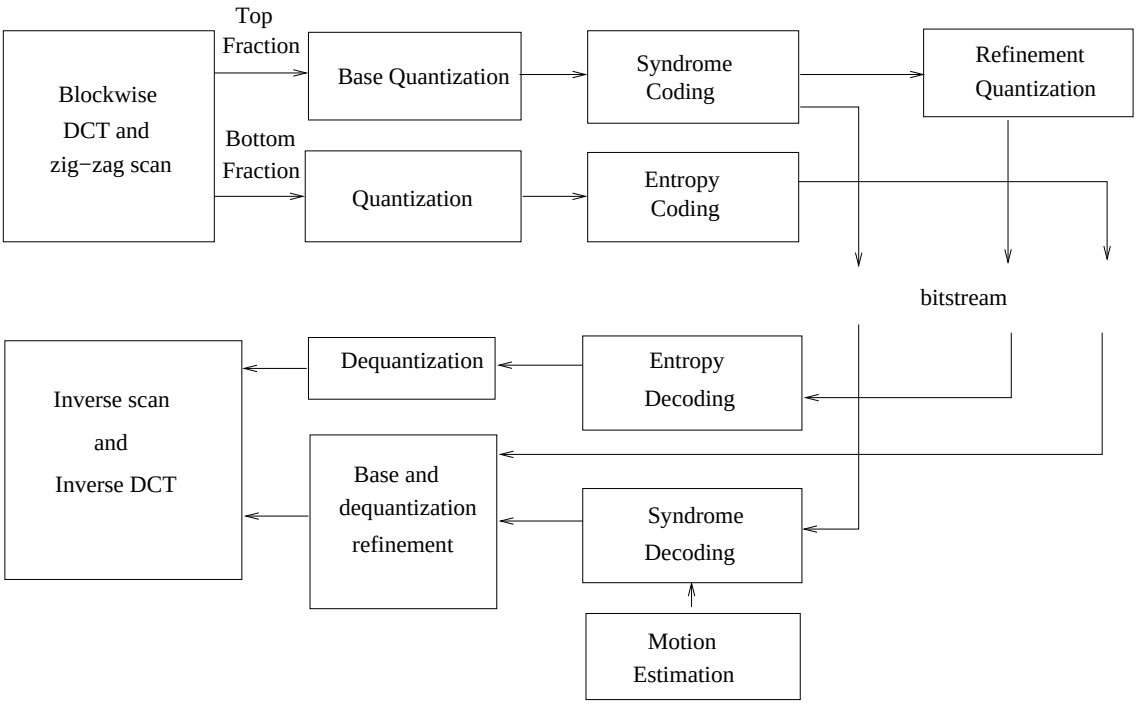


Figura 3.1: Estrutura do codificador **PRISM**

[1]No caso de codificação de vídeo, temos fontes de valores reais com estruturas de ruído correlacionadas complexas e imprecisas. Então existe uma probabilidade não-nula de erro de decodificação, que pode ser mitigada usando estratégias de confinamento e detecção de erro.

O processo de codificação começa dividindo o quadro em blocos de dimensão 16×16 ou 8×8 não sobrepostos. em seguida temos os seguintes passos.

1. Classificação : essa etapa visa classificar a estrutura do ruído de correlação (baseados em modelagem offline de misturas) do bloco sendo codificado para facilitar o uso do código de canal apropriado. A diferença quadrática entre o bloco a ser codificado e o seu correspondente no quadro anterior modela o ruído N. De acordo com este valor, escolhemos um código de canal com maior ou menor poder de correção (desde o modo SKIP até o modo INTRA).

2. Transformada : a DCT é utilizada no bloco para descorrelacionar os pixels.

3. Quantização escalar : Os coeficientes da DCT serão quantizados com o passo de quantização proporcional ao ruído de correlação N (problema conhecido como *water-filling*)[2].

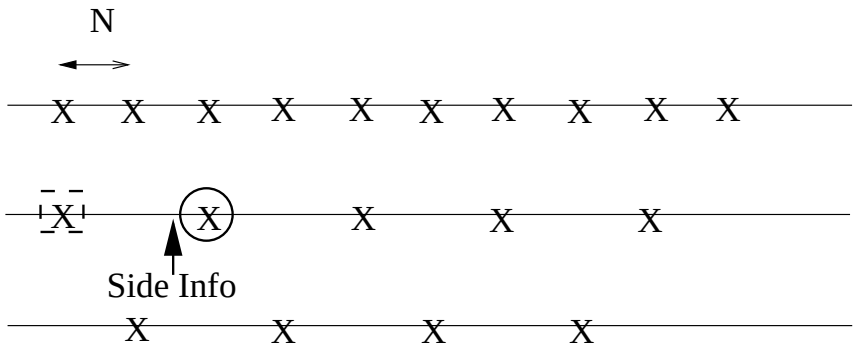


Figura 3.2: Quantizador do codificador **PRISM**

4. Codificação por síndrome : os coeficientes dos blocos são varridos (varredura zig-zag), e apenas os primeiros 20% dos coeficientes são codificados por síndrome (a síndrome irá gerar o índice dos *cosets*, que são divididos de acordo com a **estatística de N**), pois nestes coeficientes está concentrada a energia (informação). Na implementação de [14], foi usado um código treliça de taxa 1/2 e memória 7.

[2]Se o passo for muito pequeno, podemos ter erro de decodificação, uma vez que as palavras-código estarão muito próximas, e a informação lateral não conseguirá resolver esta ambigüidade

5. Codificação de fonte "*pura*" : os 80% restante dos coeficientes são codificados intra, ou melhor, quantizados e e o codificador por entropia usado é o codificador Huffman run-length.

6. Quantização de refinamento : para atingir a qualidade desejada, os coeficientes que foram quantizados pela síndrome (de acordo com N) são novamente quantizados até que se atinja a qualidade desejada, uma vez que a sua prévia quantização foi feita em função do ruído de correlação. O intervalo de quantização é subdividido para chegarmos ao passo de quantização desejado, então a diferença entre a fonte e a sua descrição previamente quantizada é novamente quantizada com este novo passo.

7. *Cyclic Redundancy Test*(CRC) : serve como uma assinatura da seqüência de palavras quantizadas, o decodificador usa para saber quando a decodificação foi feita com sucesso. É calculado usando uma seqüência quantizada de palavras-código módulo 4, e deve ter "força" suficiente para atuar como uma assinatura confiável.

Figura 3.3: CRC do codificador **PRISM**

A decodificação ocorre da seguinte maneira.

1. Geração da informação lateral (estimação de movimento) : o decodificador faz estimação de movimento e gera candidatos a preditores para poder decodificar a seqüência de palavras-código quantizadas. Qualquer estimação de movimento poderá ser usada, o que irá melhorar ou piorar a informação lateral

2. Decodificação da síndrome : o decodificador usa o algoritmo de Viterbi para identificar a seqüência que mais se aproxima da informação adicional, dada as síndromes. Se o CRC confirmar a seqüência achada, então a decodificação teve sucesso, do contrário uma outra predição (estimação de movimento) é utilizada.

3. Estimação e Reconstrução : Ao recuperarmos a seqüência de palavras-código quantizadas, usamos esta seqüência junto com a predição para estimar o quadro atual (melhor reconstrução do quadro). Nesta etapa podemos utilizar outras ferramentas de processamento de sinais para melhorar a relação sinal-ruído.

4. Decodificação "pura" de fonte : decodificação por entropia dos 80% dos coeficientes que foram codificados intra.

5. Zig-zag inverso : após a dequantização, a varredura zig-zag inversa é realizada para remontarmos o bloco.

6. Transformada inversa : a transformada inversa retorna os pixels.

3.1.2 Resultados do codificador PRISM

Dos gráficos publicados em [14] (figura 3.4) podemos ver o ganho do codificador **PRISM**. Note que o uso do quadro anterior como forma de estimar a correlação entre a informação a ser codificada e a informação lateral gera bons resultados, porém ainda não conseguimos alcançar a curva gerada por um codificador inter H.263+. Os melhores resultados foram alcançados com seqüências com pouco movimento, como a seqüência *Mother&Daughter*, porém resultados promissores foram alcançados com seqüências de alto movimento, especialmente no que tange a parte de resiliência a erros. Experiências foram feitas simulando perda de pacotes ou até quadros inteiros, e no caso de um codificador comum H.263+ vemos que a seqüência inteira é afetada pela propagação de erro, o que já não ocorre com o codificador **PRISM**

Outra aplicação interessante do codificador foi publicado em [22], onde o codificador **PRISM** foi utilizado para transmissão de vídeos *multicast*, no qual um único *stream* de vídeo pode servir para diversos clientes através de uma rede sem-fio, com transmissão variada e taxas de perda alta. A robustez do codificador vem do fato dele usar o quadro anterior apenas para estimar a correlação entre o sinal a ser transmitido e a informação lateral. Se o decodificador não tiver o quadro anterior para estimar esta estatística devido a um erro no canal ou perda de pacote, ele poderá ainda usar outros quadros para esta estimação, e desde que o ruído de

correlação entre os blocos escolhidos esteja dentro da margem de ruído para qual a síndrome foi designada, a decodificação ocorre sem problemas.

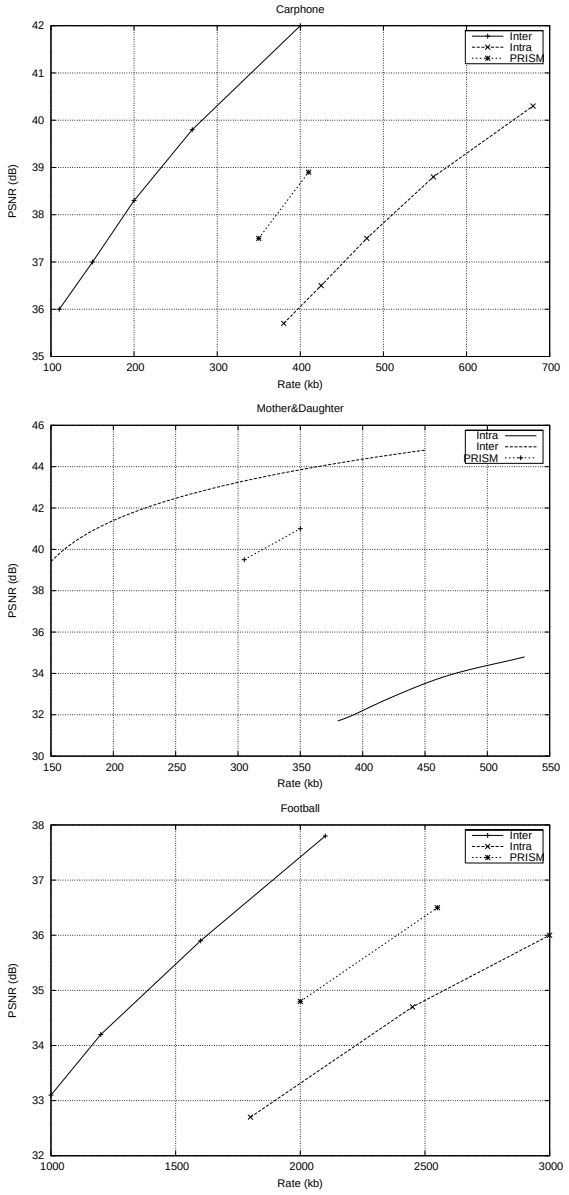


Figura 3.4: Curvas Taxa-Distorção para o codificador **PRISM**, um codificador H.263 intra e para um codificador H.263 inter.

3.2 Um codificador baseado no padrão H.264 : *A State-free causal video encoding paradigm*

Para combater o descasamento (ou "*drift*") do decodificador com o codificador, o paper [15] propõe uma estrutura de codificação onde o codificador e o decodificador não precisam manter o mesmo estado, ou equivalentemente, usar a mesma predição, um problema crítico em ambientes com altas taxas de erro (ex. transmissões móveis).

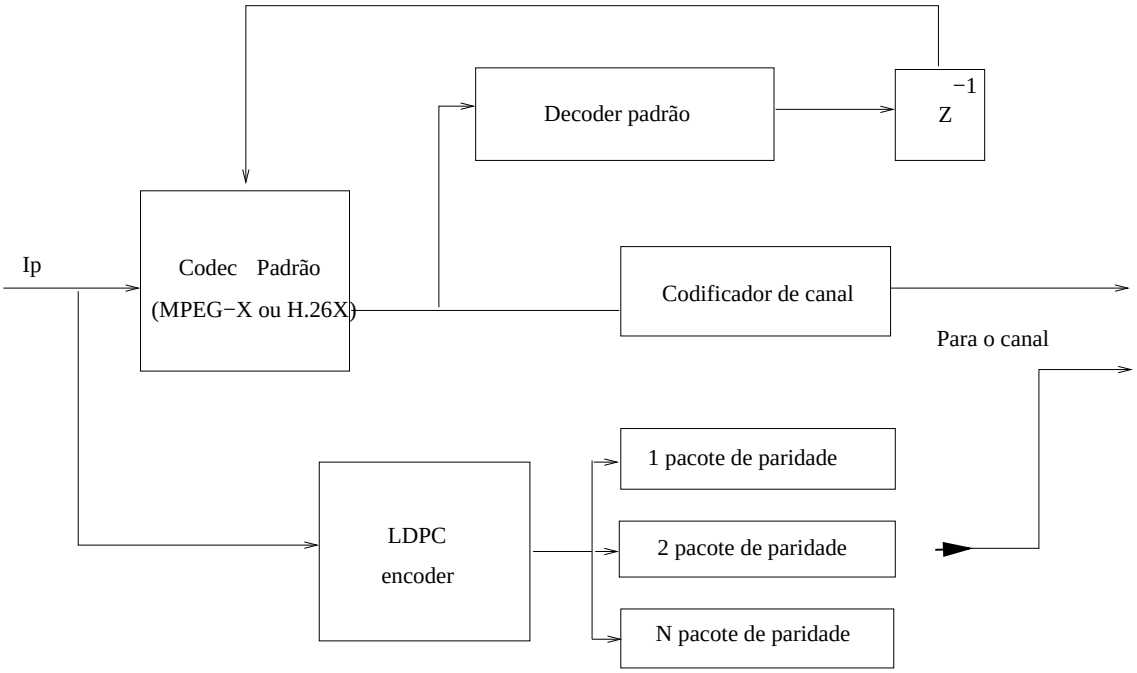


Figura 3.5: Diagrama em blocos do codificador proposto

→ Algoritmo de Codificação

Aplicamos ao quadro $\mathscr{I}_j$ a transformada *forward* do H.264 diretamente para cada bloco 4×4, sem predição, e quantizamos o resultado com o quantizador de zona morta, também do H.264.

Cada componente freqüêncial dos blocos é agrupada em um vetor $\hat{v}_j$ (componentes freqüênciais) com q elementos, e cada elemento é convertido para uma representação de L-bits binária. Cada plano de bits l será multiplicado por uma matriz geradora de um código LDPC (low density parity code) : $\hat{v}_j(l)\dot{G}_{p\times q}$, gerando $p-q$ bits de paridade, que constituem um "*coset packet*".

Para cada plano de bits l, múltiplos pacotes são gerados variando p. Durante

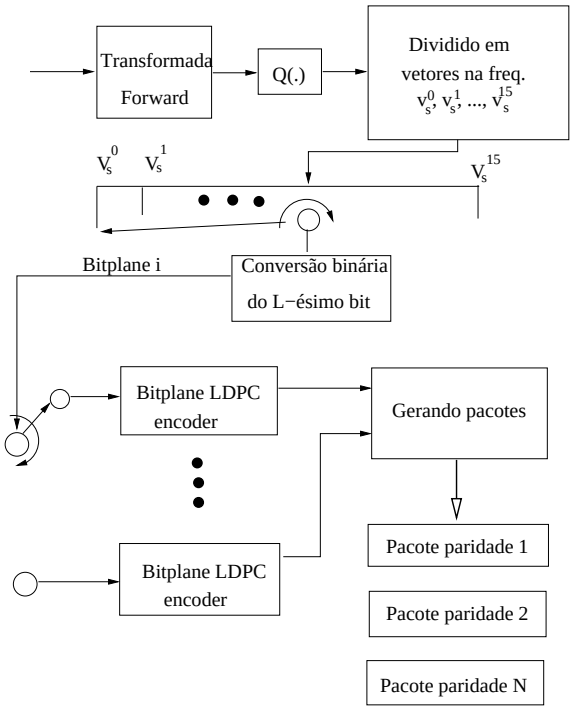


Figura 3.6: Diagrama em blocos do codificador

a transmissão, um pacote apropriado é mandado, dependendo da caracterização estatística do canal com perdas.

A codificação do quadro $\mathscr{I}_j$ não depende do quadro $\mathscr{I}_{j-1}$, portanto temos uma codificação *state-free* (independente).

→ Algoritmo de Decodificação

A decodificação do quadro $\hat{v}_j$ usa o quadro decodificado e reconstruído $\hat{v}_{j-1}^{'}$ como informação adicional e os bits de paridade como informação dos *cosets*. Cada plano de bits é decodificado seqüencialmente, sendo o plano de bits decodificado usado também como informação adicional para decodificar os planos de bits seguintes. Como as componentes freqüênciais são tratadas de maneira independente, a redundância espacial não é considerada, só é levada em conta a redundância temporal.

→ Melhorias na codificação

A informação do *coset* tem duas finalidades:

- corrigir os erros do canal[3]

[3]erros do canal são de natureza estatística e desconhecidos pelo codificador

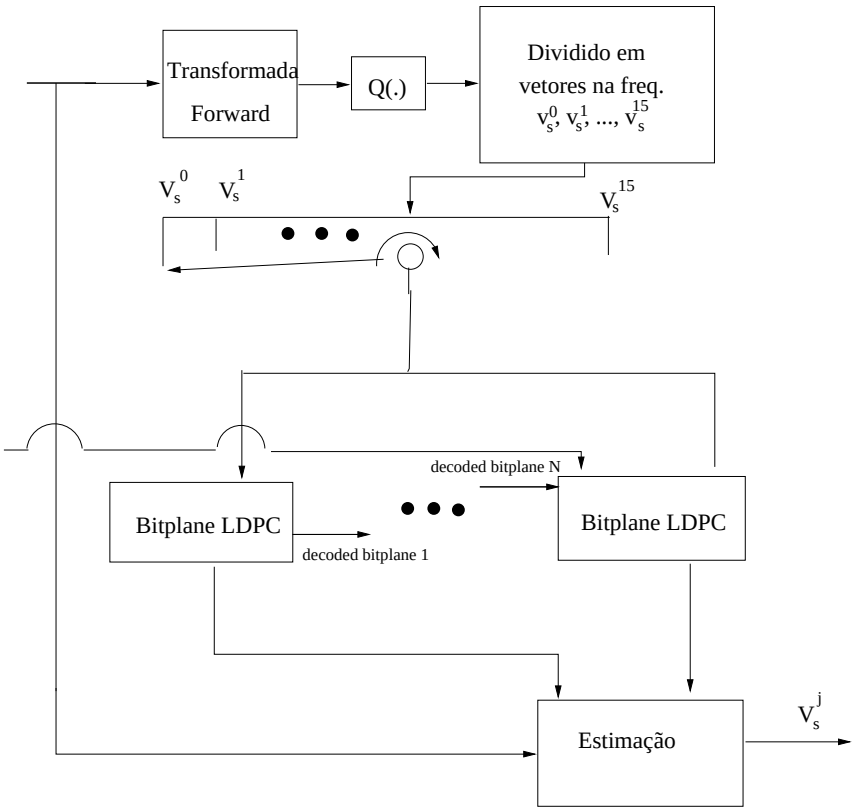


Figura 3.7: Diagrama em blocos do codificador

- mandar a informação de inovação $v_k - E[v_k|\hat{v}_{k-1}]$ [4]

Para codificar v_k, o codificador gera dois tipos de informação.

- $t_{1,k} = v_k - v_{k-1} \xrightarrow{Q[\cdot]} \hat{t}_{1,k}$
- $t_{2,k}$ =$coset$ de $\hat{v}_k$

O decodificador usa como informação lateral $\hat{v}_{k-1}$ e $\hat{t}_{1,k}$. A perda no ganho é devido apenas a não-Gaussianidade dos erros do canal. Para decodificar $\hat{I}_j$, o procedimento de estimação usa estimativa MMSE de v_j, baseado na informação adicional $\hat{v}_{j-1}$, e no vetor decodificado via LDPC $\hat{v}_j$. A estimação MMSE requer o conhecimento da pdf de $v_j - \hat{v}_{j-1}$. Simulações demonstraram que uma pdf exponencial com um impulso na origem se aproxima bem da pdf desejada.

→ Resultados

Os resultados obtidos são de 1 a 1,5 dB abaixo dos resultados para o codificador H.264, em taxas médias e baixas, como mostra a figura 3.8. A opinião dos

[4]note que a função de correlação é conhecida pelo codificador

autores é que essa perda seria apenas um pequeno preço a se pagar pelas funcionalidades adicionadas, como robustez e simplificação do codificador, e ela ocorre simplesmente pela ineficiência na compressão da informação dos *cosets* .

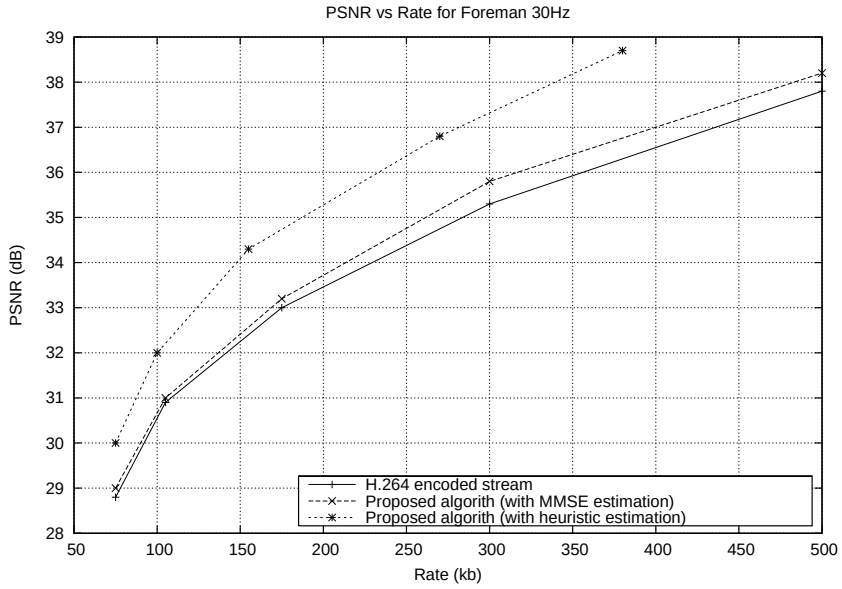


Figura 3.8: Curvas Taxa-Distorção para o codificador proposto em [15]

3.3 Os codificadores de *Stanford*

O grupo da universidade de *Stanford* produziu um vasto material sobre o assunto de codificação distribuída. Além do notável artigo [23] onde os autores dão um panorama sobre a maioria dos feitos alcançados no âmbito da codificação distribuída, vários outros artigos ([13], [17], [24] e [12]) já foram publicados, fruto do trabalho deste grupo em cima do tema central, a codificação Wyner-Ziv. Este grupo baseia o processo de codificação principalmente em cima do código turbo, e algumas variações em cima do tema central foram exploradas com um relativo sucesso, como por exemplo usar o domínio das freqüências para calcular os *cosets* . Neste processo o codificador não usa informação alguma de nenhum quadro anterior, todo o processo de comparação entre dois quadros é feito no decodificador. Para compensar a perda de não usar nenhuma informação sobre a natureza da informação lateral direto no codificador, vemos a presença de um canal de realimentação do decodificador para o codificador. Uma vez que somente parte da paridade será mandada, saber o quanto

será necessário para decodificar o bitstream vira uma tarefa do decodificador, que a realiza através do canal de retorno.

Devido ao fato da literatura, embora não tão completa, ser bastante extensa escolhemos dois codificadores propostos por esse grupo para servir de referência para nossa implementação. Portanto os capítulos seguintes irão decorrer mais extensamente sobre os codecs implementados por este grupo, abordando questões práticas que não foram publicadas, como por exemplo o ajuste dos parâmetros do codificador turbo.

3.4 Comparação entre as diversas técnicas

Uma vez que os resultados são muito dependentes da fonte utilizada, e da maneira como foi obtida a informação lateral, fica difícil comparar os resultados de cada codificador mencionado anteriormente.

Note que os codificadores das seções 3.1, 3.2 e 3.3 tentam, de certa maneira, estimar a correlação entre a informação lateral e o pixel a ser transmitido. Enquanto que os codificadores em 3.1, 3.2 usam o quadro anterior para terem uma estimativa desta correlação, o codificador em 3.3 usa um canal de retorno para poder melhor se adaptar as mudanças na correlação entre a informação lateral e o valor real a ser transmitido. O canal de realimentação traz a vantagem de ajustar os parâmetros do codificador adaptativamente, e com isso usar uma estimativa mais fiel da correlação entre fonte e informação lateral. Já os outros dois codificadores não usam o canal de retorno, em compensação não ficam atrelados às condições impostas pelo decodificador. O uso do quadro anterior resulta numa estimativa sub-ótima da correlação entre fonte e informação lateral, e com isso podemos usar diversos tipos de decodificadores, para diversas taxas-alvo.

Outro ponto em comum da implementação da seção 3.2 com as implementações em 3.3 foi o fato de ambas terem usado planos de bits para alimentarem o codificador (LDPC no primeiro caso, e turbo no segundo). As probabilidades de um plano foram usadas na decodificação dos planos seguintes, e a maneira como isso foi feito no código turbo será explicado a seguir.

Capítulo 4

Codificação de Vídeo Wyner-Ziv no domínio dos Pixels

4.1 Introdução

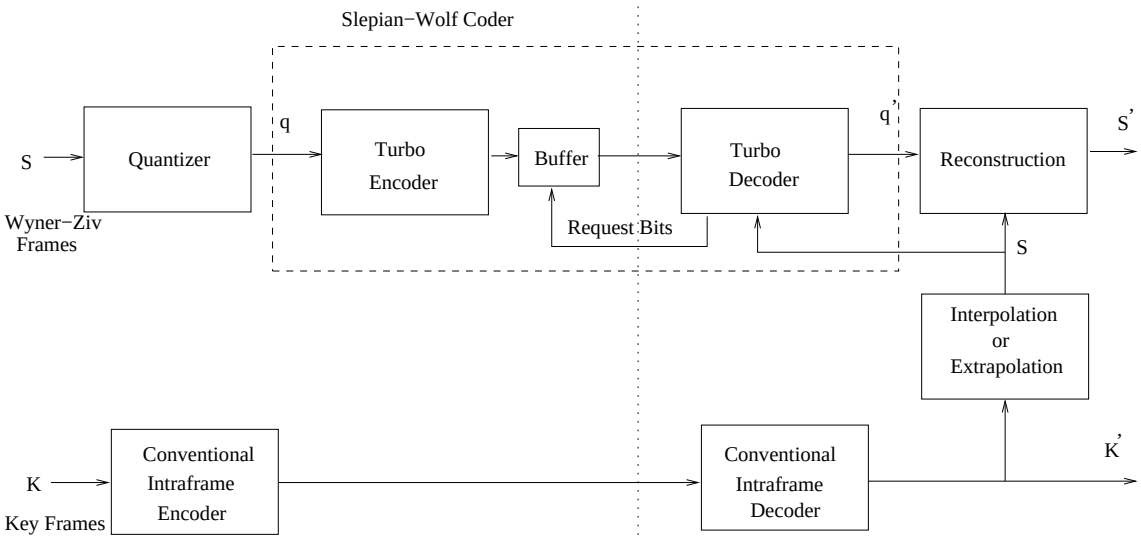


Figura 4.1: Codificador Wyner-Ziv usando o domínio dos pixels

Em [12], A. Aaron *et al* apresentaram o codificador da figura 4.1. Os resultados deste codificador e o fato dele abordar o problema de Wyner-Ziv usando um código turbo foram os motivadores para que nós implementássemos a idéia e assim, aprendêssemos a lidar com codificação distribuída. O processo de codificação ocorre da seguinte maneira:

- Cada pixel é quantizado usando um quantizador escalar uniforme com $2^M \in \{2, 4, 16\}$ níveis para formar o fluxo de símbolos quantizados

- Esses símbolos irão formar um "grande bloco de símbolos", que servirão de entrada para o codificador de Slepian-Wolf. Cada plano de bit será mandado separadamente, um após o outro.

- O codificador Slepian-Wolf é implementado usando um código turbo perfurado de taxa compatível (**RCPT**)[1]

- Para cada quadro Wyner-Ziv (S), o codificador pode utilizar os quadros Wyner-Ziv ou quadros intra (*key frames*) precedentes para formar a informação lateral ($\hat{S}$), que será utilizada para estimar o quadro S.[2]

- O decodificador usa a informação lateral ($\hat{S}$) e o subconjunto de bits de paridade para decodificar o fluxo q'. Se precisar, o decodificador requer através do canal de *realimentação* mais bits de paridade, porém devido à presença da informação lateral, o número de bits usados na decodificação (para identificar em qual dos M níveis o pixel se encontra) é sempre menor que o número de níveis ($k \leq M$) e com isso teremos **compressão de dados**.

- Após decodificar o fluxo q', o decodificador reconstrói o quadro através da informação lateral em conjunto com q'. Se o pixel de $\hat{S}$ estiver dentro da área (conjunto ou *bin*) determinada por q', então o pixel reconstruído ficará próximo da informação lateral. Se o pixel de $\hat{S}$ estiver fora do *bin*, então o pixel reconstruído assume o valor da fronteira do intervalo na direção da informação lateral.[3]

Para melhorar a qualidade subjetiva da imagem, o processo de *dithering* subtrativo foi aplicado ao pixels usando um padrão pseudo-aleatório. Isso melhora a qualidade

[1] O RCPT com realimentação provê a flexibilidade na taxa, que é essencial para adaptar as mudanças estatísticas entre a fonte e a informação lateral

[2] o decodificador assume uma dependência estatística entre a fonte e a informação lateral

[3] esta é uma maneira de limitar a distorção de reconstrução, ficando esta sempre menor do que o passo de quantização. Desta maneira eliminamos os erros muito positivos ou muito negativos que teríamos, caso usássemos a informação lateral.

subjetiva na hora da reconstrução, caso a informação lateral não seja muito confiável, e tivermos que utilizar os símbolos quantizados, o que levaria a efeitos de contorno quando o passo de quantização é grande.

Outra análise interessante feita em [12] foi o efeito da informação lateral para o codificador. Nele podemos constatar que, dependendo do número de quadros usados na estimação da informação lateral, teremos um ganho maior ou menor. Esta variação do ganho dependente da informação lateral indica que podemos atingir diversos pontos da curva taxa-distorção, dependendo apenas da informação lateral utilizada no processo. A presença de realimentação no decodificador acrescenta uma flexibilidade ao processo de codificação, fazendo com que o decodificador use uma estimativa melhor da correlação entre a fonte e a informação lateral.

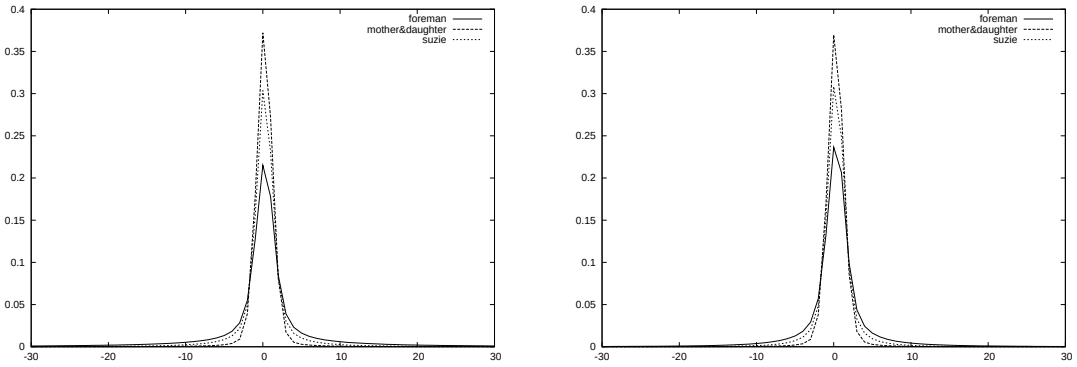
Os resultados publicados indicam um ganho de codificação considerável, relativo ao codificador intra, porém ainda existe um *gap* entre a técnica de codificação distribuída e os codificadores inter. Para alguns artefatos de interpolação na estimação de movimento, como em oclusões e seqüências com muito movimento, as seqüências Wyner-Ziv produzem resultados com PSNR de melhor qualidade visual, por evitarem blocos nas imagens reconstruídas.

Alguns parâmetros e ajustes feitos no codificador foram obtidos de forma experimental, como a variância da diferença entre a fonte e a informação lateral. Outros problemas identificados durante a implementação serão explicados neste capítulo. Os resultados obtidos na nossas simulações serão comparados com os resultados publicados. A descrição da obtenção da informação lateral se encontra no apêndice C, onde as diversas maneiras de se obtê-la são comparadas e analisadas.

4.2 A função de correlação entre a fonte e a informação lateral

A função de correlação entre a fonte e a informação lateral depende do modo de obtenção da informação lateral e do tipo de seqüência a ser analisada. De análises feitas no apêndice C, escolhemos duas maneiras de obter a informação lateral : fazendo a média do quadro anterior com o quadro posterior, ou fazendo a estimação de movimento bidirecional. Com isso, codificamos as seqüências *Foreman*,

Mother&Daughter e *Suzie*, e obtivemos a seguinte distribuição para a diferença entre a entrada e a informação lateral.



(a) (b)

Figura 4.2: Distribuição de probabilidades para informação lateral com (a) média dos quadros anterior e posterior e com (b) estimação de movimento bilateral

Note que, para a seqüência *Mother&Daughter*, a função de correlação se concentra na origem. A curva tem uma queda mais acentuada que as demais e um valor central maior, enquanto que a seqüência *Foreman* tem uma variância maior e um valor central menor. Analisando os gráficos da figura 4.2, vemos que a seqüência *Mother&Daughter* é mais correlacionada com a informação lateral, uma vez que a seqüência não tem muito movimento. Os gráficos não diferiram muito para esta seqüência com os dois tipos de obtenção da informação lateral, uma vez que a seqüência tem pouco movimento, e a estimação de movimento acaba resultando em valores similares a simplesmente fazer a média dos quadro posterior e anterior. Já com a seqüência *Foreman* podemos ver que a informação lateral obtida via estimação de movimento é mais correlacionada com a fonte, o que condiz com a prática, uma vez que esta seqüência tem muito movimento, e simplesmente fazer a média dos quadros não resulta numa informação lateral confiável.

As distribuições de probabilidade foram modeladas de acordo com distribuições laplacianas, dadas pela fórmula:

$$f(x) = \frac{\sqrt{2}}{\alpha} e^{-\frac{(x-z)^2}{\alpha}}, \text{onde } \alpha \text{ é a variância da função} \tag{4.1}$$

Após a análise, chegamos na seguinte tabela de valores para α, baseado na seqüência *foreman*.

	Média	Estimação de movimento
α	5.8	5.2

Tabela 4.1: Valores estimados para α de acordo com as distribuições em 4.2

4.3 O código turbo

O código turbo usa dois codificadores convolucionais ligados através de um *interleaver* para fazer a codificação do canal. Para códigos turbo que usam símbolos como entrada, estes podem ser considerados os bits sistemáticos, que irão gerar para cada grupo de sistemáticos dois bits de paridade. Para cada entrada teremos um determinado estado dos codificadores convolucionais (que pode ser interpretado como uma determinada posição numa estrutura de treliça). O algoritmo de decodificação usado será o **BCJR**, que usa a probabilidade a priori e as probabilidades de transição para estimar o estado da treliça usada pelos codificadores, e assim determinar qual foi a entrada do codificador num determinado instante.

Na implementação do código turbo, assumimos uma transmissão perfeita dos bits de paridade, enquanto que os bits sistemáticos eram trocados pela informação lateral. Podemos interpretar a informação lateral como uma versão ruidosa da entrada, que foi transmitida através de um canal, onde a distribuição do erro do canal é dada pela correlação entre a entrada e a informação lateral, que pode ser modelada através de uma laplaciana, como mostrou a seção anterior.

Teremos então a probabilidade de transição (γ) de um estado s' para um estado s no tempo k dada pela fórmula

$$\gamma_k(s', s) = Pr\{X^s_{k-1} = i\}.\,\delta(y^p_{k-1} - x^p_{k-1}(s', s)).\,Pr\{X^s_{k-1} = i|z_{k-1}\} \tag{4.2}$$

onde,

i é a entrada

x_p são os bits de paridade

y_p são os bits de paridade recebidos após a transmissão

$\delta(\cdot)$ é a função impulso

e z_{k-1} é a informação lateral

Na figura 4.3, vemos a estrutura do código convolucional usado no código turbo. No apêndice B encontram-se deduções mais extensas das fórmulas utilizadas no código turbo usando informação lateral no algoritmo de decodificação, no lugar dos bits sistemáticos.

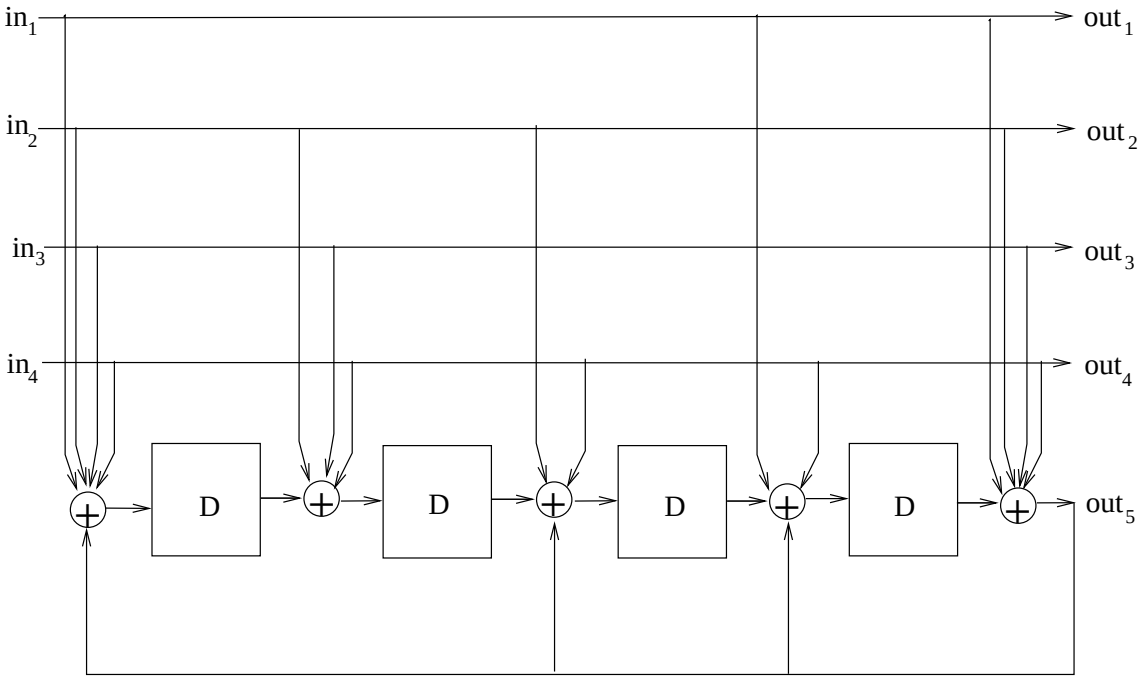


Figura 4.3: Estrutura do código turbo

O cálculo de $Pr\{X_{k-1}^s = i|z_{k-1}\}$[4], usado para obter o valor de $\gamma_{k-1}(s', s)$ é feito de modo iterativo, onde além da informação lateral, os planos de bits que já foram decodificados influenciam no cálculo da probabilidade. Os planos de bits vão representar intervalos onde a função de probabilidade condicional deve ser integrada para obtermos o valor da probabilidade usada no processo de decodificação BCJR. A medida que mandamos os planos de bits subseqüentes, o intervalo irá sendo determinado e conseqüentemente diminuindo, pois a nova probabilidade será condicionada não somente ao bit mandado, mas também aos bits já decodificados. O exemplo a seguir irá ajudar a esclarecer a questão.

Ao quantizarmos um pixel de valor 146 em quatro níveis, obtemos o índice 9 (1 0 0 1). No primeiro plano de bits a ser mandado, a probabilidade, modelada

[4]A probabilidade $Pr\{X_{k-1}^s = i|z_{k-1}\}$ é proporcional a $Pr\{z_{k-1}|X_{k-1}^s = i\}.Pr\{X_{k-1}^s = i\}$ para um determinado X.

como uma função laplaciana, é calculada da seguinte maneira:

$$Pr\{X^s_{0,k-1} = 1|z_{k-1}\} = \int_{128}^{255} \frac{\sqrt{2}}{\alpha} e^{-\sqrt{2}\cdot\frac{(|x-z_{k-1}|)}{\alpha}} \partial x$$

$$Pr\{X^s_{0,k-1} = 0|z_{k-1}\} = \int_{0}^{127} \frac{\sqrt{2}}{\alpha} e^{-\sqrt{2}\cdot\frac{(|x-z_{k-1}|)}{\alpha}} \partial x$$

e de acordo com o valor das probabilidades acima, iremos decidir se 0 ou 1 foi mandado. Se de acordo com as probabilidades acima, decidimos que $X = 1$ foi mandado, então, para o segundo plano de bits, teremos o seguinte cálculo.

$$Pr\{X^s_{1,k-1} = 1|z_{k-1}, X^s_{0,k-1} = 1\} = \int_{192}^{255} \frac{\sqrt{2}}{\alpha} e^{-\sqrt{2}\cdot\frac{(|x-z_{k-1}|)}{\alpha}} \partial x$$

$$Pr\{X^s_{1,k-1} = 0|z_{k-1}, X^s_{0,k-1} = 1\} = \int_{128}^{191} \frac{\sqrt{2}}{\alpha} e^{-\sqrt{2}\cdot\frac{(|x-z_{k-1}|)}{\alpha}} \partial x$$

E assim por diante, com os planos de bits seguintes.

O número de iterações também influencia no resultado. Em nossas simulações usamos 10 iterações, por ser um valor que já apresenta resultados bem parecidos com o valor ótimo, porém pequeno o suficiente para fazer as simulações realizáveis. A treliça do codificador **A** era também forçada a retornar ao valor zero, o que adicionava mais dois símbolos ao final do bloco, devido à estrutura do polinômio utilizada.

4.4 Controle de Taxa

O algoritmo de controle de taxa é um ponto crítico do sistema. Dependendo da sua implementação, podemos atingir diversos pontos da curva taxa-distorção. Em [12], a única menção sobre o controle de taxa foi o fato de um *bitplane* não poder ter uma probabilidade de erro maior do que 10^{-3}, o que era determinado de forma ideal do lado do decodificador. Também no mesmo artigo foi citado que para atingir taxas pequenas, uma perfuração no código era realizada conforme descrito em [25]. A tabela de perfuração usada durante as simulações foi a tabela 4.2. Os valores da tabela estão representados em octal, e cada linha representa do bits gerados pelo codificador (sistemáticos e de paridade do codificador **A** e do codificador **B**)[5]. Como o código turbo não mandou bit sistemático algum, a perfuração foi feita apenas nos bits de paridades, conforme as duas linhas dos elementos da tabela abaixo.

[5] **M** é a taxa do código (código 1/**M**), e **P** é o período de perfuração do código

Código RCPT : M=3 e P=4

BlockSize	4/3	2/3	4/7	1/2	4/9	2/5	4/11	1/3
N=256	16	17	17	17	17	17	17	17
	02	02	12	12	13	13	17	17
	02	02	02	12	12	13	13	17
N=512	16	17	17	17	17	17	17	17
	02	02	12	12	13	13	17	17
	02	02	02	12	12	13	13	17
N=1024	16	17	17	17	17	17	17	17
	02	02	12	12	13	13	17	17
	02	02	02	12	12	13	13	17
N=4096	16	17	17	17	17	17	17	17
	02	02	12	12	13	13	17	17
	02	02	02	12	12	13	13	17

Tabela 4.2: Tabela de perfuração do código turbo

O algoritmo de controle de taxa usado durante as simulações funcionava da seguinte maneira. Primeiramente o decodificador avaliava se era necessário mandar algum bit para o plano de bit a ser decodificado de acordo com a probabilidade de erro em questão. Se esta fosse maior do que 10^{-3}, então o decodificador pedia mais bits de paridade, que eram perfurados de acordo com a tabela 4.2, e mandados para o decodificador. Este realizava a decodificação iterativa e, dependendo da quantidade de erros do plano de bit decodificado, pedia mais bits de paridade ou seguia decodificando o próximo plano de bit.

4.5 Resultados e análise

O gráfico 4.4 mostra o resultado médio obtido codificando os 100 primeiros quadros da seqüência *Foreman*, isto é, os quadros pares eram codificados por um codificador H.263+ Intra, e os quadros ímpares eram codificados via Wyner-Ziv. Note o ganho de codificação gerado usando estimação de movimento no lugar da média dos quadros. Uma vez que temos uma informação lateral mais confiável, necessitaremos de menos bits para a transmissão do quadro Wyner-Ziv, além de reconstruirmos o quadro com maior fidelidade.

Podemos ver na figura 4.5 que os nossos resultados chegaram bem perto

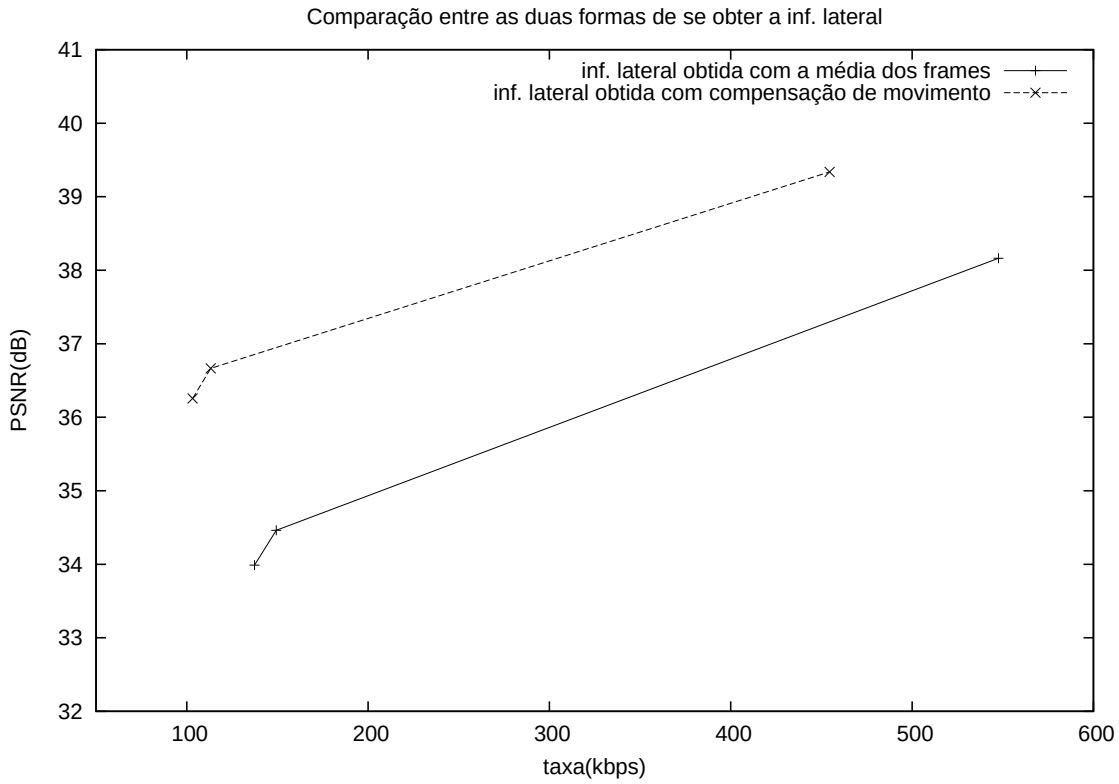


Figura 4.4: Resultados obtido codificando os 100 primeiros quadros da seqüência *Foreman*

dos resultados publicados em [12]. Tivemos apenas uma discrepância em taxas altas, onde obtivemos uma taxa maior para uma mesma distorção medida. Um dos prováveis motivos que identificamos como o causador desta discrepância para altas taxas é o fato do nosso controle de taxa não ter diferenciado os *bitplanes*. Erros nos primeiros planos de bits irão gerar mais erros nos planos subseqüentes, portanto se permitimos uma taxa de erro pequena nos primeiros *bitplanes*, esses erros irão se propagar nos planos seguintes, o que acabará resultando também numa maior requisição de bits, aumentando a taxa, além do fato de que erros nos primeiros planos serão mais visíveis, e portanto resultarão num PSNR pior. Outro provável motivo seria como a informação lateral foi obtida, que influencia diretamente no desempenho do codificador. Note que o esquema de codificação Wyner-Ziv se mostrou muito superior a um codificador intra, porém ainda temos um *gap* grande comparado a um codificador inter.

Na figura 4.6 vemos a informação lateral utilizada (média entre os quadros posterior e anterior à imagem), e o quadro reconstruído. Como a informação lateral

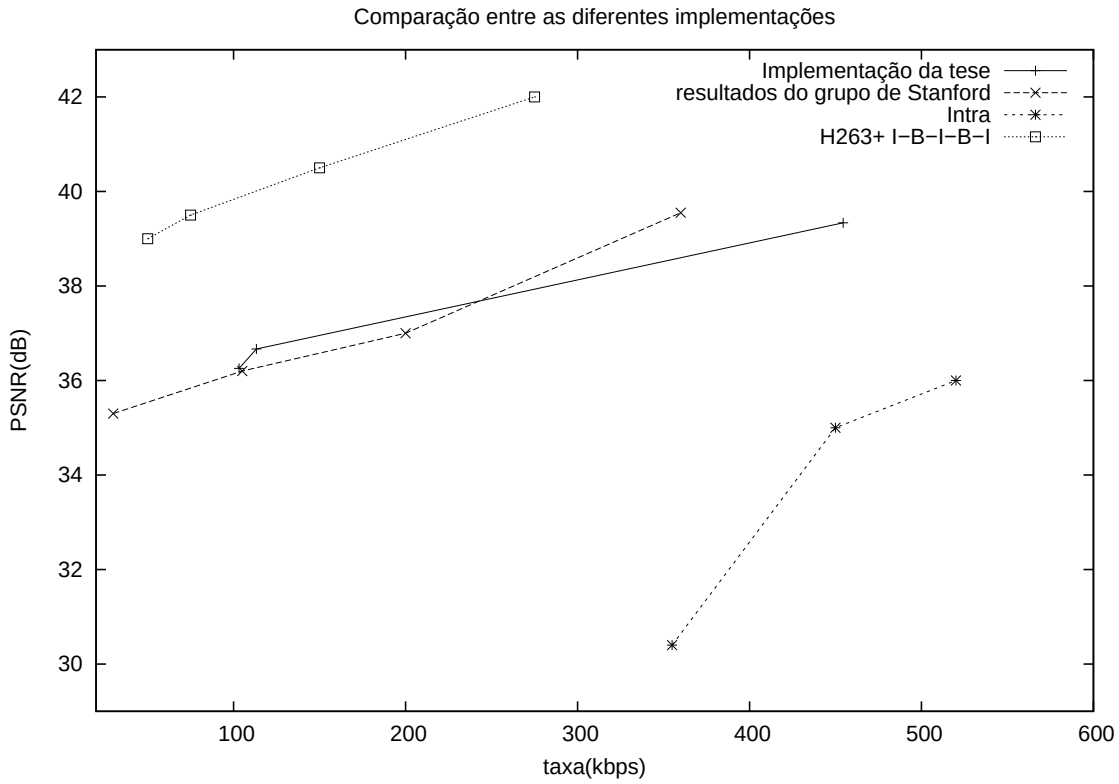
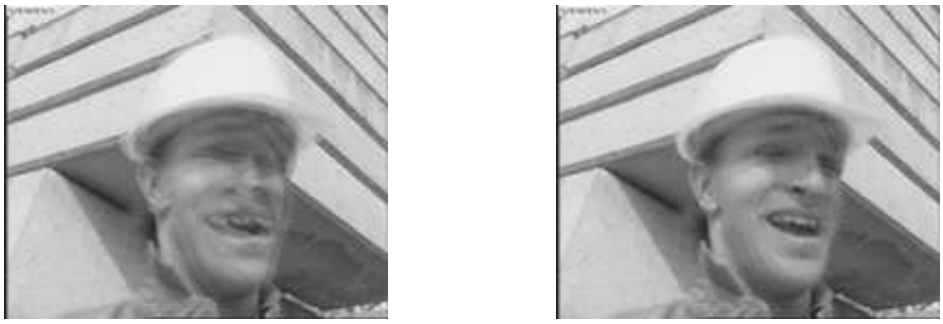


Figura 4.5: Comparação entre os resultados do grupo de *Stanford* e os implementados nesta tese para a seqüência *Foreman* com informação lateral obtida através da estimação de movimento bidirecional

era muito ruim, necessitamos de uma taxa alta para conseguir "corrigir os erros de transmissão" do canal fictício entre a fonte e a informação lateral.

Neste codificador estamos explorando apenas a correlação temporal entre os quadros. Para melhorar o desempenho do codificador, devemos também explorar a correlação espacial da imagem, o que é feito no capítulo 5.



(a) (b)

Figura 4.6: (a) informação lateral (média - 29.258241 dB) (b) quadro reconstruído (36.624809 dB @ 665.46kbps)

Capítulo 5

Codificação de Vídeo Wyner-Ziv no domínio da Transformada

5.1 Introdução

O próximo passo do grupo de *Stanford* foi evoluir o codificador proposto em [12], que codificava os pixels diretamente, aplicando uma transformada e passando o codificador do domínio dos pixels para o domínio dos coeficientes. Os resultados, mostrados em [24], indicaram que o uso da transformada traz um ganho de 0,5 até 2,0 dB em relação ao método anterior. O processo de codificação ocorre da seguinte maneira:

- No codificador, uma DCT é aplicada aos blocos de tamanho 4×4 que dividem a imagem. Os coeficientes obtidos com a transformada são agrupados em bandas X_k, e cada banda é codificada separadamente

- Para cada banda X_k, os coeficientes são quantizados usando um quantizador escalar uniforme com 2^{M_k} níveis

- Os símbolos quantizados são convertidos em palavras-código binárias de tamanho fixo. Os planos de bits são agrupados, formando M_k vetores de planos de bits.

- Cada plano de bit é codificado separadamente pelo codificador Slepian-Wolf, que usa um código turbo perfurado com taxa compatível (**RCPT**)

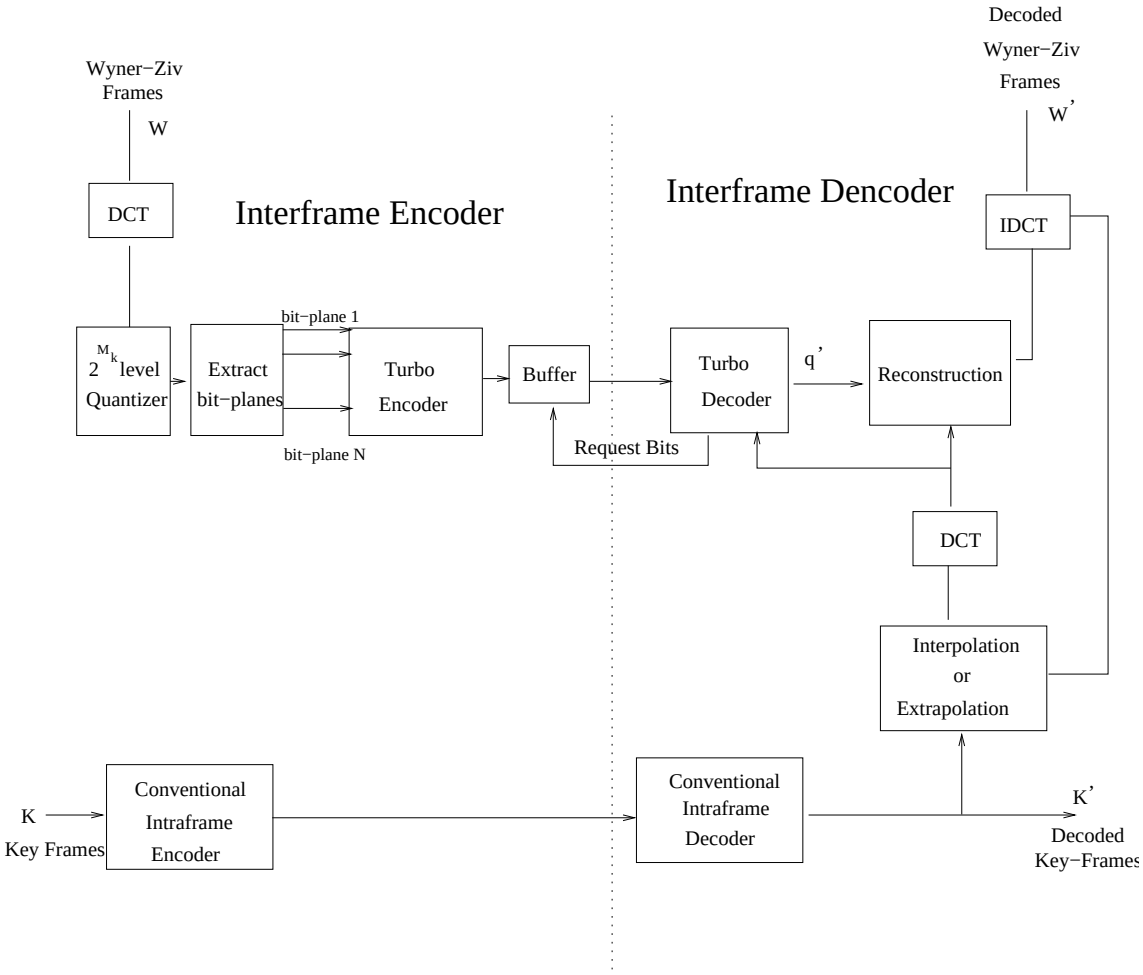


Figura 5.1: Codificador Wyner-Ziv usando o domínio das transformadas

- Os bits de paridade são guardados em um buffer, e a medida que o decodificador necessita de mais bits, ele faz a requisição através do canal de retorno. Dessa forma o processo de codificação consegue se adaptar às constantes mudanças das probabilidades entre fonte e informação lateral.

- O processo de requisição e decodificação de bits prossegue até que o decodificador atinja uma probabilidade de erro desejada. Com ajuda da informação lateral o codificador irá pedir $R_k \leq M_k$ bits, e com isso teremos a compressão desejada.

- As probabilidades encontradas para um plano de bit serão usadas nos planos de bits posteriores. Após decodificarmos o símbolo quantizado $q_k^{'}$, iremos

reconstruir o coeficiente X_k^i calculando $E(X_k|q_k^i, \hat{X}_k)$.[1]

O processo de decodificação, por ser iterativo, já exige uma maior complexidade computacional do que técnicas convencionais como Huffman ou códigos aritméticos. A geração da informação lateral também adiciona complexidade ao decodificador, porém isto pode ser compensado usando métodos simples de interpolação ou de extrapolação para obtê-la.

Algumas particularidades da implementação do codificador, pela dificuldade de compreensão e de se achar alguma referência sobre o assunto, serão explicadas nas seções seguintes, e ao final os resultados das simulações serão expostos.

5.2 A função de correlação entre a fonte e a informação lateral

No caso do codificador no domínio das transformadas, cada coeficiente apresenta uma correlação distinta com o coeficiente relativo da informação lateral. Porém todas as correlações podem ser modeladas pela função $f(d) = \frac{\alpha}{2}e^{-\alpha|d|}$, onde d é a diferença entre o coeficiente da fonte X_k e o coeficiente da informação lateral $\hat{X}_k$ Após simulações feitas com a seqüência *Foreman*, obtivemos os valores de α para cada banda mostrados na tabela 5.1, usando estimação de movimento para obter a informação lateral, ou simplesmente usando a média dos quadros posterior e anterior.

Como os valores obtidos para α não diferiram muito se usarmos estimação de movimento ou apenas a média para obtermos a informação lateral, todas as nossas simulações foram feitas com os valores determinados com estimação de movimento.

5.3 Quantizadores

Como a DCT concentra a energia do bloco nos primeiros coeficientes, a escolha certa do quantizador adequado para cada coeficiente é fundamental no processo

[1] Assim como no decodificador no domínio dos pixels, a reconstrução é limitada pelo passo de quantização, dessa forma evitando grandes erros positivos ou negativos, melhorando a qualidade visual da imagem reconstruída

Banda	Média	Estimação de movimento
α_0	0.28	0.29
α_1	0.38	0.40
α_2	0.45	0.47
α_3	0.75	0.74
α_4	0.46	0.44
α_5	0.47	0.46
α_6	0.55	0.53
α_7	0.72	0.75
α_8	0.63	0.62
α_9	0.63	0.62
α_{10}	0.66	0.66
α_{11}	0.87	0.91
α_{12}	0.91	0.91
α_{13}	0.96	0.90
α_{14}	0.75	0.75
α_{15}	0.75	0.75

Tabela 5.1: Valores estimados para α de cada banda da transformada DCT 4×4

de codificação. A alocação de bits é feita de acordo com os quantizadores escolhidos, porém a imagem também influencia na escolha destes quantizadores. Para chegar ao conjunto ótimo de quantizadores, os autores de [24] fizeram diversas experiências, testando diversos quantizadores com diferentes seqüências. Baseado na função de custo Lagrangiano, os passos de quantização para cada coeficiente foram escolhidos e divididos em sete taxas-alvo distintas, como mostra a figura 5.2. Nela podemos ver o número de níveis para cada determinado coeficiente. Note que a alocação de bits proposta prioriza os coeficientes mais próximos ao coeficiente DC, devido a sua propriedade de concentrar energia.

Num bloco 4×4, o valor máximo que um coeficiente pode assumir é de $256 \times 4 \times 4 = 2^{10}$, logo a faixa dinâmica do quantizador deverá ser de até 2^{10} níveis. Porém, no caso de coeficientes de imagens reais, especialmente os coeficientes AC de menor energia, a faixa efetivamente utilizada é bem menor. Portanto, para otimizar o processo de quantização, limitamos a excursão do sinal não quantizado de acordo com o coeficiente e a taxa a ser usada. A escolha do número de bits máximo para representar um coeficiente não quantizado foi baseada na seqüência *Foreman*, onde

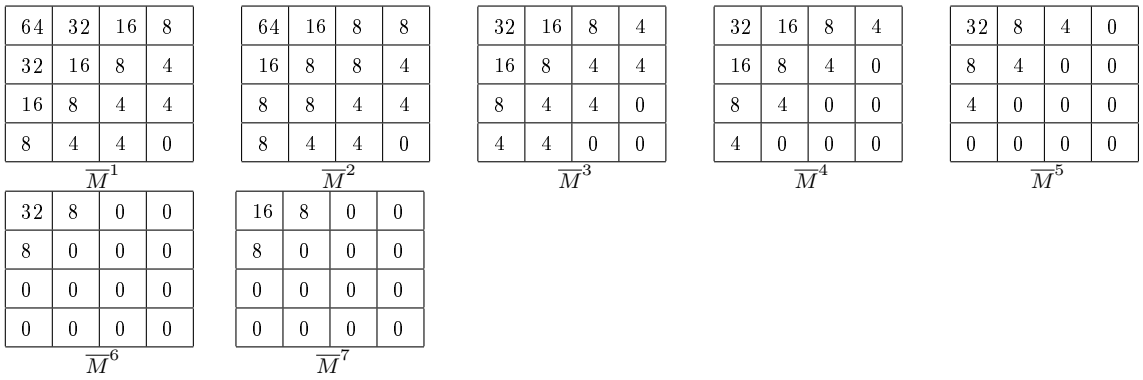


Figura 5.2: Os quantizadores em ordem crescente de distorção

obtemos o valor máximo e mínimo para cada coeficiente, e assim escolhemos o número de bits necessários para representar estes coeficientes. Na figura 5.3 vemos o número de bits usados em cada par coeficiente-taxa.

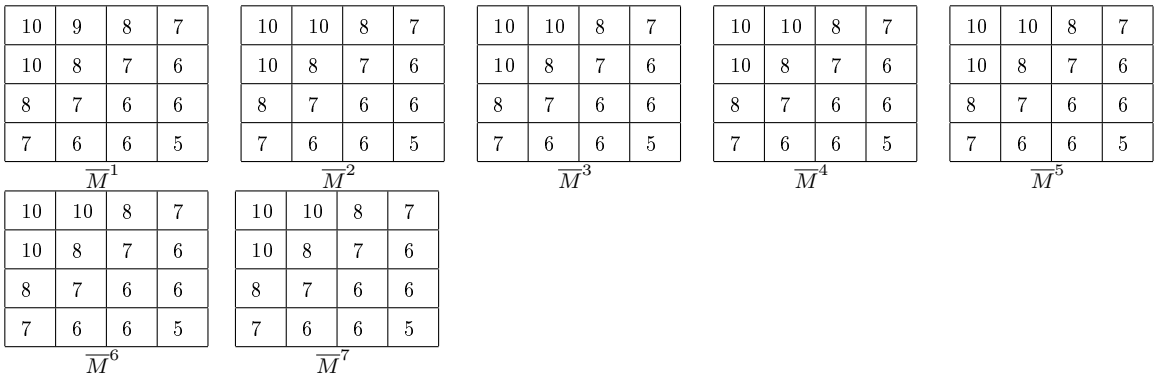


Figura 5.3: Número de bits usados para representar cada coeficiente antes da quantização

5.4 O código turbo

Para o código turbo, foram usados dois codificadores convolucionais com taxa $\frac{1}{2}$ e matriz geradora $[1 \quad \frac{1+D+D^3+D^4}{1+D^3+D^4}]$. Os bits sistemáticos gerados pelo código são descartados, e apenas os bits de paridade são mandados. Para atingirmos taxas ainda menores, uma matrix de perfuração definida em [25], e reproduzida aqui na tabela 5.2, foi usada.

Os bits de paridades dos planos de bits eram mandados, e então o decodificador avaliava os bits sistemáticos produzidos pela informação lateral com os bits de paridade vindos da fonte real. Se após o processo de decodificação o plano de bit

Código RCPT : M=3 e P=8

BlockSize	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2	8/17	4/9	8/19	2/5	8/21	4/11	8/23	1/3
N=256	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	011	011	013	013	213	213	253	253	253	353	373	373	377	377
	001	001	001	011	011	013	013	113	113	133	173	173	173	177	177	377
N=512	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	011	011	013	013	213	213	253	253	253	353	373	373	377	377
	001	001	001	011	011	013	013	113	113	133	173	173	173	177	177	377
N=1024	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	002	002	002	042	052	052	252	252	253	253	253	353	373	373	377	377
	001	001	011	011	011	051	051	071	071	073	173	173	173	177	177	377
N=1024	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	002	002	042	042	043	043	043	243	343	343	363	363	373	373	377	377
	002	002	002	042	042	052	252	252	252	253	253	273	273	373	373	377

Tabela 5.2: Tabela de perfuração do código turbo

ainda tiver uma probabilidade de erro $P_e \geq 10^{-3}$, então mais bits são requisitados através do canal de retorno. Nas simulações pudemos constatar que os planos de bits menos significativos são os responsáveis pelo maior número de requisições, como mostra o gráfico 5.4.

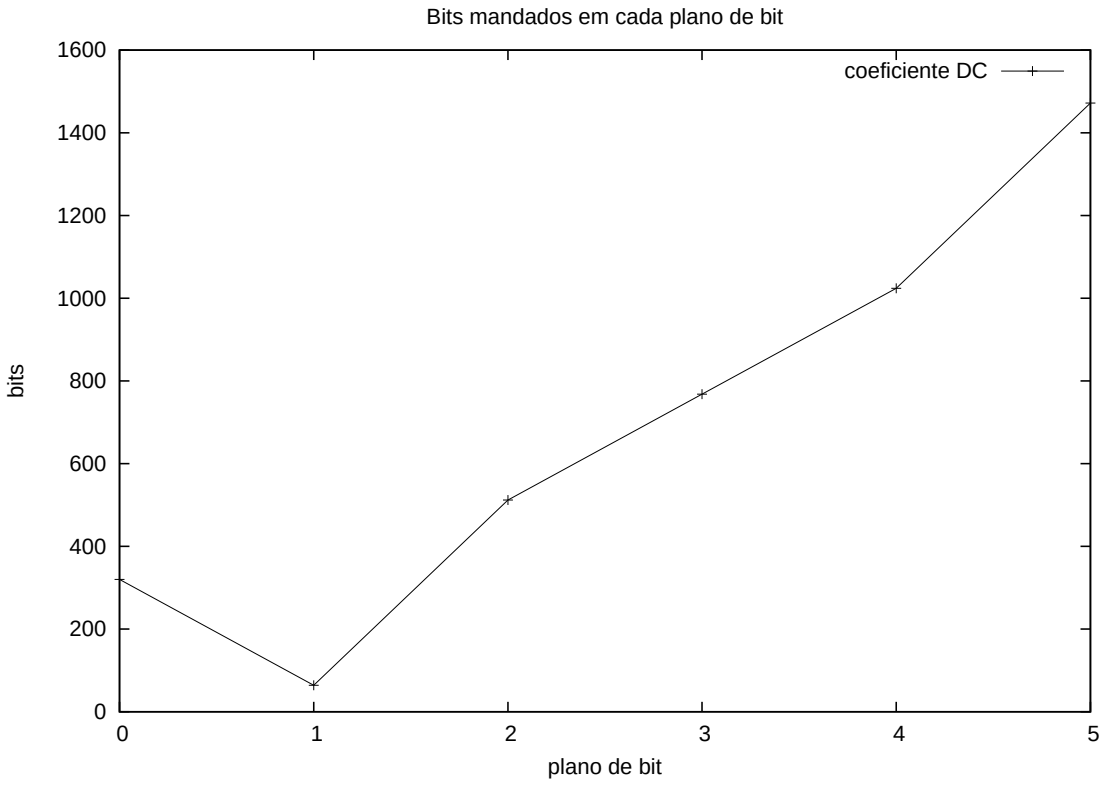


Figura 5.4: Número de bits requisitados de acordo com cada plano de bit

Na mesma maneira que foi feito no capítulo 4, a decodificação de um plano de bit limita o intervalo de integração para o cálculo da probabilidade condicional do próximo plano de bit, como foi descrito na seção 4.3.

Uma diferença entre a implementação do capítulo 4 e a deste capítulo foi o fato do plano de bit a ser transmitido ter sido previamente dividido em pacotes de tamanho 256. Com estes pacotes, a capacidade de correção do código turbo diminuiu, pois o interleaver gera seqüências menos aleatórias, porém com pacotes pequenos as áreas com erros são mais bem localizadas, fazendo com que o decodificador requisite bits somente para os pacotes com erros, diminuindo a taxa total, como mostra o gráfico 5.5.

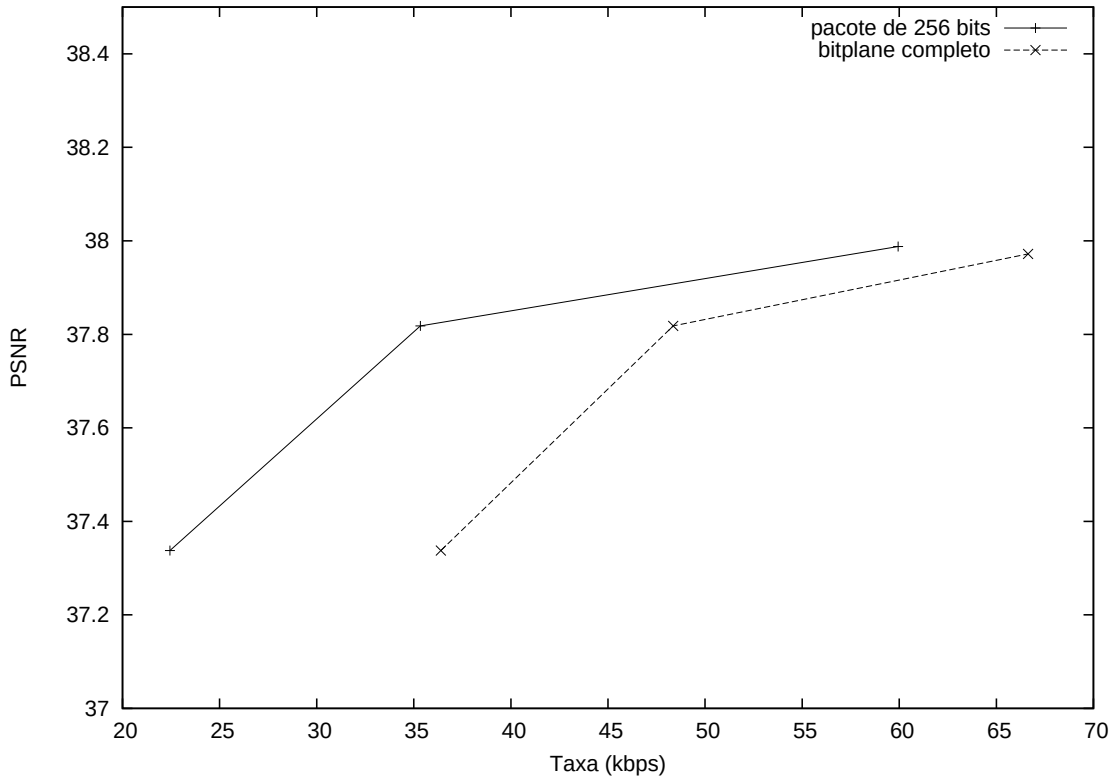


Figura 5.5: Curvas taxa-distorção para codificação do quadro 1 da seqüência *Foreman* usando estimação de movimento, dividida ou não em pacotes de 256 bits

5.5 Controle de Taxa

Assim como mencionado na seção 4.4, o controle de taxa é um elemento crítico para o sistema. Nas nossas simulações variamos alguns parâmetros de codificação para alcançar diversos pontos da curva taxa-distorção, porém o conjunto de parâmetros ideais depende da imagem a ser codificada e principalmente da informação lateral a ser usada.

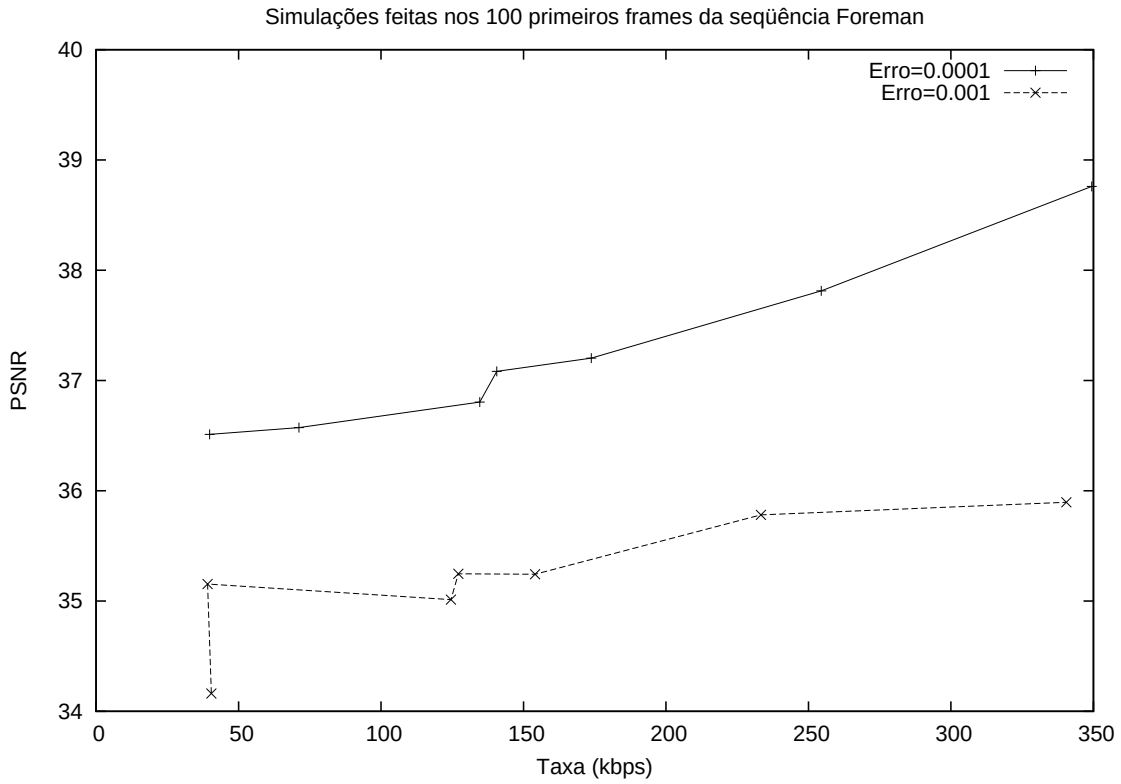


Figura 5.6: Curvas taxa-distorção para diferentes probabilidades de erro aceitáveis

No gráfico 5.6, as curvas mostram o desempenho diferenciado do codificador. Simplesmente tolerando uma probabilidade de erro de 10^{-3} em qualquer *bitplane* acaba por prejudicar a desempenho do codificador. Como a decodificação de um plano de bit leva em consideração os planos de bits anteriormente decodificador, se atribuirmos para todos os planos de bit a mesma importância, teremos os erros tolerados nos primeiros planos sendo propagados para os planos seguintes, e assim obrigando os próximos planos e pedirem mais bits de paridade, para concertarem seus erros. Com isso teremos um sinal degradado devido aos erros nos planos mais significativos, e ainda por cima teremos um aumento da taxa, por conta dos outros planos de bits.

5.6 Resultados e análise

Podemos ver do gráfico em 5.7 que o uso dos coeficientes da transformada gerou um pequeno ganho na seqüência *Foreman*, como postulado em [24]. Um ganho maior foi confirmado com seqüências de pouco movimento, como a seqüência

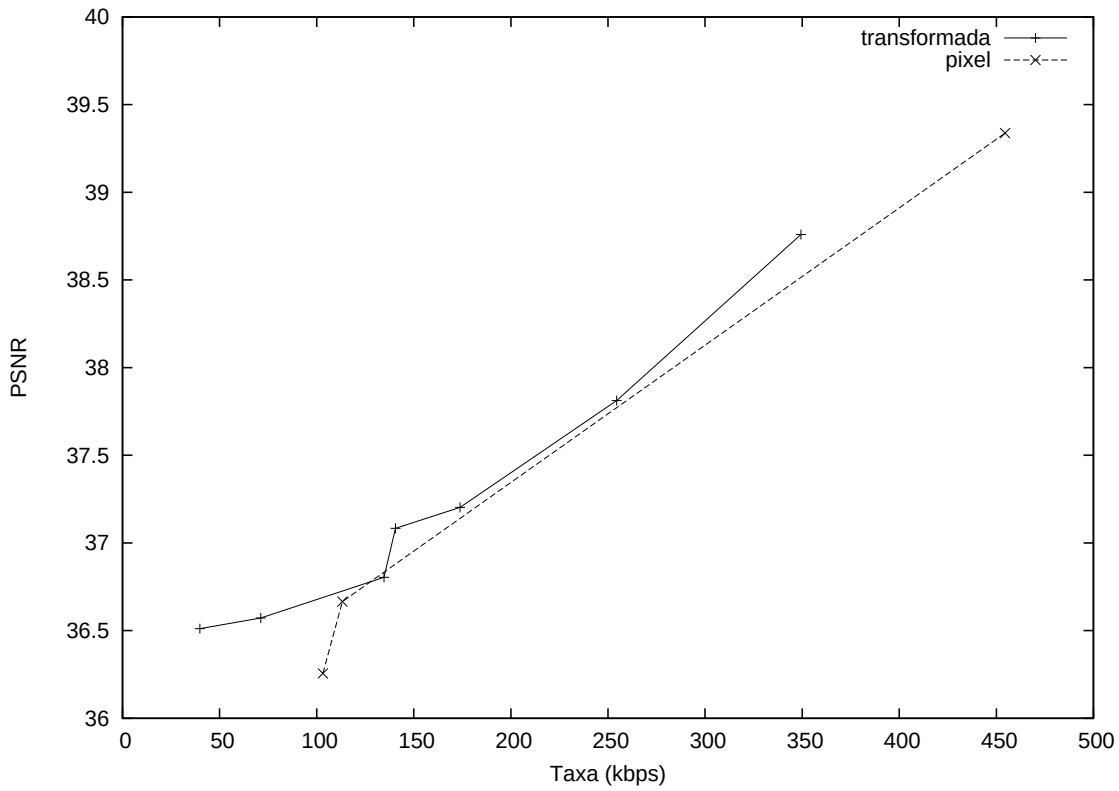


Figura 5.7: Comparação entre os métodos do cap 4 e 5

Mother&Daughter, porém não realizamos simulações com esta seqüência.

Com a implementação do algoritmo proposto pudemos confirmar os resultados publicados em [24], como mostra o gráfico 5.8. Obtivemos apenas valores um pouco piores em altas taxas, o que acreditamos poder estar ligado ao desempenho do algoritmo de controle de taxa, que por não diferenciar os planos de bits a serem codificados, acabou por degradar o sinal, tolerando erros nos primeiros planos de bits, que em seguida foram propagados para os planos seguintes, degradando o sinal e exigindo uma maior taxa. Outra variável do sistema é a obtenção da informação lateral, que como depende da seqüência e dos quadros utilizados para gerá-la, não sabemos como ela foi obtida em [24], o que pode ter influenciado no resultado final.

Nas figuras 5.9 vemos a informação lateral obtida com estimação de movimento e o quadro reconstruído. Note a reconstrução dos detalhes no rosto da figura, perdidos na informação lateral, pois o movimento grande desta seqüência não conseguiu ser modelado corretamente.

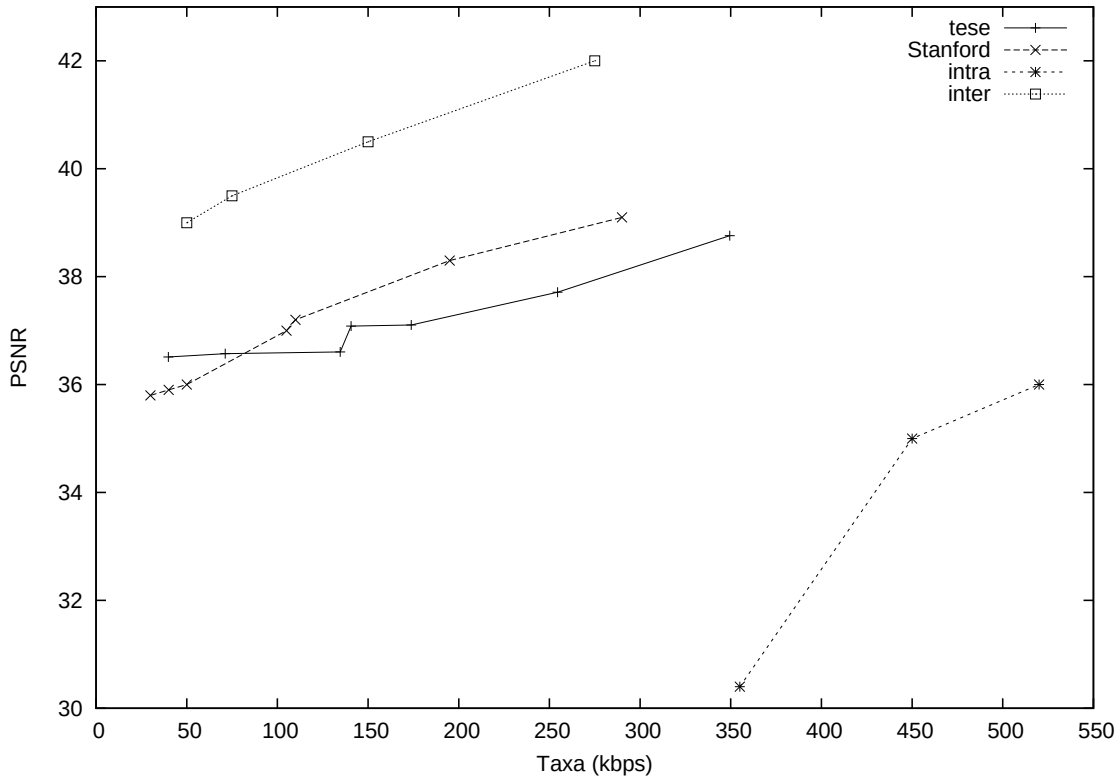


Figura 5.8: Comparação entre as implementações desta tese, do grupo de *Stanford*, com um codificador somente intra ou somente inter

(a) (b)

Figura 5.9: (a) informação lateral (estimação de movimento - 29.436678 dB) (b) quadro reconstruído (35.88360 dB @ 415.56kbps)

Capítulo 6

Codificação de Vídeo Wyner-Ziv por aproximações sucessivas

6.1 Introdução

Nos capítulos 4 e 5 implementamos dois codificadores propostos em [12] e [24]. Neste capítulo iremos propor um método alternativo de codificação distribuída, baseado em quantizadores por aproximações sucessivas.

Técnicas como a transformada *Wavelet* são o estado-da-arte em compressão de imagens. As técnicas mais populares nos dias de hoje são as que usam planos de bits. Os coeficientes são codificados usando aproximações sucessivas : em cada iteração um plano de bits dos coeficientes da transformada é codificado. Essas técnicas tem uma excelente desempenho e são utilizadas nos padrões mais atuais de codificação de imagem, como o JPEG2000 [35] ou o MPEG4 [36].

Como visto em [29], em codificação por planos de bits, um coeficiente $-1 < c < 1$ é representado por uma seqüência $\{s, b_1, b_2, \ldots, b_n, \ldots\}$ tal que

$$c = s \sum_{i=1}^{\infty} b_i 2^{-i}$$

onde $s \in \{-1, 1\}$ representa o sinal de c e $b_i \in \{0, 1\}$. Na prática a soma é truncada numa iteração P quando um determinado critério é atingido (de distorção ou de taxa).

Como mostrado anteriormente, para implementar o paradigma de Wyner-Ziv usamos ferramentas de codificação intra para atingir desempenho de codificadores

inter, e por isso seria interessante usar aproximações sucessivas devido ao seu bom desempenho. Aproximações sucessivas geram dois tipos de informação: o mapa de significância, para localizar os coeficientes mais significativos, e os planos de bits, com os valores dos coeficientes em um determinado plano. Porém, no âmbito Wyner-Ziv, podemos aplicar os conceitos de codificação apenas nos planos de bits, e não nos mapas de significância.

Algoritmos como o EZW (*embedded zerotree wavelet*), o SPIHT (*set partition in hierarchical tree*) ou o MGE (*multigrid embedding encoder*) são propostas de quantizadores escalares por aproximações sucessivas. Em [33], a idéia de quantização dos planos de bits foi generalizada para vetores, com a introdução da quantização vetorial por aproximações sucessivas. A imagem é dividida em vetores, e cada vetor de coeficientes pode ser decomposto numa soma de vetores com magnitude decrescente, como vemos na equação abaixo.

$$c = \sum_{i=1}^{\infty} \vec{u_{n_i}} \alpha^{-i}$$

onde $\vec{u_{n_i}}$ é um vetor de norma unitária que pertence a um dicionário $\mathbf{C_N}$ N-dimensional e $0.5 < \alpha < 1$ é uma constante que depende do dicionário. Também aqui a representação é truncada de acordo com um critério desejado.

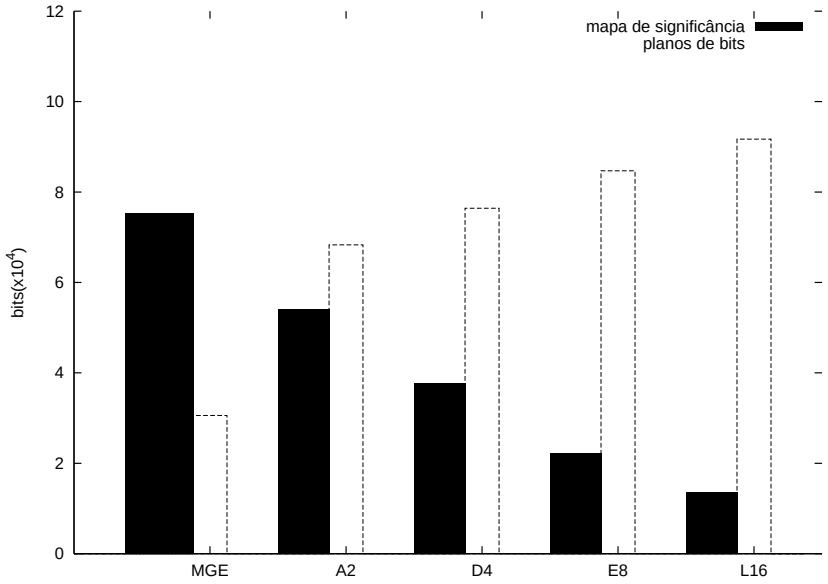


Figura 6.1: Distribuição de bits para codificar a imagem *Lena* usando o método **MGE**

Quantizadores vetoriais representam os planos de bits de um grupo de coeficientes por vetores escolhidos em um dicionário. Os resultados publicados em [30] e reproduzidos na figura 6.1 mostram a distribuição dos bits nas etapas de codificação. Para um codificador escalar, muitos bits são gastos codificando o mapa de significância, isto é, os bits usados para localizar os coeficientes significantes. Já na codificação vetorial, a maioria dos bits são gastos nos planos de bits, para codificar o índice dos vetores do dicionário usados pra quantizar a imagem.Dessa maneira, a quantização vetorial se mostra mais apropriada para ser utilizada no âmbito Wyner-Ziv, uma vez que só iremos codificar os planos de bits, e não o mapa de significância.

Motivados por isso, iremos propomos um codificador Wyner-Ziv baseado em quantização vetorial. Usamos para nossas simulações o algoritmo MGEVQ, devido ao bom desempenho apresentado em [27].

6.2 O quantizador MGEVQ

Para este trabalho, implementamos uma modificação do algoritmo MGEVQ, implementado em [28], derivado do algoritmo MGE, apresentado em [27]. Enquanto que em [28], o algoritmo de quantização vetorial foi feito em cima dos coeficientes da transformada wavelet, neste trabalho nós usamos os coeficientes da transformada DCT para manter a compatibilidade com implementações anteriores. Vamos fazer uma breve descrição do algoritmo.

1. A média da imagem é calculada e subtraída.

2. Aplica-se uma transformada DCT em blocos de tamanho 8×8 e reagrupamos os coeficientes de mesma banda juntos. Em cada banda os coeficientes são agrupados em vetores de forma matricial, dependendo do tamanho do dicionário usado. Por exemplo, se os vetores do dicionário tiverem tamanho 4, então os coeficientes serão agrupados em matrizes 2×2.

3. Acha-se a maior amplitude dos vetores $||\vec{V}||_{max}$

4. O valor de referência inicial l_0 é dado por $\alpha||\vec{V}||_{max}$, onde α depende do dicionário de vetores de orientação que será utilizado

5. Inicializamos uma lista vazia de vetores significantes (SVL)[1]

6. A região que será codificada representa a imagem completa, e é marcada como uma "região zero" ($T = 0$)

7. ***Passo Dominante*** : Para cada "região zero", é enviado para a saída um bit indicando se a região apresenta pelo menos um coeficiente significativo[2] (bit 1) ou não (bit 0). Teremos então as seguintes situações:

   - Se for uma região que contém apenas um vetor significante, então decrementamos o vetor do vetor do dicionário vezes l_0 mais próximo ao vetor em questão. Colocamos o vetor na lista de vetores significantes, e na saída teremos o índice do vetor do dicionário que foi escolhido. Se a região contém mais de um vetor, e pelo menos um deles é significante, então a região é dividida em quatro e cada sub-região passará novamente pelo passo dominante.
   - Se for uma "região zero" nada mais é feito, e passa-se a analisar a região seguinte.

8. ***Passo de refinamento*** : A lista SVL é varrida, e acha-se o vetor do dicionário que, multiplicado por l_n, mais se aproxima de elemento da lista que está sendo analisado. O índice do dicionário vai para a saída, e o vetor é decrementado deste valor novamente.

9. O valor de referência l_n é multiplicado pelo fator de escala α

10. O processo é repetido desde o passo 6, e é interrompido em qualquer ponto quando o tamanho do *bitstream* exceder um limite pré-estabelecido

A figura 6.2 ilustra o método de busca em árvore quaternária (*quadtree*).

---

[1] No algoritmo em [28], são usadas duas listas, uma de vetores significantes e outra de vetores insignificantes. A lista de vetores insignificantes será usada caso a decomposição em *quadtree* não consiga isolar um coeficiente. Como em nossas simulações sempre decompomos os sinais até isolar cada coeficiente em uma folha da estrutura *quadtree*, a presença da lista de vetores insignificantes não é necessária.

[2] O vetor será considerado significativo se $||\vec{V}|| > l_n$, onde l_n é o valor da constante no plano de bit n, que irá multiplicar os vetores do dicionário

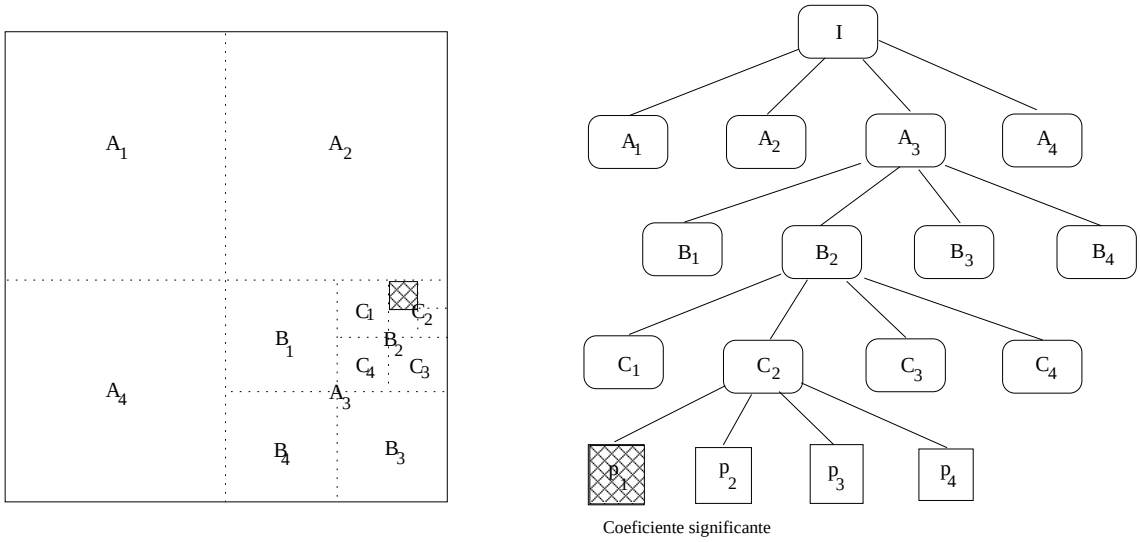


Figura 6.2: Decomposição da imagem na estrutura de árvore quaternária

6.2.1 Escolha do parâmetro α

O parâmetro α irá determinar a convergência da decomposição do sinal em aproximações sucessivas. Pode ser mostrado que se α for escolhido convenientemente, o erro de quantização do processo vai tender a zero a medida que codificamos um maior número de *bitplanes*. Apresentado em [31], as condições para que isso aconteça são

$$\frac{1}{2\cos[\Theta(\mathbf{C}_N)]} \leq \alpha < 1, \quad \Theta(\mathbf{C}_N) \leq 45^\circ \tag{6.1}$$

$$\sin[\Theta(\mathbf{C}_N)] \leq \alpha < 1, \quad \Theta(\mathbf{C}_N) \geq 45^\circ \tag{6.2}$$

onde $\Theta(\mathbf{C}_N)$ é o ângulo máximo entre qualquer vetor do espaço $\mathbb{R}^N$ e o vetor mais próximo no dicionário $\mathbf{C}_N$.

Como o método de quantização apresentado aqui difere do método apresentado em [31], os valores ótimos para α tiveram que ser experimentalmente calculados, e obtivemos os seguintes valores.

Reticulado	α
D_4	0.59
E_8	0.60
L_{16}	0.62

Tabela 6.1: Valores ótimos de α

6.2.2 Escolha do dicionário

A escolha do α está atrelado ao dicionário usado, como vemos nas equações 6.1, portanto devemos também escolher um dicionário que tenha um Θ_{max} pequeno o suficiente para minimizar o erro. A escolha deve ser feita levando em consideração o número de vetores do dicionário, pois ele tem que ter um Θ_{max} pequeno, porém sem aumentar o número total de vetores no dicionário, para não aumentar o número de bits necessários para mandar um índice do dicionário. Isso sugere que o dicionário $\mathbf{C}_N$ tenha vetores distribuídos de maneira mais uniforme possível na hiperesfera N-dimensional de raio unitário.

A resposta para esse problema são as distribuições lattice regulares para dimensão 4, 8 ou 16. Como mostrado no gráfico 6.1, obtivemos uma economia grande nos bits gastos com o mapa de significância e uma melhora na desempenho do codificador. Porém aumentar a dimensionalidade do dicionário significa também aumentar a dimensionalidade do código turbo, como veremos a seguir, o que acarreta um aumento da complexidade no processo de decodificação. Nesta tese, nos limitamos a usar o dicionário D_4 (dimensão 4), para facilitar a análise.

6.3 Descrição do codificador

O diagrama em blocos do codificador proposto pode ser visto na figura 6.3. O processo de codificação é feito da seguinte forma:

- A imagem é quantizada conforme o algoritmo descrito em 6.2.
- O mapa de significância gerado pela imagem é transmitido diretamente para o decodificador, enquanto que os planos de bits são concatenados para formar um fluxo, que será a entrada do código turbo.
- Os bits significativos não são mandados, somente mandaremos um subconjunto da paridade, determinado por um algoritmo de perfuração, dependendo das requisições feitas pelo decodificador.
- O decodificador, de posse do mapa de significância da imagem, irá usá-lo para gerar os planos de bits com a informação lateral. Isto é, o mapa de

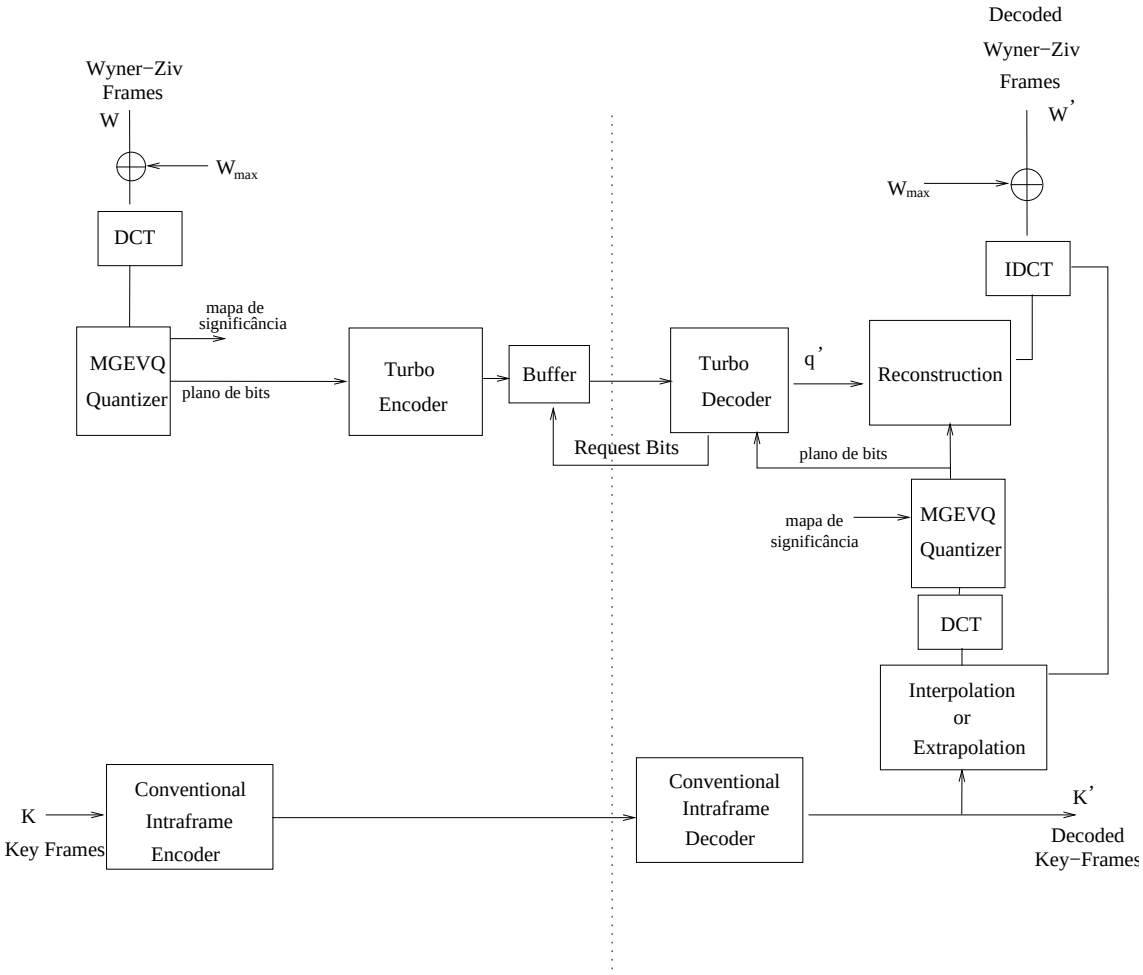


Figura 6.3: Codificador Wyner-Ziv por aproximações sucessivas

significância indicará na informação lateral a posição dos vetores significantes, e o decodificador irá obter os coeficientes deste plano usando esses vetores, para assim formar o fluxo de planos de bits concatenados, e substituir o fluxo usado no encoder.

- Após decodificarmos o plano de bits corretamente, iremos reconstruir a imagem com o plano de bits, o mapa de significância e a informação lateral.

6.4 O código turbo

Como mostrado em [30] e ilustrado na figura 6.1 , a maioria dos bits gastos na quantização **MGEVQ** é com a codificação dos coeficientes, e apenas uma pequena parte destes bits é gasta codificando a estrutura de árvore (*quadtree* ou mapa de

significância). Com isso em mente, propomos que apenas os vetores de coeficientes sejam codificados pelo codificador Slepian-Wolf. Assumindo uma alta correlação entre os coeficientes significativos da entrada e os coeficientes da informação lateral na mesma posição, poderemos atingir a compressão que outras técnicas alcançaram, seguindo o mesmo preceito de mandar apenas um conjunto de bits dos vetores de coeficientes, e decodificá-los com a ajuda da informação lateral.

A entrada do código turbo foi a lista de todos os índices escolhidos para representar os vetores de coeficientes, com cada plano de bit seguido do outro. Sendo a entrada escolhida desta forma, isso acabou por limitar o dicionário usado nas simulações. Dicionários com mais vetores, como os analisados em [31], apesar da vantagem de concentrarem o número de bits gastos nos coeficientes, e não na estrutura de árvore, acabam por exigir códigos turbo mais complexos. Por esse motivo, em nossas simulações, nos limitamos ao dicionário D_4.

A probabilidade de transição ($\gamma(s', s)$) foi calculada de acordo com a distância entre o coeficiente do plano de bit do vetor da imagem real, escolhido no processo de quantização, ao coeficiente do mesmo plano de bit do vetor da informação lateral. Uma função laplaciana foi usada para modelar a correlação entre os dois coeficientes, e para o fator de decaimento desta laplaciana (identificado também como α nos capítulos anteriores) foi usado o mesmo valor estimado no capítulo 4.

Tivemos também que organizar os índices dos vetores, de modo a atribuir a um vetor com um determinado peso o seu índice equivalente. Desta maneira os índices, que são a entrada do código turbo, irão também corresponder a distância entre os vetores do dicionário. A tabela 6.2 mostra como ficaram os vetores do dicionário D_4.

Para a implementação do código turbo, usamos o polinômio $[25, 37, 27, 31, 23, 35]$, apresentado em [37], por ter a maior distância espectral e ser um polinômio ótimo para altas taxas, segundo critérios também apresentados em [37]. Em nossas simulações, usamos 15 iterações durante a decodificação. Como mostra a figura 6.4, com 15 iterações temos o algoritmo convergindo numa taxa menor (com menos bits) do que com 5 iterações, isto é, o desempenho do código turbo com 15 iterações é melhor do que com 5 iterações.

O código turbo foi perfurado usando as mesmas tabelas de 5.2 e 4.2. A

índice	vetor
0	[0 0 1 1]
1	[-1 0 0 1]
2	[0 -1 0 1]
4	[0 0 1 -1]
5	[-1 0 1 0]
6	[0 -1 1 0]
8	[1 1 0 0]
9	[0 1 0 1]
10	[0 1 1 0]
12	[1 -1 0 0]
13	[1 0 0 1]
14	[1 0 1 0]
16	[0 0 -1 -1]
17	[1 0 0 -1]
18	[0 1 0 -1]
20	[0 0 -1 1]
21	[1 0 -1 0]
22	[0 1 -1 0]
24	[-1 -1 0 0]
25	[0 -1 0 1]
26	[0 -1 -1 0]
28	[-1 1 0 0]
29	[-1 0 0 -1]
30	[-1 0 -1 0]
31	[0 0 0 0]

Tabela 6.2: Vetores do dicionário D_4

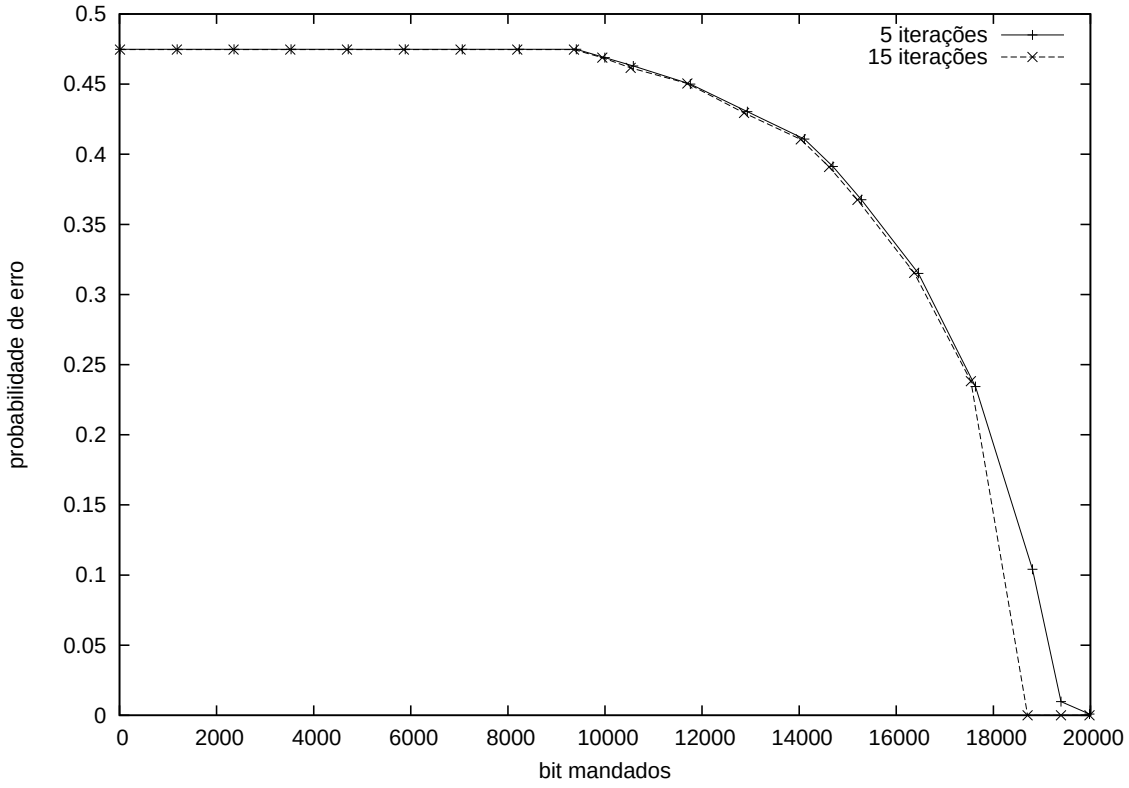


Figura 6.4: Convergência do código turbo

perfuração dos bits de paridade seguiu as tabelas, já os bits sistemáticos não foram mandados a princípio, sendo usado no seu lugar o símbolo gerados pela informação lateral. A informação lateral era quantizada seguindo o algoritmo MGEVQ, porém o mapa de significância usado era o mesmo da imagem codificada. A lista de índices que a informação lateral gerou serviu para substituir os bits sistemáticos, e assim suprimí-los da transmissão. Como o desempenho do decodificador não nos permitiu chegar a uma probabilidade de erro aceitável, resolvemos então mandar também alguns bits sistemáticos para auxiliar na decodificação.

6.5 O processo de reconstrução

No processo de reconstrução, podemos usar a informação lateral para melhorar a relação sinal-ruído após decodificarmos os coeficientes com o código turbo. Cada plano de bit representará um vetor, e a soma deles irá se aproximando do

vetor real, como mostra a figura 6.5. Ao final do passo n, estaremos no máximo a

$$|| \sum_{i=n+1}^{\infty} \alpha^n \vec{V}_{max}|| \leq \alpha^n ||\vec{V}_{max}|| \tag{6.3}$$

distante do vetor real. Podemos então formar uma hiperesfera, e considerar três opções: não usar a informação lateral, usar a informação lateral presente dentro da hiperesfera ou usar a informação lateral dentro e fora da hiperesfera.

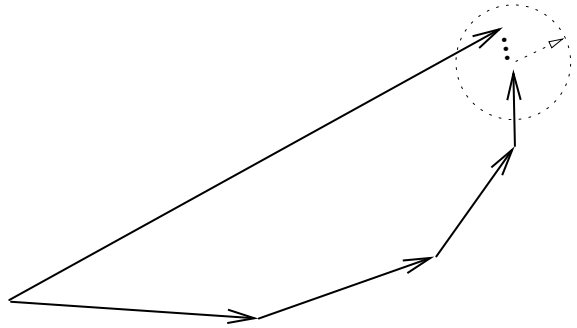


Figura 6.5: Reconstrução de um vetor pelo método MGEVQ

A reconstrução considerando a informação lateral será feita conforme a localização desta. Se ela se encontrar dentro da hiperesfera, então temos uma informação lateral bem confiável, e portanto assumimos o seu valor. Se a informação lateral não se encontra dentro da hiperesfera, então a reconstrução assume o valor da borda da hiperesfera, e dessa maneira também iremos evitar grandes erros positivos ou negativos, limitando o erro de reconstrução ao tamanho do raio da hiperesfera.

A taxa alvo com a qual será codificada a imagem influencia diretamente no tamanho do raio. Se tivermos uma taxa alta, então provavelmente vários planos de bits serão codificados, resultando em um pequeno raio ao final da quantização, já se tivermos uma taxa baixa, o processo de quantização para enquanto o raio ainda tem um valor significativo. Por causa disto, fizemos simulações com diversas taxas, para podermos medir o efeito de usarmos a informação lateral na reconstrução da imagem, e tivemos os seguintes resultados.

Em taxas baixas (figura 6.6), o uso da informação lateral melhora a relação sinal-ruído, pois o valor da reconstrução, por ter usado poucos planos de bits, ainda está distante do valor real. O uso da informação lateral dentro e fora da esfera auxilia na reconstrução, pois a informação lateral é em geral uma boa estimativa do vetor

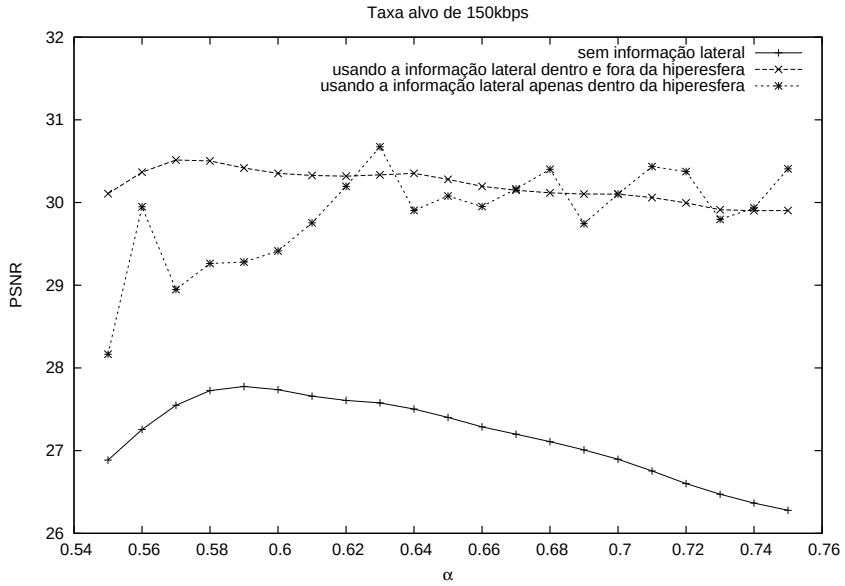


Figura 6.6: PSNR dos quadros da seqüencia *Foreman* usando ou não a informação lateral para uma taxa de 0,1 bit/pixel

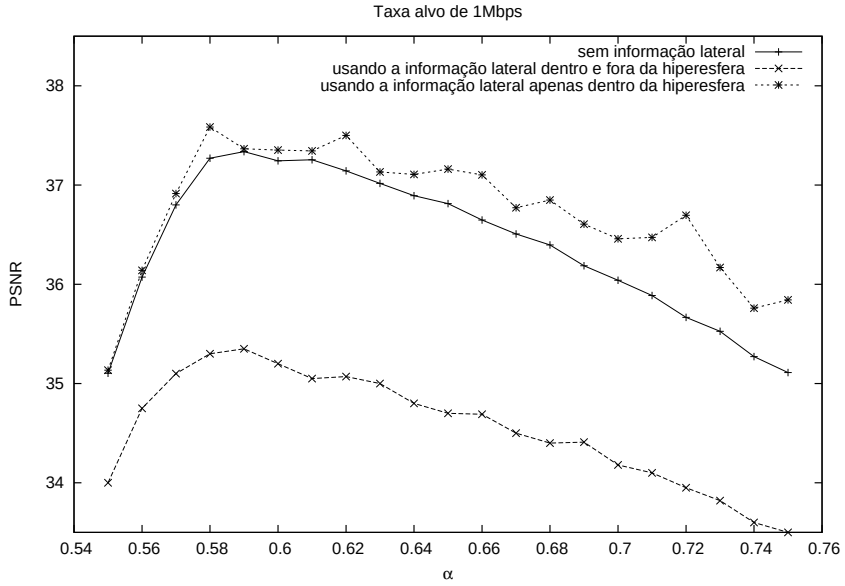


Figura 6.7: PSNR dos quadros da seqüencia *Foreman* usando ou não a informação lateral para uma taxa de 1 bit/pixel

real. Quando o α é pequeno, o raio final é menor, e assim a informação lateral que não é muito confiável fica fora da hiperesfera e não é usada na reconstrução. Porém usar o centro da hiperesfera para a reconstrução da imagem é pior do que usar a superfície desta indicada pela informação lateral, como podemos ver no gráfico. Já

quando o α é maior, o raio final é grande o suficiente para englobar boa parte da informação lateral, e a reconstrução também já tem um valor grande o suficiente para se aproximar do valor real.

Já nas taxas altas (figura 6.7), podemos notar que o uso da informação lateral dentro e fora da hiperesfera degrada a imagem reconstruída. Uma vez que o raio final é bem pequeno, e o processo de aproximação já se encontra bem próximo do valor real, se considerarmos a informação lateral fora da hiperesfera, estaremos considerando uma informação lateral não-confiável, e por isso teremos uma degradação no sinal. Já considerando somente a informação lateral dentro da hiperesfera, estaremos considerando a informação lateral confiável, e dessa maneira selecionamos somente a informação lateral que realmente interessa, e teremos um ganho na reconstrução da imagem por usar esta informação lateral.

6.6 Resultados e análise

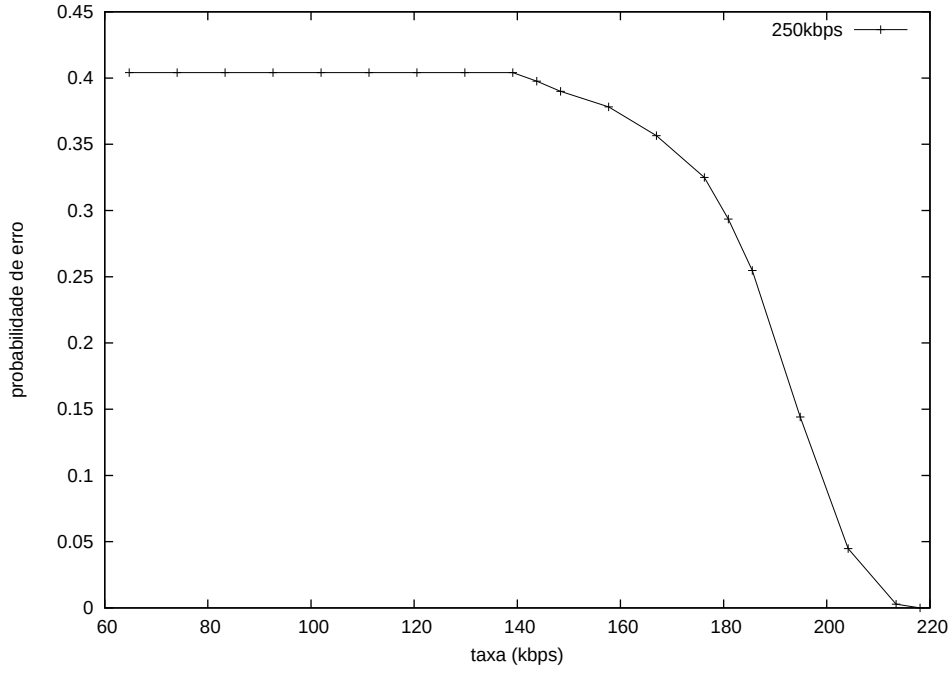


Figura 6.8: Curva da probabilidade de erro da seqüência *Foreman* para uma taxa-alvo de 250kbps

Os gráficos 6.8 e 6.9 mostram o desempenho do esquema de codificação proposto nesta tese. A taxa-alvo indica a taxa no qual a imagem foi codificada, no

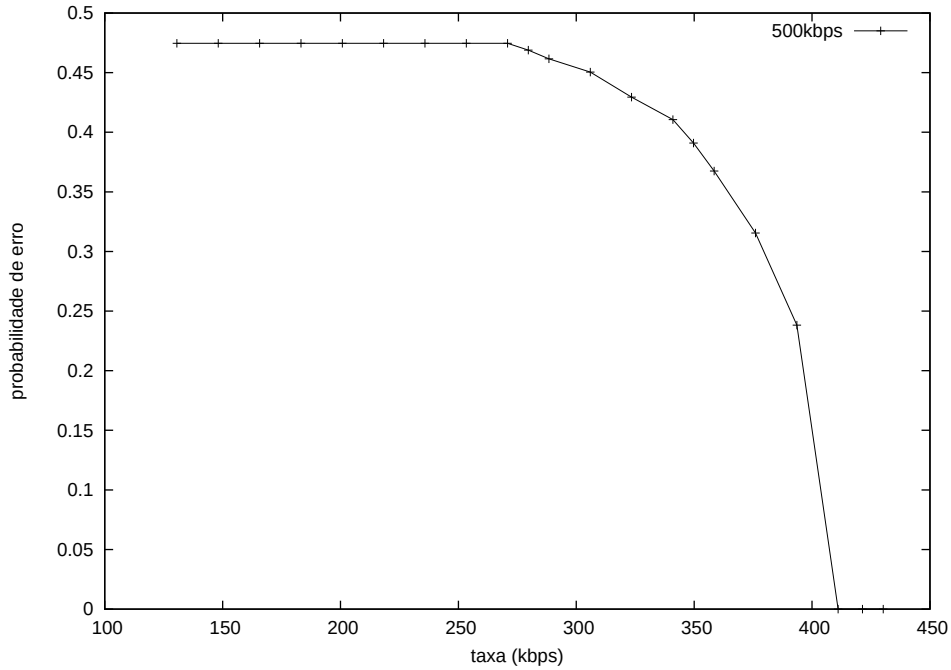


Figura 6.9: Curva da probabilidade de erro da seqüência *Foreman* para uma taxa-alvo de 500kbps

eixo x temos a taxa de bits efetivamente mandados, o que inclui os bits do mapa de significância mais os bits usados pelo decodificador turbo para recuperar os coeficientes, enquanto que no eixo y temos a probabilidade de erro. Tivemos compressão na medida que o número de bits mandados foi menor do que a taxa usada para codificar o quadro. Note que com uma taxa-alvo de 250kbps, com aproximadamente 218kbps atingimos uma probabilidade de erro nula, isto é, todos os bits dos planos de bits foram decodificados corretamente, e tivemos uma pequena compressão.

As curvas de 6.10 mostram o desempenho do codificador proposto relativo a um codificador intra apenas, e a um codificador inter (H.263+). Vemos que tivemos um desempenho razoável em taxas baixas, porém um resultado muito ruim para taxas mais altas. O ganho de codificação não se mostrou satisfatório, e a causa disto atribuímos a dois fatos: termos usado a informação lateral quantizada e a correlação entre os coeficientes gerados pela informação lateral e os coeficientes da imagem a ser transmitida.

As figuras 6.11 e 6.12 mostram a correlação entre os coeficientes gerados pela informação lateral e os coeficientes da imagem a ser transmitida, de ambas as imagens já quantizadas. Note que para taxas pequenas, esses coeficientes são

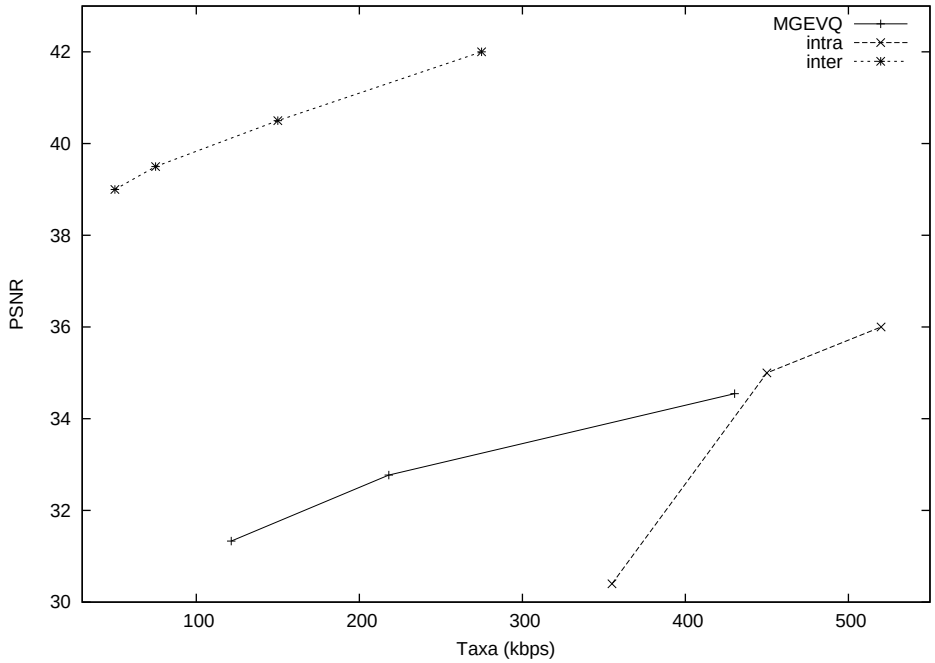


Figura 6.10: Desempenho do codificador proposto

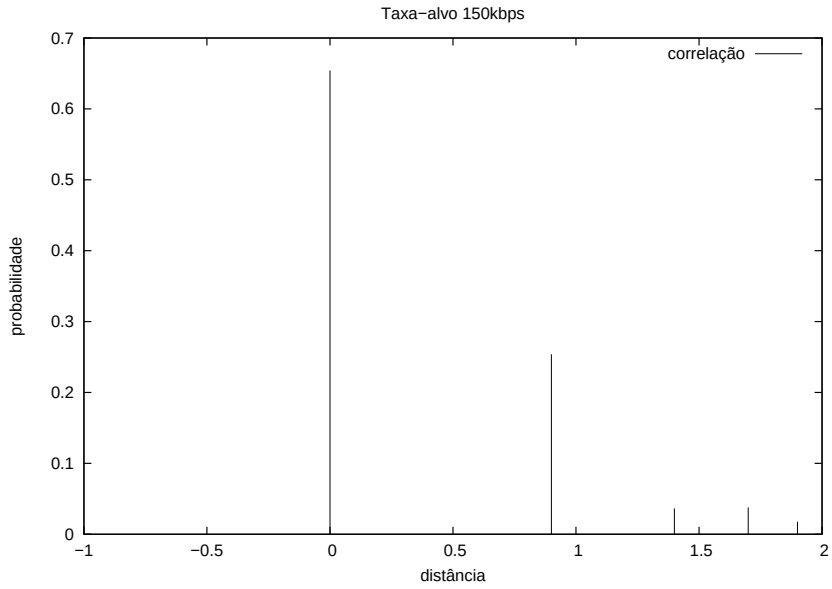


Figura 6.11: Correlação entre os coeficientes a serem transmitidos e os coeficientes gerados pela informação lateral para uma taxa-alvo de 150kbps

bem correlacionados, o que justifica o ganho verificado para estas taxas. Com taxas pequenas, apenas os primeiros planos de bits são codificados, e como já visto em outras implementações (capítulo 5, seção 5.5), estes primeiros planos de bits são

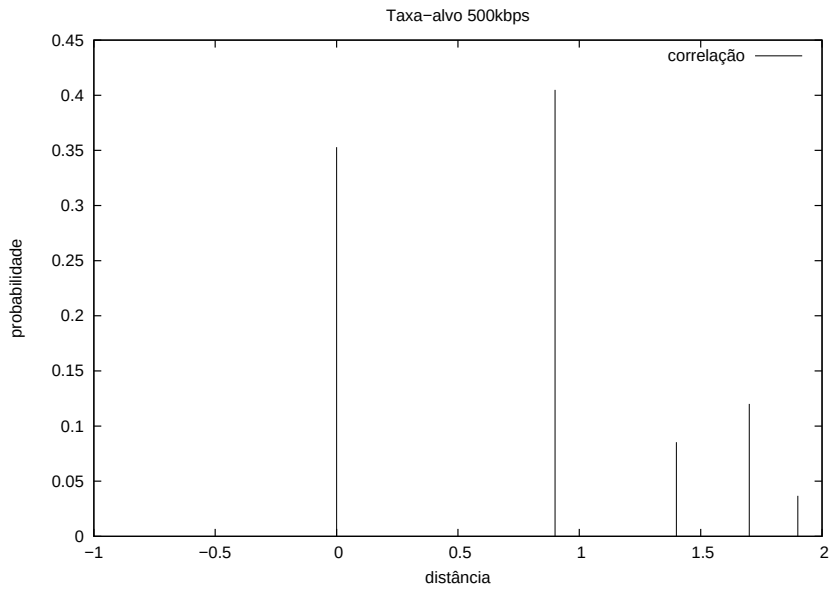


Figura 6.12: Correlação entre os coeficientes a serem transmitidos e os coeficientes gerados pela informação lateral para uma taxa-alvo de 500kbps

realmente mais correlacionados com a informação lateral. Já em taxas altas, mais planos de bits são codificados, e como não foi feita distinção alguma entre os planos de bits, a correlação dos planos menos significativos acaba por afetar o desempenho geral do codificador, verificado no gráfico 6.10.

Como a informação lateral pode não ser confiável em alguns pontos, a codificação dela irá gerar um ruído para o código turbo, não auxiliando em nada no processo de decodificação. O código turbo deveria ter uma paridade apropriada para corrigir esses erros, que aumentam com o número de planos de bits mandados, logo uma relação entre o número de planos de bits codificados e a paridade deveria ser derivada desse esquema de codificação. O cálculo da probabilidade de transição deverá levar em conta a informação lateral sem quantização, como foi feito nos capítulos 4 e 5, porém devido à complicação do algoritmo e falta de tempo, não conseguimos implementar esta idéia para a tese. Vamos então descrever os passos para uma futura implementação de como deveria ser o esquema de codificação distribuída com quantização vetorial por aproximações sucessivas.

- A imagem é quantizada, da mesma maneira como foi descrito anteriormente
- A codificação turbo deverá ser feita em planos de bits separados um do outro,

isto é, cada plano de bit, com um número variável de entradas, deverá ser codificado independentemente.

- A decodificação deverá ser feita considerando a informação lateral sem quantizá-la. A probabilidade de transição será dada pela integral da função de probabilidade condicional da imagem com a informação lateral na hiperesfera de raio $\alpha^n||\vec{V}_{max}||$, onde a área dessa hiperesfera são os limites desta integral. Devido à dificuldade de realizar esta integral, propomos uma simplificação onde usaremos apenas a diferença do módulo destes vetores (o vetor reconstruído e a informação lateral sem quantização alguma), num intervalo delimitado por $2\alpha^n||\vec{V}_{max}||$ e centrado no valor de reconstrução atual da amplitude do vetor, o que seria a projeção deste problema N-dimensional numa solução unidimensional.

- Cada plano de bit decodificado iria realimentar o decodificador, sendo usado no cálculo das probabilidades de transição, na reconstrução do vetor codificado.

- Após a decodificação correta destes planos de bits, a reconstrução da imagem será feita com o auxílio do mapa de significância, os planos de bits e a informação lateral

Desta maneira acreditamos superar os problemas encontrados na primeira implementação. Com isso sugerimos também que uma análise seja feita do código turbo, para que possamos usar uma dimensionalidade maior no dicionário, pois como vimos dicionário de dimensionalidade maior são mais apropriados para o esquema proposto de codificação, por concentrarem a informação nos planos de bits, e não nos mapas de significância.

Capítulo 7

Conclusões

Nesta tese, alguns resultados em codificação distribuída foram apresentados. Além de implementar dois algoritmos publicados anteriormente, uma nova maneira de codificar foi proposta, utilizando quantizadores por aproximações sucessivas.

A análise das simulações (feitas somente com a seqüência *Foreman*, devido a falta de tempo) mostrou que podemos fazer codificadores simples, mas decodificadores complexos, e assim nos aproximarmos das taxas de compressão dos algoritmos de hoje em sistemas de recursos limitados, como previa Wyner e Ziv.

Apesar dos resultados do método proposto não terem sido satisfatórios, muito trabalho ainda pode ser feito sobre este tema. Uma análise detalhada do codificador turbo deverá ser realizada. Propomos no capítulo 6 uma maneira de usarmos a informação lateral no processo de decodificação, porém esta não se mostrou eficiente, por termos usado a informação lateral quantizada. Vemos então que o desempenho do código turbo é uma das peças fundamentais para o bom desempenho do processo de codificação, e a continuação deste trabalho poderá ser feita no sentido de melhorar a desempenho do código turbo, levando em conta a informação lateral sem quantização.

Para implementações futuras sugerimos também o uso de outros dicionários, pois como já mencionado, quanto maior a dimensionalidade do dicionário, maior a concentração de bits nos coeficientes. Outra sugestão é o uso da diferença do quadro a ser transmitido com o quadro anterior, no lugar do canal de realimentação, como outras implementações ([14] e [15]) fizeram.

O tema da obtenção da informação lateral no decodificador merece atenção

especial, uma vez que a qualidade de informação lateral influencia no desempenho do codificador. Algumas técnicas foram apresentadas neste trabalho, porém vimos que a análise destas técnicas é muito dependente do tipo de seqüência a ser codificada. Diferentes técnicas de estimação de movimento e interpolação ainda precisam ser testadas com diferentes seqüências para que possamos chegar a uma conclusão definitiva sobre o assunto.

Referências Bibliográficas

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

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

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

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

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

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

[7] T. M. Cover e J. A. Thomas, *Elements of Information Theory*. A Wiley-Interscience Publication, 1991.

[8] D. Slepian e J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 471-480, Julho 1973.

[9] A. D. Wyner e J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 1-10, Janeiro 1976.

[10] A. D. Wyner, "On source coding with side-information at the decoder," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 294-300, Maio 1975.

[11] P. Ishwar, V. M. Prabhakaran e K. Ramchandran, "Towards a theory for video coding using distributed compression principles," *Proc. IEEE International Conference on Image Processing*, pp. 687-690, Março 2003.

[12] A. Aaron, E. Setton e B. Girod, "Towards Pratical Wyner-Ziv Coding of Video," *Proc. IEEE International Conference on Image Processing*, pp. 869-872, Março 2003.

[13] A. Aaron e B. Girod, "Compression with side information using turbo codes," *Proc. IEEE Data Compression Conference*, p. 252-261, Abril 2002

[14] R. Puri e K. Ramchandran, *"PRISM: A Video Coding Architecture Based on Distributed Compression Principles,"* In: Tech. Rep. No. UCB/ERL M03/6, ERL, UC Berkeley, Março 2003.

[15] A. Sehgal, A. Jagmohan, N. Ahuja, "A State-Free causal video encoding paradigm" *Proc. IEEE International Conference on Image Processing*, pp. 605-608, Março 2003.

[16] A. AAron, S. Rane, R. Zhang e B. Girod, "Wyner-Ziv coding for video: Applications to compression and error resilience," *Proc. IEEE Data Compression Conference*, pp. 93-102, Março 2003

[17] A. Aaron, S. Rane e B. Girod, "Systematic Lossy forward error protection for video waverforms," *Proc. IEEE International Conference on Image Processing*, pp. 609-612, Março 2003

[18] S. Rane, A. Aaron e B. Girod, "Systematic lossy forward error protection for error resilient digital video broadcasting," *Proc. SPIE Visual Communication and Image Processing*, pp. 3101-3104, Janeiro 2004

[19] S. Pradhan, J. Chou e K. Ramchandran, "Duality between source coding and channel coding and its extension to the side information case," *IEEE Transaction on Information Theory*, vol. 49, no. 6, pp. 1181-1203, Maio 2003

[20] S. Pradhan, J. Chou e K. Ramchandran, "Turbo and Trellis-Based Constructions for Source Coding with Side Information," *IEEE Computer Society DL*, pp. 33-42, Março 2003

[21] S. Pradhan e K. Ramchandran, "Distributed Source Coding Using Syndromes (DISCUS): Design and Construction," *IEEE Transaction on Information Theory*, vol. 49, no. 3, pp. 626-643, Março 2003

[22] A. Majumdar e K. Ramchandran, "Video Multicast over lossy channels based on distributed source coding," *Proc. IEEE International Conference on Image Processing*, pp. 3093-3096, Março 2004

[23] B. Girod, Anne Aaron e David Rebollo-Monedero, "Distributed Video Coding," *Proc. IEEE, Special Issue on Advances in Video Coding Delivery*, vol. 93, no. 1, pp. 71-83, Janeiro 2005

[24] A. Aaron, S. Rane, E. Setton e B. Girod, "Transform-Domain Wyner-Ziv Codec for Video," *Proc. SPIE Visual Communication and Image Processing*, pp. 520-528, Janeiro 2004

[25] D. Rowitch e L. Milstein,"On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo codes," *IEEE Transactions on Communications*, vol. 48, no. 6, pp. 948-959, Junho 2000

[26] J. Ascenso, C. Brites, F. Pereira, "Improving frame interpolation with spatial motion smoothing for pixel domain distributed video coding," *5th EURASIP Conference on Speech and Image Processing, Multimedia Communications and Services*, Junho 2005

[27] T. Lan, A. H. Tewfik, "Multigrid Embedding (MGE) Image Coding,." *Proc. IEEE International Conference on Image Processing*, pp. 369-373, Março 1999

[28] L. Feio, "Codificadores de imagens usando plano de bits generalizados," Tese de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, Dezembro 2002

[29] D. A. Fonini Júnior , "Quantização por aproximações sucessivas," Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, Março 2003

[30] L. Feio e E. A. B. Silva, "Comparative analysis of bitplane-based wavelet image coders," *IEE Proc. Image Signal Processing*, vol. 151, no. 2, pp. 109-118, Abril 2004

[31] M. Craizer, E. A. B. Silva e E. G. Ramos, "Convergent algorithms for successive approximation vector quantisation with applications to wavelet image compression," *IEE Proc. Image Signal Processing*, vol. 146, no. 3, pp. 159-164, Junho 1999

[32] D. G. Sampson, E. A. B. Silva e M. Ghanbari, "Low bit-rate video coding using wavelet vector quantisation," *IEE Proc. Vis. Image Signal Processing*, vol. 142, no. 3, pp. 141-148, Junho 1995

[33] D. G. Sampson, E. A. B. Silva e M. Ghanbari, "A successive approximation vector quantizer for wavelet transform image coding," *IEEE Trans. Image Process.*, vol. 5, no. 2, pp. 299-310, Fevereiro 1996

[34] Joint Video Team (JVT) of ISO/IEC MPEG e ITU-T VCEG, "Joint Model Reference Encoding Methods and Decoding Concealment Methods," Setembro 2003

[35] ISO/IEC JTC1/SC29/WG1, "JPEG2000 Verification Model 8.6," 2000

[36] ISO/IEC JTC1/SC29/WG11, "MPEG-4 Video Verification Model Version 8.0," Julho 1997

[37] A.G. i Amat, G. Montorsi e S. Benedetto, "Design and Decoding of Optimal High-Rate Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 50, no. 5, pp. 867-881, Maio 2004

Apêndice A

Resultados importantes de teoria de informação

A.1 Definições

Seguem algumas definições importantes que foram utilizadas neste texto, para um melhor entendimento do tema. Para maiores informações, reportamos ao livro [7].

- ✓ Entropia : $H(X) = \sum_{x \in \mathscr{X}} p(x) \frac{1}{p(x)}$
- ✓ Entropia Conjunta : $H(X,Y) = \sum_{x \in \mathscr{X}} \sum_{y \in \mathscr{Y}} p(x,y) \frac{1}{p(x,y)}$
- ✓ Entropia Condicional : $H(X|Y) = \sum_{y \in \mathscr{Y}} \sum_{x \in \mathscr{X}} p(x,y) \log \frac{1}{p(x|y)}$
- ✓ Regra da Cadeia : $H(X,Y) = H(X) + H(Y|X)$
- ✓ Informação Mútua : $I(X;Y) = \sum_{x \in \mathscr{X}} p(x) \sum_{y \in \mathscr{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$
- ✓ Inequação de processamento de dados : Se $X \to Y \to Z$ formam uma cadeia de Markov, então $I(X;Y) \geq I(X;Z)$
- ✓ Inequação de Fano : $H(P_e) + P_e \log(|\mathscr{X} + 1|) \geq H(X|Y)$
- ✓ Lei dos grandes números : $\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} X_i = E[X]$

✓ Propriedade de partição assintótica (**AEP**) : Se X_i são variáveis i.i.d. com probabilidade $p(x)$, então $-\frac{1}{n}\log\frac{1}{p(X_1,X_2,\ldots X_n)} \rightarrow H(X)$

✓ Conjunto típico $A_\epsilon^{(n)}$: com respeito a $p(x)$, é o conjunto de seqüências $(x_1, x_2, \ldots x_n) \in \mathscr{X}^n$ com a seguinte propriedade : $2^{-n(H(X)+\epsilon)} \leq p(x_1, x_2, \ldots x_n) \leq 2^{-n(H(X)-\epsilon)}$

✓ Capacidade do canal : $C = \max \sum_{p(x)} I(X;Y)$

✓ Função taxa-distorção : $R(D) = \min_{p(\hat{x}|x):\sum_{(x,\hat{x})} p(x)p(\hat{x}|x)d(x,\hat{x})\leq D} I(X;\hat{X})$

A.1.1 Construção de códigos usando conjuntos aleatórios

Para construir códigos realizáveis usando conjuntos, devemos atribuir a cada seqüência da fonte um índice aleatoriamente grande. Se o conjunto de seqüências típicas for pequeno o suficiente, então, com alta probabilidade, seqüências distintas terão índices distintos. O processo de decodificação irá receber o índice do conjunto, e irá decodificar a seqüência correta se neste conjunto tivermos apenas uma única seqüência típica, do contrário teremos um erro. Porém se tivermos um número maior de índices do que de seqüências típicas, a probabilidade deste evento será muito pequena. Se a seqüência decodificada não for típica, também teremos um erro. Vamos calcular agora a probabilidade de erro deste código.

Seja $f(X^n)$ o índice correspondente a seqüência X^n. Seja g a função decodificadora. A probabilidade de erro pode ser calculada da seguinte forma.

$$\begin{aligned}
P_e &= P(g(f(X)) \neq X) \\
&\leq P(X \notin A_\epsilon^{(n)}) + \sum_x P(\exists\, x' \neq x : x' \in A_\epsilon^{(n)},\, f(x') = f(x))p(x) \\
&\leq \epsilon + \sum_x \sum_{\substack{x'\neq x \\ x'\in A_\epsilon^{(n)}}} P(f(x') = f(x))p(x) \\
&\leq \epsilon + \sum_x \sum_{x'\in A_\epsilon^{(n)}} 2^{-nR} p(x) \\
&\leq \epsilon + \sum_{x'\in A^{(n)}} 2^{-nR} \sum_x p(x) \\
&\leq \epsilon + \sum_{x'\in A^{(n)}} 2^{-nR} \\
&\leq \epsilon + 2^{-nR} \cdot 2^{n(H(X)+\epsilon)} \leq \epsilon + 2^{n(H(X)-R+\epsilon)}
\end{aligned}$$

$$\therefore R > H(X) + \epsilon \tag{A.1}$$

Logo, se a taxa do código for maior do que a entropia, então o código será unicamente decodificável. Note que o esquema de divisão em conjuntos não requer uma caracterização explícita do conjunto típico do lado do codificador, mas somente do lado do decodificador.

A.2 Provas de teoremas importantes de teoria da informação

Nesta seção, iremos detalhar algumas provas de teoremas mencionados no capítulo 2.

Prova do Teorema 2.2.1 (**Teorema de Slepian-Wolf**): Iremos particionar o espaço $\mathscr{X}^n$ em 2^{nR_1} conjuntos, assim como o espaço $\mathscr{Y}^n$ em 2^{nR_2} conjuntos. Distribuímos cada $x \in \mathscr{X}^n$ nos 2^{nR_1} conjuntos de maneira uniforme, e fazemos o mesmo com cada $y \in \mathscr{Y}^n$, distribuindo nos 2^{nR_2} conjuntos. Teremos então as seguintes funções de mapeamento, conhecidas tanto pelo codificador quanto pelo decodificador.

$$f_1 : x \to \{1, 2, \ldots 2^{nR_1}\}$$
$$f_2 : y \to \{1, 2, \ldots 2^{nR_2}\}$$

O processo de codificação se resume a mandar o índice do conjunto escolhido pelas funções acima.

$$f_1(x) = i_0$$
$$f_2(y) = j_0$$

No decodificador, dado que recebemos (i_0, j_0), então teremos que $(\hat{x}, \hat{y}) = (x, y)$ se e somente se existir um único par de seqüências (x, y) tal que

$$f_1(x) = i_0$$
$$f_2(y) = j_0$$
$$(x, y) \in \mathcal{A}_\epsilon^{(n)},$$

Caso a seqüência recebida não preencha os quesitos acima, então um erro é declarado. Desta maneira, temos a distribuição do código conforme a figura A.1. Note que os pares conjuntamente típicos são isolados dentro de cada área, dada pelo produto dos conjuntos gerados do código $\mathscr{X}$ com os conjuntos do código $\mathscr{Y}$.

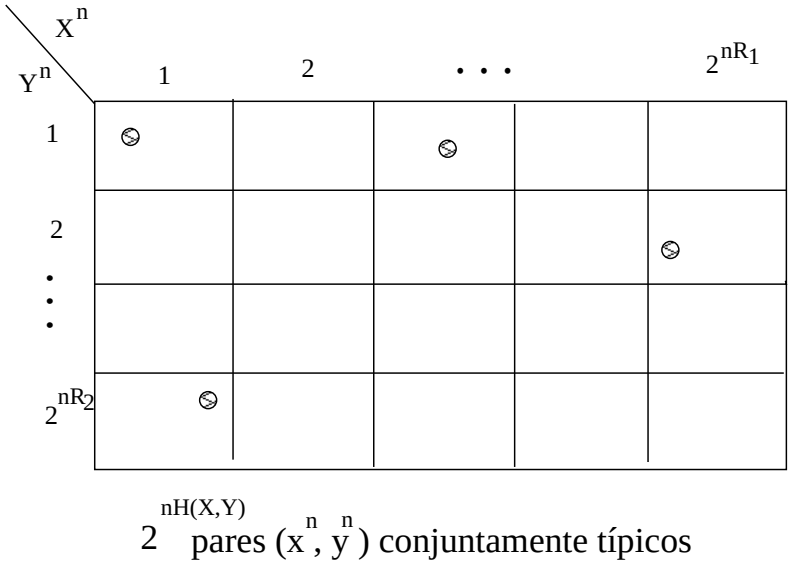


Figura A.1: Codificação Slepian-Wolf : os pares conjuntamente típicos são isolados pelo produto dos conjuntos

Para provar que o código acima é realizável, vamos calcular a probabilidade de erro do código. Se esta tender pra zero, provamos assim que tal código existe. Seja então $(x_i, y_i) \sim p(x, y)$, e definindo os seguintes eventos:

$E_0 = \{\, (X,Y) \notin A_\epsilon^{(n)}\}$
$E_1 = \{\exists\, x' \neq X : f_1(x') = f_1(X) \text{ e } (x', Y) \in A_\epsilon^{(n)}\}$
$E_2 = \{\exists\, y' \neq Y : f_2(y') = f_2(Y) \text{ e } (X, y') \in A_\epsilon^{(n)}\}$
$E_{12} = \{\exists\, (x', y') : x' \neq X, y' \neq Y, f_1(x') = f_1(X), f_2(y') = f_2(Y) \text{ e } (x', y') \in A_\epsilon^{(n)}\}$

Teremos que a probabilidade de erro definida por :

$$P_{\text{ERRO}}^{(n)} = P(E_0 \cup E_1 \cup E_2 \cup E_{12}) \leq P(E_0) + P(E_1) + P(E_2) + P(E_{12})$$

Pela propriedade de equipartição assintótica, temos $P(E_0) \to 0$, logo para n suficientemente grande $P(E_0) < \epsilon$. As outras probabilidades podem ser calculadas da

seguinte maneira:

$$\begin{aligned}
P(E_1) &= P\{\exists\, x' \neq X,\ f_1(x') = f_1(X) \text{ e } (x', Y) \in A_\epsilon^{(n)}\} \\
&= \sum_{(x,y)} p(x,y)\, P\{\exists\, x' \neq x,\ f_1(x') = f_1(x) \text{ e } (x', y) \in A_\epsilon^{(n)}\} \\
&\leq \sum_{(x,y)} p(x,y) \sum_{\substack{x' \neq x \\ (x',y) \in A_\epsilon^{(n)}}} P\{f_1(x') = f_1(x)\} \\
&= \sum_{(x,y)} p(x,y) 2^{-nR_1} \mid A_\epsilon(X \mid y) \mid \\
&= 2^{-nR_1} \mid A_\epsilon(X \mid y) \mid \sum_{(x,y)} p(x,y) \\
&\leq 2^{-nR_1} 2^{n(H(X|Y)+\epsilon)}
\end{aligned} \tag{A.2}$$

que irá tender a zero com n suficientemente grande se $R_1 > H(X|Y)$. O mesmo cálculo se estende para $P(E_2)$ com $R_2 > H(Y|X)$ e $P(E_{12})$ com $R_1 + R_2 > H(X,Y)$. E como a probabilidade de erro média é menor do que 4ϵ, teremos pelo menos um código (f_1^*, f_2^*, g^*) satisfazendo as condições acima. mais detalhes sobre a demonstração deste teorema podem ser encontrados em [7].

Prova do Teorema 2.3.1: Iremos definir as seguintes funções codificadoras

$$f_n(X^n) < R_1$$
$$g_n(Y^n) < R_2$$

e a função decodficadora h_n, que gera uma probabilidade de erro definido por:

$$P_e^{(n)} = Pr\{h_n(f_n(X^n), g_n(Y^n)) \neq X^n\} < \epsilon$$

Considerando as novas variáveis randômicas

$$S = f_n(X^n)$$
$$T = g_n(Y^n)$$

e como definimos h_n de modo a recuperar X^n de S e T com baixa probabilidade de erro, temos então da inequação de Fano que

$$H(X^n|S,T) \leq n\epsilon_n$$

As taxas para transmissão de X e Y podem ser calculadas da seguinte maneira.

$$\begin{aligned}
nR_2 &\geq H(T) \\
&\geq I(Y^n;T) \\
&= \sum_{i=1}^{n} I(Y_i;T|Y_1,\ldots Y_{i-1}), \text{ como } Y_i \text{ é independente de } Y_1 \ldots Y_{i-1} \\
&= \sum_{i=1}^{n} I(Y_i;T,Y_1,\ldots Y_{i-1}), \text{ definindo } U_i = (T,Y_1,\ldots Y_{i-1}) \\
&= \sum_{i=1}^{n} I(Y_i;U_i)
\end{aligned}$$

E para a taxa de X, teremos

$$\begin{aligned}
nR_1 &\geq H(S) \\
&\geq H(S|T) \\
&= H(S|T) + H(X^n|S,T) - H(X^n|S,T) \\
&\geq H(X^n,S|T) - n\epsilon_n \\
&= H(X^n|T) + H(S|X^n,T) - n\epsilon_n, \text{ e como S é função de } X^n \\
&= H(X^n|T) - n\epsilon_n \\
&= \sum_{i=1}^{n} H(X_i|T,X_1,\ldots X_{i-1}) - n\epsilon_n, \text{ e como o condicionamento reduz a entropia} \\
&\geq \sum_{i=1}^{n} H(X_i|T,X^{i-1},Y^{i-1}) - n\epsilon_n, \; X_i \rightarrow (T,Y^{i+1}) \rightarrow X^{i+1} \text{ é uma cadeia de Markov} \\
&= \sum_{i=1}^{n} H(X_i|T,Y^{i-1}) - n\epsilon_n \\
&= \sum_{i=1}^{n} H(X_i|U_i) - n\epsilon_n
\end{aligned}$$

Como X_i não contém informação sobre U_i que já esteja presente em Y_i, $X_i \rightarrow Y_i \rightarrow U_i$ formam uma cadeia de Markov. Então teremos as seguintes inequações:

$$R_1 \geq \frac{1}{n}\sum H(X_i|U_i) \qquad R_2 \geq \frac{1}{n}\sum I(Y_i;U_i) \tag{A.3}$$

E agora nós introduzimos uma variável randômica de *timesharing* Q, de modo que podemos reescrever as equações acima:

$$\begin{aligned}
R_1 &\geq \frac{1}{n}\sum_{i=1}^{n} H(X_i|U_i,Q=i) = H(X_Q|U_Q,Q) \\
R_2 &\geq \frac{1}{n}\sum_{i=1}^{n} I(Y_i;U_i|Q=i) = I(Y_Q;U_Q|Q)
\end{aligned} \tag{A.4}$$

Como $\mathscr{Q}$ é independente de Y_Q (a distribuição de Y_Q não depende de i), teremos

$$I(Y_Q;U_Q|Q) = I(Y_Q;U_Q,Q) - I(Y_Q,Q) = I(Y_Q,U_Q,Q) \tag{A.5}$$

X_Q e Y_Q tem probabilidade conjunta $p(x,y)$. Definindo $U = (U_Q,Q)$, $X = X_Q$ e $Y = Y_Q$, então provamos a existência da uma variável randômica U tal que:

$$R_1 \geq H(X|U) \tag{A.6}$$

$$R_2 \geq I(Y;U) \tag{A.7}$$

Antes de seguir, devemos definir as seguintes situações[1]:

- Um conjunto de seqüências x^n, y^n, z^n é dito ser ϵ-fortemente típico se

  $$|\frac{1}{n}N(a,b,c|x^n,y^n,z^n) - p(a,b,c)| < \frac{\epsilon}{|(X)||(Y)||(Z)|}$$

  se $X \to Y \to Z$ forma uma cadeia de Markov

  $$(x^n,y^n) \in A_\epsilon^{*(n)}(X,Y), (y^n,z^n) \in A_\epsilon^{*(n)}(Y,Z) \to (x^n,y^n,z^n) \in A_\epsilon^{*(n)}(X,Y,Z)$$

- $X \to Y \to Z$ formam uma cadeia de Markov ($p(x,y,z) = p(x,y)p(z|y)$). Se para um dado $(y^n,z^n) \in A_\epsilon^{*(n)}$, X^n é dado por $\sim \prod_{i+1}^n p(x_i|y_i)$, então

  $$Pr\{(x^n,y^n,z^n) \in A_\epsilon^{*(n)}(X,Y,Z)\} > 1-\epsilon \tag{A.8}$$

  para n suficientemente grande.

As equações expostas nesta seção são realizáveis da seguinte maneira:

- **Geração do *codebook*** : Gere 2^{nR_2} palavras-código independentes de tamanho n, $U(w_2)$,$w_2 \in \{1,2,\ldots 2^{nR_2}\}$ de acordo com $\prod_{i=1}^n p(u_i)$. Ligue randômicamente todas as seqüências X^n nos 2^{nR_1} conjuntos gerando independentemente um índice b uniformemente distribuído em $\{1,2,\ldots 2^{nR_1}\}$ para cada X^n. $B(i)$ denota o conjunto de seqüências X^n alocadas no conjunto i.

- **Codificação** : O codificador X manda o índice que a palavra-código X^n se encontra (i). O codificador Y procura um índice s tal que $(Y^n,U^n(s)) \in A_\epsilon^{(n)}$. Se existe mais de um s, manda o menor, se não existe $U^n(s)$ no *codebook*, manda 1.

[1] $A_\epsilon^{*(n)}(X,Y)$ é o conjunto de seqüências (x^n,y^n) fortemente típicas

- **Decodificação** : O receptor procura pelo $X^n \in B(i)$ único tal que $(X^n, U^n(s)) \in A_\epsilon^{*(n)}(X,U)$. Se não existe nenhum ou mais de um, então um erro é declarado.

Análise da probabilidade de erro

- O par $(X^n, Y^,n)$ é não-típico. A probabilidade que isto ocorra é pequena para n suficientemente grande. Sem perda de generalidade, podemos condicionar ao evento que $(x^n, y^n) \in A_\epsilon^{(n)}$.

- Y^n é típico, porém não existe $U^n(s)$ no *codebook*que seja conjuntamente típico. A probabilidade deste evento é pequena o suficiente se tivermos $R_2 > I(Y;U)$.

- A palavra-código U^n é conjuntamente típica com y^n porém não é com x^n. De acordo com a equação A.8, a probabilidade que isso ocorra é pequena, uma vez que $X \to Y \to U$ formam uma cadeia de Markov.

- Teremos um erro se existir outra seqüência $X^n \in B(i)$ que seja conjuntamente típica com $U^n(s)$. A probabilidade que isso ocorra é menor que $2^{-n(I(U;X)-3\epsilon}$, e portanto a probabilidade que este erro ocorra é de

$$|B(i) \cap A_\epsilon^{*(n)}(x)|2^{-n(I(X;U)-3\epsilon} \leq 2^{n(H(X)+\epsilon)}2^{-nR_1}2^{-n(I(X;U)-3\epsilon)}$$

que vai para zero se $R_1 > H(X|U)$

Prova do Teorema 2.4.1 (**Teorema de Wyner-Ziv**) Para provar esse teorema iremos considerar um código qualquer com informação lateral. Seja a função codificadora $f_n : \mathscr{X}^n \to \{1, 2, \ldots, 2^{nR}\}$, a função decodificadora $g_n : \mathscr{Y}^n \times \{1, 2, \ldots, 2^{nR}\} \to \mathscr{X}^n$, e o i-ésimo símbolo produzido pela função decodificadora como $g_{ni} : \mathscr{Y}^n \times \{1, 2, \ldots, 2^{nR}\} \to \mathscr{X}^n$. Considere $T = f_n(X^n)$ como sendo a versão codificada de X^n. Então para provar o teorema de Wyner-Ziv, basta mostrarmos que $Ed(X^n, g_n(Y^n, f_n(X^n))) < D$, e que $R \geq R_Y(D)$. Assim temos as seguintes

inequações:

$$
\begin{aligned}
nR &\overset{(a)}{\geq} H(T) \\
&\overset{(b)}{\geq} H(T|Y^n) \\
&\geq I(X^n, T|Y^n) \\
&\overset{(c)}{=} \sum_{i=1}^{n} I(X_i; T|Y^n, X^{i-1}) \\
&= \sum_{i=1}^{n} H(X_i|Y^n, X^{i-1}) - H(X_i|T, Y^n, X^{i-1}) \\
&\overset{(d)}{=} \sum_{i=1}^{n} H(X_i|Y_i) - H(X_i|T, Y^{i-1}, Y_i, Y_{i+1}^n, X^{i-1}) \\
&\overset{(e)}{\geq} \sum_{i=1}^{n} H(X_i|Y_i) - H(X_i|T, Y^{i-1}, Y, Y_{i+1}^n) \\
&\overset{(f)}{=} \sum_{i=1}^{n} H(X_i|Y_i) - H(X_i|W_i, Y_i) \\
&\overset{(g)}{=} \sum_{i=1}^{n} I(X_i; W_i|Y_i) \\
&= \sum_{i=1}^{n} H(W_i|Y_i) - H(W_i|X_i, Y_i) \\
&\overset{(k)}{=} \sum_{i=1}^{n} H(W_i|Y_i) - H(W_i|X_i) \\
&= \sum_{i=1}^{n} H(W_i) - H(W_i|X_i) - H(W_i) + H(W_i|Y_i) \\
&= \sum_{i=1}^{n} I(W_i; X_i) - I(W_i; Y_i) \\
&\overset{(i)}{\geq} \sum_{i=1}^{n} R_Y(E[d(X_i, g_{ni}^{,}(W_i, Y_i)])) \\
&= n\frac{1}{n}\sum_{i=1}^{n} R_Y(E[d(X_i, g_{ni}^{,}(W_i, Y_i)])) \\
&\overset{(j)}{\geq} nR_Y(E[\frac{1}{n}\sum_{i=1}^{n} d(X_i, g_{ni}^{,}(W_i, Y_i))]) \\
&\overset{(k)}{\geq} nR_Y(D)
\end{aligned}
$$

onde,

(a) vem do fato que o range de T é $\{1, 2, \ldots, 2^{nR}\}$

(b) segue do fato que condicionamento reduz a entropia

(c) vem da regra da cadeia para informação mútua

(d) vem do fato que X_i é independente dos valores passados e futuros de Y e X, dado Y_i

(e) segue do fato que condicionamento reduz a entropia

(f) vem ao definirmos $W_i = (T, Y^{i-1}, Y_{i+1}^n)$

(g) segue da definição de informação mútua

(h) segue do fato de que uma vez que Y_i depende apenas de X_i e é condicionalmente independente de T e de valores passados e futuros de Y, e portanto $W_i \to X_i \to Y_i$ formam uma cadeia de Markov

(i) vem da definição de função taxa-distorção (informação) condicional, uma vez que $\hat{X}_i = g_{ni}(T, Y^n) \triangleq g_{ni}^{,}(W_i, Y_i)$, e portanto

$$I(W_i; X_i) - I(W_i; Y_i) \geq \min_{W: E[d(X,\hat{X})] \leq D} I(W; X) - I(W; Y) = R_Y(D_i)$$

(j) segue da inequação de Jensen e da convexidade da função taxa-distorção

(k) segue da definição da distorção $D = E[\frac{1}{n}\sum_{i=1}^n d(X_i, \hat{X}_i)]$

A prova de realização do código é paralela a prova do teorema de taxa-distorção para fontes com forte tipicalidade. Ao invés de usarmos o índice da palavra-código, nós dividimos o espaço em conjuntos e mandamos o índice do conjunto. Se o número de palavras-código em cada conjunto é pequeno o suficiente, então a informação lateral poderá isolar a palavra-código específica.

- **Geração do *codebook*** : Seja $R_1 = I(X; W) + \epsilon$. Gere 2^{nR} palavras-código i.i.d. com $W^n(s) \sim \prod_{i=1}^n p(w_i)$, e índice s tal que $s \in \{1, 2, \ldots, 2^{nR_1}\}$. Seja $R_2 = I(X; W) - I(Y; W) + 5\epsilon$. Vamos associar randômicamente os índices $s \in \{1, 2, \ldots, \}$ para um dos 2^{nR_2} conjuntos usando uma distribuição uniforme, Seja $B(i)$ o índice relativo ao conjunto i. Temos aproximadamente 2^{nR_1} índices em cada conjunto.

- **Codificando o código** : Dada uma seqüência X^n, o codificador olha para a palavra-código $W^n(s)$ tal que $(X^n, W^n_{(s)}) \in A_\epsilon^{*(n)}$. Se não existe $W^n(s)$, então $s = 1$. Se existe mais de um s, o codificador escolhe o menor. O codificador manda o índice o conjunto ao qual s pertence

- **Decodificando o código** : O decodificador procura por um $W^n(s)$ tal que $s \in B(i)$ e $(W^n(s), Y^n) \in A_\epsilon^{(n)}$. Se achar um s único, ele calcula $\widehat{X}^n$, onde $\widehat{X}^n = f(W_i, Y_i)$. Se não achar um s ou achar mais de um, ele seta $\widehat{X}^n = \hat{x}^n$, onde $\hat{x}^n$ é uma seqüência arbitrária.

Analisando a probabilidade de erro, podemos ter os seguintes casos:

1. $(X^n, Y^n) \notin A_\epsilon^{*(n)}$. A probabilidade deste evento é pequena pela lei dos grandes números.

2. X^n é típico, porém $\nexists s$ tal que $(X^n, W^n(s)) \in A_\epsilon^{(n)}$. A probabilidade deste evento é pequena se $R_1 > I(W; X)$

3. O par de seqüências $(X^n, W^n(s)) \in A_\epsilon^{*(n)}$, porém $(W^n(s), Y^n) \notin A_\epsilon^{*(n)}$ (a palavra-código é conjuntamente típica com a seqüência Y^n). Pelo lema de Markov, a probabilidade deste evento é pequena para n grande.

4. $\exists s'$ com o mesmo índice do conjunto, tal que $(W^n(s'), Y^n) \in A_\epsilon^{*(n)}$. Uma vez que a probabilidade de um W^n aleatório ser conjuntamente típico com Y^n é $\approx 2^{-nI(Y;W)}$, a probabilidade que outro evento W^n esteja no mesmo conjunto e seja típico com Y^n é limitado pelo número de palavras-código vezes a probabilidade tipicamente conjunta, i.e.

$$Pr(\exists s' \in B(i)) : (W^n(s'), Y^n) \in A_\epsilon^{*(n)} \leq 2^{n(R_1 - R_2)} 2^{-n(I(W;Y) - 3\epsilon)} \qquad \text{(A.9)}$$

   que vai para zero se $R_1 - R_2 < I(Y; W) - 3\epsilon$

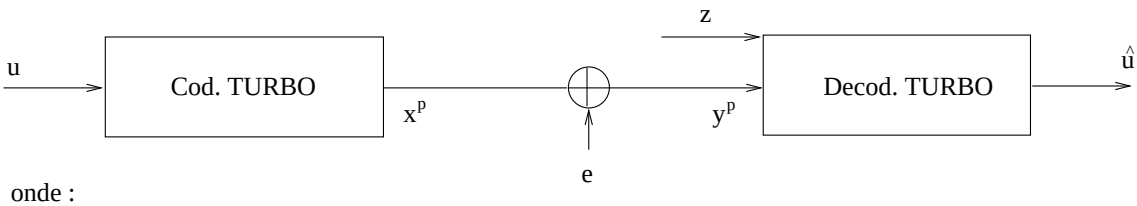
5. Se o índice s é decodificado corretamente, então $(X^n, W^n(s)) \in A_\epsilon^{*(n)}$. Pelo item 1, podemos assumir que $(X^n, Y^n) \in A_\epsilon^{*(n)}$. Então, pelo lema de Markov, nós temos que $(X^n, Y^n, W^n) \in A_\epsilon^{*(n)}$ e portanto a distribuição empírica conjunta se aproxima da distribuição original $p(x, y)p(w|x)$ com a qual começamos, e portanto $(X, \widehat{X}^n)$ terá a distribuição conjunta próxima a distribuição que atinge a distorção D.

Logo, com alta probabilidade, o decodificador vai reproduzir $\widehat{X}^n$ tal que a distorção entre X^n e $\widehat{X}^n$ seja próxima a nD.

Apêndice B

Aplicação da teoria de Wyner-Ziv para códigos turbo

Considerando a figura B.1



onde :

u é o símbolo a ser tansmitido

x^p são os bits de paridade gerados pelo código turbo

e é o erro do canal

y^p são os bits de paridade recebidos após transmissão

z é a informação lateral

$\hat{u}$ é o símbolo decodificado

Figura B.1: Código turbo com informação lateral

O decodificador BCJR-MAP decide que o símbolo $u_{k-1} = i$[1] foi transmitido se $Pr\{u_{k-1} = i|z, y^p\} > Pr\{u_{k-1} = j|z, y^p\}$, onde $j \neq i$. temos então que encontrar $i \in U^*$, $U^* = U - \{0\}$, tal que :

$$\frac{Pr\{U_{k-1} = i|z, y^p\}}{Pr\{U_{k-1} = 0|z, y^p\}} = \max\left\{\frac{Pr\{U_{k-1} = j|z, y^p\}}{Pr\{U_{k-1} = 0|z, y^p\}} : j \in U^*\right\}$$

Após esse cálculo, decidimos que o símbolo i foi mandado se $Pr\{U_{k-1} = i|z, y^p\} > Pr\{U_{k-1} = 0|z, y^p\}$, caso contrário $u_{k-1} = 0$ foi transmitido.

Agora definimos a verossimilhança logarítmica (*log-likehood*) da seguinte maneira.

[1] o índice k denota o tempo

$$L_{k-1}(i) = \ln\left[\frac{Pr\{U_{k-1}=i|z,y^p\}}{Pr\{U_{k-1}=0|z,y^p\}}\right] \Longrightarrow u_{k-1} = \begin{cases} i & ,L_{k-1}(i) > 0 \\ 0 & ,L_{k-1}(i) < 0 \end{cases}$$

onde $i \in U^*$ é o símbolo para o qual $L_{k-1}(i) = \max\{L_{k-1}(j) : j \in U^*\}$.

Definindo

$$L^a_{k-1}(i) = \ln\left[\frac{Pr\{U_{k-1}=i\}}{Pr\{U_{k-1}=0\}}\right]$$

e visto que

$$Pr\{U_{k-1}=0\} = 1 - \sum_{i=1}^{\frac{M}{2}-1} Pr\{U_{k-1}=i\}$$

então

$$Pr\{U_{k-1}=i\} = \frac{e^{L^a_{k-1}(i)}}{1 + \sum_{j=1}^{\frac{M}{2}-1} e^{L^a_{k-1}(j)}}$$

e

$$Pr\{U_{k-1}=0\} = \frac{1}{1 + \sum_{j=1}^{\frac{M}{2}-1} e^{L^a_{k-1}(j)}}$$

Interpretando : A probabilidade da entrada ser igual a $\underline{i}$ é a mesma probabilidade de estar num estado s' e passar para um estado s por causa de uma entrada $\underline{i}$. Logo,

$$L_{k-1}(i) = \ln\left[\frac{\sum_{(s',s)\in B^i_{k-1}} Pr\{S_{k-1}=s', S_k=s|z,y^p\}}{\sum_{(s',s)\in B^0_{k-1}} Pr\{S_{k-1}=s', S_k=s|z,y^p\}}\right]$$

Pela lei de Bayes

$$Pr\{S_{k-1}=s', S_k=s|z,y^p\} = \frac{Pr\{S_{k-1}=s', S_k=s, z, y^p\}}{Pr\{z,y^p\}}$$

e decompondo $\{z,y^p\}$ nos intervalos de 0 a $N \Longrightarrow \{z_0y^p_0, \ldots z_{N-1}y^p_{N-1}\}$

$$Pr\{S_{k-1}=s', S_k=s|z,y^p\} = Pr\{S_{k-1}=s', S_k=s, z_{0,k-2}, y^p_{0,k-2}, z_{0,k-1}, y^p_{0,k-1}, \ldots z_{0,N-1}, y^p_{0,N-1}\}$$

aplicando a lei de Bayes, teremos então

$$\begin{aligned} &= Pr\{S_{k-1}=s'z_{0,k-2}, y^p_{0,k-2}\} \cdot Pr\{S_k=s'z_{k-1}, y^p_{k-1}, z_{k,N-1}, y^p_{k,N-1}|S_{k-1}=s'z_{0,k-2}, y^p_{0,k-2}\} \\ &= Pr\{S_{k-1}=s'z_{0,k-2}, y^p_{0,k-2}\} \cdot Pr\{S_k=s, z_{k-1}, y^p_{k-1}|S_{k-1}=s', z_{0,k-2}, y^p_{0,k-2}\} \cdot \\ &\quad Pr\{z_{k,N-1}, y^p_{k,N-1}|S_{k-1}=s', S_k=s, z_{0,k-2}, y^p_{0,k-2}, z_{k-1}, y^p_{k-1}\} \end{aligned}$$

como códigos convolucionais geram processos de Markov, isto é, o conhecimento do estado s_{k-1} remove a influência da seqüência $y^p_{0,k-2}, z_{0,k-2}$, disponível antes do "tempo" $k-1$, sobre o estado s e sobre z_{k-1}, y^p_{k-1}. Além disso, o conhecimento de s_k remove a influência de $s_{k-1}, z_{k-1}, y_{k-1}, z_{0,k-2}, y^p_{0,k-2}$ sobre $z_{k,N-1}, y^p_{k,N-1}$.

$$
\begin{aligned}
&= Pr\{S_{k-1} = s' z_{0,k-2}, y^p_{0,k-2}\} \cdot Pr\{S_k = s' z_{k-1}, y^p_{k-1}, z_{k,N-1}, y^p_{k,N-1} | S_{k-1} = s' z_{0,k-2}, y^p_{0,k-2}\} \\
&= Pr\{S_{k-1} = s' z_{0,k-2}, y^p_{0,k-2}\} \cdot Pr\{S_k = s, z_{k-1}, y^p_{k-1} | S_{k-1} = s', z_{0,k-2}, y^p_{0,k-2}\} \\
&\quad \cdot Pr\{z_{k,N-1}, y^p_{k,N-1} | S_{k-1} = s', S_k = s, z_{0,k-2}, y^p_{0,k-2}, z_{k-1}, y^p_{k-1}\}
\end{aligned}
$$

Dividindo o resultado acima por

$$
Pr\{z, y^p\} = Pr\{z_{0,k-2}, y^0_{0,k-2}\}. Pr\{z_{k-1}, y^p_{k-1} | z_{0,k-2}, y^p_{0,k-2}\}. Pr\{z_{k,N-1}, y^p_{k,N-1} | z_{0,k-1}, y^p_{0,k-1}\}
$$

teremos

$$
\begin{aligned}
Pr\{S_{k-1} = s', S_k = s | z, y^p\} \;=\; & \frac{Pr\{S_{k-1} = s', z_{0,k-2}, y^p_{0,k-2}\}}{Pr\{z_{0,k-2}, y^p_{0,k-2}\}} \cdot \\
& \frac{Pr\{S_k = s, z_{k-1}, y^p_{k-1} | S_{k-1} = s' z_{0,k-2}, y^p_{0,k-2}\}}{Pr\{z_{k-1}, y^p_{k-1} | z_{0,k-2}, y^p_{0,k-2}\}} \cdot \\
& \frac{Pr\{z_{k,N-1}, y^p_{k,N-1} | S_k = s\}}{Pr\{z_{k,N-1}, y^p_{k,N-1} | z_{0,k-1}, y^p_{0,k-1}\}}
\end{aligned}
$$

definindo:

$$
\alpha_k(s) = \frac{Pr\{S_k = s, z_{0,k-1}, y^p_{0,k-1}\}}{Pr\{z_{0,k-1}, y^p_{0,k-1}\}}
$$

$$
\beta_k(s) = \frac{Pr\{y^p_{k,N-1}, z_{k,N-1} | S_k = s\}}{Pr\{z_{k,N-1}, y^p_{k,N-1} | z_{0,k-1}, y^p_{0,k-1}\}}
$$

$$
\gamma_k(s', s) = Pr\{S_k = s, z_{k-1}, y^p_{k-1} | S_{k-1} = s'\}
$$

e substituindo na expressão acima, teremos

$$
Pr\{S_{k-1} = s', S_k = s | z, y^p\} = \frac{1}{Pr\{z_{k-1}, y^p_{k-1} | z_{0,k-2}, y^p_{0,k-2}\}} . \alpha_{k-1}(s') . \gamma_k(s', s) . \beta_k(s)
$$

e substituindo na equações do *log-likehood*

$$
L_{k-1}(i) = \ln\left[\frac{\sum_{(s',s) \in B_{k-1}} \alpha_{k-1}(s') \gamma_k(s', s) \beta_k(s)}{\sum_{(s',s) \in B^0_{k-1}} \alpha_{k-1}(s') \gamma(s', s) \beta_k(s)}\right]
$$

os coeficientes α_k e β_k podem ser calculados recursivamente

$$
\alpha_k(s) = \frac{\sum_{s'} \alpha_{k-1}(s') \gamma_k(s', s)}{\sum_{s} \sum_{s'} \alpha_{k-1}(s') \gamma(s', s)} \quad , \quad \alpha_0(s) = Pr\{S_0 = s\} \begin{cases} 1, & \text{p/ } s = 0; \\ 0, & \text{p/ } s \neq 0. \end{cases}
$$

$$\beta_{k-1}(s') = \frac{\sum_{s} \gamma_k(s', s)\beta_k(s)}{\sum_{s}\sum_{s'} \alpha_{k-1}(s')\gamma(s', s)} \quad , \quad \beta_N(s) = 1/N_e \text{ ou } \begin{cases} 1, & \text{p/ } s = 0; \\ 0, & \text{p/ } s \neq 0. \end{cases}$$

para o cálculo do $\gamma_k(s', s)$ iremos assumir que os bits de paridade foram transmitidos por um canal com ruído Gaussiano e foram transmitidos usando uma modulação BPSK $\left\{0 \rightarrow -1 \quad 1 \rightarrow +1\right\}$. No caso da informação lateral, assumimos um conhecimento da correlação entre esta e a entrada, dada pela função $f(z|x)$.

$\rightarrow$ cálculo para uma entrada $u_{k-1} = i$

$$\begin{aligned}
\gamma_k(s', s) &= Pr\{S_k = s, z_{k-1}, y^p_{k-1} | S_{k-1} = s'\} \\
&= Pr\{S_k = s | S_{k-1} = s'\}. Pr\{z_{k-1}, y^p_{k-1} | S_{k-1} = s', S_k = s\} \\
&= Pr\{U_{k-1} = i\}. Pr\{z_{k-1}, y^p_{k-1} | X_{k-1} = x_{k-1}\} \\
&= Pr\{U_{k-1} = i\}. Pr\{z_{k-1} | X^s_{k-1} = i\}. Pr\{y^p_{k-1} | X^p_{k-1} = x^p_{k-1}(s', s)\} \\
&= \frac{e^{L^a_{k-1}(i)}}{1 + \sum_{j=1}^{\frac{M}{2}-1} e^{L^a_{k-1}(j)}}. Pr\{z_{k-1} | X^s_{k-1} = i\}. e^{-\frac{(y^p_{k-1} - x^p_{k-1}(s', s))^2}{2\sigma^2}} \\
&= \frac{e^{-(\frac{y^p_{k-1}}{2\sigma})^2 - (\frac{x^p_{k-1}(s', s)}{2\sigma^2})^2}}{1 + \sum_{j=1}^{\frac{M}{2}-1} e^{L^a_{k-1}(j)}}. e^{L^a_{k-1}(i)}. e^{\frac{y^p_{k-1} x^p_{k-1}(s', s)}{\sigma^2}}. Pr\{z_{k-1} | X^s_{k-1} = i\} \\
&= c_{k-1}. e^{L^a_{k-1}(i)}. e^{\frac{y^p_{k-1} x^p_{k-1}(s', s)}{\sigma^2}}. Pr\{z_{k-1} | X^s_{k-1} = i\}
\end{aligned}$$

o cálculo análogo pode ser feito para B^0_{k-1} (transição ocorre devido a uma entrada 0)

$$\gamma_k(s', s) = c_{k-1}. e^{\frac{yh^p_{k-1} x^p_{k-1}(s', s)}{\sigma^2}}. Pr\{z_{k-1} | X^s_{k-1} = i\}$$

Na implementação da tese, usamos um canal perfeito no lugar do canal Gaussiano, e com isso substituímos a exponencial por uma função impulso, e com isso usamos na nossa implementação o seguinte γ

$$\gamma_k(s', s) = Pr\{X^s_{k-1} = i\}. \delta(y^p_{k-1} - x^p_{k-1}(s', s)). Pr\{X^s_{k-1} = i | z_{k-1}\}$$

Substituindo os valores acima na equação de $L_{k-1}(i)$

$$
\begin{aligned}
L_{k-1}(i) &= \ln\left[\frac{\sum_{(s',s)\in B_{k-1}} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum_{(s',s)\in B^0_{k-1}} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right] \\
&= \ln\left[\frac{\sum_{(s',s)\in B_{k-1}} \alpha_{k-1}(s')(c_{k-1}e^{L^a_{k-1}(i)}e^{\frac{y^p_{k-1}x^p_{k-1}(s',s)}{\sigma^2}}Pr\{z_{k-1}|X^s_{k-1}=i\})\beta_k(s)}{\sum_{(s',s)\in B^0_{k-1}} \alpha_{k-1}(s')(c_{k-1}e^{L^a_{k-1}(i)}e^{\frac{y^p_{k-1}x^p_{k-1}(s',s)}{\sigma^2}}Pr\{z_{k-1}|X^s_{k-1}=i\})\beta_k(s)}\right] \\
&= \underbrace{L^a_{k-1}(i)}_{\text{inf. à priori}} + \underbrace{\ln\left[\frac{Pr\{z_{k-1}|X^s_{k-1}=i\}}{Pr\{z_{k-1}|X^s_{k-1}=0\}}\right]}_{\text{inf. dependente da inf. lateral}} + \underbrace{\ln\left[\frac{\sum_{(s',s)\in B_{k-1}} \alpha_{k-1}(s')\gamma^e(s',s)\beta_k(s)}{\sum_{(s',s)\in B^0_{k-1}} \alpha_{k-1}(s',)\gamma^e(s',s)\beta_k(s)}\right]}_{\text{informação extrínsica}}
\end{aligned}
$$

Note que, ao compararmos as equações obtidas com as equações do código turbo, podemos interpretar o resultado da seguinte maneira. Os bits sistemáticos x^s são transmitidos através de um canal fictício, onde recebemos z do outro lado, ao invés de y^s. Consideramos então o ruído do canal fictício dado pela relação entre x^s e z, que é a correlação entre as duas variáveis. Logo, se a informação lateral é muito boa, i.e. tem uma alta correlação com a entrada, então seria como se estivéssemos transmitindo através de um canal muito bom, quase sem erros, e portanto não seria necessário muitos bits de paridade para decodificarmos corretamente a palavra recebida. Já se a informação lateral não está correlacionada com a entrada, então o ruído do canal fictício é alto, e portanto devemos mandar muitos bits de paridade para podermos corrigir os "erros de transmissão".

Apêndice C

Método de interpolação de quadros para obtenção da informação lateral

Um ponto chave no processo de codificação distribuída , como também mencionado em [26], é a obtenção da informação lateral. Como vimos no apêndice B, a qualidade da informação lateral irá afetar diretamente a taxa resultante da codificação, onde quadros estimados que se aproximam mais da fonte a ser transmitida irão requerer menos bits, logo atingindo taxas de compressão maiores. Neste apêndice iremos mostrar diversas técnicas utilizadas na estimação da informação lateral, e justificar a escolha do método de estimação de movimento utilizado durante as simulações nos capítulos 4, 5 e 6

A dificuldade de modelarmos o movimento depende de diversos fatores, desde a imagem em questão até a velocidade do movimento e o número de quadros envolvido no processo de reconstrução. Algumas técnicas simples podem ser utilizadas, como por exemplo usar o quadro anterior (ou posterior) ou então fazer a média dos dois quadros, porém técnicas como esta não conseguem fornecer informação lateral confiável, especialmente em seqüências com muito movimento. Neste caso utilizamos algumas técnicas para a estimação de movimento.

Estimamos o movimento do quadro anterior para o quadro posterior (movimento *forward*). Depois estimamos o movimento do quadro posterior para o quadro anterior (movimento *backward*). A informação lateral é calculada como a média dos dois meio-movimentos, isto é, o vetor de movimento usado será metade do vetor de movimento achado no processo de busca, uma vez que queremos estimar o movi-

mento de um quadro até o quadro a ser codificado, e a busca feita utilizou quadros separados por dois quadros, como mostra a figura C.1. Note que estamos considerando que o movimento é um movimento uniforme, por termos quadros muito próximos, por isso o vetor de movimento é simplesmente dividido por dois.

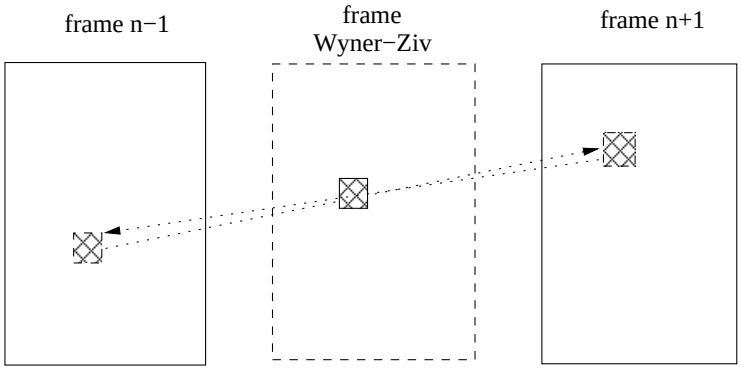


Figura C.1: Estimação de movimento bidirecional.

Para suavizarmos a estimação de movimento, limitamos a área de busca usando o algoritmo descrito em [34] e reproduzido aqui. Seja E o bloco que desejamos calcular os vetores de movimento, e considere seus macroblocos vizinhos dados por A, B e C, como mostra a figura C.2 Considere (MV_{A_x}, MV_{A_y}), (MV_{B_x}, MV_{B_y})

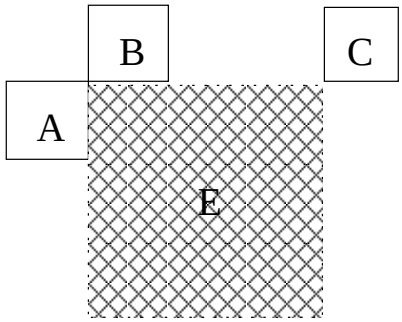


Figura C.2: Localização de blocos para a estimação de movimento

e (MV_{C_x}, MV_{C_y}) os vetores de movimento dos respectivos blocos A, B e C, e considere range_da_busca como a área de busca inicialmente estipulada. primeiramente limitamos o valor do vetor de movimento do bloco E de acordo com os vetores de movimento dos blocos vizinhos.

$$\max(MV_{E_x}) = \max(abs(MV_{A_x}), \max(abs(MV_{B_x}), abs(MV_{C_x})))$$

$$\max(MV_{E_y}) = \max(abs(MV_{A_y}), \max(abs(MV_{B_y}), abs(MV_{C_y})))$$

onde se um macrobloco estiver fora da imagem, então range_da_busca será utilizado. A nova área de busca será então calculada da seguinte maneira.

$$\text{area\_busca\_local\_x} = \max(k_x, 2 \times \max(MV_{E_x}))$$
$$\text{area\_busca\_local\_y} = \max(k_y, 2 \times \max(MV_{E_y}))$$

onde k_x e k_y são determinados da seguinte maneira

$$k_i = \begin{cases} \frac{(\text{range\_de\_busca}+2)}{4} & \text{, se } \alpha_i > 2 \\ \frac{(\text{range\_de\_busca}+4)}{8} & \text{, do contrário} \end{cases}$$
$$\alpha_i = abs(MV_{A_i}) + abs(MV_{B_i}) + abs(MV_{C_i}), \text{ para i=x,y}$$

A área máxima de busca, e por conseqüência o valor máximo que o vetor de estimação de movimento poderá assumir será de

$$\text{nova\_area\_de\_busca\_x} = \min(\text{range\_de\_busca}, \text{area\_busca\_local\_x})$$
$$\text{nova\_area\_de\_busca\_y} = \min(\text{range\_de\_busca}, \text{area\_busca\_local\_y})$$

Além de limitarmos a área de busca, utilizamos uma outra técnica apresentada em [32] para evitar os artefatos de blocos, onde os blocos de busca são sobrepostos uns aos outros (*overlapped block matching*). A técnica é ilustrada na figura C.3. A imagem é particionada em blocos $2N \times 2N$ com sobreposição, centrados em blocos $N \times N$. Assumindo uma área de busca de $\pm s$ pixels, a melhor estimativa (o melhor bloco $N \times N$) será a que minimizar a distorção média entre blocos sobrepostos de tamanho $2N \times 2N$, após ajustar o erro com uma função de janelamento dada pela função w(x,y), que nas nossas simulações usamos a função

$$w(x, y) = \cos^2(\frac{x\pi}{2N}) \cos^2(\frac{y\pi}{2N})$$

onde $x, y = -N \ldots N$.

A imagem reconstruída é formada deslocando e janelando cada bloco $2N \times 2N$ do quadro anterior. Os pixels que ficam fora da imagem são considerados como zero.

Uma outra maneira de suavizar o movimento foi criando um campo denso de vetores, e para cada bloco fazer a média destes vetores, como se estivéssemos "penteando" os vetores de movimento. Outra variável analisada foi o tamanho do

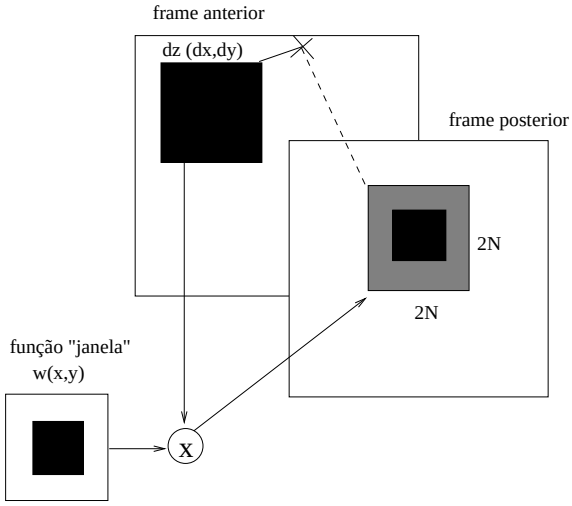


Figura C.3: Estimação de movimento de blocos com sobreposição

bloco, onde fizemos estimativas com blocos pequenos (4×4) e também com blocos maiores (8×8). Na tabela C.1 vemos os resultados obtidos com diferentes parâmetros escolhidos e também para diferentes seqüências.

	Foreman	*Mother&Daughter*	*Suzie*
teste 1	33,03493	44,10790	37,95666
teste 2	28,18189	39,48757	33,05935
teste 3	28,14011	39,49631	33,28731
teste 4	35,74654	44,48970	39,79867
teste 5	35,46806	43,94043	39,14701
teste 6	35,28765	44,22207	39,56372
teste 7	35,76807	44,49796	39,85555
teste 8	35,76807	43,67757	38,96321

Tabela C.1: Resultados obtidos com diferentes técnicas para obtenção da informação lateral

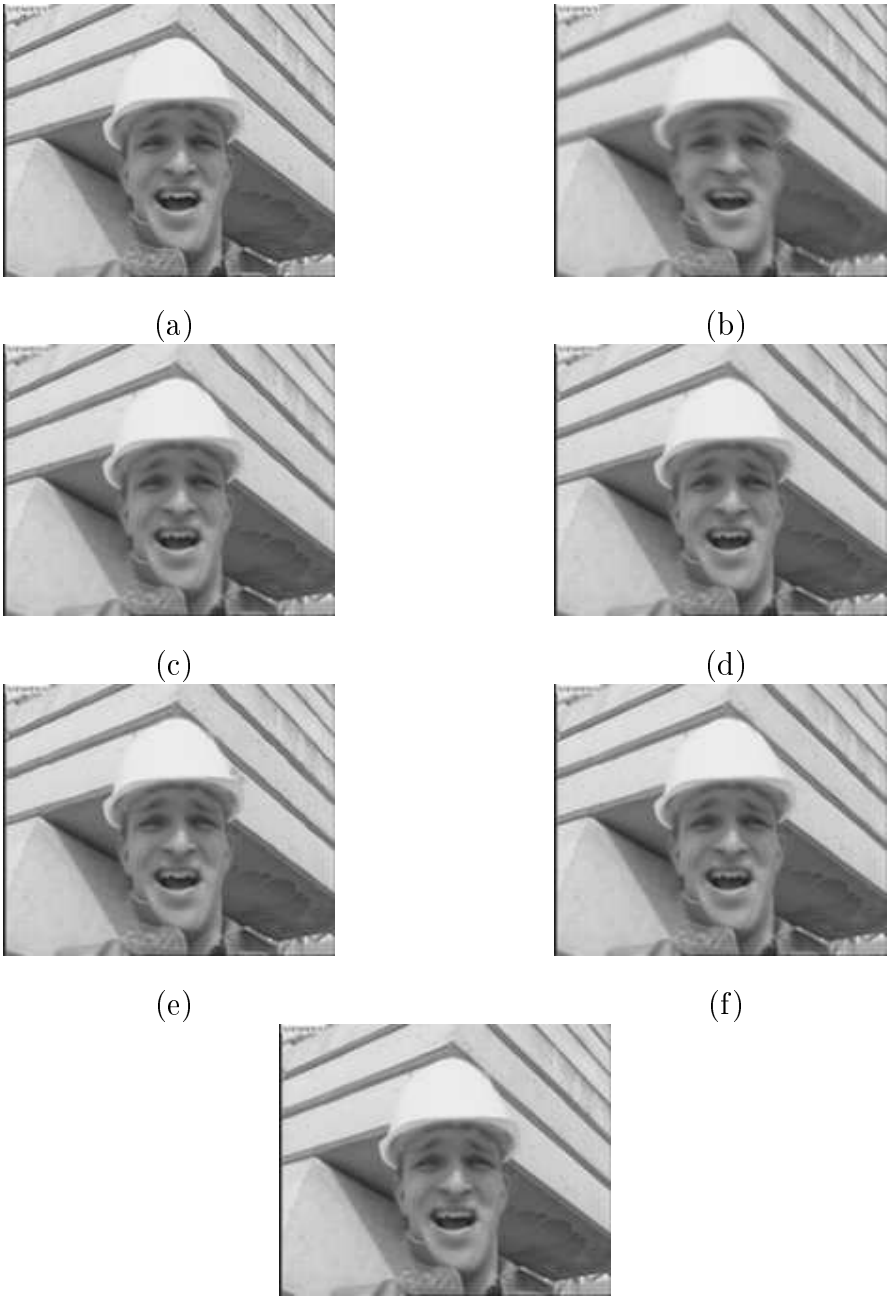
Três seqüências foram analisadas, uma com muito movimento (*Foreman*) e outras duas com poucos movimentos (*Mother&Daughter* e *Suzie*). Na tabela C.1 vemos o PSNR médio da informação lateral usando 8 maneiras diferentes.

1. Média dos quadros anterior e posterior ao quadro Wyner-Ziv

2. Quadro posterior ao quadro Wyner-Ziv

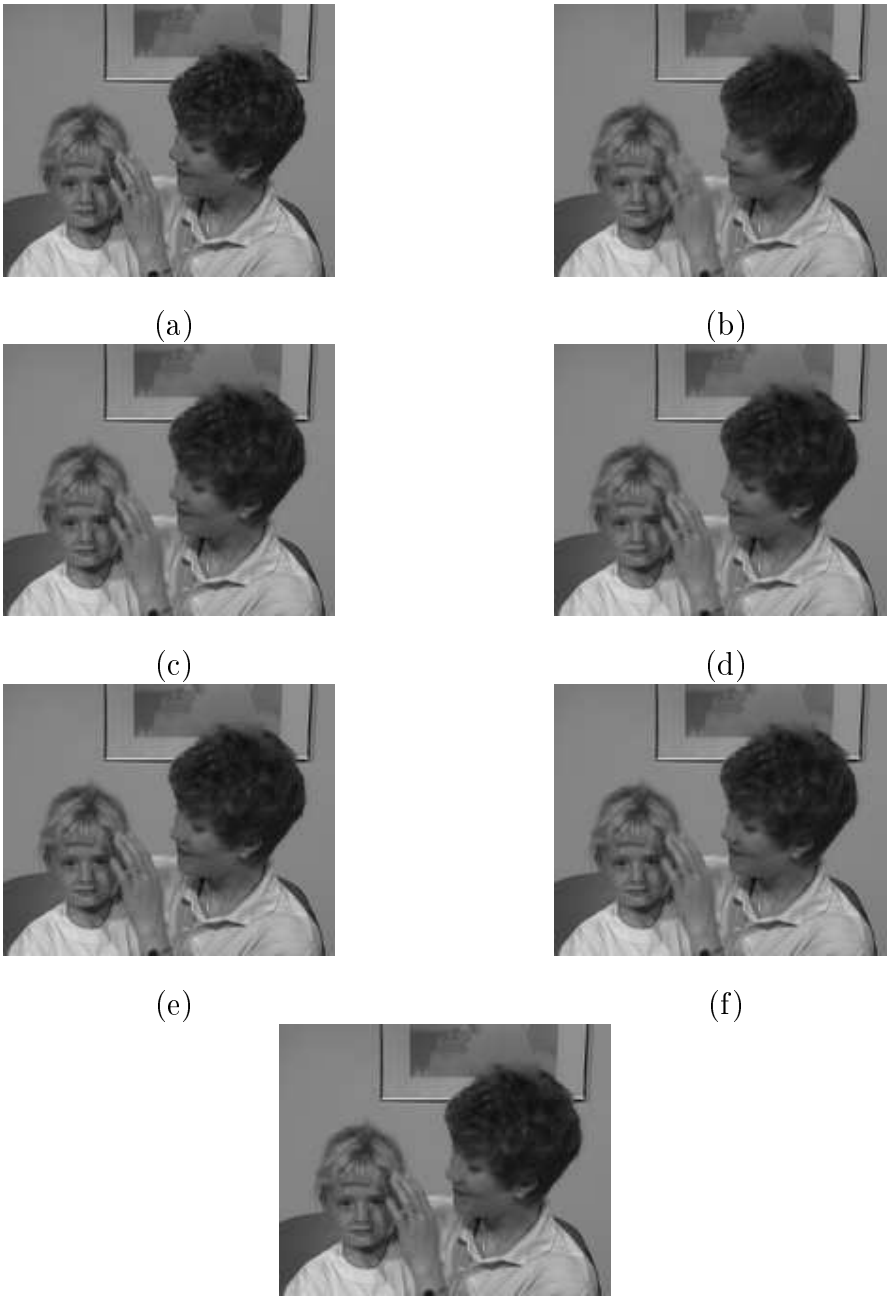
3. Quadro anterior ao quadro Wyner-Ziv
4. Estimação de movimento padrão (bloco 8×8, área de busca de 12 pixels)
5. Estimação de movimento igual ao item anterior, só que com interpolação de coordenadas fracionárias
6. Estimação de movimento com bloco de tamanho 4×4
7. Estimação de movimento com campo denso de vetores (um vetor de movimento para cada pixel)
8. Estimação de movimento com blocos pequenos e campo denso de vetores (bloco 4×4 e um vetor de movimento para cada pixel)

Note que algumas técnicas funcionaram melhor em imagens com pouco movimento, como é o caso do campo denso de vetores, que gerou resultados melhores no caso da seqüência *Mother&Daughter*. Já no caso das outras seqüências, utilizar uma estimação de movimento padrão, com blocos de tamanho 8×8 e área de busca limitada em no máximo 12 pixels é que gerou o melhor resultado. Durante a tese, quando foi mencionado estimação de movimento, utilizamos os parâmetros do teste 4.



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

Figura C.4: Exemplos de diferentes informações laterais da seqüência *Foreman* (a) Quadro de referência (Quadro 7) (b) teste 1 (c) teste 4 (d) teste 5 (e) teste 6 (f) teste 7 (g) teste 8



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

Figura C.5: Exemplos de diferentes informações laterais da seqüência *Mother&Daughter* (a) Quadro de referência (Quadro 7) (b) teste 1 (c) teste 4 (d) teste 5 (e) teste 6 (f) teste 7 (g) teste 8

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

Figura C.6: Exemplos de diferentes informações laterais da seqüência *Suzie* (a) Quadro de referência (Quadro 7) (b) teste 1 (c) teste 4 (d) teste 5 (e) teste 6 (f) teste 7 (g) teste 8