ESTUDO DE TÉCNICAS DE COMPRESSÃO COM PERDAS PARA USO DIDÁTICO

Emmanuel Sávio Silva FREIRE; Thiago da Silva LEITE; Cleiton Mulato UCHOA; Maria Heveline Vieira DUARTE

(1) Centro Federal de Educação Tecnológica – CEFETCE, Av. Treze de Maio, 2081 - Benfica - Fortaleza/CE CEP: 60040-531 Telefone: (85)3307-3607 e-mail: savio.essf@gmail.com

(2) Centro Federal de Educação Tecnológica – CEFETCE, e-mail: thiagoiguatu@gmail.com
(3) Centro Federal de Educação Tecnológica – CEFETCE, e-mail: cleitonrpg@yahoo.com.br
(4) Centro Federal de Educação Tecnológica – CEFETCE, e-mail: heveline@cefetce.br

RESUMO

Com o avanço tecnológico e o acesso globalizado às novas tecnologias, a demanda por todo tipo de informação é crescente. Isto requer maiores espaços para o armazenamento e bandas mais largas para a transmissão dessas informações, porém estes recursos são (e sempre serão) limitados, apesar de todo o desenvolvimento na área. Uma solução para este problema é o desenvolvimento de algoritmos de compressão cada vez mais eficientes, que reduzam o tamanho dos arquivos sem que haja perda de qualidade perceptível ao sistema visual humano. Ao longo dos últimos anos, várias técnicas de compressão de imagem e vídeo têm sido propostas e algumas alcançaram resultados efetivos de taxa e distorção. No entanto, estas técnicas são bastante complexas e a literatura existente que as descrevem não são satisfatórias no sentido de facilitar o aprendizado e o entendimento do seu funcionamento. Este artigo apresenta uma compilação bibliográfica contendo os principais algoritmos de compressão existentes: JPEG, SPIHT e MMP, descrevendo estas técnicas de forma didática, juntamente com um estudo comparativo entre elas. A metodologia implica na revisão bibliográfica e na elaboração de um material que possa ser usado como ferramenta em aulas sobre o assunto nos cursos de Telemática e Engenharias do CEFETCE.

Palavras-chave: técnicas de compressão de imagens, compressão com perdas, uso didático.

1. INTRODUÇÃO

Devido ao avanço tecnológico, a demanda por informações é cada vez maior, isso requer maiores larguras de banda e maiores espaços para o envio e o armazenamento de dados. Para resolver o problema do armazenamento, técnicas de compressão foram desenvolvidas e aprimoradas para garantir a diminuição do tamanho dos arquivos sem que essa perda fosse perceptível para o sistema visual humano. A compressão de dados baseia-se na redundância da informação e pode ser divida em dois grupos: compressão sem perdas, que preserva o conteúdo original, e compressão com perdas, que não preserva as informações em sua totalidade. As técnicas baseadas em compressão sem perdas não apresentam ganho significativo de espaço para o armazenamento, diferentemente das técnicas que são baseadas em compressão com perdas. Dentre as várias técnicas existentes, foram escolhidas as principais (JPEG, SPIHT e MMP) para um estudo bibliográfico, no qual serão descritas cada uma das técnicas, mostrando seus fluxogramas para a implementação e onde cada uma pode ser mais bem aplicada.

2. REVISÃO BIBLIOGRÁFICA

As técnicas de compressão se baseiam em uma premissa básica, remoção de informação redundante, para que possam diminuir a quantidade de bytes necessários para transmitir ou armazenar dados (MARQUES, VIEIRA, 1999). Algumas dessas técnicas foram criadas para comprimir imagens naturais coloridas ou monocromáticas, como é o caso do JPEG. Através da divisão da imagem em pequenos blocos e utilizando a transformada discreta de cossenos (DCT), consegue-se retirar partes da informação que não são perceptíveis ao sistema visual humano, reduzir o custo computacional e obter altas taxas de compressão (SILVA, 1998). O SPIHT foi uma técnica criada para aperfeiçoar a transmissão progressiva, bem como para compressão, e ele tem duas características importantes: em qualquer ponto durante a decodificação de uma

imagem, a qualidade da imagem apresentada é a melhor que pode ser atingida para o número de bits de entrada do decodificador até esse momento; a outra é o seu uso de codificação embutida, no qual um codificador (com codificação embutida) produz dois arquivos, um grande de tamanho M e um pequeno de tamanho m, então o arquivo menor é idêntico aos primeiros m bits do maior arquivo (SALOMON, 2004). O MMP, por exemplo, utiliza-se das recorrências de padrões existentes nas imagens para comprimi-las, utilizando-se de um dicionário criado a partir das características da imagem (DUARTE, 2002).

3. METODOLOGIA

Uma revisão bibliográfica foi feita na literatura existente para cada técnica de compressão que será relatada nesse trabalho. Através de uma compilação bibliográfica, foi feito um estudo das técnicas apresentadas, destacando as suas principais características, seus princípios de funcionamento e para qual aplicação cada uma destas possui um melhor aproveitamento quanto à taxa de compressão e ao tipo de mídia. Após a revisão bibliográfica fora feita uma simplificação na descrição dos processos dos algoritmos usados nas técnicas apresentadas visando facilitar o entendimento de tais processos por parte daqueles que estivessem interessados. A idéia principal a ser usada é a descrição com auxílio de imagens e textos curtos e simples, que ajudem qualquer indivíduo a ver o processo tanto em suas partes como um todo, e, assim, possa entender como os algoritmos dessas técnicas trabalham em conjunto para obter seu resultado de compressão final.

4. TÉCNICAS DE COMPRESSÃO

4.1. JPEG

4.1.1. INTRODUÇÃO

O JPEG, sigla de *Joint Photographic Experts Group*, é um padrão estabelecido em 1991 para comprimir imagens naturais coloridas ou monocromáticas com até 65536 x 65536 pixels. É uma técnica fundamentalmente de compressão com perdas, mesmo existindo implementação para compressão sem perdas (SALOMON, 2004).

Possui quatro modos de operação:

- Seqüencial: através de uma única varredura, que é feita da esquerda para a direita e de cima para baixo, a imagem é codificada;
- Progressiva: a imagem é codificada em várias varreduras, aumentando a qualidade e a resolução a cada nova varredura;
- Hierárquica: a imagem é codificada em múltiplas resoluções;
- Sem perdas.

4.1.2. DESCRIÇÃO DO JPEG – MODO SEQÜENCIAL

A codificação de uma imagem é dividida em uma seqüencia de operações:

- Divisão da imagem em blocos 8x8;
- Cálculo dos coeficientes da DCT (Transformada Discreta dos Cossenos);
- Quantização;
- Reordenação dos coeficientes em zig-zag;
- Codificação baseada em entropia (MARQUES, VIEIRA, 1999).

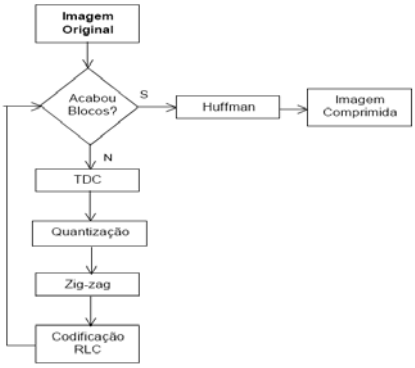

Figura 1 - Fluxograma do algoritmo do JPEG.

4.1.2.1. DIVISÃO DA IMAGEM EM BLOCOS 8X8

Faz-se necessária a divisão da imagem para que se possa ter uma diminuição do custo computacional. Antes de iniciar a divisão, deve verificar se o número de linhas e de colunas é múltiplos de 8, caso não seja, deve-se acrescentar linhas e colunas com os valores de seus pixels iguais a zero para que a imagem cumpra esse requisito.

4.1.2.2. CÁLCULO DOS COEFICIENTES DA DCT

A transformada é a chave para o processo de compressão, pois ela toma um conjunto de pontos no domínio espacial e os transforma em uma representação equivalente no domínio da freqüência. Um dos objetivos da transformada é diminuir a correlação, eliminando as redundâncias estatísticas. Outro aspecto relevante é que os novos dados devem exigir menos espaço ao mesmo tempo em que o algoritmo para obtê-los seja o mais eficiente possível (SILVA, 1998).

A transformada utilizada é a DCT, que converte um bloco de pixels em uma matriz de coeficientes, descorrelacionando a informação da imagem. Seu funcionamento é baseado na equação abaixo (GONZALEZ, WOODS, 1992):

$$F(u, v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7} \quad \sum_{y=0}^{7} f(x,y)cos\frac{\pi u(2x+1)}{16}\cos\frac{\pi v(2y+1)}{16}$$

$C(u), C(v) = \frac{1}{\sqrt{2}}$, para u, v = 0,

C(u), C(v) = 0, para u, v > 0.

Onde:

F(u, v) representa o pixel resultante da transformada,

f(x, y) representa o pixel da imagem de entrada.

4.1.2.3. QUANTIZAÇÃO

Devido aos coeficientes da DCT serem reais, eles ocuparão mais espaço que na matriz original, logo se deve fazer o processo da quantização, onde se perderá parte da informação, pois serão retirados a parte decimal de cada coeficiente, tornando inteiro. Utiliza-se a equação abaixo para criar a matriz de quantização:

$$Q[i, j] = 1+(1+i+j)*FQ$$

Onde:

Q[i, j] representa o valor do coeficiente que formará a matriz,

FQ é o fator de qualidade que pode variar de 1 a 25, sendo que valores maiores que 25 podem comprometer muito a qualidade da imagem (SALOMON, 2004).

Depois se divide a matriz resultante da DCT pela a de quantização, arredondando os valores para inteiro.

4.1.2.4. REORDENAÇÃO ZIG-ZAG

Devido ao truncamento feito no estágio anterior, muitos coeficientes são truncados para zero, utiliza-se a seqüência do zig-zag para que os coeficientes de

baixa freqüência sejam colocados na frente dos coeficientes de alta freqüência (SILVA, 1998).

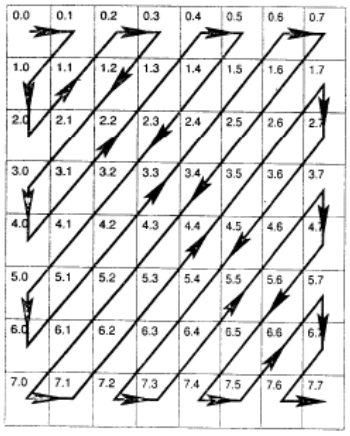

Figura 2 - Seqüencia Zig-zag.

4.1.2.5 CODIFICAÇÃO BASEADA EM ENTROPIA RUN-LENGTH CODING

A entropia de uma imagem é a medida de seu conteúdo de informação. Se entropia é alta, a informação de uma imagem tende a não ter muita correlação. Ou seja, a informação de uma imagem de entropia alta contém muita aleatoriedade e pouca redundância. Se a entropia é baixa, a informação de uma imagem é mais previsível, contém pequena aleatoriedade e sua redundância é alta.

Pode-se computar a entropia de uma imagem como a probabilidade de sua ocorrência. Isto é exibido como um número que representa o número de bits necessários para representar aquela probabilidade. Para qualquer imagem, isto seria como segue:

E = Nc x Nl x Nb

Onde:

E: entropia;

Nec: Número de colunas;

Nau: Número de linhas;

NBA: número de bits por pixel.

Por exemplo: 640 x 480 x 8, teria a seguinte entropia:

Entropia = 640 x 480 x 8 = 2.457.600.

O esquema do Run-Length Coding (RLC) trabalha como segue: a imagem original é avaliada começando do primeiro pixel no canto superior esquerdo, olhando o primeiro pixel e é seguido o vizinho pela linha, o esquema determina quantos pixels seguintes tem o mesmo valor de nível. Se o próximo um ou mais pixels na seqüência tem o mesmo valor de nível como o primeiro, eles são todos representados por um código novo. O código novo é composto de dois valores; um valor de nível seguido pelo número de pixels (run-length) que tem o mesmo valor. O processo move para o próximo pixel na linha com um novo valor de nível se repete o processo. Quando o fim da linha é alcançado, o processo volta ao começo da próxima linha. Este processo continua até que a última linha da imagem tenha sido codificada (SILVA, 1998).

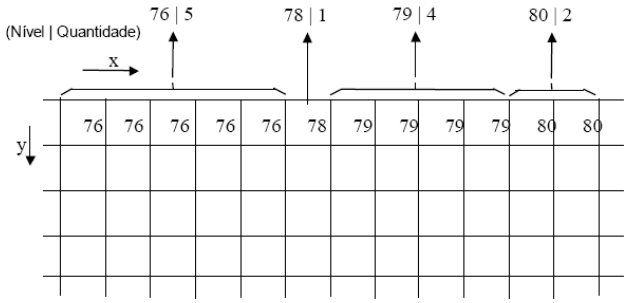

Figura 3 - Codificação de Run-Length Coding (RLC)

4.1.2.6. CODIFICAÇÃO POR HUFFMAN

Outra técnica comum para a codificação de entropia é a de Huffman, desenvolvida em 1952, por D. A. Huffman. A codificação de Huffman converte o valor do nível de pixel da imagem original para um novo código de tamanho variável, baseado nas freqüências de ocorrência na imagem. Desse modo, são atribuídos níveis que freqüentemente acontecem em códigos menores, e são atribuídos níveis que acontecem sem muita freqüência em códigos mais longos. O resultado é que a imagem comprimida exigirá menos bits globais para descrever a imagem original (SALOMON, 2004).

4.2. SPIHT

Parece que vários filtros wavelet produzem resultados diferentes, dependendo do tipo de imagem que tratam, mas atualmente não está claro qual o melhor filtro para um dado tipo de imagem.

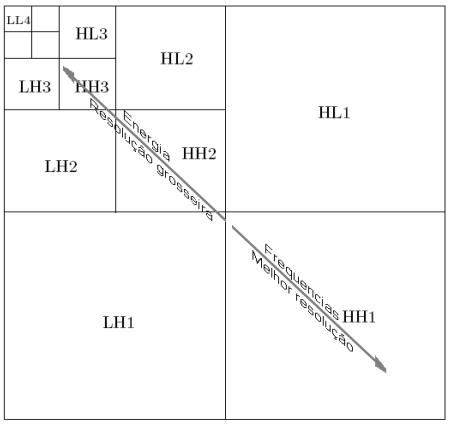

Figura 4 - Sub-bandas em níveis da decomposição Waveleet.

Independentemente do filtro particularmente utilizado, a imagem é decomposta em sub-bandas, de tal forma que as sub-bandas mais baixas correspondem às freqüências mais elevadas da imagem (eles são as highpass níveis) e as sub-bandas superiores que correspondem às freqüências mais baixas da imagem (lowpass níveis), onde a maior parte da energia da imagem está concentrada. Esta é a razão pela qual nós podemos esperar que o detalhamento dos coeficientes seja menor quando passamos de maiores níveis para menores. Além disso, existem semelhanças entre as sub-bandas espaciais. Uma parte da imagem, tanto quanto uma borda, ocupam a mesma posição espacial em cada sub-banda. (SALOMON, 2004).

Estas características da decomposição wavelet são exploradas pelo método SPIHT, portanto o SPIHT (set partitioning in hierarchical trees – particionamento em grupos de árvores hierárquicas) é um codificador de compressão de imagens baseado em wavelet.

Primeiro converte-se uma imagem em sua Transformada Wavelet e, em seguida, transmite-se informações sobre os coeficientes wavelet resultantes desta. Na decodificação o decodificador usa a informação recebida para reconstruir a wavelet e executa uma Transformada Wavelet Inversa para recuperar a imagem.

SPIHT e seu predecessor (o Embedded Zerotree Wavelet coder, ou EZW) foram significativos avanços na compressão de imagem na qual eles ofereceram significativamente melhoras qualitativas sobre o vetor de quantização, JPEG, e wavelets combinadas com quantização, ao mesmo tempo em que não exijam formação e produção de um bit-plane (plano de bits) ordenados embutido. SPIHT exibe características excepcionais ao longo de várias propriedades de uma só vez. SPIHT é um método de codificação e decodificação da Transformada Wavelet de uma imagem.

Através de codificação e transmissão de informações sobre os coeficientes wavelet, é possível para um decodificador realizar uma transformação inversa sobre a wavelet e reconstruir a imagem original. A totalidade do coeficiente wavelet não precisa ser transmitida a fim de recuperar a imagem. Em vez disso, quando o descodificador receber mais informações sobre a Transformada Wavelet original, a transformação inversa terá um rendimento melhor na qualidade de reconstrução (ou seja, maior proporção no sinal de pico sobre ruído) da imagem original. SPIHT gera uma qualidade de imagem e desempenho excelentes devido a várias propriedades do algoritmo de codificação. Eles são parcialmente ordenados por valor de coeficiente, tirando vantagem das redundâncias entre as escalas wavelets e o plano de bits a ser transmitidos. Segundo uma transformação wavelet, SPIHT divide a wavelet em árvores de orientação espacial. SPIHT codifica uma wavelet para transmitir informação sobre a significância de um pixel. Declara-se se um pixel está ou não acima de determinados limiares, a informação sobre o valor daquele pixel fica implícita. Além disso, SPIHT transmite informações indicando se um pixel ou qualquer um de seus descendentes estão acima de um limiar. Se a declaração provar-se falsa, então todos os seus descendentes são conhecidos por estarem abaixo desse limite e eles não precisam ser considerados durante o resto da passagem corrente. No final de cada passagem, o limiar é dividido por dois e o algoritmo continua. Ao proceder desta forma, as informações sobre os bits mais significativos dos coeficientes wavelet irão sempre preceder informações sobre as formas menos significativa de bits, que é designado por um plano de bits ordenados. Dentro de cada plano de bit dados são transmitidos em três listas: a lista de pixels insignificantes (LIP), a lista dos conjuntos insignificantes (LIS), e a lista de pixels significativos (LSP).

SPIHT foi concebido para otimizar a transmissão progressiva, bem como para compressão. Uma das características importantes do SPIHT (talvez a única característica) é que em qualquer ponto durante a decodificação de uma imagem, a qualidade da imagem apresentada é o melhor que pode ser atingida para o número de bits de entrada do decodificador até esse momento (de forma resumida, quanto mais a quantidade de amostras da informação chegam ao decodificador, melhor fica a imagem, é progressivo...). Outra importante característica do SPIHT é seu uso de codificação embutida. Esta característica é definida da seguinte forma: Se um (com codificação embutida) codificador produz dois arquivos, um grande de tamanho M e um pequeno de tamanho m, então o arquivo menor é idêntico aos primeiros m bits do maior arquivo.

O exemplo a seguir ilustra acertadamente o significado desta definição. Suponha que você tenha que esperar três usuários para que você possa enviar-lhes certa imagem comprimida, mas eles precisam de imagens de diferentes qualidades. O primeiro deles precisa da qualidade contida em um arquivo de 10 Kb. A qualidade de imagem exigida pelo segundo e terceiro usuários estão contidos em ficheiros de tamanhos 20 Kb e 50 Kb, respectivamente. A maioria dos métodos de compressão de imagens com perdas teria que comprimir a mesma imagem três vezes, em diferentes qualidades, para gerar arquivos com o direito três tamanhos. SPIHT, por outro lado, produz um arquivo, e, em seguida, três pedaços de comprimentos de 10 Kb, 20 Kb, e de 50 Kb, todos começados no início do arquivo - pode ser enviada para os três usuários, satisfazendo assim as suas necessidades.

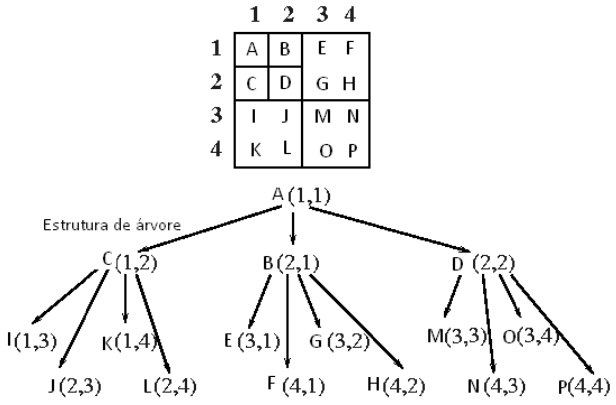

Figura 5 - Exemplo de estrutura de árvore

O PSNR (Peak Signal Noise Ratio, o método objetivo de avaliação de qualidade de vídeos e imagens mais aceito na comunidade científica hoje) é ligeiramente melhor alcançado por meio dessa dinâmica ordenação dos coeficientes wavelet (SPRLJAN, GRGIC e GRGIC, 2005).

4.3. MMP

4.3.1. INTRODUÇÃO

O MMP, sigla de *Multidimenional Multiscale Parser*, é um algoritmo de compressão de imagem com perdas baseado em casamento aproximado de padrões recorrentes com escala. Possui um dicionário inicial usado para representar os dados de entrada, esse dicionário é atualizado durante codificação de modo contenha elementos mais o mais próximo dos blocos da imagem. Uma das principais vantagens do algoritmo, é que o dicionário criado pelo codificar, também pode ser criado pelo decodificador, não havendo necessidade de transmiti-lo, aumentando assim a performance do algoritmo. O algoritmo possui basicamente cinco etapas: Criação do dicionário, cálculo dos custos, obtenção da árvore ótima, codificação dos *flags* e índices e atualização do dicionário (DUARTE, 2002).

4.3.2. DESCRIÇÃO DO MMP

Em primeiro lugar, a imagem é dividida em blocos NxN. Antes de começar o processamento, o MMP monta um dicionário com elementos inicias, blocos de matrizes, com valores baseados nos valores de luminância dos pixels da imagem. Esses valores dos pixels dos elementos iniciais do dicionário serão distribuídos de *passo* em *passo,* de acordo com o menor valor de luminância (*minpixel*) e o maior valor de luminância (*maxpixel)* da imagem, assim como o número de elementos inicial do dicionário. O *passo* é dado pela diferença entre o *maxpixel* e o *minpixel*, dividido pelo número de elementos iniciais do bloco. Para blocos de 8x8, por exemplo, com valores com valores de luminância variando de 0 a 255 teríamos 64 elementos inicias no bloco; neste caso, teríamos o *passo* = 4, ou seja, os valores dos elementos do nível mais alto do dicionário variando de 4 em 4 (DUARTE, 2002).

O número de níveis do dicionário está relacionado com as dimensões do bloco bem como o número de divisões possíveis que o bloco pode ter de acordo com uma árvore binária. A divisão do bloco pode ser horizontal ou vertical. Cada nível abaixo contém blocos com metade de pixel do bloco do nível mais alto até que no último nível os blocos contenham apenas um valor de pixel. Cada elemento do dicionário é representado da forma E[i][j], onde 'i' indica o nível do elemento e 'j' o índice em que ele se encontra.

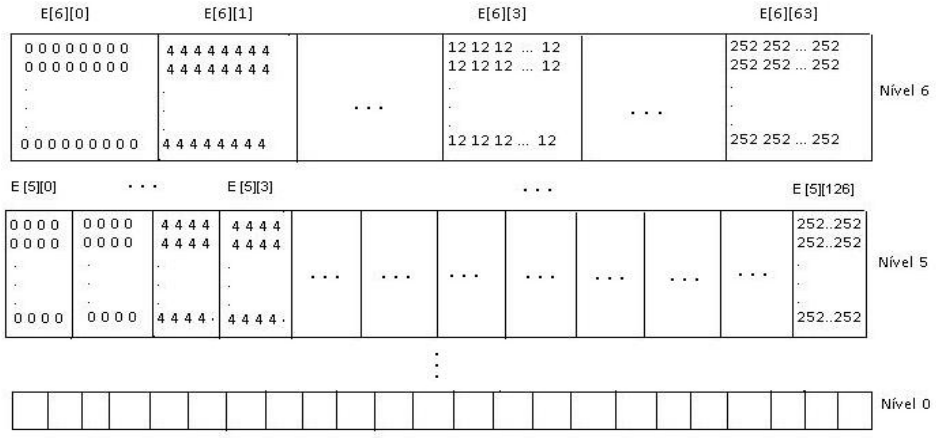

Figura 6 - Estrutura geral de um dicionário

Cada bloco é processado seqüencialmente da esquerda para a direita e à medida que vão sendo processados, o algoritmo tente representar os blocos seguintes usando versões dilatadas ou contraídas dos blocos anteriores já processados. O objetivo é que, a medidas que os dados vão sendo processado, o dicionário contenha elementos o mais próximo possível

dos blocos da imagem, pois com isso a distorção entre a imagem original e a reconstruída será menor.

Dado um bloco de tamanho NxN da imagem, o algoritmo percorre o dicionário buscando, dentre os elementos do na nível mais alto, aquele que da o melhor casamento, segunda o critério de otimização RD (Taxa e distorção). Neste critério tenta-se encontrar o menor custo para representar o bloco da imagem. O custo J para representar o bloco é dado por:

$$J = D + \lambda R$$

- R = Taxa: é o número de bits por pixel gastos para transmitir o índice deste elemento do dicionário.
- D = Distorção obtida através do erro médio quadrático entre o bloco da imagem e um dado elemento.
- λ = Constante (Multiplicador de Lagrange) (DUARTE, 2002).

Obtém-se neste nível qual índice do dicionário gerou o menor custo "J". Em seguida o algoritmo desce um nível na árvore, dividindo o bloco original da imagem em duas partes. Agora a busca se dá no nível abaixo do nível maior, onde o número de pixel de cada elemento é metade do número de pixel do bloco original. Calculam-se os custos "JÁ" e "JB" relativos aos as duas partes da imagem divididas neste nível. Segue-se esse processo até que todos os custos da árvore sejam calculados.

Após calculados todos os custos, inicia-se o processo de "podagem" que é a obtenção da árvore ótima que melhor representa o bloco.

Seja J o custo do nó pai, JÁ e JB os custos dos seus filhos, esquerdo e direito respectivamente. Compara-se J com JÁ + JB. Se a soma dos custos dos nós filhos for menor do que o custo do nó pai, a árvore não será podada, ou seja, é melhor que ela seja dividida, pois representá-la por dois blocos menores separadamente é melhor do que representá-la por um bloco grande. Para este nó pai atribui-se flag = 1.

Entretanto, se a soma dos custos dos nós filhos for maior que a soma do custo do nó pai, a árvore será podada, pois o algoritmo encontrou uma melhor representação para o bloco da imagem usando um elemento do dicionário maior. Atribui-se flag = 0 ao nó pai e o índice do elemento que gerou este custo é guardado. Esse processo continuará até que toda a árvore seja percorrida.

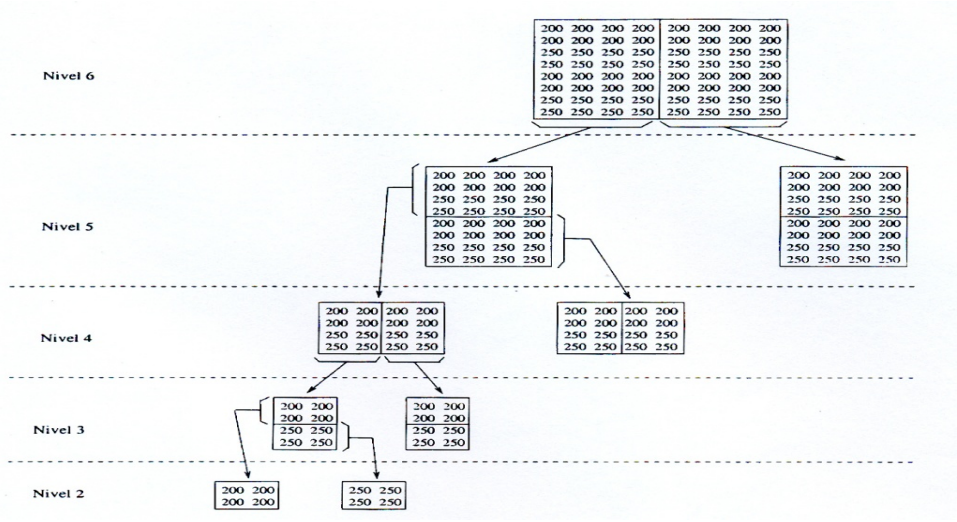

Figura 7 - Representação de uma árvore para um determinado bloco da imagem após a podagem

4.3.3. CODIFICAÇÃO

ETAPAS:

Percorre-se a árvore ótima começando do nível mais alto (nó raiz) até o mais baixo (nó terminal ou folha), verificando primeiro o nó esquerdo e depois o direito:

- Se o encontrar um flag = 1:
- Envia este flag para o decodificador;
- Desce um nível na árvore;
- Se encontrar um nó terminal:
- Envia para o decodificador um flag = 0;
- Envia o índice do elemento do dicionário que gerou o nó.

Denomina-se bitstream a seqüência final de flags e os índices enviados ao decodificador após a varredura na árvore (DUARTE, 2002).

ATUALIZAÇÃO DO DICIONÁRIO

Esse é o processo que faz do MMP um algoritmo adaptativo. Através de concatenações, expansões e contrações, são incorporados novos elementos no dicionário à medida que a imagem vai sendo processado. Toda vez que se chega a dois nós terminais, estes são concatenados, ou seja, juntados para formar um novo elemento do dicionário que será inserido no nível mais acima. Este, por sua vez, é expandido e inserido nos níveis superiores do dicionário. Essa expansão pode ser feita tanto repetindo os valores dos pixels do bloco, bem como fazendo uma média de dois pixels vizinhos e inserido entre eles de modo que no final o novo bloco possua o dobro de elementos de seu gerador.

Já na contração, o processo é inverso e análogo em relação à expansão. Através de eliminação de pixels intercalados ou da substituição de dois pixels pela média entre eles, pode-se formar novos blocos com metade de elementos de seus geradores e inserindo-os no nível mais abaixo.

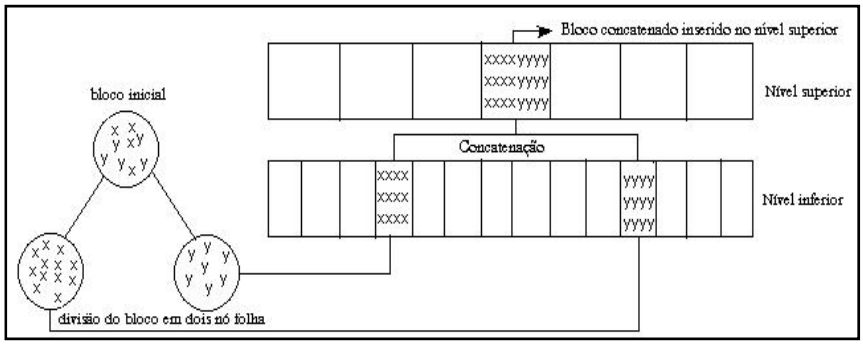

Figura 8 - Atualização do dicionário

4.3.4. DECODIFICAÇÃO

O decodificador precisa receber inicialmente os valores minpixel, maxpixel e o passo para que possa montar o dicionário inicial. Em seguida começa a processar o bloco pela raiz da árvore assim como o codificador. A imagem é recriada a partir dos elementos do dicionário, de acordo com os *flags* recebidos. O mesmo processo de atualização do

dicionário efeito pelo decodificador de modo que ao final tenha-se o mesmo dicionário necessário para compor a imagem.

O processo de decodificação se torna bem mais rápido, pois nessa etapa ao há necessidade de calcular os custos década nó bem como a podagem da árvore (DUARTE, 2002).

5. CONCLUSÃO

As técnicas de compressão, que foram apresentadas neste referente artigo, trouxeram uma abordagem mais simplificada, a respeito da descrição de cada uma delas. Neste estudo, podemos ressaltar que, o JPEG possui um melhor desempenho quando aplicado a imagens naturais monocromáticas e coloridas, porque os coeficientes que representam os pixels da imagem são descorrelacionados e consegue-se distribuí-los em relação à freqüência, aumentando assim, a taxa de compressão.

O SPIHT, em contrapartida, é um método flexível, que se adapta a necessidade do destino do arquivo a ser enviado, onde, por exemplo, se tivermos quatro destinos requisitando o mesmo arquivo de imagem, mas com qualidades diferentes com qualquer outro método seria necessário comprimir o arquivo três vezes para enviá-los aos destinos, mas com o SPIHT basta que se enviem ao destino os primeiros 'N' bytes do arquivo já comprimido de acordo com a necessidade do mesmo e isso será suficiente para reconstituir a figura.

Já o MMP, por ser um método adaptativo, pois possui um dicionário que é atualizado à medida que os blocos da imagem vão sendo processados, mostrou-se mais eficiente que o SPHIT quando aplicada em imagens mistas, ou seja, imagem que contêm texto e imagem. Quando comparado com o JPEG, o algoritmo também se mostrou melhor.

6. REFERÊNCIAS

GONZALEZ, R. C.; WOODS, R. E., **Digital Image Processing**, 2. ed., Addison-Wesley, Reading, 1992.

MARQUES, O.F.; VIEIRA, H. N. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999.

SALOMON, D. **Data compression (Computer science)**. 3. ed. New York: Springer, 2004.

SPRLJAN, N.; GRGIC, S.; GRGIC, M.. **Modified SPIHT Algoritm for Wavlet Packet Image Coding**. Disponível em: <http://www.vcl.fer.hr/papers_pdf/Modified%20SPIHT%20Algorithm%20for%20Wavelet%20Packet%20Image%20Coding.pdf> . Acesso em: 22 jul 2008.

DUARTE, M. H. V. **Codificação de Imagens Estéreo Usando Recorrência de Padrões**.[Rio de Janeiro] 2002. XX, 203 pp., 29, 7 cm (COPPE/UFRJ, D.Sc., Engenharia Elétrica, 2002) Tese – Universidade Federal do Rio de Janeiro, COPPE.

GPICE, **MMP – Multidimensional Multiscale Parser**. Disponível em: <http://gpice.cefetce.br/mmp/funcMmp.php> Acesso em : 05 ago 2008.