COMPRESSÃO DE IMAGENS USANDO PREDIÇÃO E APROXIMAÇÕES POLINOMIAIS

Renam Castro da Silva

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Eduardo Antônio Barros da Silva

Rio de Janeiro
Setembro de 2013

COMPRESSÃO DE IMAGENS USANDO PREDIÇÃO E APROXIMAÇÕES POLINOMIAIS

Renam Castro da Silva

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

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

Prof. Gelson Vieira Mendonça, Ph.D.

Prof. Lisandro Lovisolo, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2013

Silva, Renam Castro da

Compressão de Imagens usando Predição e Aproximações Polinomiais/Renam Castro da Silva. – Rio de Janeiro: UFRJ/COPPE, 2013.

XVI, 127 p.: il.; 29, 7cm.

Orientador: Eduardo Antônio Barros da Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2013.

Referências Bibliográficas: p. 85 – 87.

1. Compressão de Imagens. 2. Predição. 3. Funções Polinomiais. I. Silva, Eduardo Antônio Barros da. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Aos meus leitores

Agradecimentos

A Deus por permitir a realização deste trabalho. À minha família. Agradeço à Rossana e à Dona Marina por me receberem em sua família aqui no Rio de Janeiro. Agradeço ao professor Eduardo pela solícita orientação e aos professores e colegas do laboratório. À CAPES pelo apoio financeiro. À COPPE/UFRJ e ao LPS.

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

COMPRESSÃO DE IMAGENS USANDO PREDIÇÃO E APROXIMAÇÕES POLINOMIAIS

Renam Castro da Silva

Setembro/2013

Orientador: Eduardo Antônio Barros da Silva

Programa: Engenharia Elétrica

O *Linear Fitting and Flexible Prediction* (LFP) é um algoritmo de compressão concebido para a codificação de imagens de mapa de profundidades. Ele emprega técnicas de predição, aproximação do resíduo com funções lineares e dicionários de blocos reutilizáveis. O algoritmo exibe desempenho comparável, ou mesmo superior em relação aos algoritmos propostos na literatura. Neste trabalho, propomos esquemas para melhorar o desempenho do algoritmo na codificação de mapas de profundidade, e ainda modificamos e estendemos o LFP para a codificação de imagens de textura, cuja aplicação não tivera sido investigada.

Em técnicas que se baseiam nas informações de textura e de profundidade para gerar vistas virtuais, os mapas de profundidade precisam ser representados com um número reduzido de *bits* sem comprometer a qualidade visual das vistas sintetizadas. Investigamos aproximar o resíduo de predição utilizando, além de funções lineares, funções constantes e quadráticas, e ainda utilizar quantizadores treinados para cada taxa pretendida. O algoritmo, empregando os esquemas propostos, apresenta melhor desempenho em termos de taxa de compressão dos mapas e de qualidade visual das vistas virtuais.

Para a codificação de imagens de textura, adaptamos o algoritmo para considerar a qualidade visual da imagem reconstruída. A aproximação do resíduo de predição com funções polinomiais, assim como o uso quantizadores treinados são considerados na codificação de imagens de textura. A atualização multiescala dos dicionários é experimentada, bem como formas eficientes de codificação dos índices dos vetores e dos *flags* da árvore de segmentação. As contribuições para o desempenho em termos de taxa-distorção dos esquemas investigados são exibidas nos resultados experimentais.

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

IMAGE COMPRESSION USING PREDICTION AND POLYNOMIAL APPROXIMATIONS

Renam Castro da Silva

September/2013

Advisor: Eduardo Antônio Barros da Silva

Department: Electrical Engineering

The Linear Fitting and Flexible Prediction (LFP) is a compression algorithm designed for coding depth map images. It employs prediction techniques, approximation of the residue using linear functions and dictionaries of reusable blocks. The algorithm exhibits performance comparable or even superior compared to the algorithms proposed in the literature. In this work, we propose schemes for improving the performance of the algorithm for encoding of maps. In addition, we modify and extend the LFP for encode texture images, whose application was not investigated.

In techniques that are based on the information of texture and depth to generate virtual views, the depth maps need to be represented with a reduced number of bits without compromising the visual quality of the synthesized views. We investigate the approximation of the prediction residue using, in addition to linear functions, quadratic and constant functions as well as trained quantizers for each desired rate. The algorithm employing the proposed schemes shows better performance in terms of compression ratio of maps and visual quality of virtual views.

For coding of texture images, we adapt the algorithm to consider the visual quality of the reconstructed image. The approximation of the prediction residue using polynomial functions, as well as the use of trained quantizers are considered in coding of texture images. The multi-scale update of the dictionaries is experimented as well as efficient ways of encoding the indexes of vectors and segmentation tree flags. Experimental results show the contributions to performance, in rate-distortion sense, of the investigated schemes.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Conteúdos digitais como imagens e vídeos estão presentes em diversas atividades humanas. Na medicina, por exemplo, imagens obtidas em exames de tomografia são usadas com o propósito de ajudar no diagnóstico de doenças. Imagens de estrelas e de planetas ao redor do universo são capturadas em satélites artificiais e utilizadas em pesquisas científicas. Conteúdos digitais estão presentes também em salas de cinema, eletrônicos de consumo, redes sociais de compartilhamento de imagens e vídeos.

A consequência da proliferação desses conteúdos é a grande quantidade de dados gerados. Esses dados precisam ser manipulados adequadamente antes de serem armazenados em disco e/ou transmitidos entre os pontos de uma rede de comunicação. A compressão de dados é a ferramenta técnica empregada para reduzir a quantidade de *bits* necessária para representar imagens e vídeos. Existem técnicas apropriadas para cada aplicação, a compressão sem perda é utilizada quando é necessário que a imagem original seja perfeitamente recuperada. No entanto, os métodos de compressão com perdas são os mais utilizados na codificação de imagens e vídeos, pois atingem maior taxa de compressão sem comprometer a qualidade visual percebida. Neste trabalho tratamos de um método de compressão com perdas para a codificação de imagens de mapas de profundidade e de imagens de textura.

1.1 Mapas de profundidade

Técnicas para a exibição de imagem e vídeo 3D têm sido uma ativa área de pesquisa devido ao potencial de aplicação em produtos e serviços. Os conteúdos exibidos em 3D permitem uma percepção mais realista e imersiva na cena. Para a exibição em 3D, é necessário que a cena seja capturada de pontos distintos, gerando uma grande quantidade de dados, o que potencialmente demanda bastante recursos de armazenamento. Recentemente, o interesse em sistemas com múltiplas vistas tem crescido, pois esses sistemas permitem uma experiência mais rica e natural. Sobre-

tudo, considerando o uso de *displays* autoestereoscópicos, em que a percepção de profundidade pode ser experimentada sem o uso de óculos. As técnicas que usam informações de textura e de profundidade para gerar vistas virtuais — chamadas na literatura de *Depth Image Based Rendering* (DIBR) [1][2] — têm sido consideradas para aplicações em 3D, pois permitem a cobertura da cena com um arranjo esparso de câmeras. Esses métodos sintetizam vistas virtuais a partir de imagens de textura e de profundidade, dessa forma requerem menos recursos de armazenamento e transmissão.

Independentemente do método utilizado para gerar vistas virtuais, é necessário codificar de forma eficiente tanto a informação de profundidade como as imagens de textura da cena. Para a codificação de imagens de textura existem padrões bem estabelecidos — como JPEG [3], H264/AVC[4][5] e mais recentemente o HEVC [6] — que exibem resultados estado da arte. A aplicação direta desses algoritmos para compressão de mapas não é adequada, devido à priorização da qualidade visual adotada nesses padrões, enquanto que em mapas é mais importante preservar a informação de profundidade da cena. Mapas de profundidade frequentemente aprensentam grandes regiões suaves com abruptas transições nas fronteiras dessas regiões. Algoritmos tradicionais para compressão de textura, quando aplicados na codificação de mapas, exibem resultados com artefatos nas transições das regiões suaves penalizando severamente as vistas virtuais sintetizadas com algoritmos DIBR.

O *Linear fitting and Flexible Prediction* (LFP) [7] que foi proposto e empregado para a codificação de mapas de profundidade, é um método que utiliza segmentação flexível da imagem de mapa de profundidades, predição hierárquica e funções lineares para aproximar o resíduo resultante da etapa de predição. As vistas virtuais sintetizadas com algoritmos DIBR utilizando as imagens de textura originais e os mapas codificados com o LFP têm melhor qualidade perceptual quando comparadas às vistas virtuais geradas utilizando os mapas codificados com diversos algoritmos testados. Neste trabalho investigamos técnicas para melhorar o desempenho do LFP na codificação de mapas de profundidade. Consideramos usar, além de funções lineares, funções constantes e quadráticas para aproximar o resíduo de predição, e ainda avaliar o impacto da quantização dos coeficientes da função de aproximação na qualidade das vistas virtuais sintetizadas com algoritmos DIBR a partir de mapas codificados com o LFP.

1.2 Imagens de textura

Em imagens de textura, os métodos de compressão estado da arte, como H264/AVC [4] e HEVC [6], aplicam transformadas matemáticas nos blocos de resíduos resultantes da etapa de predição. As transformadas empregadas nesses métodos têm

a propriedade de compactar a informação nos coeficientes de baixa frequência no domínio da transformada. Uma quantização fina é aplicada nos coeficientes de baixa frequência enquanto que os coeficientes de alta frequência são quantizados grosseiramente. Essa estratégia de quantização favorece a qualidade percebida pelo sistema visual humano, pois a informação contida na faixa de frequência percebida é preservada, de acordo com a fidelidade desejada.

Os métodos baseados em transformadas assumem que as imagens são suaves, isso implica que a transformada aplicada no bloco de resíduo concentrará a energia nos coeficientes de baixa frequência. Apesar dos excelentes resultados para essa classe de imagens, essa consideração restringe esses algoritmos.

Apesar dos resultados promissores alcançados com o LFP na codificação de mapas de profundidade, a adaptação do LFP para a codificação de imagens de textura ainda não foi investigada. Neste trabalho desenvolvemos um algoritmo a partir do LFP para a codificação de imagens de textura.

1.3 Organização da dissertação

No capítulo 2 revisamos algumas técnicas que são comuns em algoritmos de compressão de dados, revisamos brevemente os principais conceitos de teoria da informação que são fundamentais para algoritmos de compressão.

No capítulo 3 é descrito com detalhes o algoritmo LFP, os modos de predição utilizados, a aproximação do resíduo de predição com funções lineares, o esquema de quantização dos coeficientes da função de aproximação, o processo de segmentação dos blocos e da predição, assim como a técnica de otimização taxa-distorção utilizada.

As técnicas que propomos utilizar no LFP para a codificação de mapas de profundidade são descritas no capítulo 4, bem como os resultados alcançados em cada esquema.

As técnicas prospostas a partir do LFP, assim como os resultados alcançados, considerando a aplicação em codificação de imagens de textura, são apresentadas no capítulo 5.

No capítulo 6 são apresentadas as conclusões a respeito dos resultados experimentais obtidos com os esquemas prospostos e sugestões de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Neste capítulo será feita uma breve revisão de teoria da informação, bem como os resultados e interpretações fundamentais, e ainda das principais técnicas compartilhadas pelos algoritmos de compressão de imagens.

2.1 Resultados de teoria da informação

O campo da teoria da informação tem se desenvolvido após a publicação de *A Mathematical Theory of Communication* de Shannon [8]. Nesse trabalho, os conceitos fundamentais e as formulações matemáticas apresentados, tornaram possíveis os modernos sistemas de comunicações e padrões de compressão de dados. A partir de uma abordagem intuitiva definiu-se conceitos como entropia, entropia conjunta e condicional, informação mútua e a teoria taxa-distorção. Nesta seção será feita uma breve revisão dos conceitos relevantes ao desenvolvimento deste trabalho.

2.1.1 Entropia, informação mútua

Segundo Shannon, a entropia é uma grandeza que mede a incerteza, a informação contida em uma variável aleatória (VA). Neste trabalho, estamos interessados nas definições para o caso de uma VA discreta.

A partir de três propriedades desejáveis, listadas em seguida, a entropia foi definida. Sejam $\{e_1, e_2, ..., e_n\}$, os eventos de uma VA e $\{p_1, p_2, ..., p_n\}$ as probabilidades desses eventos: (1) a entropia de uma VA é contínua em p_i; (2) se as probabilidades dos eventos dessa VA são todas iguais a $p_i = \frac{1}{n}$, a entropia $H(p_1, p_2, ...p_n)$ é uma função monótona crescente[1] de n; (3) Supondo que os eventos sejam definidos como $A = \{e_1, e_2\}$ e $B = \{e_3, .., e_n\}$, para indicar a ocorrência de um evento precisamos indicar primeiro o grupo ao qual ele pertence, e então, indicar o evento desse grupo, cada etapa com uma probabilidade associada. A entropia da VA com eventos definidos dessa forma não deve diferir da entropia com eventos definidos

como originalmente. A expressão que satisfaz essas propridades é definida por:

$$H = -\sum_{i=1}^{n} p_i \log_b (p_i). \tag{2.1}$$

A base logarítmica b define a unidade da entropia, para a base 2 a unidade é *bits* por símbolo. Apenas para exemplificar, podemos verificar pela equação 2.1 que se um evento tem probabilidade de ocorrer igual a 1, a entropia será igual a zero, o que faz sentido pois, nesse caso, o evento seria determinístico, não teríamos nenhuma incerteza a respeito dele.

Outros resultados da teoria da informação, intimamente relacionados à entropia, são citados pois fundamentam teoricamente as técnicas de compressão de dados, além de nos permitir fazer análises das premissas assumidas nesses algoritmos.

Sejam X e Y duas VAs discretas com função de probabilidade conjunta $p(x, y)$ e funções de probabilidades marginais $p(x)$ e $p(y)$ [9] .

A entropia conjunta $H(X, Y)$, que apenas é a aplicação direta do conceito de entropia para o caso da probabilidade conjunta de $p(x, y)$, é definida por:

$$H(X, Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_b p(x, y) \tag{2.2}$$

A entropia condicional $H(Y|X)$ é dada por:

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ H(Y|X) &= -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_b p(y|x) \end{aligned} \tag{2.3}$$

A informação mútua quantifica a informação de X contida em Y. É definida matematicamente por:

$$\begin{aligned} I(X;Y) &= \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_b \frac{p(x, y)}{p(x)p(y)} \\ I(X;Y) &= H(X) - H(X|Y) \end{aligned} \tag{2.4}$$

Uma conclusão direta desse conceito é que a informação mútua é a redução da entropia de X quando se conhece Y [8], [10], [11]. O conceito de informação mútua tem papel central na teoria taxa-distorção, que estabelece o limite fundamental em compressão com perdas, que se pode minimizar a taxa relaxando o critério de

[1]Uma função entre dois conjuntos ordenados é monótona crescente quando ela preserva a relação de ordem. $\forall(x \text{ e } y) \in A,$ se $x > y, \text{então} f(x) > y.$

fidelidade.

2.1.2 Teoria taxa-distorção

Na maioria das aplicações de compressão de imagens usam-se métodos de compressão com perdas, a teoria taxa-distorção descreve as relações dos requisitos conflitantes de minimizar a taxa e a distorção. Por exemplo, comumente se deseja representar uma fonte de informação com uma quantidade menor de *bits* por símbolo, admitindo uma distorção criteriosa, logo, temos que achar um bom compromisso entre a compressão desejada e a distorção. Shannon estabeleceu a relação matemática entre taxa e distorção, originalmente tratando do problema de quantização, de representar uma variável aleatória contínua com um número finito de *bits*. Para a compressão de imagens digitais os resultados para o caso discreto são utilizados.

Seja χ o alfabeto de uma fonte de informação X. E ainda, $\hat{X}$ e $\tilde{X}$ duas fontes com alfabetos de reconstrução reduzidos $\hat{\chi}$ e $\tilde{\chi}$, respectivamente. A medida de distorção $d(x, \hat{x})$ é uma aplicação de símbolo do par de alfabeto fonte-reconstrução definida por $d : \chi \times \hat{\chi} \to \mathbb{R}^+$. Shannon enfatizou que um critério de fidelidade é uma função de $p(x, y)$ e deve ser ordenado, ou seja, que seja possível afirmar que a qualidade da reconstrução usando $\hat{\chi}$ seja maior, menor ou igual à qualidade usando $\tilde{\chi}$. Diversas funções têm sido empregadas como medida de distorção, deseja-se usar uma equação que aproxime satisfatoriamente o requisito de uso final, que pode ser a qualidade perceptual para o sistema visual humano numa aplicação de compressão de imagem. A distorção que temos quando aproximamos a fonte X por uma fonte $\hat{X}$ usando a função de distorção d é definida por:

$$D = \sum_{i=0}^{m} \sum_{j=0}^{n} p(x_i, \hat{x_j}) d(x_i, \hat{x_j}) \tag{2.5}$$

Onde x_i é um símbolo do alfabeto χ da fonte X e $\hat{x}_j$ é um símbolo do alfabeto $\hat{\chi}$ da fonte $\hat{X}$.

A distorção mede o custo de representarmos a fonte X com alfabeto χ usando a fonte $\hat{X}$ com alfabeto $\hat{\chi}$. A função taxa-distorção $R(D)$, sujeita à restrição (D) (equação 2.5), é definida por:

$$R(D) = \min \{I(X; \hat{X})\} \tag{2.6}$$

Usando o resultado da equação 2.4, a informação mútua é a redução de entropia de X quando conhecemos $\hat{X}$. Podemos verificar que no caso de reconstrução perfeita a taxa R seria igual à entropia de X, ou seja, não teríamos nenhuma redução na entropia. Se desejarmos atingir um nível de compressão abaixo de $H(X)$ temos de admitir uma distorção, e na compressão com perdas a tarefa é balancear o compro-

misso de minimizar conjuntamente a taxa e a distorção. A função $R(D)$ estabelece o limite que podemos minimizar a taxa para uma dada distorção aceitável. Em [8] e [10] a teoria taxa-distorção é abordada de forma mais rigorosa e detalhada.

2.2 Compressão de Imagens

Apesar da diversidade de métodos de compressão de imagens, existe um conjunto comum de técnicas aos vários algoritmos. São exemplos dessas técnicas as transformadas, a quantização, codificação entrópica e predição. Alguns padrões de compressão usam transformadas como a *discrete cosine tranform* (DCT) e *discrete wavelet transform* (DWT) para concentrar a informação em alguns coeficientes. Na quantização pode seguir-se a estratégia de priorizar alguns coeficientes em detrimento de outros, fazendo a quantização de maneira mais ou menos grosseira. O método de codificação dos coeficientes quantizados adotado contribui fortemente no desempenho do algoritmo. Códigos de *Huffman* e codificadores aritméticos são os métodos mais utilizados na codificação sem perda de coeficientes. Pode-se dizer que esses são elementos típicos de um método de compressão genérico, pesquisas objetivam otimizar esses elementos. Nesta seção revisamos brevemente as técnicas utilizadas no capítulo 3.

2.2.1 Predição

Em imagens naturais a transição entre *pixels* geralmente é suave, pois *pixels* vizinhos têm valores próximos, indicando que existe uma correlacão entre amostras espacialmente próximas. É comum classificarmos as imagens em suaves e não-suaves dependendo da característica de transição entre seus *pixels*.

Num método de compressão, o bloco de predição busca retirar essa correlação com o propósito de gerar um resíduo com menos redundância, com distribuição concentrada, contribuindo para uma melhor taxa de compressão. Na literatua esses métodos são chamados de *differential encoding* ou ainda *predictive coding*. São usados para retirar redundância tanto temporal quanto espacial na codificação de vídeo, imagem e voz. Em vez de codificar diretamente o sinal, codifica-se a diferença entre a predição e o sinal original. Para cada aplicação concebe-se um modelo dependendo das características da fonte. Para vídeo, por exemplo, geralmente há muita redundância entre quadros e o modelo explora esse fato [11], [5] e [12].

O objetivo é projetar um modelo que aproxime bem a fonte de informação a fim de se obter um resíduo com menos energia que a fonte original. O resultado obtido, apesar de poder excursionar num intervalo maior, tem sua densidade mais concentrada.

Seja x_i o valor atual que dever ser predito, $\tilde{x}_i$ a predicão e $\hat{x}_{i-1}, \hat{x}_{i-2}, ..., \hat{x}_{i-n}$, n amostras passadas reconstruídas da fonte.

A predição poder ser definida por:

$$\tilde{x}_i = f(\hat{x}_{i-1}, \hat{x}_{i-2}, ..., \hat{x}_{i-n}) \tag{2.7}$$

Resumidamente, o projeto de um preditor é a busca de uma função f (equação 2.7) que aproxime bem a fonte minimizando a variância do erro ${\sigma^2}_e$ de predição :

$${\sigma^2}_e = \min\{E[(x_i - \tilde{x}_i)^2]\} \tag{2.8}$$

Num algoritmo de compressão onde o resíduo seja quantizado, as amostras passadas reconstruídas são usadas para a predição, dessa forma tanto o quantizador quanto o decodificador realizam a mesma predição. Em [11], técnicas de predição são discutidas mais detalhadamente.

2.2.2 Quantização

A quantização objetiva descrever com um número reduzido de valores representativos os possíveis valores das amostras de um sinal observando um critério de distorção.

Numa análise intuitiva podemos notar que a estratégia de quantização tem impacto direto no desempenho de métodos de compressão com perdas. Por exemplo, se fizermos uma quantização grosseira iremos reduzir o número de possíveis valores que as amostras do sinal podem assumir e dependendo da codificação que usarmos, código de tamanho fixo ou variável, pode resultar numa taxa menor que a entropia da fonte original. No entanto, estamos penalizando a fidelidade de reconstrução do sinal. Por outro lado, se refinarmos a quantização, aumentamos o número de valores que as amostras podem assumir e ganhamos na fidelidade de reconstrução, mas talvez isso não resulte na taxa de compressão pretendida. Na quantização temos que equilibrar estes dois objetivos, reduzir os possíveis valores da fonte a alguns valores representativos e manter a fidelidade de reconstrução num nível desejado.

Duas abordagens podem ser seguidas no projeto do quantizador, podemos tratar o problema de quantização e atribuição de código de forma separada ou podemos incluir a etapa de definição de código no projeto do quantizador. Essa última opção é mais complexa, especialmente com códigos de tamanho variável, sendo a primeira geralmente mais utilizada por simplicidade [11] e [12].

Em poucas palavras, a idéia da quantização é representar as amostras da fonte com um número M reduzido de valores com o mínimo de distorção.

A quantização é o mapeamento de x em y_i:

$$y_i = Q(x), \quad \text{se somente se,} \quad l_i < x < l_{i+1} \tag{2.9}$$

Queremos determinar o número de valores M, os limites dos intervalos $\{l_1, l_2, ..., l_{M+1}\}$ e os valores de reconstrução $\{y_1, y_2, ..., y_M\}$ que minimizam a distorção D na representação da fonte de densidade $p(x)$. Temos então:

$$D = E\{f(x - y_i)\} \tag{2.10}$$

$$D = \sum_{i=1}^{M} \int_{x_i}^{x_{i+1}} f(x - y_i) p(x)\, dx \tag{2.11}$$

Onde f é uma função diferenciável do erro $e = (x - y_i)$.

Em [11] e [12], são apontados vários exemplos de projeto de quantizadores, soluções para distribuições uniformes e não uniformes, bem como o que devemos levar em conta no projeto de quantizadores com distorção mínima. Em [13] é proposta uma solução iterativa para se obter os quantizadores para uma distorção mínima fixando-se o número de valores de reconstrução M, esse método muito utilizado é conhecido como algoritmo *Lloyd-Max*.

2.2.3 Codificação entrópica

A codificação resume-se em definir um código para cada valor ou sequência de valores fornecidos por uma fonte de informação. É razoável atribuirmos códigos curtos para valores mais frequentes e códigos longos para valores menos frequentes. Essa premissa nos indica que a probabilidade de cada símbolo ou conjunto de símbolos do alfabeto será levada em conta na abordagem que adotarmos para a definição de código.

Além de comprimento médio mínimo, é desejável que os códigos sejam não singulares, ou seja, que cada símbolo tenha um código distinto. Assim, será possível ao decodificador saber o valor correspondente a cada código. A atribuição de códigos não singulares (unicamente decodificáveis) aos símbolos individuais não implica que sequências de símbolos sejam unicamente decodificáveis também. Portanto, é desejável atribuirmos códigos de maneira que tanto símbolos individuais quanto sequências de símbolos sejam unicamente decodificáveis. É comum que, mesmo com códigos unicamente decodificáveis, não seja possível decodificar os símbolos instantaneamente. Ou seja, às vezes temos que ler valores futuros (no *bitstream*) para decodificar o valor atual, o que pode ser impraticável em algumas aplicações [10], [11].

Baseado nessas características desejadas do código, foram desenvolvidos os

primeiros resultados teóricos e métodos de codificação. Pelo teorema de *Kraft-McMillan*[2] [14] sabemos que um código instantâneo como também um código unicamente decodificável satisfaz a inequação 2.12 ou dado um código que satifaz a inequação 2.12 pode-se encontrar um código unicamente decodificável e instantâneo.

$$\sum_i C^{l_i} \leq 1 \tag{2.12}$$

Onde C é a cardinalidade do alfabeto de código (2 na maioria dos casos práticos) e l_i o comprimento da palavra de código. Já foi mostrado que para minimizar o comprimento médio da palavra de código, observando a inequação 2.12, temos:

$$L = \min\{\sum_i l_i p_i\} \tag{2.13}$$

Matematicamente, a mínima descrição média, conhecendo-se a distribuicação p_i, é atingida quando o comprimento da palavra de código é dada pela equação 2.14. Esse resultado leva à expressão da entropia da fonte, que é a mínima taxa que podemos atingir.

$$l_i^* = \log_C p_i \tag{2.14}$$

Fundamentados nesses resultados, vários métodos de codificação foram propostos. O código de *Huffman* [15] se enquadra nesses métodos e é provado ser um código ótimo, no sentido de menor comprimento médio de código. No entanto, para obtermos a descrição mínima com o código de *Huffman* é necessário agrupar todas as possíveis sequências de símbolos fornecida pela fonte, saber a probabilidade de cada sequência e atribuir um código para cada sequência, o que pode inviabilizar a utilização de *Huffman* dessa forma. Na prática utiliza-se o código de *Huffman* para codificar símbolos individuais e nesse caso o comprimento médio mínimo apenas é alcançado quando a probabilidade dos símbolos da fonte são potências de 1/2, caso pouco provável na prática [16], [17].

O codificador aritmético (CA) é um método muito atrativo, não é necessário listar nem tão pouco atribuir um código para cada possível sequência de símbolos ou símbolo individual, há uma separação entre o modelo de probabilidade e a codificação. Assim pode-se computar e atualizar no decorrer da codificação as probabilidades dos símbolos mantendo um modelo estatístico atualizado da fonte. A codificação com CA atinge o menor comprimento médio de código dada a distribuição de probabilidade do modelo da fonte, para isso é importante ter um modelo que descreve bem estatisticamente a fonte. Além disso, no CA a mensagem inteira é

[2]Kraft provou para o caso de código instantâneo, McMillan chegou no mesmo resultado para código unicamente decodificável.

codificada em vez de codificar símbolos individuais, na implementacão incremental pode-se decodificar assim que o primeiro símbolo for codificado [17],[18]. Enfim, verifica-se que o CA tem as características de um código ótimo: unicamente decodificável, instantâneo e comprimento médio mínimo.

O código de *Huffman* e o codificador aritmético são os métodos mais empregados para codificação sem perdas. Esse último tem sido cada vez mais utilizado nos algoritmos de compressão de imagens, e é utilizado no capítulo 3 pelas razões comentadas.

2.3 Métodos de compressão de imagens

A demanda por conteúdo multimídia visual como imagens e vídeos impulsiona o rápido crescimento da quantidade de dados. Antes de ser armazenada, essa informação precisa ser tratada de forma adequada. O campo de pesquisa de compressão de dados tem a tarefa de descrever a informação contida nesse conteúdo de forma reduzida, com a finalidade de economizar recursos de armazenamento e transmissão. Várias técnicas para melhorar os métodos existentes como também métodos novos têm sido propostos. Existem situações em que se exige que a informação seja, após ser comprimida, recuperada sem perdas e existem casos em que pode-se relaxar a fidelidade de reconstrução, métodos adequados devem ser adotados em função da aplicação. Em geral, as técnicas de compressão de imagens e vídeo são divididas em métodos sem perdas e com perdas.

Despretensiosamente falando, os métodos de compressão baseados em transformadas têm sido aceitos como os que exibem desempenho estado da arte. No padrão *Joint Photographic Experts Group* (JPEG) [3], a imagem é segmentada em blocos de tamanho de 8×8 *pixels* e escalonada (subtrai-se 128 de cada amostra). Após a aplicação da DCT, os coeficientes são quantizados, as diferenças entre os coeficientes são agrupadas de forma conveniente e então o grupo ao qual cada diferença pertence, como também o índice que corresponde ao valor dentro do grupo são codificados com um código de *Huffman*. O padrão JPEG2000 [19] desenvolvido para elevar o desempenho do padrão anterior é baseado na premissa de que as sub-bandas decompostas através da DWT podem ser codificadas de maneira que previlegiem a qualidade visual percebida.

Mais recentemente, o padrão H264/AVC (*Advanced video coding*)[5] tem sido utilizado na compressão de vídeos e imagens. Obtendo-se, geralmente, desempenho superior aos padrões anteriores. As técnicas de predição empregadas no H264/AVC contribuem para esse ganho de desempenho, no H264/AVC é empregada a DCT.

A primeira versão do padrão *High Efficiency Video Coding* (HECV) foi finalizada em janeiro deste ano. E tem como objetivo reduzir até a metade a quantidade de

bits, em relação ao padrão H264/AVC, mantendo a mesma qualidade perceptual. O ganho de desempenho no HEVC é assegurado pela flexibilidade que o algoritmo tem de otimizar, no sentido taxa-distorção, a segmentação e a predição do *Coding Tree Unit* (CTU) [6], [20]. No HEVC são empregados 35 modos de predicão intra [3], a imagem é segmentada em blocos a partir de 64×64 *pixels* o que pode permitir um aumento na eficiência da codificação.

Uma abordagem diferente da de algoritmos que empregam transformadas é aplicada em métodos baseados em quantização vetorial. Um desses métodos que tem alcançado resultado promissor é o *Multidimensional Multiscale Parser* (MMP) [21], tanto em imagens suaves (naturais) como em imagens não-suaves (imagens de texto e gráficos). Deve-se notar que essas últimas não são codificadas eficientemente pelos métodos baseados em transformadas, pois esses assumem que a informação de interesse da fonte se concentra nos coeficientes de baixa frequência.

Para situar o desempenho do método de compressão de imagens utilizado neste trabalho no contexto dos algoritmos de compressão de imagens é necessária uma métrica objetiva. A função utilizada para avaliar a eficácia é a *Peak Signal-Noise Rate* (PSNR), definida como:

$$PSNR = 10 \log_{10} \frac{255^2}{EMQ} \tag{2.15}$$

Onde o erro médio quadrático (EMQ) é:

$$EMQ = \frac{1}{MN} \sum_{i}^{M} \sum_{j}^{N} (I(i,j) - \hat{I}(i,j))^2 \tag{2.16}$$

Em 2.16, M, N são as dimensões da imagem, $I(i,j)$ é a amostra (*pixel*) da imagem original e $\hat{I}(i,j)$ é a amostra da imagem reconstruída na posição (i,j).

[3] Na predição intra apenas blocos codificados (e decodificados) da própria imagem são usados.

Capítulo 3

O Algoritmo LFP

O *Linear Fitting and Flexible Prediction* (LFP)[7] foi proposto e aplicado na codificação de mapas de profundidade partindo da premissa que a imagem de um mapa de profundidades é um conjunto de regiões suaves com bruscas transições nas fronteiras dessas regiões. Partindo dessa observação foi proposto usar técnicas de predição, que são favoráveis em regiões suaves. A predição e a segmentação de cada bloco de imagem de 32×32 *pixels* são realizadas de forma flexível e o resíduo de predição aproximado utilizando funções lineares.

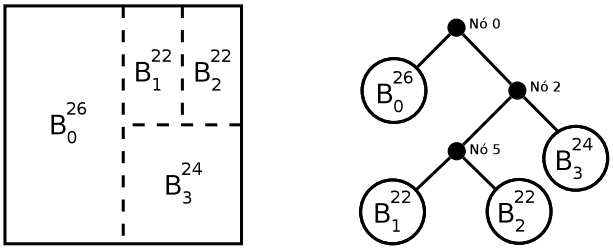
A estratégia de segmentação flexível permite que regiões suaves sejam aproximadas por blocos maiores, contribuindo para uma codificação eficiente. As transições que, em geral, indicam mudanças de profundidade (entre objetos da cena), são codificadas usando técnicas de predição e aproximações com blocos menores preservando a informação da mudança de profundidade.

As decisões de como segmentar um bloco bem como o modo de predição empregado são sinalizados ao decodificador através de *flags*. De cada bloco B^l, da escala l, é subtraída a predição $\tilde{B}^l$ gerando um bloco de resíduo S^l. O resíduo é aproximado por uma função linear ou por uma palavra do dicionário D^l. Os *flags* indicando como a segmentação e a predição foram realizadas são codificados através de um codificador aritmético, assim como a escolha da aproximação do bloco de resíduo por uma função ou por uma palavra do dicionário. Em seguida, os coeficientes da função de aproximação, caso seja escolhida a aproximação por uma função, ou o índice do vetor do dicionário da escala correspondente é codificado.

O decodificador reconstrói a imagem a partir dos *flags* de segmentação e predição bem como o método utilizado para aproximação. Neste capítulo serão descritos os detalhes do algoritmo LFP.

3.1 Processo de segmentação

Inicialmente o algoritmo LFP segmenta a imagem em blocos de 32×32 *pixels*. Os blocos de 32×32 podem ainda ser segmentados em blocos de 32×16, 16×32 e 16×16 *pixels*. A partir de blocos de 16×16 *pixels*, a segmentação é feita sem restrição, permitindo ter 25 dimensões de blocos ($2^v \times 2^w$ com $v, w = 0, 1, .., 4$). *Flags* são codificados de maneira que o decodificador possa reconstruir a imagem a partir das decisões tomadas na segmentação do bloco e da predição. No LFP são empregados cinco *flags*: dois *flags* para indicar o sentido de segmentação, horizontal ou vertical, dois para informar que não há mudança na predição, mas há uma segmentação (vertical ou horizontal), e um para indicar que a predição e o bloco não devem ser segmentados.



(a) bloco (b) árvore

Figura 3.1: Segmentação do bloco

Vamos observar a Figura 3.1b que representa a árvore de segmentação para um bloco de 32×32. [Na notação B_i^l, i indica a ordem de codificação e l a escala a que bloco pertence. Consideramos que os blocos de 32×32 *pixels* pertencem à escala 27 enquanto que os blocos de 1×1 à escala 0]. Para exemplificar, vamos assumir que tanto no bloco B_1^{22} quanto no bloco B_2^{22} foi usado o mesmo modo de predição. As dimensões dos blocos são funções de l: $m = g(l)$ (direção de X) e $n = h(l)$ (direção de Y).

Inicialmente o bloco é segmentado na vertical, um *flag* é enviado para indicar que houve a segmentação vertical do bloco. O bloco B_0^{26} não é segmentado assim como o modo de predição. Portanto, outro *flag* indica essa escolha. O modo de predição é então codificado. O bloco de resíduo é aproximado pela função linear ou por uma palavra do dicionário, um *flag* indica a escolha (função ou dicionário). Em seguida, o índice da palavra do dicionário ou os coeficientes da função de aproximação são codificados, dependendo da escolha. Por hora, vamos apenas dizer que o bloco é aproximado.

No nó 2, o bloco é segmentado na horizontal. Em seguida, o nó 5 é segmentado na vertical mas a predição não muda, como supusemos anteriormente. Um *flag* para indicar a segmentação horizontal e outro para segmentação vertical, sem mudança

de predição, são codificados. Como não há mudança no modo de predição, o modo pode ser enviado. O bloco B_1^{22} não é segmentado e nem tão pouco a predição, então um *flag* é enviado. O resíduo de predição é adequadamente aproximado.

Os demais blocos são codificados da mesma forma, indicando as decisões de segmentação, predição e aproximação. Verifica-se que a codificação segue um ordenamento. Os blocos da esquerda ou de cima, após segmentados, são codificados. Em seguida, os blocos da direita e de baixo são codificados. O decodificador reconstrói a imagem, bloco a bloco, a partir das decisões tomadas pelo codificador de como realizar a segmentação do bloco e da predição bem como o modo de predição e a aproximação empregada.

3.2 Modos de predição

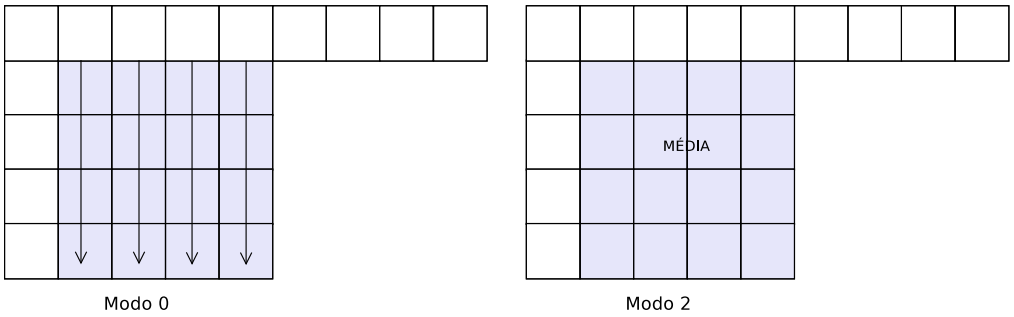
Os modos de predição empredados no LFP são precisamente os modos utilizados no padrão H264/AVC [5], mas de maneira mais flexível. No H264/AVC, para blocos de 4×4 das amostras *luma*, são permitidos 9 modos de predição. Esses 9 modos são utilizados no LFP assumindo que os blocos de referência estão disponíveis. No canto esquerdo superior de uma imagem, por exemplo, nenhum dos modos de predição podem ser aplicados. Nesse caso, o LFP prevê que as amostras do bloco têm valor 128.

No LFP, os 9 modos de predição podem ser aplicados em qualquer bloco de tamanho maior ou igual que 4×4 *pixels*. Ou seja, a predição é aplicada em blocos desde 32×32 até 4×4 *pixels*. Além disso, o filtro de atenuação de efeito de blocos que é aplicado no H264/AVC não é utilizado.

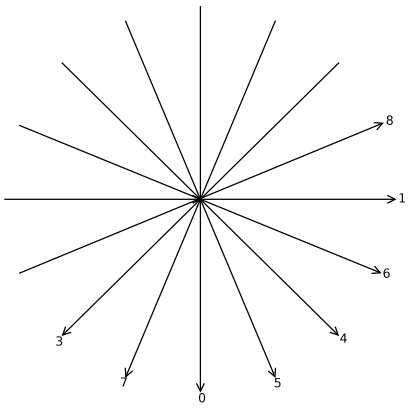
As direções dos modos de predição utilizados no LFP estão resumidas na Figura 3.2b. Na descrição do processo de segmentação, indicação do modo de predição e indicação da aproximação do resíduo, no exemplo da seção anterior, assim que não houver mudança no modo de predição, o codificador informa ao decodificador 1 dos 9 modos de predição que podem ser utilizados. Dessa forma, o decodificador poderá utilizar o mesmo modo na reconstrução da imagem.

As amostras de referência, *pixels* brancos na Figura 3.2a, são da imagem reconstruída. O codificador enquanto codifica também reconstrói a imagem. E usa *pixels* já codificados (e decodificados) como referência para a realizar a predição. Dessa forma tanto o codificador quanto o decodificador realizam exatamente a mesma predição.

A predição $\tilde{B}^l$ é subtraída do bloco original B^l gerando um resíduo S^l. Esse resíduo é aproximado por uma função ou por um vetor do dicionário da escala correspondente do bloco. Na seção seguinte é descrita aproximação. Em cada nível



(a) Modos de predição 0 e 2



(b) Direções dos modos de predição

Figura 3.2: Modos de predição usados no LFP

da árvore segmentação, o modo de predição que produz o resíduo com menor norma $L1$ é utilizado. Ou seja, todos os modos de predição disponíveis são testados e é utilizado o que fornecer o resíduo com menor norma.

3.3 Aproximação dos blocos de resíduo

Cada bloco de resíduo S^l é aproximado por uma função (equacão 3.1) ou usando uma palavra do dicionário da escala l. O dicionário é composto de blocos de resíduo aproximados anteriormente por funções lineares, cada escala tendo um próprio dicionário.

Os dicionários de todas as escalas são inicializados com um vetor nulo. Apenas o dicionário da escala 0 (1×1) é inicializado com valores de -255 a 255, que são os possíveis valores das amostras de resíduo. A cardinalidade do dicionário de cada escala cresce durante a codificação, conforme são adicionados blocos de resíduo aproximados por funções lineares. A cardinalidade dos dicionário está limitada à mil elementos. [Notemos que o dicionário de blocos de resíduo aproximados é apenas um dicionário de coeficientes de funções de aproximação. Os blocos de resíduo são armazenados dessa forma no dicionário por conveniência da implementação].

A função de aproximação é definida como:

$$f(\tilde{x}, \tilde{y}) = \alpha_0 \tilde{x} + \alpha_1 \tilde{y} + \alpha_2 \tag{3.1}$$

Onde,

$$\tilde{x} = (x - 2^{m+1}) \quad \tilde{y} = (y - 2^{n-1} - 1) \tag{3.2}$$

m é o comprimento horinzontal do bloco, n é o comprimento vertical do bloco.

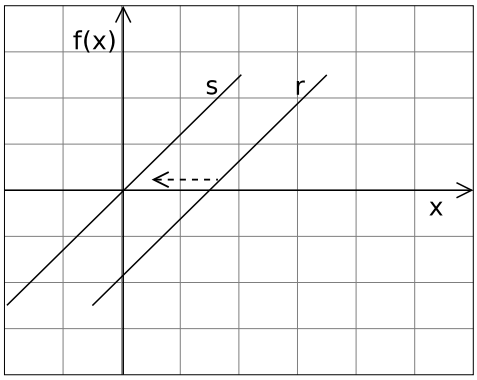


Figura 3.3: Translação horizontal

Na equação 3.1, $\tilde{x}$ e $\tilde{y}$ são as versões deslocadas das coordenadas dos *pixels* no bloco. Dessa forma o termo α_2 da equação 3.1 é igual à média dos valores do bloco de resíduo. Numa boa predição, o resíduo tem densidade concentrada e centrada em zero. Portanto, o coeficiente α_2 pode ser eficientemente codificado para blocos de média zero. Na Figura 3.3, podemos verificar o resultado do deslocamento das coordenadas para o caso de duas dimensões.

Os coeficientes são determinados minimizando a norma $L2$ do erro de aproximação.

$$E = \| e \|^2 = \sum_{i=1}^{m \times n} {e_i}^2$$
$$e_i = (s_i - (\alpha_0 \tilde{x_i} + \alpha_1 \tilde{y_i} + \alpha_2)) \tag{3.3}$$

Onde s_i é uma amostra do resíduo. O erro E quando aproximamos o bloco de resíduo por uma função linear é uma função dos coeficientes α_0, α_1 e α_2. Para um bloco de resíduo S^l com $m \times n$ amostras, temos:

$$E(\alpha_0, \alpha_1, \alpha_2) = \sum_{i=1}^{m \times n} (s_i - (\alpha_0 \tilde{x_i} + \alpha_1 \tilde{y_i} + \alpha_2))^2 \tag{3.4}$$

Ao derivar E em relação aos coeficientes, igualando a zero e depois rearrumando

as equações, obtemos a equação:

$$[\hat{A}][c] = [\hat{b}] \tag{3.5}$$

Onde,

$$[\hat{A}] = [A^t][A] \qquad [\hat{b}] = [A^t][b]$$

E por sua vez,

$$[A] = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_{mn} & y_{mn} & 1 \end{bmatrix} \qquad [b] = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{mn} \end{bmatrix} \qquad [c] = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix}$$

Os coeficientes são obtidos fazendo a decomposição na matriz triangular inferior e superior $[\hat{A}] = [L][U]$. Resolve-se para $[L][d] = [\hat{b}]$, onde $[d] = [U][c]$, e em seguida para $[U][c] = [d]$ [22],[23] .

Os coeficientes α_0 e α_1 são escalados pela metade do comprimento da dimensão. Por exemplo, α_0 é multiplicado por $m/2$ [1], o coeficiente α_2 não é escalado. $[-255, 255]$ é o intervalo de valores do coeficiente α_2 (igual ao intervalo do resíduo). Os coeficientes α_0 e α_1 variam no intervalo $[-127, 127]$, pois o coeficiente α_2 guarda a média do bloco de resíduo.

Os coeficientes são quantizados, não uniformemente, usando o passo de quantização q a seguir:

Para α_2

$$q = \begin{cases} 1, & \text{se} \quad -10 < \alpha_2 < 10, \\ 4, & \text{se} \quad -22 < \alpha_2 \leq -10 \quad \text{ou} \quad 10 \leq \alpha_2 < 22, \\ 8, & \text{se} \quad -86 < \alpha_2 \leq -22 \quad \text{ou} \quad 22 \leq \alpha_2 < 86, \\ 13, & \text{se} \quad \alpha_2 \leq -86 \quad \text{ou} \quad \alpha_2 \geq 86, \end{cases}$$

Para α_0 e α_1

$$q = \begin{cases} 1, & \text{se} \quad -10 < \alpha_0, \alpha_1 < 10, \\ 4, & \text{se} \quad -22 < \alpha_0, \alpha_1 \leq -10 \quad \text{ou} \quad 10 \leq \alpha_0, \alpha_1 < 22, \\ 8, & \text{se} \quad -62 < \alpha_0, \alpha_1 \leq -22 \quad \text{ou} \quad 22 \leq \alpha_0, \alpha_1 < 62, \\ 13, & \text{se} \quad \alpha_0, \alpha_1 \leq -62 \quad \text{ou} \quad \alpha_0, \alpha_1 \geq 62, \end{cases}$$

Os coeficientes depois de quantizados são codificados usando um codificador aritmético, a escala l é usada como contexto. O bloco reconstruído a partir dos

[1] m: Tamanho do bloco da direção de X.

coeficientes da função de aproximação é adicionado ao dicionário da escala l, podendo ser reutilizado de forma eficiente para aproximar blocos seguintes. Na Figura 3.4 pode-se verificar a aproximação por planos.

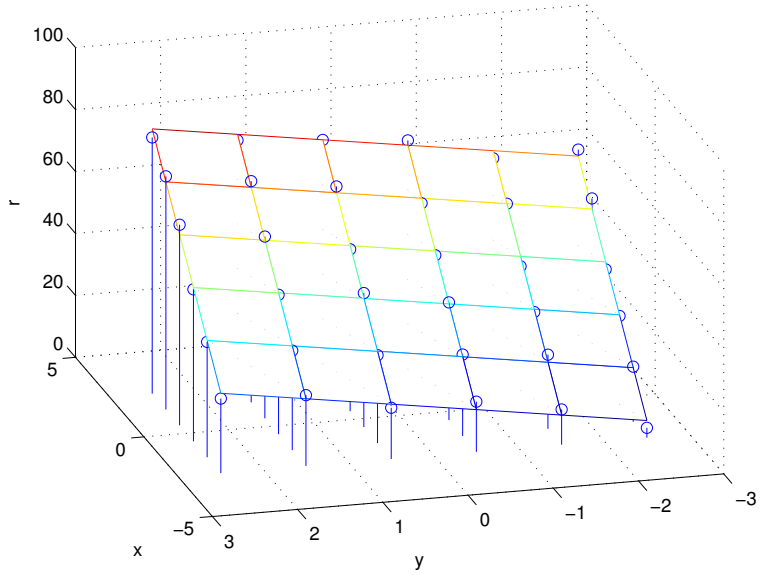


Figura 3.4: Aproximação do resíduo por planos

3.4 Otimização taxa-distorção

A árvore de segmentação $\mathcal{T}$ de cada bloco de 32×32 *pixels* é otimizada de maneira que minimize a função custo J na equação 3.6:

$$J(\mathcal{T}) = D(\mathcal{T}) + \lambda R(\mathcal{T}) \tag{3.6}$$

Onde $D(\mathcal{T})$ é a distorção, $R(\mathcal{T})$ é a taxa. O parâmetro λ pondera o compromisso entre a taxa desejada e a distorção incorrida. O erro absoluto é a métrica de distorção utilizada no LFP, pois foi verificado experimentalmente que ela é superior à métrica de erro quadrático [7]. Podemos notar que o custo de codificar cada bloco de 32×32 depende das escolhas de segmentação, da predição utilizada e da aproximação escolhida em cada escala.

A árvore de segmentação $\mathcal{T}$ ótima, para cada bloco de 32×32, é escolhida podando os ramos a partir da árvore inteiramente expandida. Verificando todas as combinações de segmentação, predição e aproximação do resíduo [7]. Em cada escala l o bloco é segmentado tanto na vertical quanto na horizontal. O custo de codificar o bloco inteiro (sem segmentação) é comparado com os custos de codificar os blocos resultantes da segmentação vertical e horizontal. A estratégia que resultar

no menor custo é escolhida. Essa técnica é aplicada recursivamente de baixo para cima, ou seja, a partir de blocos da escala 0 até os blocos da escala 27.

Em cada etapa, para cada bloco, é utilizado o modo de predição que fornecer o resíduo de menor norma $L1$ e aproximação com menor custo, em termos de *bits* e distorção. As equações 3.7 e 3.8 medem, respectivamente, os custos de aproximação da função e do vetor do dicionário.

$$J_{fun}(S^l) = D_{fun}(S^l) + \lambda[R(flg_{fun}) + \sum_{j=0}^{2} R(\alpha_j)] \tag{3.7}$$

$$J_{dic}(S^l) = D_{dic}(S^l) + \lambda[R(flg_{dic}) + R(ind)] \tag{3.8}$$

Onde flg_{fun} é o *flag* que indica o uso da função como aproximação, α_j o coeficiente da função, flg_{dic} é o *flag* que indica o uso de um vetor do dicionário para a aproximação e ind é o índice do vetor do dicionário. O resultado da otimização fornece a melhor escolha de segmentação do bloco e da predição, do modo de predição, e aproximação.

Neste capítulo descrevemos o algoritmo LFP, os modos de predição empregados, a segmentação dos blocos, a aproximação do resíduo com funções lineares e o processo de otimização da árvore de segmentação. No próximo capítulo abordamos as técnicas propostas para melhorar o desempenho do algoritmo na codificação de mapas de profundidade. Os resultados experimentais dos esquemas propostos são discutidos.

Capítulo 4

Contribuições ao algoritmo LFP na codificação de mapas de profundidade

Neste capítulo investigamos técnicas para melhorar o desempenho do algoritmo LFP na codificação de mapas de profundidade. As técnicas utilizadas, assim como os resultados alcançados são apresentados. Os resultados alcançados com o algoritmo são comparados com os resultados exibidos por métodos estado da arte.

4.1 Codificação de imagens de mapa de profundidades

Nesta seção investigamos técnicas que incorparadas no LFP possam contribuir para um melhor desempenho do algoritmo na codificação de imagens de mapa de profundidades. Para avaliar o desempenho do algoritmo na codificação de mapas de profundidade, geramos — a partir dos mapas codificados e das imagens de textura originais — imagens virtuais com o algoritmo de síntese *Depth Image Based Rendering* (DIBR)[1][2]. Calculamos a PSNR em relação às vistas virtuais sintetizadas utilizando as imagens e os mapas, ambos originais. Dessa forma, avaliamos apenas os eventuais erros imputados pela codificação dos mapas, pois as imagens de textura originais (sem codificação) são utilizadas. O algoritmo DIBR de referência utilizado no desenvolvimento do HEVC-3D[24] foi utilizado para gerar as vistas virtuais.

A Figura 4.1 pode facilitar o entendimento do esquema utilizado para avaliar a qualidade visual das vistas virtuais sintetizadas.

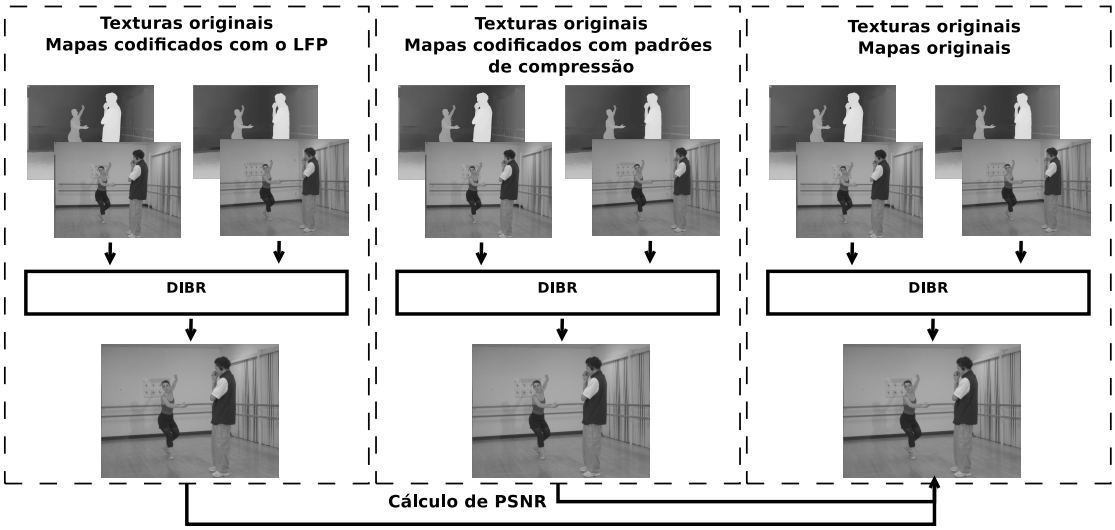


Figura 4.1: Esquema utilizado para avaliar a qualidade visual da vista virtual sintetizada

4.2 Aproximação do resíduo de predição

No LFP a aproximação do resíduo de predição é feita utilizando funções lineares. Apesar dos resultados obtidos com o LFP serem superiores aos resultados obtidos com diversos algoritmos testados [7], a aproximação do resíduo utilizando funções diferentes da linear não foi investigada.

Em geral, a aproximação de funções utilizando funções polinomiais é melhor realizada quando se emprega polinômios de maior grau. No entanto, no contexto de compressão de imagens, em particular no algoritmo LFP, a aproximação do resíduo de predição com polinômios de grau maior implica em codificar uma quantidade maior de coeficientes, apesar de eventual melhor aproximação. Se a aproximação é realizada usando polinômios de grau menor, economizamos na codificação de coeficientes, mas a imagem reconstruída pode apresentar maior distorção. Partindo dessa premissa, é importante balancear uma melhor aproximação do resíduo de predição com o dispêndio de taxa na codificação de coeficientes.

Para investigar o efeito das funções de aproximação do resíduo no desempenho do algoritmo LFP, consideramos utilizar, além de funções lineares, funções constantes e quadráticas. A avaliação do desempenho é feita em termos da taxa de compressão dos mapas de disparidade e da qualidade (medida da PSNR) das vistas virtuais geradas. A PSNR é calculada em relação às vistas virtuais geradas a partir dos mapas originais e das imagens de textura originais.

As funções constantes e quadráticas empregadas na aproximação do resíduo de predição são definidas respectivamente como:

$$g(\tilde{x}, \tilde{y}) = \alpha_0 \tag{4.1}$$

$$h(\tilde{x}, \tilde{y}) = \alpha_0\tilde{x}^2 + \alpha_1\tilde{y}^2 + \alpha_2\tilde{x} + \alpha_3\tilde{y} + \alpha_4\tilde{x}\tilde{y} + \alpha_5 \tag{4.2}$$

Onde $\tilde{x}$ e $\tilde{y}$ são as versões deslocadas das coordenadas dos *pixels* no bloco, definidas como:

$$\tilde{x} = (x - 2^{m+1}) \quad \tilde{y} = (y - 2^{n-1} - 1) \tag{4.3}$$

A translação das coordenadas é realizada no algoritmo original para que o coeficiente α_2 da função de aproximação linear (equação 3.1) armazene a média aproximada do bloco de resíduo. Para a aproximação constante e quadrática essa translação foi mantida.

A escolha da melhor função de aproximação é indicada ao decodificador através de um *flag*. No LFP é utilizado um *flag* para informar a aproximação utilizando um vetor do dicionário ou usando uma função (linear, pois originalmente apenas a função linear é utilizada). Além desse *flag*, aqui utilizamos outro *flag* para informar o grau da função de aproximação: constante, linear ou quadrática. Em seguida, os coeficientes da função de aproximação são codificados. Para a função constante (equação 4.1) o coeficiente α_0 é a média do bloco de resíduo.

Para a função quadrática (equação 4.2) minimizamos a norma $L2$ do erro de aproximação, definida por:

$$E = \| e \|^2 = \sum_{i=1}^{m \times n} {e_i}^2$$
$$e_i = (s_i - (\alpha_0\tilde{x}^2 + \alpha_1\tilde{y}^2 + \alpha_2\tilde{x} + \alpha_3\tilde{y} + \alpha_4\tilde{x}\tilde{y} + \alpha_5)) \tag{4.4}$$

Onde s_i é uma amostra do resíduo. O erro E quando aproximamos o bloco de resíduo por uma função quadrática é uma função dos coeficientes α_0, α_1, α_2, α_3, α_4 e α_5. Para um bloco de resíduo S^l com $m \times n$ amostras, temos:

$$E(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5) = \sum_{i=1}^{m \times n} \left(s_i - (\alpha_0\tilde{x}^2 + \alpha_1\tilde{y}^2 + \alpha_2\tilde{x} + \alpha_3\tilde{y} + \alpha_4\tilde{x}\tilde{y} + \alpha_5)\right)^2 \tag{4.5}$$

Ao derivar E em relação aos coeficientes, igualando a zero e depois rearrumando as equações, obtemos a equação:

$$[\hat{A}][c] = [\hat{b}] \tag{4.6}$$

Onde,

$$[\hat{A}] = [A^t][A] \qquad [\hat{b}] = [A^t][b]$$

E por sua vez,

$$[A] = \begin{bmatrix} x_1^2 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & y_2^2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{mn}^2 & y_{mn}^2 & x_{mn} & y_{mn} & 1 \end{bmatrix} \qquad [b] = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{mn} \end{bmatrix} \qquad [c] = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix}$$

Os coeficientes são obtidos fazendo a decomposição na matriz triangular inferior e superior $[\hat{A}] = [L][U]$. Resolve-se para $[L][d] = [\hat{b}]$, onde $[d] = [U][c]$, e em seguida para $[U][c] = [d]$ [22],[23].

O algoritmo calcula o custo de aproximação usando um vetor do dicionário e o custo de aproximação usando cada uma das três funções. A aproximação com menor custo é utilizada. O custo de aproximação utilizando um vetor do dicionário é calculado com a equação 3.8. Para o cálculo do custo de aproximação utilizando as funções, usamos a equação 3.7, contudo, além do *flag* que indica a aproximação com a função, é levado em conta o *flag* que indica qual a função. Os coeficientes da função de aproximação são incluídos no cálculo do custo respectivo.

As Figuras 4.2 e 4.3 exibem os resultados para as sequências *Balloons* e *Kendo*. Diversos esquemas de aproximação do resíduo foram usados, permitindo avaliar o efeito na qualidade visual das vistas virtuais geradas. O esquema de quantização original do LFP foi utilizado nessa versão. Nas curvas esboçadas nas Figuras desta seção, *const*, *lin* e *quad* correspondem ao uso das funções: constante, linear e quadrática, respectivamente. A curva *lfp* foi obtida com o algoritmo LFP original. A taxa em *bits* por *pixel* (*bpp*) nos resultados experimentais deste capítulo é calculada somando as taxas gastas na codificação dos mapas.

A partir de inspeção visual dos gráficos, podemos afirmar que é possível obter ganho de desempenho se empregarmos um adequado esquema de aproximação do resíduo de predição. Como podemos observar, os esquemas que têm entre as opções o uso de funções constantes fornecem resultados imediatamente superiores aos demais.

Os resultados empregando diferentes funções de aproximação do resíduo para as sequências *Newspaper*, *PoznanStreet* e *PoznanHall2* podem ser consultados no Apêndice C.1.1.

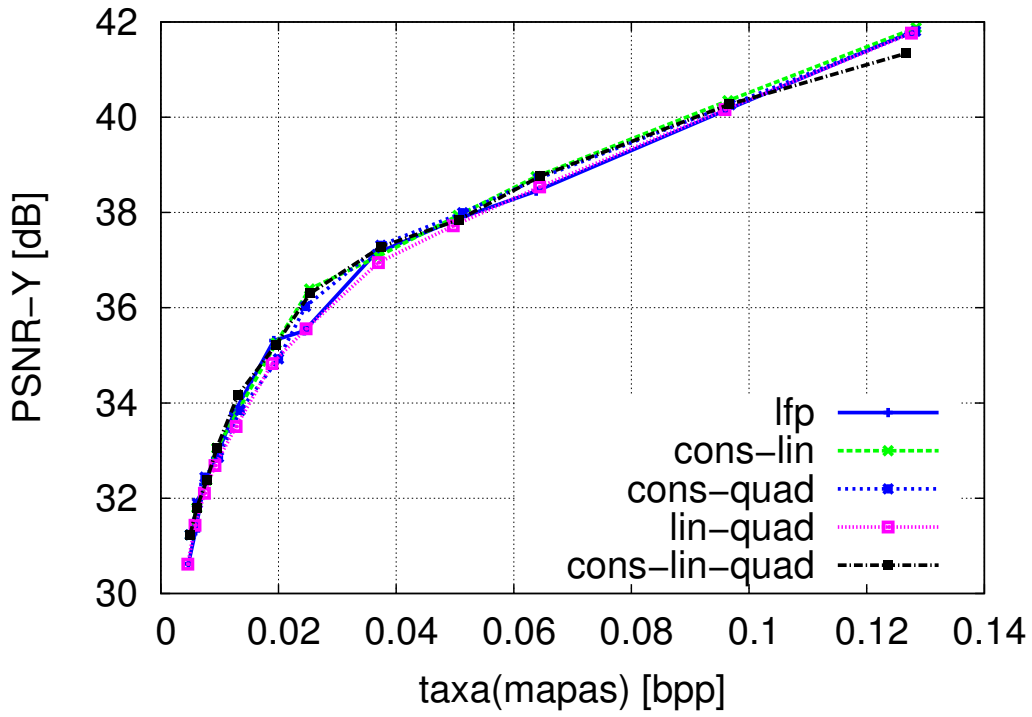


Figura 4.2: Desempenho taxa-distorção para diversas funções de aproximação (*Balloons* câmera 3)

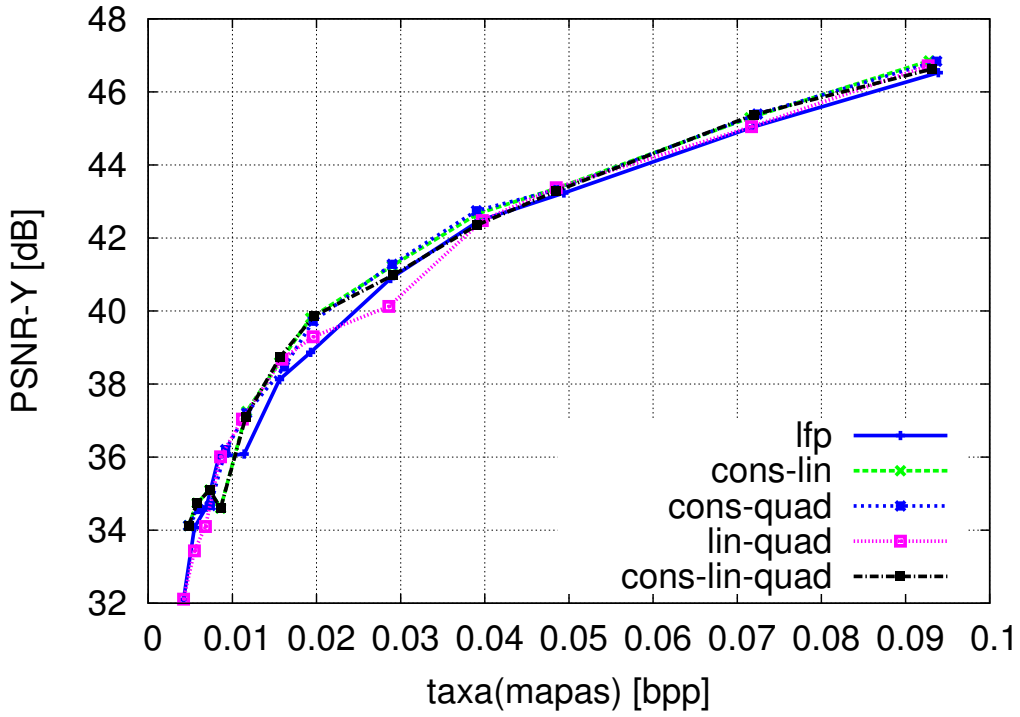


Figura 4.3: Desempenho taxa-distorção para diversas funções de aproximação (*Kendo* câmera 3)

4.3 Quantização dos coeficientes das funções de aproximação

O esquema de quantização empregado no LFP foi definido de maneira heurística, mas intuitiva. Como pode ser verificado na seção 3.3, o esquema de quantização dos coeficientes α_0, α_1 e α_2 prioriza as amostras com valor próximo de zero, fazendo uma quantização mais fina. À medida que o valor das amostras se afasta de zero, a quantização é realizada de forma mais grosseira. Apesar do conhecido desempenho do algoritmo LFP na codificação de mapas de profundidade, o comportamento empregando outros esquemas de quantização não foi investigado.

Para avaliar o efeito da quantização dos coeficientes dos polinômios de aproximação, foram testados quantizadores com quantidades diferentes de intervalos de decisão e níveis de reconstrução. Os quantizadores testados foram concebidos conforme descritos a seguir. A partir das amostras dos valores dos coeficientes das funções de aproximação, estimamos os parâmetros da gaussiana generalizada que modela a distribuição de cada coeficiente [25] [26]. As amostras dos coeficientes foram obtidas, em princípio, utilizando o algoritmo LFP com função de aproximação linear para um conjunto de imagens de textura (imagens de treinamento[1]). Em seguida, geramos dados segundo a gaussiana generalizada que modela a distribuição de cada coeficiente. Então, dada a quantidade de níveis de reconstrução, o esquema de quantização ótimo foi obtido usando o algoritmo *Lloyd-Max*[13][27].

A gaussiana generalizada (equação 4.7) que modela a distribuição de cada coeficiente é completamente definida se conhecermos os valores da média μ, da variância σ^2 e do coeficiente de forma p.

$$gg(x;\mu,\sigma,p) = \frac{1}{2\Gamma\left(1+1/p\right)A\left(p,\sigma\right)} e^{-\left|\frac{x-\mu}{A(p,\sigma)}\right|^p} \tag{4.7}$$

Onde,

$$A(p,\sigma) = \left[\frac{\sigma^2\Gamma(1/p)}{\Gamma(3/p)}\right]^{1/2} \tag{4.8}$$

Nas equações 4.7 e 4.8, $\Gamma(p)$ é a função gama, onde para p inteiro é definida por $\Gamma(p) = (p-1)!$ e para o caso geral é definida por $\Gamma(p) = \int_0^\infty x^{p-1}e^{-p}dx$. A gaussiana generalizada definida pela equação 4.7 corresponde à distribuição gaussiana para $p = 2$. Para $p = 1$ tem-se a distribuição laplaciana.

A média μ foi estimada a partir das amostras dos coeficientes das funções de aproximação para o conjunto de imagens de treinamento. A soma das amostras de cada coeficiente dividida pelo número de amostras. Depois da média estimada,

[1] As imagens de treinamento utilizadas estão no Apêndice B.

estimamos a variância. O parâmetro de forma p de cada coeficiente foi estimado usando o procedimento descrito em [25][26].

Temos que assegurar que em cada intervalo de quantização haja uma quantidade suficiente de amostras, que permita a convergência do algoritmo *Lloyd-Max*. O intervalo é dividido em um número de subintervalos igual a 10 vezes a quantidade de níveis de reconstrução. Depois disso, são geradas amostras aleatórias (uniformes) em cada subintervalo. A quantidade de amostras por subintervalo é M vezes a área sob a curva de probabilidade delimitada por cada subintervalo, onde M é a quantidade total de amostras que se deseja gerar. Isso está ilustrado na Figura 4.4 para facilitar o entendimento.

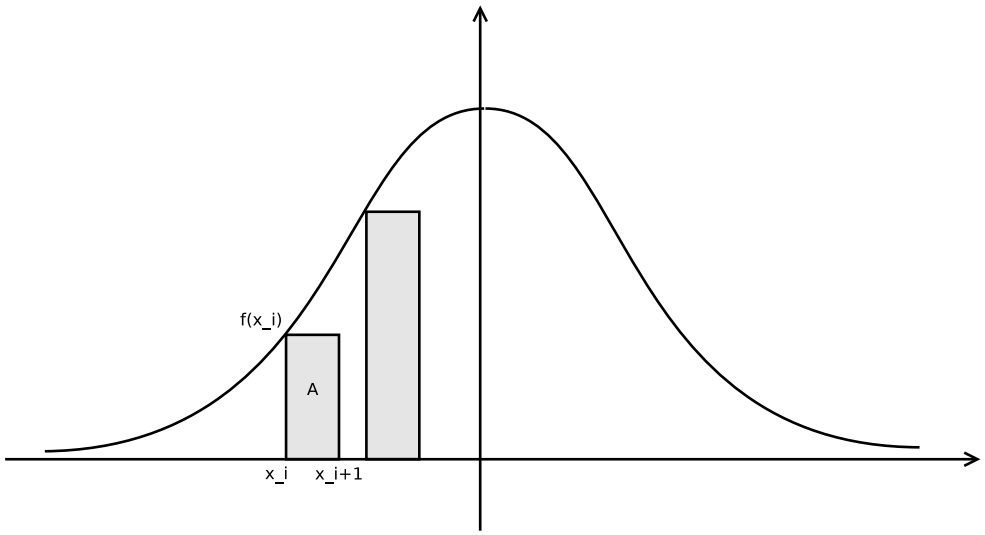


Figura 4.4: Número de amostras geradas em cada intervalo

As amostras geradas são as entradas do algoritmo *Lloyd-Max*[13][27], que então fornece os quantizadores usados para cada coeficiente. O esquema de quantização obtido é utilizado no LFP. Os resultados alcançados utilizando o esquema de quantizadores treinados são exibidos para as sequências *Balloons* e *Kendo* nas Figuras 4.5 e 4.6. Esses resultados exibem apenas o efeito da quantização na qualidade das vistas virtuais sintetizadas e apenas funções lineares foram utilizadas para a aproximação do resíduo. Os resultados para as sequências *Newspaper*, *PoznanHall2* e *PoznanStreet* podem ser consultados no Apêndice C.1.2. A taxa em *bits* por *pixel* (*bpp*) nos resultados experimentais deste capítulo é calculada somando as taxas gastas na codificação dos mapas.

Nas curvas *lfp*, os mapas foram codificados com o LFP original. As curvas *convex hull* são as cascas convexas dos quantizadores com a quantidade de níveis de reconstrução variando de 5 a 81, onde o melhor quantizador para aquela imagem é escolhido para cada taxa. O esquema de quantização empregado tem implicação direta na qualidade das vistas sintetizadas como pode ser observado. Para taxas

altas, o desempenho geralmente é superior com quantizadores com mais níveis de recontrução, com ganho significativo de desempenho. Para taxas baixas, o comportamento é muito parecido entre os quantizadores.

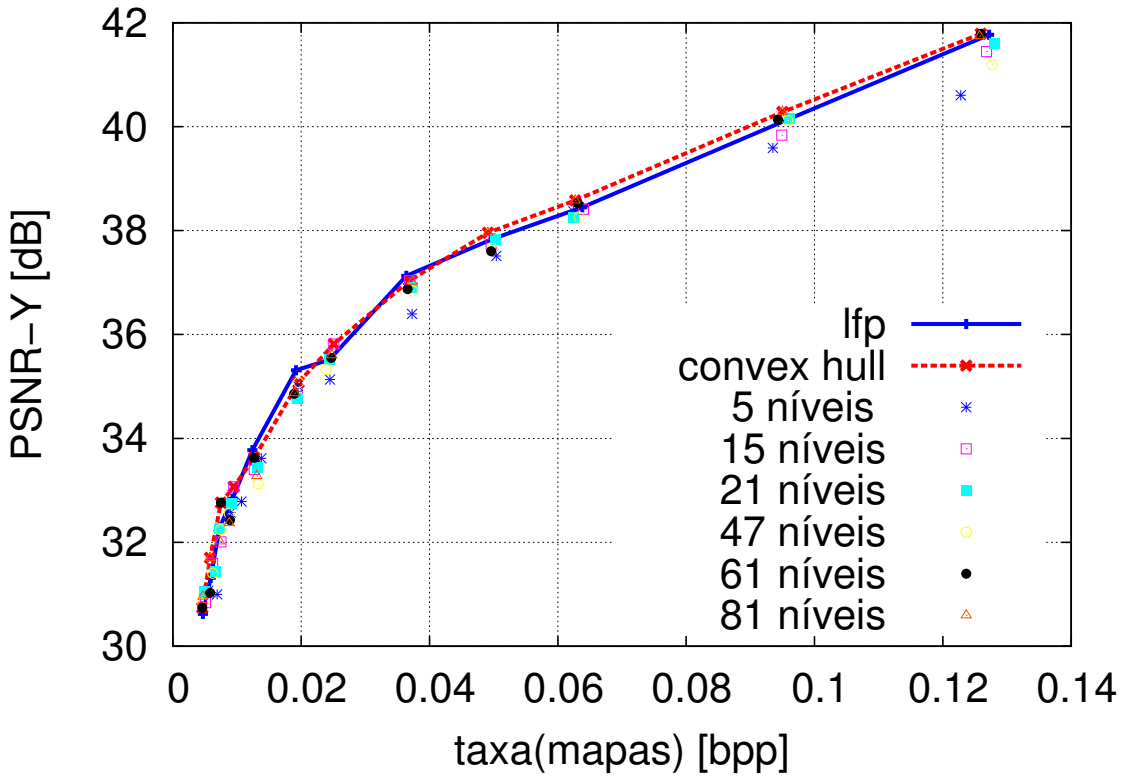


Figura 4.5: Efeito da quantização dos coeficientes de aproximação na qualidade das vistas reconstruídas usando os mapas codificados (*Balloons* câmera 3)

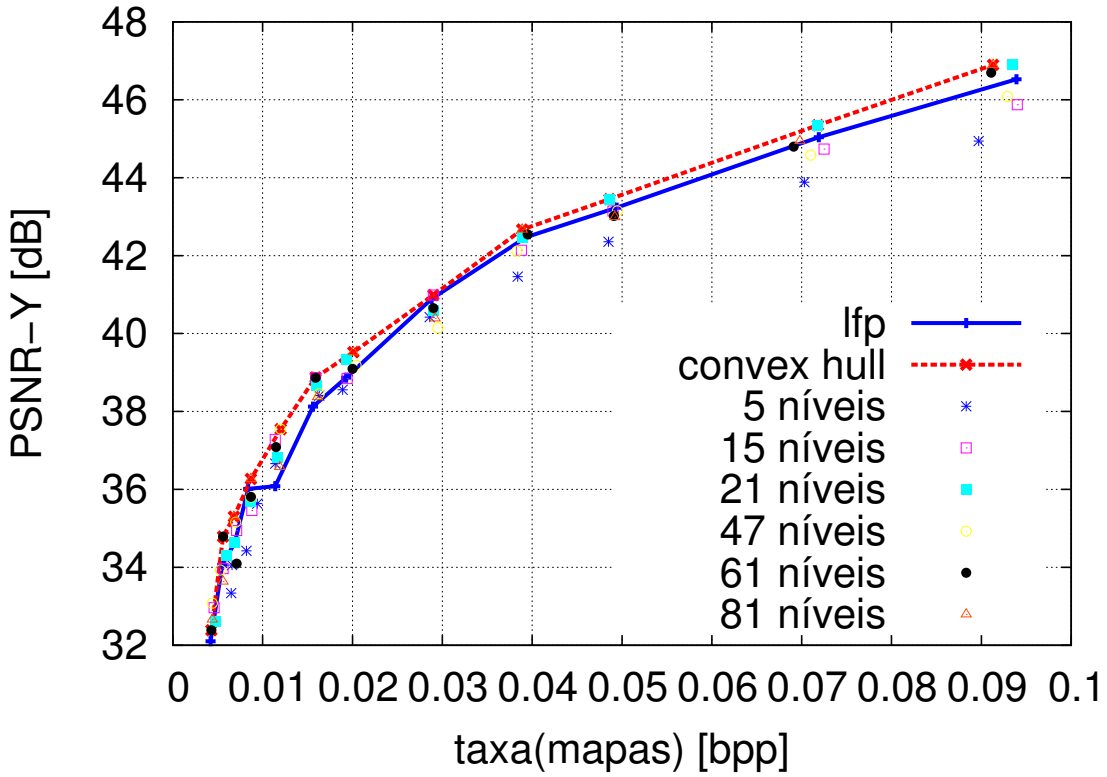


Figura 4.6: Efeito da quantização dos coeficientes de aproximação na qualidade das vistas reconstruídas usando os mapas codificados (*Kendo* câmera 3)

4.4 Combinação do esquema de quantizadores treinados e aproximação do resíduo com diferentes funções

Diversos experimentos foram realizados combinando o esquema de quantização e a aproximação do resíduo usando funções constantes e quadráticas. Além dos quantizadores treinados empregando apenas funções lineares para a aproximação do resíduo. A mesma estratégia foi seguida utilizando apenas funções constantes e também quadráticas. É importante notar que foi obtido o quantizador ótimo de cada coeficiente das funções constante, linear e quadrática. A quantidade de níveis de reconstrução também variou de 5 a 81. Os níveis de reconstrução estão no Apêndice A.3.

Os resultados experimentais para as sequências *Balloons* e *Kendo*, mostrados nas Figuras 4.7 e 4.8, exibem as cascas convexas dos diferentes esquemas de quantização testados. Note que nesse experimento foi empregada a escolha ótima, segundo o custo lagrangeano, empregando combinações de funções constantes (*cons*), lineares (*lin*) e quadráticas (*quad*) para a aproximação do resíduo. Por exemplo, na

curva denominada *cons-quad* a escolha se deu apenas entre funções constantes e quadráticas.

Por inspeção visual, podemos verificar que o uso combinado da quantização ótima e da escolha ótima da ordem de aproximação é vantajoso em relação ao uso de cada uma individualmente. As curvas com o rótulo *lfp* foram obtidas com o algoritmo LFP original.

Os resultados para outras sequências estão no Apêndice C.1.3

Resumo da configuração utilizada. Esquema de quantizadores treinados e aproximação do resíduo de predição com funções constantes, lineares e quadráticas

- Aproximação do resíduo de predição utilizando combinações das opções de funções constantes, lineares e quadráticas.
- Esquema de quantização com quantidades diferentes de níveis de reconstrução. Cada coeficiente das funções de aproximação tendo seu próprio esquema de quantização. Por exemplo, os coeficientes α_0, α_1 e α_2 da função de aproximação linear têm esquemas de quantização com quantidades de níveis de reconstrução variando de 5 a 81.

  Apenas as cascas convexas das curvas empregando diferentes opções de funções de aproximação do resíduo estão exibidas.

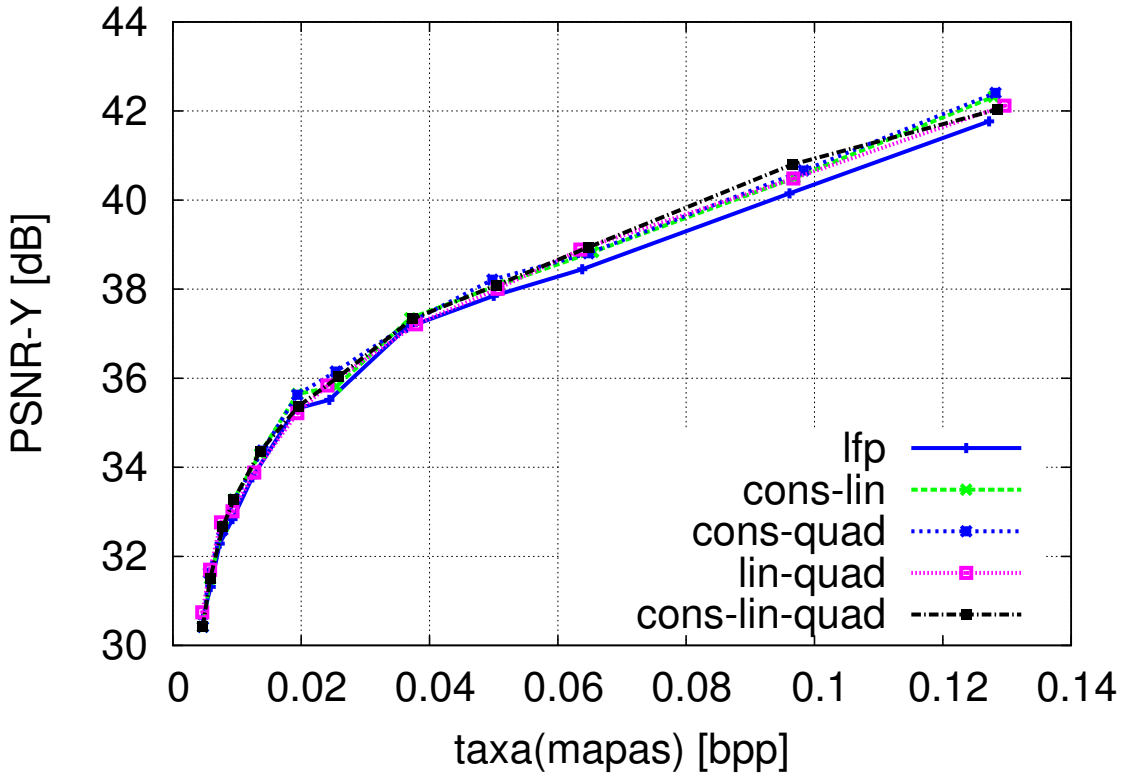


Figura 4.7: Combinando os esquemas: quantização ótima e escolha ótima das funções de aproximações (*Balloons* câmera 3)

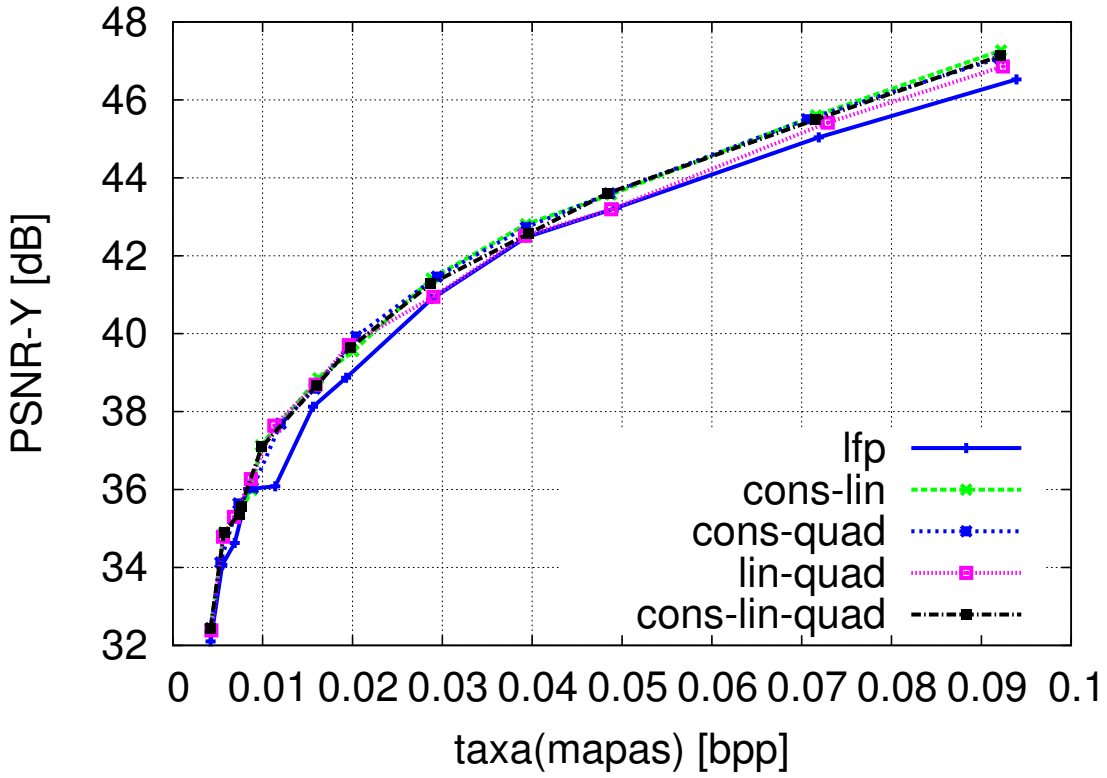


Figura 4.8: Combinando os esquemas: quantização ótima e escolha ótima das funções de aproximações (*Kendo* câmera 3)

4.5 Resultados experimentais

Baseado nos resultados obtidos empregando o esquema de quantização e a aproximação do resíduo de predição, definimos na equação 4.9 a heurística para a escolha do quantizador em função do multiplicador de Lagrange (equação 3.6), cujos resultados se aproximam da casca convexa dos resultados obtidos com os quantizadores testados como:

$$Q = \begin{cases} \text{Quant. com 5 níveis,} & \text{se} \quad \lambda > 1000, \\ \text{Quant. com 15 níveis,} & \text{se} \quad 1000 \geq \lambda > 299, \\ \text{Quant. com 21 níveis,} & \text{se} \quad 299 \geq \lambda > 75 \quad , \\ \text{Quant. com 47 níveis,} & \text{se} \quad 75 \geq \lambda > 49 \quad , \\ \text{Quant. com 61 níveis,} & \text{se} \quad 49 \geq \lambda > 14 \quad , \\ \text{Quant. com 81 níveis,} & \text{se} \quad \lambda \leq 14 \quad , \end{cases} \tag{4.9}$$

A heurística para a escolha do quantizador definida na equação anterior foi escolhida dessa forma pela frequência de escolha de uso do quantizador em cada taxa para as imagens de teste. A escolha do melhor quantizador é feita traçando a casca convexa das curvas dos quantizadores testados para cada imagem. Essa escolha foi feita baseada nos resultados experimentais empregando funções constantes, lineares e quadráticas. Os níveis de reconstrução dos coeficientes das funções de aproximação estão no Apêndice A.3.

O esquema de aproximação do resíduo que emprega a escolha ótima (em termos do custo lagrangeano) entre funções constantes, lineares e quadráticas foi adotado nos resultados exibidos nesta seção, pois verificamos que exibe resultado igual ou superior aos demais testados.

A proposta de usar, além de funções lineares, funções constantes e quadráticas na aproximação do resíduo de predição permite uma melhor relação taxa-distorção, isso se deve à flexibilidade do algoritmo de escolher a aproximação com menor custo lagrangeano. Além disso, o uso de quantizadores treinados propicia uma melhor alocação de *bits*.

Os resultados são exibidos para as sequências *Balloons* (síntese da câmera 3), *Kendo* (síntese da câmera 3), *Newspaper* (síntese da câmera 4), *PoznanHall2* (síntese da câmera 6) e *PoznanStreet* (síntese da câmera 4). Apenas o quadro 0 de cada sequência foi sintetizado.

Os resultados foram comparados com os resultados obtidos com o *software* de referência do padrão H264[4] da ITU-T[1], com o *software* de referência do novo padrão de codificação de vídeos HEVC[6], bem como sua extensão 3D, o HEVC-3D[28]. No HEVC-3D a codificação das imagens de textura e dos mapas de profundidade

é realizada conjuntamente. Além disso, a codificação é conduzida em função da qualidade da vista sintetizada, sendo essa técnica conhecida como *View Synthesis Optimization* (VSO) [28]. Esse motivo torna a comparação direta com os resultados obtidos com o HEVC-3D injusta. No entanto, o HEVC-3D permite que apenas as imagens de textura sejam codificadas sem o uso do VSO. Nos resultados apresentados nesta seção para o HEVC-3D, em vez de codificar apenas as imagens de textura, os mapas foram codificados. Ou seja, a técnica de codificação de imagens de textura foi aplicada em mapas. Isso implica que a codificação não se favorece de técnicas de predição específicas para mapas. As imagens virtuais foram obtidas usando os mapas codificados com HEVC-3D e as imagens de textura originais.

No H264 o processo de codificação é realizado usando blocos de 16×16 *pixels*, chamados de *Macroblocks* (MB). Já o processo de codificação no LFP é realizado usando blocos a partir de 32×32 *pixels*. No HEVC (e no HEVC-3D) a unidade de processamento é chamada de *Coding Tree Unit* (CTU), cujo tamanho é selecionado pelo codificador. Cada CTU consiste de um *Coding Tree Block* (CTB) de amostras *luma* e dois CTBs de amostras *chroma*. Nesta dissertação, apenas as amostras *luma* são codificadas, pois as imagens utilizadas são em tons de cinza. O tamanho do CTB de amostras *luma* pode ser escolhido entre 16, 32 ou 64. O HEVC pode segmentar cada CTB em blocos menores chamados de *Coding Block* (CB), com dimensão mínima de 8×8 *pixels*. O CB de amostras *luma* juntamente com os dois CBs de amostras *chroma* formam um *Coding Unit* (CU). O tamanho máximo do CU pode ser passado como parâmetro para o codificador. Os experimentos para o HEVC e para o HEVC-3D foram conduzidos usando tanto o tamanho máximo de 32 quanto de 64 para o CU. Assim avaliamos quanto o tamanho de bloco implica no desempenho do algoritmo.

Os resultados exibidos para as sequências testadas mostram a competitividade do algoritmo LFP em relação aos métodos estado da arte. Os resultados para as sequências *Ballons*, *Kendo*, *Newspaper*, *PoznanHall2* e *PoznanStreet* estão nas Figuras 4.9, 4.12, 4.15, 4.18 e 4.22, respectivamente.

Nas Figuras 4.10, 4.13, 4.16, 4.19, 4.20, 4.23 e 4.24 são exibidos os mapas originais e os mapas codificados das vistas das câmeras usadas na síntese das vistas virtuais. Enquanto que nas Figuras 4.11, 4.14, 4.17, 4.21 e 4.25 são exibidas as vistas virtuais sintetizadas tanto com os mapas e as texturas, ambos originais, quanto com os mapas codificados e as texturas originais. Uma taxa baixa foi escolhida para permitir visualizar os artefatos nas vistas virtuais causados pela codificação dos mapas.

[1] *International Telecommunication Union*

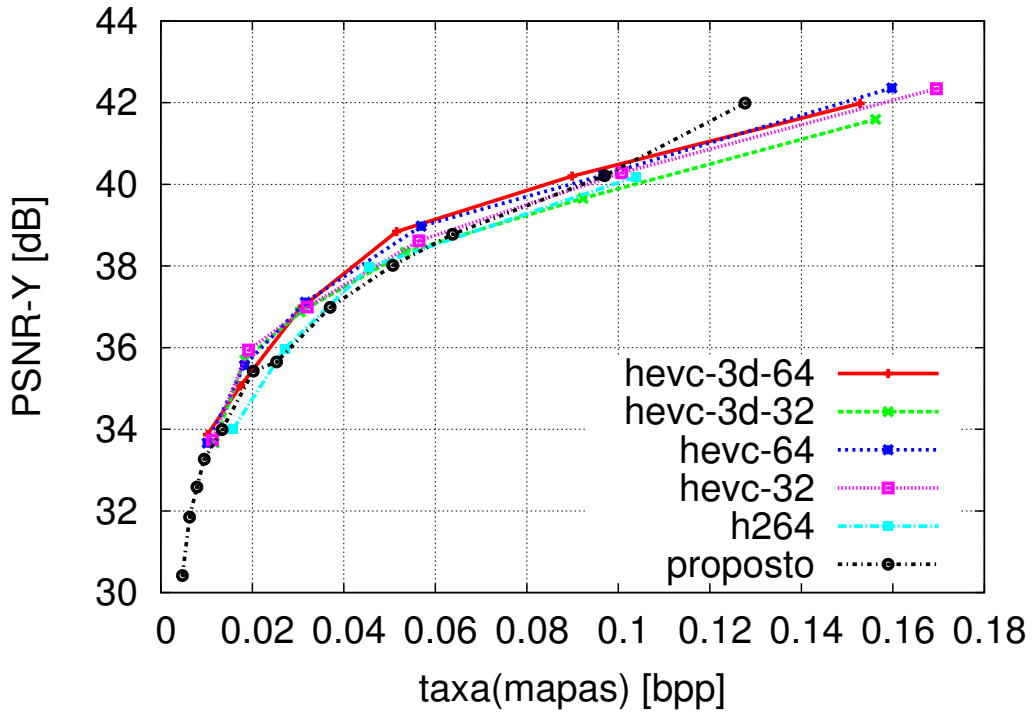
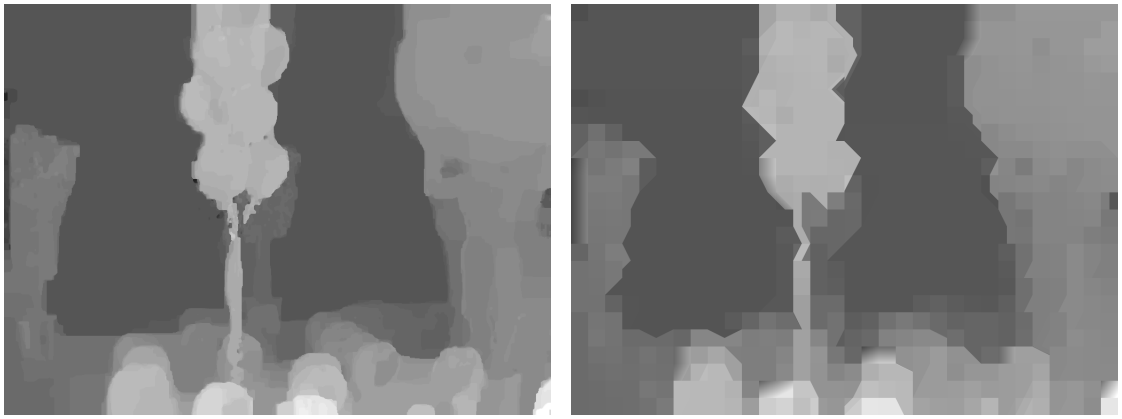
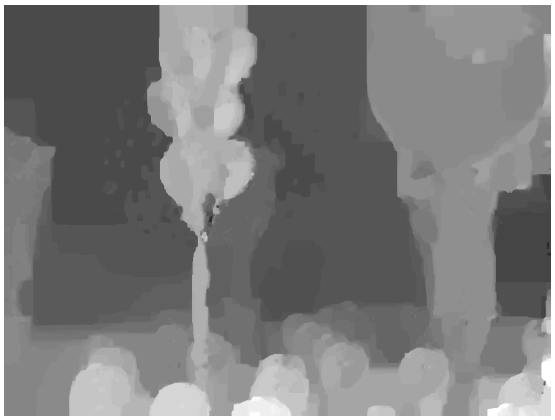


Figura 4.9: Resultado taxa-distorção do LFP usando a heurística da equação 4.9 (*Balloons* câmera 3)

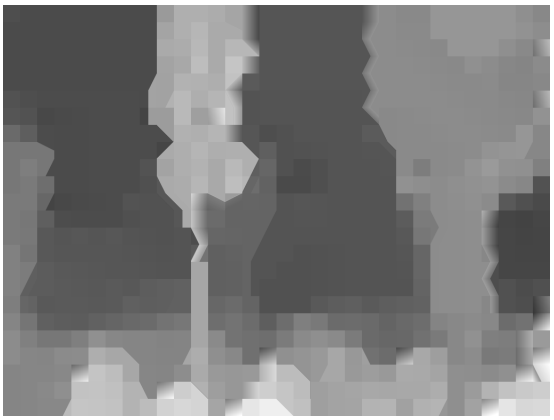


(a) Mapa original da vista da câmera 1

(b) Mapa codificado da vista da câmera 1



(c) Mapa original da vista da câmera 5



(d) Mapa codificado da vista da câmera 5

Figura 4.10: *Balloons* — Mapas originais e mapas reconstruídos codificados com o LFP

(a) Imagens de textura e mapas, ambos originais

(b) Imagens de textura originais e mapas codificados. Taxa = 0,0063 bpp PNSR = 31,8481 dB

Figura 4.11: *Balloons* — Síntese da vista virtual da câmera 3

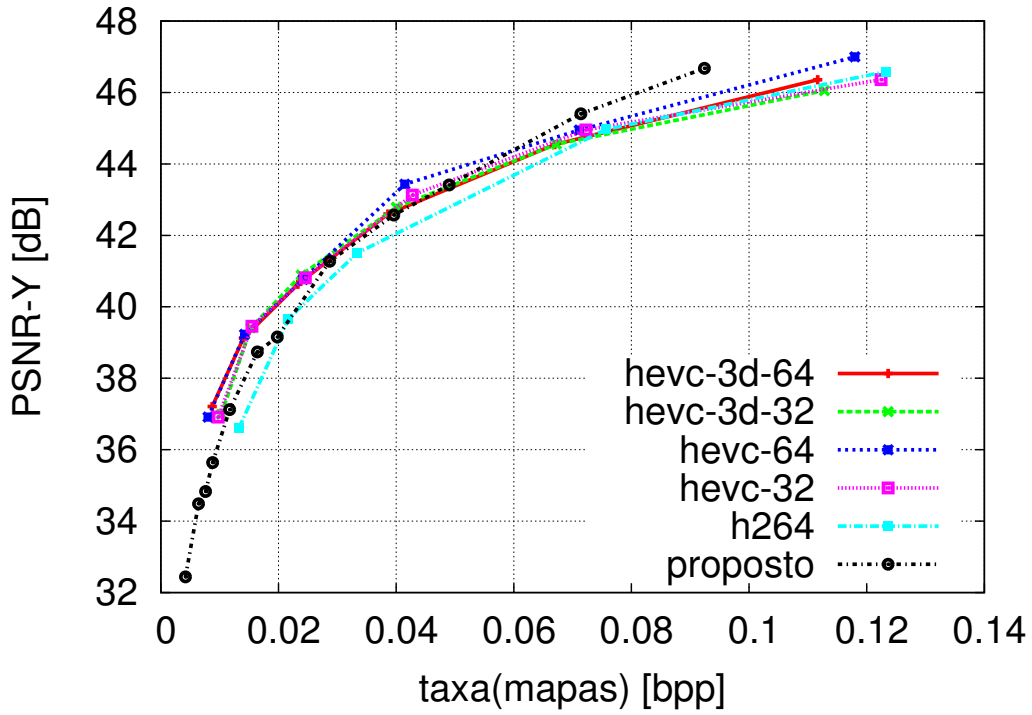
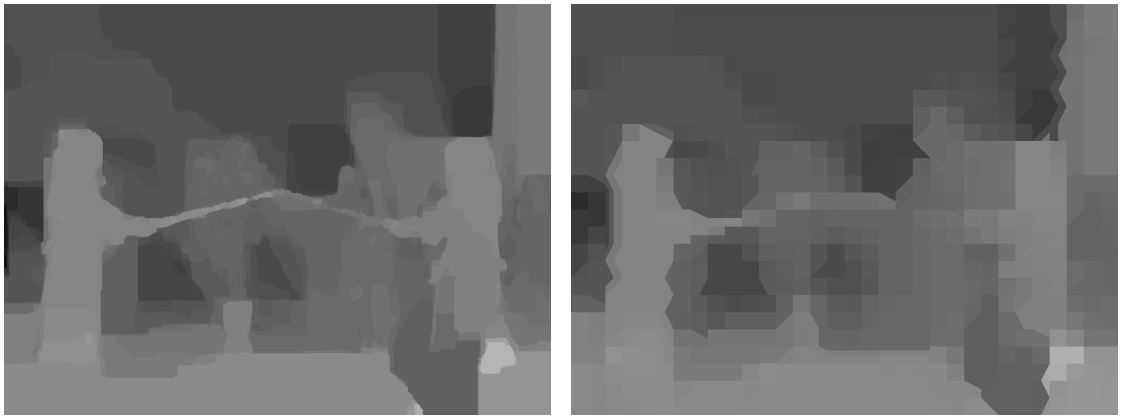
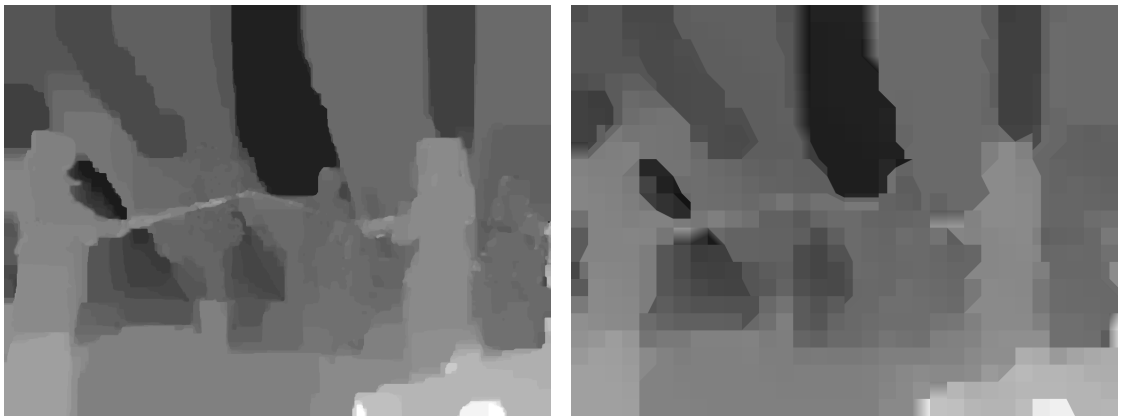


Figura 4.12: Resultado taxa-distorção do LFP usando a heurística da equação 4.9 (*Kendo* câmera 3)



(a) Mapa original da vista da câmera 1

(b) Mapa codificado da vista da câmera 1



(c) Mapa original da vista da câmera 5

(d) Mapa codificado da vista da câmera 5

Figura 4.13: *Kendo* — Mapas originais e mapas reconstruídos codificados com o LFP

(a) Imagens de textura e mapas, ambos originais

(b) Imagens de textura originais e mapas codificados. Taxa = $0,0064$ bpp PNSR = $34,4844$ dB

Figura 4.14: *Kendo* — Síntese da vista virtual da câmera 3

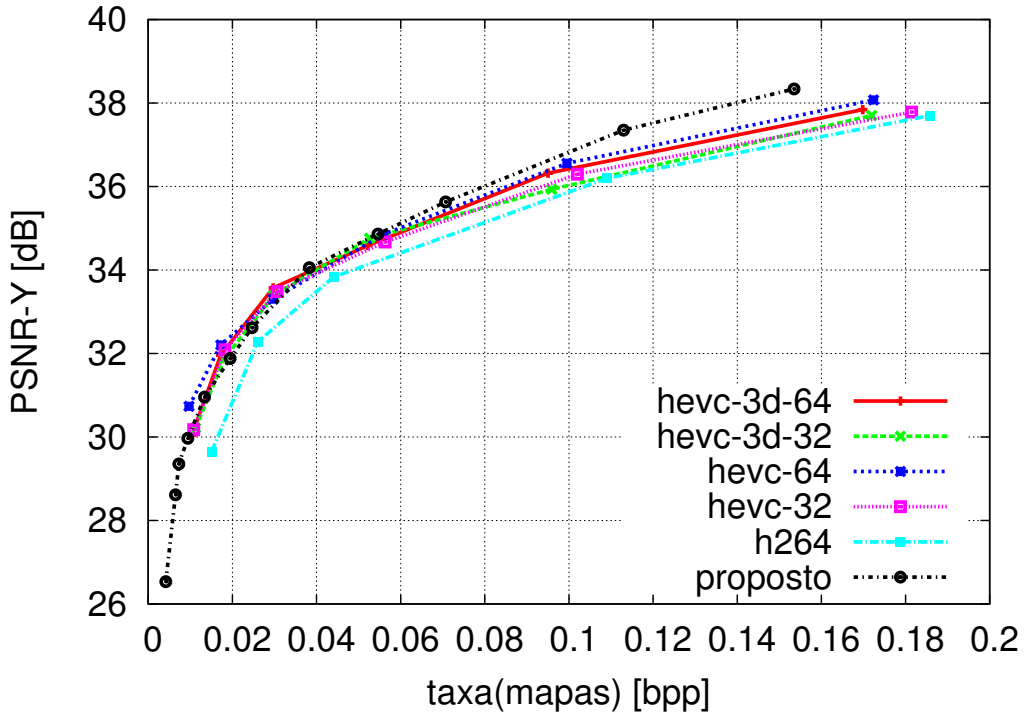
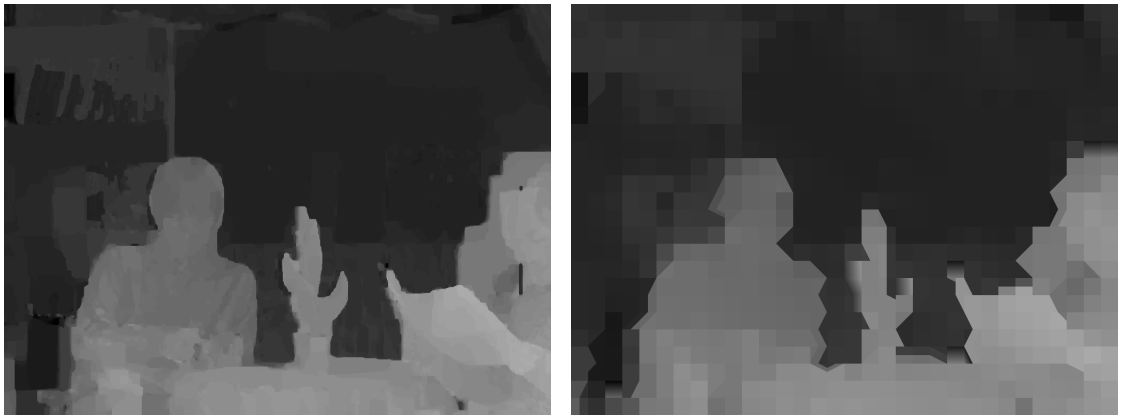


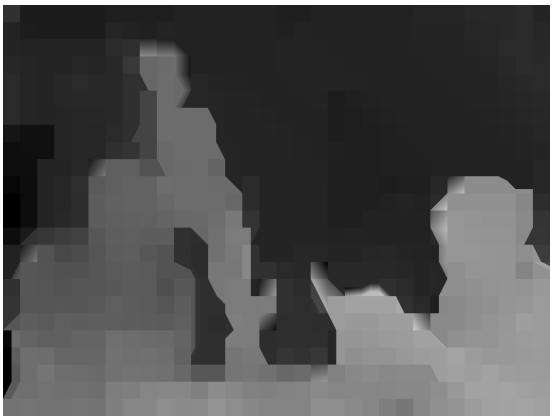
Figura 4.15: Resultado taxa-distorção do LFP usando a heurística da equação 4.9 (*Newspaper* câmera 4)



(a) Mapa original da vista da câmera 2

(b) Mapa codificado da vista da câmera 2

(c) Mapa original da vista da câmera 6



(d) Mapa codificado da vista da câmera 6

Figura 4.16: *Newspaper* — Mapas originais e mapas reconstruídos codificados com o LFP

(a) Imagens de textura e mapas, ambos originais

(b) Imagens de textura originais e mapas codificados. Taxa = $0,0065$ bpp PNSR = $28,6143$ dB

Figura 4.17: *Newspaper* — Síntese da vista virtual da câmera 4

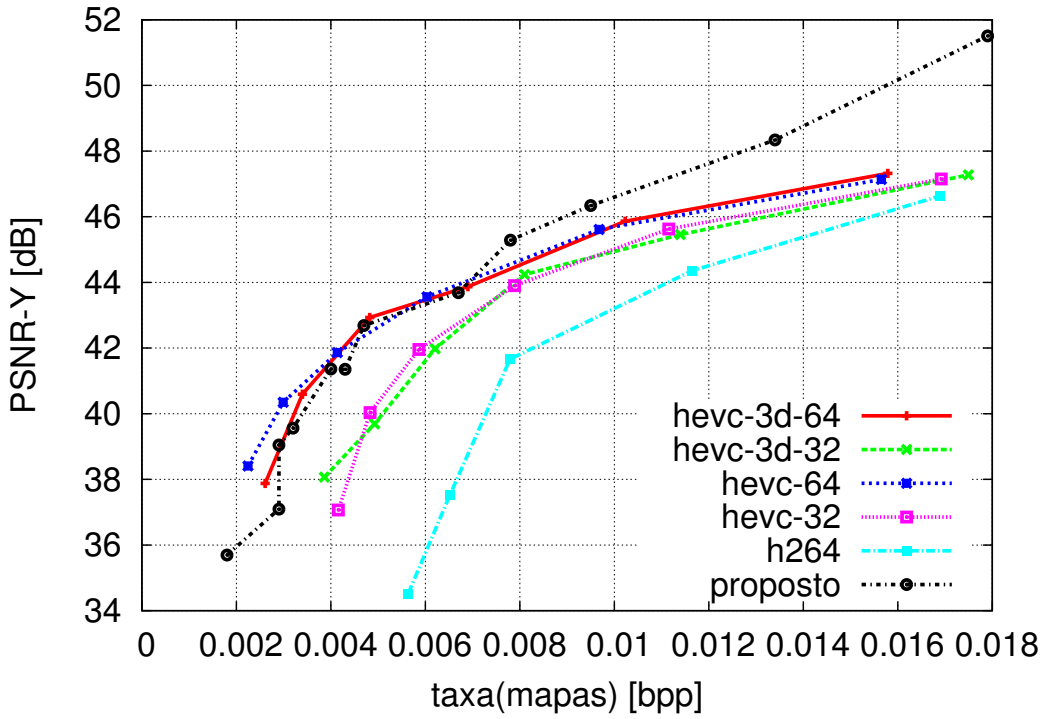


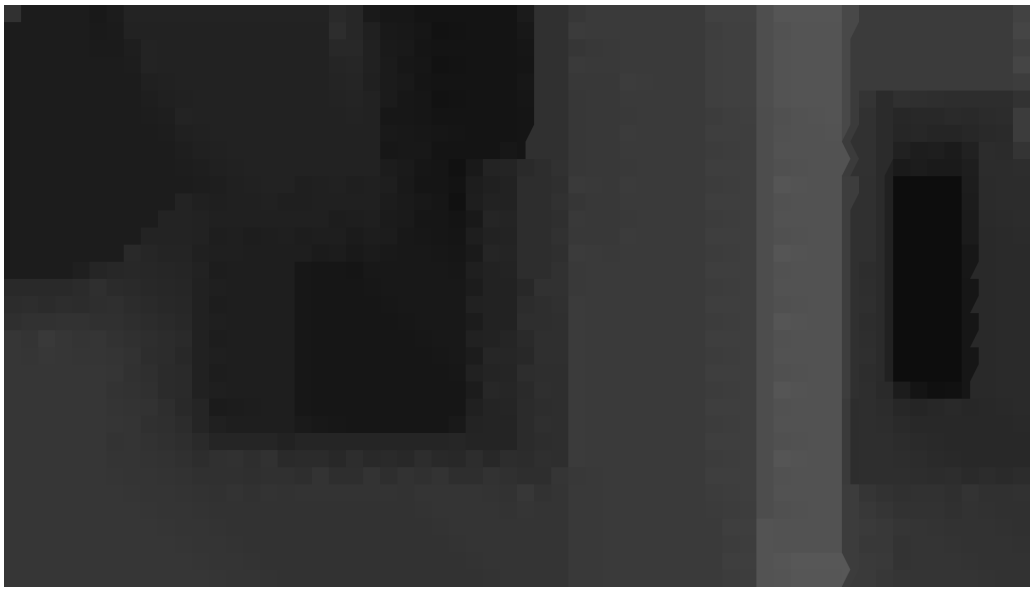
Figura 4.18: Resultado taxa-distorção do LFP usando a heurística da equação 4.9 (*PoznanHall2* câmera 6)

(a) Mapa original da vista da câmera 5

(b) Mapa codificado da vista da câmera 5

Figura 4.19: *PoznanHall2* — Mapas originais e mapas reconstruídos codificados com o LFP

(a) Mapa original da vista da câmera 7



(b) Mapa codificado da vista da câmera 7

Figura 4.20: *PoznanHall2* — Mapas originais e mapas reconstruídos codificados com o LFP

(a) Imagens de textura e mapas, ambos originais

(b) Imagens de textura originais e mapas codificados. Taxa = $0,0029$ bpp PNSR = $37,0934$ dB

Figura 4.21: *PoznanHall2* — Síntese da vista virtual da câmera 6

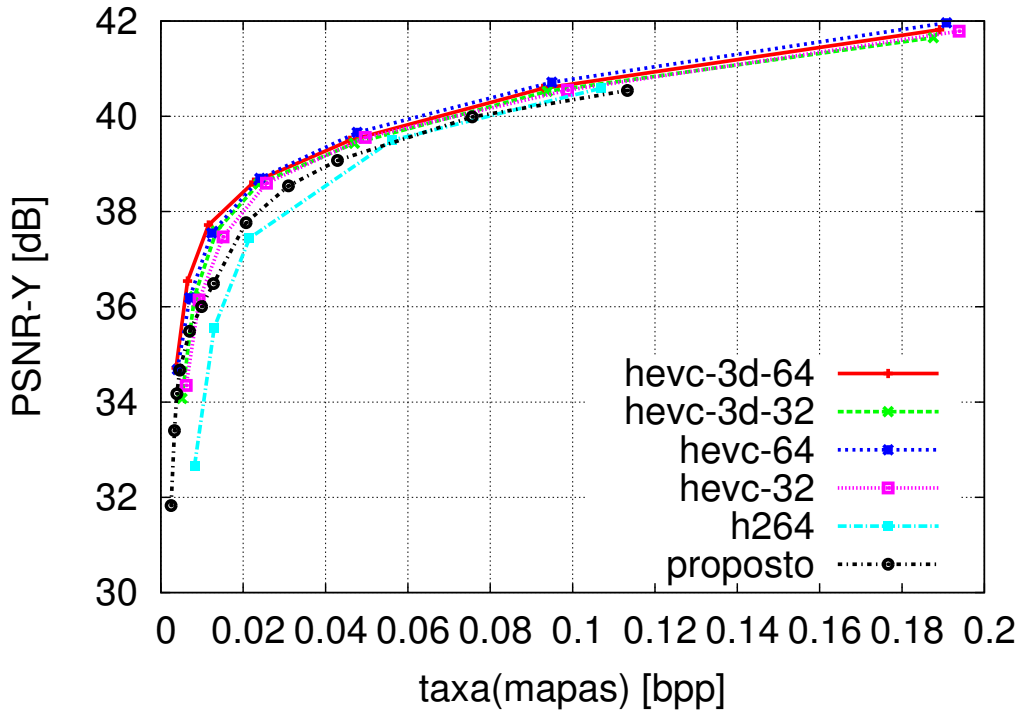


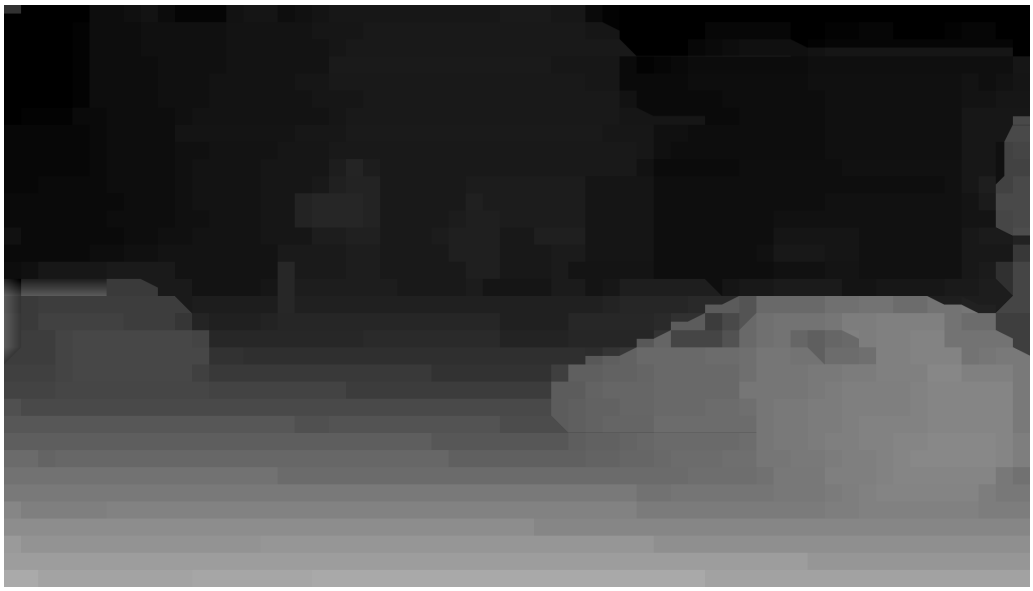
Figura 4.22: Resultado taxa-distorção do LFP usando a heurística da equação 4.9 (*PoznanStreet* câmera 4)

(a) Mapa original da vista da câmera 3

(b) Mapa codificado da vista da câmera 3

Figura 4.23: *PoznanStreet* — Mapas originais e mapas reconstruídos codificados com o LFP

(a) Mapa original da vista da câmera 5



(b) Mapa codificado da vista da câmera 5

Figura 4.24: *PoznanStreet* — Mapas originais e mapas reconstruídos codificados com o LFP

(a) Imagens de textura e mapas, ambos originais

(b) Imagens de textura originais e mapas codificados. Taxa = $0,0033$ bpp PNSR = $33,4021$ dB

Figura 4.25: *PoznanStreet* — Síntese da vista virtual da câmera 4

Capítulo 5

Aplicação do LFP na codificação de imagens de textura

Neste capítulo iremos adaptar o algoritmo LFP para ser utilizado na codificação de imagens de textura. As técnicas utilizadas, assim como os resultados alcançados são apresentados. Os resultados alcançados com o algoritmo são comparados com os resultados exibidos por métodos estado da arte.

5.1 Codificação de imagens de textura

O algoritmo LFP foi originalmente proposto para a codificação de mapas de profundidade. Assim sendo, considerando a aplicação do algoritmo na codificação de imagens de textura, duas alterações devem ser realizadas inicialmente. A soma do erro absoluto — na literatura chamada de *Sum of Absolute Differences* (SAD) — adotada como critério de distorção na otimização da árvore de segmentação objetiva preservar a informação de profundidade presente em mapas, além de experimentalmente ter desempenho superior [7].

No entanto, a soma do erro quadrático — na literatura chamada de *Sum of Squared Differences* (SSD) — é usualmente utilizada em imagens de textura, pois objetiva a qualidade visual percebida pelo sistema visual[29][7]. Para a codificação dessas, adotamos a soma do erro quadrático como critério de distorção utilizado na otimização da árvore de segmentação. Enfatizamos que para a escolha do modo de predição é utilizado aquele que fornece o resíduo com menor norma L1. Na otimização taxa-distorção da árvore de segmentação é utilizada como critério de distorção a soma do erro quadrático.

Portanto, a árvore $\mathcal{T}$ é otimizada minimizando a função custo dada pela equação 3.6, usando a soma do erro quadrático como critério de distorção.

Além disso, como os mapas de profundidade não são diretamente visualizados,

o filtro de atenuação de artefatos introduzidos pelos modos direcionais de predição não foi utilizado. Na codificação de imagens de textura, a filtragem das amostras nas fronteiras dos blocos contribui para uma melhor predição favorecendo o desempenho do algoritmo [4][21]. Portanto, usamos esse filtro passa-baixas para obter uma melhor predição. Inspirado nos resultados apresentados na etapa de predição de [21], os coeficientes do filtro utilizado são:

$$h = \left\{ \frac{1}{4}, \frac{2}{4}, \frac{1}{4} \right\} \tag{5.1}$$

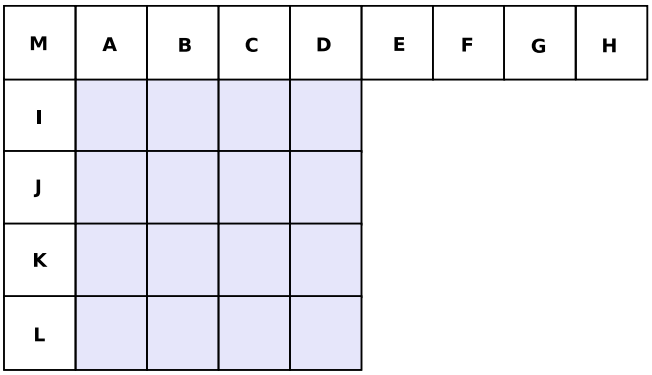


Figura 5.1: Filtragem dos *pixels* usados para a predição

Os *pixels* utilizados como referência para a predição são indicados pelas letras na Figura 5.1. A filtragem é aplicado tanto no bloco horizontal (*pixels* A,...,H) quanto no bloco vertical (*pixels* I,...,J). Por exemplo, aplicando a filtragem, o valor do *pixel* C ficaria $C_{filt} = \frac{1}{4}B + \frac{2}{4}C + \frac{1}{4}D$. O *pixel* da borda, A, ficaria $A_{filt} = \frac{1}{4}A + \frac{2}{4}A + \frac{1}{4}B$, enquanto que o *pixel* H ficaria $H_{filt} = \frac{1}{4}G + \frac{2}{4}H + \frac{1}{4}H$. Da mesma forma é realizada a filtragem dos demais *pixels*.

O ganho de desempenho do algoritmo LFP na codificação de imagens de textura é significativo adotando a modificação do critério de distorção e o uso da filtragem dos *pixels* utilizados como referência para a predição. Pode-se verificar nas Figuras 5.2, 5.3, 5.4, para as imagens *lena*, *D108* e *pp1205*, respectivamente, a contribuição para o desempenho de cada modificação.

Nas Figuras desta seção, as curvas com o rótulo *filtro* foram obtidas empregando a filtragem dos *pixels* utilizados para a predição. As curvas com o rótulo *erro-quad* foram obtidas empregando como critério de distorção a soma do erro quadrático. As curvas *lfp-tex* foram obtidas com a implementação tanto do critério de distorção quanto da filtragem dos *pixels* de referência.

O uso da soma do erro quadrático como critério de distorção e a filtragem aplicada nos *pixels* das fronteiras entre blocos — usados como referência para a predição — colaboram para um melhor desempenho do algoritmo mesmo em imagens com

características diferentes. Ou seja, tanto em imagens suaves (naturais) quanto em imagens não suaves o desempenho é superior. Consideramos, por exemplo, a imagem *lena* como sendo suave e a imagem *pp1205* não suave.

Os resultados alcançados com o LFP com as alterações desta seção serão usados como referência. O desempenho do LFP, combinado com as técnicas investigadas nas seções seguintes, será avaliado em relação aos resultados desta seção. Dessa forma, consideramos a aplicação do LFP na codificação de imagens de textura, com o uso do erro quadrático para medida da distorção e da filtragem. Na codificação de mapas, o algoritmo LFP segmenta a imagem em blocos a partir de 32×32 *pixels*, e isso foi mantido para a codificação de mapas neste trabalho. No entanto, para a codificação de imagens de textura, utilizamos blocos de tamanho 16×16 *pixels* como unidades de processamento. Dessa forma, a comparação do desempenho do algoritmo em relação ao H264/AVC é mais justa.

Os resultados para outras imagens podem ser consultados em C.2.1. Por inspeção visual, podemos verificar um significativo ganho no desempenho do algoritmo. O algoritmo utilizando o erro quadrático para a medida da distorção teve desempenho superior em relação ao algoritmo original para todas as imagens testadas. Enquanto o algoritmo que emprega a filtragem dos *pixels* de referência teve desempenho igual ou superior em relação ao LFP original.

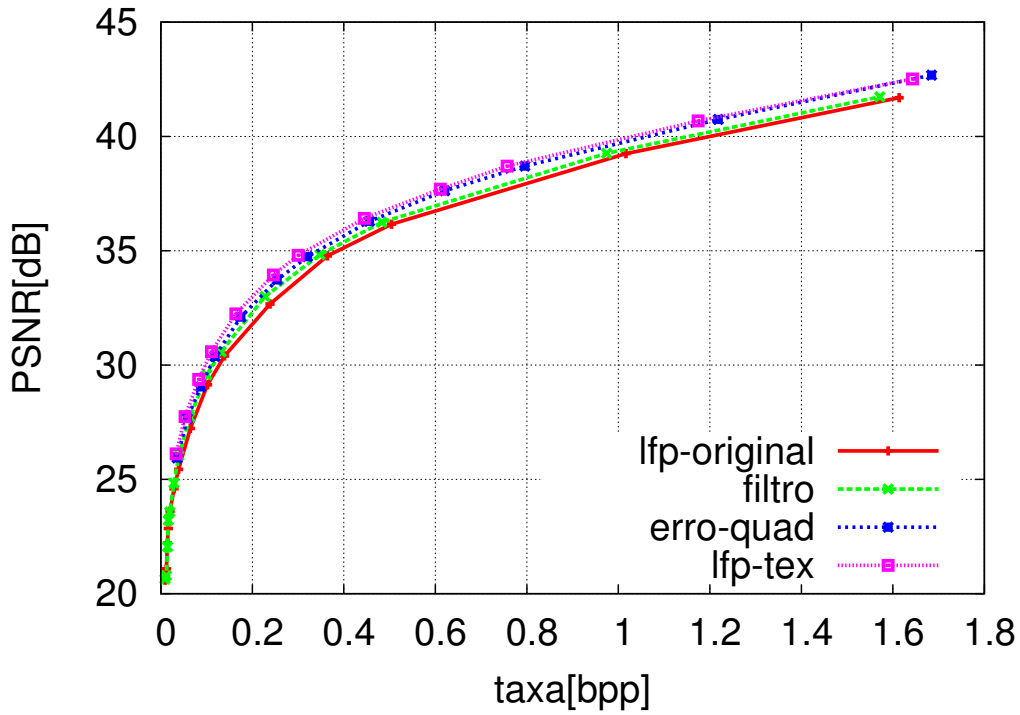


Figura 5.2: Comparação de desempenho taxa-distorção dos algoritmos *lfp* e *lfp-tex*: *lena*

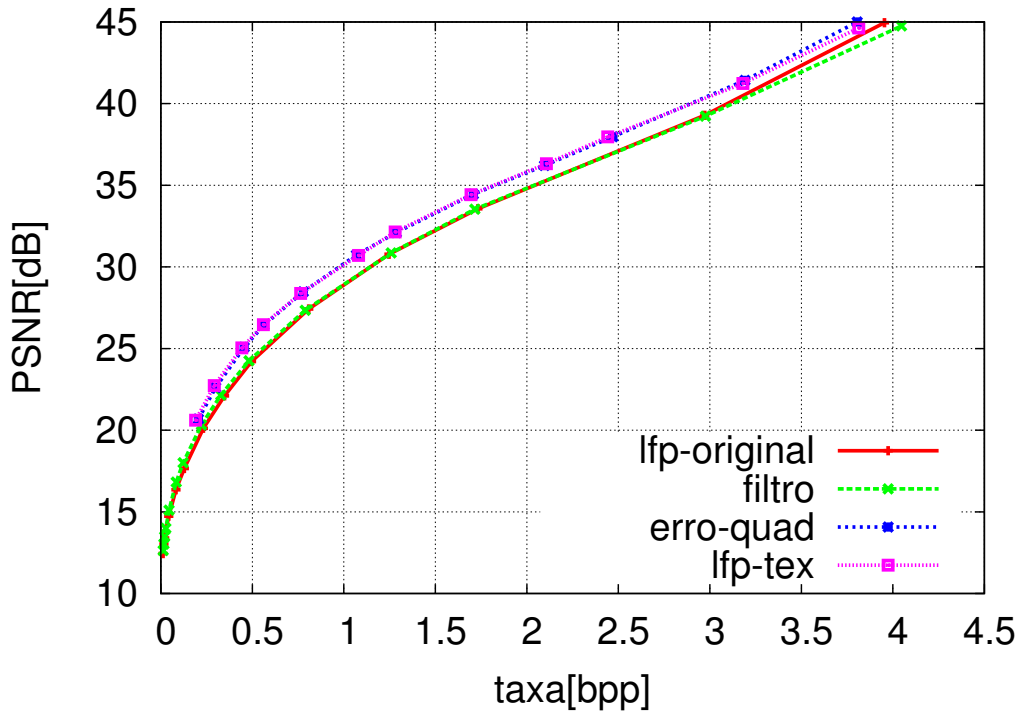


Figura 5.3: Comparação de desempenho taxa-distorção dos algoritmos *lfp* e *lfp-tex*: *D108*

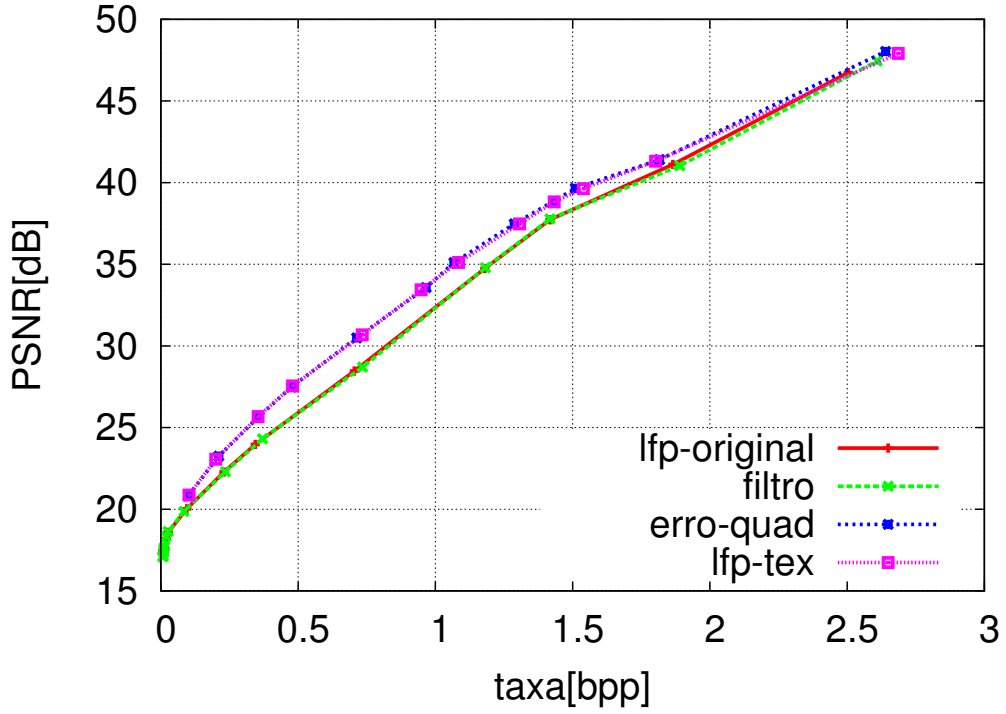


Figura 5.4: Comparação de desempenho taxa-distorção dos algoritmos *lfp* e *lfp-tex*: *pp1205*

5.2 Aproximação do resíduo de predição

Como no caso da codificação de mapas de profundidade (ver seção 4.2), investigamos a aproximação do resíduo de predição com funções polinomiais na codificação de imagens de textura.

Nos resultados apresentados nesta seção, a aproximação do resíduo foi realizada empregando, além de funções lineares, funções constantes e quadráticas, exatamente como no caso da codificação de mapas. Por esse motivo, concentraremos em expor os resultados experimentais. Lembrando que na codificação de textura o critério de distorção utilizado é o erro quadrático e é aplicada a filtragem nos blocos de predição.

Para avaliarmos apenas o impacto de diferentes funções de aproximação do resíduo no desempenho do algoritmo, o esquema de quantização original foi utilizado [7]. Foram testados diversos esquemas de aproximação do bloco de resíduo utilizando as funções constante, linear e quadrática. Os resultados para as imagens *lena*, *D108* e *pp1205* são exibidos nas Figuras 5.5, 5.6 e 5.7, respectivamente. Os resultados para outras imagens podem ser consultados no Apêndice C.2.2. Também foram realizados experimentos utilizando ou apenas funções constantes ou apenas funções quadráticas, porém, individualmente, não apresentaram desempenho superior em relação aos esquemas combinados. Para permitir melhor visualização, os resultados desses esquemas não são mostrados.

Como podemos verificar, o emprego de funções constantes e quadráticas não fornece significativo ganho para o desempenho do algoritmo. Em algumas imagens podemos verificar apenas um pequeno ganho com o esquema que emprega as três funções. Podemos argumentar que o possível ganho de aproximação do resíduo com funções quadráticas é mitigado pela quantidade maior de coeficientes que precisam ser codificados. No caso da função constante, apesar da economia na codificação de coeficientes, a acurácia da aproximação pode ser penalizada. Dessa forma, os diferentes esquemas de aproximação do resíduo fornecem desempenhos semelhantes.

Nos gráficos esboçadas nesta seção, as curvas *lfp-tex* foram obtidas após as mudanças do critério de distorção e da filtragem (ver seção 5.1). As curvas *cons-lin-quad*, conforme explicado anteriormente, foram obtidas empregando funções constantes, lineares e quadráticas, da mesma forma as demais.

Resumo da configuração utilizada nesta seção.

- Aproximação do resíduo de predição utilizando combinações das opções de funções constantes, lineares e quadráticas.
- Filtragem dos *pixels* de referência utilizadas para realizar a predição.
- Soma do erro quadrático como medida da distorção na otimização da árvore de segmentação.
- Esquema de quantização original.

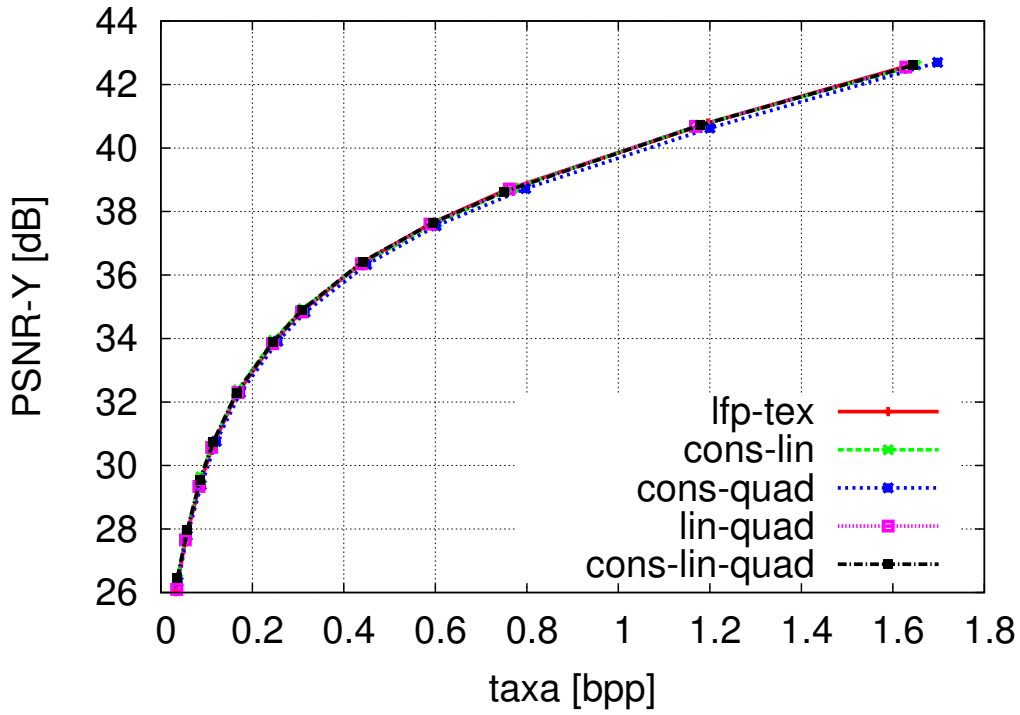


Figura 5.5: Comparação do desempenho taxa-distorção para diferentes funções de aproximação do resíduo: *lena*

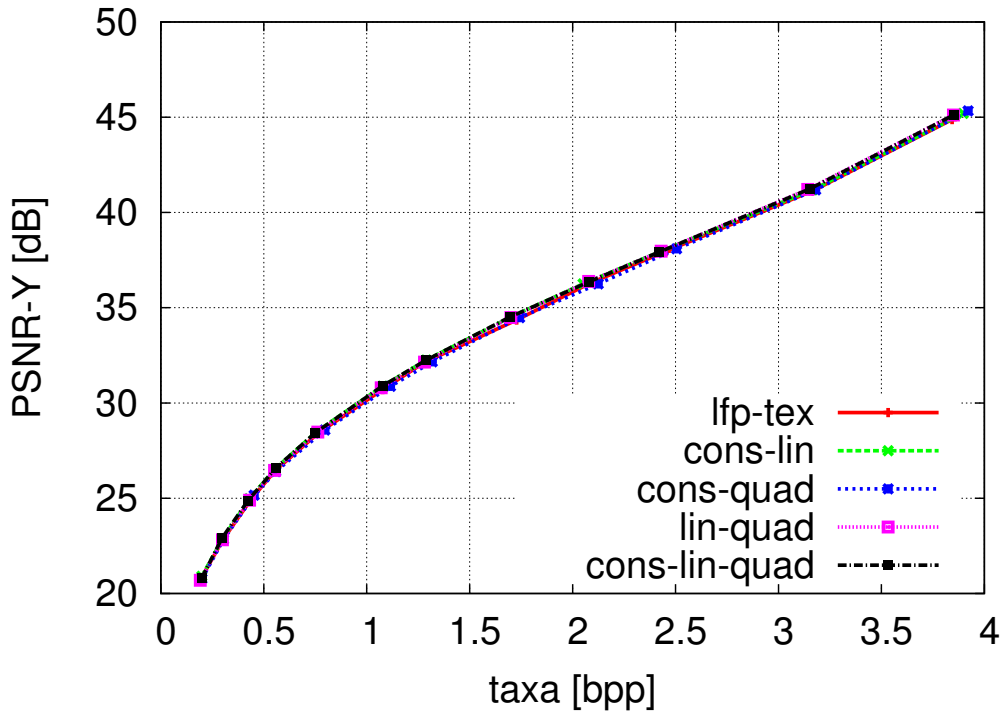


Figura 5.6: Comparação do desempenho taxa-distorção para diferentes funções de aproximação do resíduo: *D108*

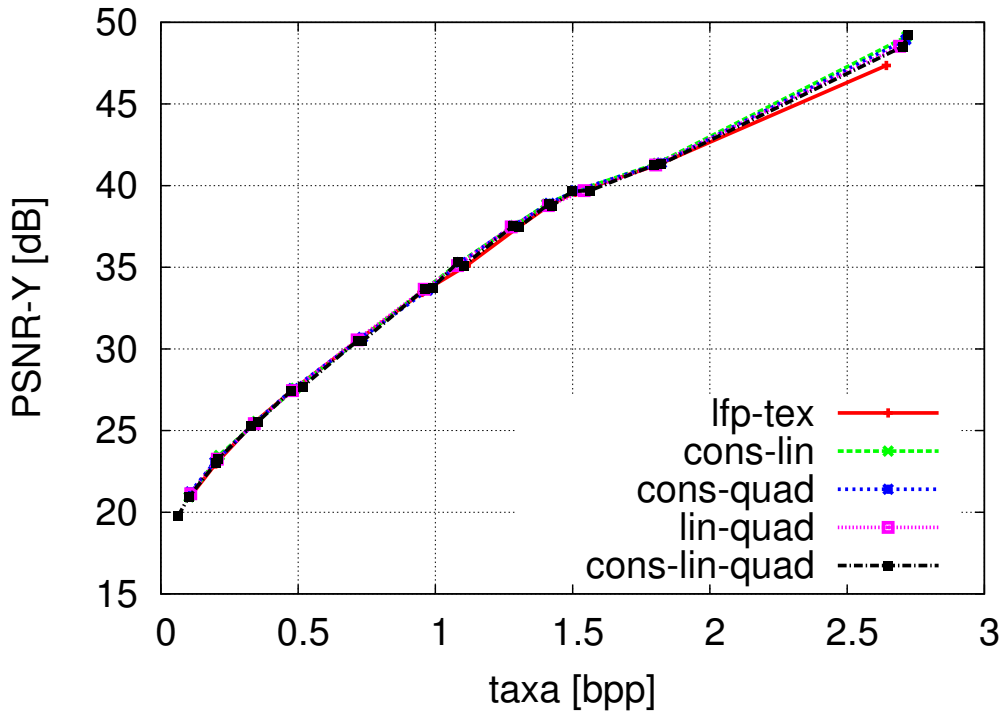


Figura 5.7: Comparação do desempenho taxa-distorção para diferentes funções de aproximação do resíduo: *pp1205*

5.3 Quantização dos coeficientes das funções de aproximação

Nesta seção será avaliado o efeito da quantização dos coeficientes das funções de aproximação dos blocos de resíduo. O mesmo procedimento utilizado para a quantização dos coeficientes na codificação de mapas (ver seção 4.3) foi utilizado em imagens de textura. Ou seja, modelamos a distribuição de cada coeficiente utilizando uma gaussiana generalizada para um conjunto de imagens de treinamento[1]. A partir dos parâmetros das gaussianas generalizadas, foi obtido o quantizador ótimo para cada coeficiente utilizando o algoritmo *Lloyd-Max*[13][27]. Ressaltamos que o conjunto de imagens de treinamento é diferente do conjunto de imagens utilizadas nas simulações. Os níveis de reconstrução dos coeficientes das funções de aproximação estão no Apêndice A.3.

Exibimos nesta seção os resultados utilizando apenas funções lineares na aproximação do resíduo, de modo a avaliar apenas a quantização dos coeficientes no desempenho do algoritmo. Os resultados para as imagens *lena*, *D108* e *pp1205* são exibidos nas Figuras 5.8, 5.9 e 5.10. Os resultados para outras imagens podem ser consultados no Apêndice C.2.3.

Como podemos verificar, diferentemente do que ocorre na codificação de mapas, o algoritmo que emprega o esquema de quantização original do LFP teve melhor desempenho em relação aos esquemas experimentados. Os resultados experimentais mostram que, em geral, o desempenho obtido com quantidades diferentes de níveis de reconstrução se manteve o mesmo ou mesmo piorou em relação ao esquema de quantização original. Em algumas imagens pode-se observar uma ínfima superioridade em baixas taxas, por exemplo, na imagem *lena*. Na imagem *pp1205* em taxas elevadas obteve-se melhor desempenho. No entanto, na imagem *barb* (no Apêndice C.2.3) o esquema de quantização original foi significamente superior. Esses resultados corroboram a importância da quantização dos coeficientes das funções de aproximação, lembrando que o poder de aproximação dos dicionários está relacionado com o esquema de quantização empregado, o que pode explicar a sensibilidade do algoritmo decorrente de uma mudança no esquema de quantização.

Nos gráficos mostrados nesta seção, a curva *lfp-tex* foi obtida após incorporadas as mudanças descritas na seção 5.1. A curva *convex hull* é casca convexa das curvas obtidas com quantidades diferentes de níveis de reconstrução.

[1] As imagens de treinamento utilizadas estão no Apêndice B.

Resumo da configuração utilizada nesta seção.

- Aproximação do resíduo de predição com funções lineares.
- Filtragem dos *pixels* de referência utilizadas para realizar a predição.
- Soma do erro quadrático como medida da distorção na otimização da árvore de segmentação.
- Quantizadores treinados para cada coeficiente da função de aproximação. Quantidades de níveis de reconstrução variando de 5 a 81.

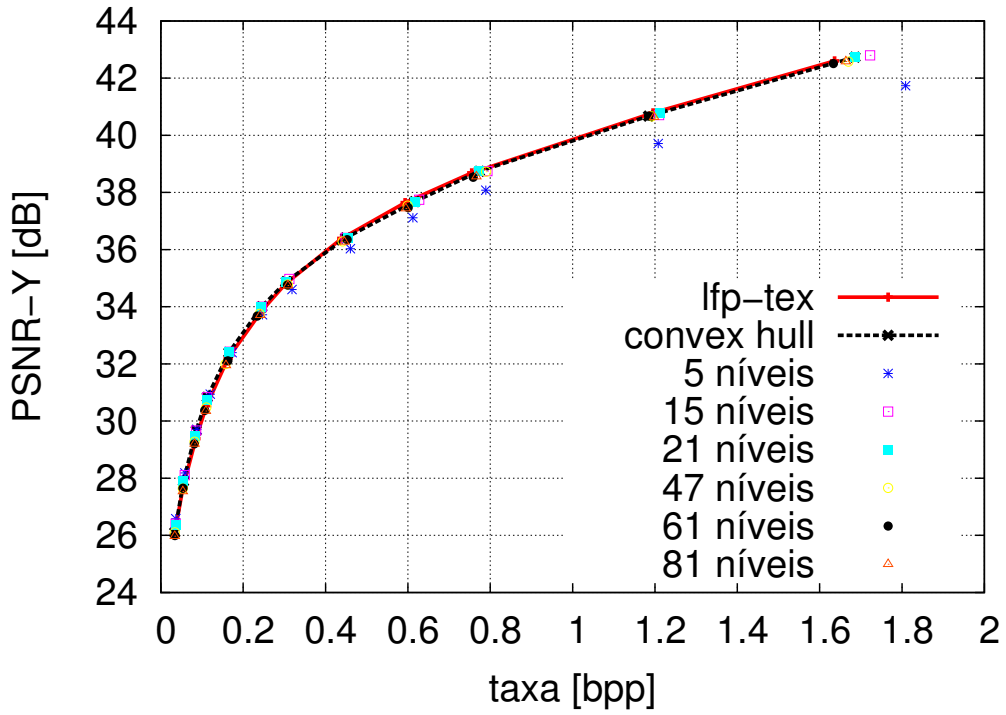


Figura 5.8: Comparação dos resultados taxa-distorção obtidos para diversos quantizadores dos coeficientes: *lena*

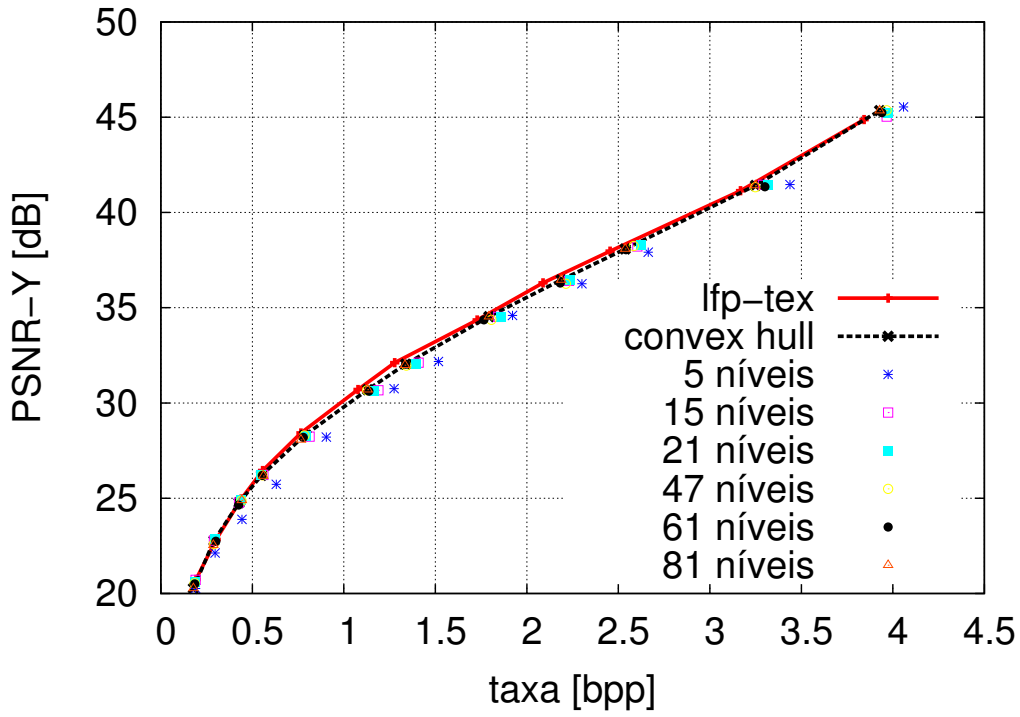


Figura 5.9: Comparação dos resultados taxa-distorção obtidos para diversos quantizadores dos coeficientes: *D108*

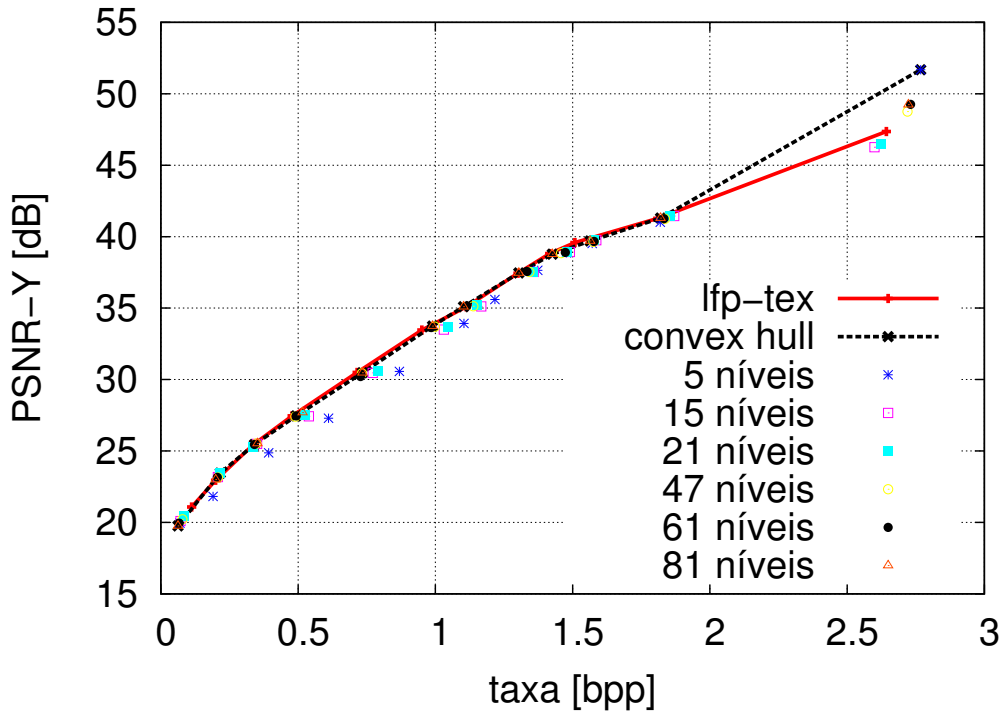


Figura 5.10: Comparação dos resultados taxa-distorção obtidos para diversos quantizadores dos coeficientes: *pp1205*

5.4 Codificação de *flags*

Originalmente o algoritmo LFP utiliza apenas funções lineares para aproximar o resíduo de predição. Portanto, o algoritmo codifica um *flag* para indicar a escolha da aproximação do resíduo: por uma função ou por um vetor do dicionário.

Como consideramos utilizar, além de funções lineares, funções constantes e quadráticas, temos que verificar a melhor maneira de indicar a escolha da aproximação do resíduo. Para isso, realizamos um experimento para saber como codificar as escolhas tomadas pelo algoritmo.

Podemos indicar a escolha da aproximação do resíduo de duas formas. Primeiro, utilizando apenas um *flag* com quatro possíveis valores: dicionário, constante, linear ou quadrática. Ou, podemos indicar utilizando dois *flags*, o primeiro para informar o uso do dicionário ou da função, e o segundo — caso seja escolhida a aproximação por uma função — informar o tipo da função: constante, linear ou quadrática.

Como pode ser verificado nos resultados experimentais, a codificação da escolha da aproximação em dois estágios tem desempenho igual ou superior em relação à codificação com apenas um *flag*. Apesar do *flag* adicional na estratégia de codificação com dois *flags*, a entropia da fonte é atingida mais rapidamente, resultando em melhor desempenho. É importante observar que a codificação do *flag* indicando a escolha da aproximação implica no comportamento estatístico dos coeficientes das funções de aproximação.

Nas Figuras 5.11, 5.12 e 5.13 são exibidos os resultados para as imagens *lena*, *D108* e *pp1205*, respectivamente. No Apêndice C.2.4 estão os resultados para outras imagens.

Resumo da configuração utilizada nesta seção.

- Aproximação do resíduo de predição utilizando combinações das opções de funções constantes, lineares e quadráticas.
- Filtragem dos *pixels* de referência utilizadas para realizar a predição.
- Soma do erro quadrático como medida da distorção na otimização da árvore de segmentação.
- Esquema de quantização original.

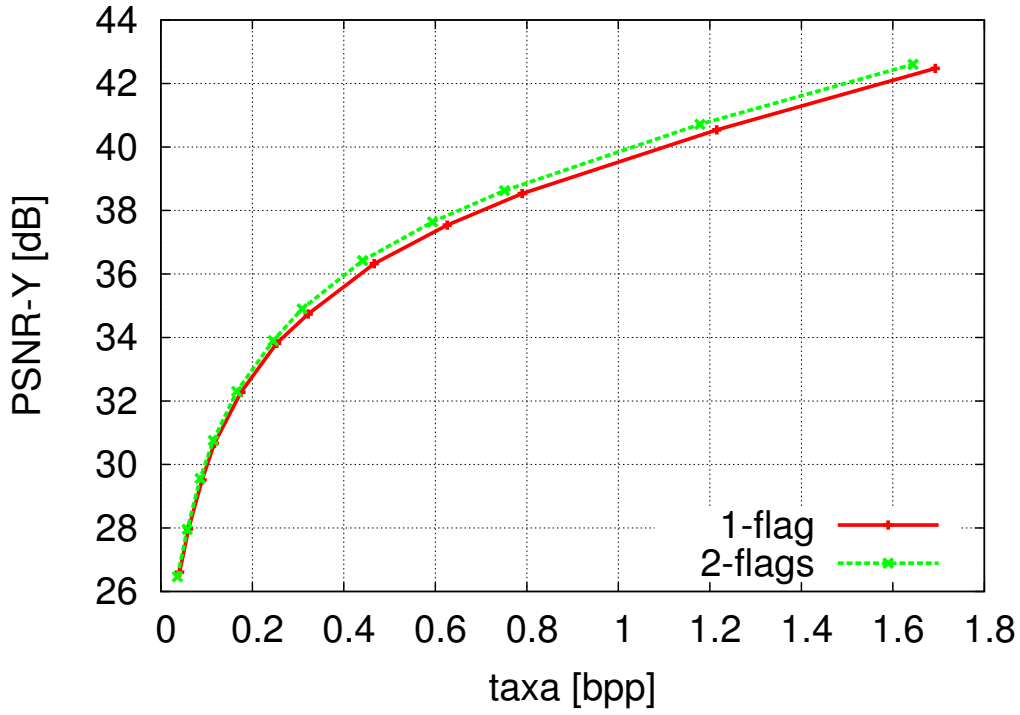


Figura 5.11: Comparação dos resultados taxa-distorção obtidos utilizando diferentes esquemas de codificação de *flags*: *lena*

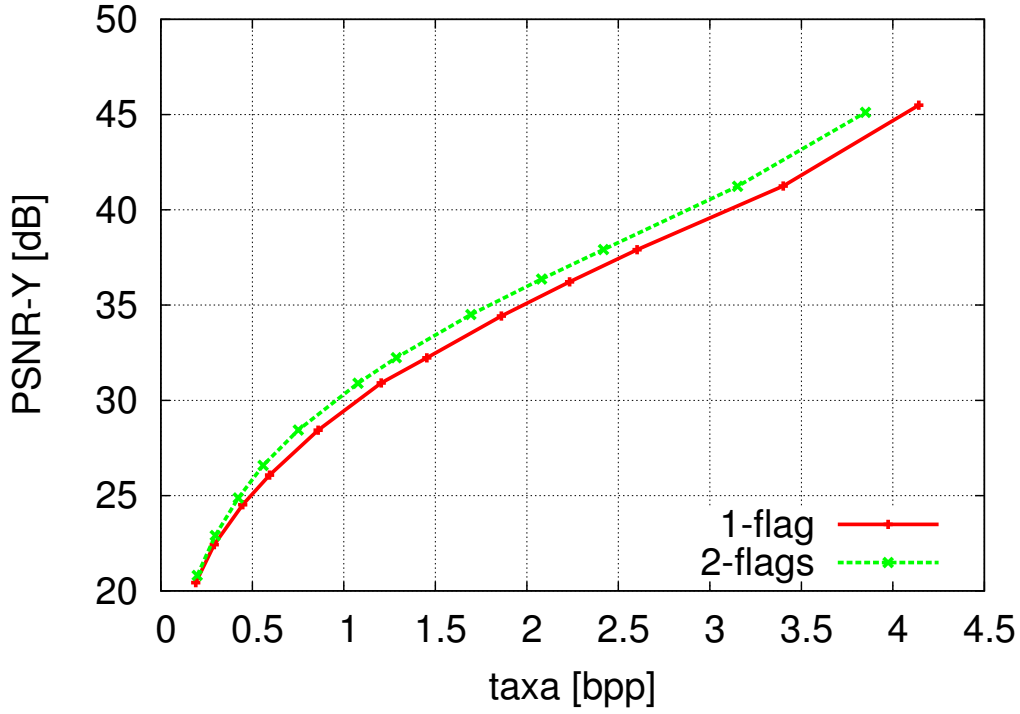


Figura 5.12: Comparação dos resultados taxa-distorção obtidos utilizando diferentes esquemas de codificação de *flags*: *D108*

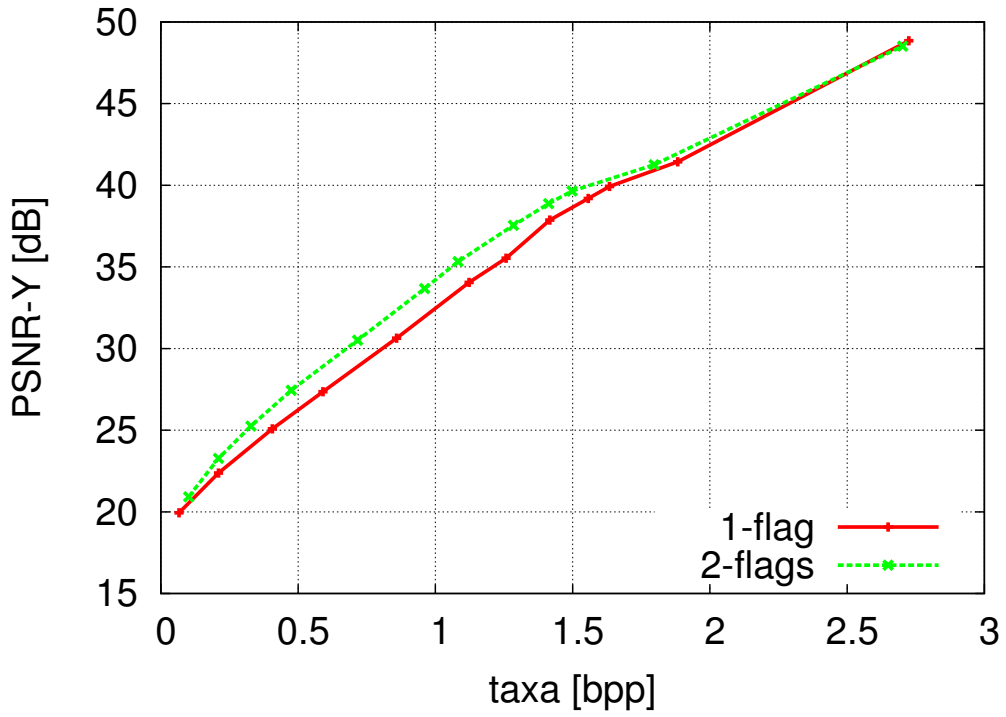


Figura 5.13: Comparação dos resultados taxa-distorção obtidos utilizando diferentes esquemas de codificação de *flags*: *pp1205*

5.5 Dicionários multiescalas

Em métodos baseados em dicionários de múltiplas escalas como o MMP[21], no processo de atualização dos dicionários, o bloco de resíduo aproximado é utilizado para atualizar também dicionários de escalas diferentes da escala de origem, e para tal é realizada uma transformação de escala no bloco. Inspirado no poder de aproximação dos algoritmos que utilizam dicionários multiescalas, consideramos utilizar os blocos reconstruídos (aproximados pelas funções de aproximação) para atualizar não apenas o dicionário da escala correspondente à do bloco, mas também os dicionários de outras escalas. Nesta seção serão discutidas as técnicas que serão experimentadas, e apresentados os resultados alcançados.

5.5.1 Transformação de escala

A atualização multiescala provoca um rápido crescimento da cardinalidade dos dicionários, originalmente o dicionário de cada escala é limitado à 1000 elementos, esse limite foi mantido mesmo com a atualização multiescala descrita nesta subseção.

Para atualizar os dicionários das escalas diferentes da escala de origem, temos que representar o bloco original na escala de destino. Podemos realizar uma contração ou expansão do bloco aproximado de resíduo. Como as coordenadas dos *pixels* são

discretas, reduzimos ou aumentamos o domínio e o número de amostras. Apesar do abuso de linguagem, a transformação de escala pode ser realizada adotando a seguinte condição: o conjunto imagem da função que aproxima o resíduo em dada escala l_o (escala de origem) é igual ao conjunto imagem da função na escala de destino l. Um exemplo numérico é mostrado no Apêndice A.1.

Seja $\mathbf{R}^{l_o}$ um bloco de resíduo da escala de origem l_o, de dimensões $M \times N$. Para o caso de aproximação dos blocos de resíduo de predição utilizando funções lineares, temos:

$$\begin{aligned} 0 \leq x \leq M-1 \text{ e } 0 \leq y \leq N-1 \\ f(x,y) = \alpha_0 x + \alpha_1 y + \alpha_2 \end{aligned} \tag{5.2}$$

Seja $T^{l}_{l_o}(\mathbf{R}^{l_o})$ a operação de transformação (da escala l_o para escala l). E ainda, $g(x', y')$ a função de aproximação do bloco transformado de dimensões $M' \times N'$:

$$\begin{aligned} 0 \leq x' \leq M'-1 \text{ e } 0 \leq y' \leq N'-1 \\ g(x',y') = \alpha_0{}' x' + \alpha_1{}' y' + \alpha_2{}' \end{aligned} \tag{5.3}$$

O domínio de $g(x', y')$ pode ser escrito como uma versão escalada do domínio de $f(x, y)$ como:

$$0 \leq x' \leq \frac{M-1}{\kappa} \text{ e } 0 \leq y' \leq \frac{N-1}{\gamma} \tag{5.4}$$

Dessa forma, $\kappa = \frac{M-1}{M'-1}$ e $\gamma = \frac{N-1}{N'-1}$. Podemos escrever ainda, $x = \kappa x'$ e $y = \gamma y'$.

Admitindo que o conjunto imagem da função $f(x, y)$ seja igual ao conjunto imagem da função $g(x', y')$ e substituindo x e y como anteriormente, temos:

$$\begin{aligned} f(x,y) &= g(x',y') \\ \alpha_0 x + \alpha_1 y + \alpha_2 &= \alpha_0{}' x' + \alpha_1{}' y' + \alpha_2{}' \\ \alpha_0 \kappa x' + \alpha_1 \gamma y' + \alpha_2 &= \alpha_0{}' x' + \alpha_1{}' y' + \alpha_2{}' \\ (\alpha_0 \kappa - \alpha_0{}') = (\alpha_1 \gamma - \alpha_1{}') &= (\alpha_2 - \alpha_2{}') = 0 \end{aligned} \tag{5.5}$$

Podemos escrever então, $\alpha_0{}' = \alpha_0 \kappa$, $\alpha_1{}' = \alpha_1 \gamma$ e $\alpha_2{}' = \alpha_2$. Conduzindo a atualização dessa forma, representamos o bloco aproximado pela função numa escala diferente da escala de origem. Esse procedimento de transformação de escala é realizado antes da atualização dos dicionários de outras escalas. No entanto, antes

de inserir o bloco transformado no dicionário, temos que realizar a equalização da norma como descrito na subseção seguinte.

Aqui tratamos apenas da transformação de escala de blocos aproximados com funções lineares. Os blocos de resíduos aproximados utilizando funções constantes e quadráticas também sofrem transformações de escala. A transformação de escala aplicada em blocos aproximados com funções quadráticas pode ser consultada no Apêndice A.2. No caso da função constante, a transformação é direta, interpolando ou decimando amostras com o valor da média do bloco, dependendo da transformação (contração ou expansão). Notemos que as transformações das funções lineares e quadráticas também são decimações ou interpolações.

5.5.2 Equalização da norma dos blocos escalados

Fundamentado em premissas teóricas apresentadas em [30], [31] e [32] e em resultados experimentais, em [21] foi proposta a equalização da norma. Resultados experimentais mostraram que uma melhor aproximação do resíduo de predição pode ser alcançada se os vetores de resíduo (armazenados nos dicionários), após a transformação de escala, preservarem a norma.

Diversos experimentos foram realizados considerando várias normas e práticos comprimentos de vetores [21]. E, apesar da violação da condição de vetores com comprimento assintoticamente grande, foi mostrado que há uma consistência na norma dos vetores.

Sendo assim, a equalização da norma foi definida como:

$$\mathbf{R}^{l} = S_{p}^{l_o,l} T_{l_o}^{l}(\mathbf{R}^{l_o}) \tag{5.6}$$

Onde,

$$S_{p}^{l_o,l} = \frac{|\mathbf{R}^{l_o}|_p}{|T_{l_o}^{l}(\mathbf{R}^{l_o})|_o} \tag{5.7}$$

Nas equações 5.6 e 5.7, $\mathbf{R}^{l_o}$ é o bloco de resíduo pertencente a escala l_o, $T_{l_o}^{l}(\centerdot)$ é a operação de transformação da escala l_o para a escala l, p é o coeficiente de forma da gaussiana generalizada que modela a distribuição do resíduo de predição e $|\centerdot|_p$ é a norma utilizada definida como:

$$|\mathbf{x}|_p = \left(\sum_i |x_i|^p \right)^{1/p} \tag{5.8}$$

A equalização é aplicada após a transformação de escala e antes do bloco ser inserido em dicionários de escalas diferentes da escala de origem. Baseado em resultados experimentais foi sugerido usar a norma $|\centerdot|_{p=1}$. Dessa forma, os resultados

obtidos são, em geral, superiores aos resultados obtidos utilizando outras normas testadas. É importante notar que a equalização da norma é efetiva para um melhor desempenho apenas quando aplicada nas transformações de expansão. A justificativa é que nas transformações de contração a probabilidade do novo bloco ser efetivamente util é pequena [21].

5.5.3 Controle de redundância

O uso de transformação de escala para a atualização de dicionários de outras escalas pode provocar um rápido crescimento da cardinalidade dos dicionários, criando redundância dos dicionários. Um dicionário populoso pode representar uma melhor aproximação para os blocos de resíduo. No entanto, pode significar também um aumento de taxa necessária para a codificação dos índices dos vetores.

Podemos adotar um critério para controlar a redundância dos dicionários. A restrição da inclusão de um novo vetor se próximo dos vetores existentes no dicionário, considerando que ofereceria pouco ganho de poder de aproximação, pode ser adotada, de maneira semelhante ao realizado em [21].

Logo, é razoável calcular a distância do vetor a ser inserido em relação aos vetores presentes no dicionário, e apenas inserir os vetores que distam no mínimo d de cada vetor presente no dicionário. Onde d é o raio da hiperesfera no espaço multidimensional de cada escala defina por:

$$\sum \left(\mathbf{X}^l(m,n) - \mathbf{S}^l(m,n)\right)^2 > d^2 \tag{5.9}$$

Onde $\mathbf{X}^l$ é o vetor candidato a ser inserido no dicionário da escala l e $\mathbf{S}^l$ é um vetor pertencente ao dicionário. A distância é calculada em relação a cada vetor presente no dicionário.

Intuitivamente, um valor pequeno de d significa a inserção de palavras próximas, o que não necessariamente contribui para um significativo ganho de aproximação por esse dicionário. Um d pequeno ainda provoca o crescimento no número de elementos no dicionário, que pode elevar a taxa necessária para a codificação dos índices, além de não ser efetivo em controlar a redundância.

Por outro lado, um valor grande de d implica num elevado espaçamento entre os vetores de reconstrução. Portanto, o poder de aproximação do dicionário ficaria comprometido, pois teríamos regiões sem vetores representativos. Existe um compromisso entre o controle de redundância e o desempenho do algoritmo. Sabemos da equação 3.6 que o parâmetro λ pondera a taxa pretendida e a distorção incorrida.

Em [21] foi hipotetizado relacionar heuristicamente o valor de d em função de λ. Vários experimentos foram realizados usando diversos valores de d e λ resultando

na seguinte equação:

$$d(\lambda) = \begin{cases} 5, & \text{se} \quad \lambda \leq 15, \\ 10, & \text{se} \quad 15 < \lambda \leq 50 \\ 20, & \text{outros casos} \end{cases}$$

Além da heurística relacionando o valor de d ao de λ, foi apontando que o uso do mesmo valor de d para as diversas escalas é benéfico para o desempenho do algoritmo. Para os dicionários das escalas maiores, o controle de redundância não é tão restritivo. Isso se justifica pelo fato do dicionário populoso aumentar a probabilidade de uma grande região ser codificada eficientemente, enquanto que em escalas menores um controle mais severo é realizado, impedindo a inserção de palavras que não ofereçam significativo ganho de aproximação para esse dicionário.

5.5.4 Codificação de índices dos dicionários

A atualização multiescala dos dicionários pode causar significativa diferença de probabilidade entre as escalas de origem dos vetores. Isso pode ser útil para a codificação eficiente dos índices do dicionário em duas etapas [33][21].

Baseado em dados estatísticos do uso dos vetores oriundos de diferentes escalas, foi proposta a segmentação do dicionário de cada escala de acordo com a escala de origem de cada vetor [33]. Ou seja, os vetores de cada dicionário seriam identificados primeiro pela escala de sua origem e então pelo indíce dentro desse subconjunto.

Na prática, em vez de codificar diretamente apenas o índice do vetor no dicionário (a escala é implicita para o decodificador), são codificados dois símbolos: um *flag* para indicar o subdicionário (escala de origem do vetor) e o índice do vetor dentro desse subdicionário.

No estudo citado anteriormente, a segmentação do dicionário segundo o modo de predição aplicado também foi experimentada. No entanto, a segmentação pela escala de origem fornece, em geral, resultado superior.

Inspirados nos resultados obtidos com o algoritmo MMP[21] que utiliza segmentação do dicionário segundo a escala de origem, e ainda pela similaridade com o MMP, considerando a atualização multiescala dos dicionários investigada no LFP, experimentamos utilizar a segmentação dos dicionários no LFP com atualização multiescala dos dicionários.

Resultados experimentais — Dicionários multiescalas

Diversos experimentos foram realizados para verificar a eficácia de técnicas multiescalas. Nos gráficos exibidos nesta seção, *ua* (*update all*) siginifica o uso da atualização multiescala, *rc* (*redundancy control*) significa o uso do controle de re-

dundância e *dp* (*dictionary partition*) a segmentação do dicionário segundo a escala de origem.

Nas Figuras 5.14, 5.15 e 5.16 estão esboçados os resultados experimentais das imagens *lena*, *D108* e *pp1205*, respectivamente. Os resultados para outras imagens são encontradas no Apêndice C.2.5.

Baseados nos resultados experimentais, podemos verificar que individualmente o controle de redundância e a atualização multiescala têm desempenhos inferiores em relação ao esquema que combina ambas. Podemos deduzir que o uso do controle de redundância impede a inserção de vetores no dicionário que seriam úteis, limitando o poder de aproximação do dicionário. Enquanto, a atualização multiescala provoca um rápido crescimento da cardinalidade dos dicionários, o que pode implicar num aumento de taxa para codificar os índices dos dicionários.

O emprego da segmentação dos dicionários teve desempenho, em geral, similar em relação aos esquemas sem segmentação dos dicionários. Apenas nas imagens *pp1205* e *pp1209* a segmentação dos dicionários teve melhor desempenho.

Resumo da configuração utilizada nesta seção.

- Aproximação do resíduo de predição com as opções de funções constantes, lineares e quadráticas .
- Filtragem dos *pixels* de referência utilizadas para realizar a predição.
- Soma do erro quadrático como medida da distorção na otimização da árvore de segmentação.
- Esquema de quantização original.

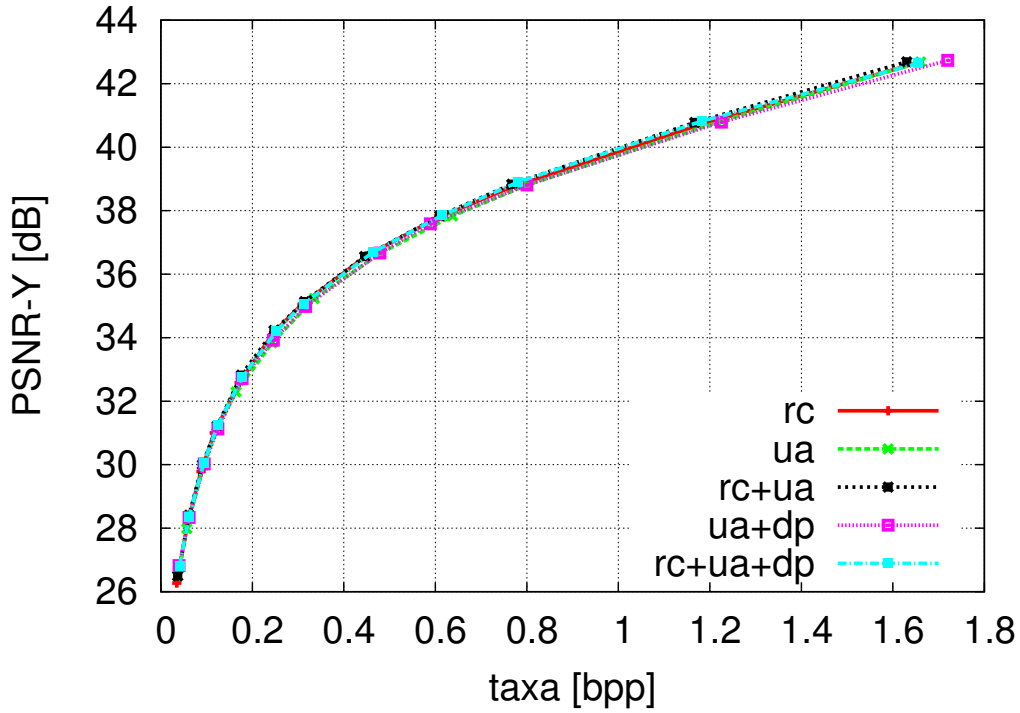


Figura 5.14: Comparação desempenho taxa-distorção para diferentes estratégias de atualização dos dicionários: *lena*. *rc*-controle de redundância, *ua*-atualização multiescalas, *dp*-partição do dicionário por escala de origem

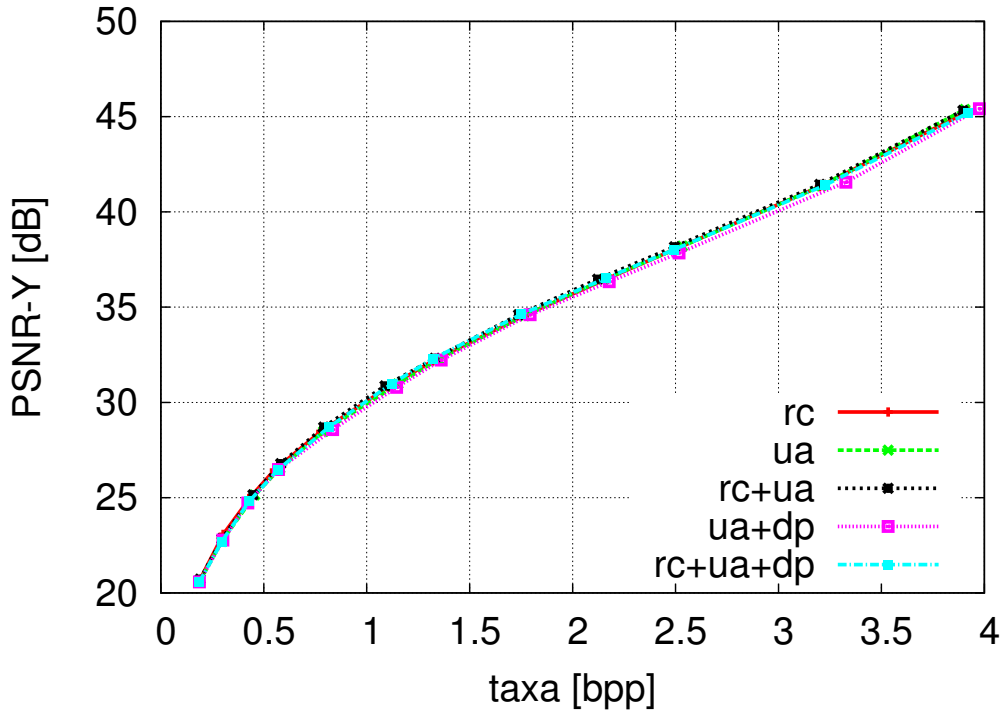


Figura 5.15: Comparação desempenho taxa-distorção para diferentes estratégias de atualização dos dicionários: *D108*. *rc*-controle de redundância, *ua*-atualização multiescalas, *dp*-partição do dicionário por escala de origem

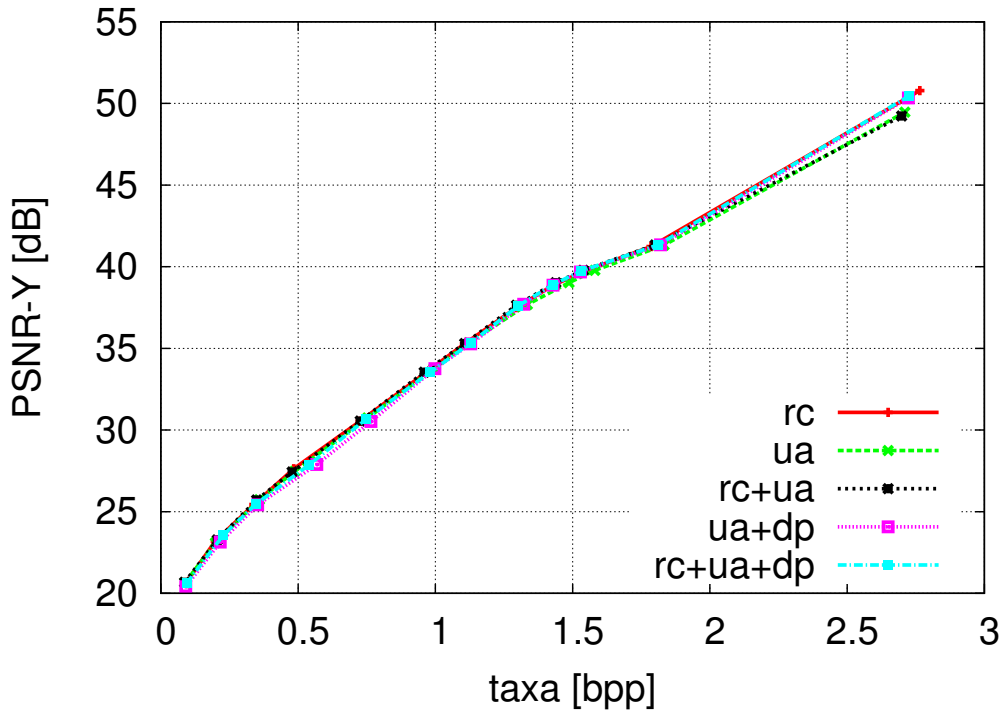


Figura 5.16: Comparação desempenho taxa-distorção para diferentes estratégias de atualização dos dicionários: *pp1205*. *rc*-controle de redundância, *ua*-atualização multiescalas, *dp*-partição do dicionário por escala de origem

5.6 Resultados experimentais

Baseados nos resultados experimentais das seções anteriores, conduzimos os experimentos desta seção utilizando a soma erro quadrático como critério de distorção e a filtragem dos *pixels* nas fronteiras dos blocos utilizados na predição. A aproximação do resíduo de predição foi realizada utilizando funções constantes, lineares e quadráticos, permitindo assim ao algoritmo escolher adaptativamente a melhor função de aproximação baseado no custo taxa-distorção. O esquema de quantização original foi utilizado, pois experimentalmente teve melhor desempenho [7]. A atualização multiescala com controle de redundância nos dicionários foi utilizada, pois também exibiu desempenho superior nos resultados experimentais.

Os resultados obtidos mostram que o algoritmo tem desempenho inferior em relação aos padrões de compressão H264/AVC[4][5] e HEVC[6][20] em imagens naturais, conforme ilustrado nas Figuras 5.17, 5.23 e 5.25, para as imagens *lena*, *barb* e *zelda*, respectivamente. Nas imagens de texto e composta, o algoritmo teve bom desempenho em comparação aos padrões de compressão avaliados. Na imagem *pp1205* (imagem de texto) e na *pp1209* (imagem composta) teve desempenho superior em relação ao H264/AVC e próximo do HEVC. Na imagem *D108* (gerada por computador) teve desempenho inferior aos padrões.

O HEVC emprega 35 modos de predição, o que permite obter um resíduo muito mais concentrado em torno de zero, além disso, ele realiza uma otimização taxa-distorção a partir de blocos de 64×64 *pixels*, permitindo uma eficiente codificação de regiões uniformes. Portanto, é natural a superioridade de desempenho do HEVC. Os experimentos utilizando o HEVC[6][20] foram conduzidos de duas formas. Em princípio, não alteramos nenhum parâmetro do codificador (curvas *hevc-64*). Na segunda, alteramos o tamanho máximo da *Coding Unit* (CU) de 64 para 32 *pixels* (curvas *hevc-32*). Mesmo com a mudança desse parâmetro, a otimização continua sendo realizada a partir de blocos de 64×64. O ganho de desempenho do HEVC utilizando máximo CU igual à 64 é siginificativo apenas em imagens não suaves.

O padrão H264/AVC[4][5] em imagens naturais teve desempenho, em geral, superior em relação ao LFP. No entanto, na imagem *zelda* o desempenho do LFP fica muito próximo do H264/AVC. Em imagens de textos e compostas, o LFP é superior ao H264/AVC, como nos casos das imagens *pp1205* e *pp1209* Figuras 5.21 e 5.27, respectivamente.

Utilizamos a implementação JM-18.5[1] do H264/AVC. No caso do HEVC utilizamos o HM-8.2[2]. Os arquivos de configuração *defaults* desses codificadores foram utilizados. Esses arquivos de configuração são distribuídos juntos com esses *softwares*.

As Figuras 5.18, 5.20, 5.22, 5.24, 5.26 e 5.28 exibem as imagens originais e as

imagens reconstruídas codificadas com o LFP. Para cada imagem, dois exemplos são exibidos, permitindo verificar o desempenho em taxas diferentes.

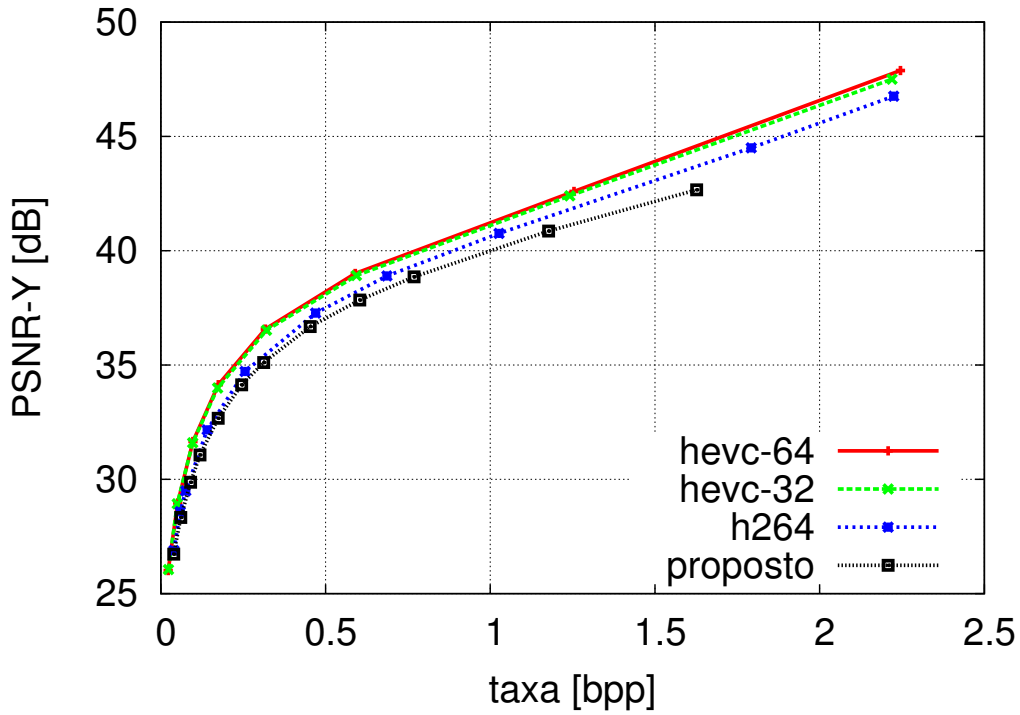


Figura 5.17: Comparação do desempenho taxa-distorção do esquema proposto com os dos *softwares* de referência do H.264 e do HEVC: *lena*. *hevc-64* e *hevc-32* referem-se ao tamanho inicial das CUs.

[1]Disponivel para download em `http://iphome.hhi.de/suehring/tml/download/`

[2]Pode-se fazer uma cópia do projeto com o sistema de controle de versão SVN em `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/`

(a) Imagem *lena* original

(b) Taxa $= 0,1740$ bpp PSNR $= 32,673$ dB

(c) Taxa $= 0,6039$ bpp PSNR $= 37,846$ dB

Figura 5.18: Imagem *lena* original e as imagens reconstruídas codificadas com o LFP.

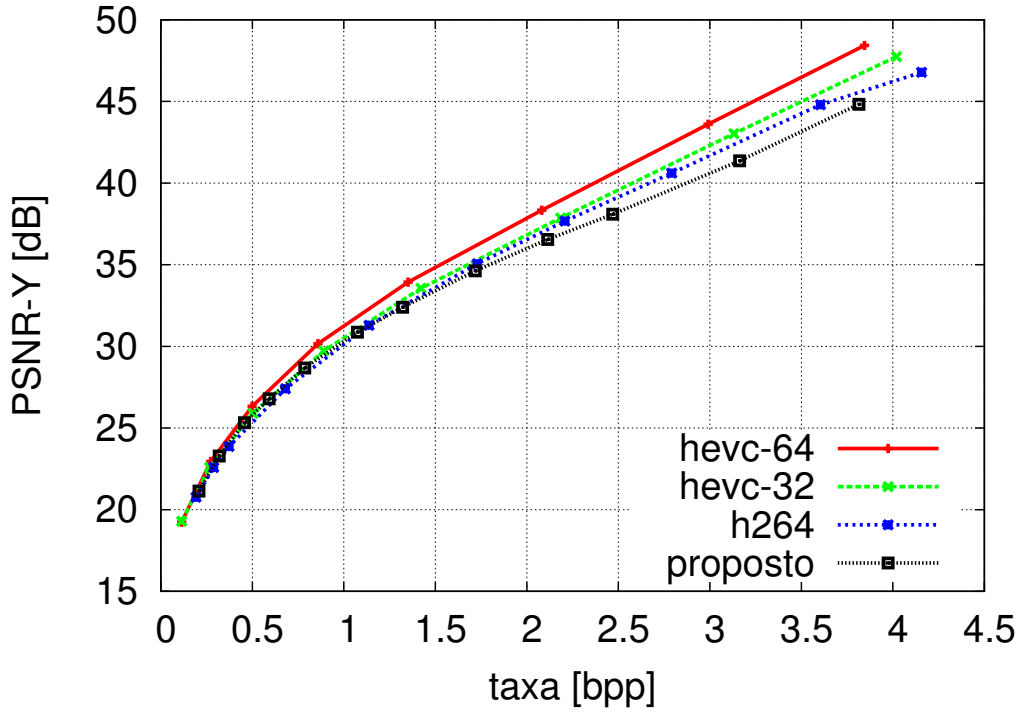
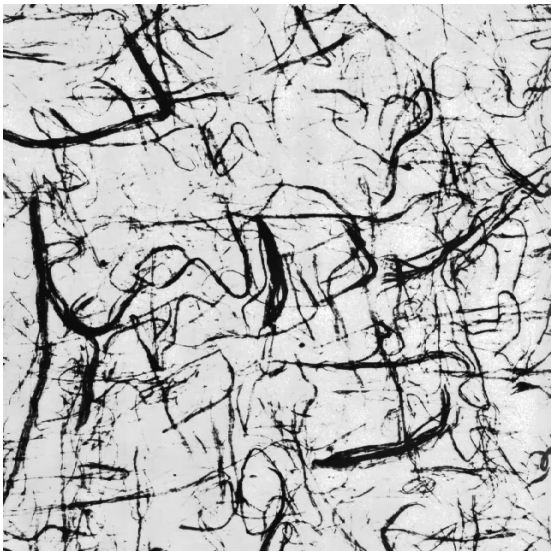


Figura 5.19: Comparação do desempenho taxa-distorção do esquema proposto com os dos *softwares* de referência do H.264 e do HEVC: *D108*. *hevc-64* e *hevc-32* referem-se ao tamanho inicial das CUs.

(a) Imagem *D108* original

(b) Taxa $= 0,7863$ bpp PSNR $= 28,675$ dB



(c) Taxa $= 2,1139$ bpp PSNR $= 36,544$ dB

Figura 5.20: Imagem *D108* original e as imagens reconstruídas codificadas com o LFP.

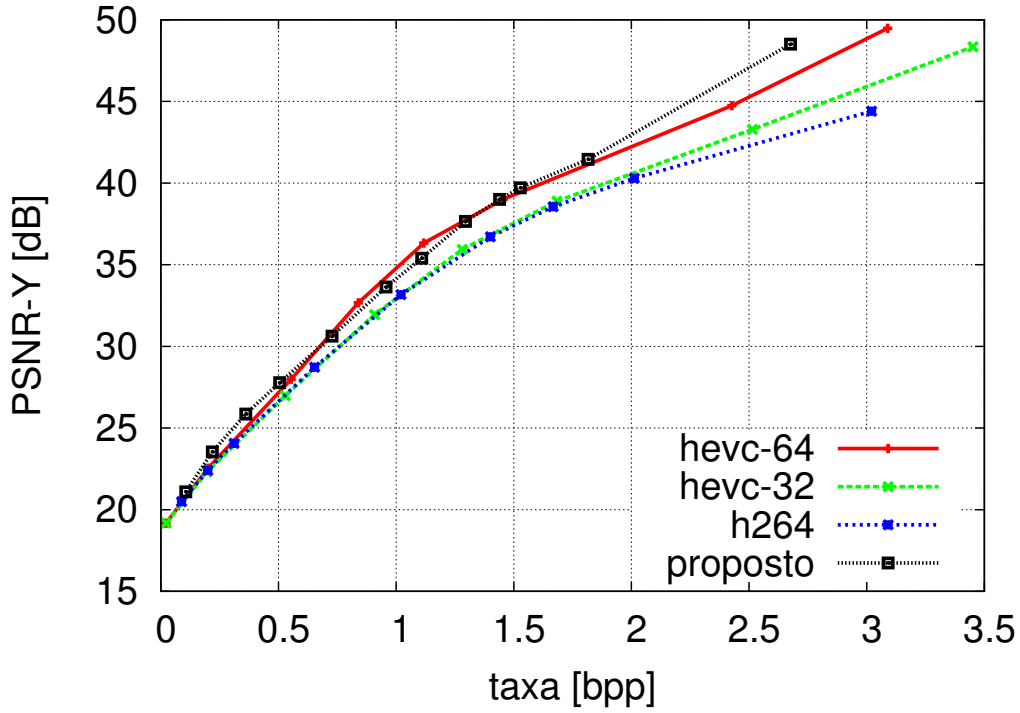
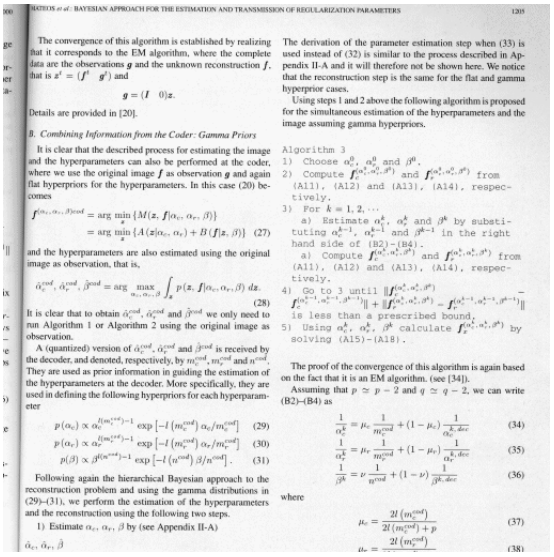
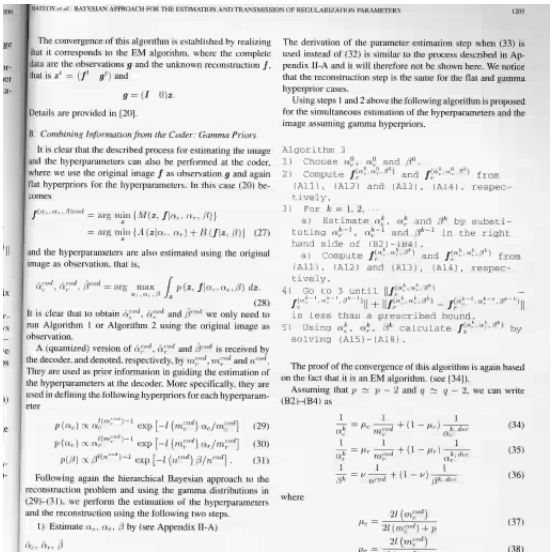


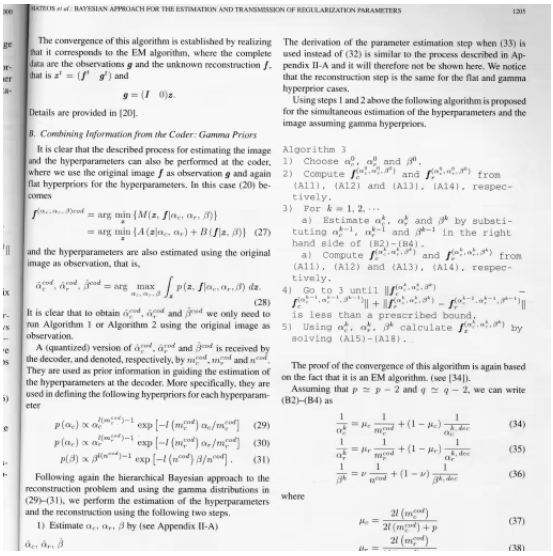
Figura 5.21: Comparação do desempenho taxa-distorção do esquema proposto com os dos *softwares* de referência do H.264 e do HEVC: *pp1205*. *hevc-64* e *hevc-32* referem-se ao tamanho inicial das CUs.



(a) Imagem *pp1205* original



(b) Taxa $= 0,7279$ bpp PSNR $= 30,629$ dB



(c) Taxa $= 1,4399$ bpp PSNR $= 38,999$ dB

Figura 5.22: Imagem *pp1205* original e as imagens reconstruídas codificadas com o LFP.

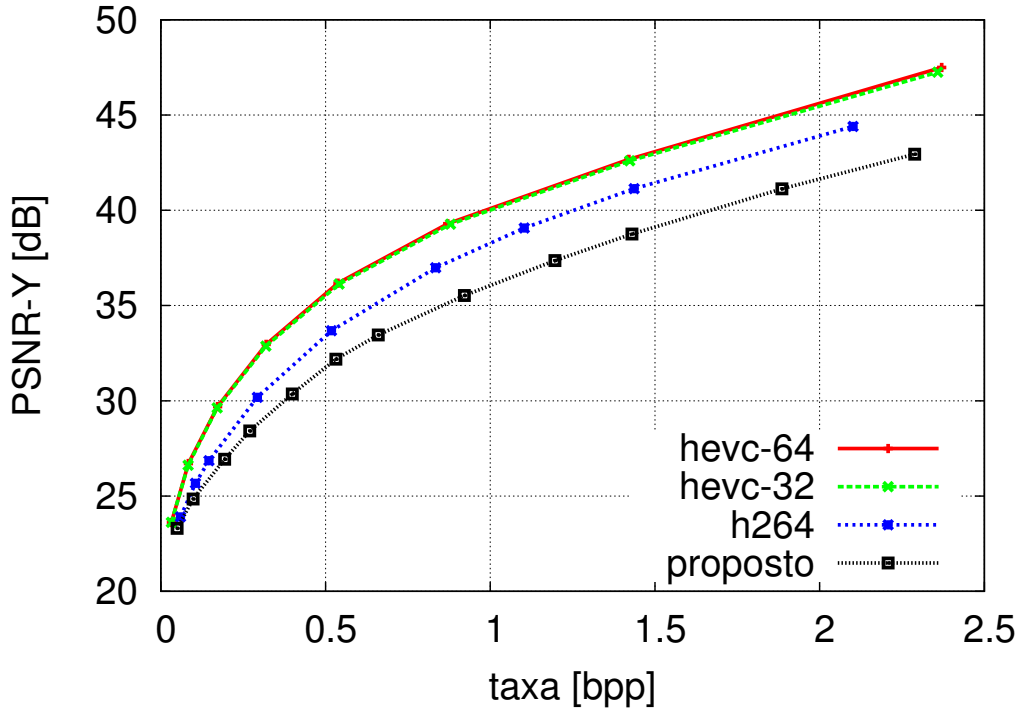


Figura 5.23: Comparação do desempenho taxa-distorção do esquema proposto com os dos *softwares* de referência do H.264 e do HEVC: *barb*. *hevc-64* e *hevc-32* referem-se ao tamanho inicial das CUs.

(a) Imagem *barb* original

(b) Taxa = 0,3990 bpp PSNR = 30,359 dB

(c) Taxa = 1,1963 bpp PSNR = 37,359 dB

Figura 5.24: Imagem *barb* original e as imagens reconstruídas codificadas com o LFP.

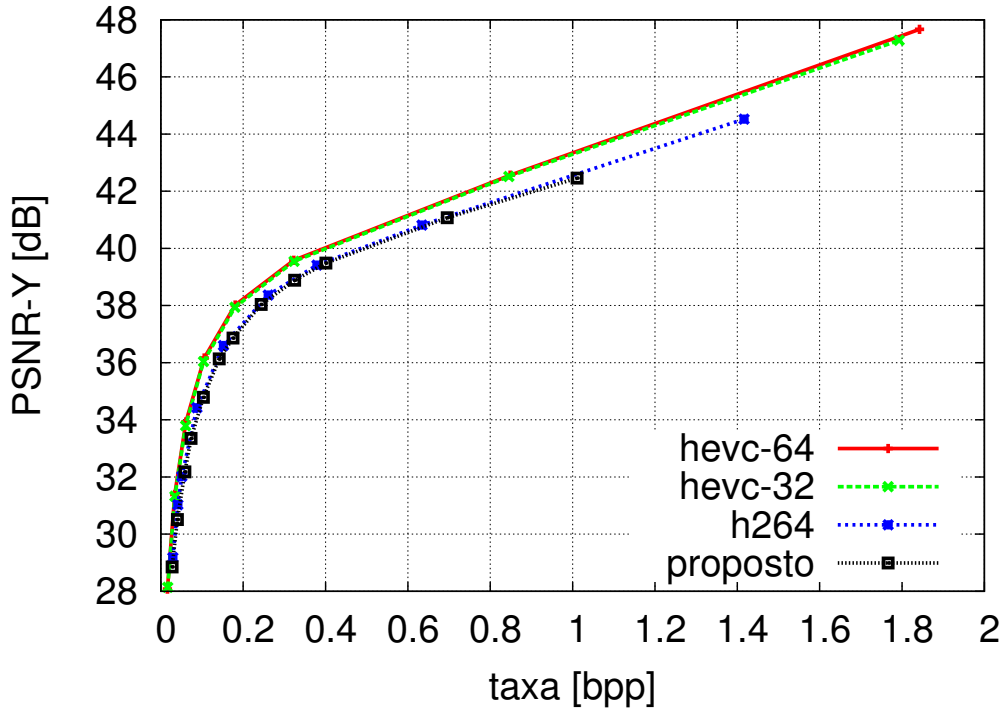


Figura 5.25: Comparação do desempenho taxa-distorção do esquema proposto com os dos *softwares* de referência do H.264 e do HEVC: *zelda*. *hevc-64* e *hevc-32* referem-se ao tamanho inicial das CUs.

(a) Imagem *zelda* original

(b) Taxa $= 0,1033$ bpp PSNR $= 34,789$ dB

(c) Taxa $= 0,3251$ bpp PSNR $= 38,887$ dB

Figura 5.26: Imagem *zelda* original e as imagens reconstruídas codificadas com o LFP.

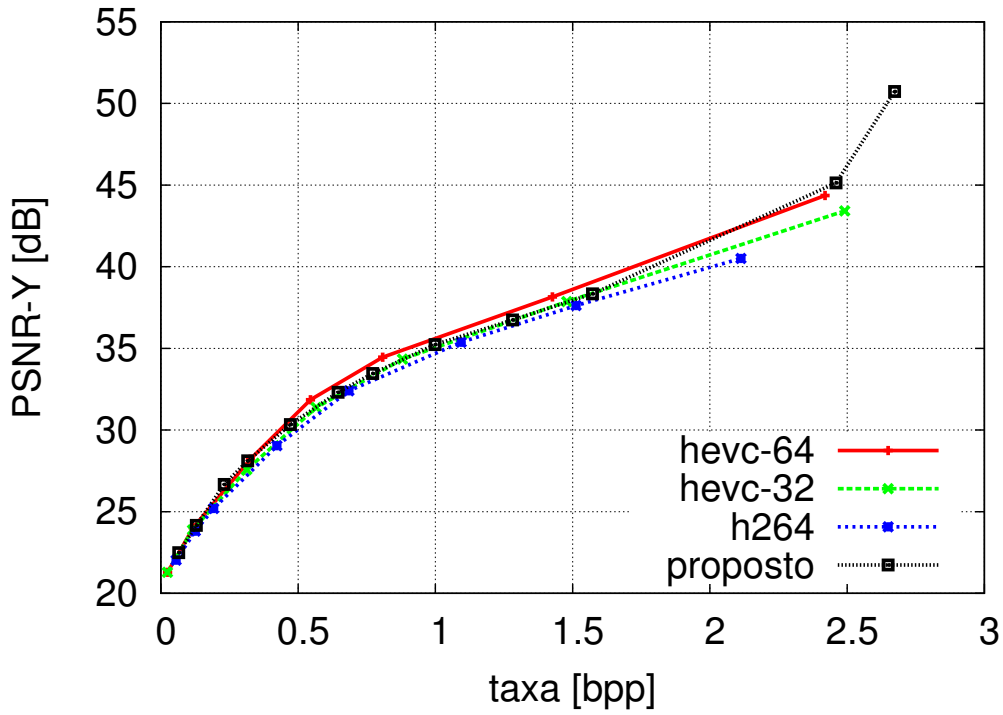
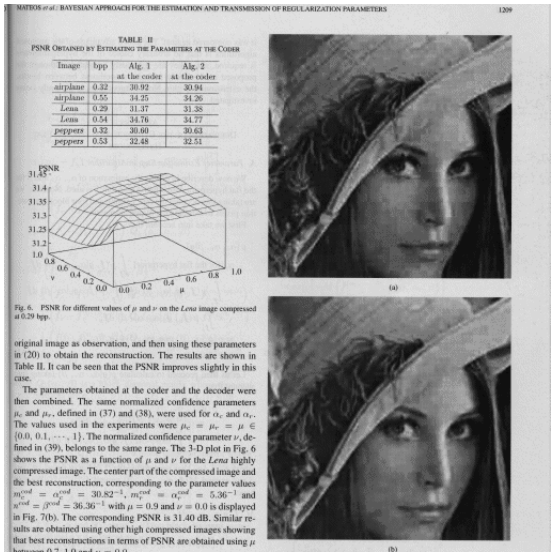
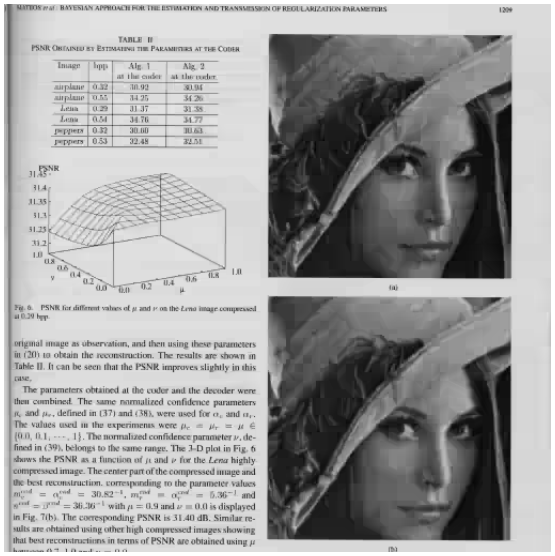


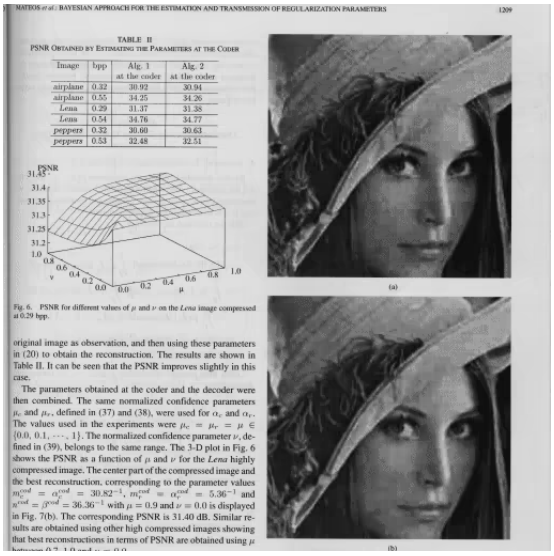
Figura 5.27: Comparação do desempenho taxa-distorção do esquema proposto com os dos *softwares* de referência do H.264 e do HEVC: *pp1209*. *hevc-64* e *hevc-32* referem-se ao tamanho inicial das CUs.



(a) Imagem *pp1209* original



(b) Taxa $= 0,4719$ bpp PSNR $= 30,334$ dB



(c) Taxa $= 1,2812$ bpp PSNR $= 36,738$ dB

Figura 5.28: Imagem *pp1209* original e as imagens reconstruídas codificadas com o LFP.

Capítulo 6

Conclusões

Neste trabalho, a partir do algoritmo LFP, investigamos novos esquemas para melhorar o desempenho do algoritmo na codificação de mapas de profundidade, bem como a adaptação do algoritmo para a codificação de imagens de textura.

Em relação à codificação de mapas de profundidade, investigamos a aproximação do resíduo de predição utilizando, além de funcões lineares, funções constantes e quadráticas. Os resultados experimentais mostram que o uso da função constante contribui para um efetivo ganho de desempenho. Isso acontece sobretudo em regiões suaves, pertencentes a um mesmo objeto da cena, onde a técnica de predição fornece um resíduo uniforme. A função quadrática contribui para um ganho de desempenho em taxas (*bit rate*) mais elevadas e em algumas imagens. Individualmente a aproximação do resíduo apenas com funções constantes ou quadráticas não fornece ganho de desempenho. Em alguns mapas de profundidade, observamos que o desempenho do algoritmo — empregando funções constantes e lineares — é superior ao que utiliza funções constantes, lineares e quadráticas. No entanto, optamos pelo esquema que emprega as três funções nas simulações finais. Dessa forma, o algoritmo pode escolher adaptativamente como realizar a aproximação com as funções. Assim, em geral, obtem-se melhor desempenho sem fazer asserções sobre os mapas de profundidade a serem codificados.

No que diz respeito à codificação de mapas, investigamos um esquema de quantização dos coeficientes. A partir de amostras dos coeficientes das funções de aproximação, modelamos as distribuições de probabilidade dos coeficientes. Depois disso, com as amostras geradas com os modelos, foi obtido o quantizador ótimo para cada coeficiente. Quantidades diferentes de níveis de reconstrução foram testadas, e baseados em resultados experimentais, definimos um esquema de quantização em função da taxa controlada pelo multiplicador de Lagrange (λ). Em vez de usarmos um esquema de quantização fixo para todas as taxas, obtemos um desempenho superior empregando um quantizador mais grosseiro em taxas baixas[1] e um quantizador

com mais níveis de reconstrução em taxas mais elevadas.

Depois, empregamos os esquemas de aproximação do resíduo e de quantização dos coeficientes. Comparamos os resultados das vistas virtuais geradas a partir dos mapas codificados e das imagens de textura originais. Os resultados mostram que o algoritmo, em geral, tem desempenho superior ao H264. E ainda, desempenho próximo ao do HEVC em algumas sequências, ou mesmo superior em outras. Esse pode ser considerado um excelente resultado, principalmente se levarmos em conta que o HEVC emprega 35 modos de predição e otimização da taxa-distorção a partir de segmentos de blocos maiores, contra apenas 9 modos de predição disponíveis no LFP.

Na codificação de imagens de textura, a utilização da soma do erro quadrático no processo de otimização da árvore de segmentação e o uso da filtragem dos *pixels* de referência para predição fornecem um ganho de desempenho. Pois, dessa forma, consideramos a qualidade visual da imagem reconstruída. A aproximação do resíduo com funções polinomiais e o esquema de quantização dos coeficientes foram investigados também na codificação de imagens de textura. Resultados experimentais, diferentemente dos obtidos para a codificação de mapas, relevaram que o uso de funções constantes e quadráticas, além de funções lineares, na aproximação do resíduo confere pouco de ganho ao desempenho do algoritmo, como também o esquema de quantização em função da taxa desejada.

O uso de funções adicionais para a aproximação do resíduo, aumenta o número *flags* a serem codificados. Foram realizados experimentos para encontrar a melhor maneira de codificação. Obtemos o melhor desempenho codificando um *flag* para indicar a aproximação por um vetor do dicionário ou pela função. Depois disso, no caso de escolha da função, outro *flag* indicando a função escolhida: constante, linear ou quadrática.

Ainda na codificação de imagens de textura, investigamos utilizar atualização multiescala dos dicionários. Após a transformação de escala e a equalização da norma do bloco de resíduo, o vetor foi utilizado para atualizar os dicionários de escalas diferentes da de origem, e ainda a atualização da escala de origem com o bloco original aproximado. Diversas técnicas foram experimentadas como o controle de redundância dos dicionários e a segmentação dos dicionários. Os resultados experimentais mostram que o uso da atualização multiescala dos dicionários resulta apenas em pequeno ganho de desempenho, em geral. Em algumas imagens pode-se observar um ganho mais significativo, no entanto, o crescimento do dicionário devido à atualização multiescala torna o algoritmo mais complexo computacionalmente.

Nos resultados experimentais finais, utilizamos a soma do erro quadrático como critério de distorção, além da filtragem para uma melhor predição, bem como a atu-

[1]No sentido de *bit rate* e não de taxa de compressão

alização multiescala e controle de redundância dos dicionários. Para a aproximação do resíduo empregamos funções constantes, lineares e quadráticas. O esquema de quantização original foi utilizado [7].

O algoritmo teve desempenho inferior em relação aos padrões de compressão avaliados em imagens naturais. No caso de imagens de textos e compostas, o algoritmo, em geral, teve desempenho superior ou igual aos padrões.

Trabalhos futuros

Como propostas de investigação, pode-se estender o algoritmo para a codificação de vídeos — tanto de mapas quanto de textura — com técnicas de predição temporal e predição intra mais apuradas. Resultados preliminares mostraram, por exemplo, significativo ganho de desempenho utilizando o modo de predição *least square* [34]. Pode-se investigar a utilização dos vetores dos dicionários para predição. Porém, como os dicionários guardam vetores de resíduos (pelo menos na implementação por conveniência), os blocos da imagem deverão ser reconstruídos antes de serem utilizados como blocos de predição.

A otimização da árvore de segmentação utiliza, em cada nível, o modo de predição que fornece o resíduo com menor norma L1. A escolha do modo de predição pode ser realizada com base no custo lagrangeano, ou seja, em termos de taxa para codificar o modo de predição e da distorção incorrida pelo modo de predição.

Atualmente, em alguns algoritmos de compressão de mapas, a codificação é otimizada em função da qualidade de vistas sintetizadas durante a codificação. A adaptação do LFP para a codificação de mapas considerando a síntese de vistas pode ser explorada. Dessa forma, podemos fazer uma comparação direta com algoritmos estado da arte, como o HEVC-3D.

Referências Bibliográficas

[1] BOURGE, A., GOBERT, J., BRULS, F. "MPEG-C PART 3: Enabling the Introduction of Video Plus Depth Contents". 2006. Philips Applied Technologies.

[2] MULLER, K., MERKLE, P., WIEGAND, T. "3-D Video Representation Using Depth Maps", *Proceeding of the IEEE*, v. 99, n. 4, pp. 643–656, 2011.

[3] WALLACE, G. K. "The JPEG Still Picture Compression Standard", *IEEE Transactions on Consumer Electronics*, v. 38, n. 1, pp. 18–34, 1992.

[4] "Advanced video coding for generic audiovisual services". 2010. ITU-T and ISO/IEC JTC1 - Recommendation H.264.

[5] RICHARDSON, I. E. G. *H.264 and MPEG-4 Video Compression*. 1st ed. Chichester, West Sussex, England, John Wiley e Sons Ltd., 2003.

[6] SULLIVAN, G. J., OHM, J.-R., HAN, W.-J., et al. "Overview of the High Efficient Video Coding (HEVC) Standard", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 22, n. 12, pp. 1858–1870, 2012.

[7] LUCAS, L. F. R., RODRIGUES, N. M. M., PAGLIARI, C. L., et al. "Efficient Depth Map Coding using Linear Residue Approximation and a Flexible Prediction Framework". In: *International Conference on Image Processing*, pp. 1305–1308, 2012.

[8] SHANNON, C. E. "A Mathematical Theory of Communication", *The Bell System Technical Journal*, v. 27, pp. 379–423, July 1948.

[9] PEEBLES, P. Z. *Probability, Random Variables, and Random Signal Principles*. 3rd ed. New York, NY, McGraw-Hill, Inc., 1993.

[10] COVER, T. M., THOMAS, J. A. *Elements of Information Theory*. 2nd ed. Hoboken, New Jersey, John Wiley e Sons, Inc., 2006.

[11] SAYOOD, K. *Introduction to Data Compression*. 3rd ed. San Francisco, CA, Elsevier Inc., 2006.

[12] SHI, Y. Q., SUN, H. *Image and Video Compression for Multimedia Engineering*. 2nd ed. Boca Raton, FL, CRC Press, 2008.

[13] MAX, J. "Quantizing for Minimum Distortion". In: *IRE Transactions on Information Theory*, v. 6, pp. 7–12, 1960.

[14] MCMILLAN, B. "Two Inequalities Implied by Unique Decipherability", *IRE Transactions on Information Theory*, v. 2, n. 4, pp. 115–116, 1956.

[15] HUFFMAN, D. A. "A Method for the Construction of Minimum Redundancy Codes". In: *Proceeding of IRE*, v. 40, pp. 1098–1101, 1952.

[16] HAMMING, R. W. *Coding and Information Theory*. 2nd ed. Englewood, New Jersey, Prentice-Hall, 1986.

[17] BELL, T. C., CLEARY, J. G., WITTEN, I. H. *Text Compression*. 1st ed. Englewoods Cliffs, New Jersey, Prentice Hall., 1990.

[18] WITTEN, I. H., NEAL, R. M., CLEARY, J. G. "Arithmetic Coding for Data Compression", *Communications of the ACM*, v. 30, n. 6, pp. 520–540, 1987.

[19] SKODRAS, A., CHRISTOPOULOS, C., EBRAHIMI, T. "The JPEG 2000 Still Image Compression Standard", *IEEE Signal Processing Magazine*, v. 18, n. 5, pp. 36–58, 2001.

[20] BOSSEN, F., BROSS, B., SUHRING, K., et al. "HEVC Complexity and Implementation Analysis", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 22, n. 12, pp. 1685–1696, 2012.

[21] RODRIGUES, N. M. M. *Multiscale Recurrent Pattern Matching Algorithms for Image and Video Coding*. Tese de doutorado, Universidade de Coimbra, Coimbra, Portugal, 2008.

[22] STRANG, G. *Linear Algebra and Its Applications*. 4th ed. Belmont, CA, Thomson Learning, 2006.

[23] STRANG, G. *Introduction to Applied Mathmatics*. 2nd ed. Wellesley, MA, Wellesley-Cambridge Press, 1986.

[24] "High Efficiency Video Coding Reference Software, Fraunhofer Institute". Website. `https://hevc.hhi.fraunhofer.de/`. Acessado em agosto de 2013.

[25] "A practical procedure to estimate the shape parameter in the generalized Gaussian Distribution". Website. `http://www.cimat.mx/reportes/enlinea/I-01-18_eng.pdf`. Acessado em agosto de 2013.

[26] WANG, T., LI, H., LI, Z., et al. "A Fast Parameter Estimation of Generalized Gaussian Distribution". In: *International Conference on Signal Processing*, Beijing, China, 2006.

[27] LLOYD, S. P. "Least Squares Quantization in PCM". In: *IRE Transactions on Information Theory*, v. 28, pp. 129–137, 1982.

[28] "Test Model under Consideration for HEVC based 3D video coding". 2012. International Organisation for Standardisation.

[29] SULLIVAN, G. J., WIEGAND, T. "Rate-Distortion Optimization of Video Compression", *IEEE Signal Processing Magazine*, v. 15, pp. 74–90, November 1988.

[30] SAKRISON, D. J. "A Geometric Treatment of the Source Encoding of a Gaussian Random Variable", *IEEE Transactions on Information Theory*, v. 14, n. 3, pp. 481–486, maio 1968.

[31] FISCHER, T. R. "A Pyramid Vector Quantizer", *IEEE Transactions on Information Theory*, v. 32, n. 4, pp. 568–583, 1986.

[32] CHEN, F., GAO, Z., VILLASENOR, J. "Lattice Vector Quantization of Generalized Gaussian Sources", *IEEE Transactions on Information Theory*, v. 43, n. 1, pp. 92–103, 1997.

[33] DA SILVA PINAGÉ, F. *Avaliação do Desempenho de Algoritmos de Compressão de Imagens usando Recorrência de Padrões Multiescalas*. Dissertação de mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2005.

[34] LUCAS, L. F. R., RODRIGUES, N. M. M., DA SILVA, E. A. B., et al. "Adaptive Least Squares Prediction for Stereo Image Coding". In: *International Conference on Image Processing*, pp. 2013–2016, 2011.

[35] "MPEG-FTV Test Sequence, Nagoya University". Website. `http://www.tanimoto.nuee.nagoya-u.ac.jp/~fukushima/mpegftv/`. Acessado em agosto de 2013.

[36] HO, Y.-S., LEE, E.-K., LEE, C. "Multiview Video Test Sequence and Camera Parameters". Website. `http://vclab.gist.ac.kr/papers/03/2008/`. Acessado em agosto de 2013.

[37] "Poznań Multiview Video Test Sequences and Camera Parameters". Website. `http://new.owieczka.net/en/2013/01/21/`

poznan-3d-video-test-sequences-with-depth-maps/. Acessado em agosto de 2013.

Apêndice A

Algumas Demonstrações

A.1 Exemplo numérico da transformação de escala (função linear)

Supondo que tenhamos um bloco aproximado de resíduo da escala l, de dimensões $M \times N$. Para $l_o = 15$, $M = 8$ e $N = 8$. Para uma dada função de aproximação $f(x, y)$, temos:

$$\begin{aligned} 0 \leq x \leq 7 \text{ e } 0 \leq y \leq 7 \\ f(x, y) = 3x + 7y + 11 \end{aligned} \tag{A.1}$$

Vamos realizar a transformação para a escala de destino l, de dimensões $M' \times N'$. Supondo $l = 8$, $M' = 4$ e $N' = 4$.

Usando o desenvolvimento da seção 5.5.1, temos: $\kappa = \frac{(8-1)}{(4-1)}$, $\gamma = \frac{(8-1)}{(4-1)}$. Portanto, $g(x', y')$ é dada por:

$$\begin{aligned} 0 \leq x' \leq 3 \text{ e } 0 \leq y' \leq 3 \\ g(x', y') = \left(\frac{7}{3}\right) 3x' + \left(\frac{7}{3}\right) 7y' + 11 \end{aligned} \tag{A.2}$$

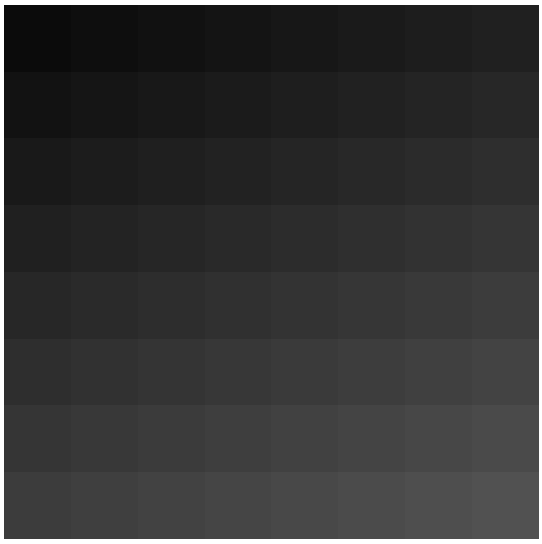
A Figura A.1 exibe os resultados numéricos para ambas funções. Como podemos verificar, o conjunto imagem está confinado no intervalo $[11, 81]$ antes e depois da transformação. As amostras do bloco escalado foram arredondadas.

11	14	17	20	23	26	29	32
18	21	24	27	30	33	36	39
25	28	31	34	37	40	43	46
32	35	38	41	44	47	50	53
39	42	45	48	51	54	57	60
46	49	52	55	58	61	64	67
53	56	59	62	65	68	71	74
60	63	66	69	72	75	78	81

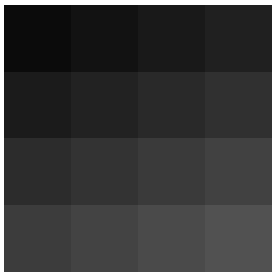
$\Longrightarrow$

11	18	25	32
27	34	41	48
44	51	58	65
60	67	74	81

(a)



(b) 8×8



(c) 4×4

Figura A.1: Transformação de escala

A.2 Transformação de escala para bloco aproximado com função quadrática

Seja $\mathbf{R}^{l_o}$ um bloco de resíduo da escala l de origem, de dimensões $M \times N$. O resíduo é aproximado pela função quadrática:

$$0 \leq x \leq M-1 \text{ e } 0 \leq y \leq N-1$$
$$h(x,y) = \alpha_0 x^2 + \alpha_1 y^2 + \alpha_2 x + \alpha_3 y + \alpha_4 xy + \alpha_5 \tag{A.3}$$

Seja $q(x', y')$ a função de aproximação do bloco $\mathbf{R}^{l_o}$ da escala de origem l_o representado na escala de destino l, de dimensões $M' \times N'$.

$$0 \leq x' \leq M'-1 \text{ e } 0 \leq y' \leq N'-1$$
$$q(x',y') = \alpha_0{}' x'^2 + \alpha_1{}' y'^2 + \alpha_2{}' x' + \alpha_3{}' y' + \alpha_4{}' x' y' + \alpha_5{}' \tag{A.4}$$

O domínio de $q(x', y')$ pode ser escrito como uma versão escala do domínio de $h(x, y)$ como:

$$0 \leq x' \leq \frac{M-1}{\kappa} \text{ e } 0 \leq y' \leq \frac{N-1}{\gamma} \tag{A.5}$$

Portanto, $\kappa = \frac{M-1}{M'-1}$ e $\gamma = \frac{N-1}{N'-1}$. Podemos escrever ainda, $x = x'\kappa$ e $y = y'\gamma$.

Admitindo que o conjunto imagem da função $h(x, y)$ seja igual ao conjunto imagem da função $q(x', y')$, e substituindo x e y como definido anteriormente, temos:

$$h(x, y) = q(x', y')$$

$$\alpha_0 x^2 + \alpha_1 y^2 + \alpha_2 x + \alpha_3 y + \alpha_4 xy + \alpha_5 =$$
$$\alpha_0{}' x'^2 + \alpha_1{}' y'^2 + \alpha_2{}' x' + \alpha_3{}' y' + \alpha_4{}' x'y' + \alpha_5{}' \therefore$$
$$\alpha_0 \kappa^2 x'^2 + \alpha_1 \gamma^2 y'^2 + \alpha_2 \kappa x' + \alpha_3 \gamma y' + \alpha_4 \kappa\gamma x'y' + \alpha_5 =$$
$$\alpha_0{}' x'^2 + \alpha_1{}' y'^2 + \alpha_2{}' x' + \alpha_3{}' y' + \alpha_4{}' x'y' + \alpha_5{}' \therefore$$
$$(\alpha_0\kappa^2 - \alpha_0{}')x'^2 + (\alpha_1\gamma^2 - \alpha_1{}')y'^2 + (\alpha_2\kappa - \alpha_2{}')x' + (\alpha_3\gamma - \alpha_3{}')y' + (\alpha_4\kappa\gamma - \alpha_4{}')x'y' + (\alpha_5 - \alpha_5{}') = 0$$

Podemos escrever então, $\alpha_0{}' = \alpha_0\kappa^2$, $\alpha_1{}' = \alpha_1\gamma^2$, $\alpha_2{}' = \alpha_2\kappa^2$, $\alpha_3{}' = \alpha_3\gamma^2$, $\alpha_4{}' = \alpha_4\kappa\gamma$ e $\alpha_5{}' = \alpha_5$.

A.3 Níveis de reconstrução dos quantizadores utilizados

Níveis de reconstrução dos coeficientes das funções constante, linear e quadrática utilizadas na aproximação do resíduo, definidas nas equações 4.1, 3.1 e 4.2, respectivamente.

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-32 -10 0 10 32
15 níveis	-79 -50 -32 -21 -13 -7 -3 0 3 7 13 21 32 50 79
21 níveis	-92 -65 -47 -35 -25 -18 -13 -8 -5 -2 0 2 5 8 13 18 25 35 47 65 92
47 níveis	-111 -93 -79 -68 -58 -51 -44 -39 -34 -29 -25 -22 -19 -16 -13 -11 -9 -7 -6 -4 -3 -2 -1 0 1 2 3 4 6 7 9 11 13 16 19 22 25 29 34 39 44 51 58 68 79 93 111
61 níveis	-114 -99 -87 -77 -69 -62 -55 -50 -45 -40 -36 -32 -29 -26 -23 -21 -18 -16 -14 -12 -11 -9 -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 8 9 11 12 14 16 18 21 23 26 29 32 36 40 45 50 55 62 69 77 87 99 114
81 níveis	-111 -100 -91 -83 -77 -70 -65 -60 -55 -51 -47 -44 -40 -37 -35 -32 -29 -27 -25 -23 -21 -19 -17 -16 -14 -13 -12 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 12 13 14 16 17 19 21 23 25 27 29 32 35 37 40 44 47 51 55 60 65 70 77 83 91 100 111

Tabela A.1: Níveis de reconstrução do coeficiente α_0 da função constante

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-32 -10 0 10 32
15 níveis	-76 -48 -31 -20 -13 -7 -3 0 3 7 13 20 31 48 76
21 níveis	-89 -62 -45 -33 -24 -18 -12 -8 -5 -2 0 2 5 8 12 18 24 33 45 62 89
47 níveis	-107 -89 -75 -64 -55 -48 -42 -37 -32 -28 -24 -21 -18 -15 -13 -11 -9 -7 -6 -4 -3 -2 -1 0 1 2 3 4 6 7 9 11 13 15 18 21 24 28 32 37 42 48 55 64 75 89 107
61 níveis	108 -94 -82 -73 -65 -58 -52 -47 -42 -38 -34 -31 -27 -25 -22 -20 -18 -16 -14 -12 -10 -9 -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 8 9 10 12 14 16 18 20 22 25 27 31 34 38 42 47 52 58 65 73 82 94 108
81 níveis	-111 -100 -91 -83 -76 -70 -64 -59 -54 -50 -46 -43 -39 -36 -34 -31 -29 -26 -24 -22 -20 -19 -17 -16 -14 -13 -12 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 12 13 14 16 17 19 20 22 24 26 29 31 34 36 39 43 46 50 54 59 64 70 76 83 91 100 111

Tabela A.2: Níveis de reconstrução do coeficiente α_0 da função linear

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-30 -9 0 9 30
15 níveis	-73 -46 -31 -20 -12 -7 -3 0 3 7 12 20 31 46 73
21 níveis	-86 -60 -44 -32 -24 -17 -12 -8 -5 -2 0 2 5 8 12 17 24 32 44 60 86
47 níveis	-104 -86 -73 -63 -54 -47 -41 -36 -31 -27 -24 -21 -18 -15 -13 -11 -9 -7 -6 -4 -3 -2 -1 0 1 2 3 4 6 7 9 11 13 15 18 21 24 27 31 36 41 47 54 63 73 86 104
61 níveis	-106 -91 -80 -71 -63 -57 -51 -46 -41 -37 -34 -30 -27 -24 -22 -20 -18 -16 -14 -12 -10 -9 -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 8 9 10 12 14 16 18 20 22 24 27 30 34 37 41 46 51 57 63 71 80 91 106
81 níveis	-107 -97 -88 -80 -73 -67 -62 -57 -53 -49 -45 -41 -38 -35 -33 -30 -28 -26 -24 -22 -20 -18 -17 -15 -14 -13 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 13 14 15 17 18 20 22 24 26 28 30 33 35 38 41 45 49 53 57 62 67 73 80 88 97 107

Tabela A.3: Níveis de reconstrução do coeficiente α_1 da função linear

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-33 -10 0 10 33
15 níveis	-80 -50 -33 -21 -13 -7 -3 0 3 7 13 21 33 50 80
21 níveis	-92 -64 -47 -34 -25 -18 -13 -8 -5 -2 0 2 5 8 13 18 25 34 47 64 92
47 níveis	-110 -92 -78 -67 -58 -50 -44 -38 -33 -29 -25 -22 -19 -16 -13 -11 -9 -7 -6 -4 -3 -2 -1 0 1 2 3 4 6 7 9 11 13 16 19 22 25 29 33 38 44 50 58 67 78 92 110
61 níveis	-113 -98 -86 -77 -69 -61 -55 -50 -45 -40 -36 -32 -29 -26 -23 -21 -18 -16 -14 -12 -11 -9 -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 8 9 11 12 14 16 18 21 23 26 29 32 36 40 45 50 55 61 69 77 86 98 113
81 níveis	-111 -100 -91 -83 -77 -70 -65 -60 -56 -52 -48 -44 -41 -38 -35 -32 -30 -28 -26 -23 -22 -20 -18 -17 -15 -14 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 14 15 17 18 20 22 23 26 28 30 32 35 38 41 44 48 52 56 60 65 70 77 83 91 100 111

Tabela A.4: Níveis de reconstrução do coeficiente α_2 da função linear

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-50 -18 0 18 50
15 níveis	-114 -73 -48 -32 -20 -12 -5 0 5 12 20 32 48 73 114
21 níveis	-132 -92 -67 -50 -36 -26 -18 -12 -7 -3 0 3 7 12 18 26 36 50 67 92 132
47 níveis	-161 -133 -113 -97 -84 -73 -64 -56 -48 -42 -37 -32 -27 -23 -20 -16 -14 -11 -9 -6 -5 -3 -1 0 1 3 5 6 9 11 14 16 20 23 27 32 37 42 48 56 64 73 84 97 113 133 161
61 níveis	-170 -150 -134 -120 -107 -96 -86 -77 -69 -62 -56 -50 -45 -41 -36 -32 -29 -25 -22 -19 -17 -15 -12 -10 -8 -7 -5 -4 -2 -1 0 1 2 4 5 7 8 10 12 15 17 19 22 25 29 32 36 41 45 50 56 62 69 77 86 96 107 120 134 150 170
81 níveis	-165 -150 -137 -126 -116 -106 -98 -91 -84 -78 -72 -66 -62 -57 -53 -49 -45 -42 -38 -35 -33 -30 -27 -25 -23 -21 -18 -17 -15 -13 -11 -10 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 10 11 13 15 17 18 21 23 25 27 30 33 35 38 42 45 49 53 57 62 66 72 78 84 91 98 106 116 126 137 150 165

Tabela A.5: Níveis de reconstrução do coeficiente α_0 da função quadrática

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-47 -16 0 16 47
15 níveis	-107 -68 -44 -29 -18 -10 -4 0 4 10 18 29 44 68 107
21 níveis	-127 -89 -66 -49 -36 -26 -19 -12 -8 -3 0 3 8 12 19 26 36 49 66 89 127
47 níveis	-154 -128 -109 -94 -81 -71 -62 -54 -47 -41 -35 -31 -26 -23 -19 -16 -13 -11 -8 -6 -4 -3 -1 0 1 3 4 6 8 11 13 16 19 23 26 31 35 41 47 54 62 71 81 94 109 128 154
61 níveis	-162 -142 -125 -112 -100 -89 -80 -72 -65 -59 -53 -48 -43 -38 -34 -31 -27 -24 -21 -19 -16 -14 -12 -10 -8 -6 -5 -3 -2 -1 0 1 2 3 5 6 8 10 12 14 16 19 21 24 27 31 34 38 43 48 53 59 65 72 80 89 100 112 125 142 162
81 níveis	-159 -145 -133 -123 -115 -106 -99 -92 -85 -79 -74 -68 -63 -59 -54 -50 -46 -43 -39 -36 -33 -30 -28 -25 -23 -21 -19 -17 -15 -13 -12 -10 -9 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 9 10 12 13 15 17 19 21 23 25 28 30 33 36 39 43 46 50 54 59 63 68 74 79 85 92 99 106 115 123 133 145 159

Tabela A.6: Níveis de reconstrução do coeficiente α_1 da função quadrática

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-85 -30 0 30 85
15 níveis	-208 -129 -84 -55 -36 -21 -9 0 9 21 36 55 84 129 208
21 níveis	-228 -155 -110 -79 -57 -40 -28 -19 -10 -4 0 4 10 19 28 40 57 79 110 155 228
47 níveis	-279 -236 -201 -172 -146 -124 -106 -91 -78 -67 -58 -49 -42 -35 -29 -24 -20 -16 -12 -9 -6 -4 -2 0 2 4 6 9 12 16 20 24 29 35 42 49 58 67 78 91 106 124 146 172 201 236 279
61 níveis	-285 -256 -231 -208 -189 -171 -154 -139 -125 -113 -101 -91 -82 -74 -66 -59 -52 -46 -40 -35 -31 -27 -23 -20 -16 -13 -10 -7 -4 -2 0 2 4 7 10 13 16 20 23 27 31 35 40 46 52 59 66 74 82 91 101 113 125 139 154 171 189 208 231 256 285
81 níveis	-292 -283 -272 -259 -243 -228 -214 -199 -183 -168 -155 -143 -132 -121 -110 -101 -91 -83 -75 -68 -62 -56 -50 -45 -40 -36 -32 -28 -25 -21 -19 -16 -13 -11 -9 -7 -5 -3 -2 -1 0 1 2 3 5 7 9 11 13 16 19 21 25 28 32 36 40 45 50 56 62 68 75 83 91 101 110 121 132 143 155 168 183 199 214 228 243 259 272 283 292

Tabela A.7: Níveis de reconstrução do coeficiente α_2 da função quadrática

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-94 -35 0 35 94
15 níveis	-179 -112 -73 -48 -31 -18 -8 0 8 18 31 48 73 112 179
21 níveis	-197 -135 -96 -70 -51 -37 -27 -18 -10 -5 0 5 10 18 27 37 51 70 96 135 197
47 níveis	-252 -214 -182 -156 -134 -116 -101 -88 -76 -66 -57 -49 -42 -36 -30 -25 -21 -17 -14 -10 -7 -4 -2 0 2 4 7 10 14 17 21 25 30 36 42 49 57 66 76 88 101 116 134 156 182 214 252
61 níveis	-251 -223 -197 -175 -157 -140 -124 -111 -99 -88 -78 -69 -61 -54 -48 -42 -37 -32 -28 -24 -21 -18 -15 -12 -9 -7 -6 -4 -2 -1 0 1 2 4 6 7 9 12 15 18 21 24 28 32 37 42 48 54 61 69 78 88 99 111 124 140 157 175 197 223 251
81 níveis	-259 -249 -237 -224 -211 -197 -184 -172 -160 -147 -136 -125 -114 -105 -97 -88 -81 -73 -67 -60 -55 -49 -45 -40 -36 -32 -28 -25 -22 -19 -16 -14 -12 -10 -8 -6 -4 -3 -2 -1 0 1 2 3 4 6 8 10 12 14 16 19 22 25 28 32 36 40 45 49 55 60 67 73 81 88 97 105 114 125 136 147 160 172 184 197 211 224 237 249 259

Tabela A.8: Níveis de reconstrução do coeficiente α_3 da função quadrática

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-30 -10 0 10 30
15 níveis	-68 -44 -29 -20 -13 -7 -3 0 3 7 13 20 29 44 68
21 níveis	-79 -56 -41 -31 -23 -17 -12 -8 -5 -2 0 2 5 8 12 17 23 31 41 56 79
47 níveis	-95 -79 -68 -59 -51 -44 -39 -34 -30 -26 -23 -20 -17 -15 -13 -11 -9 -7 -6 -4 -3 -2 -1 0 1 2 3 4 6 7 9 11 13 15 17 20 23 26 30 34 39 44 51 59 68 79 95
61 níveis	-100 -88 -78 -70 -63 -56 -51 -46 -42 -38 -34 -31 -28 -25 -22 -20 -18 -16 -14 -12 -11 -9 -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 8 9 11 12 14 16 18 20 22 25 28 31 34 38 42 46 51 56 63 70 78 88 100
81 níveis	-98 -88 -80 -74 -68 -63 -58 -53 -50 -46 -43 -40 -37 -34 -32 -29 -27 -25 -23 -22 -20 -18 -17 -15 -14 -13 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 13 14 15 17 18 20 22 23 25 27 29 32 34 37 40 43 46 50 53 58 63 68 74 80 88 98

Tabela A.9: Níveis de reconstrução do coeficiente α_4 da função quadrática

Quantidade de níveis de reconstrução	Níveis de reconstrução
5 níveis	-35 -12 0 12 35
15 níveis	-82 -52 -34 -23 -14 -8 -4 0 4 8 14 23 34 52 82
21 níveis	-94 -66 -48 -36 -26 -19 -13 -9 -5 -2 0 2 5 9 13 19 26 36 48 66 94
47 níveis	-114 -95 -81 -70 -60 -53 -46 -40 -35 -31 -27 -23 -20 -17 -14 -12 -10 -8 -6 -5 -3 -2 -1 0 1 2 3 5 6 8 10 12 14 17 20 23 27 31 35 40 46 53 60 70 81 95 114
61 níveis	-117 -103 -91 -81 -72 -65 -58 -52 -47 -43 -39 -35 -31 -28 -25 -22 -20 -18 -15 -14 -12 -10 -9 -7 -6 -5 -3 -2 -1 0 1 2 3 5 6 7 9 10 12 14 15 18 20 22 25 28 31 35 39 43 47 52 58 65 72 81 91 103 117
81 níveis	-118 -110 -102 -93 -86 -79 -73 -68 -63 -58 -54 -50 -46 -43 -40 -37 -34 -31 -29 -26 -24 -22 -20 -19 -17 -15 -14 -12 -11 -10 -9 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 9 10 11 12 14 15 17 19 20 22 24 26 29 31 34 37 40 43 46 50 54 58 63 68 73 79 86 93 102 110 118

Tabela A.10: Níveis de reconstrução do coeficiente α_5 da função quadrática

Apêndice B

Imagens Originais Utilizadas nas Simulações

B.1 Imagens de textura

Foram usadas imagens suaves (naturais), não suaves (imagens de textos), imagens geradas por computador e compostas. Esse conjunto permite avaliar o desempenho do algoritmo em várias classes de imagens.

Figura B.1: lena (512×512 *pixels*)

Figura B.2: barb (720×576 *pixels*)

Figura B.3: zelda (720×576 *pixels*)

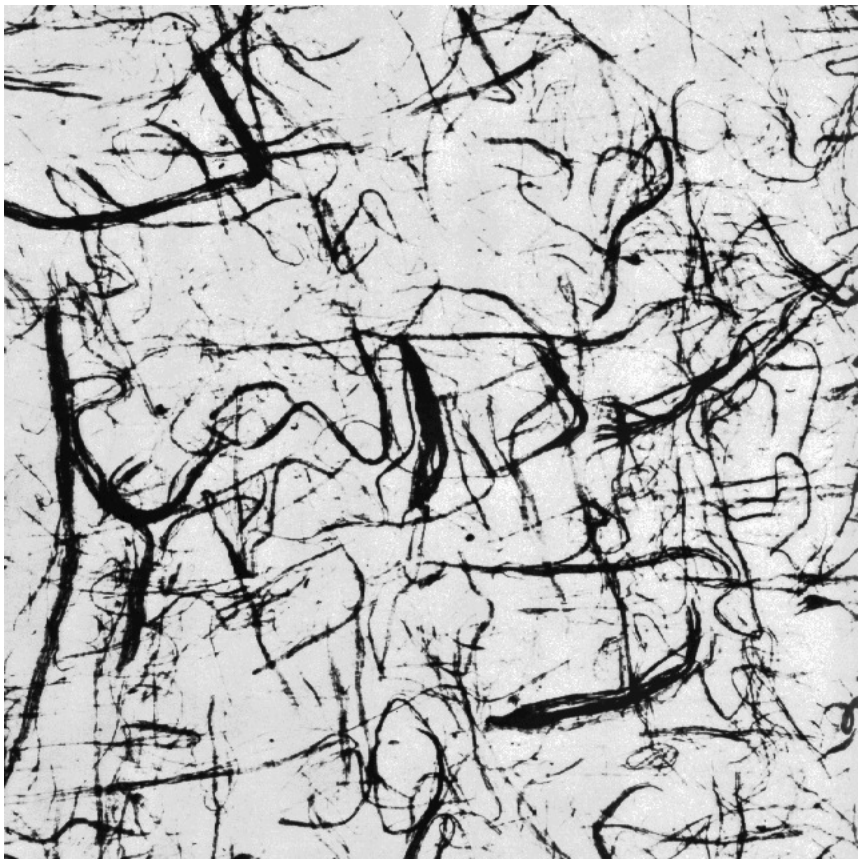
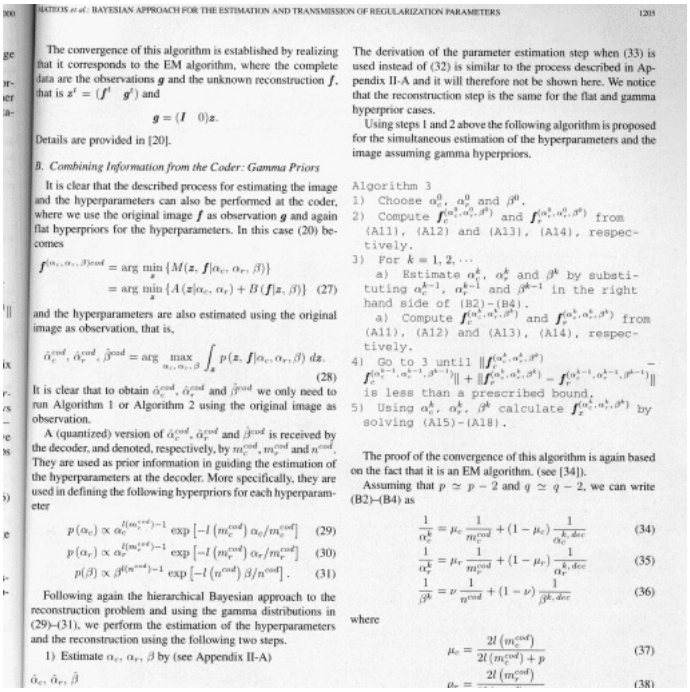


Figura B.4: D108 (640×640 *pixels*)



MATEOS *et al.*: BAYESIAN APPROACH FOR THE ESTIMATION AND TRANSMISSION OF REGULARIZATION PARAMETERS 1205

The convergence of this algorithm is established by realizing that it corresponds to the EM algorithm, where the complete data are the observations $\boldsymbol{g}$ and the unknown reconstruction $\boldsymbol{f}$, that is $\boldsymbol{z}^t = (\boldsymbol{f}^t \quad \boldsymbol{g}^t)$ and

$$\boldsymbol{g} = (I \quad 0)\boldsymbol{z}.$$

Details are provided in [20].

B. Combining Information from the Coder: Gamma Priors

It is clear that the described process for estimating the image and the hyperparameters can also be performed at the coder, where we use the original image $\boldsymbol{f}$ as observation $\boldsymbol{g}$ and again flat hyperpriors for the hyperparameters. In this case (20) becomes

$$\begin{aligned} \boldsymbol{f}^{(\alpha_\epsilon,\alpha_r,\beta)^{cod}} &= \arg\min_{\boldsymbol{z}} \{M(\boldsymbol{z}, \boldsymbol{f}|\alpha_\epsilon, \alpha_r, \beta)\} \\ &= \arg\min_{\boldsymbol{z}} \{A(\boldsymbol{z}|\alpha_\epsilon, \alpha_r) + B(\boldsymbol{f}|\boldsymbol{z}, \beta)\} \quad (27) \end{aligned}$$

and the hyperparameters are also estimated using the original image as observation, that is,

$$\hat{\alpha}_\epsilon^{cod}, \hat{\alpha}_r^{cod}, \hat{\beta}^{cod} = \arg\max_{\alpha_\epsilon,\alpha_r,\beta} \int_{\boldsymbol{z}} p(\boldsymbol{z}, \boldsymbol{f}|\alpha_\epsilon, \alpha_r, \beta)\, d\boldsymbol{z}. \quad (28)$$

It is clear that to obtain $\hat{\alpha}_\epsilon^{cod}$, $\hat{\alpha}_r^{cod}$ and $\hat{\beta}^{cod}$ we only need to run Algorithm 1 or Algorithm 2 using the original image as observation.

A (quantized) version of $\hat{\alpha}_\epsilon^{cod}$, $\hat{\alpha}_r^{cod}$ and $\hat{\beta}^{cod}$ is received by the decoder, and denoted, respectively, by m_ϵ^{cod}, m_r^{cod} and n^{cod}. They are used as prior information in guiding the estimation of the hyperparameters at the decoder. More specifically, they are used in defining the following hyperpriors for each hyperparameter

$$p(\alpha_\epsilon) \propto \alpha_\epsilon^{l(m_\epsilon^{cod})-1} \exp\left[-l\left(m_\epsilon^{cod}\right)\alpha_\epsilon/m_\epsilon^{cod}\right] \quad (29)$$

$$p(\alpha_r) \propto \alpha_r^{l(m_r^{cod})-1} \exp\left[-l\left(m_r^{cod}\right)\alpha_r/m_r^{cod}\right] \quad (30)$$

$$p(\beta) \propto \beta^{l(n^{cod})-1} \exp\left[-l\left(n^{cod}\right)\beta/n^{cod}\right]. \quad (31)$$

Following again the hierarchical Bayesian approach to the reconstruction problem and using the gamma distributions in (29)–(31), we perform the estimation of the hyperparameters and the reconstruction using the following two steps.

1) Estimate α_ϵ, α_r, β by (see Appendix II-A)

$$\hat{\alpha}_\epsilon, \hat{\alpha}_r, \hat{\beta}$$

The derivation of the parameter estimation step when (33) is used instead of (32) is similar to the process described in Appendix II-A and it will therefore not be shown here. We notice that the reconstruction step is the same for the flat and gamma hyperprior cases.

Using steps 1 and 2 above the following algorithm is proposed for the simultaneous estimation of the hyperparameters and the image assuming gamma hyperpriors.

Algorithm 3

1) Choose α_ϵ^0, α_r^0 and β^0.
2) Compute $\boldsymbol{f}_\epsilon^{(\alpha_\epsilon^0,\alpha_r^0,\beta^0)}$ and $\boldsymbol{f}_r^{(\alpha_\epsilon^0,\alpha_r^0,\beta^0)}$ from (A11), (A12) and (A13), (A14), respectively.
3) For $k = 1, 2, \cdots$
   a) Estimate α_ϵ^k, α_r^k and β^k by substituting α_ϵ^{k-1}, α_r^{k-1} and β^{k-1} in the right hand side of (B2)–(B4).
   a) Compute $\boldsymbol{f}_\epsilon^{(\alpha_\epsilon^k,\alpha_r^k,\beta^k)}$ and $\boldsymbol{f}_r^{(\alpha_\epsilon^k,\alpha_r^k,\beta^k)}$ from (A11), (A12) and (A13), (A14), respectively.
4) Go to 3 until $\|\boldsymbol{f}_\epsilon^{(\alpha_\epsilon^k,\alpha_r^k,\beta^k)} - \boldsymbol{f}_\epsilon^{(\alpha_\epsilon^{k-1},\alpha_r^{k-1},\beta^{k-1})}\| + \|\boldsymbol{f}_r^{(\alpha_\epsilon^k,\alpha_r^k,\beta^k)} - \boldsymbol{f}_r^{(\alpha_\epsilon^{k-1},\alpha_r^{k-1},\beta^{k-1})}\|$ is less than a prescribed bound.
5) Using α_ϵ^k, α_r^k, β^k calculate $\boldsymbol{f}_x^{(\alpha_\epsilon^k,\alpha_r^k,\beta^k)}$ by solving (A15)–(A18).

The proof of the convergence of this algorithm is again based on the fact that it is an EM algorithm. (see [34]).

Assuming that $p \simeq p-2$ and $q \simeq q-2$, we can write (B2)–(B4) as

$$\frac{1}{\alpha_\epsilon^k} = \mu_\epsilon \frac{1}{m_\epsilon^{cod}} + (1-\mu_\epsilon)\frac{1}{\alpha_\epsilon^{k,dec}} \quad (34)$$

$$\frac{1}{\alpha_r^k} = \mu_r \frac{1}{m_r^{cod}} + (1-\mu_r)\frac{1}{\alpha_r^{k,dec}} \quad (35)$$

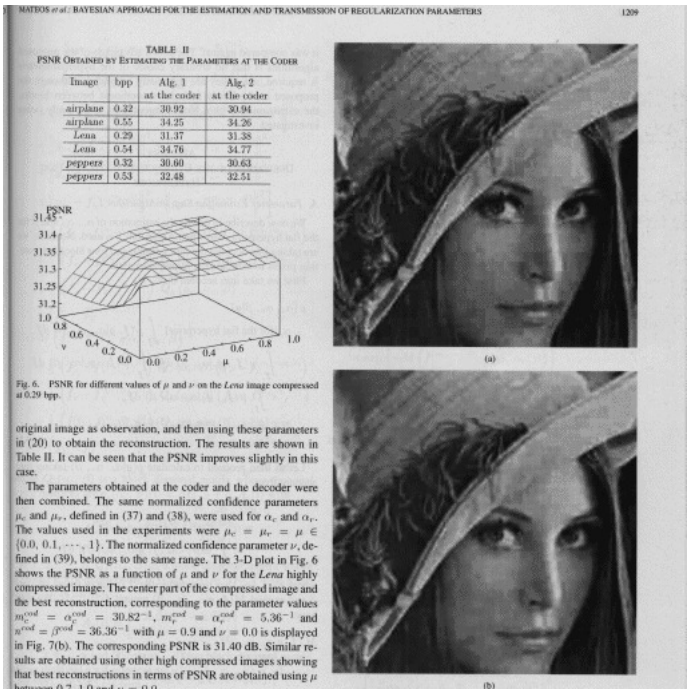
$$\frac{1}{\beta^k} = \nu \frac{1}{n^{cod}} + (1-\nu)\frac{1}{\beta^{k,dec}} \quad (36)$$

where

$$\mu_\epsilon = \frac{2l\left(m_\epsilon^{cod}\right)}{2l\left(m_\epsilon^{cod}\right)+p} \quad (37)$$

$$\mu_r = \frac{2l\left(m_r^{cod}\right)}{\quad} \quad (38)$$

Figura B.5: pp1205 (512×512 *pixels*)



MATEOS *et al.*: BAYESIAN APPROACH FOR THE ESTIMATION AND TRANSMISSION OF REGULARIZATION PARAMETERS 1209

TABLE II
PSNR OBTAINED BY ESTIMATING THE PARAMETERS AT THE CODER

Image	bpp	Alg. 1 at the coder	Alg. 2 at the coder
airplane	0.32	30.92	30.94
airplane	0.55	34.25	34.26
Lena	0.29	31.37	31.38
Lena	0.54	34.76	34.77
peppers	0.32	30.60	30.63
peppers	0.53	32.48	32.51

Fig. 6. PSNR for different values of μ and ν on the *Lena* image compressed at 0.29 bpp.

original image as observation, and then using these parameters in (20) to obtain the reconstruction. The results are shown in Table II. It can be seen that the PSNR improves slightly in this case.

The parameters obtained at the coder and the decoder were then combined. The same normalized confidence parameters μ_c and μ_r, defined in (37) and (38), were used for α_c and α_r. The values used in the experiments were $\mu_c = \mu_r = \mu \in \{0.0, 0.1, \cdots, 1\}$. The normalized confidence parameter ν, defined in (39), belongs to the same range. The 3-D plot in Fig. 6 shows the PSNR as a function of μ and ν for the *Lena* highly compressed image. The center part of the compressed image and the best reconstruction, corresponding to the parameter values $m_c^{cod} = \alpha_c^{cod} = 30.82^{-1}$, $m_r^{cod} = \alpha_r^{cod} = 5.36^{-1}$ and $n^{cod} = \beta^{cod} = 36.36^{-1}$ with $\mu = 0.9$ and $\nu = 0.0$ is displayed in Fig. 7(b). The corresponding PSNR is 31.40 dB. Similar results are obtained using other high compressed images showing that best reconstructions in terms of PSNR are obtained using μ

(a)

(b)

Figura B.6: pp1209 (512×512 *pixels*)

B.2 Imagens de mapas de profundidade

Para a codificação de mapas de profundidade foram utilizados apenas o primeiro quadro das sequências *Balloons*, *Kendo* [35], *Newspaper* [36], *Poznanhall2* e *Poznanstreet* [37]. As sequências *Balloons*, *Kendo* e *Newspaper* têm resolução de 1024×768 *pixels*, enquanto que as sequências *Poznanhall2* e *Poznanstreet* têm 1920×1088 *pixels*. Abaixo são exibidos apenas os mapas do primeiro quadro de cada sequência utilizada.

(a) Mapa de profundidades (b) Textura

Figura B.7: *Balloons* câmera 1

(a) Mapa de profundidades (b) Textura

Figura B.8: *Balloons* câmera 5

(a) Mapa de profundidades (b) Textura

Figura B.9: *Kendo* câmera 1

(a) Mapa de profundidades (b) Textura

Figura B.10: *Kendo* câmera 5

(a) Mapa de profundidades (b) Textura

Figura B.11: *Newspaper* câmera 2

(a) Mapa de profundidades (b) Textura

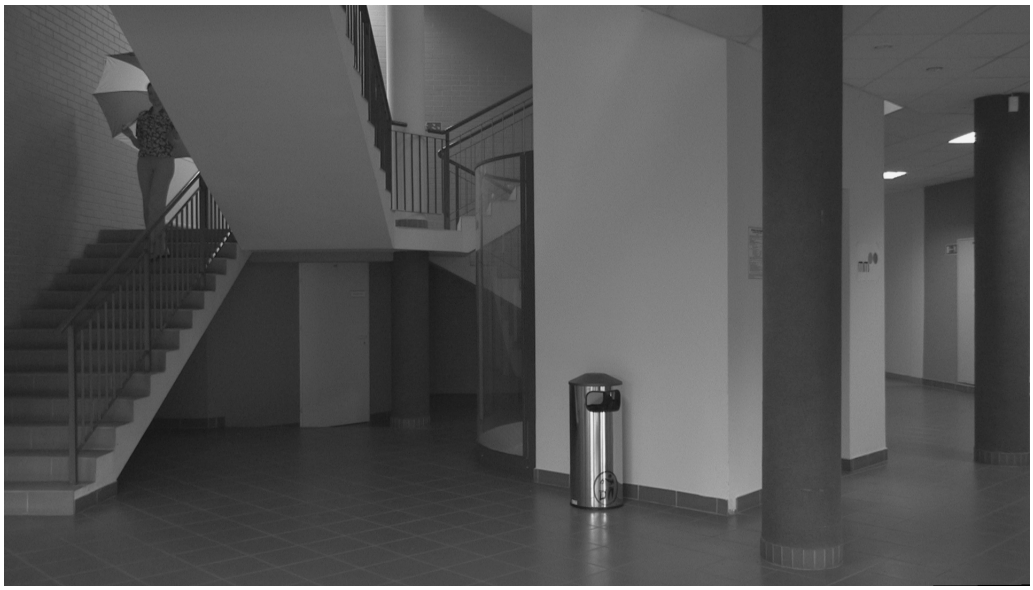
Figura B.12: *Newspaper* câmera 6

(a) Mapa de profundidades

(b) Textura

Figura B.13: *Poznanhall2* câmera 5

(a) Mapa de profundidades



(b) Textura

Figura B.14: *Poznanhall2* câmera 7

(a) Mapa de profundidades

(b) Textura

Figura B.15: *Poznanstreet* câmera 3

(a) Mapa de profundidades

(b) Textura

Figura B.16: *Poznanstreet* câmera 5

B.3 Imagens de textura utilizadas no treinamento para obtenção dos quantizadores

Figura B.17: gold (720×576 *pixels*)

Figura B.18: f16 (512×512 *pixels*)

Figura B.19: bridge (512×512 *pixels*)

Figura B.20: aerial (512×512 *pixels*)

Figura B.21: peppers (512×512 *pixels*)

Figura B.22: boats (720×576 *pixels*)

Apêndice C

Resultados Experimentais Complementares

C.1 Resultados Complementares (Mapas)

C.1.1 Efeito da função de aproximação do resíduo

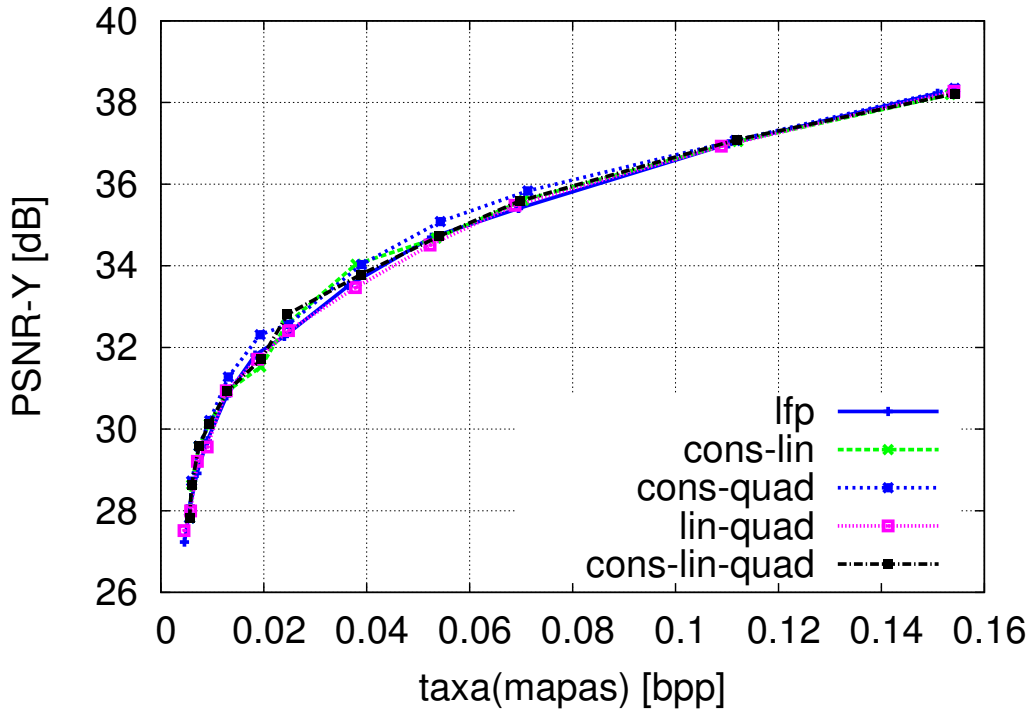


Figura C.1: Desempenho taxa-distorção para diversas funções de aproximação (*Newspaper* câmera 4)

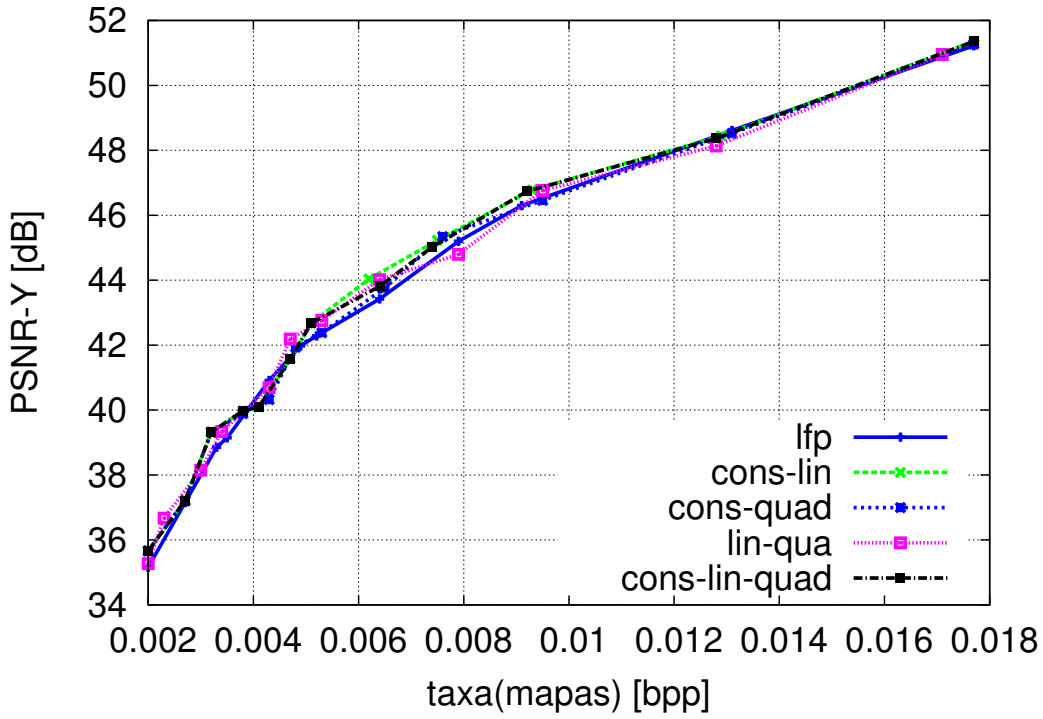


Figura C.2: Desempenho taxa-distorção para diversas funções de aproximação (*PoznanHall2* câmera 6)

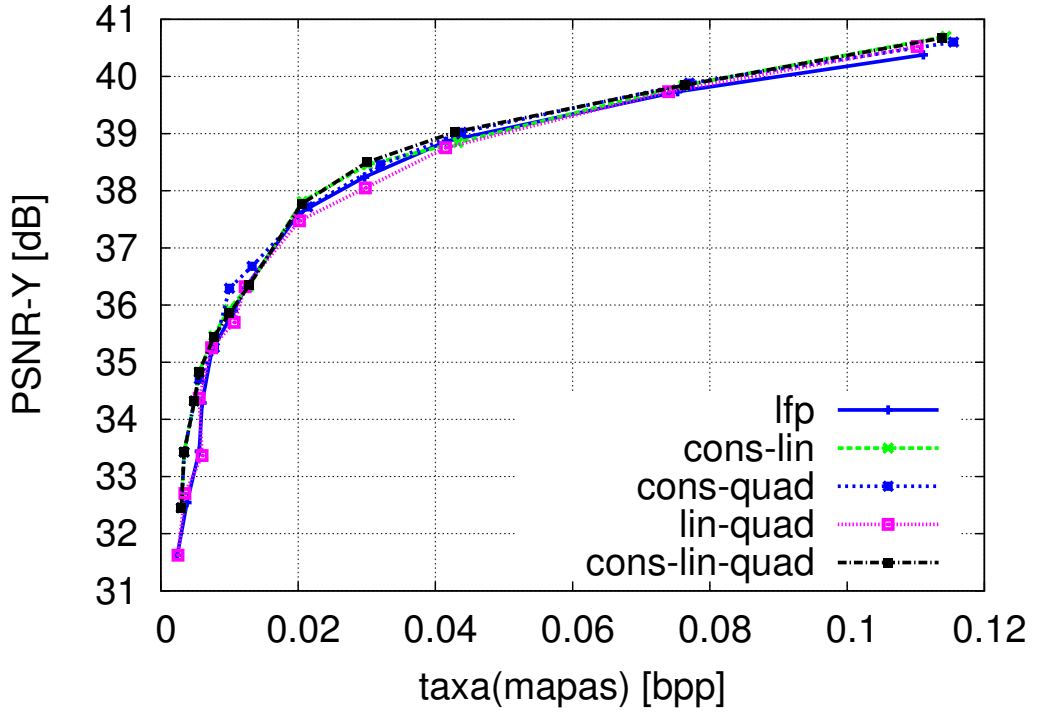


Figura C.3: Desempenho taxa-distorção para diversas funções de aproximação (*PoznanStreet* câmera 4)

C.1.2 Efeito da quantização dos coeficientes

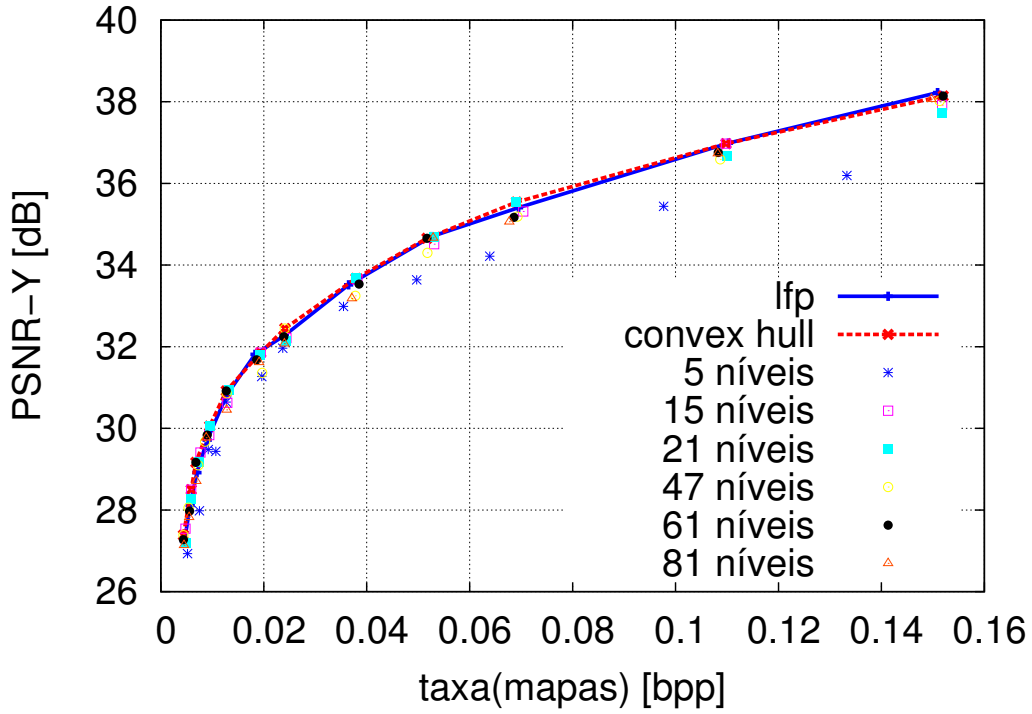


Figura C.4: Efeito da quantização dos coeficientes de aproximação na qualidade das vistas reconstruídas usando os mapas codificados (*Newspaper* câmera 4)

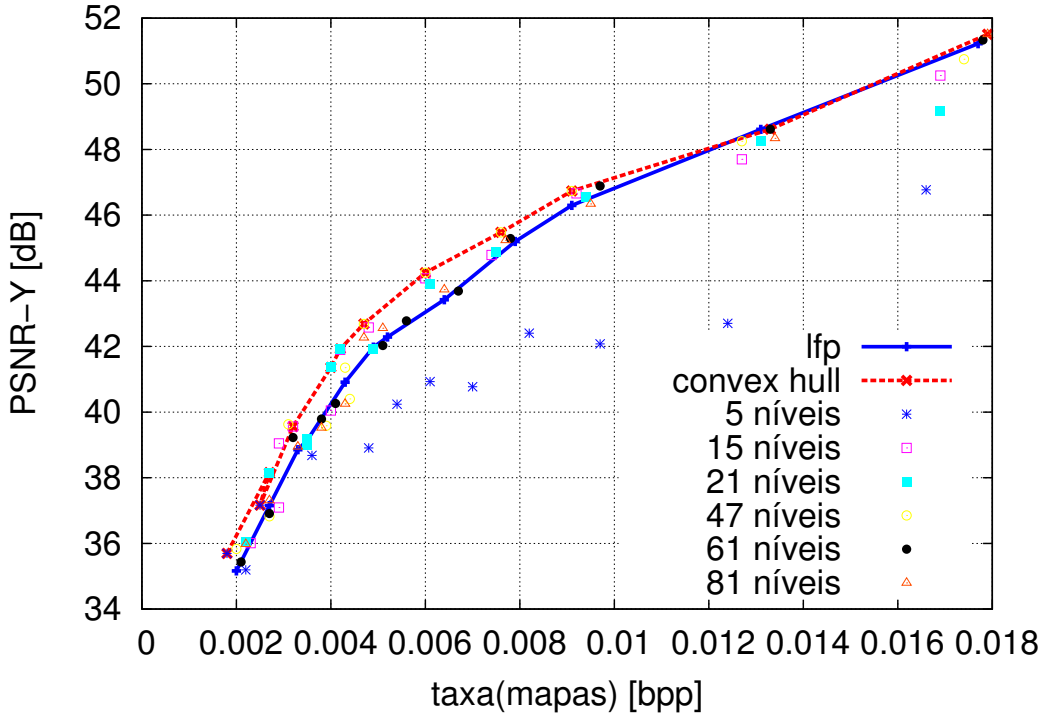


Figura C.5: Efeito da quantização dos coeficientes de aproximação na qualidade das vistas reconstruídas usando os mapas codificados (*PoznanHall2* câmera 6)

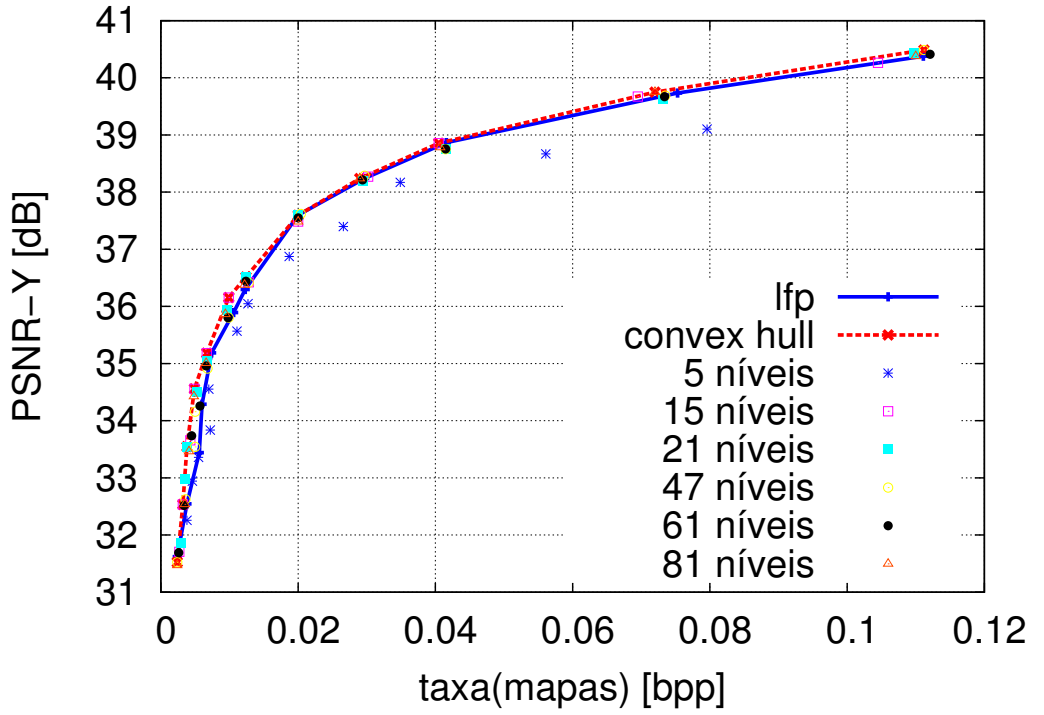


Figura C.6: Efeito da quantização dos coeficientes de aproximação na qualidade das vistas reconstruídas usando os mapas codificados (*PoznanStreet* câmera 4)

C.1.3 Combinando a quantização e aproximação do resíduo

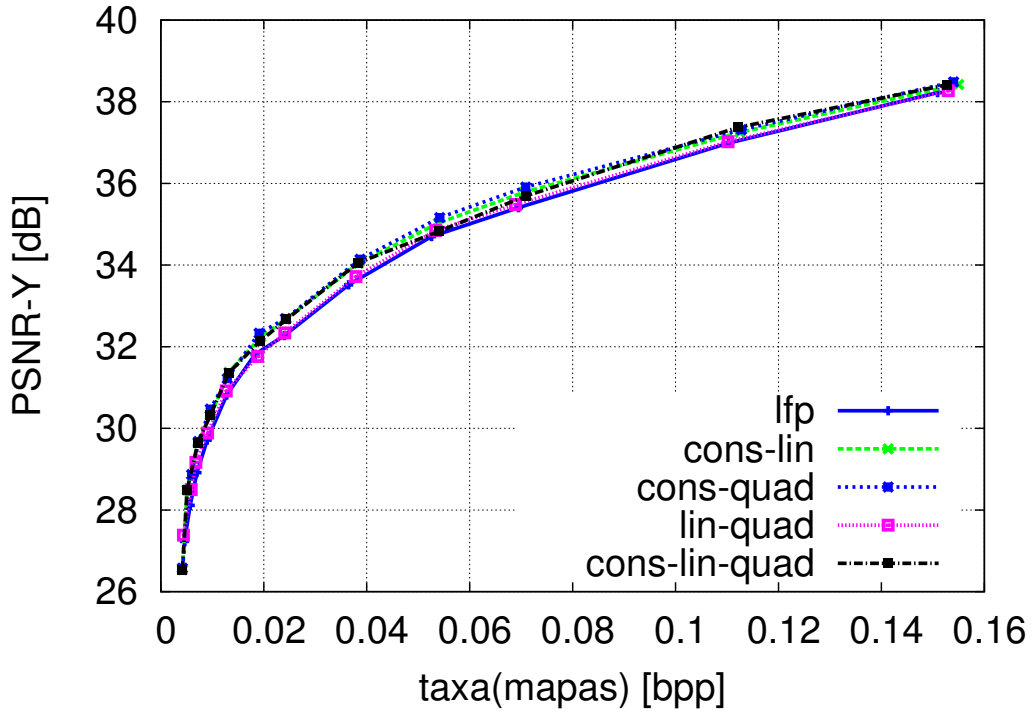


Figura C.7: Combinando os esquemas: quantização ótima e escolha ótima das funções de aproximações (*Newspaper* câmera 4)

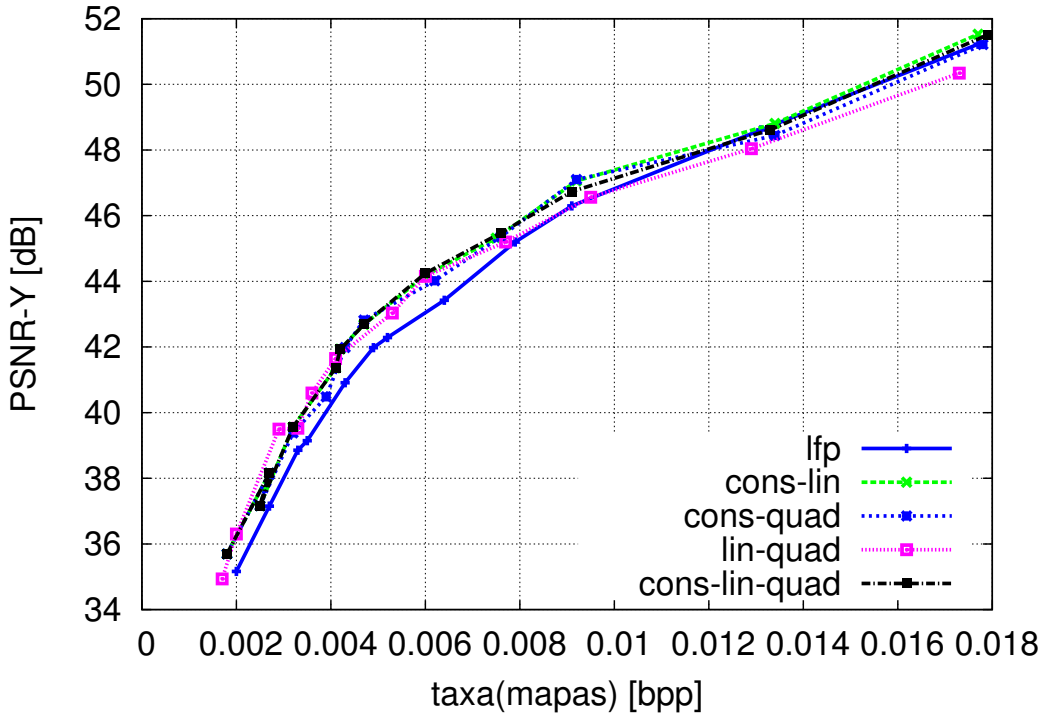


Figura C.8: Combinando os esquemas: quantização ótima e escolha ótima das funções de aproximações (*PoznanHall2* câmera 6)

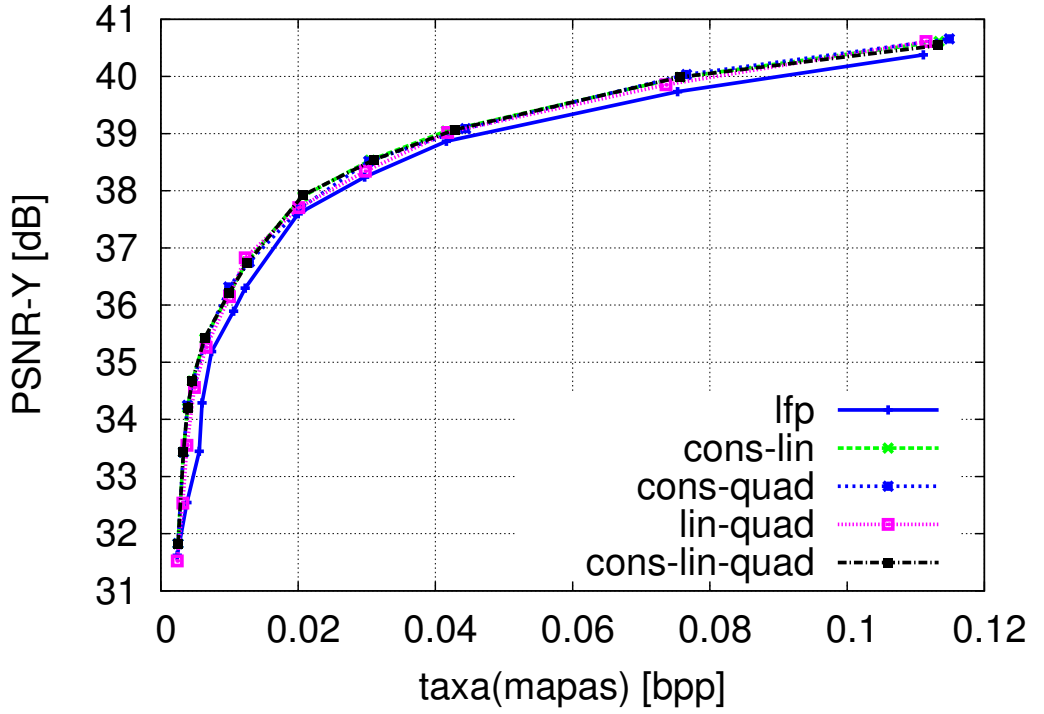


Figura C.9: Combinando os esquemas: quantização ótima e escolha ótima das funções de aproximações (*PoznanStreet* câmera 4)

C.2 Resultados Complementares (Textura)

C.2.1 Codificação de imagens de textura

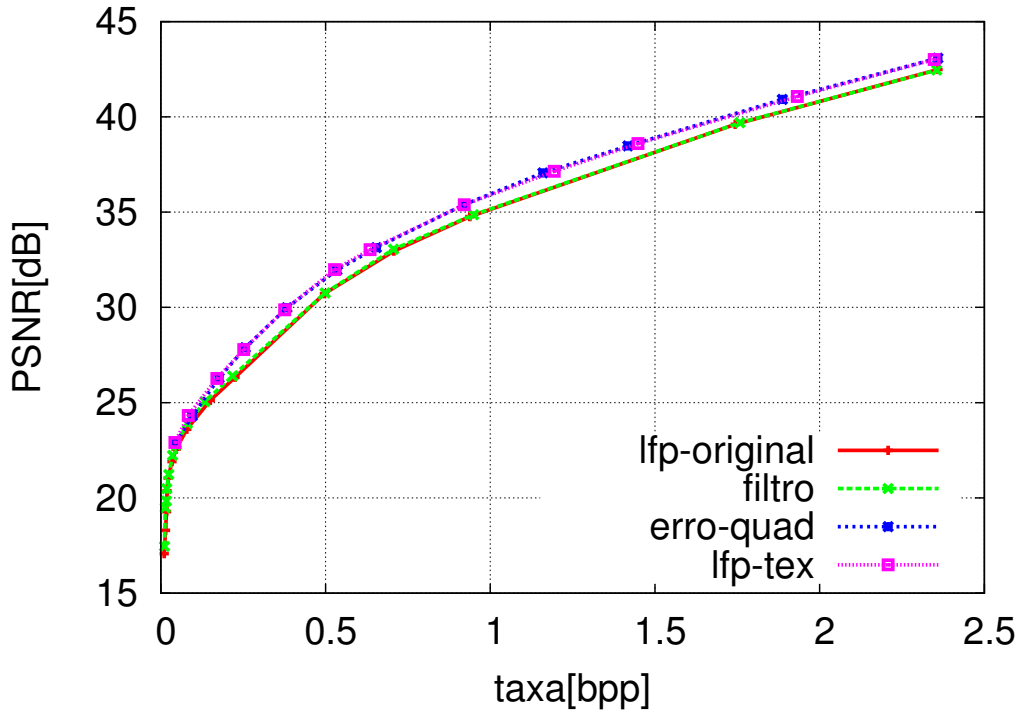


Figura C.10: Comparação de desempenho taxa-distorção dos algoritmos *lfp* e *lfp-tex*: *barb*

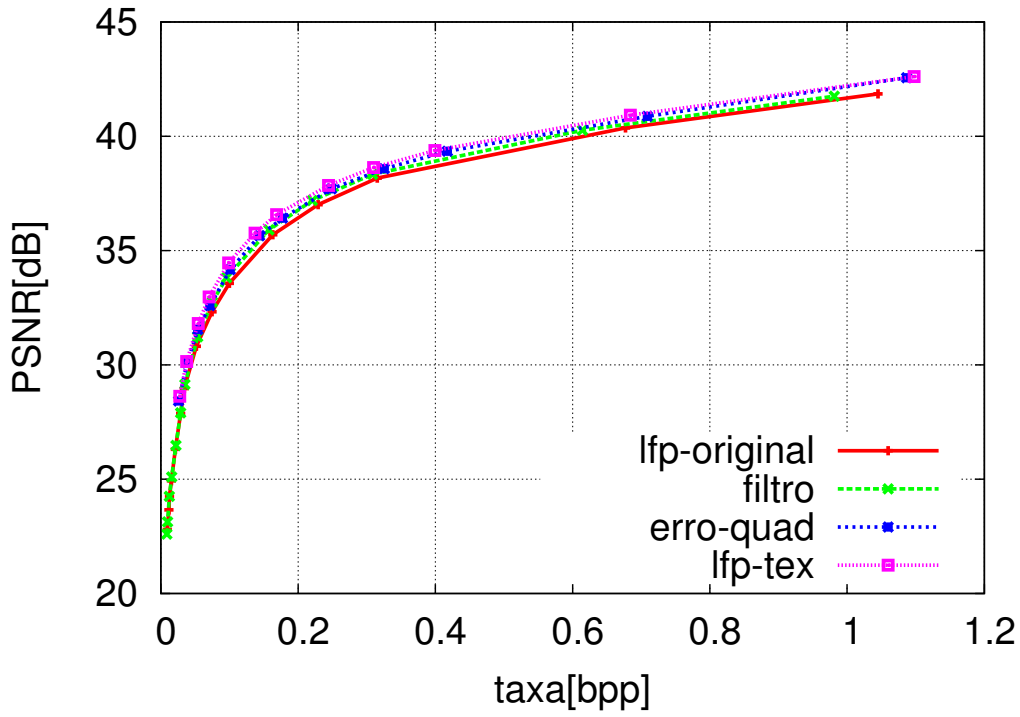


Figura C.11: Comparação de desempenho taxa-distorção dos algoritmos *lfp* e *lfp-tex*: *zelda*

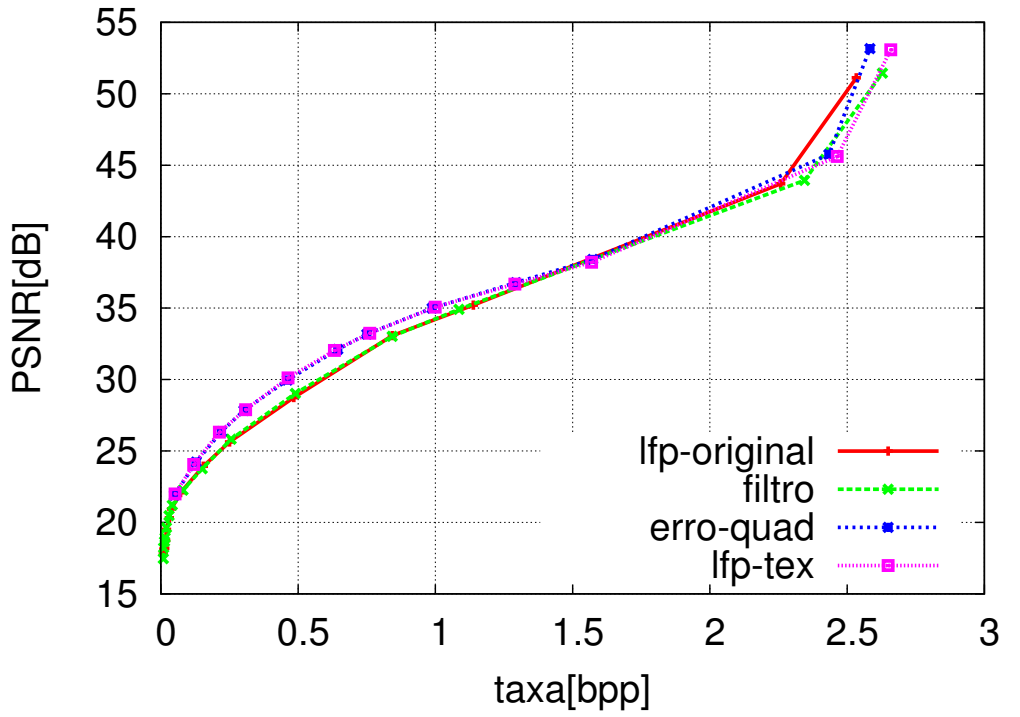


Figura C.12: Comparação de desempenho taxa-distorção dos algoritmos *lfp* e *lfp-tex*: *pp1209*

C.2.2 Funções de aproximação do resíduo

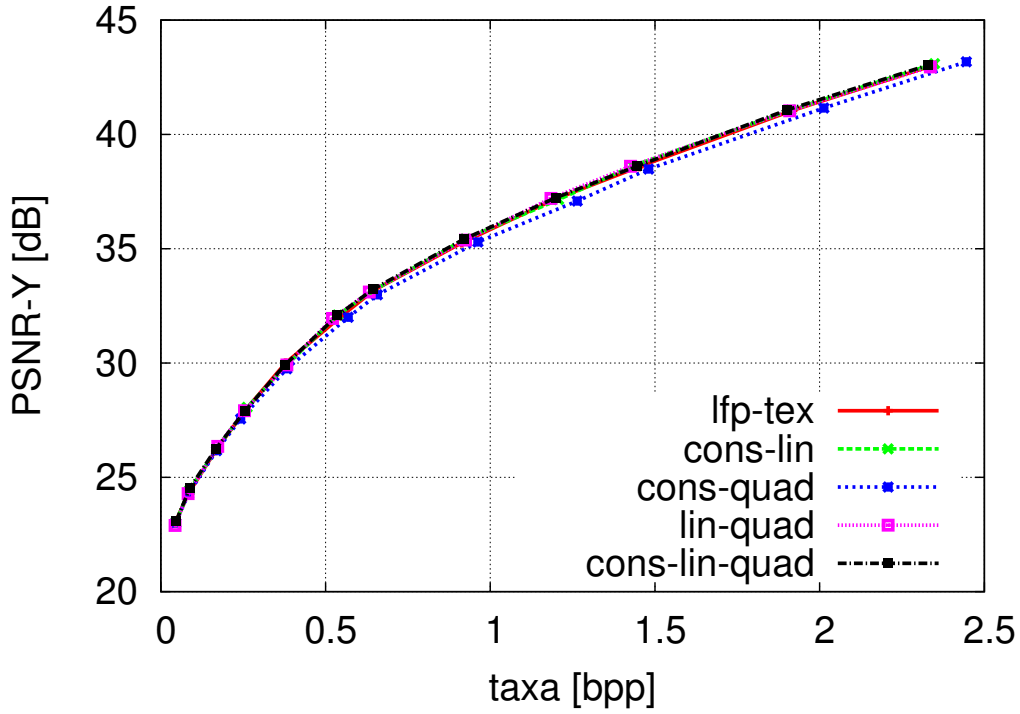


Figura C.13: Comparação do desempenho taxa-distorção para diferentes funções de aproximação do resíduo: *barb*

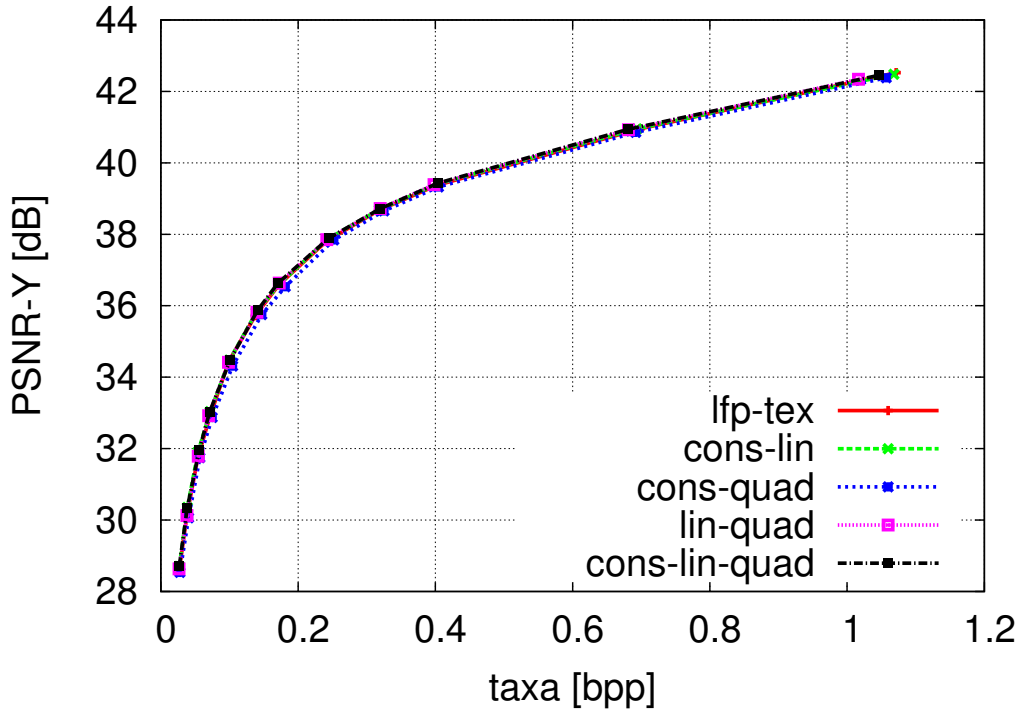


Figura C.14: Comparação do desempenho taxa-distorção para diferentes funções de aproximação do resíduo: *zelda*

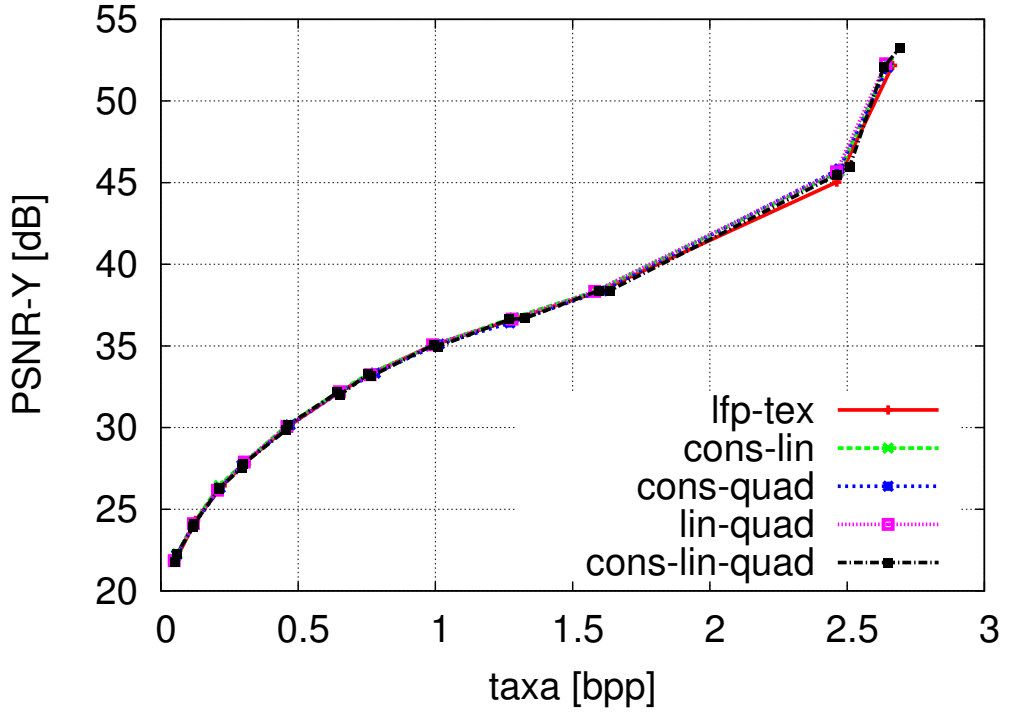


Figura C.15: Comparação do desempenho taxa-distorção para diferentes funções de aproximação do resíduo: *pp1209*

C.2.3 Quantização dos coeficientes das funções de aproximação

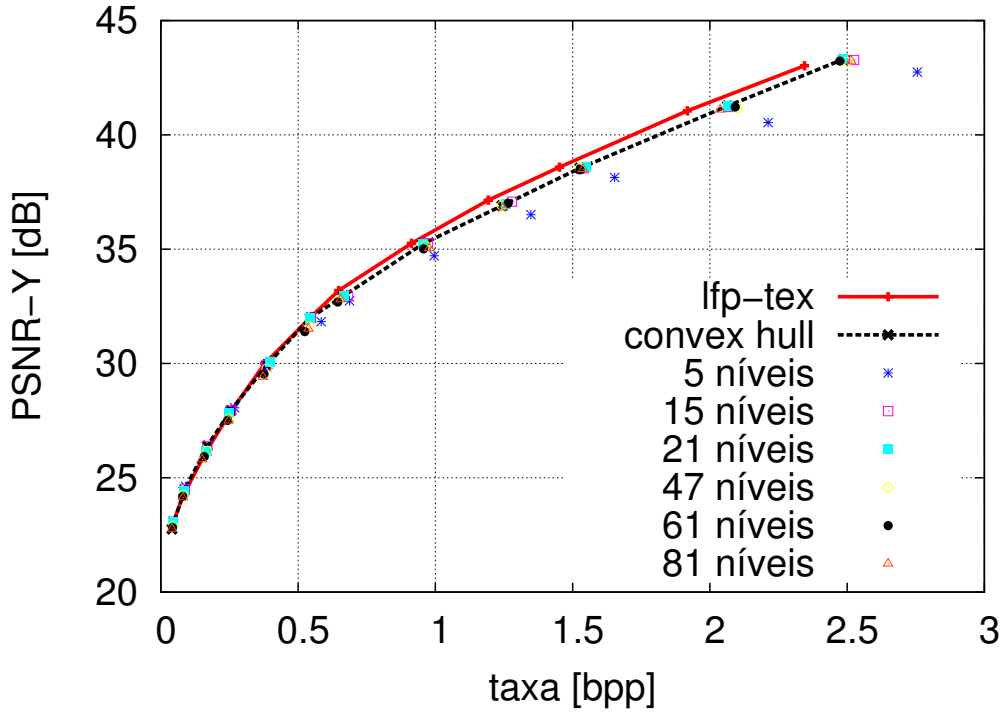


Figura C.16: Comparação dos resultados taxa-distorção obtidos para diversos quantizadores dos coeficientes: *barb*

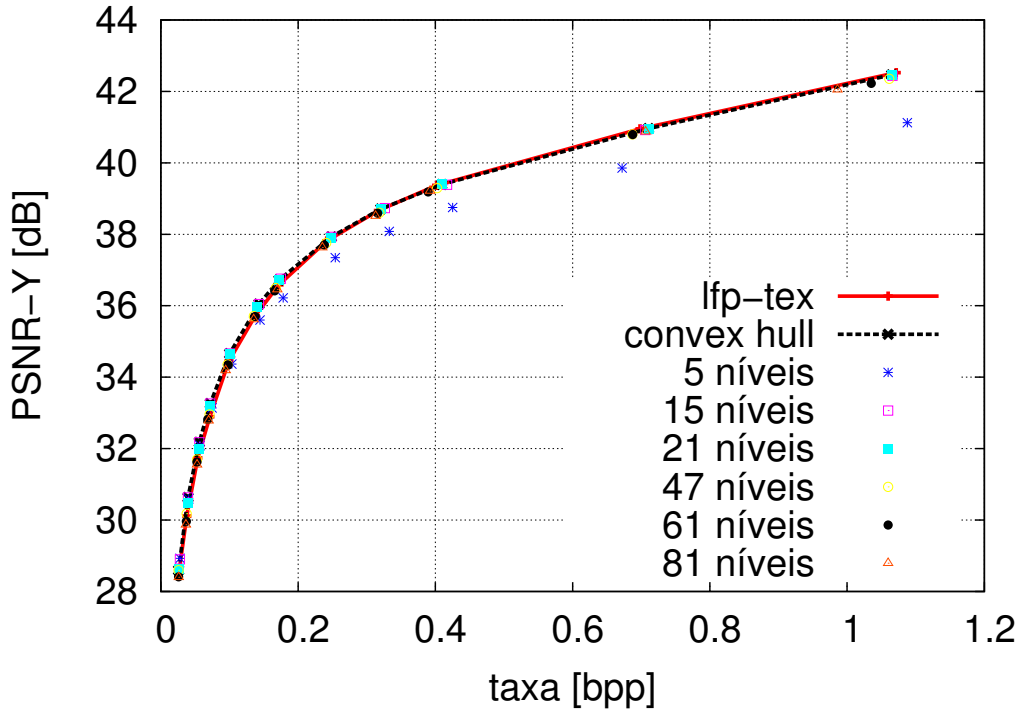


Figura C.17: Comparação dos resultados taxa-distorção obtidos para diversos quantizadores dos coeficientes: *zelda*

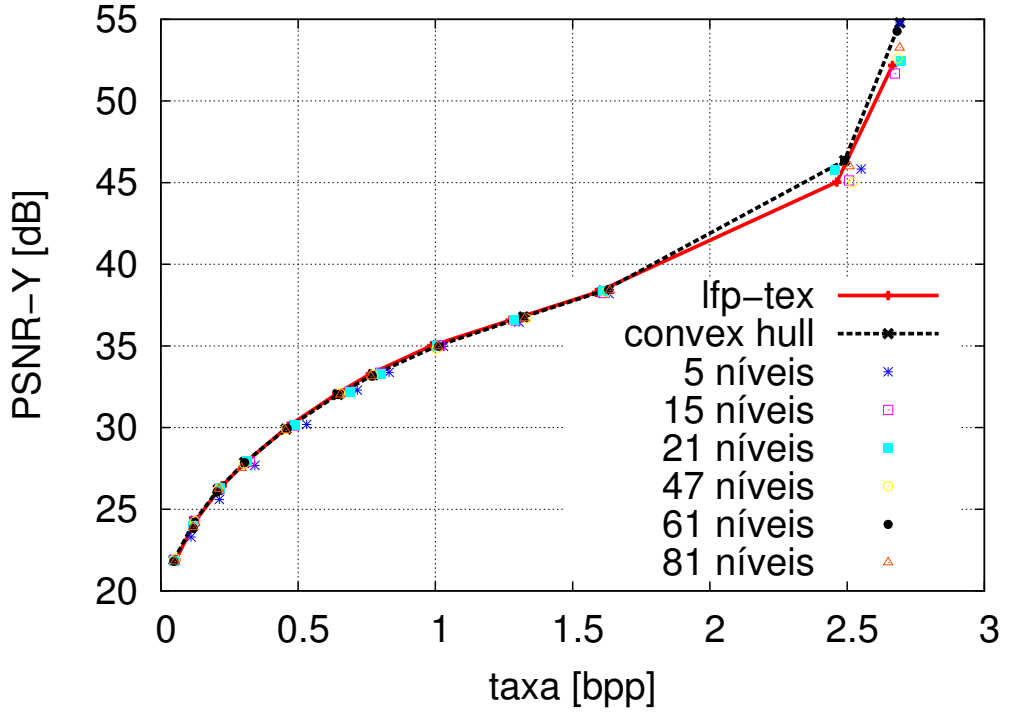


Figura C.18: Comparação dos resultados taxa-distorção obtidos para diversos quantizadores dos coeficientes: *pp1209*

C.2.4 Codificação de *flags*

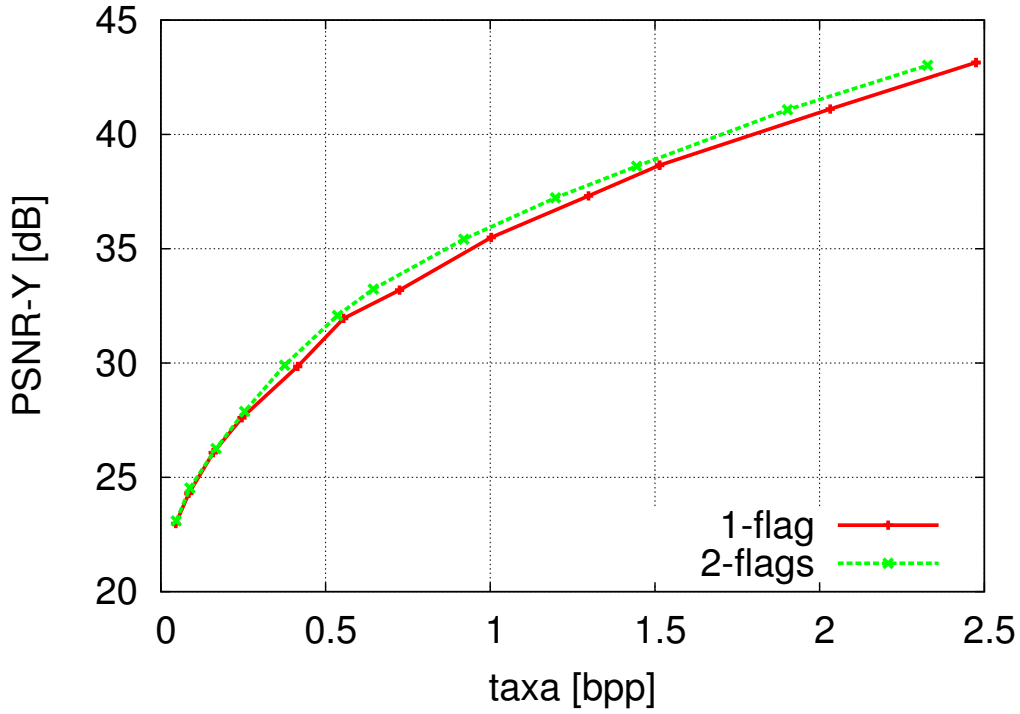


Figura C.19: Comparação dos resultados taxa-distorção obtidos utilizando diferentes esquemas de codificação de *flags*: *barb*

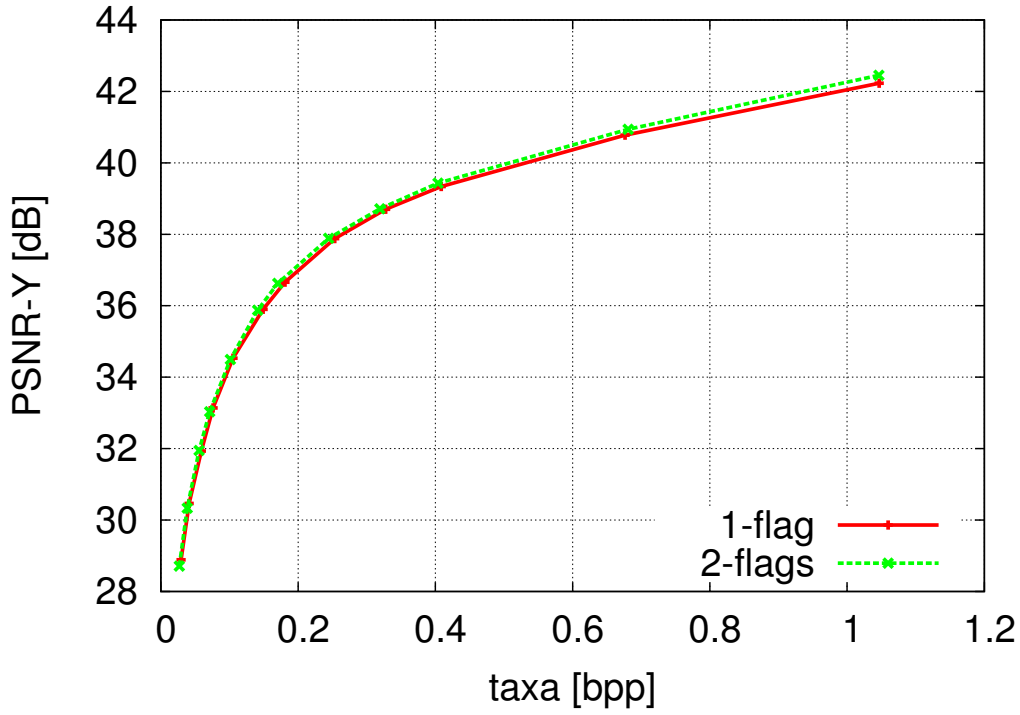


Figura C.20: Comparação dos resultados taxa-distorção obtidos utilizando diferentes esquemas de codificação de *flags*: *zelda*

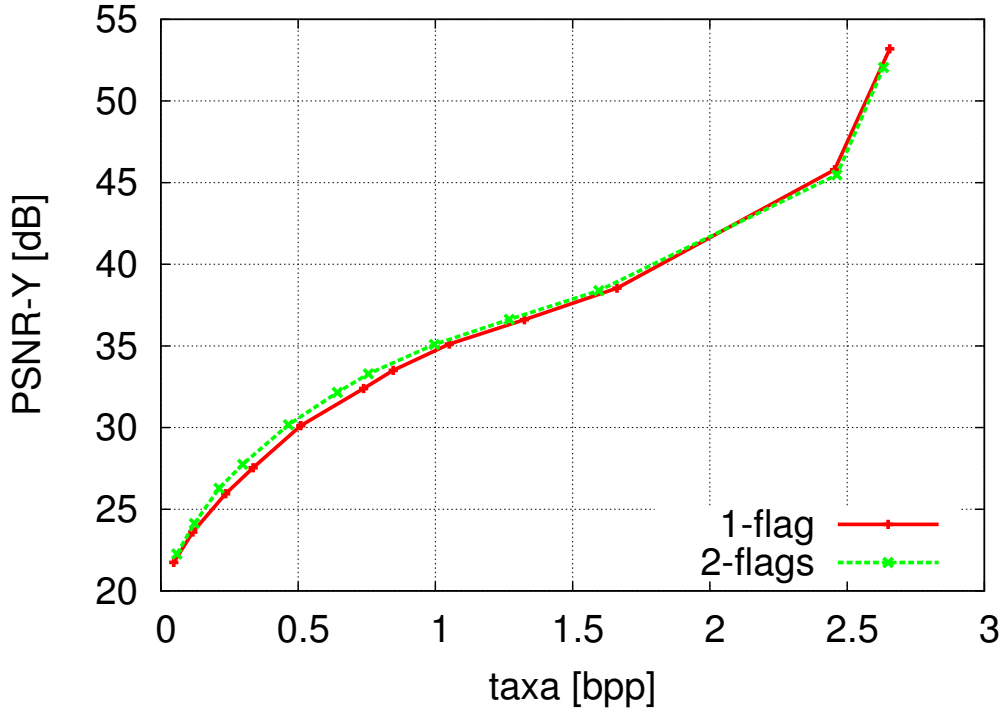


Figura C.21: Comparação dos resultados taxa-distorção obtidos utilizando diferentes esquemas de codificação de *flags*: *pp1209*

C.2.5 Dicionários multiescalas

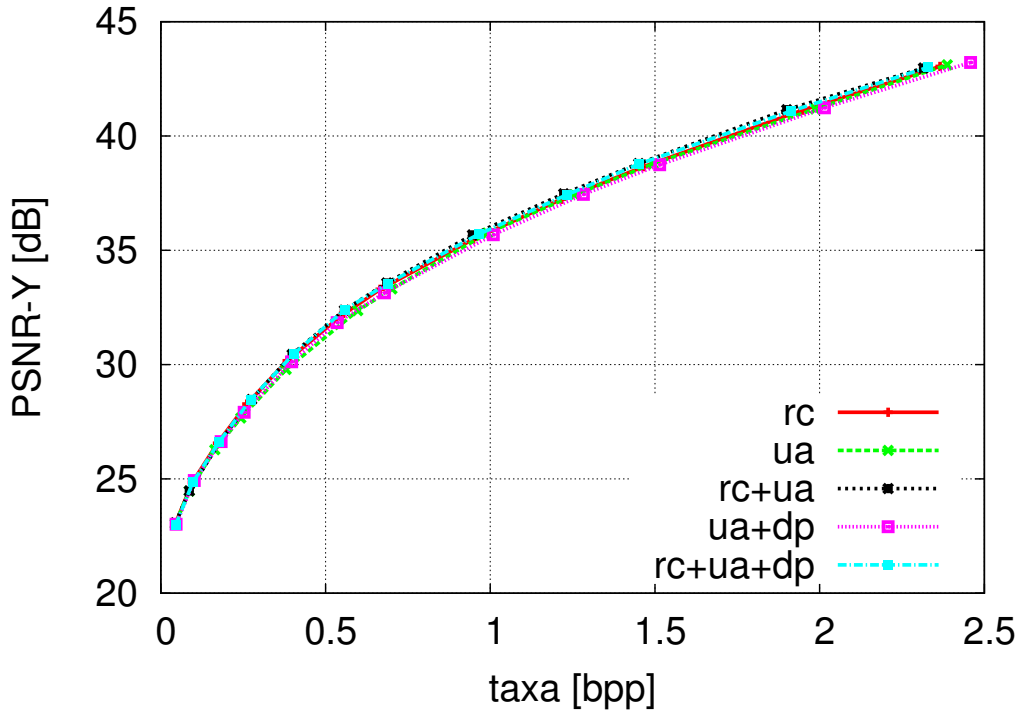


Figura C.22: Comparação desempenho taxa-distorção para diferentes estratégias de atualização dos dicionários: *barb*. *rc*-controle de redundância, *ua*-atualização multiescalas, *dp*-partição do dicionário por escala de origem

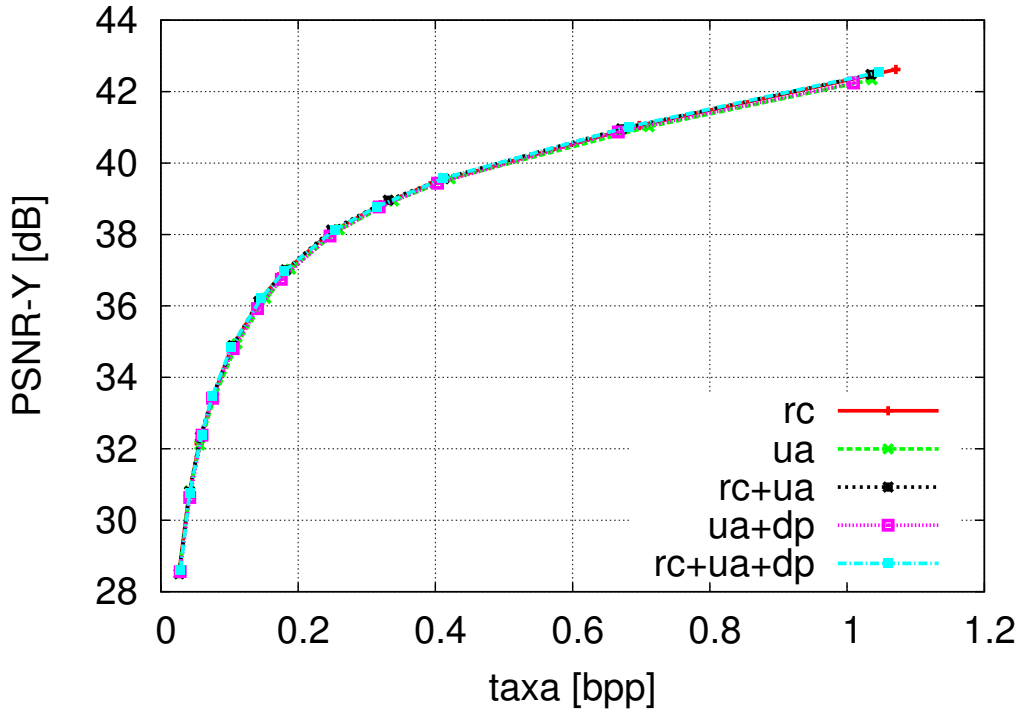


Figura C.23: Comparação desempenho taxa-distorção para diferentes estratégias de atualização dos dicionários: *zelda*. *rc*-controle de redundância, *ua*-atualização multiescalas, *dp*-partição do dicionário por escala de origem

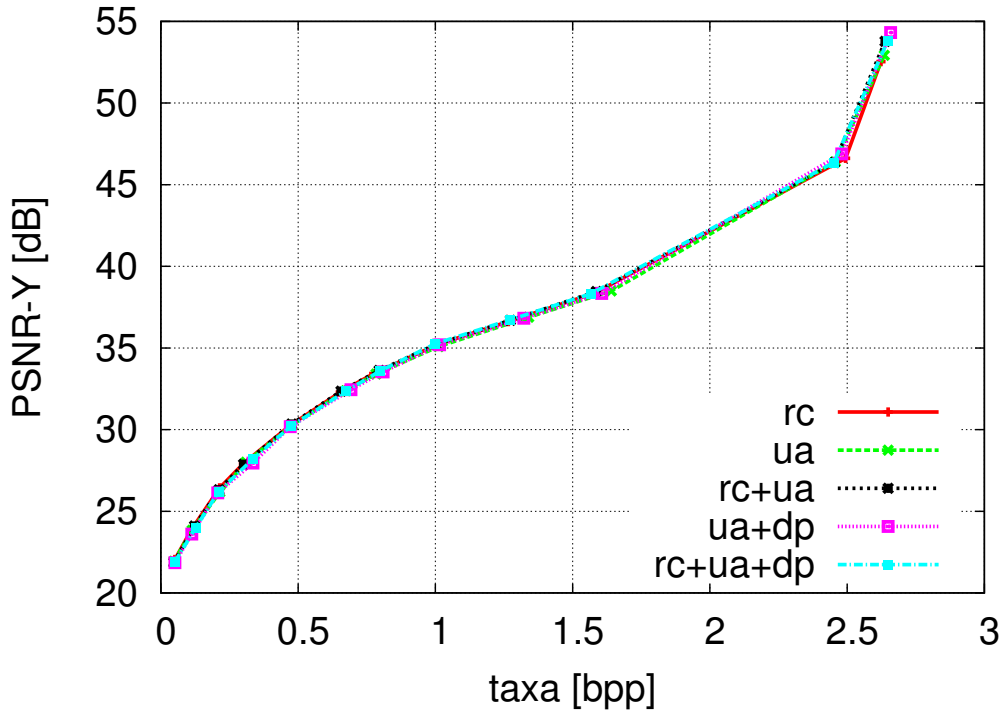


Figura C.24: Comparação desempenho taxa-distorção para diferentes estratégias de atualização dos dicionários: *pp1209*. *rc*-controle de redundância, *ua*-atualização multiescalas, *dp*-partição do dicionário por escala de origem