UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM
INSTITUTO DE CIÊNCIAS EXATAS - ICE
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

COMPRESSÃO DE IMAGENS UTILIZANDO RECORRÊNCIA DE PADRÕES MULTIESCALA COM CASAMENTO LATERAL GENERALIZADO

ÚRSULA VASCONCELOS ABECASSIS

MANAUS
2006

UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM
INSTITUTO DE CIÊNCIAS EXATAS - ICE
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

ÚRSULA VASCONCELOS ABECASSIS

COMPRESSÃO DE IMAGENS UTILIZANDO RECORRÊNCIA DE PADRÕES MULTIESCALA COM CASAMENTO LATERAL GENERALIZADO

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Informática, área de concentração Engenharia da Computação.

Orientador: Prof^{o}. Dr. Ayres Mardem Almeida do Nascimento

MANAUS

2006

UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM
INSTITUTO DE CIÊNCIAS EXATAS - ICE
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

ÚRSULA VASCONCELOS ABECASSIS

COMPRESSÃO DE IMAGENS UTILIZANDO RECORRÊNCIA DE PADRÕES MULTIESCALA COM CASAMENTO LATERAL GENERALIZADO

Aprovada por:

Prof. Ayres Mardem A. do Nascimento, Dr.

Prof. Evaldo Gonçalves Pelaes, Dr.

Prof. José Raimundo Gomes Pereira, Dr.

Prof. Raimundo da Silva Barreto, Dr.

MANAUS

2006

Agradecimentos

Agradeço ao meu Deus e pai pela forma maravilhosa e poderosa com que tem me conduzindo até hoje, pelas muitas bençãos recebidas e pelas vitórias conquistadas, e essa realidade se faz presente mais uma vez através da execução deste trabalho e pelas pessoas formidáveis que Ele tem colocado em minha vida, das quais posso citar e agradecer:

Aos meus pais Salete e Samuel Abecassis que me ensinaram os verdadeiros valores desta vida, através de seu amor e dedicação. As minhas irmãs Michelle e Ingrid pelo apoio e amor. Ao meu esposo Marcos Cunha, pela paciência e compreensão da minha ausência neste período de descobertas, crescimento e desesperos que passam todos os mestrandos. Aos meus amigos Waldir Sabino, que sempre esteve disposto a me auxiliar, incentivar com seus conselhos e aguentar os meus estresses e vice-versa e Eddie Lima, pelas palavras de ânimo e de como via os problemas de forma tão simples e apesar da distância sempre esteve presente, são como irmãos para mim, a vocês minha eterna gratidão. Ao amigo Ilton Seixas pelas valiosas dicas da linguagem C. Aos amigos Walter Lucas Pini, Vanderson Reis, Frederico Pinagé e Dr. Ricardo Wilson pelo companheirismo, amizade e descontração. Aos amigos Dina Soares, Arlete Auzier e Luiz Nagata pelas orações sinceras. Ao professor Dr. Edleno Moura pelo incentivo na área de programação, que antes para mim era um terreno desconhecido. Ao professor Dr. José Raimundo, pela sua maravilhosa didática, que conseguiu transformar a disciplina processos estocásticos em um assunto interessante e envolvente e com quem me identifico muito quando estou em sala de aula. À FUCAPI, através da pessoa de Niomar Pimenta pela oportunidade da realização deste trabalho. Ao meu orientador Ayres Marden pelos conselhos e por ter acreditado em meu potencial, muito obrigada à todos vocês.

Úrsula Vasconcelos Abecassis

Resumo da Tese apresentada ao PPGI/UFAM como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática (M.Sc.)

COMPRESSÃO DE IMAGENS UTILIZANDO RECORRÊNCIA DE PADRÕES MULTIESCALA COM CASAMENTO LATERAL GENERALIZADO

Úrsula Vasconcelos Abecassis

Outubro/2006

Orientador: Ayres Mardem Almeida do Nascimento

Programa: Programa de Pós-Graduação em Informática - PPGI

O algoritmo MMP (*Multi-dimensional Multiscale Parser*) é um esquema de compressão com perdas que tem como base o casamento aproximado de padrões multiescalas, no qual blocos de sinais de dimensões diferentes podem ser aproximados pelo seu dicionário. Tal aproximação é possível devido a uma transformação de escala que adapta as dimensões dos blocos, em versões dilatadas e contraídas, antes do casamento ser realizado. O MMP vem apresentando um bom desempenho em images com grande conteúdo de alta freqüência, mas em imagens suaves não apresentou resultados satisfatórios. Neste trabalho propõem-se uma melhoria da capacidade de compressão de imagens suaves, sem diminuir a qualidade de imagens mais complexas, tendo como base o algoritmo MMP, este algoritmo é chamado de GSM-MMP (*Generalized Side-Match Multidimensional Multiscale Parser*). O GSM-MMP realiza uma predição para a estruturação do dicionário de codificação baseada em semelhança com os três vizinhos do bloco atual que será codificado.

Abstract of Thesis presented to PPGI/UFAM as a partial fulfillment of the requirements for the degree of Master of Informatics (M.Sc.)

COMPRESSION OF IMAGES BASED ON RECURRENT MULTISCALE PATTERNS WITH GENERALIZED SIDE-MATCH

Úrsula Vasconcelos Abecassis

October/2006

Advisor:Ayres Mardem Almeida do Nascimento

Department: Informatics

The algorithm MMP (*Multi-dimensional Multiscale Parser*) it is a compression outline with losses that he has as base the marriage approximate of patterns multiescalas, in which blocks of signs of different dimensions can be approximate for his dictionary. Such approach is possible due to a scale transformation that it adapts the dimensions of the blocks, in extensive versions and contracted, before the marriage to be accomplished. MMP comes presenting a good acting in images with great content of high frequency, but in soft images it didn't present results satisfactory. In this work they intend an improvement of the capacity of compression of soft images, without reducing the quality of more complex images, tends as base the algorithm MMP, this algorithm is called of GSM-MMP (*Generalized Side-Match Multidimensional Multiscale Parser*). GSM-MMP accomplishes a prediction for the structuring of the code dictionary based on similarity with the three neighbors of the current block that it will be codified.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A contínua produção de tecnologia digital nos últimos anos tem proporcionado uma revolução mundial conhecida por *era da informação*. Nos dias de hoje, expressões como tempo real e sob demanda tem dado uma idéia da rapidez com que ocorre a informação através dos canais de comunicação que envolvem o globo. E tais informações sobretudo de imagens, tornou-se um instrumento multidisciplinar e até sócio-cultural, como pode ser notado na telemedicina, geoprocessamento, automação industrial, entretenimento, internet entre outras, comprovando o fato da revolução digital em várias áreas. Da mesma forma que há uma quantidade expressiva de informação em formato digital. Em virtude deste fato surge a necessidade de espaço para o armazenamento, ou banda para a transmissão, de tais informações em uma forma mais compacta capaz de contemplar a qualidade da mesma.

A ciência que representa a informação em uma forma compacta é chamada de compressão de dados, que sem a mesma não poderia ser possível tais revoluções na área tecnológica digital. O termo dado está se referindo aos meios pelos quais a informação é conduzida, e quando tal informação é conduzida através de imagens é comum usar o termo - compressão de imagens.

O foco deste trabalho consiste na compressão de imagens, que é a técnica natural para a manipulação de dados visuais de alta resolução como sensores de imageamento e vídeo para broadcasting. Além disso, tal técnica desempenha um papel crucial em muitas aplicações importantes e diferentes, incluindo sensoria-

mento remoto (ex. uso de imagens de satélites para aplicações climáticas e outros recursos da terra), compressão de imagens para programas de edição (ex. *Microsoft Paint*), envio de dados em sistemas de banda estreita (ex. telefonia celular), armazenamento de imagens médicas, etc.

A proposta do presente trabalho consiste no aperfeiçoamento de um algoritmo de compressão de sinais multidimensionais chamado MMP (*Multidimensional Multiscale Parser*), cuja principal característica é processar o dado de entrada através de execução em um único passo das etapas de transformação, quantização e codificação de entropia. A base do algoritmo está na recorrência de padrões multiescalas, que consiste na aproximação dos blocos de imagens por elementos previamente codificados e armazenados em um dicionário.

O algoritmo MMP vem apresentando um bom desempenho na compressão de imagens com grande conteúdo de alta freqüência, mas falhava na compressão de imagens mais suaves onde a maior parte da energia é concentrada em baixas freqüências. Com base nisso, propõem-se o desenvolvimento de um novo critério para a seleção de um dicionário de codificação, baseado na semelhança dos três vizinhos do bloco atual este algoritmo é chamado de GSM-MMP (*Generalized Side-Match Multidimensional Multiscale Parser*).

Os resultados obtidos são bastante motivantes e mostram que este é um caminho que pode ser seguido para a obtenção de resultados mais promissores comparados com os apresentados pelo estado da arte nessa área e resultando também na contribuição para o desenvolvimento de tais pesquisas e aumentando o número de ferramentas disponíveis para tal algoritmo.

Esta dissertação está organizada da seguinte forma:

No capítulo 2 deste trabalho serão apresentados os fundamentos matemáticos de compressão de dados, onde serão mostrados os conceitos da *teoria da informação* e da *taxa-distorção* que são elementos fundamentais para compreensão dos esquemas de compressão de dados.

No capítulo 3 descreve-se as técnicas de compressão de sinais existentes, os critérios objetivos e subjetivos para a avaliação de desempenho dos algoritmos de compressão de dados, bem como as métricas adotadas nesta dissertação de

mestrado.

O capítulo 4 apresenta o algoritmo de compressão com perdas chamado MMP (*Multidimensional Multiscale Parser*), que é baseado em casamento aproximado de padrões multiescalas e suas versões. Mostra os resultados de simulações de tais versões, com o MMP padrão e outros algoritmos para fazer as devidas comparações.

O capítulo 5 aborda o tema principal deste trabalho onde será apresentado a descrição e implementação do algoritmo GSM-MMP (*Generalized Side-Match Multidimensional Multiscale Parser*) capaz de melhorar o desempenho do MMP padrão tanto nas imagens suaves como nas imagens em que o MMP se destaca (gráficos, textos ou mistas). Abordando também os resultados obtidos deste novo método.

No capítulo 6 faz-se um balanço das conclusões e resultados obtidos, apresentando sugestões para trabalhos futuros, visando dar continuidade aos avanços sobre o tema exposto.

Capítulo 2

Fundamentos matemáticos para compressão de dados

Neste capítulo serão mostrados os conceitos da *teoria da informação* e da *taxa-distorção*. Elementos importantes para o entendimento dos esquemas de compressão de dados.

2.1 Introdução à teoria da informação

A área da ciência que descreve de forma quantitativa a informação, abordando aspectos como modelos matemáticos, capacidade de canal e compromisso entre a taxa e distorção, é chamada de *Teoria da informação* [1, 2].

O engenheiro eletricista Claude E. Shannon, estabeleceu os fundamentos mais importantes e genéricos da *Teoria da Informação* em [2]. Ele utilizou uma quantidade chamada de *auto-informação* que tem por definição mensurar a incerteza de um determinado evento [1]. Considerando um evento aleatório E, e sua medida de probabilidade P(E), então a *auto-informação* associada ao evento E é dada pela equação 2.1 :

$$i(E) = \log_b \frac{1}{P(E)} \tag{2.1}$$

Onde a base do logaritmo define a unidade na qual a *auto-informação* é obtida. Se a base b for igual a 2 a unidade será expressa em bits.

Observando a equação 2.1, temos que a quantidade de informação atribuída ao evento E é inversamente relacionada à probabilidade do evento E. Se P(E)=1 (isto é, o evento sempre ocorre) a *auto-informação* é zero ou seja nenhuma informação é a ele atribuída. Isto é, como nenhuma incerteza está associada com o evento, nenhuma informação seria transferida pela comunicação de que o evento tenha ocorrido. Em suma, se um evento tem probabilidade baixa, a informação associada a ele é alta; caso contrário, a informação associada é baixa.

Shannon definiu também a *auto-informação média*, conhecida como *entropia*, que pode ser definida conforme [1, 3, 4] do seguinte modo:

Seja um experimento S, com um conjunto de eventos E_i associados tais que

$$\bigcup E_i = S, \tag{2.2}$$

onde S é o espaço amostral [1]. Os eventos E_i serão independentes se :

$$P(E_i \cap E_j) = P(E_i).P(E_j), i \neq j. \tag{2.3}$$

Neste caso, a *auto-informação média* associada ao experimento S é dada pela fórmula :

$$\begin{aligned} H(S) &= \sum_{i=1}^{n} P(E_i).i(E_i) \\ &= -\sum_{i=1}^{n} P(E_i).\log_2(P(E_i)), \end{aligned} \tag{2.4}$$

que também pode ser chamada de *entropia* associada ao experimento.

Na *teoria da informação*, o experimento S descrito acima é chamado de fonte, os eventos E_i associados ao experimento S, são chamados de símbolos ou letras e o conjunto de símbolos $E = \{E_i\}$, são denominados de alfabeto da fonte. Segundo Shannon[5] esta quantidade possui um limite teórico para a compressão, que é a entropia. Isto significa que o menor comprimento médio em bits, que se pode representar os símbolos de uma fonte é igual à entropia.

Tratando-se de esquemas de compressão que inserem perdas, os alfabetos de reconstrução e a fonte podem ser distintos, com isto necessita-se quantificar as relações de informação entre duas variáveis aleatórias que assumam valores

de dois alfabetos diferentes. Uma medida de relação entre essas duas variáveis muito utilizada, é a entropia condicional. Supondo-se uma variável aleatória $X = \{x_0, x_1, ..., x_{n-1}\}$, que assume valores do alfabeto da fonte, e outra variável $Y = \{y_0, y_1, ..., y_{m-1}\}$, que assume valores do alfabeto de reconstrução, cujas entropias segundo a equação 2.4 são respectivamente:

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} P(x_i) \log_2 (x_i) \\ H(Y) &= -\sum_{j=1}^{m} P(y_j) \log_2 (y_j) , \end{aligned} \tag{2.5}$$

logo a entropia condicional é definida segundo [1] como :

$$H(X|Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i|y_j) P(y_j) \log_2 P(x_i|y_j) \tag{2.6}$$

Onde $H(X|Y)$ é a entropia de X dado que Y ocorreu. A entropia condicional pode ser interpretada como a incerteza sobre a variável X dado que se conhecem os valores de reconstrução que Y assume [5]. Esse conhecimento adicional de Y reduz a incerteza sobre X.

Outra medida de incerteza sobre duas variáveis aleatórias, também muito utilizada, é a *informação mútua média*. Supondo-se as mesmas variáveis aleatórias X e Y definidas para a entropia condicional, a *informação mútua média* é dada em bits por :

$$\begin{aligned} I(X|Y) &= \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i; y_j) i(x_i; y_j) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i; y_j) \log_2 \left[\frac{P(x_i|y_j)}{P(x_i)} \right] \end{aligned}$$

Usando-se a equação 2.5 temos:

$$I(X|Y) = H(X) - H(X|Y) \tag{2.7}$$

A informação mútua média pode ser interpretada como a redução na incerteza de X devido ao conhecimento de Y [1]. Com estes conceitos podemos definir a *teoria taxa-distorção* e analisar a função taxa-distorção.

2.2 Teoria taxa-distorção

Quando são utilizados esquemas de compressão que inserem perdas pode-se aumentar a compactação dos dados obtidos, desde que seja aceita uma quantidade maior de distorção. Todavia, surge a necessidade de avaliar esta quantidade de bits necessárias para representar esses dados após a sua compactação, de modo que se possa tomar uma decisão se uma redução compensa o aumento da distorção.

A teoria taxa-distorção procura ver a relação satisfatória entre a taxa de compressão R e a distorção D, e isto é realizado através da função chamada taxa-distorção representada por $R(D)$. Esta função taxa-distorção $R(D)$, irá especificar qual é a menor taxa média R que a saída da fonte pode ser codificada, sempre mantendo uma distorção média menor ou igual a distorção D [1, 5].

A função taxa-distorção define o melhor desempenho possível para atingir qualquer código de compressão, satisfazendo sempre a desigualdade $R_{codigo} \geq R(D)$.

A definição matemática da função taxa-distorção $R(D)$, será descrita a seguir na forma de um teorema definido por [1].

Teorema 2.2.1 (Função taxa-distorção) *Dada uma fonte X identicamente distribuída com distribuição $P(x_i)$ e uma função distorção limitada $d(x_i, y_j)$ então a função taxa-distorção será dada por:*

$$R(D) = \min_{P(y_j|x_i) \,|\, D \leq D^*} I(X;Y) \tag{2.8}$$

Onde a distorção média é $D = \sum_i \sum_j P(x_i)P(y_j|x_i)d(x_i, y_j)$ e D^* é a distorção alvo.

O gráfico da Figura 2.1, mostra o comportamento da curva taxa-distorção onde percebe-se dois casos extremos. O primeiro caso extremo ocorre quando não se transmite qualquer informação, ou seja, quando a taxa é zero, definido por $(R_x, D_x) = (0, D_{max})$, e o segundo caso extremo ocorre quando se transmite toda a informação, o que leva a uma distorção zero, mostrada pela coordenada $(R_y, D_y) = (R_{max}, 0)$.

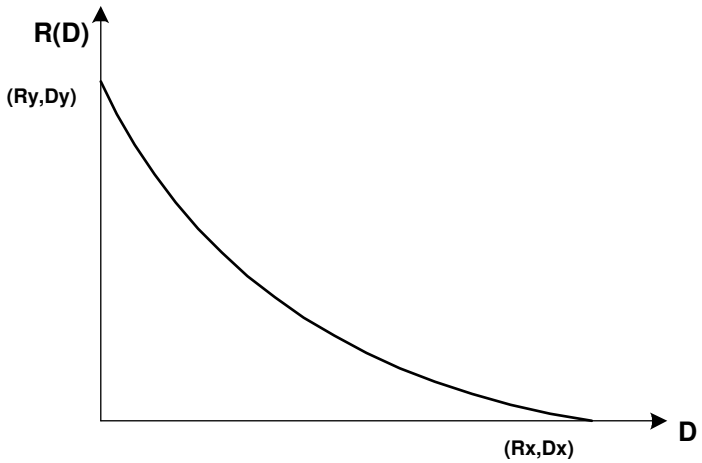


Figura 2.1: Curva Taxa-Distorção $R(D)$

Capítulo 3

Compressão de dados

A compressão de dados está se tornando cada vez mais uma ferramenta cotidiana, a sua utilização é variada e extensa, de tal forma que determinados padrões como o JPEG [6] e MPEG [7], já são termos comuns no dia a dia das pessoas, o advento da TV digital, internet, o uso de modem, fax, os serviços dos aparelhos celulares e TV via satélite, não poderiam ser possíveis sem a compressão de dados. Nas seções à seguir mostraremos o conceito de compressão de dados, as técnicas existentes e as medidas adotadas para a avaliação de desempenho de tais técnicas.

3.1 Conceito de compressão de dados

A compressão de dados é a arte de representar a informação numa forma compacta, e tais representações são obtidas através de estruturas peculiares existentes nos dados, como: identificação de estruturas comuns e repetitivas, que são conhecidas como informações redundantes e irrelevantes.

O termo *dado*,pode se referir a caracteres de um arquivo de texto, seqüencia de números que são amostras de um sinal de voz, matrizes numéricas que são os níveis de cinza de uma imagem, etc.

Quando se analisa alguma técnica de compressão (ou algoritmo de compressão), na verdade estamos analisando dois algoritmos. O primeiro é chamado de codificador (algoritmo de codificação), que processa os dados de entrada X e

gera uma representação X_c com quantidade menor de bits, e o segundo algoritmo é chamado de decodificador (algoritmo de decodificação), responsável por processar a saída produzida pelo codificador, e gera uma saída Y, que representam os dados reconstruídos, este esquema é mostrado na Figura 3.1.

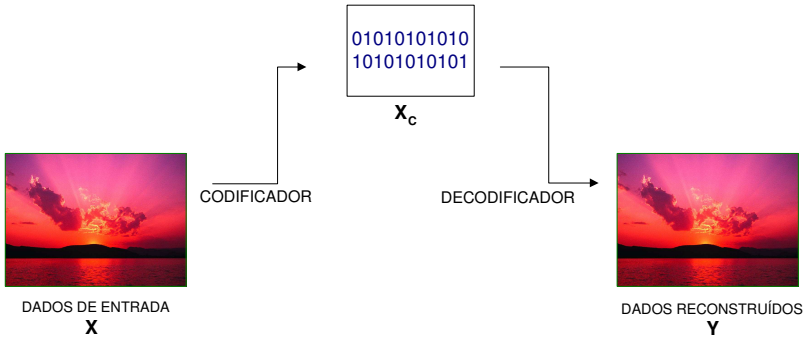


Figura 3.1: Esquema de compressão

Com base no resultado da saída do decodificador, ou seja com os dados reconstruídos, e dependendo das necessidades da reconstrução, as técnicas de compressão de dados podem ser classificadas como: Técnicas de compressão sem perdas e técnicas de compressão com perdas.

3.1.1 Técnicas de compressão sem perdas

A técnica de compressão sem perdas, como o próprio nome sugere, não há a perda da informação, ou seja os dados reconstruídos da saída do decodificador são exatamente iguais aos dados originais da entrada do codificador, como mostrado na Figura 3.2:

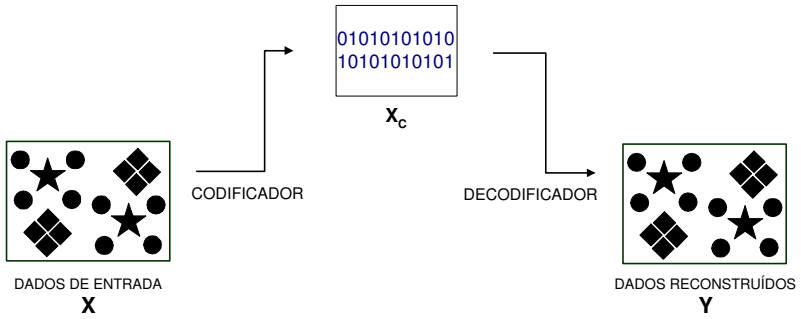


Figura 3.2: Esquema de compressão sem perdas

Esta técnica não provoca distorção no sinal de entrada X do codificador, podendo recuperá-la completamente, sem nenhuma perda quando decodificado. Mas possui a característica de não possuir uma alta taxa de compressão.

Várias são as aplicações de compressão sem perdas, dentre elas podemos citar, imagens médicas onde qualquer alteração da imagem pode ocasionar um diagnóstico falso, arquivos de textos nos quais qualquer diferença pode distorcer toda a informação. Exemplos de alguns algoritmos de compressão sem perdas são: o código de Huffman [8], o codificador aritmético descrito em [9], que associa uma única palavra-código a cada conjunto de dados da fonte, o código de Lempel - Ziv [10, 11], que são técnicas baseadas em dicionários etc. A taxa resultante de tais métodos tende a aproximar a entropia da fonte, ou seja, a quantidade média mínima de bits necessárias para se codificar a saída da mesma. Devido a este fato, os métodos de compressão sem perdas são conhecidos como codificadores de entropia.

3.1.2 Técnicas de compressão com perdas

A técnica de compressão com perdas por sua vez envolve alguma perda de informação, resultando em uma reconstrução não exata na saída do decodificador comparando-se com os dados de entrada do codificador, como mostra a Figura 3.3.

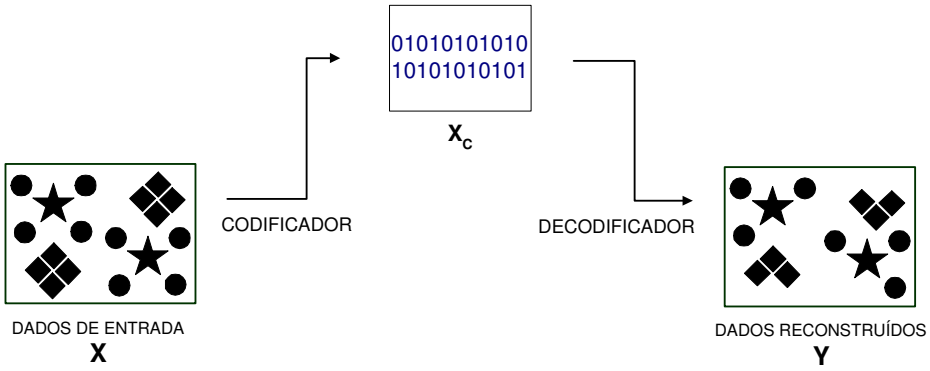


Figura 3.3: Esquema de compressão com perdas

As aplicações desta técnica podem ser vistas na transmissão de voz, áudio e sinais de vídeo para broadcasting, neste último caso o olho humano acaba in-

tegrando a imagem e o erro, e só é perceptível em larga escala. Exemplos desta técnica de compressão são o JPEG [6] e o JPEG2000 [12], ambos desenvolvidos para imagens. Vale ressaltar que uma vez ocorrida a perda de informação, a forma original dos dados não pode ser recuperada.

Em 3.1.1, foi visto que para técnicas de compressão sem perdas, a taxa é a medida principal. Como a técnica de compressão que estamos analisando envolve a perda de informação, logo surge a necessidade de uma medida para caracterizar essa diferença entre X (dados de entrada) e Y (dados reconstruídos) e esta medida é chamada de distorção D. E a relação entre a taxa R e a distorção D é dada pela função taxa-distorção $R(D)$ definida em 2.2. Além disso, não há um método que aproxime a curva $R(D)$, para qualquer fonte, como ocorre nos esquemas de compressão sem perdas, devido a este fato adotam-se métodos que possuem soluções em dois ou três passos.

Na solução em dois passos, cria-se uma versão χ da fonte X, com menor entropia, ou seja, $H(\chi) < H(X)$. Este primeiro passo é chamado de quantização, caracterizado pelo mapeamento $\chi = Q(X)$. Há dois tipos diferentes de quantização: a quantização escalar e a quantização vetorial, que podem ser aplicadas dependendo do resultado pretendido e da complexidade esperada. A quantização escalar tem como base as características estatísticas dos dados de entrada sendo que cada elemento deste conjunto é quantificado isoladamente. Já na quantização vetorial isto não ocorre pois, a mesma, não considera cada elemento isoladamente, ela os agrupa em pequenos vetores e realiza um levantamento estatístico determinando quais são os vetores de dados mais apropriados para formar a base da quantização. Os quantizadores vetoriais apresentam melhores desempenhos em determinadas situações que os escalares, o que aumenta com o número de dimensões. É importante ressaltar que a etapa responsável pela alta compactação dos dados é a quantização. Podemos encontrar referências sobre quantizadores escalares em [13, 14] e sobre quantizadores vetoriais em [15, 16, 17]. No segundo passo chamado de codificação de entropia, consiste em codificar os símbolos fornecidos pelo quantizador, aplicando um algoritmo de compressão sem perdas a χ (versão da fonte com menor entropia), exemplos de tais técnicas são a codificação aritmética [9, 18]

e o código de Huffman [8, 3].

Na solução em três passos, a primeira tarefa executada é uma transformação dos dados. Como o próprio nome diz, consiste em transformar uma seqüencia de entrada $\{x_n\}$ em outra seqüencia $\{\Phi_n\}$, com o objetivo de encontrar um modelo estatístico mais apropriado para a fonte. Para recuperar a entrada, a partir do resultado da transformação, é usada a chamada transformação inversa:

$$x_n = \sum_{i=0}^{N-1} \Phi_i b_{n,i} \tag{3.1}$$

A transformação pode ser escrita na forma matricial, como mostra as equações 3.2 e 3.3.

$$\Phi = Ax \tag{3.2}$$

$$x = A^{-1}\Phi \tag{3.3}$$

Como exemplo de transformadas citaremos: A transformada cosseno discreta (DCT) [19], a transformada de walsh-hadamard (DWHT) [20], a transformada wavelet (DWT) [21, 22, 23]. O segundo passo consiste na quantização, com a qual a entropia da fonte transformada pode ser significativamente reduzida. No terceiro passo, uma codificação de entropia é aplicada, ajudando na diminuição na redundância e irrelevância dos dados quantizados.

3.2 Medidas de desempenho

Nesta sessão serão definidas as medidas de desempenho que foram utilizadas neste trabalho, direcionadas para técnicas de compressão com perdas. As imagens utilizadas para realizar os testes de simulação são imagens em nível de cinza codificadas com 255 níveis distintos.

Uma técnica de compressão pode ser avaliada por diversas maneiras, seja por critérios subjetivos, como por exemplo observações humanas, e/ou critérios objetivos que são eles: complexidade, rapidez, quantidade de memória, taxa de compressão (fidelidade à imagem original). Os critérios de desempenho escolhidos vão depender basicamente da técnica utilizada e das características da aplicação.

Os critérios subjetivos são basicamente analisados por observações humanas, levando em conta que o olho não processa tudo aquilo que observa. Tal técnica é analisada mostrando uma imagem original e a reconstruída para um grupo de pessoas, onde tira-se a média das avaliações. Estas avaliações são feitas usando uma escala definida por [24] ou através das comparações lado à lado das imagens original e reconstruída.

Os critérios objetivos são aqueles que adotam algum modelo numérico. O primeiro parâmetro objetivo a ser analisado é a taxa de compressão (*CR - Compression Ratio*) que é a relação entre o tamanho do dado original (imagem original) e o tamanho do dado após a compressão (imagem compactada). Esta relação é dada por:

$$CR = \frac{\text{Tamanho Original}}{\text{Tamanho Compactado}} \tag{3.4}$$

A taxa de compressão de uma imagem, pode ser medida em número médio de bits (*bpp - Bits Per Pixel*) necessários para representar um pixel, conhecida por taxa (R). O seu cálculo é dado por:

$$R = \frac{\text{TamImComp.8}}{\text{NumImOrig}} bpp, \tag{3.5}$$

onde TamImComp é o tamanho do arquivo da imagem compactada dada em bytes, e NumImOrig é o número de pixels da imagem original. Neste trabalho a taxa atual é sempre igual a 8, porque as imagens são codificadas a 8 *bpp*.

Como mencionado em 3.1.2, o dado reconstruído é diferente do original em esquemas de compressão com perdas, logo torna-se necessário quantificar esta diferença, chamada de distorção, neste trabalho a medida adotada de distorção será a relação sinal ruído de pico (PSNR - *Peak Signal to Noise Ratio*). Esta medida é avaliada em dB e é definida em termos do erro médio quadrático σ^2 e do máximo nível de cinza da imagem considerada x_p. A equação é mostrada em 3.6.

$$\text{PSNR} = 10 \log_{10} \frac{x_p^2}{\sigma^2} \tag{3.6}$$

Supondo que a imagem x_i e sua reconstrução y_i possuem dimensões (N, M) então σ^2 vale [3, 5, 25]:

$$\sigma^2 = \frac{1}{NM} \sum_{i=1}^{NM} (x_i - y_i)^2 \tag{3.7}$$

A PSNR pode ser expressa como:

$$\text{PSNR} = 10 \log_{10} \frac{x_p^2}{\frac{1}{NM} \sum_{i=1}^{NM} (x_i - y_i)^2} \tag{3.8}$$

onde, $x_p = 255$, N é o número de linhas e M é o número de colunas.

Os gráficos mostrados no decorrer desta dissertação serão expressos em PSNR da imagem reconstruída em relação à taxa de compressão dada em *bpp*.

Capítulo 4

O algoritmo *Multidimensional Multiscale Parser* (MMP) e suas versões

Neste capítulo será mostrado o algoritmo chamado de MMP (*Multidimensional Multiscale Parser*) [26], que consiste numa técnica de compressão com perdas baseado em casamento aproximado de padrões multiescalas. Este algoritmo apresenta algumas características interessantes: (1) ele pode ser ajustado de modo a efetuar uma compressão sem perdas; (2) seu dicionário é adaptativo porque ele é atualizado na hora em que o sinal de entrada está sendo processado; e (3) não se baseia em nenhum conhecimento prévio do sinal. Com isto é dito que este algoritmo possui um comportamento universal, pois tudo é feito de forma adaptativa sem qualquer pressuposição. A seguir serão mostrados alguns detalhes do seu funcionamento e suas versões.

4.1 Descrição do algoritmo MMP

O MMP possui um dicionário adaptativo, que é construído ao longo de todo processo de compressão. Ele é constituído por vários subdicionários de blocos com escalas diferentes. A quantidade de subdicionário é limitada e definida em função do tamanho do bloco da imagem de entrada que será processada. Se o maior bloco

de entrada possui dimensões $L \times C$, $(Linha \times Coluna)$ onde $L = 2^i$ e $C = 2^j$, o número de subdicionários e suas dimensões podem ser calculadas por:

$$Num_D = i + j + 1 \tag{4.1}$$

$$Dim[D] = \{2^{\lfloor 0.5(n+1) \rfloor} \times 2^{\lfloor 0.5n \rfloor}\}, \tag{4.2}$$

onde Num_D é o número de subdicionários, i e j são os valores que irão apontar para as dimensões do bloco da imagem, (ex:i=j=4, a dimensão do bloco será 16×16), $Dim[D]$ são as dimensões possíveis dos subdicionários e $n = \{0, 1, 2....Num_D - 1\}$.

Supondo-se que o bloco de entrada possui dimensões 16×16 logo as dimensões dos subdicionários serão $(16 \times 16), (8 \times 16), (8 \times 8), (4 \times 8), \ldots, (1 \times 1)$, como mostra a figura 4.1:

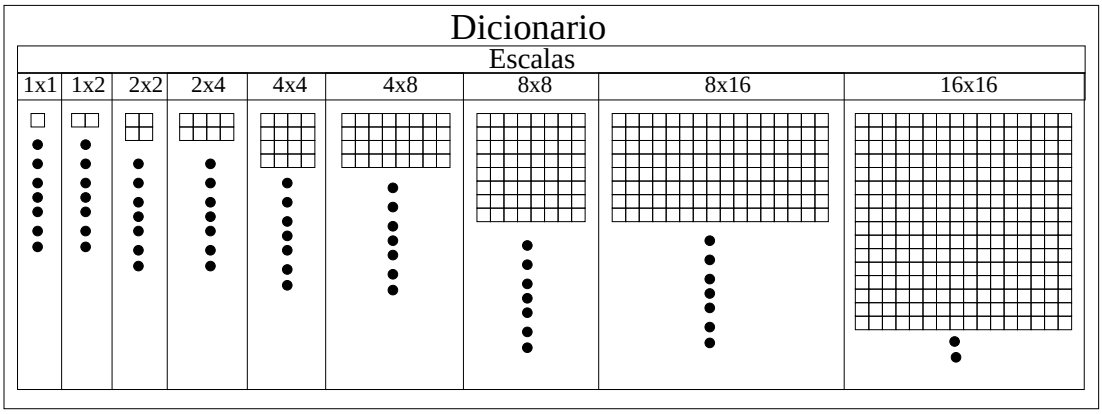


Figura 4.1: Esquema do Dicionário MMP

Como pode ser notado através da Figura 4.1, o dicionário inicial deve existir para todas as escalas. Estes pontos em preto representa as sequencias de blocos composto pela mesma dimensão para cada subdicionário, mais detalhes da inicialização deste dicionário será apresentada no final desta Seção.

Quando o MMP está efetuando a compressão em uma imagem, ele divide a imagem em blocos de dimensão $(L \times C)$, nesta implementação foi especificado dimensões de 16×16 ou (16×16 pixels), e cada bloco da imagem dividida é codificada individualmente, no sentido da esquerda para direita de cima para baixo, como pode ser observado na Figura 4.2 .

O primeiro bloco de entrada B^j, que é o bloco da imagem original, mostrado na Figura 4.2 é aproximado por elementos S^i presentes no dicionário adaptativo

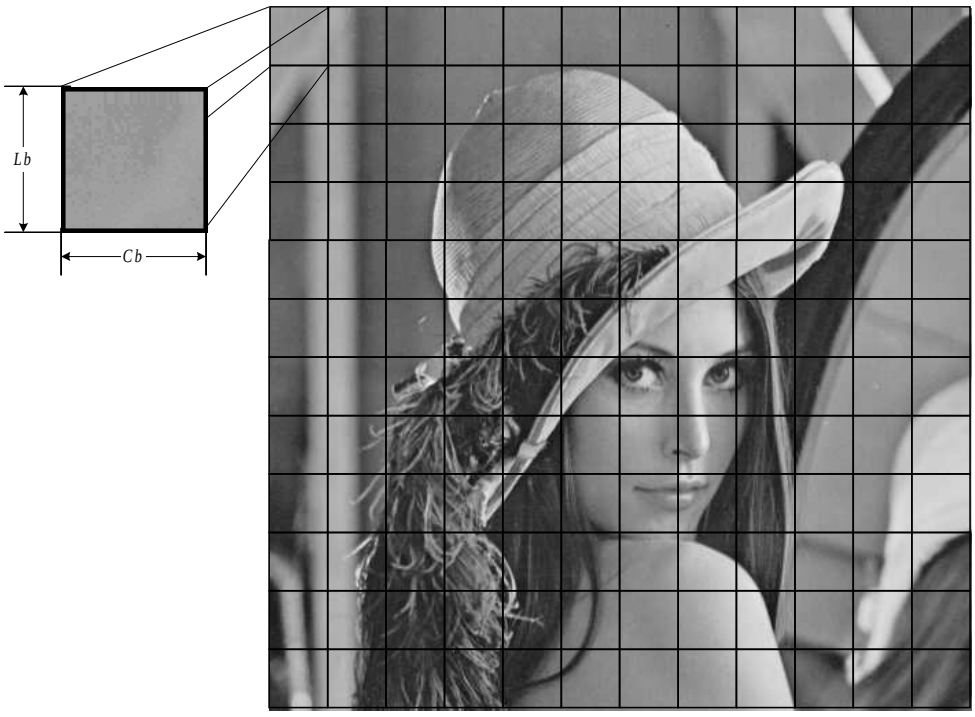


Figura 4.2: Imagem dividida em blocos para codificação

D, de mesma dimensão. A melhor aproximação é feita através da minimização do erro quadrático, dado por:

$$\delta = min\{\|B^j - S^i\|^2\} \tag{4.3}$$

A codificação é confirmada quando δ é menor que um dado valor de distorção alvo d^* ou seja, $(\delta < d^*)$, logo o bloco é codificado retornando um flag com valor igual a 1 e o índice i do elemento do dicionário que teve a menor distorção apresentada. Quando o bloco B^j não consegue nenhuma aproximação com os elementos S^i do dicionário , ou seja δ ultrapassou o valor da distorção alvo $(\delta > d^*)$, o bloco de entrada é particionado na horizontal se o número de linhas for igual ao número de colunas, caso contrário ocorrerá a partição vertical. E a busca por um elemento aproximado reinicia recursivamente para os dois novos blocos, B^{2j+1} e B^{2j+2}. Com isto o bloco particionado retorna um flag com valor igual a zero, indicando que houve uma partição. Observando-se o comportamento do algoritmo onde toda vez que algum bloco é segmentado, geram-se dois novos blocos e assim sucessivamente, até que $(\delta < d^*)$ ou os blocos resultantes tenham dimensões $L = C = 1$, logo é possível associar este comportamento a uma estrutura em árvore binária. Como

pode ser verificado na figura 4.3.

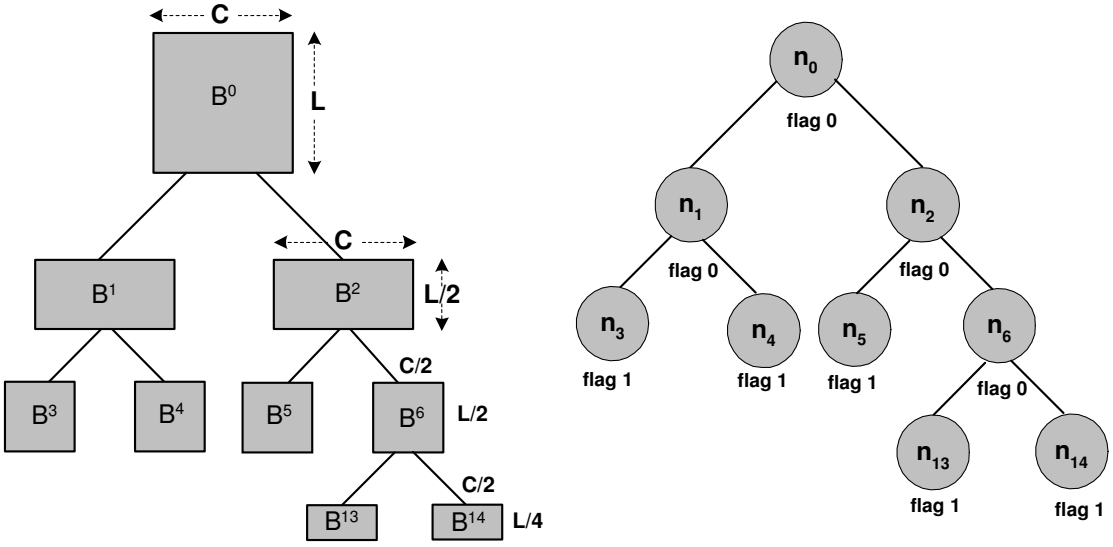


Figura 4.3: Partição de blocos no MMP e sua respectiva Árvore binária $A(n_0)$

Na Figura 4.3, onde mostra a árvore binária $A(n_0)$, o nó n_j é associado ao bloco B^j, como exemplo, o bloco (B^o) corresponde ao nó (n_0), e os nós que não foram segmentados são chamados de nós folhas, nós que não possuem filhos, e representam as codificações ocorridas.

A seqüência da formação da árvore de segmentação é dada pelos seguintes nós $n_0, n_1, n_3, n_4, n_2, n_5, n_6, n_{13}, n_{14}$ ou pode ser representada pela sua seqüência de *flags* $0, 0, 1, 1, 0, 1, 0, 1, 1$.

A atualização do dicionário oficial D, ocorre quando dois nós filhos n_{2j+1} e n_{2j+2}, descendentes de um mesmo nó pai n_j, já tiverem sido codificados. A atualização é realizada com a concatenação dos elementos do dicionário S^i associados aos nós codificados $\hat{B}^j$, atribuindo assim novos elementos a todas as escalas. Esta atualização em escalas diferentes só é possível através de uma transformação de escala do tipo:

$$S^s = T_{L_0}^{L}\left[S\right], \tag{4.4}$$

onde S é um vetor de tamanho L_0 e L é a nova escala do vetor S^S, isto para uma

transformação unidimensional T.

Através da transformação escalar, pode-se dilatar ou contrair vetores, como descrito nas Equações 4.5 e 4.6 respectivamente, definido por [26, 27].

- **Dilatação Vetorial** ($L_0 < L$): Considerando $n = 0, 1, \ldots, L-1.$, como coordenadas do vetor transformado, então define-se que o vetor transformado é dado por:

$$S_n^s = \left\lfloor \frac{\alpha_n \left(S_{m_n^1} - S_{m_n^0} \right)}{L} \right\rfloor + S_{m_n^0} \tag{4.5}$$

onde $\lfloor a \rfloor$ = menor inteiro maior que a:

$$\begin{aligned} m_n^0 &= \left\lfloor \frac{n\,(L_0 - 1)}{L} \right\rfloor \\ m_n^1 &= \begin{cases} m_n^0 + 1 & , \quad m_n^0 < L_0 - 1 \\ m_n^0 & , \quad m_n^0 = L_0 - 1 \end{cases} \\ \alpha_n &= n\,(L_0 - 1) - L m_n^0 \end{aligned}$$

- **Contração Vetorial** ($L_0 > L$): Considerando $n = 0, 1, \ldots, L-1.$, como coordenadas do vetor transformado, então define-se que o vetor transformado é dado por:

$$S_n^s = S_{m_{n,k=0}^0} + \frac{1}{L_0 + 1} \sum_{k=0}^{L_0} \left\lfloor \frac{\alpha_{n,k} \left(S_{m_{n,k}^1} - S_{m_{n,k}^0} \right)}{L} \right\rfloor \tag{4.6}$$

onde,

$$\begin{aligned} m_{n,k}^0 &= \left\lfloor \frac{n\,(L_0 - 1) + k}{L} \right\rfloor \\ m_{n,k}^1 &= \begin{cases} m_{n,k}^0 + 1 & , \quad m_{n,k}^0 < L_0 - 1 \\ m_{n,k}^0 & , \quad m_{n,k}^0 = L_0 - 1 \end{cases} \\ \alpha_{n,k} &= n\,(L_0 - 1) + k - L m_{n,k}^0 \end{aligned}$$

Tratando-se de blocos de imagens é necessário uma transformação bidimensional definida por:

$$S^s = T^{L,C}_{L_0,C_0}[S] \tag{4.7}$$

onde S é o bloco de dimensões $L_0 \times C_0$ e $L \times C$ são as novas dimensões do bloco S^S. Esta transformação bidimensional é realizada usando uma aplicação dupla do caso unidimensional onde, primeiramente transformamos todas as linhas segundo a Equação 4.8 com $l(S_i)$ sendo o tamanho de cada linha e o número de linhas variando de $i = 0, 1, \ldots, l(S_i) - 1$.

$$Y_i = T_C[S_i] \tag{4.8}$$

A transformação das colunas é realizada usando a transposta de Y_i obtida na Equação 4.8 com o número de colunas variando entre $j = 0, 1, \ldots, M - 1$.

$$S^s_j = \left(T_L\left[\left(Y^T\right)_j\right]\right)^T \tag{4.9}$$

Logo para converter um bloco 16×16 em 4×8, a transformação de escala modifica o comprimento de um bloco através de sua contração ou dilatação, permitindo a aproximação de elementos com comprimento fixo por elementos com comprimento variável [27, 28].

A Figura 4.4 mostra a atualização do dicionário de um bloco da imagem com dimensões 16×16 *pixels*, onde cada escala do dicionário é atualizada com versões contraídas ou dilatadas do bloco (16×16) concatenado. Observa-se que a transformação escalar resulta em outros blocos de dimensões diferentes com pixels diferentes do bloco que está sendo concatenado e não a divisão do bloco 16×16 em outras escalas.

Vale ressaltar que o dicionário inicial D_i, não pode ser muito grande nem tão pouco muito pequeno, pois provoca erros de reconstrução. Como exemplo, para imagens mais complexas (imagens que contém gráficos, figuras e texto), que são comprimidas a uma taxa de 0,5 *bpp* exigem valores maiores de distorção alvo d^*, que necessitam de dicionários iniciais menores. Com isso percebe-se que o tamanho do dicionário inicial depende da distorção alvo d^*.

A formação do dicionário é mostrado na Equação 4.10 seguindo os passos descritos a seguir.

$pixelmin = 0$ (valor de pixel mínimo),

$pixelmax = 255$ (valor de pixel máximo),

$El = \lfloor \frac{2.(pixelmax - pixelmin)}{d^*} \rfloor$,

$Of = \frac{(pixelmax - pixelmin)}{El - 1}$,

Para $n = (0, 1, \ldots, Num_D - 1)$

$L = 2^{\lfloor 0.5(n+1) \rfloor}$,

$C = 2^{\lfloor 0.5n \rfloor}$,

$$D_i^{L,C} = \{T_{1,1}^{L,C}[0], T_{1,1}^{L,C}[\lfloor 1.Of \rfloor], ..., T_{1,1}^{L,C}[\lfloor (El-2).Of \rfloor], T_{1,1}^{L,C}[255]\}, \qquad (4.10)$$

onde tem-se que El é o número de elementos do dicionário inicial, Of é o *offset* para a geração dos elementos do dicionário e $D_i^{L,C}$ é o dicionário inicial com blocos de dimensões $(L \times C)$.

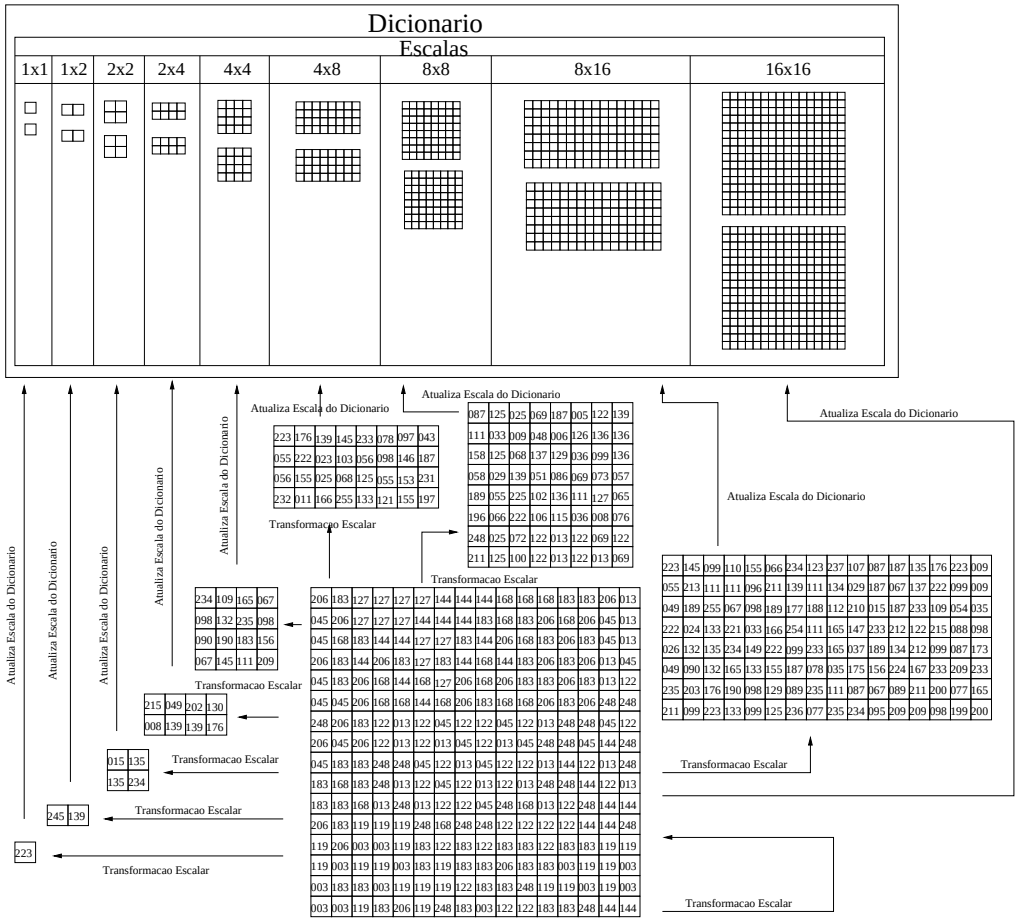


Figura 4.4: Atualização do dicionário

No decodificador a atualização do dicionário é feita da mesma forma, e as informações que ele precisa para realizar a reconstrução da imagem corretamente são os *flags* '1' e '0' resultantes da codificação da árvore de segmentação e os índices i do dicionário correspondentes às aproximações dos nós-folhas. Um detalhe importante é que esses índices e flags são codificados por um codificador aritmético [18], proporcionando assim um desempenho bem melhor. O cabeçalho do arquivo que contêm a imagem que será processada, é escrito tanto no codificador como no decodificador, sem a necessidade de se acrescentar *overhead* no arquivo.

Para maiores detalhes dos algoritmos de codificação e decodificação do MMP verificar nas referências [26, 27].

4.2 O MMP com Otimização Taxa-Distorção

Como visto na Sessão 4.1, o algoritmo MMP possui um critério de distorção alvo d^* estipulada para os blocos, decidindo assim se um bloco será codificado ou não. Se o bloco aproximado causar uma distorção maior que a distorção alvo, o bloco de entrada é particionado em dois outros iguais; caso contrário, não. Como um exemplo será mostrada a Figura 4.5, de uma árvore de segmentação com seus respectivos *flags* e índices, e as menores distorções encontrada em cada bloco (ou nó).

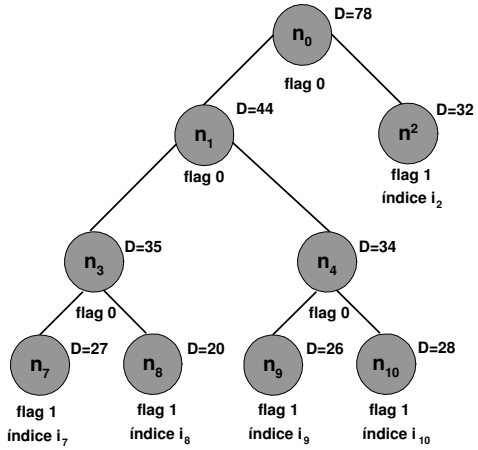


Figura 4.5: Árvore de segmentação $A(n_0)$ e suas distorções

Considerando-se uma distorção alvo $d^* = 33$, para a árvore $A(n_0)$ da Figura 4.5, fica evidente que os nós n_3 e n_4, possuem uma distorção que se aproxima da distorção alvo $d^* = 33$ estipulada, mas como os valores de distorção são um pouco maiores a decisão tomada é partir os nós. Com esta decisão aumenta o consumo de *bits* para a codificação, pois ao invés de codificar 5 *flags* e 3 índices será codificado 9 *flags* e 5 índices. Com isto percebe-se que este critério de decisão é local pois leva-se em conta apenas o bloco em questão não se preocupando com o consumo de índices e *flags* para a codificação de tal bloco. Para um melhoramento deste resultado é necessário avaliar a distorção e a taxa (*bits* necessários para codificação), de forma global ou seja estendido para toda a árvore $A(n_0)$, analisando se vale apena dividir ou não o bloco. Esta avaliação de distorção e taxa é feita por uma variação do MMP chamada de MMP-RDI.

Em [29], foi implementado um algoritmo de otimização intermediária chamado de MMP-RDI que tem a preocupação de avaliar a distorção e a taxa por um critério de decisão global que é baseado no custo de *Lagrange* [30, 31]. Várias aplicações deste método podem ser encontradas em [32, 33, 34].

Este critério define uma relação de compromisso entre a taxa e a distorção para uma sub-árvore, que pertence a uma árvore de otimização $A(n_0)$ e é dado pela Expressão 4.11.

$$J_{n_x} = D_{n_x} + \lambda RI_{n_x} \tag{4.11}$$

onde D_{n_x}, é a distorção entre o bloco B_x (ou nó n_x) e a aproximação dos elementos do dicionário de mesma dimensão S_i; λ é o fator ponderador entre a taxa e a distorção; RI_{n_x} é a taxa ou número de *bits* para a representação do índice i do bloco aproximado. A expressão da taxa é dada por, $RI_{n_x} = -\log_2 P(n_x)$ onde $P(n_x)$ é a probabilidade de ocorrência de n_x, logo podemos reescrever como:

$$RI_{n_x} = \log_2 \left(\frac{\text{total acumulado para } i}{\text{frequência de } i} \right) \tag{4.12}$$

Agora, ao invés de analisarmos somente a distorção de um determinado nó, será analisado o custo deste nó na árvore de segmentação. Através deste custo é que será tomada a decisão de podar ou não o nó. Neste momento, calcula-se os custos do nó-pai n_x e dos nós-filhos n_{2x+1} *e* n_{2x+2}. Se o custo do-nó pai n_x for menor que a soma dos custos dos nós-filhos n_{2x+1} *e* n_{2x+2}, estes serão podados e a representação do bloco imagem será dada pelo nó-pai.

O custo do nó-pai é calculado pela equação 4.13 abaixo:

$$J_{n_x} = J_{n_x} + \lambda R_1(n_x) \tag{4.13}$$

onde J_{n_x} é o custo *Lagrangeano* do nó-folha (ou nó-pai) e $R_1(n_x)$ é o número de *bits* necessário para representar o *flag* '1', indicando que o nó n_x é um nó-folha.

Como nos nós-folhas, o par de nós filhos n_{2x+1} *e* n_{2x+2} devem considerar os *flags* descritos na árvore de segmentação para a sua devida representação, seu cálculo é mostrado abaixo.

$$J_{2n_x+1,2n_x+2} = J_{2n_x+1} + J_{2n_x+2} + \lambda R_0(n_x) \tag{4.14}$$

onde $J_{2n_x+1,2n_x+2}$ é o custo *Lagrangeano* dos nós-filhos n_{2x+1} *e* n_{2x+2} e $R_0(n_x)$ é o número de *bits* necessário para representar o *flag* '0', indicando que ocorreu uma segmentação no nó n_x.

O algoritmo MMP-RDI começa com uma árvore completa $\alpha(n_0)$ com todos os seus nós folhas, partindo dos nós folhas até o nó n_0, a profundidade da árvore completa é calculada por $1 + log_2(Linha) + log_2(Coluna)$. Sempre seguindo o critério de decisão do custo *Lagrangeano*, podando o par de nós-filhos n_{2x+1} *e* n_{2x+2} quando o custo *Lagrangeano* da árvore que os contem for maior que o custo da árvore de segmentação sem os nós-filhos. Quando a poda dos nós-filhos ocorre, as atualizações do dicionário geradas pelas suas concatenações devem ser tiradas, e para calcular os custos de *Lagrange* de forma correta deve-se manter um registro da freqüência que cada vetor foi utilizado, bem como o número de ocorrências de cada *flag*. Com isto tem-se algumas considerações:

(1) Utiliza-se um dicionário rascunho D_R, independente do dicionário oficial D que simula todas as atualizações que ocorreriam na codificação;

(2) São criados contadores C_{D_R} para as freqüências dos índices dos blocos do dicionário rascunho D_R, onde esses valores serão utilizados para o cálculo dos custos, quando algum elemento pertencente ao dicionário rascunho for o escolhido durante a otimização para representar um bloco; e

(3) São criados contadores complementares $\hat{C}_D$ e $\hat{C}_{D_F}$ que irão guardar as respectivas freqüências adicionais dos índices do dicionário oficial e *flags* provindos da árvore de segmentação, adequando sempre o modelo de freqüências ao molde da árvore de segmentação.

A transformação escalar utilizada pelo algoritmo MMP é a mesma para o MMP-RDI, no entanto devido à utilização da segmentação otimizada no MMP-RDI, o dicionário inicial perde parte da dependência da distorção alvo d^* e passa a ter 64 vetores e tais elementos são igualmente espaçados definido pela variável El entre 0 e 252 para todas as imagens. A formação do dicionário inicial do MMP-RDI é definida por:

$El = 64$,
$Of = \lfloor \frac{255}{El-1} \rfloor = 4$,
Para $n = (0, 1, \ldots, Num_D - 1)$
$L = 2^{\lfloor 0.5(n+1) \rfloor}$,
$C = 2^{\lfloor 0.5n \rfloor}$,

$$D_i^{L,C} = \{T_{1,1}^{L,C}[0], T_{1,1}^{L,C}[4], ..., T_{1,1}^{L,C}[248], T_{1,1}^{L,C}[252]\} \tag{4.15}$$

O MMP-RDI torna-se um espelho da codificação, o algoritmo executa todos os passos da codificação, fazendo simulações das atualizações do dicionário, codificações de índices e *flags*, realizando assim uma predição do comportamento da codificação. Este comportamento similar à codificação é efetuada pela função otimização que é executada antes da codificação com o objetivo de determinar o melhor molde para a árvore de segmentação.

As próximas versões que serão apresentadas terão este comportamento espelhado da codificação em suas implementações.

4.3 O algoritmo MMP-U

No MMP padrão executam-se partições no sinal de forma que encontre um elemento do dicionário que o represente de forma satisfatória. Estas partições são associadas à uma árvore binária, conforme mencionado na Sessão 4.1, que pode ser otimizada em relação a taxa-distorção. Contudo, esta estrutura em árvore binária não analisa a dependência de nós-filhos com nós-pais diferentes, como mostra a Figura 4.6.

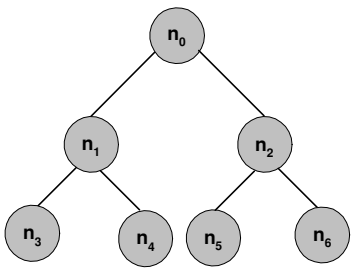


Figura 4.6: Árvore binária

Na Figura 4.6 da árvore binária observa-se que pelo comportamento do MMP original não existe nenhuma relação dos nós-filhos n_3, n_4, n_5, n_6, pelo fato de serem de pais diferentes, apesar destes nós-filhos possuírem a mesma informação. Com estas considerações, foi proposto em [35] um algoritmo que é capaz de analisar essas dependências, as uniões destes blocos, além de permitir uma extensão da partição do MMP original através dos elementos que serão codificados, tal algoritmo é chamado de MMP-U que realiza as uniões de blocos, que foi baseado num método de compressão de sinais chamado *prune-join* proposto em [36], com objetivo de aproximar funções unidimensionais por funções polinomiais por partes.

Para os algoritmos da classe MMP, este método *prune-join* foi extendido para o caso bidimensional que não é tão simples, pois a codificação conjunta dos blocos deve ser realizada somente com blocos que possuem as mesmas dimensões horizontais ou verticais, sempre apresentando o resultado do bloco retangular, e satisfazendo o critério de menor custo *Lagrangeano*. Este algoritmo é dividido em duas partes principais, a primeira é a obtenção da árvore otimizada e a segunda se refere a avaliação de junção dos blocos.

Na primeira parte é obtida à árvore otimizada no sentido taxa-distorção como comentada na Seção 4.2; na avaliação de junção é realizada a união dos nós-filhos que possuem nós-pais diferentes, onde os blocos associados estão localizados à direita ou abaixo do bloco que está sendo analisado de acordo com o sentido da formação da árvore de segmentação, sempre seguindo o critério de decisão do custo de *Lagrange*. Nas Figuras 4.7 e 4.8 será mostrado uma árvore e seu bloco associado antes da união e depois da união respectivamente.

Na Figura 4.7 vemos a árvore A_{n0} com seu bloco correspondente à direita, onde será aplicado o método podagem- união (*prune-join*), nos nós-filhos que possuem nós-pais diferentes, lembrando que esta junção só será feita com blocos que possuem as mesmas dimensões.

Na Figura 4.8 é mostrada a árvore resultante depois de ser analisada pelo método *prune-join*, observando que ouve economia nos *bits* para serem codificados comparado com a Figura 4.7.

Mais detalhes deste método será encontrado em [35, 36].

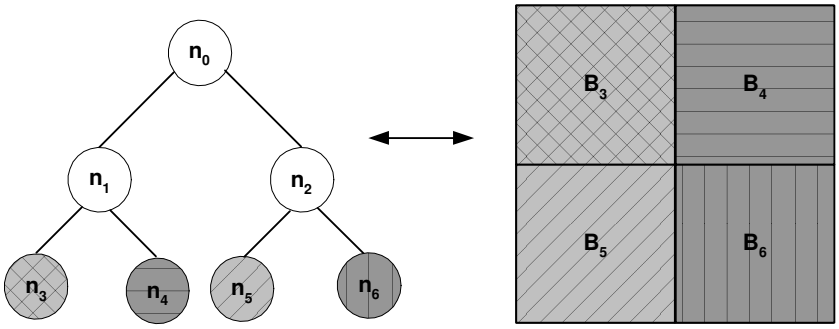


Figura 4.7: Árvore binária e seu bloco correspondente B

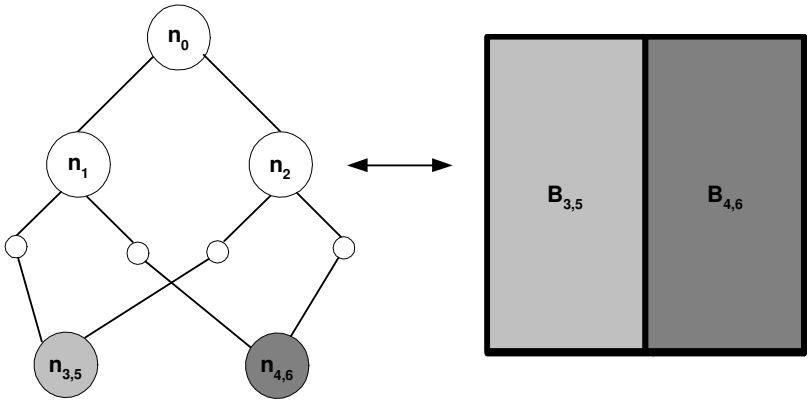


Figura 4.8: Árvore binária resultante e bloco que será codificado B

4.4 O algoritmo SM-MMP

Tratando-se do MMP original, o sinal é divido em blocos e cada bloco de sinal é analisado e codificado de maneira independente, sem se preocupar com as características dos seus blocos vizinhos previamente codificados. Em [29] foi proposto um algoritmo chamado Side-match MMP (SM-MMP), que utiliza as informações dos blocos vizinhos que já foram codificados, com o principal objetivo de fazer uma predição dos elementos do dicionário mais prováveis para a codificação do bloco que está sendo analisando. Na Figura 4.9 é ilustrado o bloco B^j que será codificado e seus vizinhos B_S^j e B_E^j.

A Figura 4.9 mostra o bloco B^j com dimensões $(L \times C)$ que é associado à um nó n_j, da árvore de segmentação. O bloco vizinho superior B_S^j é definido por dimensões $(L \times C)$ que se localiza acima de B^j e o vizinho esquerdo B_E^j também

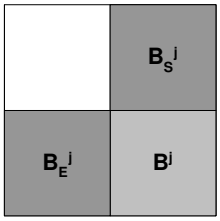


Figura 4.9: bloco B^j com seus vizinhos superior e esquerdo

definido por dimensões $(L \times C)$ localizado à esquerda de B^j. Observa-se que os blocos B_S^j e B_E^j são codificados antes do bloco B^j, pois a ordem que o MMP processa os blocos de entrada é no sentido da leitura de cima para baixo da esquerda para direita.

É utilizado no MMP original um dicionário que contém blocos de dimensões $(L \times C)$ para codificar o bloco de sinal de entrada B^j, contudo no algoritmo Side-match MMP usa-se um critério de continuidade onde a probabilidade dos elementos escolhidos pelo dicionário obedece a este critério. Na implementação deste algoritmo foi adotado que os elementos do dicionário que não obedecem à este critério de continuidade é atribuída a probabilidade de ocorrência igual a zero, levando a redução do tamanho do dicionário original. Este novo dicionário é nomeado como dicionário de estado. Sua construção é baseada na medida de rugosidade $r_{ij} = R(B_S^j, B_E^j, \hat{B}^i)$ essa medida estabelece a inclusão ou não de um elemento no dicionário de estado, esta medida é calculada por:

$$\begin{aligned} r_{ij} &= \sum_{k=0}^{m-1} \left| \left\lfloor \frac{B^j_{S_{(m-2,k)}} - B^j_{S_{(m-1,k)}} + B^{\hat{i}}_{(o,k)} - B^{\hat{i}}_{(1,k)}}{2} \right\rfloor + B^{\hat{i}}_{(o,k)} - B^j_{S_{(m-1,k)}} \right| \\ &+ \sum_{p=0}^{n-1} \left| \left\lfloor \frac{B^j_{E_{(p,n-2)}} - B^j_{E_{(p,n-1)}} + B^{\hat{i}}_{(p,0)} - B^{\hat{i}}_{(p,1)}}{2} \right\rfloor + B^{\hat{i}}_{(p,0)} - B^j_{S_{(p,n-1)}} \right| \end{aligned} \quad (4.16)$$

O algoritmo SM-MMP escolhe um dicionário de estado D_{est} formado pelos elementos pertencentes ao dicionário D que possuem as menores medidas de rugosidades r_{ij}, baseado na equação acima 4.16. O tamanho do dicionário de estado D_{est} vai depender do nível da atividade dos blocos B_S^j e B_E^j. A função atividade $AC(B)$ de um bloco B que possui dimensões $(l \times c)$ é definida por:

$$AC(B) = max_{l,c}\left\{\left(\sum_{k=0}^{c-1}|B_{l+1,k} - B_{l,k}|\right), \left(\sum_{p=0}^{l-1}|B_{p,c+1} - B_{p,c}|\right)\right\} \quad (4.17)$$

O tamanho M do dicionário de estado utilizado para se codificar um dado bloco B^j é determinado através do cálculo:

$$M = M_{max}\left(\frac{AC(B_S^j) + AC(B_E^j)}{2}\right) AC_{max}^{-1} \quad (4.18)$$

onde AC_{max} é o maior valor de atividade $AC(B)$ dentre todos os blocos de entrada de maior hierarquia da imagem e M_{max} é o tamanho máximo permitido para o dicionário de estado.

No SM-MMP somente o primeiro bloco é codificado com o MMP original que utiliza a otimização intermediária. Este bloco é definido como bloco fundamental e para ser codificado ele utiliza o dicionário oficial D.

4.5 Resultados de simulações

Nesta sessão serão mostrados os resultados das simulações dos algoritmos MMP, MMP-RDI, MMP-U e SM-MMP. Tais algoritmos foram implementados em linguagem C, e executados em ambiente *Linux*. Estes resultados são mostrados por curvas taxa-distorção onde na coordenada abcissa temos a taxa dada em *bits* por *pixel* (*bbp*) e na coordenada ordenada a distorção dada pela PSNR, como definido anteriormente na Sessão 2.2.

As imagens para o teste foram, lena, aerial, gold, barbara, f16, pp1205,pp1209, todas com tamanho de (512×512) *pixels*, no formato *pgm*. Estas imagens foram adquiridas no *site* http://sipi.usc.edu/service/database/Database.html com exceção das imagens pp1205 e pp1209 que foram digitalizadas do IEEE *Transactions on Image Processing*, volume 09, número 07, mês de julho, ano 2000, as páginas correspondem aos nomes das imagens. Todas essas imagens foram divididas em blocos de (16×16) para serem processadas bloco a bloco pelo algoritmo MMP e suas versões na seqüencia da esquerda para direita e de cima para baixo.

Para análise de comparação com os resultados do MMP original e suas versões, serão apresentados os resultados referentes aos algoritmos JPEG [6] baseado em DCT e SPIHT [37] baseado em Wavelets.

As figuras apresentadas a seguir mostram os resultados obtidos para cada imagem já especificada, e elas são ampliadas com o intuito de melhorar suas análises. Pois alguns resultados das curvas dos algoritmos aproximam-se uma da outra dificultando a visualização dos resultados de simulação, como mostra a Figura 4.10, que representa os resultados da figura lena, e a Figura 4.11 que mostra a sua ampliação . Por questão de simplicidade, tais figuras serão referenciadas pelos seus respectivos nomes que estão localizados em cima de cada figura exposta. As imagens originais podem ser observadas no apêndice A.

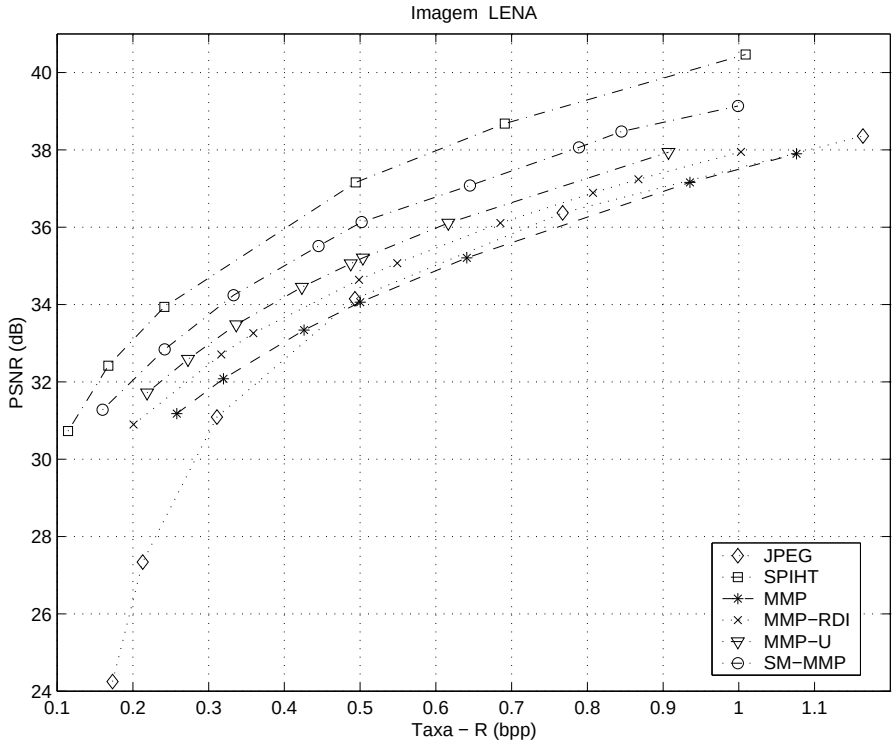


Figura 4.10: Taxa x distorção para a imagem Lena 512 x 512

Nos gráficos que se referem as imagens lena, aerial, gold, bárbara e f16 que são imagens suaves o SPIHT apresentou os melhores resultados, pois este algoritmo assume que as imagens que serão processadas por ele possuem uma alta concentração de energia em freqüências mais baixas, com uma queda em freqüências mais altas, o que não acontece com as imagens PP1205 e PP1209. Logo seu desempenho nessas imagens foram inferiores comparado com os algoritmos baseados no MMP.

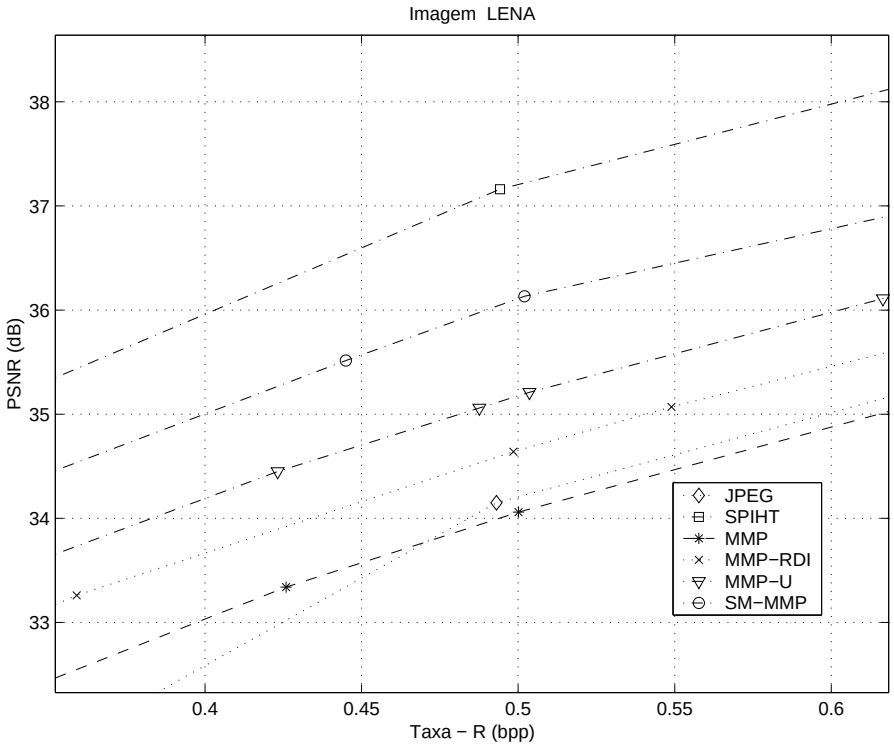


Figura 4.11: Zoom da Taxa x distorção para a imagem Lena 512 x 512

Os algoritmos baseados no MMP não fazem qualquer predição com as imagens que serão processadas, adaptando o seu dicionário ao tipo de dado que está sendo codificado. Apesar da imagem aerial ser uma imagem suave, os algoritmos da família MMP superaram em PSNR o algoritmo JPEG e o algoritmo SM-MMP a uma taxa de 0.6 *bpp* chega bem próximo ao resultado do SPIHT, sem fazer nenhuma predição de tal imagem, é o que mostra as Figuras 4.12 e 4.13.

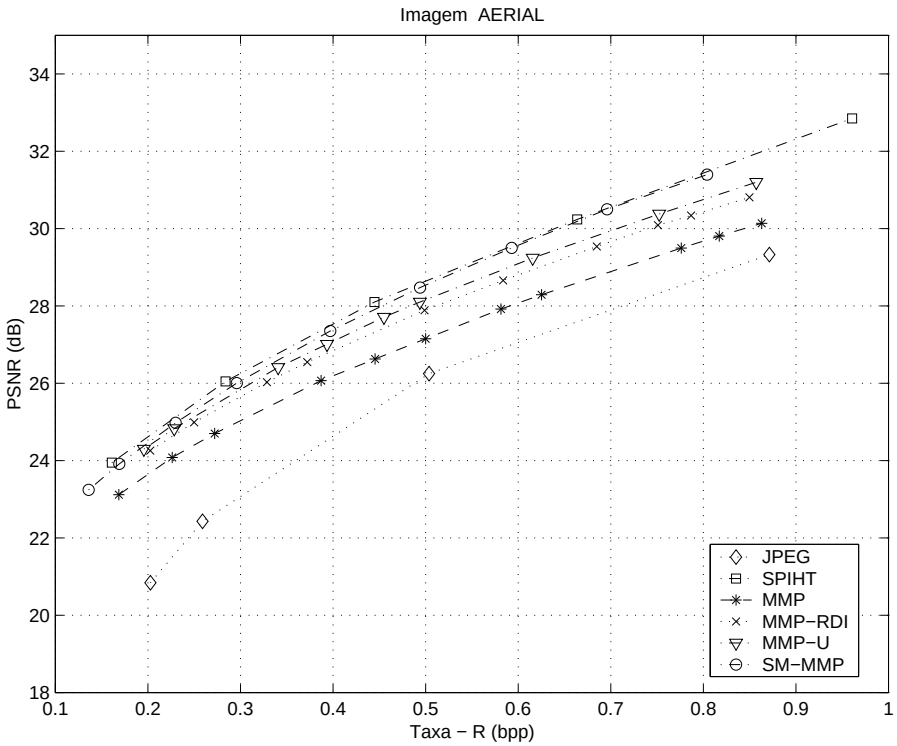


Figura 4.12: Taxa x distorção para a imagem Aerial 512 x 512

Analisando os algoritmos da classe MMP, observa-se que o SM-MMP apresentou melhores resultados em quase todas as imagens, devido ao fato de que na sua implementação ser realizado o casamento lateral com seus dois blocos vizinhos superior e esquerdo, com exceção na imagem PP1205 que praticamente o desempenho igualou-se aos algoritmos MMP-RDI e MMP-U.

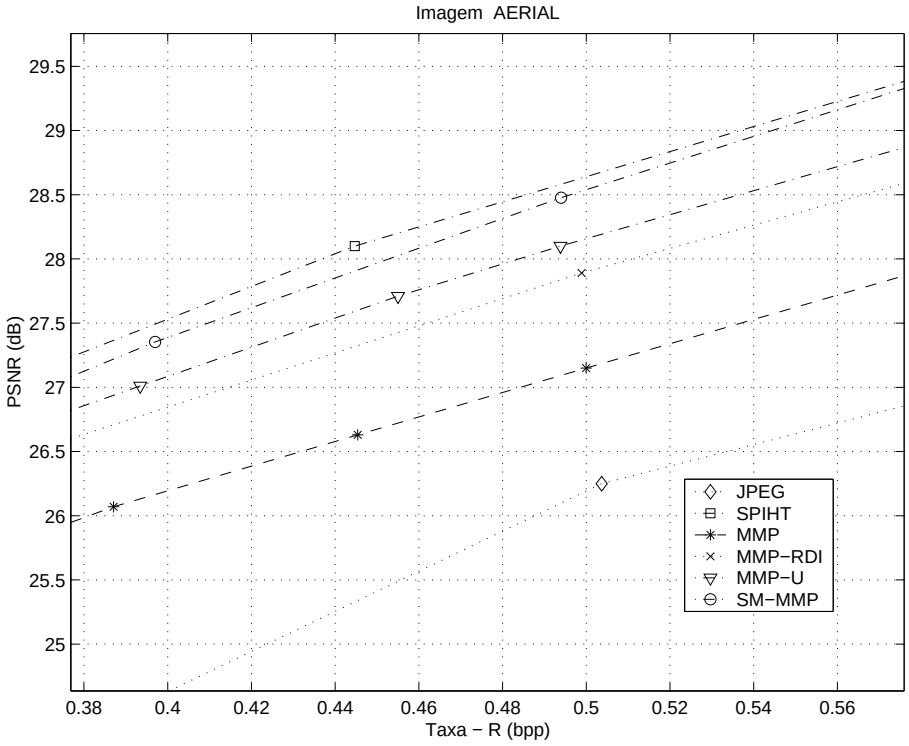


Figura 4.13: Zoom da Taxa x distorção para a imagem Aerial 512 x 512

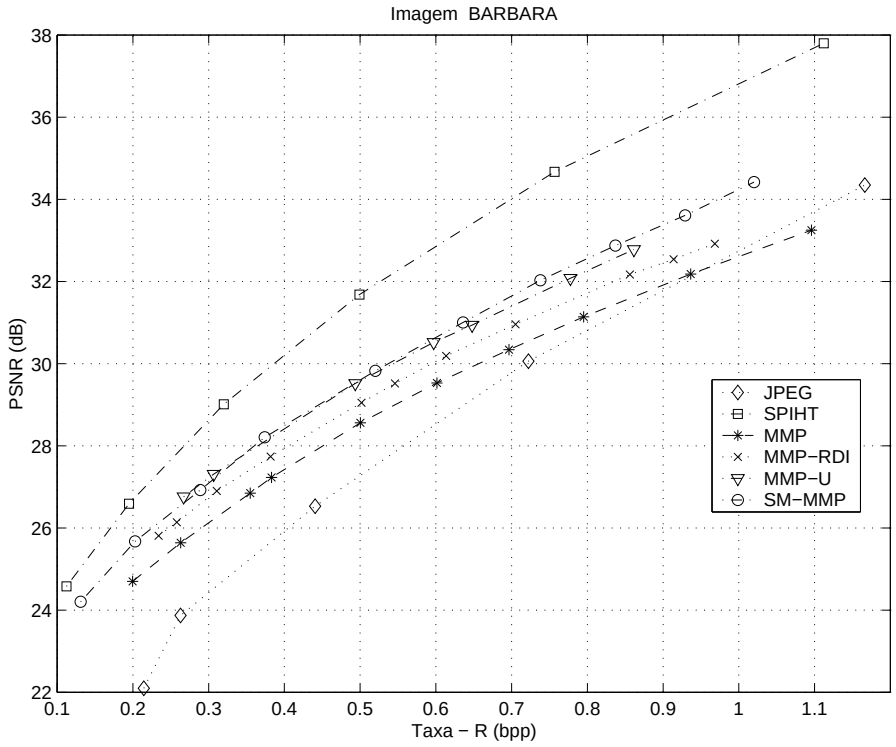


Figura 4.14: Taxa x distorção para a imagem Bárbara 512 x 512

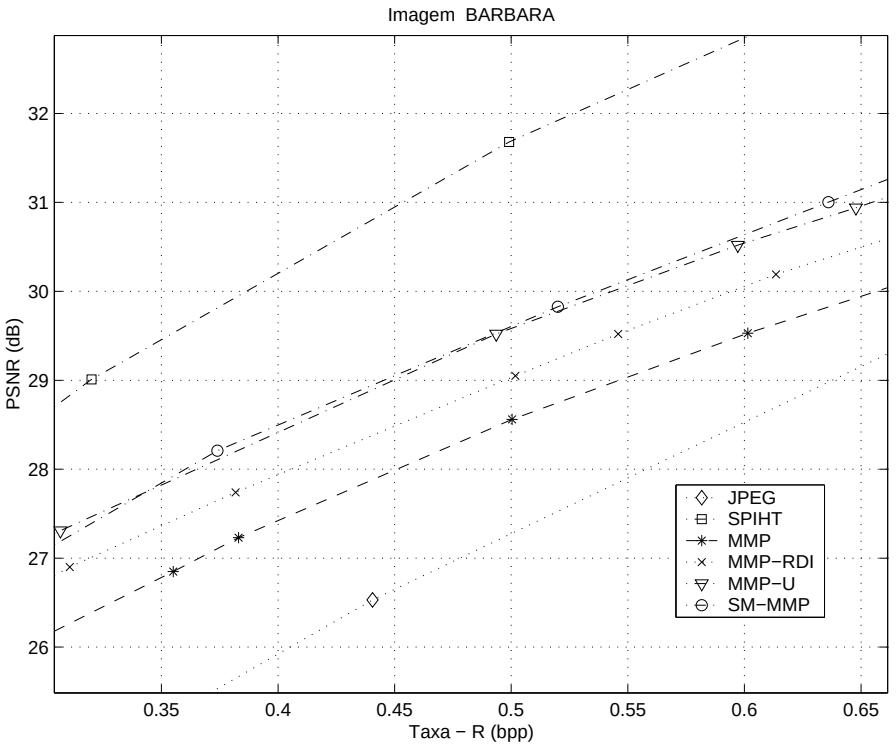


Figura 4.15: Zoom da Taxa x distorção para a imagem Bárbara 512 x 512

Na Figura 4.17, que mostra a ampliação da Figura 4.16, para ser melhor analisada, verifica-se que entre as taxas 0.2 e 0.5 *bpp* as versões do algoritmo MMP atingiram resultados muito bons em PSNR comparando com o algoritmo SPIHT para imagens suaves que é o caso da imagem gold.

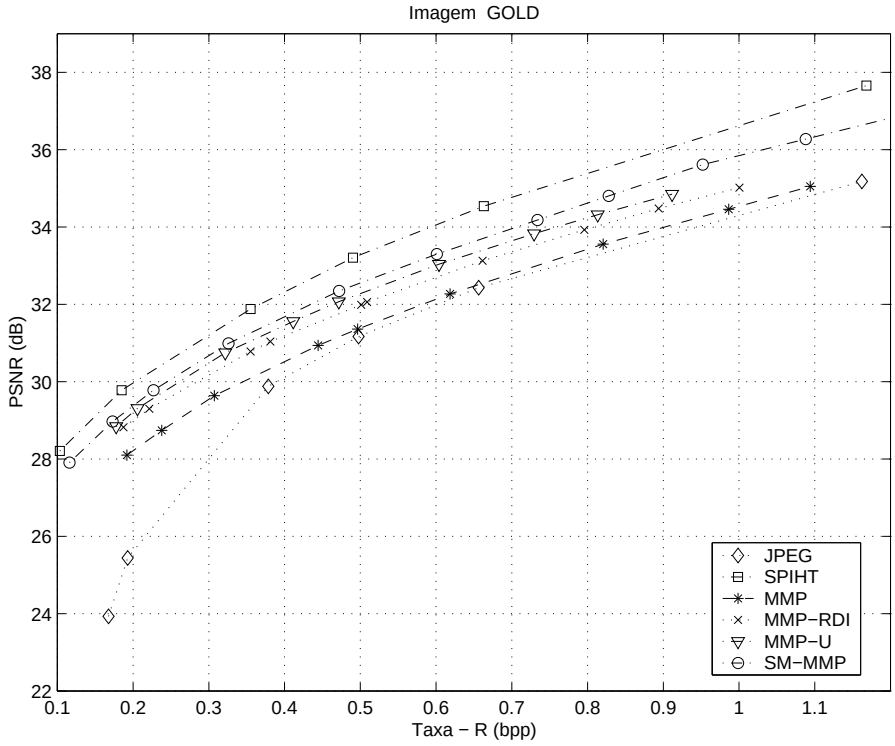


Figura 4.16: Taxa x distorção para a imagem Gold 512 x 512

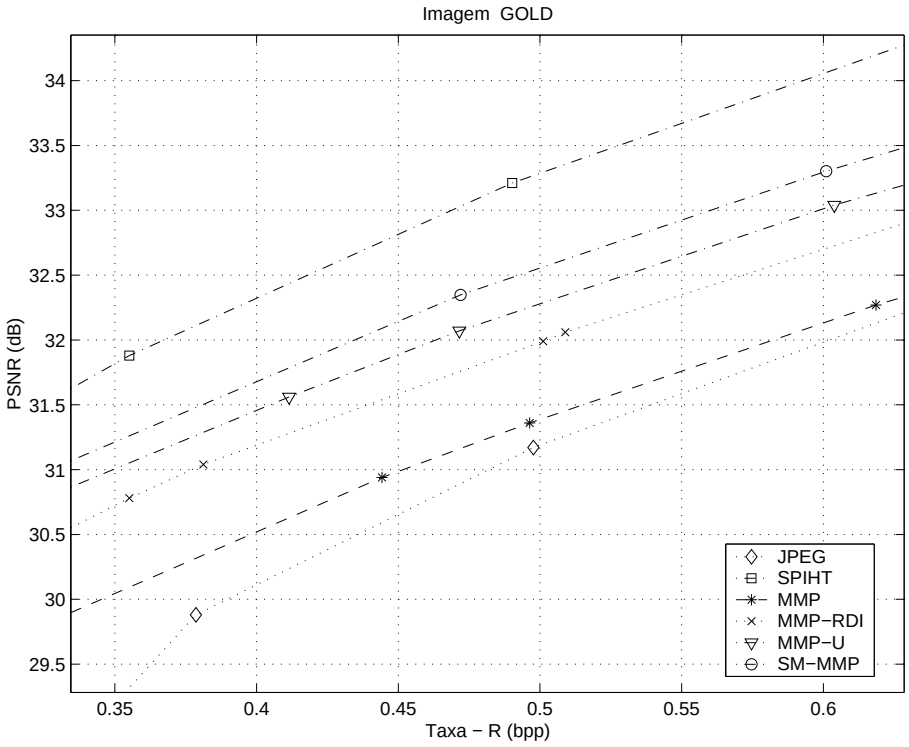


Figura 4.17: Zoom da Taxa x distorção para a imagem Gold 512 x 512

Observa-se que na imagem F16 a versão SM-MMP chega bem próximo dos resultados obtidos pelo algoritmo baseado em *Wavelets* que no caso o SPIHT.

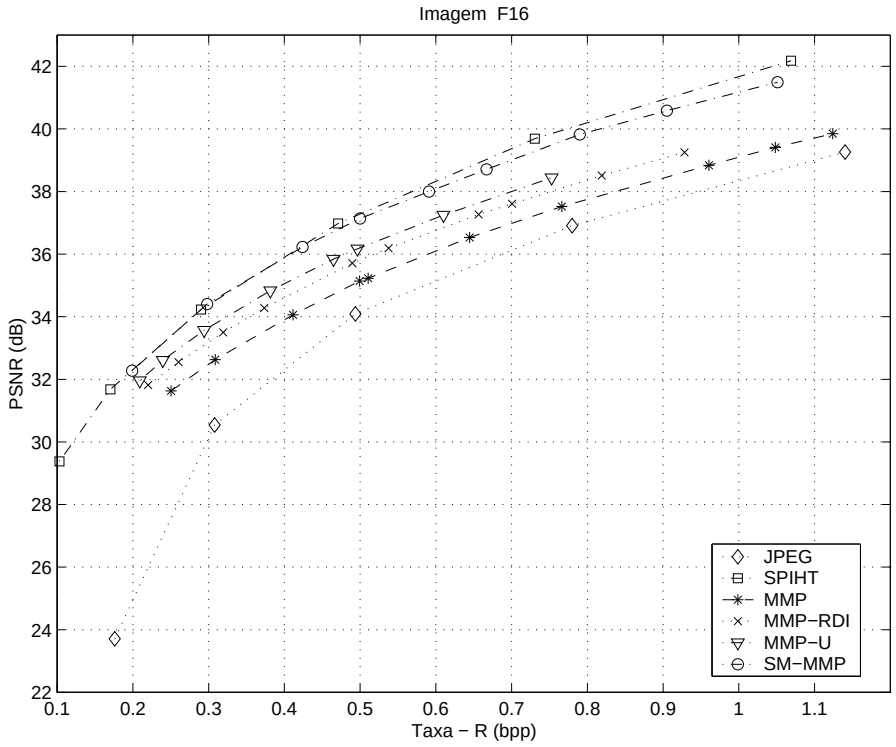


Figura 4.18: Taxa x distorção para a imagem F16 512 x 512

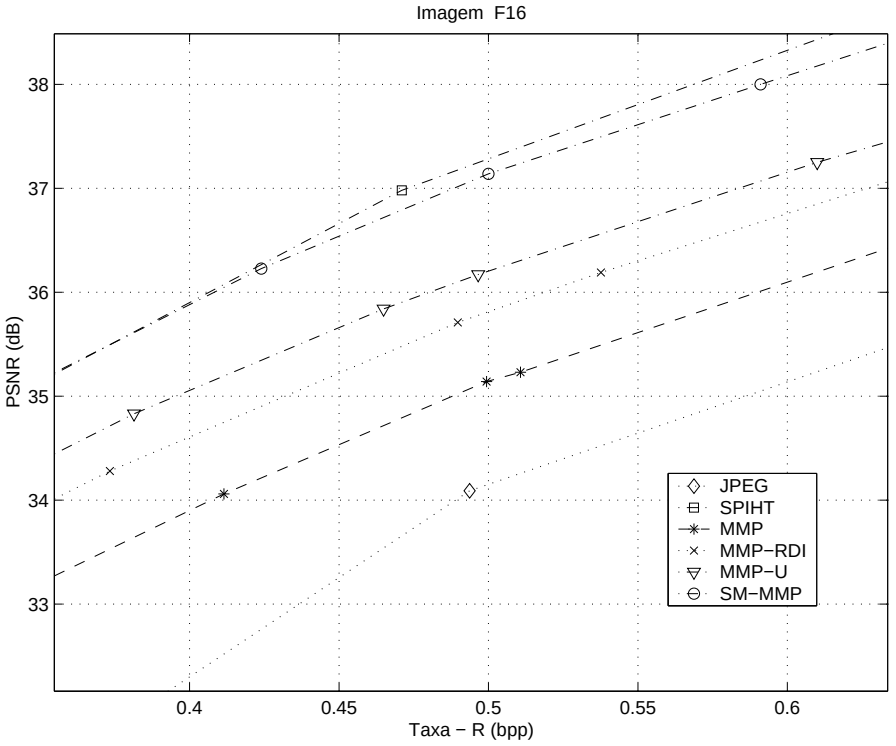


Figura 4.19: Zoom da Taxa x distorção para a imagem F16 512 x 512

Os melhores resultados destes algoritmos foram para as imagens mistas (imagens que possuem texto e/ou imagens), que são as imagens PP1205 e PP1209. Nestas imagens o MMP atinge 2,5 dB de PSNR acima do algoritmo SPIHT que é considerado como estado da arte para compressão de imagens.

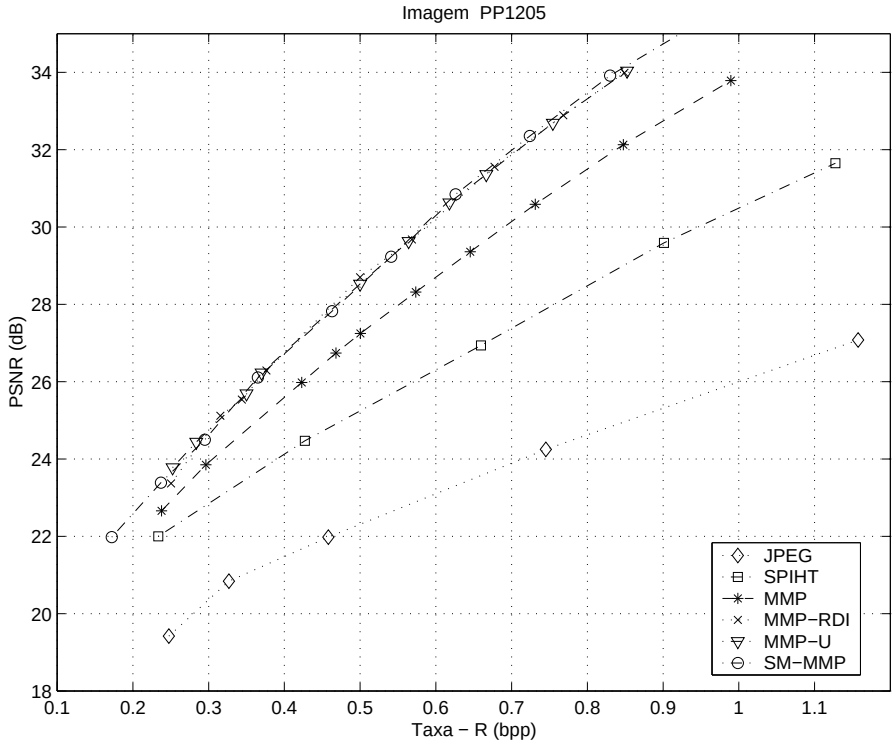


Figura 4.20: Taxa x distorção para a imagem PP1205 512 x 512

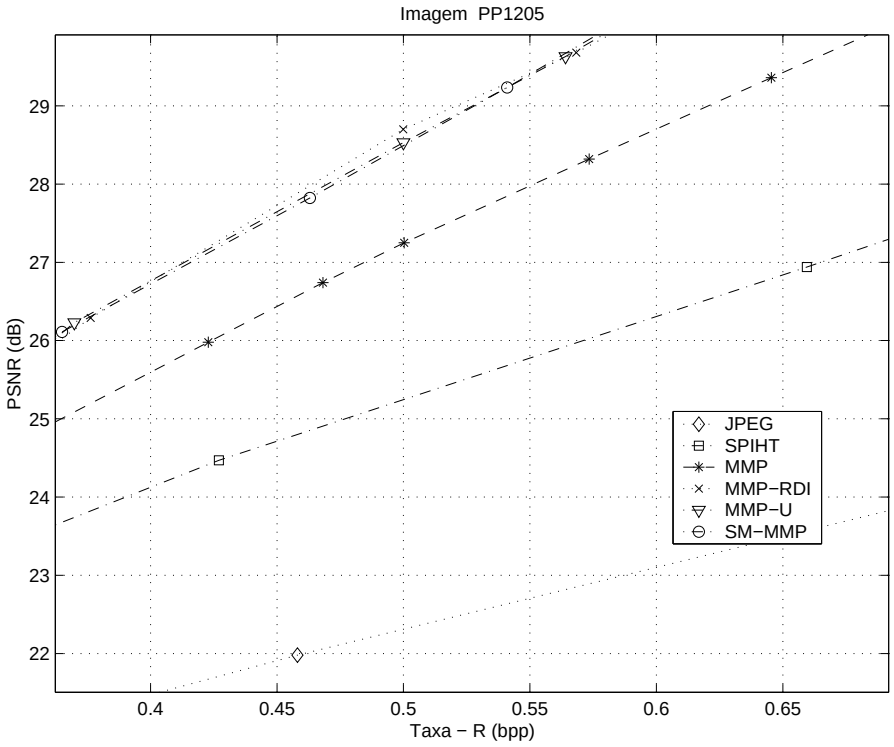


Figura 4.21: Zoom da Taxa x distorção para a imagem PP1205 512 x 512

Claramente pode ser observado nas Figuras 4.20, 4.22 e suas respectivas ampliações, que as versões da família do algoritmo MMP, apresentam os melhores resultados nas curvas taxa-distorção, para imagens pertencentes a classes de imagens mistas.

Com estes resultados apresentados é notório que o desempenho das versões do MMP original tiveram um crescimento bastante significativo em imagens suaves chegando a resultados bem próximos comparados com o algoritmo baseado em *Wavelets*. E nas imagens complexas, os resultados foram satisfatórios levando em conta que o MMP padrão já se destacava.

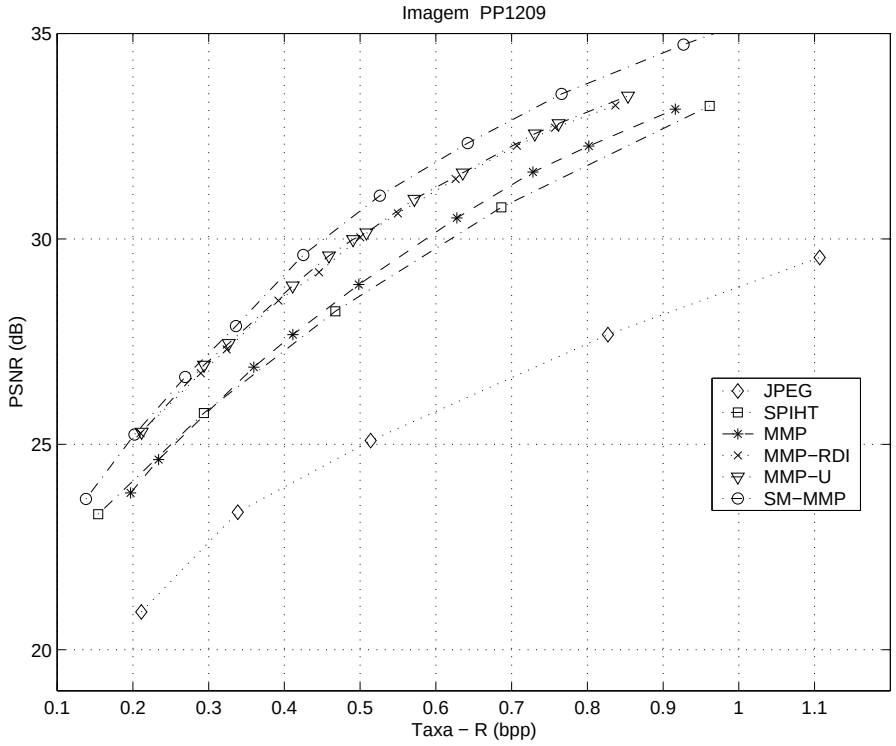


Figura 4.22: Taxa x distorção para a imagem PP1209 512 x 512

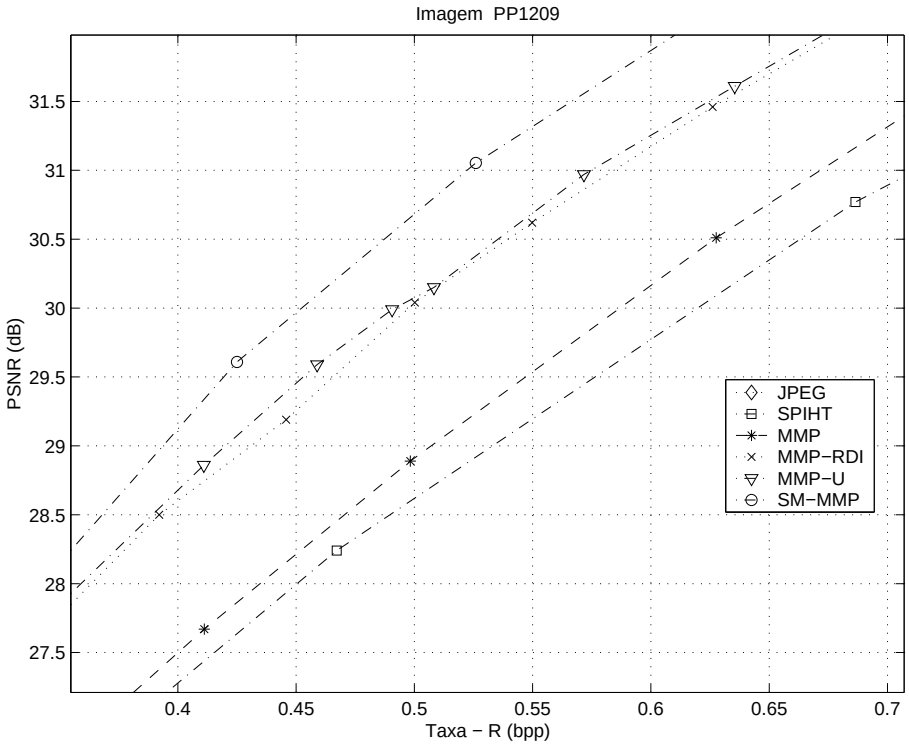


Figura 4.23: Zoom da Taxa x distorção para a imagem PP1209 512 x 512

Capítulo 5

O algoritmo GSM-MMP

O MMP padrão introduzido por [26], apresentou um desempenho eficaz na compressão de um conjunto de imagens, superando em imagens com texto, gráficos ou mistas os algoritmos que constituem o estado da arte na área de compressão de imagens. No caso do algoritmo JPEG [6] baseado em DCT, os resultados do MMP superaram para imagens suaves, entretanto com os codificadores baseados em DWT como o JPEG2000 [38] e SPIHT [37] tal fato não foi verdadeiro. Os algoritmos baseados em transformadas possuem a suposição de que as imagens a serem processadas são de natureza passa-baixa, enquanto que o mesmo não ocorre com o MMP pois ele processa o sinal de entrada sem fazer nenhuma suposição.

Com base no que foi exposto, para melhorar os resultados do MMP para imagens suaves, deve-se assumir um modelo estatístico para a fonte ajustando o algoritmo para um melhor desempenho. Preservando o desempenho em imagens de texto, gráficos ou mistas no qual obteve excelentes resultados.

O algoritmo SM-MMP visto na Sessão 4.4 teve como objetivo melhorar a codificação de imagens suaves através das informações dos blocos adjacentes já codificados, o que levou a resultados satisfatórios no processo de codificação de tais imagens conforme [28, 29]. Com isto propõem-se um estudo para melhoria da técnica SM-MMP, que será descrito a seguir.

5.1 Descrição do algoritmo GSM-MMP

Neste trabalho está sendo proposto um método que se adeque à estrutura do MMP [39, 40, 41, 42] baseado no casamento lateral generalizado chamado GSM-MMP (*Generalized Side-Match Multidimensional Multiscale Parser*) capaz de melhorar o desempenho do MMP padrão tanto nas imagens suaves como nas imagens em que o MMP se destaca (gráficos, textos ou mistas).

Este esquema de codificação com perdas GSM-MMP, seleciona seu dicionário de codificação baseado em três vizinhos *(Three-Sided)*, este dicionário é chamado de dicionário de estado D_{est}, o que aumenta a eficiência da predição, pois o mesmo utiliza as características da vizinhança para prever a codificação do bloco de entrada.

O GSM-MMP codifica os blocos de entrada no sentido da leitura, de cima para baixo da esquerda para de direita, logo existirão blocos vizinhos que ainda não foram processados durante a codificação do bloco de entrada B^j, prejudicando assim as informações laterais para o casamento aproximado com estes blocos vizinhos. Para resolver este problema se faz necessário codificar estes blocos primeiro, com isto este esquema de codificação possui duas fases.

Na primeira fase existe um conjunto de blocos que devem ser codificados e transmitidos para o decodificador, esses blocos são chamados de pré-codificados. O MMP original com taxa-distorção é responsável por codificar os blocos pré-codificados. Na segunda fase os blocos restantes são codificados pelo *Three-Sided* utilizando as informações da vizinhança que no caso são três blocos vizinhos, para um melhor entendimento destas fases será analisada a Figura 5.1.

Na Figura 5.1 os blocos rachurados são os blocos pré-codificados que são codificados pelo MMP original com taxa-distorção; os blocos em branco são os blocos que serão codificados pelo *Three-Sided*, depois que os blocos pré-codificados são codificados. Observando a Figura 5.1 nota-se casos de predição e alguns deles são: nos casos em que os blocos que possuem os números 1, 2 e 3 tomados como exemplos na figura, as informações são obtidas através de três blocos vizinhos, (esquerdo,direito,abaixo), (esquerdo,direito,acima) e (esquerdo,acima,abaixo) respectivamente. No bloco que possui o número 4 (como exemplo), as informações

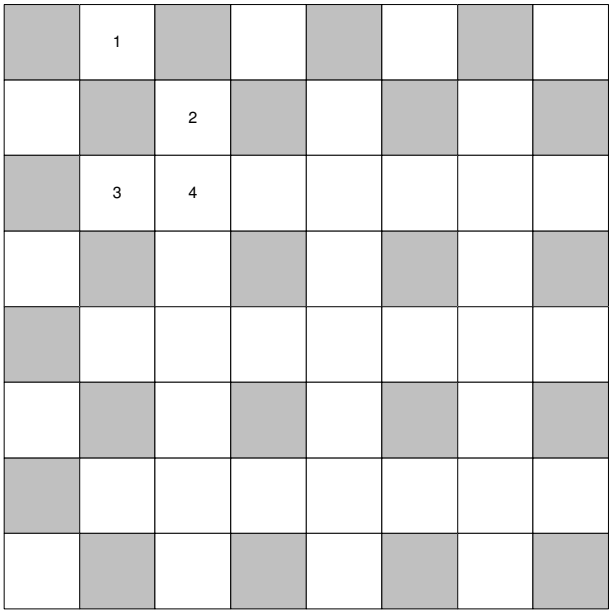


Figura 5.1: Blocos pré-codificados e codificados

usadas pelo *Three-Sided* são os blocos esquerdo e acima, pois o algoritmo GSM-MMP também processa seus blocos no sentido da leitura como no MMP padrão, com isto estes vizinhos já estarão processados e prontos para serem utilizados.

Antes da codificação é necessário ter conhecimento de um parâmetro importante para dimensionar o tamanho máximo do dicionário de estado $tam_{max}(D_{est})$, para que o mesmo se adeque a cada tipo de imagem que será processada, bem como o tamanho do dicionário de estado $tam(D_{est})$ para cada bloco de entrada que será codificado pelo *Three-Sided*. Levando-se em consideração que os dicionários de estado são temporários, uma vez que para cada bloco de entrada que será codificado existirá um dicionário de estado baseado nas informações dos blocos vizinhos.

O parâmetro a ser analisado é conhecido como atividade (AC -*Activity*). Esta medida permite identificar de forma eficaz qualquer variação (ou atividade) presente nos blocos vizinhos. Com isto o tamanho do dicionário de estado de cada bloco de entrada B^j vai depender do nível de atividade dos blocos vizinhos. Para calcular a atividade (AC) de um bloco B com dimensões $(L \times C)$, deve-se calcular as atividades de todas as linhas e colunas deste bloco e depois é escolhido o maior valor, como é visto por:

$$
\begin{aligned}
AC_{horiz}(B) &= \max_{1\leq i\leq L}\sum_{j=1}^{C-1}|b_{i,j}-b_{i,j+1}|,\\
AC_{verti}(B) &= \max_{1\leq j\leq C}\sum_{i=1}^{L-1}|b_{i,j}-b_{i+1,j}|,
\end{aligned}
$$

$$
AC_{bloco}(B) = max\{AC_{horiz}(B), AC_{verti}(B)\}. \tag{5.1}
$$

Para se obter o tamanho máximo do dicionário de estado $tam_{max}(D_{est})$, é necessário ter o cálculo da maior atividade AC presente na imagem original, considerando o tamanho do bloco de codificação que possui a maior hierarquia. No caso desta implementação o tamanho do bloco de entrada para codificação é (16×16). Com isto é calculado todas as atividades dos blocos (16×16) da imagem original e é armazenado o maior valor encontrado ($AC_{imag_{max}}$), calcula-se também a média aritmética de todas as atividades (AC_{media}).

O cálculo do tamanho máximo do dicionário de estado $tam_{max}(D_{est})$ segue em dois passos por [29] são eles;

$Primeiro : AC_{final} = \left\lfloor \frac{AC_{imag_{max}}}{4} \right\rfloor + AC_{media}$

$Segundo : SeAC_{final} \geq 285, tam_{max}(D_{est}) = \left\lfloor \frac{(AC_{final}-285).6000}{39} \right\rfloor + 5000$

Senão, $tam_{max}(D_{est}) = \left\lfloor \frac{(AC_{final}).4999}{285} \right\rfloor + 1$

O tamanho máximo do dicionário de estado não necessita ser preciso, pois pode-se obter ótimos resultados aproximados do valor ideal dentro de uma larga faixa. Como exemplo, podemos citar a imagem Lena que apresenta a mesma qualidade comprimida utilizando tamanhos máximos de dicionário de estado entre 4500 e 5500 elementos.

Para cada bloco de entrada B^j a ser codificado temos um dicionário de estado temporário. O seu tamanho final é definido de acordo com a razão entre as médias das atividades dos blocos vizinhos (AC_{media}) em questão e o maior valor da atividade encontrada na imagem original de maior hierarquia ($AC_{imag_{max}}$). É importante notar que esses blocos vizinhos podem ser blocos de entrada completos (16×16) que já foram codificados, partes dos blocos de entrada já codificados (ex:bloco 4×4) ou ainda partes já codificadas do bloco que está sendo processado,

isto depende da posição atual do bloco de entrada B^j na árvore de segmentação. Com esta dependência da localização na árvore de segmentação ocorrerá casos em que não haverá três blocos vizinhos suficientes para fazer o casamento lateral, como visto na Figura 5.2.

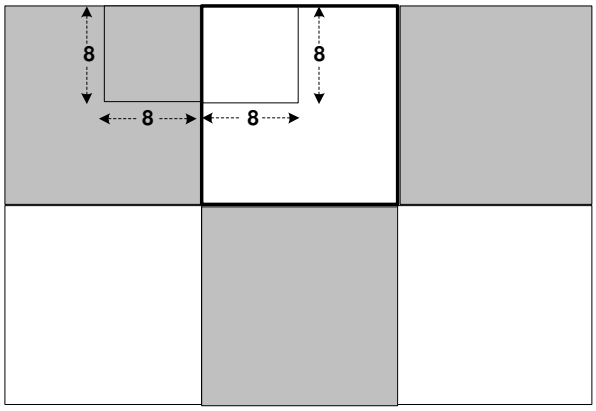


Figura 5.2: Casamento lateral com apenas um bloco vizinho

Na Figura 5.2 observa-se que os blocos rachurados são os blocos processados pelo MMP original com taxa-distorção e os blocos em branco são os blocos que irão ser processados pelo *Three-Sided*. O bloco branco de dimensão (16×16) que está em destaque possui três blocos vizinhos, esquerdo,direito e abaixo, sabe-se que são vizinhos pois são os únicos blocos nas adjacências rachurados. Este bloco sofre duas segmentações até encontrar a aproximação adequada para codificar esta parte do bloco, como é mostrado, o bloco de entrada em branco com dimensões (8×8) resultante das segmentações no bloco de entrada (16×16). Como o *Three-Sided* codifica seus blocos no sentido da leitura, começando do bloco superior esquerdo, o bloco resultante possui apenas um vizinho disponível para fazer o casamento lateral, que é o bloco à esquerda com a mesma dimensão (8×8) do bloco que será codificado. Sendo assim existirá apenas um dos valores para o cálculo do tamanho do dicionário de estado $tam(D_{est})$.

Para escolher os elementos que serão inseridos no dicionário de estado é adotado um critério que se baseia na seguinte premissa: em imagens suaves os valores dos *pixels* vizinhos são bem parecidos ou até iguais, ou seja, os elementos que farão parte do dicionário de estado serão aqueles que possuem em suas bordas os *pixels*

mais parecidos ou iguais com os blocos vizinhos que já foram codificados. Este critério é chamado de critério de continuidade onde os elementos escolhidos para o dicionário de estado são aqueles que possuem os menores valores de rugosidade, obedecendo ao critério de continuidade. O cálculo do critério de continuidade é visto a seguir através de um exemplo da Figura 5.3.

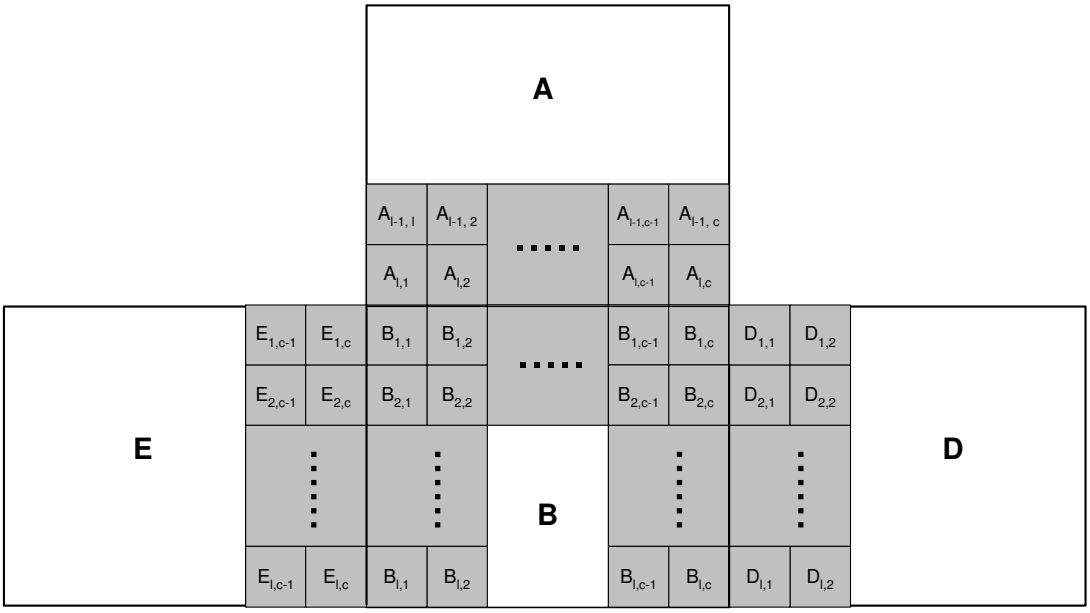


Figura 5.3: *Pixels* que são utilizados para o cálculo da rugosidade

A Figura 5.3 apresenta o bloco B com seus três blocos vizinhos, vizinho esquerdo E, vizinho direito D, vizinho acima A. Para este exemplo com três vizinhos o cálculo de rugosidade é dado por:

$$
\begin{aligned}
Rug\_vert_{acima}(B) &= \sum_{i=1}^{c} \left| \left\lfloor \frac{(A_{l-1,i} - A_{l,i}) + (B_{1,i} - B_{2,i})}{2} \right\rfloor - (A_{l,i} - B_{1,i}) \right|, \\
Rug\_hori_{esquerda}(B) &= \sum_{j=1}^{l} \left| \left\lfloor \frac{(E_{j,c-1} - E_{j,c}) + (B_{j,1} - B_{j,2})}{2} \right\rfloor - (E_{j,c} - B_{j,1}) \right|, \\
Rug\_hori_{direita}(B) &= \sum_{j=1}^{l} \left| \left\lfloor \frac{(B_{j,c-1} - B_{j,c}) + (D_{j,1} - D_{j,2})}{2} \right\rfloor - (B_{j,c} - D_{j,1}) \right|,
\end{aligned}
$$

$$
Rug(B) = Rug\_vert_{acima}(B) + Rug\_hori_{esquerda}(B) + Rug\_hori_{direita}(B) \quad (5.2)
$$

Observando a Equação 5.2, verifica-se que são utilizados dois *pixels* para fazer uma subtração, na verdade o que se está calculando é a derivada de ordem 0 (um *pixel* de cada borda) e ordem 1 (dois de cada borda). A derivada de ordem 0 procura por blocos que apresentem valores de *pixels* de borda iguais, já a derivada de ordem 1 procura por blocos que acompanhem uma reta através dos blocos. Sendo assim, se o valor dos *pixels* da borda de um bloco estiver aumentando, o melhor bloco para o casamento será aquele que também apresentar valores de *pixels* aumentando. Se os valores estiverem estacionários (ex: 150), o melhor bloco para o casamento será aquele com valores de borda em torno de 150. Este critério de continuidade que é expresso pelos menores valores de rugosidade dos blocos vizinhos, é que irá definir o rendimento do algoritmo. Pois se este critério for mal projetado, os elementos que forem escolhidos para fazer parte do dicionário de estado D_{est} serão diferentes dos elementos presente na imagem original, ou seja, esses elementos irão apresentar uma enorme distorção comparando com o bloco original mesmo para *pixels* que se encontrem bem perto das bordas, resultando assim uma propagação de erro para a realização do casamento generalizado com os demais blocos, produzindo na codificação imagens reconstruídas com qualidade baixa . É por esse motivo que utilizamos dois pixels de borda em vez de um, tornando mais eficaz a escolha dos blocos para a codificação, considerando as transições existentes entre os blocos sendo elas crescentes ou decrescentes. Contudo, o algoritmo GSM-MMP realiza a segmentação dos blocos, como em blocos de (1×2), neste caso o cálculo da distorção vertical fica comprometido pois será utilizado apenas um *pixel*, que ainda pode ser muito diferente daquele presente na borda do seu vizinho.

Os elementos que serão inseridos no dicionário de estado D_{est} além de possuírem as menores rugosidades (critério de continuidade) deverão ser ordenados, o primeiro elemento do dicionário de estado D_{est} deverá possuir a menor rugosidade e o último elemento a maior rugosidade. Se acontecer o caso em que dois elementos possuírem a mesma rugosidade, o elemento mais recente fica em uma posição mais à frente do D_{est}.

Como foi dito o algoritmo GSM-MMP, seleciona o dicionário de estado D_{est} do bloco de entrada B que será codificado, através das predições dos elementos

vizinhos previamente codificados, onde esses elementos S^i seriam do dicionário oficial D e elementos S_R^i do dicionário rascunho D_R, no caso da otimização, mais indicados para a sua codificação. Com isto tem-se um dicionário de estado D_{est} menor que o dicionário oficial D, ou menor que a união dos dicionários oficial D e rascunho D_R se for dentro da função de otimização; possuindo assim um número de bits bem menor para codificar um dado índice i por ter um dicionário D_{est} com números bem menores de elementos, pois o algoritmo varrerá todo o dicionário oficial D ou $D \cup D_R$ para obter os elementos com as menores rugosidades.

Neste trabalho para obter o dicionário de estado D_{est}, foram atribuídos sinalizadores para cada elemento do dicionário D e D_R, com o objetivo de informar se um dado elemento faz parte do dicionário de estado D_{est} ou não. Este é um recurso mais fácil e eficaz para adequar a implementação do *Three-Sided* na estrutura do MMP. Não é criado o dicionário de estado D_{est} para os elementos que possuem as dimensões (1×1), pois o dicionário já é pequeno e a sua diminuição não se traduz em ganho de PSNR. Analisando em nível de pixel (1×1) a variação no nível de cinza pode ser grande, principalmente se o mesmo estiver localizado em uma borda.

A função que gera os dicionários de estado D_{est} será executada à cada chamada da função de otimização ou codificação, para que cada bloco seja analisado ou codificado utilizando o seu próprio dicionário de estado D_{est}, bem como a execução do algoritmo *Three-Sided*, para que a árvore de segmentação se ajuste aos dicionários de estado eleito para cada bloco de entrada B^j que está sendo analisado. Com isto, deve-se utilizar os dicionários de estado D_{est} no momento de executar os cálculos dos custos de cada nó da árvore de segmentação, para decidir a poda da mesma.

Na função de otimização obtém-se um custo computacional muito alto na execução do algoritmo *Three-Sided*, porque serão analisados todos os nós da árvore de segmentação. Vale ressaltar que na otimização há casos em que a construção dos dicionários de estado D_{est} de alguns blocos, é feito através do casamento com blocos $\hat{B}$ previamente analisados pela otimização, mas estes blocos previamente analisados não são os definitivos, são chamados de blocos provisórios ou pretendentes , devido à uma possibilidade de poda que a árvore de segmentação pode sofrer mais à frente.

Todos estes valores previamente analisados requerem uma área de memória para executar todas essas avaliações.

Como na função de otimização teremos a geração do dicionário de estado D_{est}, há a necessidade de implementar um novo modelo temporário de freqüências, contemplando apenas os elementos que fazem parte do dicionário de estado D_{est}, ou seja, estarão somente no somatório das freqüências acumuladas os elementos escolhidos, sem levar em consideração os dicionários oficial D e rascunho D_R em suas totalidades. Assim os cálculos de custo serão executados apenas nestes elementos, utilizando este novo modelo de freqüências.

Na codificação a árvore de segmentação já está definida, logo a execução do algoritmo *Three-Sided* não requer muito custo computacional na construção dos dicionários de estado, só apresentará alguns gastos mínimos se a geração dos dicionários de estado percorrerem até os nós folhas. Na função de codificação o casamento lateral do bloco de entrada com os blocos previamente analisados já são os definitivos, pois todas as podas necessárias na árvore de segmentação já foram analisadas e realizadas.

Com a geração do dicionário de estado D_{est} na codificação, torna-se necessário também a construção de um novo modelo de freqüências para os elementos que fazem parte do D_{est} como na função de otimização. Tratando-se da função de codificação, os elementos escolhidos para fazerem parte do dicionário de estado D_{est} são elementos selecionados com as menores rugosidades do dicionário oficial D, com um novo modelo de freqüências adaptado ao mesmo. Com o D_{est} completo pode-se efetuar todos os cálculos de custo utilizando o novo modelo de freqüências. É notório que ocorre uma diminuição no novo dicionário que chamamos de dicionário de estado D_{est}, pois o mesmo se adapta ao bloco de entrada que será codificado, com este número de elementos reduzidos no D_{est}, reduz-se também o número de bits necessários para codificar um dado índice i, apresentando assim imagens com boa qualidade, em determinadas taxas. É importante citar que para blocos que possuem vizinhos com uma elevada atividade final AC_{final}, irão ter dicionários de estado bem maiores em relação aqueles que possuem uma atividade final AC_{final} menor, pois os dicionários destes são bem menores.

A árvore de segmentação é formada por *flags* com valores iguais a 0 e 1 e os índices dos blocos que foram escolhidos para serem codificados e essa codificação é feita através do codificador aritmético [18]. A vantagem de utilizar o codificador aritmético consiste em que o modelo de probabilidade é independente (separado) da máquina de codificação. Assim, o modelo pode ter qualquer formato que explore de maneira mais eficiente os dados, no caso desta implementação apenas acumula estatísticas pois não possui contextos, mantendo assim a mesma máquina de codificação.

Para o algoritmo GSM-MMP o dicionário inicial é composto por 128 vetores com os mesmos espaços entre 0 e 254. A equação da formação do dicionário é mostrada em :

$pixelmin = 0$ (valor de pixel mínimo),

$pixelmax = 255$ (valor de pixel máximo),

$El = 128$

$Of = \frac{(pixelmax - pixelmin)}{El-1} = 2$,

$pixelmax = Of.(El - 1) = 254$,

Para $n = (0, 1, \ldots, Num_D - 1)$

$L = 2^{\lfloor 0.5(n+1) \rfloor}$,

$C = 2^{\lfloor 0.5n \rfloor}$,

$$D_i^{L,C} = \{T_{1,1}^{L,C}[0], T_{1,1}^{L,C}[\lfloor 1.Of \rfloor], ..., T_{1,1}^{L,C}[\lfloor (El-2).Of \rfloor], T_{1,1}^{L,C}[254]\} \tag{5.3}$$

No decodificador também é importante a ordem das funções que são chamadas como no algoritmo de codificação. O decoficador lê a árvore de segmentação transmitida pelo codificador e os índices dos blocos do dicionário de estado, reconstruindo assim a imagem.

O algoritmo GSM-MMP será descrito nas próximas páginas, primeiramente os blocos que são processados pelo MMP original com taxa-distorção, conhecidos como blocos pré-codificados e a seguir os blocos que são processados pelo *Three-Sided*.

A seguir será mostrado o algoritmo MMP com taxa-distorção que possui algumas considerações que devem ser observadas, são elas:

1. O bloco de entrada B^j, é o bloco da imagem original, onde possui dimensões $L \times C = (Linha \times Coluna)$, o índice j equivale ao nó n_j da árvore de segmentação $A(no)$ que associa ao bloco codificado $\hat{B}^j$;

2. O dicionário oficial D é simulado por um novo dicionário chamado rascunho D_R, que simula todas as atualizações que ocorreriam na codificação;

3. Na função de otimização a árvore de segmentação começa com uma árvore completa $\alpha(n_0)$ com todos os seus nós folhas, onde todas as posições possuem valor igual a 1;

4. Para calcular os custos dos elementos dos dicionários oficial S^i e rascunho S^i_R, deve-se utilizar a somatória dos contadores C_D (oficial) ,$\hat{C}_D$ (complementar) e C_{D_R} (rascunho), e nos custos dos *flags*, deve-se utilizar a somatória dos contadores C_{D_F} (ofical) e $\hat{C}_{D_F}$ (complementar), isto tratando-se da função de otimização;

5. O resultado da função de otimização é o bloco aproximado $\hat{B}^j$ e a árvore de segmentação $A(no)$ que será utilizada pela função de codificação para decidir se um bloco B^j será codificado ou particionado;

6. Os nós n_x serão representados da seguinte maneira na árvore de segmentação $A(no)$; 1 existe o nó n_x, 0 o nó n_x é podado.

Procedimento $\left[\hat{B}^j,\, A(n_0)\right]$ = `OtimizacaoMMP`(N, M, B^j, no, $\alpha(n_0)$)

Passo 1: Faz $A(n_0) = \alpha(n_0)$.

Passo 2: Localizar no dicionário oficial D o índice i do bloco S^i com mesma dimensão (L, C) do bloco de entrada B^j e que melhor se aproxime de acordo com o menor custo $J_{n_j} = D_{n_j} + \lambda R I_{n_j}$. Guarde i e faça $\hat{B}^j = S^i$. O custo é calculado utilizando os seguintes contadores: $\sum C_D, \sum \hat{C}_D, \sum C_{D_R}, C_i$ e $\hat{C}_i$.

Passo 3: Verificar se algum elemento S_R^i pertencente ao dicionário rascunho D_R, possui uma melhor aproximação e um custo menor do que foi escolhido no **Passo 2**. Se tal fato ocorrer, guarde o índice i e faça $\hat{B}^j = S_R^i$. O custo é calculado utilizando os seguintes contadores: $\sum C_D, \sum \hat{C}_D, \sum C_{D_R}$ e C_{i_R}.

Passo 4: **Se** (L = C = 1) então

> **Se** (A aproximação foi feita pelo dicionário oficial D) então Incrementa o contador que simula o comportamento da contagem para os índices dentro da otimização $\hat{C}_i$.
>
> **Se** (A aproximação foi feita pelo dicionário rascunho D_R) então Incrementa o contador para os índices do dicionário rascunho C_{i_R}.
>
> **Retorne** $\hat{B}^j$ e $A(n_0)$.

Se não vai para o **Passo 5**.

Passo 5: Acrescenta, ao custo J_{n_j}, o valor da taxa para o *flag* de segmentação igual a 1, λR_1. Esse valor representa completamente o custo para um nó folha e deve levar em consideração os contadores dos flags:$\sum C_{D_F}, \sum \hat{C}_{D_F}, C_{F_1}, \hat{C}_{F_1}$.

Passo 6: Calcule e armazene, o valor da taxa para o *flag* de segmentação igual a 0, λR_0, que completará o custo dos nós-filhos, a taxa do *flag* 0 é calculada com os seguintes contadores:$\sum C_{D_F}, \sum \hat{C}_{D_F}, C_{F_0}, \hat{C}_{F_0}$.

Passo 7: Incremente o contador do *flag* de segmentação igual a 0, $\hat{C}_{F_0}$.

Passo 8: **Se**$(C > L)$ **entao** divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{C}{2}$ e chame a `OtimizacaoMMP` na ordem:

$\left[B^{\hat{2j+1}},\, A(n_0)_a\right] =$ `OtimizacaoMMP`$(\ L,\, \frac{C}{2},\, B^{2j+1},\, 2\,no+1,\, A(n_0)\ )$

$\left[B^{\hat{2j+2}},\, A(n_0)_b\right] =$ `OtimizacaoMMP`$(\ L,\, \frac{C}{2},\, B^{2j+2},\, 2\,no+2,\, A(n_0)\ )$

Se Não divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{L}{2}$ e chame o procedimento de `OtimizacaoMMP` na ordem:

$\left[B^{\hat{2j+1}},\, A(n_0)_a\right] =$ `OtimizacaoMMP`$(\ \frac{L}{2},\, C,\, B^{2j+1},\, 2\,no+1,\, A(n_0)\ )$

$\left[B^{\hat{2j+2}},\, A(n_0)_b\right] =$ `OtimizacaoMMP`$(\ \frac{L}{2},\, C,\, B^{2j+2},\, 2\,no+2,\, A(n_0)\ )$

Passo 9: Faz $A(n_0) = A(n_0)_a$ **e** $A(n_0)_b$.

Passo 10: Se o custo do nó pai for menor ou igual ao custo dos nós filhos então não devemos continuar a segmentação. Matematicamente temos:

$$\{J_{no}\} \leq \left\{J_{(2\,no+1)} + J_{(2\,no+2)} + \lambda R_0(no)\right\}$$

Se a sentença for verdadeira siga para o **Passo 11**.

Se não vá para o **Passo 17**.

Passo 11: Decrementa os contadores do flag 0 em todas as dimensões relacionadas as árvores $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$ a serem podadas.

Passo 12: Decrementa os contadores $\hat{C}_{F_1}, \hat{C}_D$ ou $\hat{C}_{D_R}$ para todas as dimensões relacionadas aos nós-folhas das árvores $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$ e nas devidas posições dos seus índices, devendo verificar a origem de cada elemento. Se pertence ao dicionário oficial D ou se pertence ao dicionário rascunho D_R.

Passo 13: Decrementa o contador $\hat{C}_{F_0}$.

Passo 14: Incrementa os contadores $\hat{C}_{F_1}, \hat{C}_i$ ou C_{i_R}, dependendo do elemento utilizado S como aproximação. Se pertence ao dicionário oficial D ou se pertence ao dicionário rascunho D_R.

Passo 15: Elimina as atualizações do dicionário rascunho D_R que foram inseridas devido às árvore $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$.

Passo 16: Seta em $A(n_0)$ que as árvores $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$ foram eliminadas.
Retorne $\left[\hat{B^j},\, A(n_0)\right]$.

Passo 17: Realize a concatenação dos blocos $B^{\hat{2j+1}}$ e $B^{\hat{2j+2}}$ da seguinte forma:
Se$(C > L)$ **entao** concatene na vertical, ou seja:

$$\hat{B^j} = \begin{pmatrix} B^{\hat{2j+1}} \\ B^{\hat{2j+2}} \end{pmatrix}$$

Se Não concatene na horizontal, ou seja:

$$\hat{B^j} = \begin{pmatrix} B^{\hat{2j+1}} & B^{\hat{2j+2}} \end{pmatrix}$$

Passo 18: Atualize o dicionário rascunho D_R em todas as dimensões da seguinte forma:
Para $n = (0, 1, \ldots, w-1)$ **faça**

$$L_x = 2^{\lfloor 0.5(n+1) \rfloor}$$
$$C_x = 2^{\lfloor 0.5n \rfloor}$$
$$S_{L_x, C_x} = \{S_{L_x, C_x}\} \cup \left\{T^{L,C}_{L_x, C_x}\left[\hat{B^j}\right]\right\}$$

Passo 19: Atualize o custo do nó.

$$\{J_{no}\} = \left\{J_{(2\,no+1)} + J_{(2\,no+2)} + \lambda R_0(no)\right\}$$

Passo 20: **Retorne** $\left[\hat{B^j},\, A(n_0)\right]$.

A seguir será mostrado na página seguinte o algoritmo de codificação do MMP.

Procedimento $\hat{B^j}$ = `CodificaMMP`(N, M, B^j, no, $A(n_0)$)

Passo 1: **Se** ($A(n_{2\,no+1}) = 0$ e $A(n_{2\,no+2}) = 0$ **ou** L=C=1) **então** segue para o **Passo 2** seguinte.

Se não vai para o **Passo 5**.

Passo 2: Localizar no dicionário oficial D o índice i do bloco S^i com mesma dimensão (L, C) do bloco de entrada B^j e a melhor aproximação deste bloco (B^j), segundo o menor custo *Lagrangeano* $J_{n_j} = D_{n_j} + \lambda RI_{n_j}$. Guarde o valor do índice i do elemento S^i escolhido e faça $\hat{B^j} = S_i$.

Passo 3: **Se**($L = C = 1$) **então** guarde o índice i do elemento escolhido S^i e retorne o bloco aproximado $\hat{B^j}$.

Se Não Vai para o passo 4.

Passo 4: Codifica o *flag* de valor igual a 1, o índice i do elemento S^i escolhido pela aproximação e retorna o bloco $\hat{B^j}$.

Passo 5: Codifica o *flag* de valor igual a 0.

Se($C > L$) **entao** divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{C}{2}$ e chame o procedimento de codificação na ordem:

$\hat{B^{2j+1}}$ = `CodificaMMP`(L, $\frac{C}{2}$, B^{2j+1}, $2\,no+1$, $A(n_0)$)

$\hat{B^{2j+2}}$ = `CodificaMMP`(L, $\frac{C}{2}$, B^{2j+2}, $2\,no+2$, $A(n_0)$)

Se Não divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{L}{2}$ e chame o procedimento de codificação na ordem:

$\hat{B^{2j+1}}$ = `CodificaMMP`($\frac{L}{2}$, C, B^{2j+1}, $2\,no+1$, $A(n_0)$)

$\hat{B^{2j+2}}$ = `CodificaMMP`($\frac{L}{2}$, C, B^{2j+2}, $2\,no+2$, $A(n_0)$)

Passo 6: Faça a concatenação dos blocos $\hat{B^{2j+1}}$ e $\hat{B^{2j+2}}$ da seguinte forma:

Se($C > L$) **entao** concatene na vertical,

$$\hat{B^j} = \begin{pmatrix} \hat{B^{2j+1}} \\ \hat{B^{2j+2}} \end{pmatrix}$$

Se Não concatene na horizontal,

$$\hat{B^j} = \begin{pmatrix} \hat{B^{2j+1}} & \hat{B^{2j+2}} \end{pmatrix}$$

Passo 7: Atualize o dicionário oficial D em todas as dimensões da seguinte maneira:

Para $n = (0, 1, \ldots, w-1)$ **faça**

$$L_x = 2^{\lfloor 0.5(n+1) \rfloor}$$
$$C_x = 2^{\lfloor 0.5n \rfloor}$$
$$S_{L_x, C_x} = \{S_{L_x, C_x}\} \cup \left\{ T^{L,C}_{L_x, C_x} \left[\hat{B}^j \right] \right\}$$

Passo 8: Retorne $\hat{B}^j$.

A seguir será mostrado o pseudo-código do algoritmo *Three-Sided* que possui algumas considerações que devem ser observadas, são elas:

1. Será descrito a função que gera os dicionários de estado D_{est}, a função de otimização *Three-Sided* e a codificação respectivamente;

2. Os resultados da função que gera o dicionário de estado D_{est} são as freqüências do dicionário oficial $Freq_D$, se a chamada for na codificação ou as freqüências do dicionário oficial $Freq_D$ união com as freqüências do dicionário rascunho $Freq_{D_R}$, se a chamada for na otimização. Para calcular os custos dos elementos dos dicionários oficial S^i e rascunho S^i_R;

3. Foram utilizados sinalizadores para cada elemento do dicionário oficial S^i e elementos do dicionário rascunho S^i_R, para informar se este elemento faz parte do dicionário de estado D_{est} ou não. Se o elemento faz parte do D_{est} e veio do dicionário oficial D, logo o sinalizador receberá valor igual a 1, $Freq_D = 1$. Se o elemento faz parte do D_{est} e veio do dicionário rascunho D_R, logo o sinalizador receberá valor igual a 0, $Freq_{D_R} = 0$.

4. O dicionário de estado D_{est} não é gerado para os elementos de dimensões (1×1);

5. Na função de otimização tem-se um dicionário menor, pois somente estarão no D_{est} os elementos com as menores rugosidades dos dicionário oficial D e dicionário rascunho D_R;

6. Na otimização, para calcular os custos serão utilizados os contadores, pois existe a necessidade de um controle nas contagens das freqüências dos índices i e *flags* de segmentação (0 e 1), como na função de otimização do MMP padrão.

7. Na função de codificação, também tem-se um dicionário menor que o dicionário oficial D, pois no D_{est} estarão somente os elementos de menores rugosidades do dicionário oficial D;

8. Tanto no algoritmo de otimização e codificação são passados os valores das coordenadas dos vértices (V_l, V_c) superiores esquerdos dos vizinhos adjacentes, para os devidos cálculos de continuidade e dicionário de estado D_{est}.

Procedimento $[Freq_D,\ Freq_{D_R}] = Gera\_D_{est}(\ N,\ M,\ V_l, V_c\ )$

Passo 1: **Se**$(L = C = 1)$, faça $Freq_D = 1$ e $Freq_{D_R} = 0$, e retorna $Freq_D$. O esquema de freqüências utilizado para dimensões (1×1) é o mesmo do dicionário oficial D, ou $D \cup D_R$ se for na função de otimização.

Se Não, Segue para o passo 2.

Passo 2: **Se** existir bloco esquerdo vizinho B_{esq} (se $V_c \neq 0$);

Calcula atividade do bloco $AC_{bloco}(B_{esq})$.

Se existir bloco direito vizinho B_{dir} (se $(V_c + C) < (M - 1)$);

Calcula atividade do bloco $AC_{bloco}(B_{dir})$.

Se existir bloco inferior vizinho B_{inf} (se $(V_l + L) < (M - 1)$);

Calcula atividade do bloco $AC_{bloco}(B_{inf})$.

Se existir bloco superior vizinho B_{sup} (se $V_l \neq 0$);

Calcula atividade do bloco $AC_{bloco}(B_{sup})$.

Passo 3: Calcula a média das atividades do bloco $(AC_{media}) = \frac{(AC_{media})}{3}$.

Se Existir apenas dois blocos vizinhos: $(AC_{media}) = \frac{(AC_{media})}{2}$.

Se Existir apenas um bloco vizinho: $(AC_{media}) = \frac{(AC_{media})}{1}$.

Passo 4: Calcula o tamanho do dicionário de estado $tam(D_{est})$.

Se$(tam(D) > tam_{max}(D_{est}))$ **entao** $tam(D_{est}) = \frac{(AC_{media}.tam_{max}(D_{est}))}{AC_{imag_{max}}}$.

Se Não $tam(D_{est}) = \frac{(AC_{media}.tam(D_{est}))}{AC_{imag_{max}}}$.

Obs: Se a geração do dicionário de estado D_{est} for dentro da função de otimização, o tamanho do dicionário oficial será : $tam(D) = tam(D) + tam(D_R)$.

Passo 5: **Se** $tam(D_{est}) < tam_{min}(D_{est})$ **entao** $tam(D_{est}) = tam_{min}(D_{est})$.

Se Não Siga para o **Passo 6**.

Obs: Nesta dissertação o menor tamanho do dicionário de estado será: $tam_{min}(D_{est}) = 16$.

Passo 6: Insira os elementos no dicionário de estado D_{est}.

Para isto é necessário localizar no dicionário oficial D os elementos S^i que possuem as menores rugosidades, sinalizando com o valor 1 indicando que faz parte do dicionário de estado D_{est}, $Freq_D = 1$. Os elementos inseridos no D_{est} deverão ser ordenados de acordo com as suas respectivas rugosidades, de tal forma que, o primeiro elemento do D_{est} deverá possuir a menor rugosidade e o último elemento a maior rugosidade dentro do D_{est}.

Se Existir um elemento S^i que possui uma rugosidade menor que o último elemento no D_{est}, quando o mesmo encontrar-se preenchido, **então** retira este elemento da última posição no D_{est} e coloca o elemento com a menor rugosidade dentro do dicionário de estado D_{est}, reordenando-o logo em seguida.

Se Existir dois elementos com a mesma rugosidade, **então** o elemento que for mais recente fica em uma posição mais à frente do dicionário de estado D_{est}.

Passo 7: **Se** a chamada da função que gera o D_{est} for realizada na função de otimização, **então** repita o **Passo 6** para os elementos S^{i_R} do dicionário rascunho D_R, sinalizando com o valor igual a 0 indicando que faz parte do D_{est}, $Freq_{D_R} = 0$. Esta sinalização é de suma importância pois, se existir um elemento S^{i_R} de menor rugosidade e seja o elemento escolhido para representar o bloco de entrada B^j, fica fácil encontrá-lo.

Passo 8: Faça um novo modelo de freqüências com os elementos pertencentes ao dicionário de estado D_{est}.

Estes elementos serão ordenados conforme suas freqüências relativas e com isto, constituirão um novo modelo para o codificador aritmético. E é com este novo modelo de freqüências que será calculado o custo de cada elemento pertencentes ao dicionário de estado D_{est}, tanto na função de otimização como na função de codificação.

Passo 9: Retorna $Freq_D$ e $Freq_{D_R}$.

Como foi visto, cada bloco de entrada B^j terá seu próprio dicionário de estado D_{est}. Onde será localizado um elemento aproximado para o bloco de entrada, bem como a realização dos cálculos de custo de cada elemento agora pertencentes ao D_{est}. O D_{est} é criado dentro das funções de otimização e codificação resultando assim, algumas modificações nestas funções, que serão mostradas à seguir.

Procedimento $\left[\hat{B}^j,\, A(n_0)\right]$ = `OtimizaT-Sided`(N, M, B^j, no, $\alpha(n_0)$,V_l,V_c)

Passo 1: Faz $A(n_0) = \alpha(n_0)$.

Passo 2: Computa $[Freq_D,\, Freq_{D_R}] = Gera\_D_{est}(\ N,\, M,\, V_l,V_c\ )$.

Passo 3: Localizar no dicionário oficial D o índice i do bloco S^i com mesma dimensão (L, C) do bloco de entrada B^j, utilizando $Freq_D = 1$, e que melhor se aproxime de acordo com o menor custo $J_{n_j} = D_{n_j} + \lambda RI_{n_j}$. Guarde i e faça $\hat{B}^j = S^i$. O custo é calculado utilizando os seguintes contadores: $\sum C_D, \sum \hat{C}_D, \sum C_{D_R}, C_i$ e $\hat{C}_i$, para todo elemento pertencente à $Freq_D = 1$.

Passo 4: Verificar se algum elemento S_R^i pertencente ao dicionário rascunho D_R, com $Freq_{D_R} = 0$, possui uma melhor aproximação e um custo menor do que foi escolhido no **Passo 3**. Se tal fato ocorrer, guarde o índice i e faça $\hat{B}^j = S_R^i$. O custo é calculado utilizando os seguintes contadores: $\sum C_D, \sum \hat{C}_D, \sum C_{D_R}$ e C_{i_R}, para todo elemento pertencente à $Freq_{D_R} = 0$.

Passo 5: **Se** (L = C = 1) então

> **Se** (A aproximação foi feita pelo dicionário oficial D) então Incrementa o contador que simula o comportamento da contagem para os índices dentro da otimização $\hat{C}_i$.
>
> **Se** (A aproximação foi feita pelo dicionário rascunho D_R) então Incrementa o contador para os índices do dicionário rascunho C_{i_R}.
>
> **Retorne** $\hat{B}^j$ e $A(n_0)$.

Se não vai para o **Passo 6**.

Passo 6: Acrescenta, ao custo J_{n_j}, o valor da taxa para o *flag* de segmentação igual a 1, λR_1. Esse valor representa completamente o custo para um nó folha e deve levar em consideração os contadores dos flags:$\sum C_{D_F}, \sum \hat{C}_{D_F}, C_{F_1}, \hat{C}_{F_1}$.

Passo 7: Calcule e armazene, o valor da taxa para o *flag* de segmentação igual a 0, λR_0, que completará o custo dos nós-filhos, a taxa do *flag* 0 é calculada com os seguintes contadores:$\sum C_{D_F}, \sum \hat{C}_{D_F}, C_{F_0}, \hat{C}_{F_0}$.

Passo 8: Incremente o contador do *flag* de segmentação igual a 0, $\hat{C}_{F_0}$.

Passo 9: **Se**$(C > L)$ **entao** divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{C}{2}$ e chame a `OtimizaT-Sided` na ordem:

$\left[B^{\hat{2j+1}},\, A(n_0)_a\right]$ = `OtimizaT-Sided`(L, $\frac{C}{2}$, B^{2j+1}, $2\,no+1$, $A(n_0)$,V_{1l},V_{1c})

$\left[B^{\hat{2j+2}},\, A(n_0)_b\right]$ = `OtimizaT-Sided`(L, $\frac{C}{2}$, B^{2j+2}, $2\,no+2$, $A(n_0)$,V_{2l},V_{2c})

Se Não divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{L}{2}$ e chame o procedimento de `OtimizaT-Sided` na ordem:

$\left[B^{\hat{2j+1}},\, A(n_0)_a\right]$ = `OtimizaT-Sided`($\frac{L}{2}$, C, B^{2j+1}, $2\,no+1$, $A(n_0)$,V_{1l},V_{1c})

$\left[B^{\hat{2j+2}},\, A(n_0)_b\right]$ = `OtimizaT-Sided`($\frac{L}{2}$, C, B^{2j+2}, $2\,no+2$, $A(n_0)$,V_{2l},V_{2c})

Passo 10: Faz $A(n_0) = A(n_0)_a$ **e** $A(n_0)_b$.

Passo 11: Se o custo do nó pai for menor ou igual ao custo dos nós filhos então não devemos continuar a segmentação. Matematicamente temos:

$$\{J_{no}\} \leq \left\{J_{(2\,no+1)} + J_{(2\,no+2)} + \lambda R_0(no)\right\}$$

Se a sentença for verdadeira siga para o **Passo 12**.

Se não vá para o **Passo 18**.

Passo 12: Decrementa os contadores do flag 0 em todas as dimensões relacionadas as árvores $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$ a serem podadas.

Passo 13: Decrementa os contadores $\hat{C}_{F_1}, \hat{C}_D$ ou $\hat{C}_{D_R}$ para todas as dimensões relacionadas aos nós-folhas das árvores $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$ e nas devidas posições dos seus índices, devendo verificar a origem de cada elemento. Se pertence ao dicionário oficial D ou se pertence ao dicionário rascunho D_R.

Passo 14: Decrementa o contador $\hat{C}_{F_0}$.

Passo 15: Incrementa os contadores $\hat{C}_{F_1}, \hat{C}_i$ ou C_{i_R}, dependendo do elemento utilizado S como aproximação. Se pertence ao dicionário oficial D ou se pertence ao dicionário rascunho D_R.

Passo 16: Elimina as atualizações do dicionário rascunho D_R que foram inseridas devido às árvore $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$.

Passo 17: Seta em $A(n_0)$ que as árvores $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$ foram eliminadas, observando as coordenadas de V_l e V_c.
Retorne $\left[\hat{B^j},\, A(n_0)\right]$.

Passo 18: Realize a concatenação dos blocos $B^{\hat{2j+1}}$ e $B^{\hat{2j+2}}$ da seguinte forma:
Se$(C > L)$ **entao** concatene na vertical, ou seja:

$$\hat{B^j} = \begin{pmatrix} B^{\hat{2j+1}} \\ B^{\hat{2j+2}} \end{pmatrix}$$

Se Não concatene na horizontal, ou seja:

$$\hat{B^j} = \begin{pmatrix} B^{\hat{2j+1}} & B^{\hat{2j+2}} \end{pmatrix}$$

Passo 19: Atualize o dicionário rascunho D_R em todas as dimensões da seguinte forma:
Para $n = (0, 1, \ldots, w-1)$ **faça**

$$L_x = 2^{\lfloor 0.5(n+1) \rfloor}$$
$$C_x = 2^{\lfloor 0.5n \rfloor}$$
$$S_{L_x, C_x} = \{S_{L_x, C_x}\} \cup \left\{T^{L,C}_{L_x, C_x}\left[\hat{B^j}\right]\right\}$$

Passo 20: Atualize o custo do nó.

$$\{J_{no}\} = \left\{J_{(2\,no+1)} + J_{(2\,no+2)} + \lambda R_0(no)\right\}$$

Passo 21: **Retorne** $\left[\hat{B^j},\, A(n_0)\right]$.

Procedimento $\hat{B^j}$ = `CodificaThree-Sided`(N, M, B^j, no, $A(n_0)$,V_l,V_c)

Passo 1: **Se** ($A(n_{2\,no+1}) = 0$ e $A(n_{2\,no+2}) = 0$ **ou** L=C=1) **então** segue para o **Passo 2** seguinte.

Se não vai para o **Passo 6**.

Passo 2: Computa $[Freq_D] = Gera\_D_{est}$(N, M, V_l,V_c).

Passo 3: Localizar no dicionário oficial D o índice i do bloco S^i com mesma dimensão (L, C) do bloco de entrada B^j, utilizando $Freq_D = 1$, e a melhor aproximação deste bloco (B^j), segundo o menor custo *Lagrangeano* $J_{n_j} = D_{n_j} + \lambda RI_{n_j}$. Guarde o valor do índice i do elemento S^i escolhido e faça $\hat{B^j} = S_i$. O custo é calculado utilizando os seguintes contadores: $\sum C_D, C_i$, para todo elemento pertencente à $Freq_D = 1$.

Passo 4: **Se**($L = C = 1$) **então** guarde o índice i do elemento escolhido S^i e retorne o bloco aproximado $\hat{B^j}$.

Se Não Vai para o **Passo 5**.

Passo 5: Codifica o *flag* de valor igual a 1, o índice i do elemento S^i escolhido pela aproximação e retorna o bloco $\hat{B^j}$.

Passo 6: Codifica o *flag* de valor igual a 0.

Se($C > L$) **entao** divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{C}{2}$ e chame o procedimento de codificação na ordem:

$\hat{B^{2j+1}}$ = `CodificaThree-Sided`(L, $\frac{C}{2}$, B^{2j+1}, $2\,no+1$, $A(n_0)$,V_{1l},V_{1c})

$\hat{B^{2j+2}}$ = `CodificaThree-Sided`(L, $\frac{C}{2}$, B^{2j+2}, $2\,no+2$, $A(n_0)$,V_{2l},V_{2c})

Se Não divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{L}{2}$ e chame o procedimento de codificação na ordem:

$\hat{B^{2j+1}}$ = `CodificaThree-Sided`($\frac{L}{2}$, C, B^{2j+1}, $2\,no+1$, $A(n_0)$,V_l,V_c)

$\hat{B^{2j+2}}$ = `CodificaThree-Sided`($\frac{L}{2}$, C, B^{2j+2}, $2\,no+2$, $A(n_0)$,V_{2l},V_{2c})

Passo 7: Faça a concatenação dos blocos $\hat{B^{2j+1}}$ e $\hat{B^{2j+2}}$ da seguinte forma:

Se($C > L$) **entao** concatene na vertical,

$$\hat{B^j} = \begin{pmatrix} \hat{B^{2j+1}} \\ \hat{B^{2j+2}} \end{pmatrix}$$

Se Não concatene na horizontal,

$$\hat{B}^j = \left(\begin{array}{cc} B^{2\hat{j}+1} & B^{2\hat{j}+2} \end{array} \right)$$

Passo 8: Atualize o dicionário oficial D em todas as dimensões da seguinte maneira:

Para $n = (0, 1, \ldots, w-1)$ **faça**

$$L_x = 2^{\lfloor 0.5(n+1) \rfloor}$$
$$C_x = 2^{\lfloor 0.5n \rfloor}$$
$$S_{L_x, C_x} = \{S_{L_x, C_x}\} \cup \left\{ T^{L,C}_{L_x, C_x} \left[\hat{B}^j \right] \right\}$$

Passo 9: Retorne $\hat{B}^j$.

5.2 Resultados de simulações

Nesta sessão serão apresentados os resultados obtidos das simulações dos algoritmos MMP, MMP-RDI, MMP-U, SM-MMP e GSM-MMP, que é a técnica desenvolvida neste trabalho de dissertação. Todos estes algoritmos foram implementados em linguagem C, e executados em ambiente *Linux*. Estes resultados são mostrados por curvas taxa-distorção onde na coordenada abcissa tem-se a taxa dada em *bits* por *pixel* (*bbp*) e na coordenada ordenada a distorção dada pela PSNR, como definido anteriormente na Sessão 2.2.

Foram utilizadas as seguintes imagens para o teste: lena, aerial, gold, barbara, f16, pp1205,pp1209, todas com tamanho de (512×512) *pixels*, no formato *pgm*, que foram utilizadas no capítulo anterior. Estas imagens foram adquiridas no *site* http://sipi.usc.edu/service/database/Database.html com exceção das imagens pp1205 e pp1209 que foram digitalizadas do IEEE *Transactions on Image Processing*, volume 09, número 07, mês de julho, ano 2000, as páginas correspondem aos nomes das imagens. Todas essas imagens foram divididas em blocos de (16×16) para serem processadas bloco a bloco pelo algoritmo MMP e suas versões na seqüencia da esquerda para direita e de cima para baixo, e estas imagens podem ser observadas no apêndice A.

Os resultados obtidos pelo algoritmo GS-MMP além de serem comparados com os resultados do MMP padrão e as demais versões, também serão comparados com os resultados dos algoritmos JPEG [6] e SPIHT [37] que são codificadores baseados em DCT e Wavelets respectivamente.

Para uma análise subjetiva serão apresentadas as imagens lena, pp1205 e pp1209 processadas pelos algoritmos GSM-MMP, SM-MMP, MMP-RDI, MMP padrão e SPIHT. Será apresentada também a imagem lena comprimida a 0.3 *bpp* pelos algoritmos GSM-MMP e MMP com o intuito de compararmos a melhora do efeito de blocagem. E será apresentado uma tabela no final desta Seção para ser analisado o desempenho global do algoritmo GSM-MMP, tendo como referência o valor de $PSNR$ para a taxa de 0.5 bpp.

Os resultados obtidos dos algoritmos neste capítulo, serão apresentados tanto na forma original como na forma ampliada para uma melhor visualização nos detalhes das mesmas, como no Capítulo 4, devido à aproximação das curvas. Por questão de simplicidade, tais figuras serão referenciadas pelos seus respectivos nomes que estão localizados em cima de cada figura exposta.

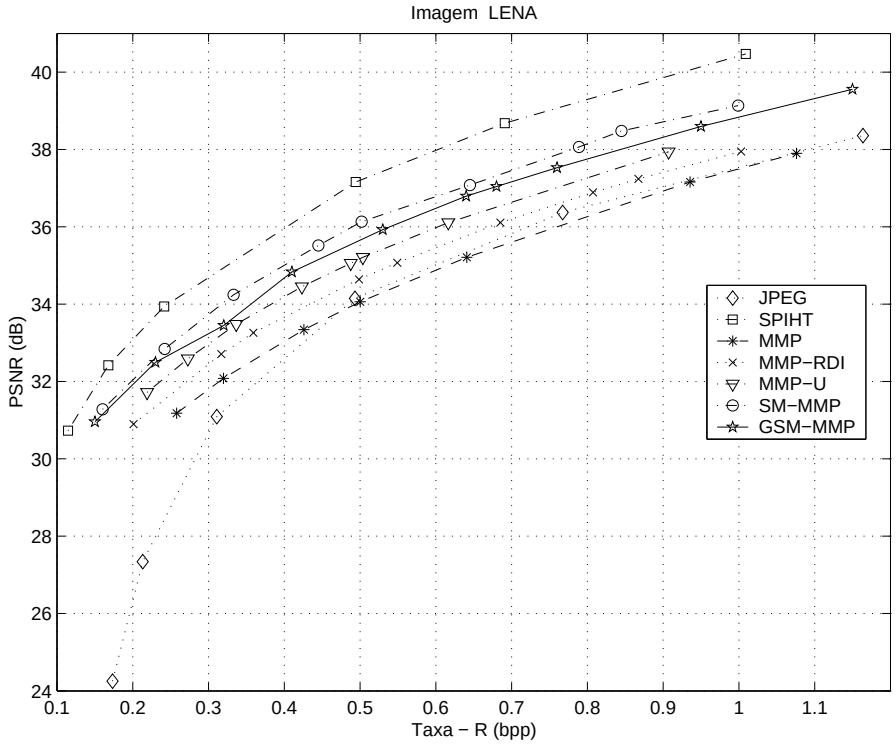


Figura 5.4: Taxa x distorção para a imagem Lena 512 x 512

A Figura 5.4 apresenta a curva da imagem lena processada pelos algoritmos já mencionados, e a Figura 5.5 mostra em detalhes a curva taxa-distorção da imagem lena e como pode-se observar o algoritmo GSM-MMP obteve um aumento de 1dB em relação ao algoritmo MMP-RDI e 0.5 dB em relação ao algoritmo MMP-U, e em 0.5 bpp obteve um ganho negativo de 0.5 dB em relação ao algoritmo SM-MMP.

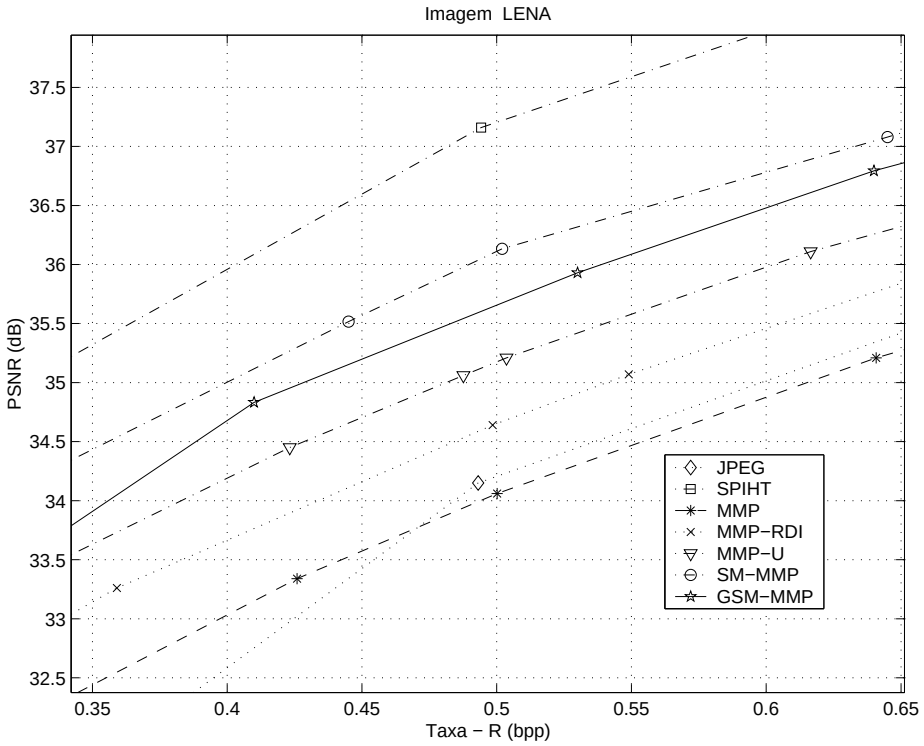


Figura 5.5: Zoom da Taxa x distorção para a imagem Lena 512 x 512

As Figuras expostas à seguir de 5.6 a 5.15 apresentam a imagem lena, comprimida a uma taxa de 0.5 bpp pelos algoritmos MMP, MMP-RDI, SM-MMP, GSM-MMP e SPIHT, acompanhadas pelas imagens que mostram uma visão detalhada das áreas de baixa freqüência de cada imagem apresentada. Para a imagem lena o algoritmo GSM-MMP obteve um aumento em PSNR em relação aos algoritmos MMP de 1.42 dB e MMP-RDI de 0.79 dB, e para o SM-MMP obteve um ganho inferior de 0.46 dB. Também houve uma melhoria na qualidade subjetiva que pode ser observada nas Figuras de 5.6 a 5.15, sendo que o SPIHT apresenta os melhores resultados para esta imagem por se tratar de uma imagem suave.

Figura 5.6: Lena comprimida pelo MMP a taxa de 0.5 bpp. PSNR=34.25dB

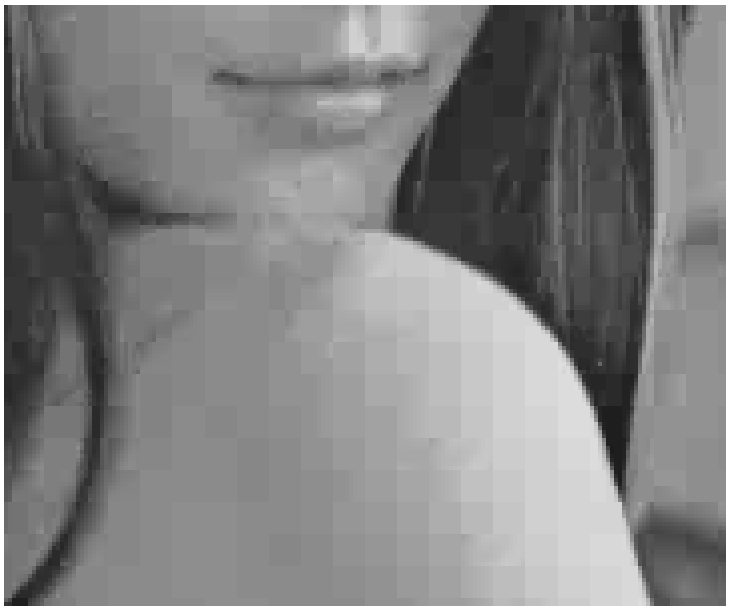


Figura 5.7: Visão detalhada do ombro e parte da face da Figura 5.6

Figura 5.8: Lena comprimida pelo MMP-RDI a taxa de 0.5 bpp. PSNR=34.88dB

Figura 5.9: Visão detalhada do ombro e parte da face da Figura 5.8

Figura 5.10: Lena comprimida pelo SM-MMP a taxa de 0.5 bpp. PSNR=36.13dB

Figura 5.11: Visão detalhada do ombro e parte da face da Figura 5.10

Figura 5.12: Lena comprimida pelo GSM-MMP a taxa de 0.5 bpp. PSNR=35.67dB

Figura 5.13: Visão detalhada do ombro e parte da face da Figura 5.12

Figura 5.14: Lena comprimida pelo SPIHT a taxa de 0.5 bpp. PSNR=37.22dB

Figura 5.15: Visão detalhada do ombro e parte da face da Figura 5.14

A Figura 5.16 e a Figura 5.18, apresentam em ordem as imagens lena comprimida na taxa de 0.3bpp pelo algoritmo MMP-RDI e GSM-MMP. As figuras 5.17 e 5.19 mostram em detalhes o efeito blocagem das figuras 5.16 e 5.18 respectivamente à taxa de 0.3bpp. A compressão nesta taxa tem como objetivo verificar o efeito blocagem que existe nos algoritmos que utilizam a técnica do MMP. Na imagem comprimida pelo algoritmo GSM-MMP verifica-se a melhora do efeito blocagem da imagem lena em comparação ao MMP-RDI, ou seja na suavidade da imagem lena, como pode ser observada na Figura 5.18. Pode-se perceber esta melhora, observando os detalhes da imagem, como nos contornos dos lábios, olhos, cílios, no nariz.

Figura 5.16: Lena comprimida pelo MMP-RDI a taxa de 0.3 bpp.

Figura 5.17: Visão detalhada do ombro e parte da face da Figura 5.16

Figura 5.18: Lena comprimida pelo GSM-MMP a taxa de 0.3 bpp.

Figura 5.19: Visão detalhada do ombro e parte da face da Figura 5.18

O que pode ser observado para a imagem aerial da Figura 5.20 é que em taxas baixas de bpp o algoritmo GSM-MMP teve um excelente resultado pois se aproximou do resultado do algoritmo SPIHT que é o estado da arte na área de compressão de imagens suaves, pois o mesmo possui a característica de supor que as imagens a serem processadas por ele pertencem a esta categoria, o que não acontece com o GSM-MMP.

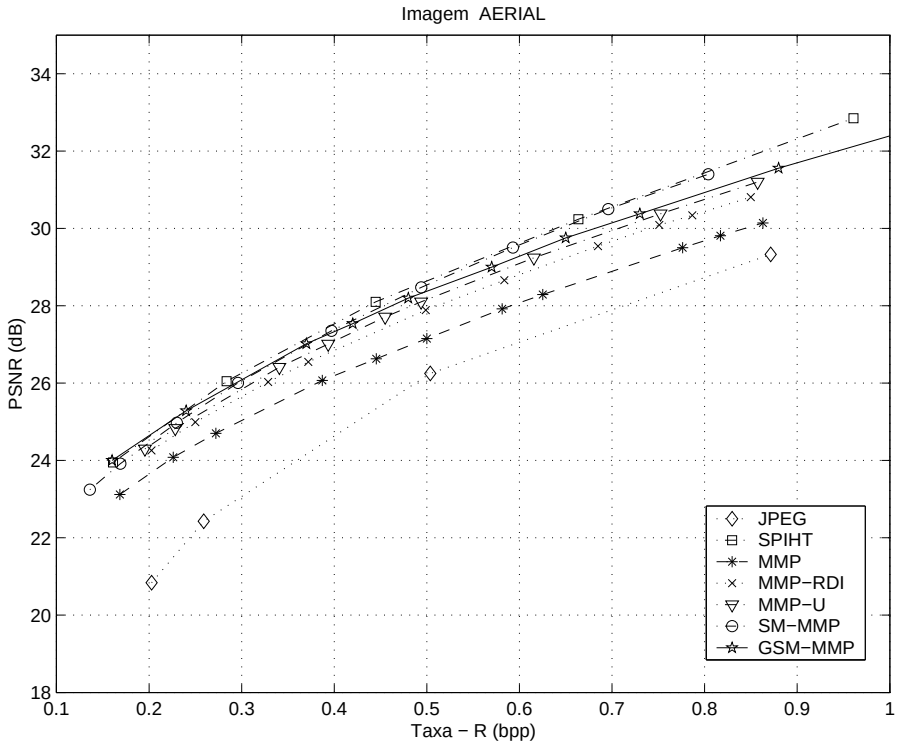


Figura 5.20: Taxa x distorção para a imagem Aerial 512 x 512

Na Figura 5.21 constata-se essa aproximação em dB da imagem aerial com mais detalhes dos resultados dos algoritmos GSM-MMP, SM-MMP, e SPIHT.

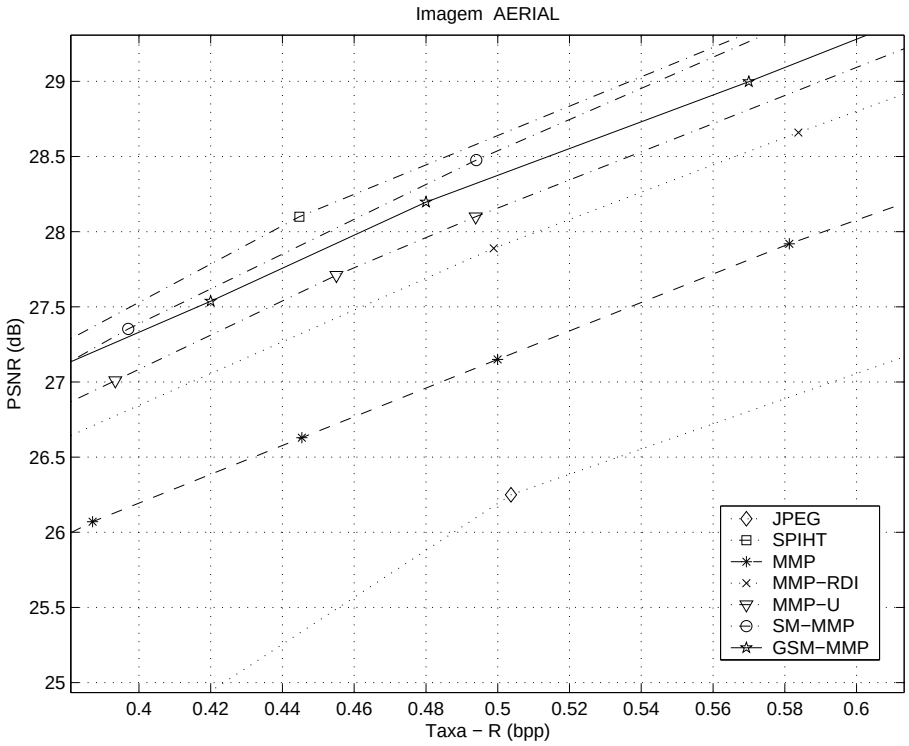


Figura 5.21: Zoom da Taxa x distorção para a imagem Aerial 512 x 512

A Figura 5.22, apresenta o gráfico da taxa-distorção para a imagem barbara onde são apresentadas as curvas relativas aos resultados dos algoritmos analisados. Para a imagem barbara o algoritmo GSM-MMP obteve um ganho de aproximadamente 0.5 dB em relação ao algoritmo MMP-RDI e um resultado equiparado aos algoritmos SM-MMP e MMP-U.

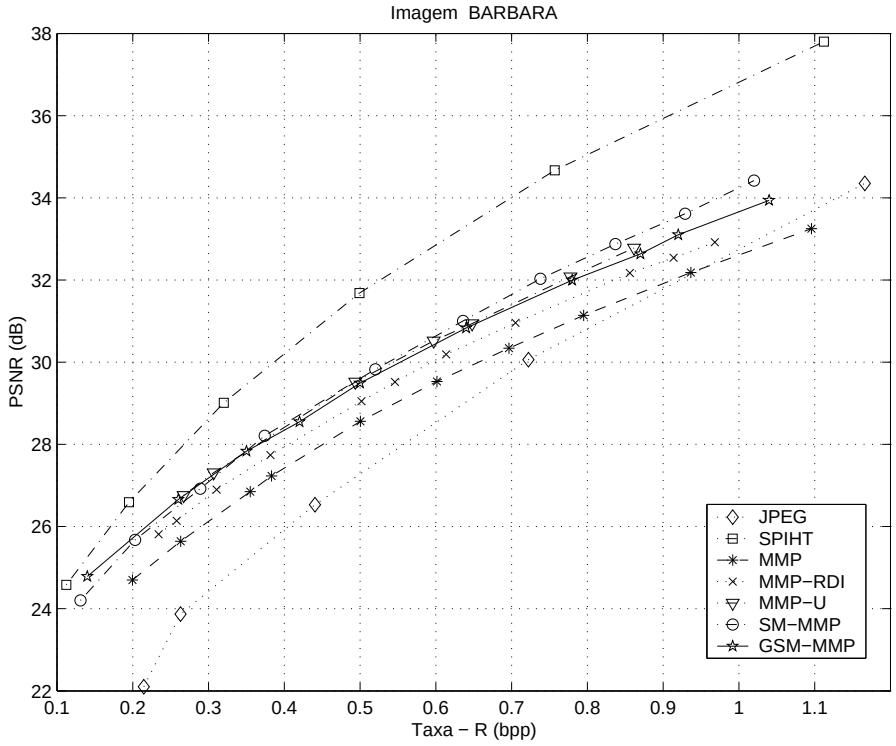


Figura 5.22: Taxa x distorção para a imagem Bárbara 512 x 512

A Figura 5.23 mostra o zoom da imagem barbara. Na taxa de 0.5 bpp os resultados dos algoritmos GSM-MMP, SM-MMP e MMP-U mostram resultados bem próximos em PSNR, comparando-os com o algoritmo SPIHT estes obtiveram um ganho inferior de aproximadamente 2 dB.

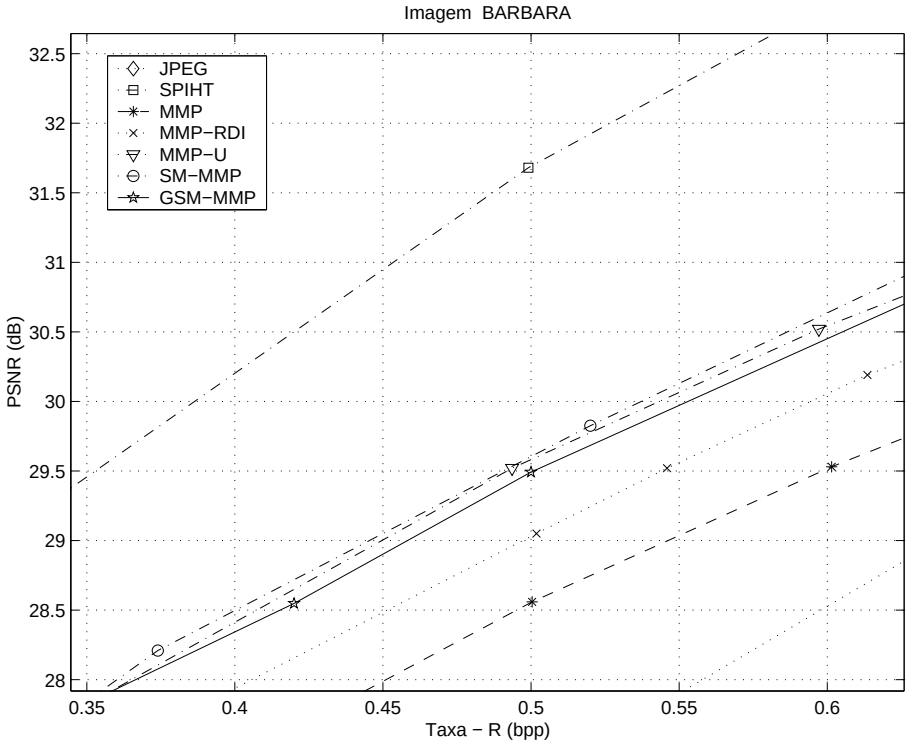


Figura 5.23: Zoom da Taxa x distorção para a imagem Bárbara 512 x 512

A Figura 5.24 mostra a curva taxa-distorção para a imagem gold. Para esta imagem o algoritmo GSM-MMP chegou bem próximo dos resultados obtidos pelo algoritmo baseado em *Wavelets* que no caso é o SPIHT, e superou em dB as versões do MMP. Demonstrando que é válido a idéia de explorar este tipo de algoritmo que utiliza a vizinhança para realizar a codificação das imagens.

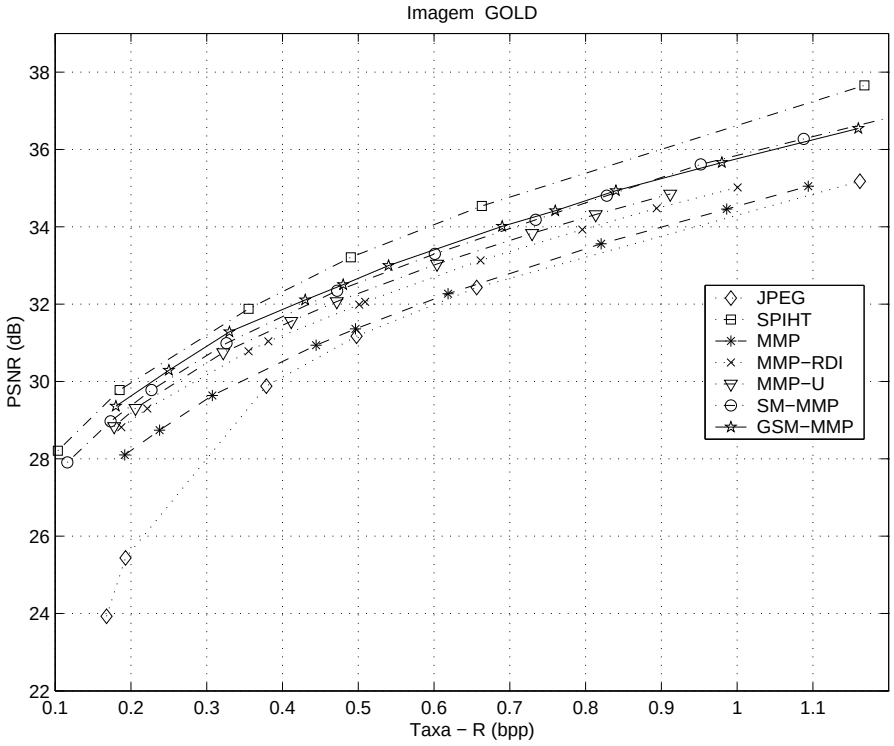


Figura 5.24: Taxa x distorção para a imagem Gold 512 x 512

Pode-se perceber pelas curvas detalhadas da imagem gold na Figura 5.25 que o resultado do algoritmo GSM-MMP na taxa de 0.5 bpp superou em 0.6 dB o algoritmo RDI-MMP e em aproximadamente 0.4 dB o algoritmo MMP-U e 0.12 dB o algoritmo SM-MMP.

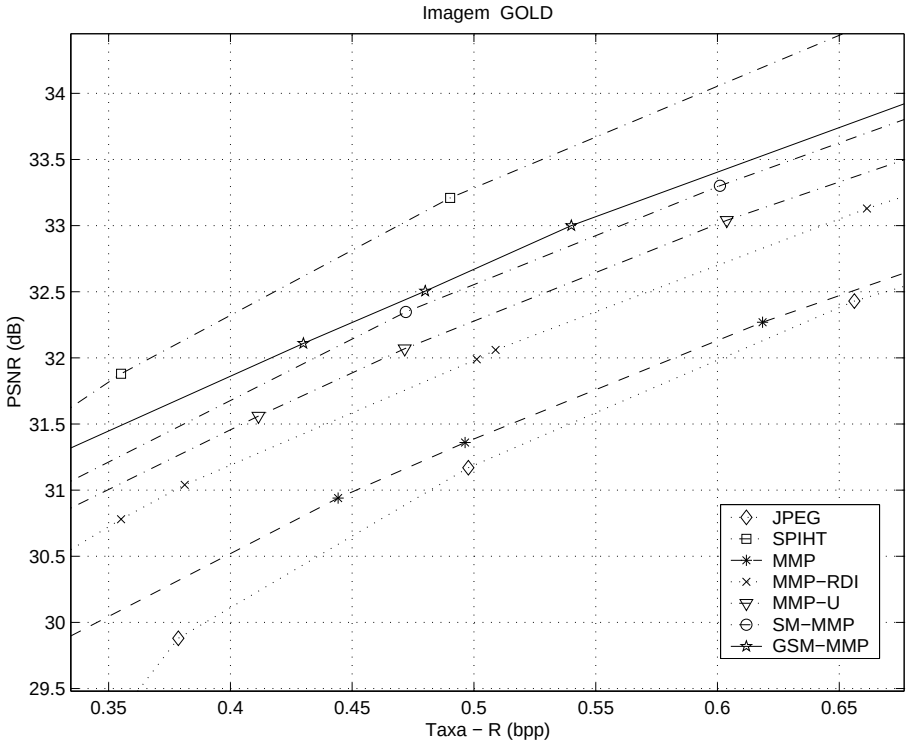


Figura 5.25: Zoom da Taxa x distorção para a imagem Gold 512 x 512

Observa-se que na imagem F16 as versões do algoritmo MMP chegaram bem próximo dos resultados obtidos pelo algoritmo SPIHT. Demonstrando mais uma vez que tal técnica aplicada ao GSM-MMP é de fundamental importância para ser explorada.

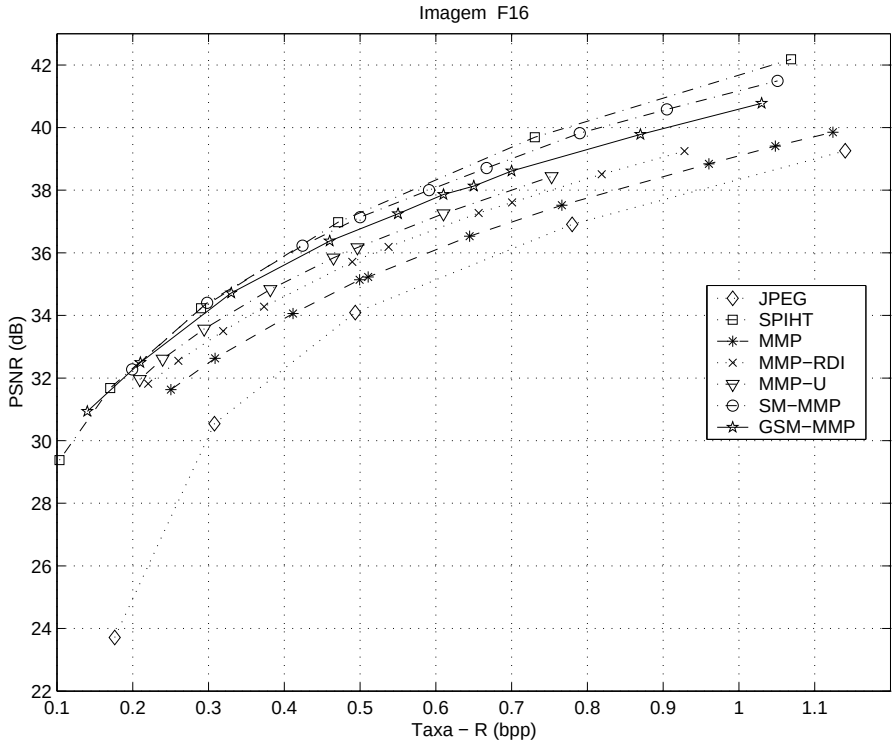


Figura 5.26: Taxa x distorção para a imagem F16 512 x 512

A Figura 5.27, mostra a ampliação da Figura 5.26. Onde percebe-se em detalhes a aproximação das versões do algoritmo MMP com o algoritmo SPIHT. Nesta imagem como nas demais o algoritmo JPEG tem apresentado um desempenho muito baixo comparado com os demais algoritmos.

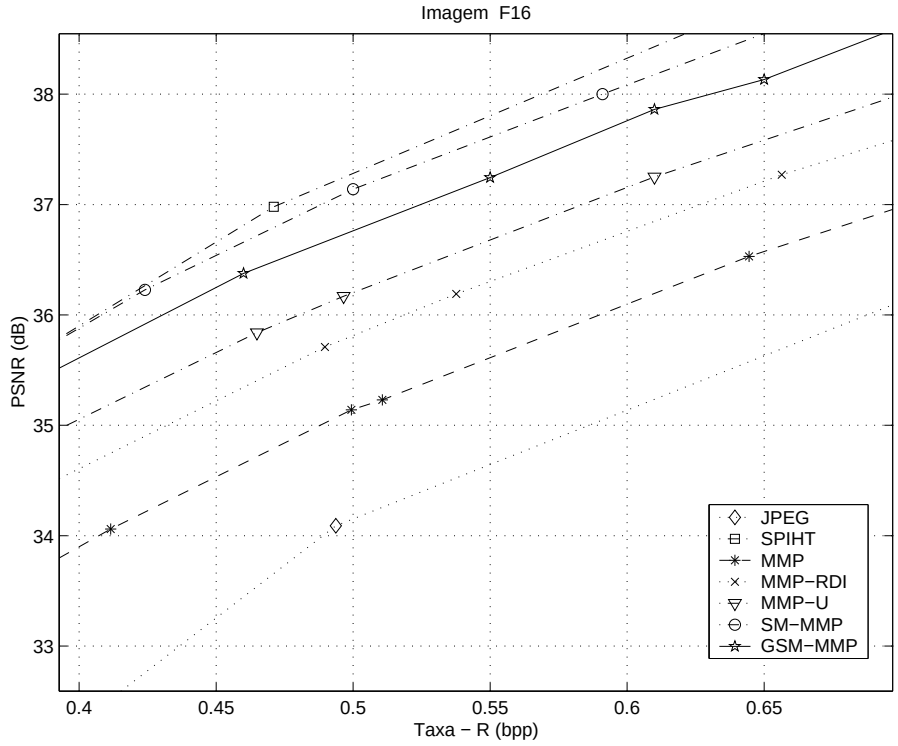


Figura 5.27: Zoom da Taxa x distorção para a imagem F16 512 x 512

Na Figura 5.28, é apresentado o gráfico da curva taxa-distorção para a imagem pp1205. Para esta respectiva imagem o algoritmo GSM-MMP teve um ganho maior em nível de PSNR entre as taxas de 0.1 bpp a 0.3 bpp em relação aos algoritmos MMP-RDI,SM-MMP e MMP-U, em 0.5 bpp obteve um resultado aproximado aos algoritmos MMP-RDI,SM-MMP e MMP-U, mas para uma taxa maior que 0.5 bpp teve um ganho inferior em dB.

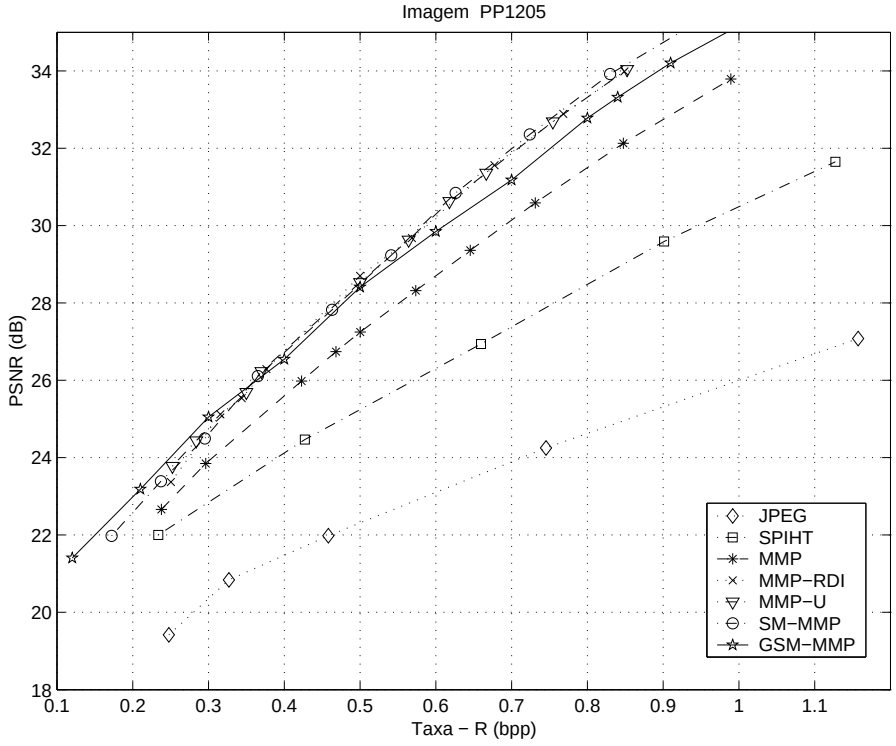


Figura 5.28: Taxa x distorção para a imagem PP1205 512 x 512

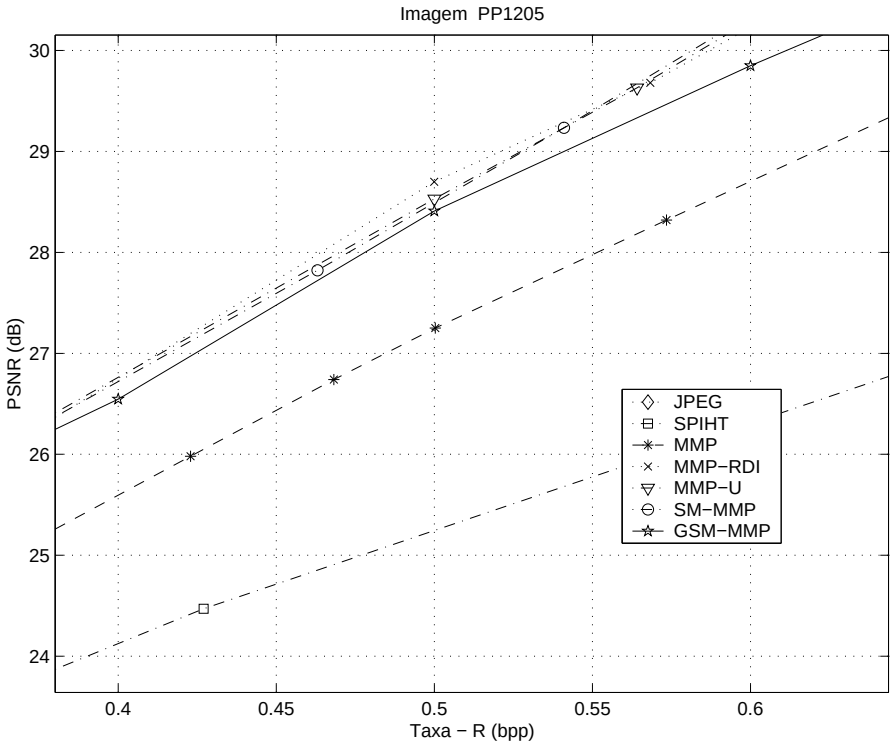


Figura 5.29: Zoom da Taxa x distorção para a imagem PP1205 512 x 512

Na Figura 5.29, que é a ampliação da Figura 5.28, percebe-se que tanto o algoritmo MMP padrão e suas versões apresentaram resultados superiores em nível de dB quando comparados aos algoritmos SPIHT e JPEG.

Nas Figuras de 5.30 a 5.34 mostram a imagem pp1205 onde a mesma é composta somente por texto. Para esta imagem o SPHIT não apresentou bons resultados subjetivos, como pode-se observar na Figura 5.34 que mostra uma imagem ruidosa comparado com o MMP e as suas versões. O GSM-MMP obteve uma leve melhora subjetiva comparada com o MMP e MMP-RDI, chegando a se equiparar com o GSM-MMP.

Figura 5.30: PP1205 comprimida pelo MMP a taxa de 0.5 bpp. PSNR=27.45dB

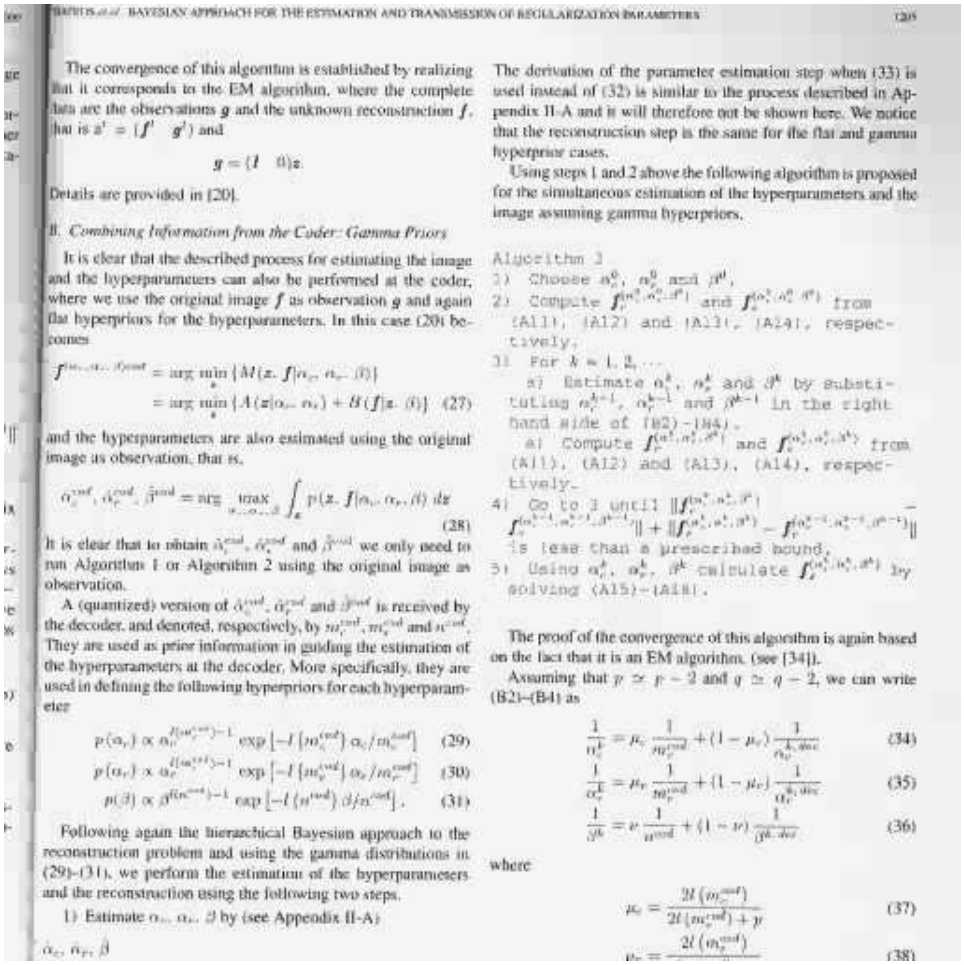


Figura 5.31: PP1205 comprimida pelo MMP-RDI a taxa de 0.5 bpp. PSNR=28.50dB

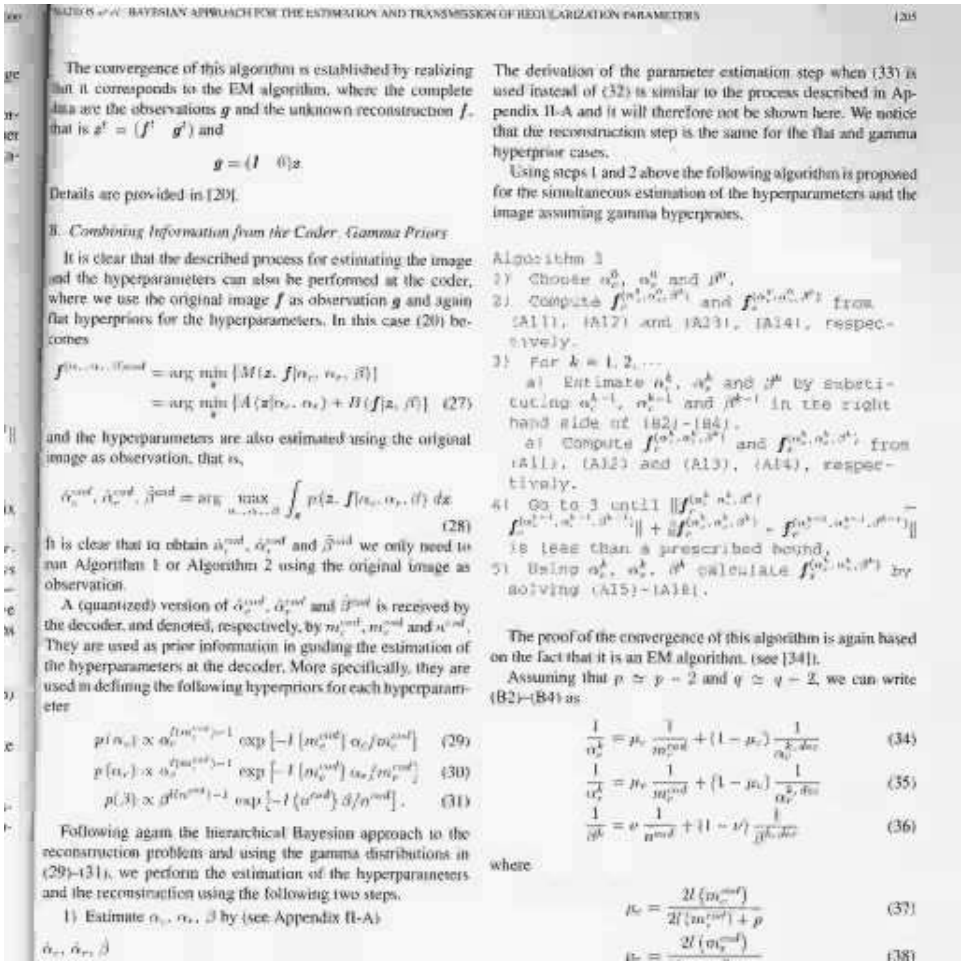


Figura 5.32: PP1205 comprimida pelo SM-MMP a taxa de 0.5 bpp. PSNR=28.54dB

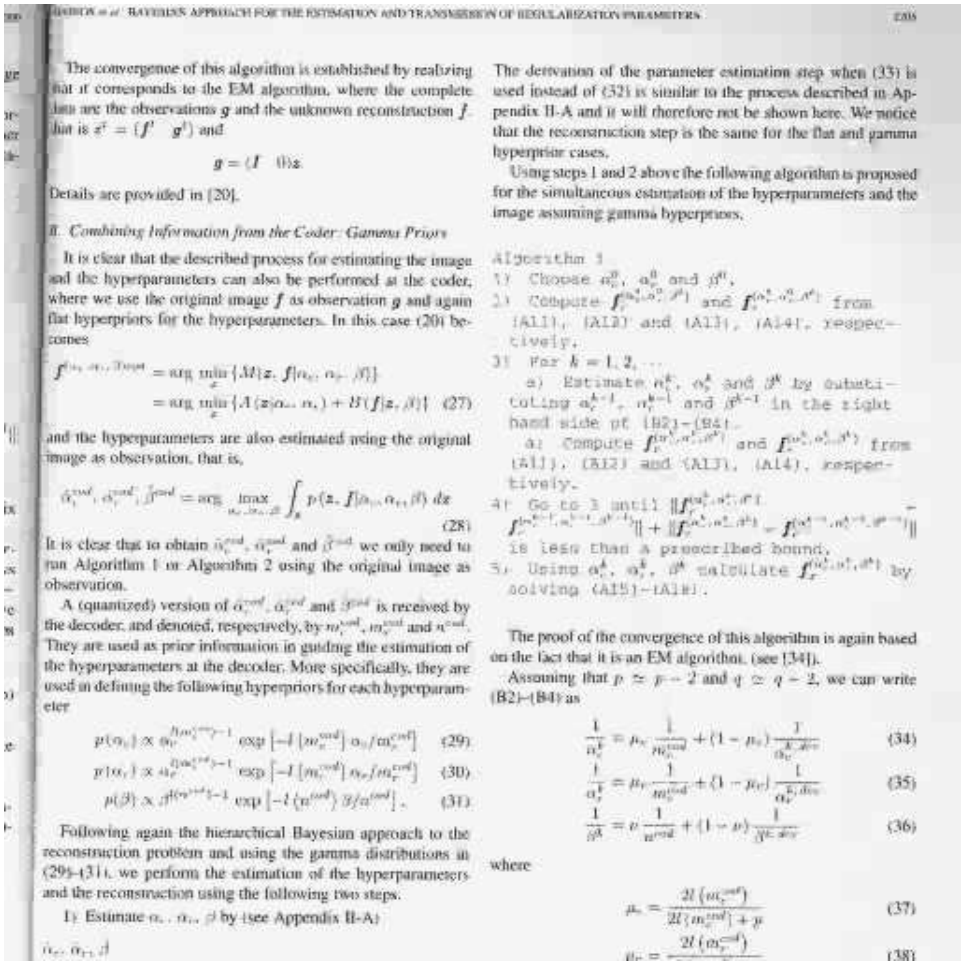


Figura 5.33: PP1205 comprimida pelo GSM-MMP a taxa de 0.5 bpp. PSNR=28.41dB

Figura 5.34: PP1205 comprimida pelo SPIHT a taxa de 0.5 bpp. PSNR=25.21dB

As Figuras 5.35 e 5.36, que mostram a imagem pp1209 e sua ampliação respectivamente, ainda constatam os desempenhos satisfatórios dos algoritmos baseados no MMP. O algoritmo GSM-MMP obteve ganhos positivos em PSNR de 0.25 dB e 0.2 dB comparado com os algoritmos MMP-RDI e MMP-U respectivamente e ganho negativo em aproximadamente 0.4 dB comparado com o SM-MMP, isto pode ser percebido analisando a Figura 5.36.

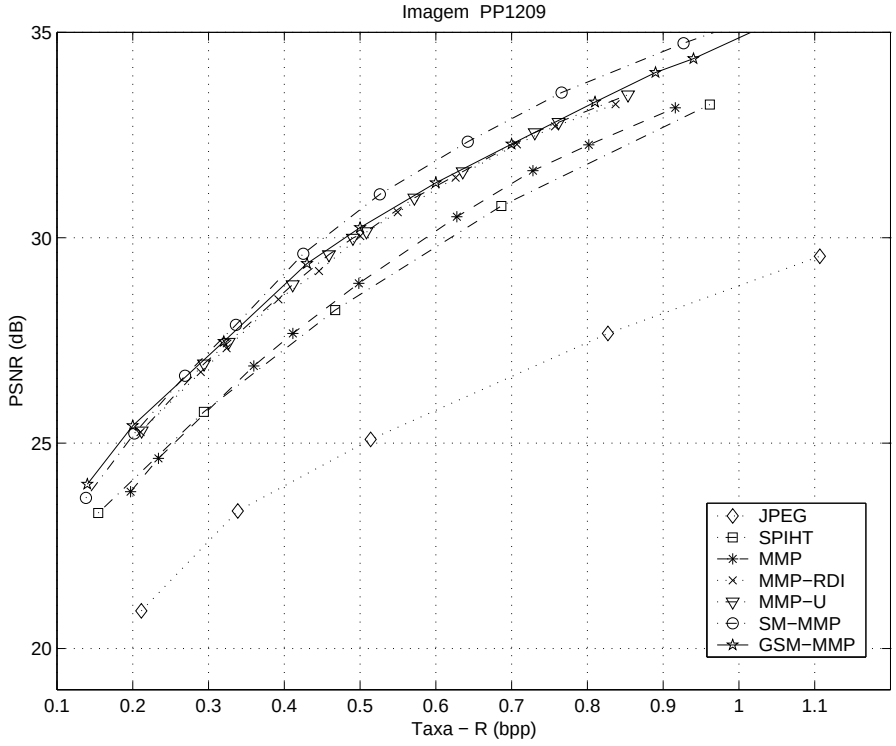


Figura 5.35: Taxa x distorção para a imagem PP1209 512 x 512

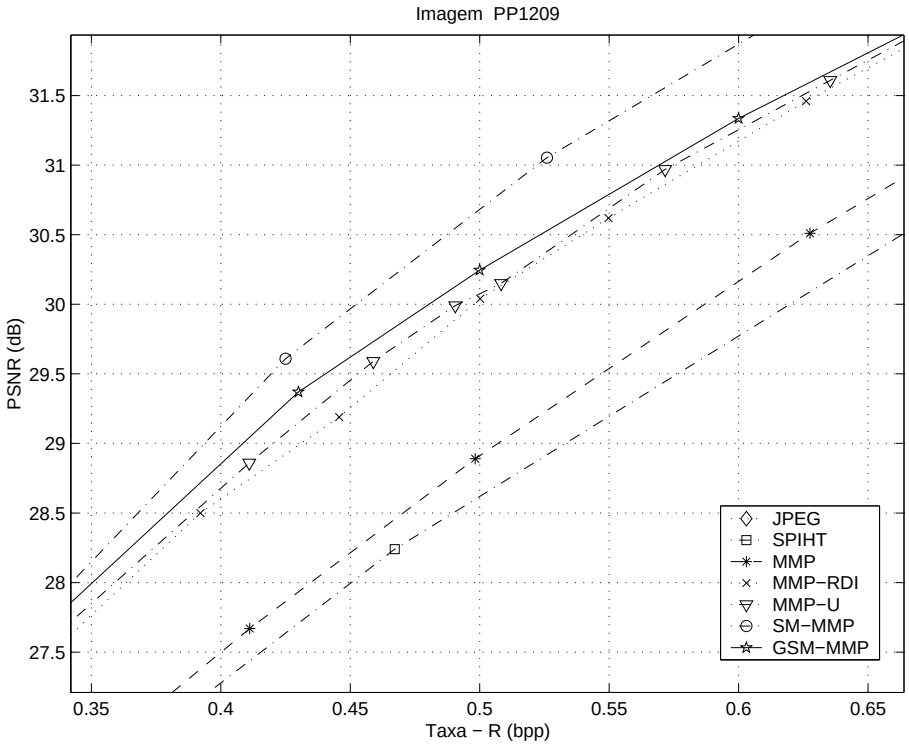
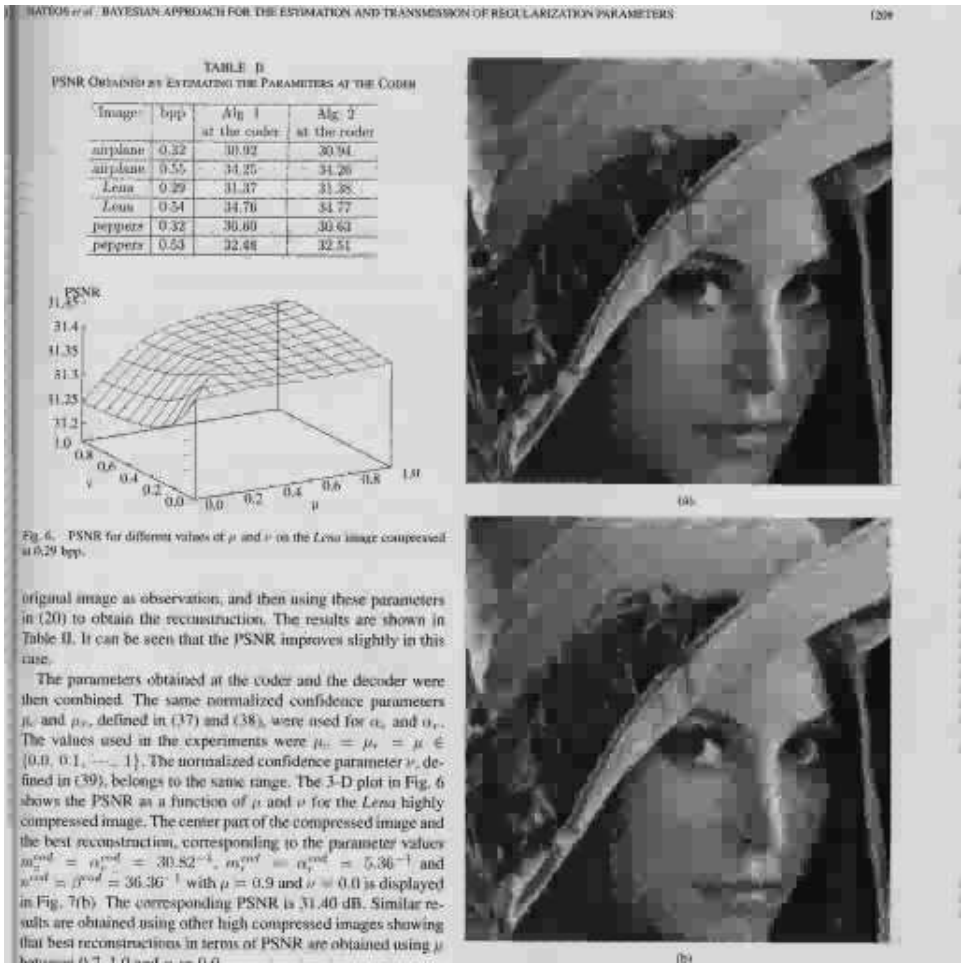


Figura 5.36: Zoom da Taxa x distorção para a imagem PP1209 512 x 512

A imagem PP1209 é uma imagem composta de texto, gráficos e imagens em nível de cinza, como pode ser verificada nas Figuras 5.37 a 5.41. O algoritmo GSM-MMP obteve um melhor desempenho em nível de PSNR comparado com o SPIHT devido ao ganho na região de texto que compensou a área onde se localiza as regiões de imagens de nível de cinza, que no caso são as imagens lenas.



RATOS *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.

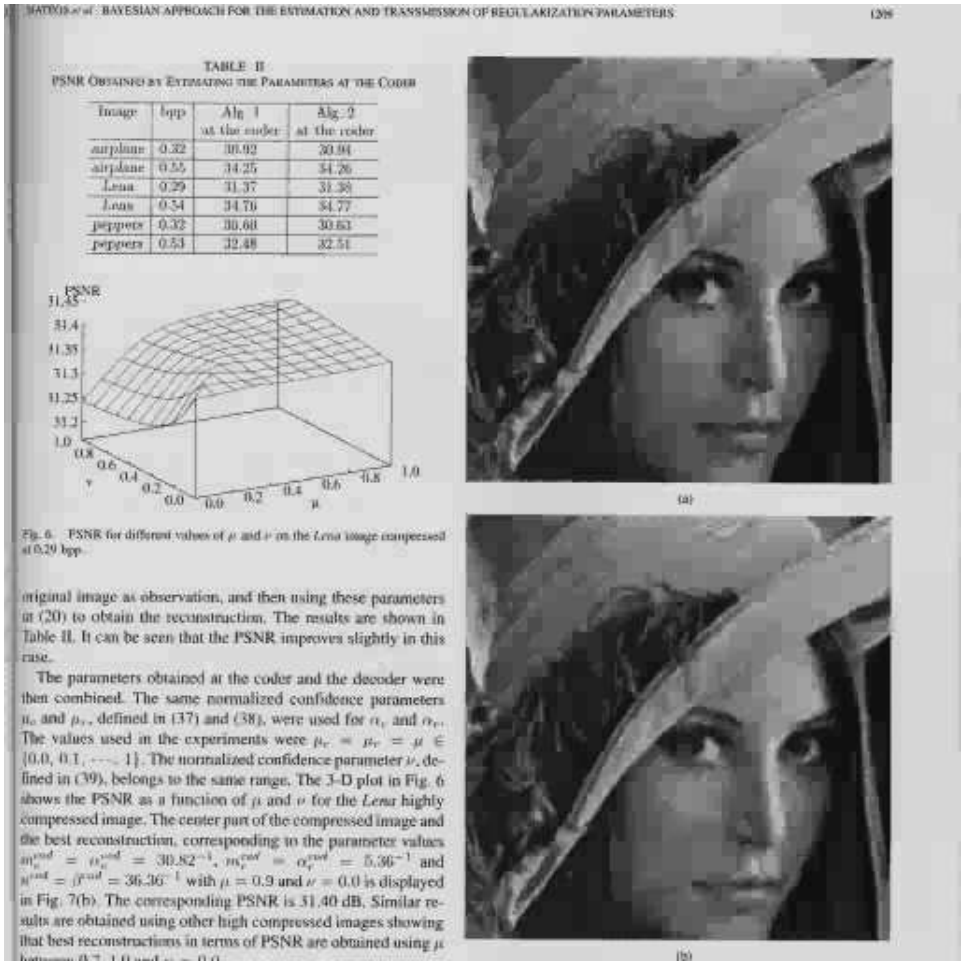
(a)

(b)

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_ν, defined in (37) and (38), were used for α_c and α_ν. The values used in the experiments were $\mu_c = \mu_\nu = \mu \in \{0.0, 0.1, \cdots, 1\}$. 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 $m_c^{cod} = \alpha_c^{cod} = 30.82^{-1}$, $m_\nu^{cod} = \alpha_\nu^{cod} = 5.36^{-1}$ and $\nu^{cod} = \beta^{cod} = 36.36^{-1}$ 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

Figura 5.37: PP1205 comprimida pelo MMP a taxa de 0.5 bpp. PSNR=29.08dB



MATAS *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.

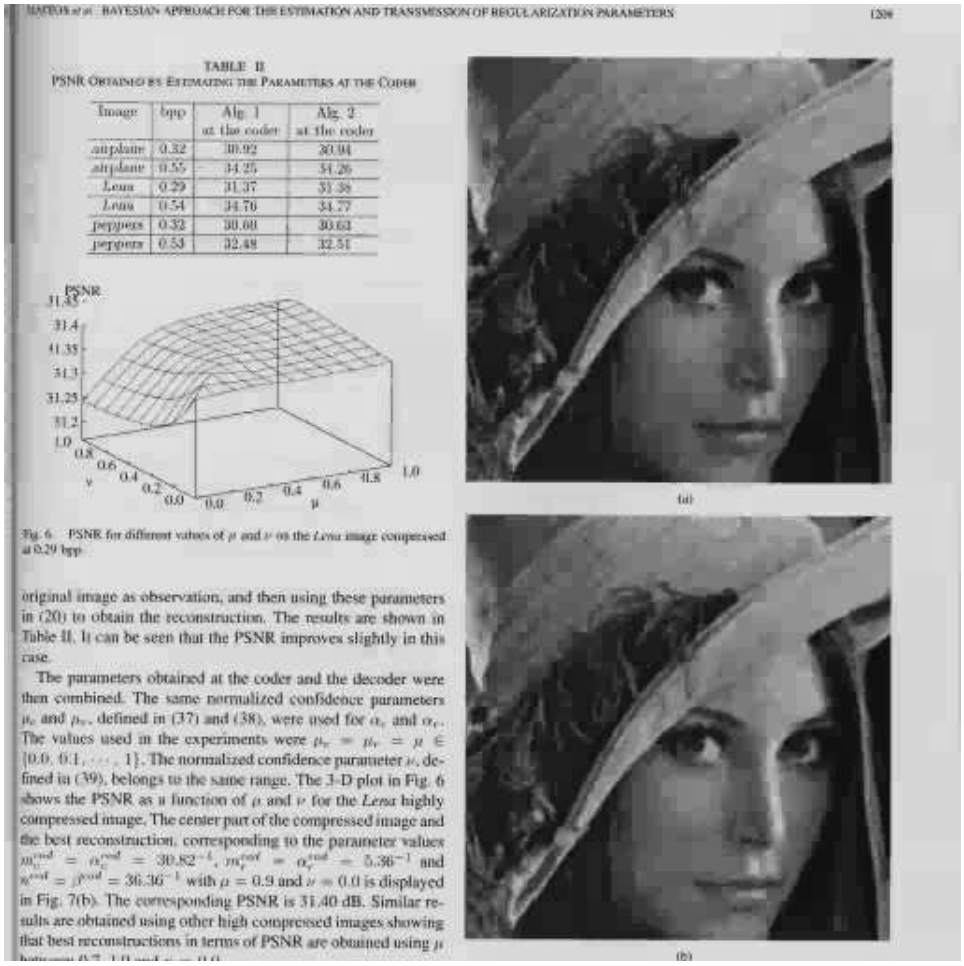
original image as observation, and then using these parameters in (20) to obtain the reconstruction. The results are shown in Table II. It can be seen that the PSNR improves slightly in this case.

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 $m_c^{cod} = v_c^{cod} = 30.82^{-1}$, $m_r^{cod} = \alpha_r^{cod} = 5.36^{-1}$ and $\beta^{cod} = 36.36^{-1}$ 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 5.38: PP1209 comprimida pelo MMP-RDI a taxa de 0.5 bpp. PSNR=30.01dB



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	30.61
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.

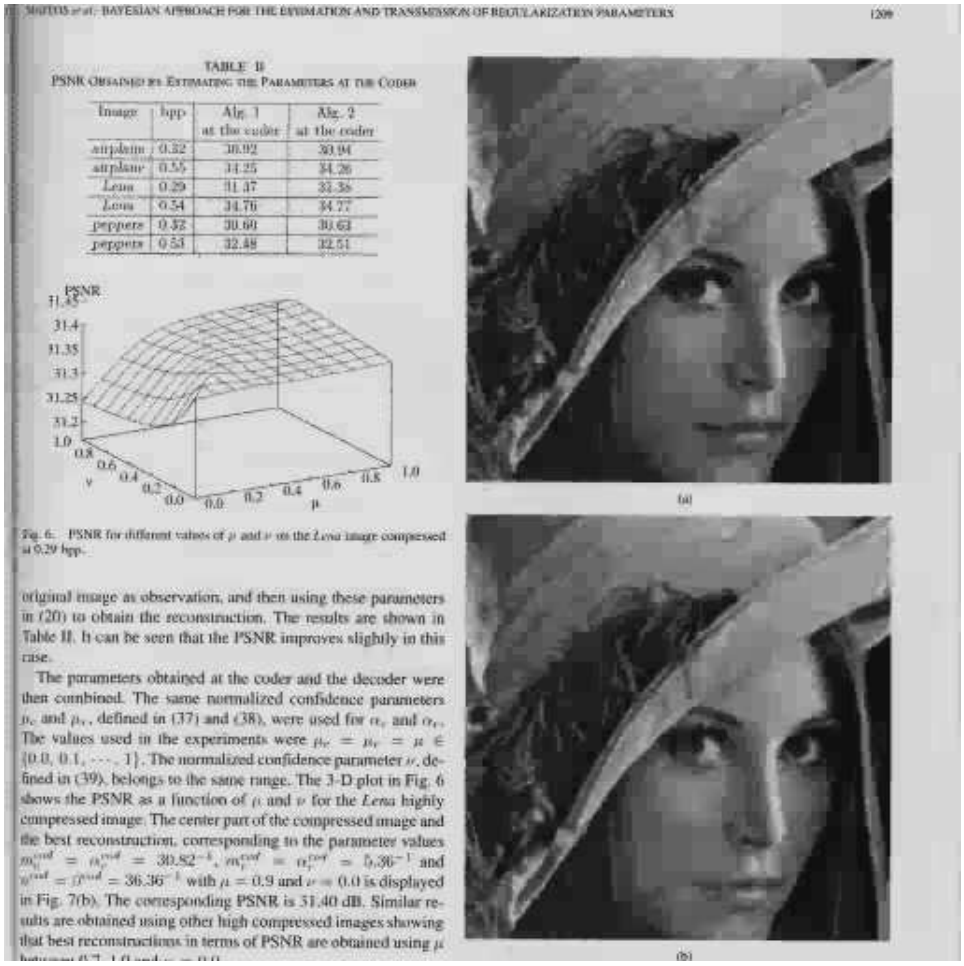
original image as observation, and then using these parameters in (20) to obtain the reconstruction. The results are shown in Table II. It can be seen that the PSNR improves slightly in this case.

The parameters obtained at the coder and the decoder were then combined. The same normalized confidence parameters μ_v and $\mu_{v^\perp}$ defined in (37) and (38), were used for α_v and $\alpha_{v^\perp}$. The values used in the experiments were $\mu_v = \mu_{v^\perp} = \mu \in \{0.0, 0.1, \cdots, 1\}$. 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 $m_v^{cod} = \alpha_v^{cod} = 30.82^{-1}$, $m_{v^\perp}^{cod} = \alpha_{v^\perp}^{cod} = 5.36^{-1}$ and $\kappa^{cod} = \beta^{cod} = 36.36^{-1}$ 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 5.39: PP1209 comprimida pelo SM-MMP a taxa de 0.5 bpp. PSNR=30.69dB



MATTUS *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	32.36
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_{c^*}, defined in (37) and (38), were used for α_c and α_{c^*}. The values used in the experiments were $\mu_c = \mu_{c^*} = \mu \in \{0.0, 0.1, \cdots, 1\}$. 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 $m_0^{conf} = \alpha_c^{conf} = 30.82^{-1}$, $m_1^{conf} = \alpha_{c^*}^{conf} = 5.36^{-1}$ and $\gamma^{conf} = \beta^{conf} = 36.36^{-1}$ 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 5.40: PP1209 comprimida pelo GSM-MMP a taxa de 0.5 bpp. PSNR=30.34dB

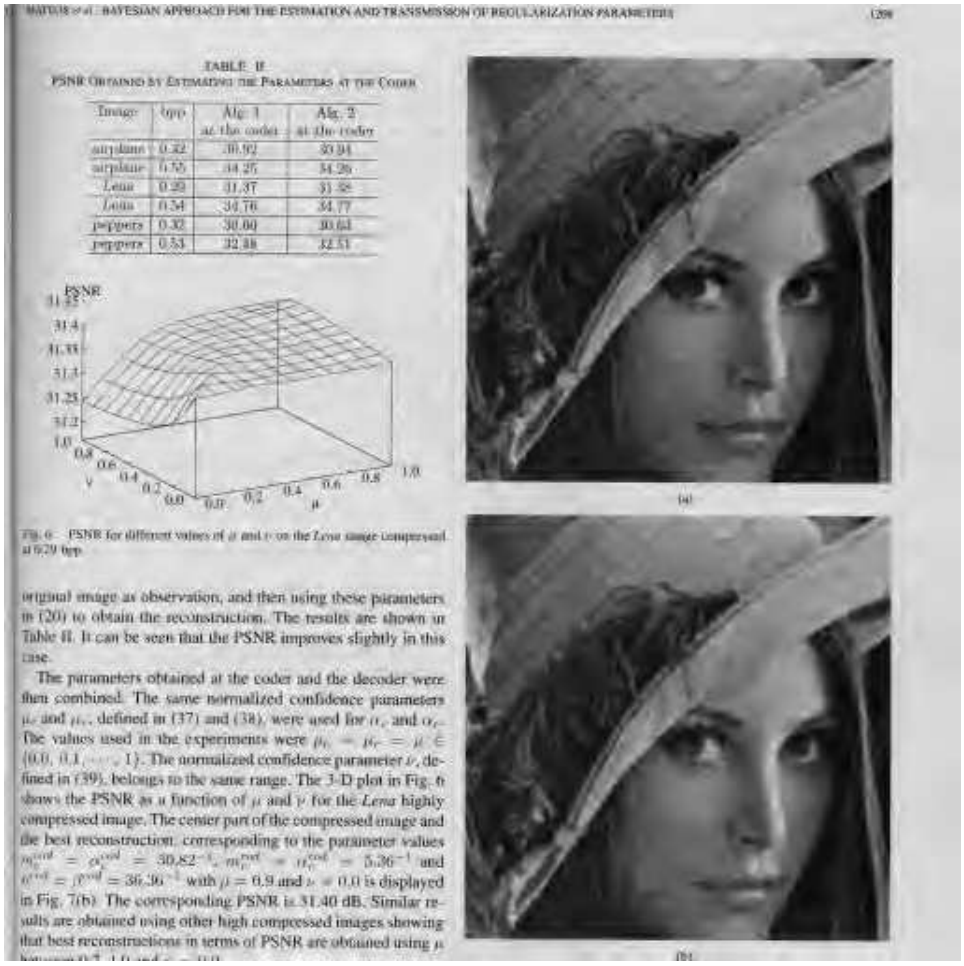


TABLE II
PSNR OBTAINED BY ESTIMATING THE PARAMETERS AT THE CODER

Image	bpp	Alg. 1 at the coder	Alg. 2 at the coder
airplane	0.32	30.92	30.94
airplane	0.52	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_{c^*}, defined in (37) and (38), were used for α_c and α_{c^*}. The values used in the experiments were $\mu_c = \mu_{c^*} = \mu \in \{0.0, 0.1, \cdots, 1\}$. 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 $\alpha_c^{cod} = \alpha_{c^*}^{cod} = 30.82^{-1}$, $\alpha_{c^*}^{cod} = \alpha_{c^*}^{cod} = 5.36^{-1}$ and $\alpha^{cod} = \beta^{cod} = 36.36^{-1}$ 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 5.41: PP1209 comprimida pelo SPIHT a taxa de 0.5 bpp. PSNR=28.73dB

O desempenho do algoritmo GSM-MMP pode ser visto na tabela abaixo. Nesta tabela, tem-se como referencial, a $PSNR$ para a taxa de 0.5 bpp. Os maiores ganhos do algoritmo GSM-MMP foram obtidos para as imagens *gold* e *pp1209*.

Tabela 5.1: Comparação dos resultados (em dB)

Imagem	MMP	MMP-RDI	MMP-U	SM-MMP	GSM-MMP	SPIHT
lena	34,25	34,88	35,20	36,13	35,67	37,22
aerial	27,17	27,89	28,16	28,56	28,39	28,61
gold	31,39	32,01	32,25	32,56	32,73	33,27
barbara	28,61	29,06	29,58	29,62	29,51	31,72
f16	35,13	35,39	36,20	37,21	36,78	37,39
pp1205	27,45	28,50	28,58	28,54	28,41	25,21
pp1209	29,08	30,01	30,13	30,69	30,34	28,73

Através dos resultados mostrados pela tabela 5.1, vale a pena utilizar o método de casamento lateral com os blocos para o esquema de compressão de sinais MMP. Sendo assim, existe uma boa indicação para se continuar a investigação deste método.

Capítulo 6

Considerações finais

Neste trabalho de dissertação, foi desenvolvido um algoritmo que utiliza a estrutura do algoritmo MMP, adequando-o ao modelo do casamento lateral generalizado, onde tal algoritmo usa a vizinhança do bloco que será codificado para fazer uma predição eficiente do mesmo, que no caso desta implementação a previsão é feita utilizando as características de três blocos vizinhos. Esta adequação trouxe mudanças na composição do algoritmo MMP, tais mudanças ocorreram na regra de construção do dicionário inicial, otimização, transformação de escala, codificação e decodificação.

O algoritmo desenvolvido é chamado de GSM-MMP, *Generalized Side-Match Multidimensional Multiscale Parser*, que mostrou resultados bastante motivadores em relação às imagens suaves e também nas imagens mistas, que são compostas por texto, gráficos e imagens. Comparando com os resultados obtidos pelos algoritmos MMP original, MMP-U, SM-MMP, JPEG e SPIHT, apresentando assim um desempenho aproximado e até melhor em alguns casos.

Para a imagem gold, que é uma imagem suave, o algoritmo GSM-MMP obteve um desempenho bem próximo comparado com o resultado do algoritmo SPHIT, o qual possui a premissa de que as imagens a serem processadas por ele sejam imagens suaves, imagens estas que possuem uma alta concentração de energia em freqüências mais baixas, no caso do algoritmo GSM-MMP isto não ocorre pois o mesmo não faz nenhuma predição do sinal de entrada. Também, obteve um ganho positivo em 0.12 dB comparado com o algoritmo SM-MMP, e nas outras imagens,

o GSM-MMP teve uma ótima aproximação positiva.

É perceptível e incômodo o efeito de blocagem que existe no MMP original e, com o algoritmo GSM-MMP, houve uma sensível melhoria em nível de qualidade subjetiva em todas as imagens, devido ao fato que o GSM-MMP realiza uma predição para a estruturação do dicionário de codificação baseada em semelhança com os três vizinhos do bloco atual que será codificado. Assim, para que a codificação seja realizada de maneira satisfatória, o dicionário utilizado é construído de modo que contenha apenas os elementos semelhantes aos três vizinhos do bloco atual, resultando na diminuição do efeito de blocagem nas imagens processadas.

O algoritmo SM-MMP foi o pioneiro em melhorar a codificação de imagens suaves através das informações de blocos vizinhos, utilizando a estrutura do algoritmo MMP, onde o mesmo utiliza dois blocos vizinhos para obter essas informações. O algoritmo concebido neste trabalho teve como proposta, almejar a melhoria do desempenho das imagens do MMP padrão utilizando os conceitos do *Side-Match*, bem como aperfeiçoar o mesmo. Como pode-se observar pelos resultados obtidos através do GSM-MMP, as metas foram alcançadas em parte, só não foi por completo pelo fato de que o algoritmo GSM-MMP não obteve um ganho positivo suficiente para superar o algoritmo SM-MMP em todas as imagens, mas isso se dá pelo fato que nem todos os blocos são processados pelo *Three-sided*, parte dos blocos são processados pelo MMP original que são os blocos chamados de pré codificados. Com isto, a porcentagem de blocos codificados com o *Three-sided* não conseguiu resultar um ganho suficiente para superar a perda sofrida pelos blocos codificados com o MMP padrão. Mas a solução deste problema é resolvido com a incorporação de uma super-atualização no dicionário, pois logo no início, o dicionário é bem pequeno, produzindo uma codificação pobre para os primeiros blocos. Além disso, pode ser desenvolvido um dicionário de vizinhança utilizando os blocos numa área ao redor do bloco atual para a codificação. Pode-se implementar classificadores de borda no dicionário principal e no dicionário de estado com o objetivo de buscar de forma mais rápida o elemento adequado.

Percebe-se que melhorias podem ser feitas para aumentar ainda mais o potencial do algoritmo GSM-MMP garantido resultados mais plausíveis e aumen-

tando o número de ferramentas disponíveis para tal algoritmo, ficando aqui como sugestões para trabalhos futuros que com certeza resultará em contribuições significativas para o desenvolvimento de pesquisas na área de compressão de dados.

Referências Bibliográficas

[1] COVER, T. M., Elements of Information Theory. 2 ed., 1991.

[2] SHANNON, C. E., "A Mathematical Theory of Communication", Bell Syst. Tech. Journal, v. 27, 1948.

[3] Processamento Digital de Imagens. Addison-Wesley Publishing Company, 2000.

[4] PAPOULIS, A., Probability Rondom Variables, and Stochastic Processes. 2 ed. McGraw-Hill, 1984.

[5] SAYOOD, K., Introduction to Data Compression. 2 ed. San Francisco, Morgan Kaufmann Publishers, 2000.

[6] PENNENBAKER, W. B., MITCHELL, J. L., JPEG Still Image Data Compression Standard. Van Nostrand Reinhold, 1994.

[7] MITCHELL, J. L., PENNEBAKER, W. B., FOGG, C. E., et al., MPEG Video Compression Standard. Kluwer Academic Publishers, 2001.

[8] HUFFMAN, D. A., "A Method for the Construction of Minimum Redundancy Codes". In: Proceedings of the IRE, v. 40, pp. 1098–1101, 1951.

[9] ABRAMSON, N., Information Theory and Coding. New York, McGraw-Hill, 1963.

[10] ZIV, J., LEMPEL, A., "A Compression of Individual Sequences Via Variable-Rate Coding". In: IEEE Trans. Inf. Theory, v. 24, pp. 530–536, 1978.

[11] ZIV, J., LEMPEL, A., "A Universal Algorithm for Data Compression". In: IEEE Trans. Inf. Theory, v. 23, pp. 337–343, 1977.

[12] C.CHRISTOPOULOS, A. S., EBRAHIMI, T., "The JPEG2000 Still Image Coding System: An Overview", IEEE Transactions on Consumer Electronics, v. 46, n. 2, pp. 1103–1127, November 2000.

[13] GERSHO, A., "Quantization", IEEE Comm. Mag., v. 15, pp. 16–29, 1977.

[14] MAX, J., "Quantizing for Minimum Distortion", IRE Trans. Inf. Theory, v. 6, pp. 7–12, 1960.

[15] GERSHO, A., CUPERMAN, V., "A Pattern Matching Technique for Speech Coding", IEEE Communication Magazine, v. 21, pp. 15–21, 1983.

[16] GRAY, R. M., "Vector Quantization". In: IEEE Transaction on Acoustics, Speech and Signal Processing, v. 1, pp. 4–29, 1984.

[17] MAKHOUL, J., ROUCOS, S., GISH, H., "Vector Quantization in Speech Coding". In: Proc. IEEE, n. 73, pp. 1551–1588, 1985.

[18] WITTEN, I., NEAL, R., CLEARY, J. G., "Arithmetic Coding for Data Compression", Comm. ACM, v. 30, pp. 520–540, June 1987.

[19] CLARKE, R. J., Digital Compression of Still Images and Video. Academic Press, 1995.

[20] The Theory of Error Correcting Codes. New York, North-Holland, 1977.

[21] TOPIWALA, P. N., Wavelet Image and Video Compression. 1 ed. Norwell, Kluwer Academic Publishers, 1998.

[22] VETTERLI, M., KOVACEVIC, J., Wavelets and Subband Coding. Prentice Hall, 1995.

[23] STRANG, G., NGUYEN, T., Wavelets and Filter Banks. 1 ed. Wellesley-Cambrige Press, 1996.

[24] FRENDENDALL, G. L., BEHREND, W. L., "Picture Quality - Procedures for Evaluating Subjetive Effects of Interference", IRE, v. 48, pp. 1030–1034, 1970.

[25] JAIN, A. K., Fundamentals of Digital Image Processing. Prentice Hall, 1989.

[26] CARVALHO, M. B., Compression of Multidimensional Signals based on Recurrent Multiscale Patterns. Tese de d.sc., PEE/Coppe/UFRJ, Rio de Janeiro, RJ, Março 2001.

[27] CARVALHO, M. B., SILVA, E. B., FINAMORE, W. A., "Multidimensional Signals Compression Using Recurrent Patterns". In: Elsevier, pp. 1559–1580, 1995.

[28] E.B.L.FILHO, M.B.CARVALHO, SILVA, E. D., "Multidimensional signal compression using multi-scale recurrent patterns whith smooth side-match criterion", IEEE International Conference on Image Processing, , October 2004.

[29] FILHO, E. B. L., Compressão de Imagens Utilizando Recorrência de Padrões Multiescala com Critério de Continuidade Inter-blocos. Tese de m.sc., PEE/Coppe/UFRJ, Rio de Janeiro, RJ, Março 2004.

[30] DENN, M. M., Optimization by Variational Methods. 1 ed. New York, McGraw-Hill, 1969.

[31] EVERETT, H., "Generalized Lagrange Multiple Method For Solve Problems of Optimum Allocation Resouces". In: Operation Research, v. 11, pp. 399–417, 1963.

[32] WU, S. W., GERSHO, A., "Rate-Constrained Optimal Block-Adaptative Codging for Digital Tape Recording of HDTV". In: IEEE Trans. on Circuits and System for Video Tech., pp. 100–112, 1991.

[33] SULLIVAN, G. J., BAKER, B. L., "Efficient Quadtree Coding of Imagens and Video". In: IEEE Trans. on Image Processing, v. 3, pp. 327–331, 1994.

[34] KIANG, S. Z., BAKER, R. L., SULLIVAN, G. J., et al., "Recursive Optimal Prune With Applications to Tree Structured Vector Quantizers". In: IEEE Trans. on Image Processing, v. 1, pp. 162–169, 1992.

[35] JúNIOR, W. S., Compressão de Imagens Utilizando Recorrência de Padrões Multiescala com Segmentação Flexível. Tese de m.sc., PEE/Coppe/UFRJ, Rio de Janeiro,RJ, Dezembro 2004.

[36] SHUKLA, DRAGOTTI, MINH, et al., "Rate-Distortion Optimized Tree Structured Compression Algorithms for Piecewise Smooth Images". In: IEEE Transaction Image Processing, v. 14, March 2005.

[37] SAID, A., PEARLMAN, W., "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees". In: IEEE Transactions on Circuits and Systems for Video Technology, v. 6, Chicago, Illinois, June 1996.

[38] TAUBMAN, D., MARCELLIN, M., JPEG2000: Image Compressin Fundamentals, Standards and Practice. Kluwer Academic Publishers, 2001.

[39] H.C.WEI, P.C.TSAI, WANG, J., "Three-Sided Side Match Finite-State Vector Quantization", IEEE Transactions on Circuits and Systems for Video Technology, v. 10, n. 1, February 2000.

[40] T.KIM, "Side Match and Overlap Match Vector Quantizers for Images", IEEE Transactions on Image Processing, v. 1, n. 2, pp. 170–185, February 1992.

[41] S.B.YANG, L.Y.TSENG, "Smooth Side Match Vector Classified Vector Quantizer with Variable Block Size", IEEE Transactions on Image Processing, v. 10, n. 5, pp. 677–685, May 2001.

[42] T.S.CHEN, CHANG, C., "A New Image Coding Algorithm Using Variable-Rate Side-Match Finite-State Vector Quantization", IEEE Transactions on Image Processing, v. 6, n. 8, pp. 1185–1187, August 1997.

Apêndice A

Imagens Originais

Neste apêndice são mostradas as imagens originais que foram utilizadas nas simulações dos testes desta dissertação, são elas: *lena, aerial, gold, barbara, f16, pp1205* e *pp1209* .

A.1 Imagem *Lena*

Figura A.1: Imagem Lena original

A.2 Imagem *aerial*

Figura A.2: Imagem Aerial original

A.3 Imagem *gold*

Figura A.3: Imagem Gold original

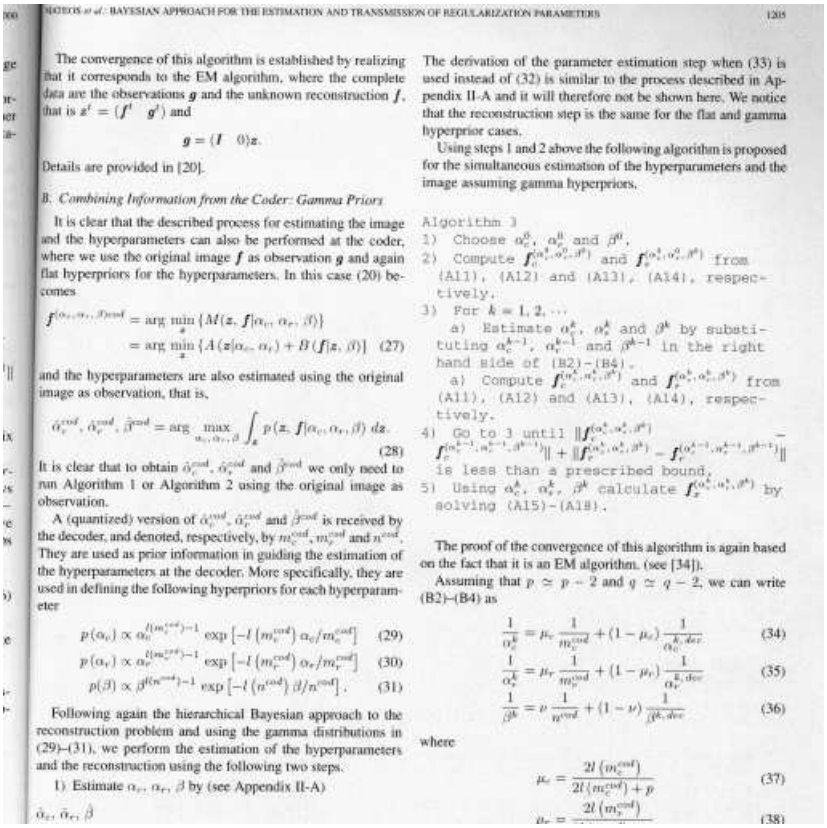
A.4 Imagem *Barbara*

Figura A.4: Imagem Barbara original

A.5 Imagem *f16*

Figura A.5: Imagem F16 original

A.6 Imagem *pp1205*



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

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

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

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

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

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

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

A (quantized) version of $\hat{\alpha}_\varepsilon^{cod}$, $\hat{\alpha}_r^{cod}$ 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

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

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

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

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)

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

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 $\boldsymbol{f}_\varepsilon^{(\alpha_\varepsilon^0,\alpha_r^0,\beta^0)}$ and $\boldsymbol{f}_r^{(\alpha_\varepsilon^0,\alpha_r^0,\beta^0)}$ 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 $\boldsymbol{f}_\varepsilon^{(\alpha_\varepsilon^k,\alpha_r^k,\beta^k)}$ and $\boldsymbol{f}_r^{(\alpha_\varepsilon^k,\alpha_r^k,\beta^k)}$ from (A11), (A12) and (A13), (A14), respectively.
4) Go to 3 until $\|\boldsymbol{f}_\varepsilon^{(\alpha_\varepsilon^k,\alpha_r^k,\beta^k)} - \boldsymbol{f}_\varepsilon^{(\alpha_\varepsilon^{k-1},\alpha_r^{k-1},\beta^{k-1})}\| + \|\boldsymbol{f}_r^{(\alpha_\varepsilon^k,\alpha_r^k,\beta^k)} - \boldsymbol{f}_r^{(\alpha_\varepsilon^{k-1},\alpha_r^{k-1},\beta^{k-1})}\|$ is less than a prescribed bound.
5) Using α_ε^k, α_r^k, β^k calculate $\boldsymbol{f}_x^{(\alpha_\varepsilon^k,\alpha_r^k,\beta^k)}$ 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

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

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

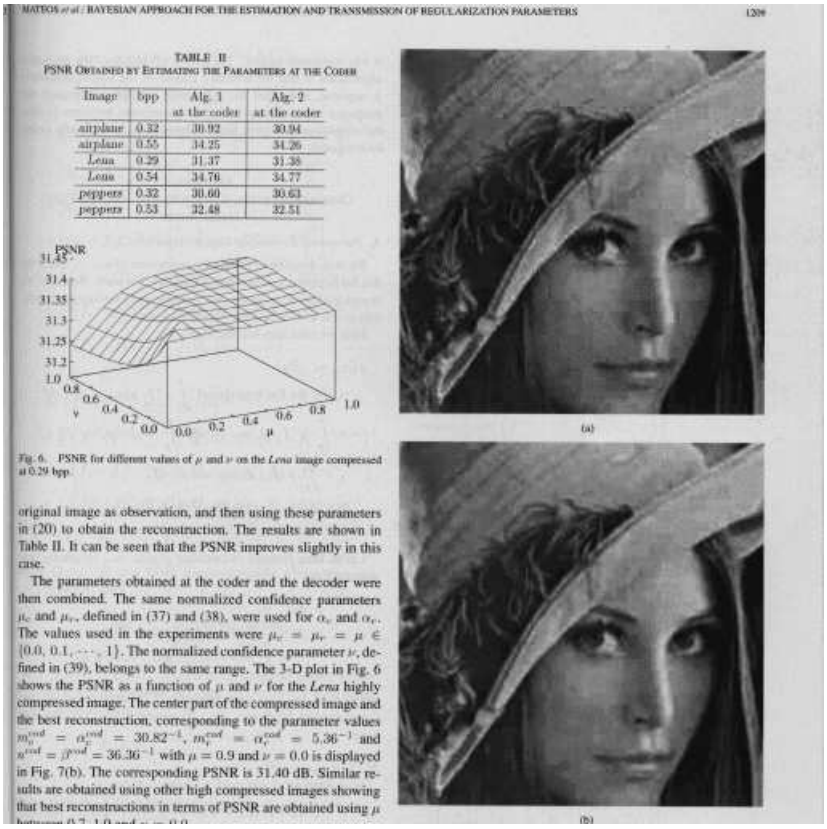
$$\frac{1}{\beta^k} = \nu \frac{1}{n^{cod}} + (1 - \nu)\frac{1}{\beta^{k,dec}} \quad (36)$$

where

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

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

A.7 Imagem *pp1209*



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 $m_v^{cod} = \alpha_c^{cod} = 30.82^{-1}$, $m_i^{cod} = \alpha_r^{cod} = 5.36^{-1}$ and $\kappa^{cod} = \beta^{cod} = 36.36^{-1}$ 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 A.7: Imagem PP1209 original