UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE COMUNICAÇÕES

TESE DE MESTRADO

PROCEDIMENTOS PARA MEDIÇÃO E MINIMIZAÇÃO DO EFEITO DE BLOCO DECORRENTE DO PROCESSAMENTO DIGITAL DE IMAGENS (PDI)

Fernando Silvestre da Silva

Orientador: Prof. Dr. Yuzo Iano

Banca Examinadora:

Prof. Dr. Guillermo Fernández Segovia (UCV – Valparaíso – Chile)

Prof. Dr. João Baptista Tadanobu Yabu-uti (FEEC – UNICAMP)

Prof. Dr. Edson Moschim (FEEC – UNICAMP)

Prof. Dr. João Batista Destro Filho (FEEC – UNICAMP)

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica

Campinas, SP – Brasil

Fevereiro – 2001

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Si38p

Silva, Fernando Silvestre da
Procedimentos para medição e minimização do efeito de bloco decorrente do processamento digital de imagens (PDI) / Fernando Silvestre da Silva.--Campinas, SP: [s.n.], 2001.

Orientador: Yuzo Iano.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e Computação.

1. Processamento de imagens – Técnicas digitais. 2. Compressão de dados (Telecomunicações). 3. Teoria da codificação. I. Iano, Yuzo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

RESUMO

Este trabalho apresenta uma nova aproximação para o problema de medição e redução do Efeito de Bloco em codificação por transformada baseada em blocos objetivando melhorar a qualidade subjetiva das imagens. Uma nova medida de blocagem chamada Medida de Borda na Fronteira do Bloco (MBFB) foi introduzida, baseada nas diferenças das derivadas adjacentes próximas à fronteira do bloco, visando obter uma medida mais precisa da impressão subjetiva do olho humano à respeito desse artefato. O MBFB é calculado para todas as fronteiras dos blocos da imagem e então armazenado num vetor, que é codificado e anexado ao cabeçalho da imagem para ser transmitido/armazenado. No esquema proposto, de acordo com as mudanças nos elementos dos vetores das imagens original e reconstruída, o decodificador é capaz de determinar as áreas onde o Efeito de Bloco ocorre e então, um procedimento interpolativo localizado para redução da blocagem é executado. O esquema proposto apresenta uma melhora significativa da qualidade subjetiva das imagens, especialmente para baixos *bit rates* sem causar significativo "borramento". Essa técnica pode ser aplicada a qualquer codificação por transformada baseada em blocos convencional, tal como os padrões MPEG ou JPEG, através de uma simples rotina adicional sem introduzir modificações em seus códigos originais.

ABSTRACT

This work presents a new approach to the Blocking Effect measurement and reduction problem in block-based transform coding in order to improve pictures subjective quality. A new blockiness measure called Block Boundary Edge Measure (BBEM) is introduced, based on differences of adjacent slopes near block boundary, in order to obtain an accurate measure of the human eye subjective impression concerning this artifact. The BBEM is evaluated for all image block boundaries and then stored in a vector, which is encoded and attached to the picture header to be transmitted/stored. In the proposed scheme, according to changes in vectors' elements for original and reconstructed pictures, the decoder is able to determine the areas where Blocking Effect occurs and then a localized interpolative procedure for blockiness reduction is performed. The proposed scheme presents a significant improvement of pictures subjective quality, specially for lower bit rates without causing significant blurring. This technique can be applied to any conventional block-based transform coding, such as MPEG or JPEG standards by a simple additional routine, without introducing modifications on their original codes.

OFERECIMENTO

"Disseram-vos que a vida é escuridão; e no vosso cansaço, repetis o que os cansados vos disseram.
E eu vos digo que a vida é realmente escuridão, exceto quando há um impulso.
E todo o impulso é cego, exceto quando há saber.
E todo o saber é vão, exceto quando há trabalho.
E todo o trabalho é vazio, exceto quando há amor.
E quando trabalhais com amor, vós vos unis a vós próprios, e uns aos outros, e a Deus.
E que é trabalhar com amor?
É tecer o tecido com fios desfiados de vosso próprio coração, como se vosso bem-amado fosse usar esse tecido.
É construir uma casa com afeição, como se vosso bem-amado fosse habitar essa casa.
É semear as sementes com ternura e recolher a colheita com alegria, como se vosso bem-amado fosse comer os frutos.
É pôr em todas as coisas que fazeis um sopro de vossa alma."

G. K. Gibran

À Deus eu ofereço este trabalho, pois foi feito com Amor, Dedicação, Esforço e Alegria. Obrigado Senhor, por ter iluminado os meus caminhos e me conduzido com Tua Sabedoria, especialmente nas horas mais difíceis.

AGRADECIMENTO À FAPESP

Agradeço a **FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo** – pelo apoio indispensável que tem sido oferecido à este pesquisador através dos Projetos: "**Algoritmo de Medição de Efeito de Bloco em Ambiente MPEG-2 (PDI)**", Processo FAPESP no. 98/95318-7 e "**Procedimentos para Medição e Minimização de Distorções Decorrentes do Processamento Digital de Imagens (PDI) – Restauração de Imagens",** Processo FAPESP no. 98/12898-0, que viabilizaram a realização desta pesquisa, desde a Iniciação Científica até o Mestrado, provendo suporte financeiro e possibilitando a aquisição dos recursos e equipamentos fundamentais para o desenvolvimento desta pesquisa.

AGRADECIMENTOS

Ao alcançar o fim de uma etapa importante como esta atingindo o objetivo inicial, eu gostaria de expressar meus agradecimentos às pessoas e instituições que tanto contribuíram direta ou indiretamente para que este trabalho fosse realizado.

Primeiramente, desejo agradecer ao Professor Doutor Yuzo Iano que, como orientador, sempre foi um ponto de referência e um modelo de força, paciência e perseverança que me guiou, desde a Iniciação Científica, ao longo destes três anos e que certamente ainda contribuirá muito durante a realização do meu Doutorado.

À Universidade Estadual de Campinas (UNICAMP) pela oportunidade e pelo apoio que recebi durante a realização desta pesquisa. À FAPESP pelo suporte financeiro concedido à mim desde a Iniciação Científica.

Aos colegas de departamento e, em especial, ao amigo Vicente pela ajuda sempre presente e independente de horário. Aos amigos Camilo, Júlio, Rodrigo, Wilson e Luís Felipe pelas divertidas reuniões que muitas vezes me fizeram encarar os problemas com mais disposição e alegria.

Aos meus amigos Alexandre e Lia que, no momento em que nós não víamos nenhum caminho, indicaram a direção de um possível horizonte a ser alcançado.

Aos meus pais, Damásio e Lúcia que sempre me incentivaram, compreenderam e apoiaram nos momentos difíceis. Aos meus sogros, Ary e Tereza pelo apoio e amizade que têm me dedicado.

À minha família, especialmente aos meus Avós Francisco e Marcelina, que tanto me ajudaram no início de minha caminhada em Campinas.

À minha esposa Ana Lúcia que com o seu Amor, Dedicação e Inteligência deu-me força e coragem nos momentos em que eu mais precisei e sugestões sempre muitos positivas e fundamentais para o desenvolvimento deste trabalho.

À Deus, pela Iluminação e pelo Trabalho.

Fernando Silvestre da Silva

Sumário

Lista de Figuras

Lista de Tabelas

Abreviaturas

AC – Alternate Current

bpp – bits por pixel

cbp – Coded Block Pattern

CCITT – International Consultative Committee for Telephone and Telegraph

DC – Direct Current

DCT: Discrete Cosine Transform

DFT: Discrete Fourier Transform

DPCM – Differential Pulse Code Modulation

DSTr – Discrete Sine Transform with Axis Rotation

EOB – End of Block

FB – Função de Borda

FE – Fator de Escala

GOP – Group of Pictures

HDTV – High Definition Television

HVS – Human Visual System

IEEE - Institute of Electrical and Electronics Engineers

JPEG – Joint Photographic Experts Group

KLT: Karhunen-Loève Transform

MBFB – Medida de Borda na Fronteira do Bloco

Mbps: Mega bits por segundo

MBs: Macroblocos

MPEG-2 – Moving Picture Experts Group

MSE – Mean Square Error

PCM – Pulse Code Modulation

PDI: Processamento Digital de Imagens

PDS: Processamento Digital de Sinais

Pixel: Picture element

PSNR: Peak Signal to Noise Ratio

REB – Região Entreblocos

RL – Run-Length

RLB – Run-Length Binário

SNR: Signal to Noise Ratio

TM5 – Test Model 5

TEB – Taxa de Erro de Bits no VBFB

VBFB – Vetor de Borda na Fronteira dos Blocos

VBV – Virtual Buffering Verifier

VLC: Variable Length Coding

Capítulo 1

Introdução

Atualmente, o Processamento Digital de Imagens (PDI) possui uma ampla gama de aplicações que vai desde aplicações médicas, processamento de imagens adquiridas por satélites de sensoreamento remoto, teleconferência, tratamento de imagens geológicas, radares e sonares, até cinema digital e televisão digital de alta definição (HDTV – *High Definition Television*), sendo esta de especial interesse pela utilização do padrão MPEG-2 (*Moving Picture Experts Group*) de compactação de vídeo.

A representação digital da informação visual, em sua forma "crua" (não processada), gera, por característica inerente, um número significativo de grandes arquivos. Dessa forma, o custo em capacidade de armazenamento ou largura de banda de transmissão dessa informação seria muito alto.

Nesse contexto, objetivando a melhoria e economia no armazenamento e transmissão em largura de banda (e, por conseguinte, economia também do ponto de vista comercial), surgiram as técnicas de compressão utilizadas no processamento digital de imagens. Segundo Watkinson [1], talvez seja em *broadcasting* onde o uso da compressão terá seu maior impacto, uma vez que há somente um espectro eletromagnético e a pressão de outros serviços tais como a telefonia celular, torna o eficiente uso da largura de banda, uma exigência.

1.1 – Compressão Digital

A compressão digital de imagens é fundamentada, basicamente, no conceito de informação visual da imagem. É importante salientar que esse conceito não está ligado a Teoria de Informação e sim à um conceito subjetivo de qualidade de imagem dado pelo observador (ou pelo sistema ao qual se destina a imagem).

Dentro dessa classificação podemos dividir a informação visual em duas classes [2]: 'informação necessária' e 'informação supérflua'. Informação necessária é aquela que, se retirada da imagem original mesmo que parcialmente, provoca grandes deteriorações na qualidade subjetiva da imagem dados os parâmetros de percepção e tolerância do sistema-destino (HVS – *Human Visual System*). Informação supérflua é aquela que se retirada provoca pouca ou nenhuma mudança na imagem sendo que esta pode ser utilizada ou bem avaliada pelo sistema-destino.

A Informação supérflua ainda pode ser dividida em duas classes diferentes segundo a sua origem [2]:

Informação redundante*:* também relacionada com Redundância Estatística.

Esse tipo de informação está relacionado à correlação e à interdependência dos dados. Essa informação pode ser removida dos dados sem deteriorar a qualidade da imagem pois ela pode ser completamente predita a partir do restante da imagem.

Informação irrelevante*:* também relacionada com Redundância Subjetiva.

Esse tipo de informação está relacionado às características que podem ser removidas causando apenas perdas toleráveis para o sistema que receberá a imagem. Ao contrário da Informação redundante, esse tipo de informação não pode ser recuperada após sua eliminação e, assim, deve ser eliminada com critério para evitar perdas significativas na qualidade da imagem.

Assim, as técnicas de compressão são classificadas segundo o tipo de informação que é explorada : Compressão sem Perdas (*Lossless Compression*) e Compressão com Perdas (*Lossy Compression*).

A compressão sem perdas explora somente a redundância estatística dos dados e portanto, não provoca deterioração da imagem e sempre pode ser aplicada. Por característica, geralmente apresenta taxas de compressão baixas (com relação à compressão com perdas).

A compressão com perdas, por sua vez, apresenta altas taxas de compressão e explora a redundância subjetiva. Assim, essa técnica nem sempre pode ser aplicada por acarretar perdas de informação da imagem.

Uma situação onde essas perdas poderiam causar graves erros de interpretação é, por exemplo, na compactação de imagens médicas (como tomografia computadorizada). Se, devido às perdas de informação, ocorrerem deteriorações na imagem, essa imagem pode ser mal interpretada levando a um diagnóstico errado, o que poderia ser danoso para o paciente.

No entanto, em uma grande parte das aplicações comerciais, pequenas falhas não levam a conseqüências tão graves. Assim, a compressão com perdas é largamente utilizada nos padrões de compactação de dados como o MPEG-2.

Uma das técnicas de compressão digital mais utilizada nos codificadores padrão de imagem é a Codificação por Transformada Baseada em Blocos [3]. Esse esquema de compressão baseia-se na divisão dos dados originais em blocos distintos, na aplicação de uma transformação ortonormal nesses blocos e na subsequente quantização dos coeficientes transformados de forma a aproveitar as características de concentração de energia da transformada, o que propicia altas taxas de compressão.

A desvantagem dessa técnica é a geração de distorções na imagem devido a perda de informação na quantização. A distorção mais prejudicial à qualidade subjetiva da imagem recuperada é o Efeito de Bloco, que consiste no evidenciamento das fronteiras dos blocos em detrimento do restante da imagem e ocorre quando são utilizados poucos bits para coeficientes significativos.

Várias técnicas tem sido desenvolvidas com o intuito de reduzir esse efeito indesejável da compressão. Essas técnicas variam desde a simples filtragem passa-baixas, passando por DPCM-DC (*Differential Pulse Code Modulation – Direct Current*) até a aplicação de desenvolvimentos mais recentes como Redes Neurais.

1.2 – Contribuições e Organização deste Trabalho

A proposta dessa pesquisa é a criação de um algoritmo de medição e redução de Efeito de Bloco em imagens compactadas por esquemas de codificação por transformada baseada em blocos com altas taxas de compressão. Com esse objetivo, foi desenvolvida uma medida de borda localizada, o MBFB – Medida de Borda na Fronteira do Bloco, que se inserida no cabeçalho da imagem codificada, possibilita um procedimento de correção, melhorando a qualidade subjetiva da imagem recuperada sem causar uma perda excessiva (borramento) dos seus detalhes.

Dessa forma, são apresentadas aqui as contribuições principais deste trabalho: a) o desenvolvimento do critério subjetivo de deteção de borda MBFB; b) um esquema de compressão de imagens por transformada utilizando o procedimento interpolativo de redução de efeito de bloco; c) e a medida de qualidade subjetiva de imagens com relação ao efeito de bloco TEB – Taxa de Erro de Bits no Vetor de Bordas na Fronteira dos Blocos (VBFB).

Assim, esta dissertação foi dividida nos capítulos:

Capítulo 2: Transformações Ortonormais usadas em PDI

Apresenta-se nesse capítulo um estudo sobre as transformações lineares discretas e suas propriedades mais importantes para a aplicação em PDI. Pela sua importância, também são apresentadas as caraterísticas principais da transformada DCT (*Discrete Cosine Transform*) de forma a situar o estudo da compressão digital de imagens neste trabalho.

Capítulo 3: Codificação por Transformada e Efeito de Bloco

Um estudo sobre o princípio de funcionamento da codificação por transformada destacando-se as etapas de quantização e de codificação é realizado nesse capítulo. Por último é apresentada uma descrição detalhada da formação do efeito de bloco, suas características e são citadas algumas técnicas tradicionais para sua correção.

Capítulo 4: Padrões de Compressão

Neste capítulo descreve-se sumariamente o padrão de compressão de imagens JPEG (*Joint Photographic Experts Group*) e o padrão de compressão de vídeo MPEG-2 (*Moving Picture Experts Group*) que foram utilizados como base de comparação para os resultados obtidos neste trabalho.

Capítulo 5: Proposta para Redução de Efeito de Bloco

São detalhadas nesse capítulo as etapas da técnica de medição e redução de efeito de bloco proposta: a) o critério de deteção de bordas MBFB e a sua representação na forma vetorial (VBFB – Vetor de Bordas na Fronteira dos Blocos); b) o procedimento de interpolação utilizado para a redução do efeito de bloco; c) uma nova medida de qualidade visual de imagens referente ao efeito de bloco. Também são apresentados o esquema de codificação por transformada genérico utilizando a técnica proposta e os métodos de codificação do VBFB.

Capítulo 6: Apresentação e Análise dos Resultados

São apresentados neste capítulo os resultados da simulações e as respectivas análises realizadas durante o desenvolvimento desse trabalho. Esses resultados são divididos em seções conforme o padrão utilizado como base para a aplicação do procedimento proposto: JPEG e MPEG-2 e são analisados quanto a qualidade visual (subjetiva) e quanto às medidas de PSNR (Peak Signal-to-Noise Ratio) e TEB (Taxa de Erro de Bits no Vetor de Borda na Fronteira dos Blocos VBFB).

Capítulo 7: Conclusões e Sugestões para Trabalhos Futuros

Finaliza-se este trabalho apresentando-se as conclusões à luz dos resultados obtidos, analisando-se as perspectivas de continuação e avanços que poderão ser implementados a partir dos resultados já obtidos.

Capítulo 2

Transformações Ortonormais

2.1 – Introdução

Como foi comentado anteriormente, operações de compressão reduzem significativamente o número de bits necessários para se descrever uma imagem de forma que possam ser eficientemente armazenadas ou eletronicamente transportadas sem perdas consideráveis de informação, mantendo-se a fidelidade da imagem.

Dentro dos esquemas de compressão utilizados atualmente, os que oferecem as maiores taxas de compressão (admitindo-se pequena perda de informação) são os que utilizam o sistema de codificação por transformada (incluindo-se o JPEG e o MPEG-2).

O processo de codificação por transformada engloba a aplicação de uma transformação ortonormal na imagem original, usualmente feita dividindo-se as imagens em blocos menores para acelerar a execução, seguida de uma quantização não-uniforme e por uma codificação com códigos de comprimento variável (VLC – *Variable Length Coding*), como no esquema mostrado na Fig. 2.1 abaixo.

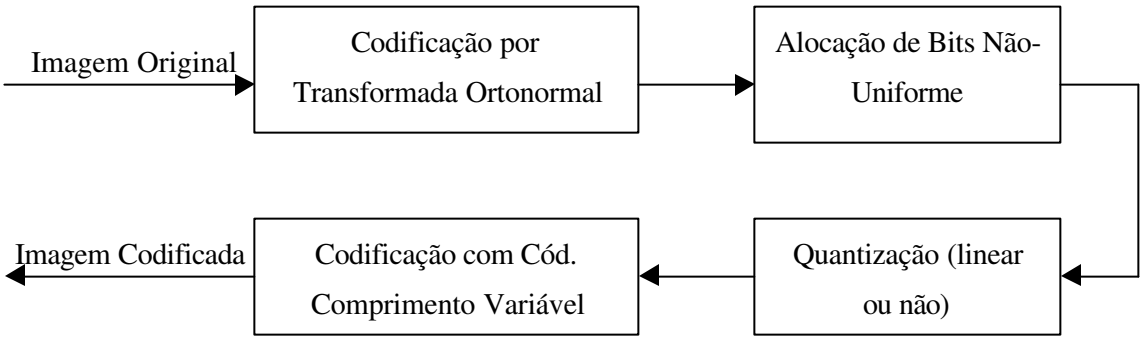


Fig. 2.1 – Esquema genérico de Codificação por Transformada.

Dessa forma apresenta-se neste capítulo, um resumo das características das transformações ortonormais e das transformadas mais importantes no Processamento Digital de Imagens. No próximo capítulo serão apresentados detalhes sobre a quantização e a codificação VLC geralmente utilizada nos padrões de compressão de imagens.

2.2 – Transformadas Ortogonais Discretas

Uma Interpretação Visual da Operação Transformada [3]

Colocar uma interpretação visual da operação transformada é extremamente interessante para conferir uma noção intuitiva acerca de suas propriedades básicas, em especial, nas características redutoras da correlação.

Suponha que a imagem NxN seja encarada como um vetor N^2x1 e que se observe seus elementos em conjuntos de três (equivalentes à três coordenadas). Normalmente, vetores de dados maiores são obviamente usados, mas, como o número de coordenadas implica diretamente no número de dimensões da representação e o maior espaço com representação plana intuitiva é o tridimensional , a escolha por três é imediata.

Suponha agora que os dados da imagem tenham razoavelmente alta correlação (portanto aproximadamente a mesma energia) entre os elementos, que é uma condição necessária para compressão eficiente. Essa suposição, na prática, não é muito restritiva, pois as imagens comuns costumam apresentar essa característica.

Marcando-se num gráfico os valores dos três elementos (coordenadas) como um vetor de dados num sistema de coordenadas 3D, pode-se esperar que a região ocupada pela maioria dos vetores desse tipo fique próxima à linha **x = y = z** (dada a alta correlação entre os 3), semelhante a um cilindro alongado, ilustrado na figura a seguir.

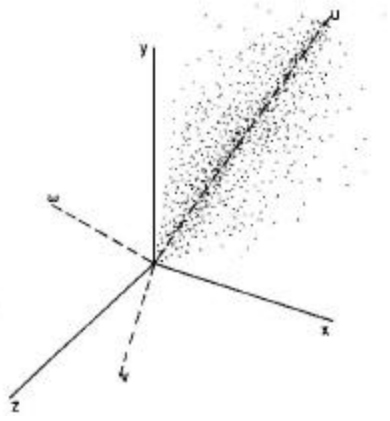


Fig. 2.2 – Representação de posições semelhantes de vetores de 3 elementos em 3D. [3]

Considere agora os vetores de dados transformados pela rotação de coordenadas para os eixos **u**, **v** e **w** rotacionados com relação ao sistema de coordenadas originais. Nesse novo conjunto

de coordenadas, todos os elementos têm o componente em **u** muito maior que nas direções **v** e **w**, indicando uma compactação de energia no coeficiente associado ao vetor base **u**.

Pode-se então especificar os dados (agora no domínio da transformada) para se transmitir ou armazenar o componente **u** com precisão e, dependendo das necessidades, **v** e **w** de forma aproximada.

Em termos de energia, a perda pela eliminação ou aproximação de **v** e **w** é bem menor do que se fosse realizado o mesmo procedimento para qualquer um dos três componentes originais (**x**, **y**, ou **z**).

Essa interpretação toda pode ser estendida conceitualmente à um vetor de 8 ou 16 elementos, como é usual em codificação por transformada. Outra importante propriedade dessas rotações é que o produto escalar (interno) permanece invariante sob tal operação. Isso leva à importante propriedade da invariância da energia entre dados e domínio transformado.

O objetivo desta subseção é apenas conferir a noção intuitiva acerca das propriedades envolvidas na operação transformada. Essas serão devidamente discutidas nos itens subsequentes.

Operação Transformada 1D

Na subseção anterior, tratou-se de dados tridimensionais sendo que o conjunto **(x, y, z)** formava o sistema de eixos (espaço) original dos dados e que **(u, v, w)** formava o espaço transformado. Os vetores formadores do espaço **(u, v, w)** são comumente referidos como **Vetores Base**.

A passagem de um espaço para o outro é feita pela projeção ortogonal do ponto que vai ser transformado (suas três coordenadas) sobre o conjunto de eixos do espaço transformado (Vetores Base).

A realização matemática dessa projeção é feita pela operação vetorial chamada Produto Interno, ou Produto Escalar. O Produto Interno entre dois vetores é definido como a soma dos produtos das coordenadas uma a uma do segundo vetor com o complexo conjugado das coordenadas do primeiro.

Nesse ponto faz-se necessário observar que o presente pesquisador, tendo por objetivo uniformizar a notação por todo o trabalho, alterou algumas das fórmulas originais constantes nas referências citadas para que se adequassem à perspectiva dessa pesquisa.

Dessa forma, dando continuidade ao exemplo, visualiza-se matematicamente, por exemplo [3], um vetor de dados de 4 elementos por:

$$\boldsymbol{f}^{T} = [\ f_1\ f_2\ f_3\ f_4\ ] \tag{2.1}$$

O primeiro coeficiente transformado é o produto desses dados com o primeiro vetor base $\boldsymbol{t_1}$:

$$\boldsymbol{t_1}^T = [\ t_{11}\ t_{12}\ \ t_{13}\ \ t_{14}\ ] \tag{2.2}$$

Logo o produto interno fica:

$$c_1 = t_{11}^*.f_1 + t_{12}^*.f_2 + t_{13}^*.f_3 + t_{14}^*.f_4 \tag{2.3}$$

Analogamente, para o segundo vetor base $\boldsymbol{t_2}$:

$$\boldsymbol{t_2}^T = [\ t_{21}\ t_{22}\ \ t_{23}\ \ t_{24}\ ] \tag{2.4}$$

$$c_2 = t_{21}^*.f_1 + t_{22}^*.f_2 + t_{23}^*.f_3 + t_{24}^*.f_4 \tag{2.5}$$

E assim por diante. Podemos agora agrupar esses produtos internos todos em uma única representação matricial da forma [3]:

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} t^*_{11} & t^*_{12} & t^*_{13} & t^*_{14} \\ t^*_{21} & t^*_{22} & t^*_{23} & t^*_{24} \\ t^*_{31} & t^*_{32} & t^*_{33} & t^*_{34} \\ t^*_{41} & t^*_{42} & t^*_{43} & t^*_{44} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \Rightarrow \boldsymbol{c} = \boldsymbol{T}^{*\boldsymbol{T}} \cdot \boldsymbol{f} \tag{2.6}$$

Onde $\boldsymbol{c}$ é o vetor coluna dos coeficientes transformados $[\ c_1\ c_2\ c_3\ c_4\ ]^T$, $\boldsymbol{f}$ é o vetor coluna dos dados e $\boldsymbol{T}$ é a matriz de transformação cujas colunas são os vetores base. Por simplicidade de notação denominaremos $\boldsymbol{T}^{*\boldsymbol{T}}$ por $\boldsymbol{T'}$.

Também podemos escrever na forma de somatório [3]:

$$c_p = \sum_{k=1}^{N} f_k \cdot t^*_{pk}, \qquad 1 \le p \le N \tag{2.7}$$

Os elementos de $\boldsymbol{f}$ e $\boldsymbol{T}$, nesse caso, são definidos como números complexos e os índices dos vetores base e de dados podem ser alternativamente trabalhados como se fossem de 0 à N – 1.

Como a matriz de transformação $\boldsymbol{T'}$ é formada por vetores ortogonais, o $\boldsymbol{rank(T')=N}$ e a matriz é, portanto, inversível. Assim, para calcular a transformação inversa basta multiplicar ambos os lados da equação (2.6) por $\boldsymbol{T'^{-1}}$ para se obter a expressão :

$$\boldsymbol{T'^{-1}.T'.f} = \boldsymbol{T'^{-1}.c} \Rightarrow \boldsymbol{I.f} = \boldsymbol{T'^{-1}.c} \Rightarrow \boldsymbol{F} = \boldsymbol{T'^{-1}.c} \tag{2.8}$$

e o par de transformadas 1D genérico é portanto[3]:

$$\boldsymbol{c} = \boldsymbol{T'.f} \qquad c_p = \sum_{k=0}^{N-1} f_k \cdot t'_{pk} \qquad 0 \le p \le N-1 \tag{2.9}$$

$$\boldsymbol{f} = \boldsymbol{T'^{-1}.c} \qquad f_k = \sum_{p=0}^{N-1} c_p \cdot t'^{-1}_{pk} \qquad 0 \le k \le N-1 \tag{2.10}$$

Nesse ponto é interessante concentrar-se nas propriedades de ortogonalidade e conservação de energia das transformadas que serão úteis na simplificação destas duas expressões.

Ortogonalidade, Ortonormalidade e Conservação de Energia :

Seja a matriz de transformação $\boldsymbol{T}$ de tamanho NxN. A Propriedade da Ortogonalidade estabelece que se [3]:

$$\sum_{p=0}^{N-1} t'^{-1}_{pk} \cdot \left(t'^{-1}_{pq}\right)^* = \begin{cases} 0 \ , k \neq q \\ A \ , k = q \end{cases} \tag{2.11}$$

Ou na forma de Produto Interno:

$$\left\langle \boldsymbol{t'^{-1}_k} , \boldsymbol{t'^{-1*}_q} \right\rangle = \begin{cases} 0 \ \ , se\ k \neq q \\ A \ \ , se\ k = q \end{cases} \tag{2.12}$$

onde o vetor base $\boldsymbol{t'^{-1}_k}$ é dado por:

$$\boldsymbol{t'^{-1}_k} \overset{\Delta}{=} \left[\boldsymbol{t'^{-1}_{0k}} \quad \boldsymbol{t'^{-1}_{1k}} \quad \cdots \quad \boldsymbol{t'^{-1}_{(N-1)k}} \right]^T \tag{2.13}$$

A partir da equação (2.11) pode-se afirmar que:

$$\boldsymbol{T'^{-1}.T'^{-1\,*T}} = A\boldsymbol{.I} \Leftrightarrow \boldsymbol{T'^{-1*T}} = A\boldsymbol{.T'} \tag{2.14}$$

Lembrando que $\boldsymbol{T'=T^{*T}}$, e fazendo arbitrariamente $A=1$, então tem-se :

$$\boldsymbol{T^{-1}=T^{*T}} \tag{2.15}$$

Assim, as expressões para a transformada direta e inversa ficam:

$$\boldsymbol{c} = \boldsymbol{T^{*T}.f} \qquad c_p = \sum_{k=0}^{N-1} f_k \cdot t^*_{pk} \qquad 0 \leq p \leq N-1 \tag{2.16}$$

$$\boldsymbol{f} = \boldsymbol{T.c} \qquad f_k = \sum_{p=0}^{N-1} c_p \cdot t_{pk} \qquad 0 \leq k \leq N-1 \tag{2.17}$$

Considerando-se agora a energia dos coeficientes transformados por uma transformada ortonormal, pode-se observar que a energia total do sinal também é conservada. Matematicamente, sendo a energia do sinal $\boldsymbol{f}$ [4]:

$$\|f\|^2 = \sum_{k=0}^{N-1} |f_k|^2 \tag{2.18}$$

e, para uma transformada ortonormal, $\boldsymbol{T^{-1} = T^{*T}}$, pode-se calcular a energia dos coeficientes transformados como sendo [4]:

$$\| \boldsymbol{c} \|^2 \equiv \sum_{p=0}^{N-1} \left| c_p \right|^2 = \boldsymbol{c}^{*T} \cdot \boldsymbol{c} = \boldsymbol{f}^{*T} \cdot [\boldsymbol{T}] \cdot [\boldsymbol{T}]^{*T} \cdot \boldsymbol{f} = \boldsymbol{f}^{*T} \cdot \boldsymbol{f} = \sum_{k=1}^{N} \left| f_k \right|^2 \equiv \| \boldsymbol{f} \|^2 \quad (2.19)$$

Percebe-se que a transformada unitária preserva a energia do sinal e portanto, realmente a transformação é simplesmente uma rotação do vetor $\boldsymbol{f}$, conforme colocado na subseção "Interpretação Visual da Transformada".

Operação Transformada 2D e suas Propriedades

Estendendo o caso unidimensional para o contexto do processamento de imagem, tem-se o seguinte par de transformadas genérico para o caso 2D:

$$C_{pq} = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} F_{jk} \cdot t^*_{pqjk} \qquad 0 \le p,q \le N-1 \quad (2.20)$$

$$F_{jk} = \sum_{p=0}^{N} \sum_{q=0}^{N} C_{pq} \cdot t_{pqjk} \qquad 0 \le j,k \le N-1 \quad (2.21)$$

onde: t_{pqjk} , também chamada transformada imagem, é um conjunto de funções base discretas ortonormais, satisfazendo as seguintes propriedades (extensões do caso unidimensional) :

Ortogonalidade e Ortonormalidade:

Generalizando o conceito de Ortogonalidade para o caso bidimensional, pode-se escrever que [4]:

$$\sum_{j=0}^{N-1} \sum_{k=0}^{N-1} t_{pqjk} \cdot t^*_{p'q'jk} = A.\delta\left(p - p', q - q'\right) \quad (2.22)$$

onde :

$$\delta(p,q) = \begin{cases} 1 & , se \;\; p = q = 0 \\ 0 & , caso \;\; contrário \end{cases} \quad (2.23)$$

Da mesma forma que para a transformada unidimensional, fazendo-se $A=1$, então a transformada é chamada Ortonormal.

Conservação de energia

Como no caso unidimensional pode-se provar que se a transformada for ortonormal a energia é conservada na transformação. A dedução dessa propriedade é análoga à unidimensional e, sua expressão final é dada por [4]:

$$\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}\left|F_{jk}\right|^2 = \sum_{p=0}^{N-1}\sum_{q=0}^{N-1}\left|C_{pq}\right|^2 \tag{2.24}$$

Separabilidade:

A representação matricial genérica de uma transformada bidimensional é [4]:

$$\mathbf{C} = \mathbf{T}^{*T}\cdot\mathbf{F} \tag{2.25}$$

onde $\mathbf{F}$ e $\mathbf{C}$ são respectivamente a imagem original e os coeficientes transformados na forma vetorial e $\mathbf{T}^{*\mathbf{T}}$ é a matriz de transformação $N^2 x\ N^2$.

Se essa matriz de transformação puder ser escrita como o produto de Kronecker de duas matrizes *NxN*, então essa transformação é dita separável. Ou seja [4]:

$$\mathbf{C} = \mathbf{T}^{*T}\cdot\mathbf{F} \;\Rightarrow\; \mathbf{C} = \left(\boldsymbol{T}^{*T}\otimes\boldsymbol{T}^{*T}\right)\cdot\mathbf{F} \;\Rightarrow\; \boldsymbol{C} = \boldsymbol{T}^{*T}\boldsymbol{F}\boldsymbol{T}^{*} \tag{2.26}$$

onde $\boldsymbol{C}$ e $\boldsymbol{F}$ são respectivamente a imagem original e a transformada na forma matricial; $\boldsymbol{T}$ é a matriz de transformação inversa unidimensional e $\left(\boldsymbol{T}^{*T}\otimes\boldsymbol{T}^{*T}\right)$ é o produto de Kronecker da matriz $\boldsymbol{T}^{*T}$ com ela mesma .

Dessa forma, para uma transformada bidimensional ortonormal e separável, as equações da transformação direta e inversa ficam [4]:

$$C_{pq} = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} t_{pj}^{*}\cdot F_{jk}\cdot t_{qk}^{*} \qquad \boldsymbol{C} = \boldsymbol{T}^{*T}\boldsymbol{F}\,\boldsymbol{T}^{*} \tag{2.27}$$

$$F_{jk} = \sum_{p=1}^{N-1}\sum_{q=1}^{N-1} t_{pj}\cdot C_{pq}\cdot t_{qk} \qquad \boldsymbol{F} = \boldsymbol{T}\,\boldsymbol{C}\,\boldsymbol{T}^{T} \tag{2.28}$$

Imagens Base:

Assim como um sinal unidimensional pode ser representado por séries ortogonais de vetores base, uma imagem também pode ser expandida em termos de um conjunto discreto de *arrays* base, comumente chamados imagens base.

Assim, por exemplo, uma transformação inversa 2x2 separável pode ser representada a partir da equação (2.28) por [3]:

$$\boldsymbol{T}.\boldsymbol{C}.\boldsymbol{T}^{\boldsymbol{T}} = C_{00}\cdot\overbrace{\begin{bmatrix} t_{00}t_{00} & t_{00}t_{01} \\ t_{01}t_{00} & t_{01}t_{01}\end{bmatrix}}^{M_{00}} + C_{01}\cdot\overbrace{\begin{bmatrix} t_{00}t_{10} & t_{00}t_{11} \\ t_{01}t_{10} & t_{01}t_{11}\end{bmatrix}}^{M_{01}} + C_{10}\cdot\underbrace{\begin{bmatrix} t_{10}t_{00} & t_{10}t_{01} \\ t_{11}t_{00} & t_{11}t_{01}\end{bmatrix}}_{M_{10}} + C_{11}\cdot\underbrace{\begin{bmatrix} t_{10}t_{10} & t_{10}t_{11} \\ t_{11}t_{10} & t_{11}t_{11}\end{bmatrix}}_{M_{11}} \tag{2.29}$$

$\boldsymbol{M_{pq}}$, $0 \le p,q \le 1$ são as *imagens base*, que ponderadas apropriadamente pelos coeficientes transformados e somadas, recompõem a imagem original.

Para uma imagem de dimensão N x N, tem-se então N^2 imagens base.

Relação das imagens base com os vetores base:

Continuando o desenvolvimento anterior, tem-se que cada uma dessas imagens base pode ser escrita como o produto externo dos vetores base correspondentes [3]:

$$\begin{bmatrix} t_{00} \\ t_{01}\end{bmatrix}\begin{bmatrix} t_{00} & t_{01}\end{bmatrix} = \begin{bmatrix} t_{00}t_{00} & t_{00}t_{01} \\ t_{01}t_{00} & t_{01}t_{01}\end{bmatrix} = \boldsymbol{M}_{00} \tag{2.30}$$

$$\begin{bmatrix} t_{00} \\ t_{01}\end{bmatrix}\begin{bmatrix} t_{10} & t_{11}\end{bmatrix} = \begin{bmatrix} t_{00}t_{10} & t_{00}t_{11} \\ t_{01}t_{10} & t_{01}t_{11}\end{bmatrix} = \boldsymbol{M}_{01}\text{, e assim por diante.} \tag{2.31}$$

Dessa forma, denotando-se um vetor base por : $\boldsymbol{t_a} = [\ t_{a0}\ \ t_{a1}\ \ t_{a2}\ \ldots\ t_{a(N-1)}\ ]^T$, $a=0,..N-1$, pode-se escrever uma imagem $\boldsymbol{F}$ não só em termos de imagens base, mas também de vetores base, (antes utilizados para 1D somente). Essa equação fica [3]:

$$F = \sum_{p=0}^{N-1}\sum_{q=0}^{N-1} C_{pq}\cdot\underbrace{\boldsymbol{t_a}\cdot\boldsymbol{t_b^T}}_{M_{ab}} \tag{2.32}$$

Na Fig. 2.3 são apresentadas as imagens base da Transformada Walsh-Hadamard 4x4:

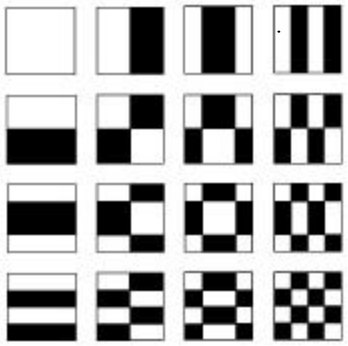


Fig. 2.3 – Conjunto de imagens base para WHT 4x4. [3]

2.3 – Eficiência da Transformada: Descorrelação e Empacotamento de Energia:

Ao se aplicar a transformada 1D ou 2D em um conjunto de dados que se deseja comprimir, tem-se por objetivo principal obter, no domínio transformado, uma distribuição de energia entre os coeficientes que propicie uma maior taxa de compressão com poucas perdas.

Através dos conceitos de Teoria de Informação [4], tendo-se, por exemplo, um vetor com N elementos, a informação média (entropia) dos elementos desse vetor é máxima quando se tem uma distribuição uniforme de energia entre eles e quanto mais concentrada a distribuição de energia, menor a entropia. O conceito de entropia e a sua expressão estão expostas de forma mais completa no Capítulo 3.

Sabe-se, no entanto, que a representação da imagem no domínio do espaço é altamente correlata, ou seja, um elemento qualquer da imagem tem, em média, um valor muito próximo ao valor dos elementos da sua vizinhança. Dessa forma, a distribuição de energia destes elementos é aproximadamente uniforme, aproximando-se do pior caso de compressão descrito acima.

Assim, a aplicação da transformada em um conjunto de dados tem por objetivo ideal a representação desses dados em um conjunto de coeficientes incorrelatos e com uma distribuição de energia mais concentrada possível.

Definindo-se um vetor de dados $\boldsymbol{f}$ de comprimento N estacionário no sentido amplo [3] e de média nula, uma transformação ortonormal $\boldsymbol{T}$ como na equação (2.16), denotando $\boldsymbol{T}^{*T}=\boldsymbol{T'}$, pode-se calcular a covariância do vetor de coeficientes transformados $\boldsymbol{c}$ como:

$$COV(\boldsymbol{c}) = E\left\{(\boldsymbol{c}-\bar{\boldsymbol{c}})\cdot(\boldsymbol{c}-\bar{\boldsymbol{c}})^T\right\} = E\left\{\boldsymbol{c}\cdot\boldsymbol{c}^T\right\} - \bar{\boldsymbol{c}}\cdot\bar{\boldsymbol{c}}^T \tag{2.33}$$

onde $\bar{\boldsymbol{c}}$ é o vetor de médias do processo $\boldsymbol{c}$. Combinando (2.16) e (2.33), tem-se:

$$COV(\boldsymbol{c}) = E\left\{\boldsymbol{T'}\,\boldsymbol{f}\cdot\boldsymbol{f}^T\boldsymbol{T'}^T\right\} - \boldsymbol{T'}\,\bar{\boldsymbol{f}}\cdot\bar{\boldsymbol{f}}^T\boldsymbol{T'}^T = \boldsymbol{T'}\left(E\left\{\boldsymbol{f}\cdot\boldsymbol{f}^T\right\} - \bar{\boldsymbol{f}}\cdot\bar{\boldsymbol{f}}^T\right)\boldsymbol{T'}^T$$

$$COV(\boldsymbol{c}) = \boldsymbol{T'}\cdot COV(\boldsymbol{f})\cdot\boldsymbol{T'}^T \tag{2.34}$$

A eficiência de descorrelação da transformada é medida pela capacidade de eliminar a correlação cruzada entre elementos do vetor, ou seja, anular os elementos fora da diagonal da matriz de covariâncias. Pode-se, então, definir a eficiência de descorrelação de uma transformada pela expressão [3]:

$$\eta_C = 1 - \left(\sum Y \Big/ \sum X\right) \tag{2.35}$$

onde :

$$\sum Y = \sum_{j=0}^{N-1} \sum_{\substack{k=0 \\ k \neq j}}^{N-1} \left| COV(\boldsymbol{c})_{jk} \right| \Rightarrow \text{Soma dos módulos das covariâncias cruzadas do vetor } \boldsymbol{c} \tag{2.36}$$

$$\sum X = \sum_{j=0}^{N-1} \sum_{\substack{k=0 \\ k \neq j}}^{N-1} \left| COV(\boldsymbol{f})_{jk} \right| \Rightarrow \text{Soma dos módulos das covariâncias cruzadas do vetor } \boldsymbol{f} \tag{2.37}$$

A outra característica desejável em uma transformada é a capacidade de empacotamento da energia dos sinal. Como $\boldsymbol{f}$ tem média nula (por hipótese), $\boldsymbol{c}$ também terá media nula e a sua energia é a dada pelas variâncias dos seus coeficientes. Pode-se então medir o empacotamento de energia através da energia relativa dos M coeficientes de maior variância em relação ao total das variâncias. Matematicamente, pode-se escrever [3]:

$$\eta_E = \frac{\sum_{j=0}^{M-1} COV(\boldsymbol{c})_{jj}}{\sum_{j=0}^{N-1} COV(\boldsymbol{c})_{jj}} \tag{2.38}$$

Dessa forma, pode-se afirmar que a transformada ideal é a transformada que apresenta $\eta_C=1$ e a maior compactação de energia possível, ou seja, que a matriz de covariâncias do domínio transformado seja uma matriz diagonal e que tenha poucos coeficientes com muita energia (alta variância) e muitos coeficientes com baixa variância. Como mostrado em [3], a transformada que diagonaliza a matriz de covariâncias de um vetor $\boldsymbol{f}$ é a transformada cujos vetores base são os *autovetores* desta matriz de covariância e, por conseqüência, os elementos da diagonal da matriz de covariâncias transformada são os *autovalores* da matriz original. Além disso, aplicando-se um critério de média quadrática (*Mean Square Criterion*), esta é a transformação que apresenta um desempenho ótimo no sentido de empacotamento de energia [3]. Esta transformada em Processamento Digital de Sinais é chamada de *Karhunen-Loève Transform* (KLT).

Apesar de apresentar um desempenho ótimo com relação aos dois critérios (descorrelação e empacotamento de energia,) a transformada KLT é de pouca utilização em compressão. Isso se dá porque seus vetores base são dependentes dos dados de entrada e, portanto, variam à medida que os dados de entrada se modificam, o que requer que esses vetores sejam transmitidos juntamente com o sinal para possibilitar a decodificação. Além disso, a determinação da matriz de covariância dos

dados de entrada, dos seus autovalores e autovetores apresenta grande demanda computacional. No caso de imagens, essa demanda computacional é ainda maior pois requer a determinação em separado do modelo para as linhas e para as colunas para que seja mantida a otimização do processo.

Dessa forma, a KLT é utilizada, geralmente, apenas como parâmetro ótimo de comparação para o desempenho das transformadas de vetores-base fixos como a DFT (*Discrete Fourier Transform*) e a DCT (*Discrete Cosine Transform*). A seguir apresenta-se um resumo dessas duas transformadas por serem de grande importância no processamento digital de sinais e na compressão de imagens, respectivamente.

2.4 – Transformada Discreta de Fourier (DFT)

A DFT é uma transformada discreta muito utilizada em Processamento Digital de Sinais. Por ser originada a partir da Transformada de Fourier, a DFT é de importância fundamental para o Processamento Digital de Sinais como um todo.

A utilização em compressão de dados da DFT apresenta a desvantagem de seus vetores base serem compostos por números complexos, o que representa maiores dificuldades com relação ao armazenamento e à manipulação. No entanto, a principal transformada discreta utilizada em compressão de imagens, a DCT, foi originada a partir da DFT, o que justifica o seu detalhamento neste trabalho.

Por se tratar de uma transformada bem conhecida e utilizada em operações de análise espectral e filtragem digital, a literatura sobre suas implementações rápidas é muito extensa. Para um sinal unidimensional $\boldsymbol{f}$ de comprimento N, suas equações de transformação direta e inversa são dadas por [2][3][4]:

$$c_p = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \cdot W_N^{pk} \quad 0 \le p \le N-1 \tag{2.39}$$

$$f_k = \frac{1}{\sqrt{N}} \sum_{p=0}^{N-1} c_p \cdot W_N^{-pk} \quad 0 \le k \le N-1 \tag{2.40}$$

onde:

$$W_N = exp\left(\frac{-j2\pi}{N} \right) \tag{2.41}$$

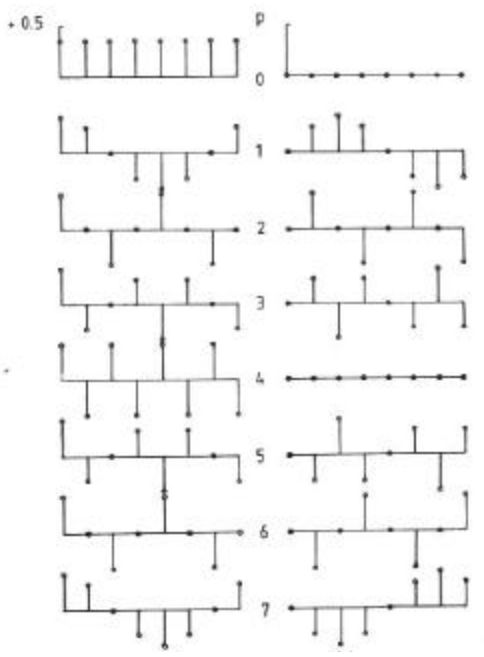
Cabe aqui explicar que as expressões (2.39) e (2.40) são chamadas de DFT unitária (por causa da normalização $1/\sqrt{N}$). Por se tratar de uma transformada complexa e simétrica, o par de transformadas na forma matricial, para a DFT, é definida por [4]:

$$\textit{Direta:} \quad \boldsymbol{c} = \boldsymbol{T}^{*}.\boldsymbol{f} \qquad \textit{Inversa:} \quad \boldsymbol{f} = \boldsymbol{T}.\boldsymbol{c} \tag{2.42}$$

onde:

$$\boldsymbol{T} = \left[\frac{1}{\sqrt{N}} W_N^{-pk} \right], \; 0 \le p,k \le N-1 \tag{2.43}$$

Na Fig. 2.4 abaixo são apresentadas na forma matricial e gráfica as partes real e imaginária da matriz de transformação para a DFT com N=8.



(a)

DFT(R) =

0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536
0.3536	0.2500	0.0000	−0.2500	−0.3536	−0.2500	0.0000	0.2500
0.3536	0.0000	−0.3536	0.0000	0.3536	0.0000	−0.3536	0.0000
0.3536	−0.2500	0.0000	0.2500	−0.3536	0.2500	0.0000	−0.2500
0.3536	−0.3536	0.3536	−0.3536	0.3536	−0.3536	0.3536	−0.3536
0.3536	−0.2500	0.0000	0.2500	−0.3536	0.2500	0.0000	−0.2500
0.3536	0.0000	−0.3536	0.0000	0.3536	0.0000	−0.3536	0.0000
0.3536	0.2500	0.0000	−0.2500	−0.3536	−0.2500	0.0000	0.2500

DFT(I) =

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.2500	0.3536	0.2500	0.0000	−0.2500	−0.3536	−0.2500
0.0000	0.3536	0.0000	−0.3536	0.0000	0.3536	0.0000	−0.3536
0.0000	0.2500	−0.3536	0.2500	0.0000	−0.2500	0.3536	−0.2500
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	−0.2500	0.3536	−0.2500	0.0000	0.2500	−0.3536	0.2500
0.0000	−0.3536	0.0000	0.3536	0.0000	−0.3536	0.0000	0.3536
0.0000	−0.2500	−0.3536	−0.2500	0.0000	0.2500	0.3536	0.2500

(b)

Fig. 2.4 – Partes real e imaginária para a DFT com N=8 pontos:

(a) Forma gráfica; (b) Forma matricial. [3]

Transformada DFT bidimensional

Por ser baseada em exponenciais, a DFT 2D é separável (pode ser decomposta em um produto de Kronecker). Assim, a expressão do par de transformadas 2D para a DFT unitária fica:

$$C_{pq} = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} F_{jk} \cdot W_N^{pj} \cdot W_N^{qk}, \quad 0 \le p,q \le N-1 \qquad \textit{Forma matricial: } \boldsymbol{C} = \boldsymbol{T}^* \cdot \boldsymbol{F} \cdot \boldsymbol{T}^* \quad (2.44)$$

$$F_{jk} = \frac{1}{N}\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} C_{pq} \cdot W_N^{-pj} \cdot W_N^{-qk}, \quad 0 \le j,k \le N-1 \qquad \textit{Forma matricial: } \boldsymbol{F} = \boldsymbol{T} \cdot \boldsymbol{C} \cdot \boldsymbol{T} \quad (2.45)$$

onde $\boldsymbol{W_N}$ é dado na equação (2.41) e $\boldsymbol{T}$ é dado em (2.43).

Na Fig. (2.5) abaixo são mostrados exemplos da aplicação da DFT 2D em imagens típicas. É interessante notar a simetria da magnitude em todas as imagens (em destaque no primeiro exemplo). Essa simetria se deve ao fato de que as intensidades dos pixels são números reais.

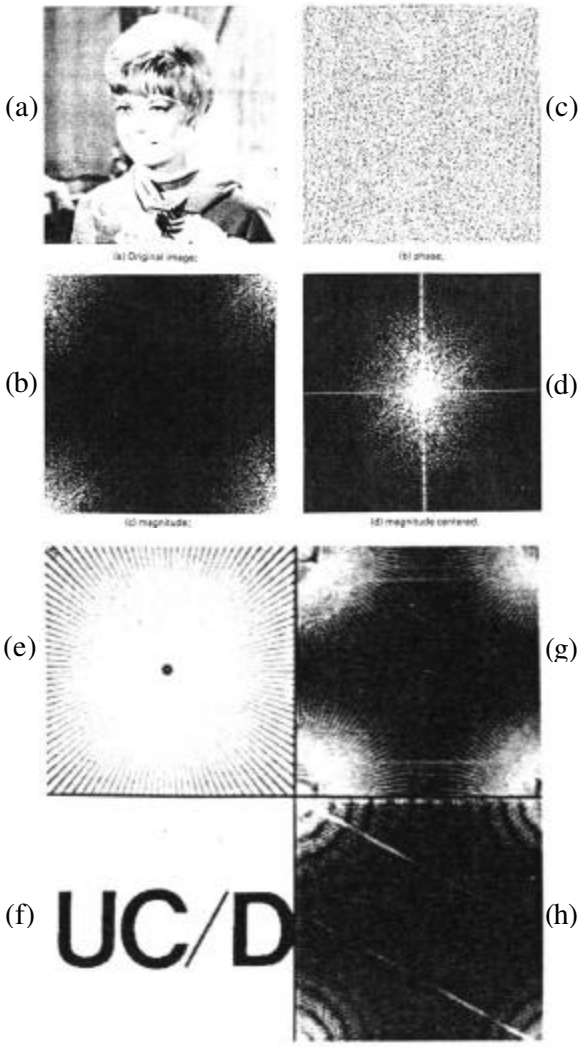


Fig. 2.5 – Exemplos da aplicação da DFT 2D em imagens: (a,e,f) Imagens originais; (b,g,h) Magnitudes; (c) Fase; (d) Magnitude centrada. [3]

2.5 – Transformada Discreta Cosseno (DCT)

A transformada DCT por apresentar uma desempenho sub-ótimo muito próximo à KLT para uma ampla gama de imagens e possuir a vantajosa característica de não trabalhar com números complexos, tem tido grande aplicação no campo do processamento de imagens desde sua introdução por Ahmed et al [5], e se constitui na base para todos os padrões atuais de compressão de imagem e vídeo [6][7].

Dada uma seqüência $\boldsymbol{f}$ de comprimento N, o par de transformadas DCT 1D, segundo [3][6] é definida por:

$$c_p = a_p \sum_{k=0}^{N-1} f_k \cdot cos\left[\frac{p\ (2k+1)\,p}{2N}\right] \qquad 0 \le p \le N\text{-}1 \tag{2.46}$$

$$f_k = \sum_{p=0}^{N-1} a_p . c_p \cdot cos\left[\frac{p\ (2k+1)\,p}{2N}\right] \qquad 0 \le k \le N\text{-}1 \tag{2.47}$$

onde:

$$\alpha_p = \begin{cases} \sqrt{\frac{1}{N}}, & para\ p = 0; \\ \sqrt{\frac{2}{N}}, & para\ 1 \le p \le N - 1 \end{cases} \tag{2.48}$$

Ao contrário da DFT, pode-se notar pelas equações acima que a DCT é real e assimétrica, assim, $\boldsymbol{T^{-1}=T^T}$, e o par de transformada na forma matricial pode ser expressa como:

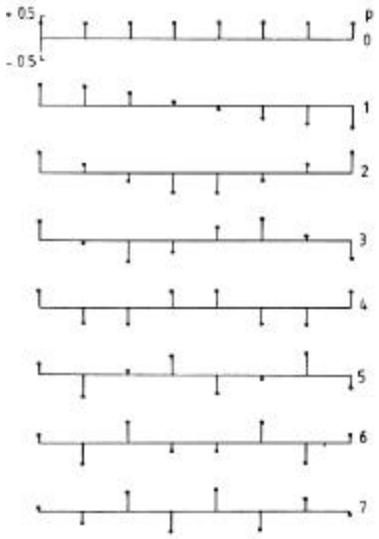
$$Direta: \quad \boldsymbol{c = T^T . f} \qquad Inversa: \boldsymbol{f = T . c} \tag{2.49}$$

onde:

$$\boldsymbol{T} = \left[\alpha_p \cdot cos\left(\frac{\pi(2k+1)p}{2N}\right)\right], \qquad 0 \le p,k \le N - 1 \tag{2.50}$$

A DCT utilizada na definição acima é chamada DCT par e foi escolhida por ser a mais utilizada. Contudo deve ser notado que existem outras versões para ela, com formas diferentes de implementação rápida também.

Nas figuras a seguir, encontra-se a matriz de transformação DCT direta para N=8, bem como os vetores base:



$$DCT(8) = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}$$

Fig. 2.6 – Vetores-base para a DCT 1D com N=8: (a) Forma gráfica; (b) Forma matricial.[3]

Note que o primeiro vetor-base (p=0) é constante, ou seja, tem freqüência nula. Dessa forma, o coeficiente formado a partir da sua projeção é proporcional à média do vetor de dados e porisso, é conhecido como coeficiente DC (*Direct Current* – Corrente Contínua). Todos os outros coeficientes são referenciados como coeficientes AC (*Alternate Current* – Corrente Alternada).

Relação entre a DFT e a DCT

A DFT de uma seqüência qualquer de comprimento finito $x[n]$ é, por definição, a transformada de Fourier da seqüência periódica $\tilde{x}[n]$ definida como:

$$\tilde{x}[n] = \sum_{k=-\infty}^{+\infty} x[kN+n] \tag{2.51}$$

sendo N o comprimento da seqüência $x[n]$.

Analisando a seqüência $\tilde{x}[n]$ pode-se observar que nos pontos de junção entre dois períodos consecutivos pode ocorrer um ponto de quebra da seqüência. Isso ocorre porque a primeira amostra da seqüência $x[n]$ é, genericamente, diferente da última amostra. Esse ponto de quebra, no domínio espectral, é representado por um aumento no valor dos coeficientes de alta freqüência.

Porém se o sinal $x[n]$ for muito correlato, apresentará, no domínio espectral, uma alta concentração de energia nas baixas freqüências. Assim, ao se aplicar a DFT em um sinal correlato

ocorrerá uma divisão da energia total entre as baixas e as altas freqüências. Do ponto de vista de compressão, este efeito é indesejável porque aumenta a quantidade média de bits necessária para se codificar os coeficientes transformados.

Assim, para se evitar esse problema Ahmed e Rao [5] propuseram a realização de uma expansão simétrica no lugar da simples justaposição dos períodos *x[n]*. Assim, definiu-se $\tilde{y}[n]$ de período 2N tal que:

$$\tilde{y}[n] = \sum_{k=-\infty}^{+\infty} x[2kN+n] + x[2N(k+1)-1-n] \tag{2.52}$$

Dessa forma, elimina-se a quebra entre períodos da seqüência $\tilde{y}[n]$ reduzindo, para sinais correlatos, o espalhamento dos coeficientes transformados. Outro ponto de grande interesse é que, por $\tilde{y}[n]$ ser real e par, sua Transformada de Fourier (TF) é também real e par. Essa característica é interessante pois possibilita que a TF do período (*2N* pontos) de $\tilde{y}[n]$ possa ser completamente representado por apenas N coeficientes reais. Uma explicação detalhada deste processo pode ser encontrada em [4][8]. Essa relação não é utilizada nas implementações práticas pois requer a manipulação de dados complexos (devido à DFT), ao contrário da implementação direta que trabalha apenas com números reais.

Transformada DCT 2D

A transformação DCT 2D é separável. Assim, a DCT 2D de uma imagem pode ser definida como:

$$C_{pq} = \alpha_p \cdot \alpha_q \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} F_{jk} \cos\left[\frac{\pi(2j+1)p}{2N}\right]\cos\left[\frac{\pi(2k+1)q}{2N}\right] \quad Forma\ Matricial: \boldsymbol{C} = \boldsymbol{T}^{\boldsymbol{T}} \cdot \boldsymbol{F} \cdot \boldsymbol{T} \tag{2.53}$$

$$0 \le p,q \le N-1$$

$$F_{jk} = \sum_{p=0}^{N-1}\sum_{q=0}^{N-1} \alpha_p \cdot \alpha_q \cdot C_{pq} \cos\left[\frac{\pi(2j+1)p}{2N}\right]\cos\left[\frac{\pi(2k+1)q}{2N}\right] \quad Forma\ matricial: \boldsymbol{F} = \boldsymbol{T} \cdot \boldsymbol{C} \cdot \boldsymbol{T}^{\boldsymbol{T}} \tag{2.54}$$

$$0 \le j,k \le N-1$$

onde α_p é dado em (2.48) e $\boldsymbol{T}$ é dado em (2.50).

Analogamente ao caso unidimensional, para a DCT 2D, o coeficiente DC é o coeficiente associado à imagem-base constante (freqüências vertical e horizontal nulas).

Na Fig. 2.7 encontram-se alguns exemplos de aplicação da DCT. Note que os coeficientes transformados na imagem (b) tem uma representação espectral bem mais compacta comparada à DFT (Fig. 2.6), o que representa uma compactação da energia total da imagem em um número menor de coeficientes.

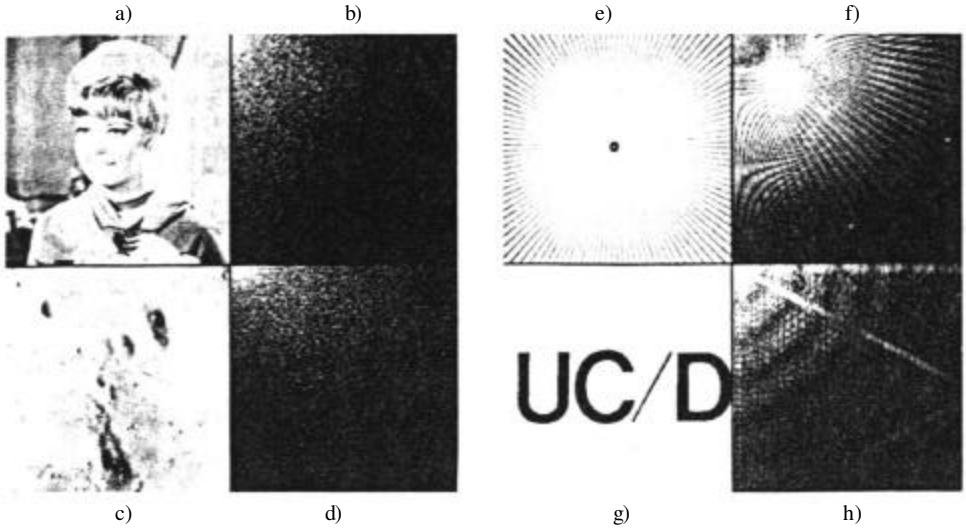


Fig. 2.7 – Exemplo da aplicação de DCT em imagens típicas.[3]

2.6 – Comentários

Neste Capítulo apresentou-se um resumo breve a respeitos de Transformadas Ortonormais e suas propriedades, enfatizando as de maior relevância para a aplicação na área de Compressão Digital de Imagens. Também foram mostradas as transformadas DFT (*Discrete Fourier Transform*) e DCT (*Discrete Cosine Transform*), sendo que esta última constitui a base de todos os padrões de compressão de imagens utilizados hoje em dia.

Continuando o processo de codificação por transformada, há ainda duas etapas de grande importância: a Quantização e a Codificação Entrópica onde são exploradas as características vantajosas de compressão de energia e descorrelação geradas pela aplicação da transformada. Essas etapas serão melhor descritas no Capítulo 3 a seguir, juntamente com uma visão global do processo de compressão.

Capítulo 3

Codificação por Transformada

Dando continuidade ao estudo do processo de codificação por transformada, parte-se agora para os métodos de Quantização e Codificação Entrópica. Segundo Clarke [3], as etapas de quantização e codificação são mais importantes que a escolha do tipo de transformada e do que o tamanho do bloco, pois a escolha de uma quantização e de um código eficientes produzirão resultados muito mais expressivos do que um trabalho extensivo com relação às transformadas.

Além desses processos, neste capítulo é ainda apresentada uma explanação a respeito do principal artefato de reconstrução dos processos de codificação por transformada: o Efeito de Bloco, suas origens e comentários sobre alguns métodos clássicos para sua redução.

3.1 – Introdução

A operação básica na codificação de sinais digitais é a quantização: processo de converter valores analógicos (ou com grande número de níveis) para níveis discretos (mais esparsos). Quanto mais níveis discretos forem associados à saída do quantizador, menor o erro em se aproximar a variável original, mas também maior o número de bits necessário para codificar o sinal quantizado. A técnica de codificação mais simples é o *Pulse Code Modulation* (PCM), na qual se amostra periodicamente uma forma de onda contínua e quantiza-se cada amostra com um número pré-estabelecido de bits. No logPCM, que é largamente usado em telefonia, os níveis discretos de reconstrução na saída do quantizador não são linearmente distribuídos sobre a faixa dinâmica do sinal. Isso permite uma melhor reconstrução de sinais não estacionários como os de voz.

Mas à medida que o sinal de entrada se torna mais correlacionado, o PCM deixa de ser um sistema de codificação efetivo. Isso porque como a redundância do sinal aumenta, a Informação contida no sinal cai e, portanto, são necessários menos bits (em média) para se transmitir cada amostra do sinal, o que leva a um código de comprimento variável (que não é suportado pelo PCM).

Entre as técnicas que conseguem realizar a codificação em níveis mais próximos ao limiar (sem perda) estão os códigos preditivos, codificação por subbanda e codificação por transformada. Dado o objetivo do trabalho, está-se dando ênfase à codificação por transformada.

A codificação por transformada foi desenvolvida originalmente com o objetivo de extrair a redundância existente entre elementos de um vetor aleatório. Essa correlação pode existir ou em um sistema multicanal com acoplamento cruzado entre os canais ou quando um vetor aleatório é formado por um bloco de amostras consecutivas de um sinal escalar . A idéia básica é quantizar o vetor transformado ao invés de quantizar o vetor original, como mostrado na Fig. 3.1.

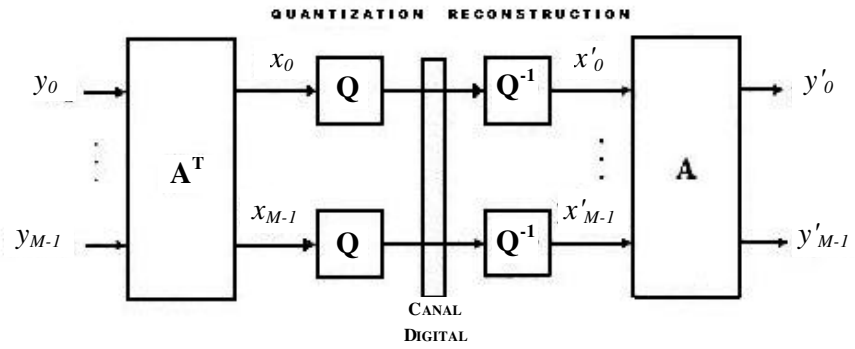


Fig. 3.1 – Diagrama em bloco de um codificador por transformada. [9]

3.2 - Quantização

O processo de quantização pode ser feito para cada coeficiente em separado (chamada quantização escalar) ou para grupos de coeficientes (quantização vetorial). Apesar do melhor desempenho apresentado pela quantização vetorial, a velocidade e a simplicidade de implementação fazem da quantização escalar a mais utilizada em implementações práticas [8].

Num codificador por transformada, os dados de entrada estão, em geral, representados em 8 bits, isto é, são inteiros entre 0 e 255. Quando se aplica a transformada DCT (usada em todos os padrões de compressão), os coeficientes resultantes podem apresentar uma infinidade de valores distintos dentro de uma faixa limitada. Para representá-los num número finito de bits faz-se necessária a quantização. Como os dois padrões utilizados como base para esta pesquisa, o JPEG e o MPEG-2 utilizam quantização escalar, descreve-se esse processo com mais detalhes a seguir.

Quantização Escalar de um Coeficiente

Seja x um coeficiente genérico na entrada de um quantizador escalar que deve ser representado por um dos L níveis de quantização possíveis.

Então a saída do quantizador $\hat{x}$ pode ser expresso por:

$$\hat{x} = Q(x) = r_i, \qquad d_i < x \leq d_{i+1} \tag{3.1}$$

onde:

- $Q(\cdot)$ representa a operação de quantização;
- r_i para $1 \le i \le L$ representa os L níveis de quantização;
- d_i para $1 \le i \le L+1$ representa os $L+1$ níveis de decisão;

A Fig. 3.2 ilustra o gráfico da função de entrada-saída de um quantizador escalar genérico de L níveis com passo central (existência do nível L=0) . Note que se x assumir qualquer valor entre d_i e d_{i+1}, será mapeado para o nível discreto r_i em $\hat{x}$.

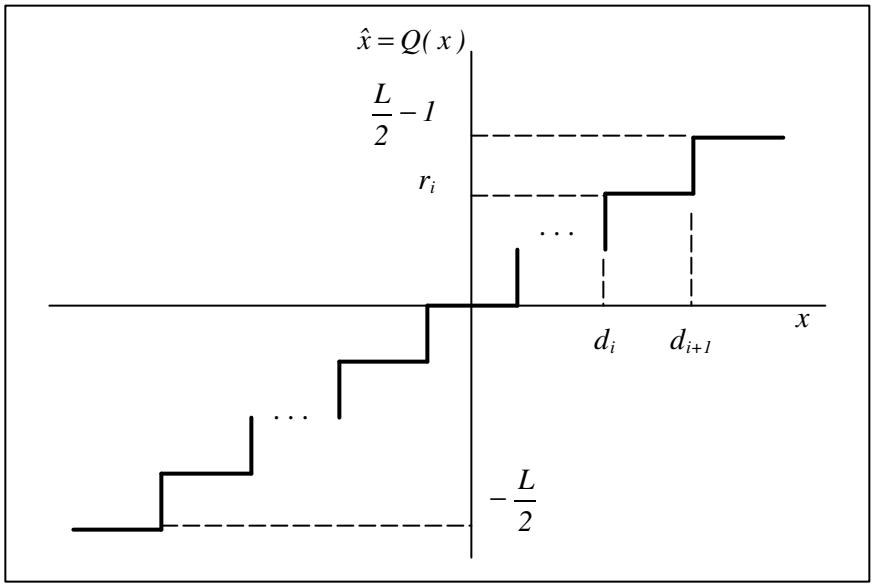


Fig. 3.2 – Função entrada-saída do quantizador escalar $\hat{x} = Q(x)$ de L níveis com passo central.

Em geral a aproximação $\hat{x}$ é transmitida ou armazenada na forma binária. Se $L=2^R$, então a taxa de bits gerada é definida por [8]:

$$R=\log_2 L \quad bits/nível \tag{3.2}$$

E assim, o erro de quantização pode ser expresso pela diferença:

$$\varepsilon = x - \hat{x} \tag{3.3}$$

Supondo que a variável de entrada x do quantizador é uma variável aleatória com média nula e função densidade de probabilidade $p_x(x)$, podemos expressar sua variância σ_x^2 por:

$$\sigma_x^2 = E[x^2] - E[x]^2 = E[x^2], \quad pois\ E[x]=0. \tag{3.4}$$

Assim, o erro de quantização também é uma variável aleatória cuja variância σ_ε^2 pode ser expressa por [8]:

$$\sigma_\varepsilon^2 = E[\varepsilon^2] = \int_{-\infty}^{+\infty} \varepsilon^2 \cdot p_\varepsilon(\varepsilon)\, d\varepsilon, \quad ou \tag{3.5}$$

$$\sigma_\varepsilon^2 = E[\varepsilon^2] = \int_{-\infty}^{+\infty} (x - \hat{x})^2 \cdot p_x(x)\, dx \tag{3.6}$$

Como $\hat{x}$ é dividido em L níveis de reconstrução, a integral em (3.6) é dividida em L intervalos diferentes, sendo escrita da forma:

$$\sigma_\varepsilon^2 = E[\varepsilon^2] = \sum_{i=1}^{L} \int_{d_i}^{d_{i+1}} (x - r_i)^2 \cdot p_x(x)\, dx \tag{3.7}$$

Esta variância do erro de quantização é um dos parâmetros mais importantes para a avaliação do desempenho dos quantizadores e é dado, em geral, pela expressão (3.7). Há vários métodos para se determinar a função entrada-saída de um quantizador, a mais simples é o quantizador linear, no qual todos os níveis de reconstrução são igualmente espaçados, ou seja:

$$d_{i+1} - d_i = \Delta, \quad e \quad r_i = \frac{d_{i+1} + d_i}{2\Delta}, \qquad 1 \le i \le L \tag{3.8}$$

onde Δ representa o passo de quantização (espaço entre dois níveis de reconstrução consecutivos). Supondo que a amplitude dos dados de entrada sejam limitadas ao intervalo entre $+x_m$ e $-x_m$, e que o quantizador linear tenha $L=2^R$ níveis, pode-se expressar Δ por [8]:

$$\Delta = \frac{2x_m}{L} = \frac{2x_m}{2^R} \tag{3.9}$$

e o erro de quantização fica então limitado a faixa:

$$-\frac{\Delta}{2} \le \varepsilon \le \frac{\Delta}{2} \tag{3.10}$$

Supondo que o erro de quantização seja uniformemente distribuído neste intervalo e utilizando-se a equação (3.5), pode-se determinar que a variância do erro do quantizador linear é dado por:

$$\sigma_\varepsilon^2 = \frac{\Delta^2}{12} = \frac{1}{3} \cdot x_m^2 \cdot 2^{-2R} \tag{3.11}$$

Dessa forma, pode-se observar que a variância do erro de quantização para um quantizador linear cresce proporcionalmente ao quadrado do passo de quantização ou, alternativamente, decresce exponencialmente com a taxa de bits. Assim, para se projetar um quantizador linear basta determinar a taxa de bits ou o passo de quantização.

No âmbito da codificação por transformada baseada em blocos, a determinação do quantizador se resume, então, à determinação do número de bits (ou do passo de quantização) para cada um dos coeficientes transformados. A matriz que contém o número de bits ou o passo de quantização para cada coeficiente do bloco é referenciada como Matriz de Quantização. A título de exemplo, apresenta-se a seguir um método de alocação de bits baseado na minimização da variância do erro de quantização dos coeficientes transformados dada uma taxa média de bits por amostra. Para exemplificar os sistemas de alocação de passo de quantização, serão apresentados no Capítulo 4 os sistemas usados nos padrões JPEG e MPEG-2 que são baseados na determinação do passo de quantização.

Algoritmo Básico de Alocação de Bits para Conjunto de Coeficientes

Considerando novamente a Fig. 3.1, substitui-se a variável aleatória *x* da seção passada pelo processo aleatório ***y*** que será representado por um vetor de dados de entrada.

Suponha-se, então que o vetor de entrada ***y*** seja formado por *N* amostras de um processo estacionário com média nula que tenha função de autocorrelação $\boldsymbol{R_{yy}(m)}$.

Assim, da teoria de sistema lineares [9], aplicando-se ao vetor ***y*** uma transformação ortonormal ***T***, a função de autocorrelação do vetor resultante ***x*** é dada por:

$$\boldsymbol{R}_{xx} = \boldsymbol{T}^{\boldsymbol{T}} \cdot \boldsymbol{R}_{yy} \cdot \boldsymbol{T} \tag{3.12}$$

Quando se quantiza cada coeficiente do vetor transformado, obtém-se uma aproximação de ***x*** referida como ***x'*** na Fig. 3.1. Assumindo que um quantizador linear ideal é usado, o erro médio quadrático (MSE – *Mean Square Error*) incorrido no k-ésimo coeficiente é dado por:

$$MSE(k) \overset{\Delta}{=} \mathrm{E}\left\{(\boldsymbol{x'_k} - \boldsymbol{x_k})^2\right\} = 2^{-2B_k} \cdot \mathrm{E}\left[\boldsymbol{x_k^2}\right] = 2^{-2B_k} \cdot \left[\boldsymbol{R}_{xx}\right]_k = 2^{-2B_k} \cdot \sigma_k^2 \tag{3.13}$$

onde B_k é o número de bits alocados para quantizar o k-ésimo coeficiente e $\sigma_{\boldsymbol{k}}^{\boldsymbol{2}}$ é a variância desse coeficiente.

Assim a MSE total ε é dada por [9]:

$$\varepsilon = \sum_{k=0}^{N-1} 2^{-2B_k} \cdot \sigma_k^2 \tag{3.14}$$

Neste ponto, alcança-se o problema clássico de alocação de bits. Desejando-se minimizar esse erro dado que se quer utilizar uma taxa média de *R* bits por nível, tem-se então um total de *N.R* bits disponíveis para cada bloco, ou seja:

$$\sum_{k=0}^{N-1} B_k = N \cdot R \tag{3.15}$$

Pode-se provar [9] que a minimização de (3.14) sujeita a restrição (3.15) leva a :

$$\sigma_k^2 \cdot 2^{-2B_k} = C \tag{3.16}$$

onde *C* é uma constante. Assim isolando-se B_k em (3.16), tem-se [9]:

$$B_k = \frac{1}{2}\left(log_2 \sigma_k^2 - log_2 C\right) \tag{3.17}$$

Aplicando-se agora esse resultado na restrição (3.15) :

$$2.N.R = \sum_{k=0}^{N-1}\left(log_2 \sigma_k^2 - log_2 C\right) = \sum_{k=0}^{N-1}\left(log_2 \sigma_k^2\right) - N.log_2 C$$

$$log_2 C = \frac{1}{N}\left\{\sum_{k=0}^{N-1}\left(log_2 \sigma_k^2\right) - 2N.R\right\} \tag{3.18}$$

Logo:

$$B_k = \frac{1}{2}\left(log_2 \sigma_k^2 - \frac{1}{N}\left\{\sum_{k=0}^{N-1} log_2 \sigma_k^2 - 2.N.R\right\}\right)$$

$$B_k = \frac{1}{2} log_2 \sigma_k^2 - \frac{1}{2N}\sum_{k=0}^{N-1} log_2 \sigma_k^2 + .R$$

$$B_k = R + \frac{1}{2}\left(log_2 \sigma_k^2 - \frac{1}{N}\sum_{k=0}^{N-1} log_2 \sigma_k^2 \right) \tag{3.19}$$

Como :

$$\sum_{k=0}^{N-1}\left(log_2\,\sigma_k^2\right)=log_2\left[\prod_{k-0}^{N-1}\sigma_k^2\right], \tag{3.20}$$

então a equação final apara a alocação de bits fica [9]:

$$B_k = R+\frac{1}{2}log_2\left(\frac{\sigma_k^2}{\left(\prod_{k-0}^{N-1}\sigma_k^2\right)^{1/N}}\right) \tag{3.21}$$

que é conhecida como regra de Log-Variância. Note que é possível, matematicamente, obter-se um número negativo de bits para um valor particular de k. Nesses casos é possível, sem perda de otimização, ajustar o número de bits para zero[9].

Também pode-se demonstrar substituindo-se (3.19) em (3.14) e denotando-se a média geométrica das variâncias do denominador dentro do logaritmo por s_{GM}^2 que [9]

$$\varepsilon_{mín} = N\cdot\sigma_{GM}^2\cdot 2^{-2R} \tag{3.22}$$

Codificando-se esse mesmo sinal em PCM sem a transformada, obter-se-ía o erro total:

$$\varepsilon_{PCM} = N\cdot\sigma_y^2\cdot 2^{-2R} \tag{3.23}$$

onde s_y^2 é a variância do sinal de entrada (suposto estacionário) e constituem os elementos da diagonal da matriz $\boldsymbol{R}_{yy}$.

Devido à característica da ortonormalidade imposta à transformada, podemos afirmar que o traço da matriz de covariância se mantém constante, ou seja :

$$\sigma_y^2 = \frac{1}{N}\sum_{k=0}^{N-1}\sigma_k^2 \tag{3.24}$$

Portanto pode-se deduzir a redução do MSE devido à codificação por transformada para uma taxa de bits fixa como sendo [9]:

$$G_{TC} = \frac{\varepsilon_{PCM}}{\varepsilon_{min}} = \frac{\sigma_y^2}{\sigma_{GM}^2} = \frac{\frac{1}{N}\sum_{k=0}^{N-1}\sigma_k^2}{\left(\prod_{k=0}^{N-1}\sigma_k^2\right)^{1/N}} \tag{3.25}$$

É interessante salientar aqui que apesar dessa técnica apresentar resultados bons em termos do ganho G_{TC} , existem outras técnicas mais elaboradas que oferecem resultados muito mais contundentes como o usado nos sistemas JPEG e MPEG que são, respectivamente, padrões de compactação de imagem e vídeo.

Outros algoritmos

Visto que para uma determinada transformada, sabe-se aproximadamente qual a estruturação da matriz de coeficientes transformados (baseado na direção do empacotamento de energia); alguns sistemas fazem uso de máscaras zonais fixas antes da alocação de bits e quantização.

Chamando de I_t, o conjunto de endereços das amostras transmitidas [10]:

$$I_t = \{(p,q); n_{p,q} \geq 1\} \tag{3.26}$$

e fazendo N_t o número de amostras transmitidas, pode-se definir uma máscara zonal fixa que vale 1 na zona das N_t variâncias das amostras transmitidas:

$$\mathrm{M}(p,q) = \begin{cases} 1 & (p,q) \in I \\ 0 & caso\ contrário \end{cases} \tag{3.27}$$

Assim, a alocação e quantização são feitas só nos elementos diferentes de zero para posterior transmissão ou armazenamento.

Terão que ser transmitidos então a informação quantizada e codificada e o padrão de alocação para cada imagem (supondo que não seja fixo), à semelhança dos elementos que recebem um número não nulo de bits pelo algoritmo. O receptor já conhecendo essa máscara fixa saberá quais os coeficientes desprezados, não havendo necessidade de sua transmissão.

Note que esse procedimento está sujeito à perdas de informação porque presume quais os coeficientes que podem ser desprezados baseados no efeito estatístico da transformação.

Se ao invés de se transmitir ou armazenar os N_t elementos de máxima variância, considerar-se as N_t amostras de máxima amplitude (ou seja haveria uma leitura dos dados para cada imagem ou bloco) no domínio transformado, tem-se o que se chama de *threshold coding*.

O conjunto de endereços das amostras transmitidas é então [10]:

$$I_t' = \{(p,q); |C_{pq}| > \eta\} \tag{3.28}$$

onde η é o limiar escolhido convenientemente para controlar a razão média de bits atingível.

Para uma dada classe de imagens semelhantes, devido às variâncias das amostras transformadas serem aproximadamente fixas, a máscara zonal não muda de uma imagem para outra (ou bloco a bloco) para uma dada razão de bits.

Entretanto, a máscara definida como:

$$M_\eta(p,q) = \begin{cases} 1 & (p,q) \in I' \\ 0 & caso\ contrário \end{cases} \tag{3.29}$$

poderia mudar de um bloco para outro porque o conjunto I_t' das amostras de maior amplitude não necessariamente é o mesmo para diferentes blocos. Assim, terão que ser transmitidos a informação quantizada e codificada, o padrão de alocação e a máscara em si.

Embora para o mesmo número de amostras transmitidas ou número de bits de quantização, a máscara de *threshold* providencie a melhor escolha para transmissão de amostras (i.e., menor distorção), pode também resultar numa razão de bits maior, já que os endereços das amostras transmitidas terão que ser codificados para cada bloco da imagem.

Um esquema alternativo na máscara adaptativa é dado se os coeficientes transformados 2D forem mapeados em um vetor arranjado numa ordem pré-determinada. Esse esquema resolve o problema da máscara adaptativa pois não há necessidade de se transmitir a máscara já que o decodificador sabe que daquele ponto para frente todos os coeficientes tem zero bits alocados. Um esquema semelhante a esse é usado no MPEG-2. O sistema de quantização e codificação do MPEG-2 será comentado mais profundamente no Capítulo 4.

3.3 – Códigos de Comprimento Variável (VLC)

A última das três etapas descritas na Fig. 2.1 é a codificação com comprimento variável. Como visto no seção anterior, para que suas características de compactação de energia sejam bem aproveitadas, a codificação por transformada precisa ser codificada com um número variável de bits por coeficiente baseado na energia que cada um carrega de forma a minimizar a perda de energia.

Com isso, cada coeficiente a ser transmitido/armazenado é quantizado uniformemente com um dado número de bits. Assim, cada nível recebe um código com esse número de bits,

independente da sua freqüência de ocorrência. Isso nos leva a pensar a respeito da aplicação de um esquema de codificação de comprimento variável para se aproveitar melhor as características predizíveis dos coeficientes transformados.

Neste ponto é interessante notar que as distribuições de probabilidade dos coeficientes transformados são diferentes entre si (pela própria diferença nas variâncias). Uma regra geral de classificação é apresentada em [11] e considera que, num esquema de codificação por transformada de imagens baseada em blocos, o coeficiente DC (canto superior esquerdo do bloco) é bem modelado pela distribuição uniforme e, por isso geralmente é codificado também uniformemente. Já os coeficientes AC (restante do bloco) apresentam uma concentração maior de ocorrência em torno do zero e decaindo com o afastamento, o que proporciona maiores possibilidades de compactação.

Codificação de Huffmann

A codificação de Huffmann é uma técnica baseada na medida de Entropia (originária da Teoria de Informação). A Entropia de um coeficiente representa a Informação média, em número de bits, que aquele coeficiente carrega. Em outras palavras, a Entropia é o mínimo número de bits médio no qual um conjunto de dados pode ser codificado sem que haja perda de Informação.

A proposta geral do código de Huffmann é associar aos eventos mais prováveis (de maior freqüência relativa) um número menor de bits e aos menos prováveis um número maior de bits. Assim, a imagem comprimida vai requerer menos bits no total para descrever a imagem original.

Os códigos de Huffmann são atribuídos através da criação de uma Árvore de Huffmann que coloca lado a lado os valores de brilho (que chamaremos símbolos) com suas respectivas freqüências de ocorrência. A Árvore de Huffmann assegura que os códigos mais longos sejam associados aos símbolos menos freqüentes e os menores aos mais freqüentes.

A Árvore começa sendo construída colocando-se em linha os símbolos e seus respectivos valores de freqüência ordenados por essa última. O segundo nível é calculado a partir do primeiro agrupando-se (somando) as freqüências dos dois símbolos menos prováveis e associando arbitrariamente a um deles o valor 0 e ao outro o valor 1.

O terceiro nível é calculado a partir do segundo agrupando-se novamente os dois valores de menor freqüência e associando-lhes os valores 0 e 1, e assim por diante até que se agrupem todos os símbolos.

A Fig. 3.3 mostra uma imagem de 400x300 pixels codificados originalmente em 3 bits por pixel e o respectivo histograma na qual se irá aplicar, a título de exemplo o código de Huffmann.

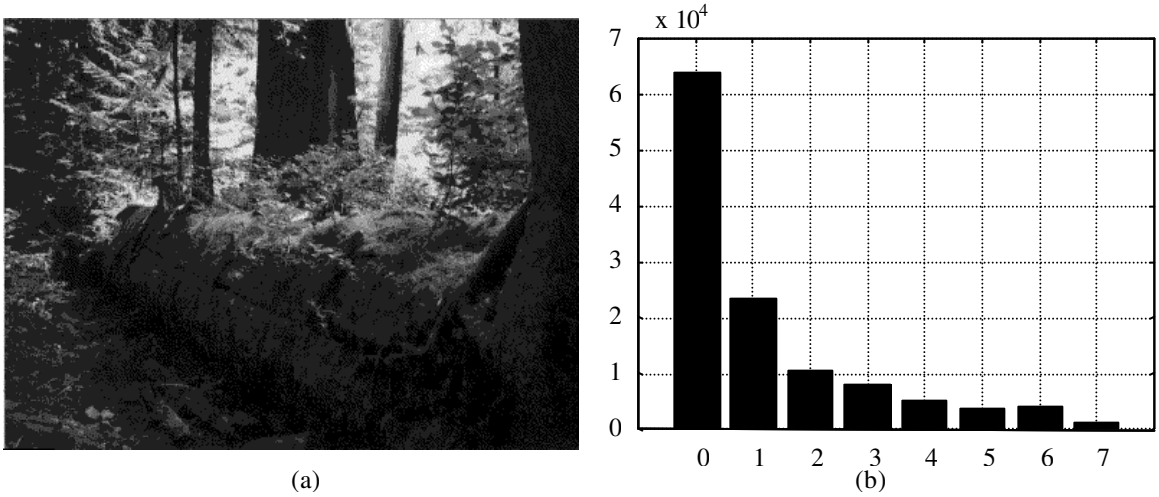


Fig. 3.3 – (a) Imagem Floresta; (b) Histograma da imagem Floresta.

O processo da Árvore feito para essa imagem é mostrado na Fig 3.4.

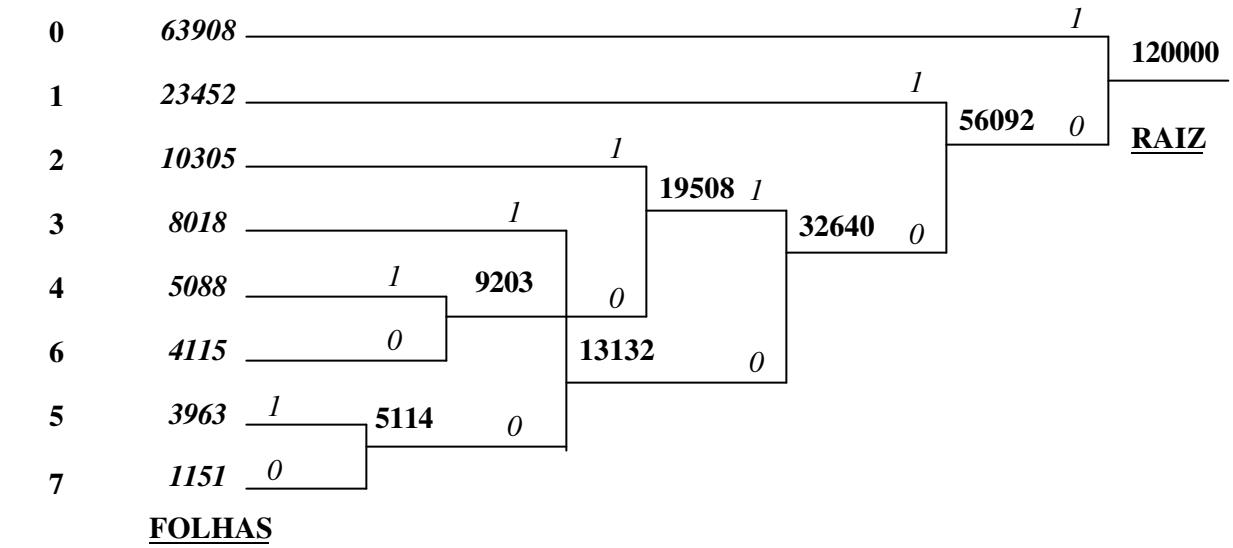


Fig. 3.4 – Árvore de Huffmann para a imagem Floresta.

Note que cada símbolo é uma "folha" da Árvore e que só há um caminho possível partindo da raiz (sem voltar níveis) que chega a cada folha. Assim, o Código de Huffmann para um determinado símbolo é extraído da Árvore partindo-se da Raiz até chegar à respectiva folha. Por exemplo, para codificar o símbolo '4' nós usaremos o código '00101'.

A tabela de codificação completa para este exemplo é:

Símbolo	0	1	2	3	4	5	6	7
Código	1	01	0011	0001	00101	00001	00100	00000
Nº de Bits	1	2	4	4	5	5	5	5
Freq. Relativa (Probabilidade)	0,5326	0,1954	0,0859	0,0668	0,0432	0,0322	0,0343	0,0096

TABELA III.1 – Código de Huffmann para a imagem Floresta.

Calculando-se agora o número de bits médio do código pela soma ponderada do comprimento do código pela sua freqüência relativa temos :

$$\overline{L} = \sum_{k=0}^{7} f_k . B_k = 2{,}1307 \; {}^{bits}\!/_{símbolo} \tag{3.30}$$

O que resultou em uma economia de praticamente 0,87 bits por símbolo sem nenhuma perda de informação. Assim, uma imagem original que contava originalmente com *400* x *300* x *3* = *360.000* ≅ *360 Kbits* pode ser armazenada/transmitida com praticamente *400* x *300* x *2,1307* = *255.684* ≅ *256* Kbits proporcionando uma taxa de compressão de (1,4:1).

Para determinar a eficiência do código de Huffmann proposto, comparando esse resultado com a Entropia *(H)* que representa o menor comprimento médio do código em bits/símbolo. Segundo a Teoria de Informação [3], para um dado conjunto de símbolos $S = \{S_0, S_1, \ldots, S_{N-1}\}$ associados às probabilidades $P=\{p_0, p_1, \ldots, p_{N-1}\}$, a Entropia é dada por:

$$H = \sum_{k=0}^{N-1} p_k \cdot log_2(p_k) \tag{3.31}$$

Utilizando-se (3.20) e a *TABELA III.1*, determina-se, para a imagem na Fig. 3.3, uma entropia de *H=2,0963 bits/símbolo*. Dessa forma, pode-se notar que o comprimento médio do código é muito próximo ao mínimo o que representa uma boa eficiência de compressão.

Também é importante notar aqui que ao número de 256 Kbits deve ser acrescido a tabela código que, neste caso como no caso geral, não provocará mudanças significativas no tamanho final dos dados.

Várias adaptações podem ser feitas nesta técnica para permitir maiores taxas de compressão como considerar a imagem divididas em regiões onde a redundância é maior e que, portanto, permitem maior compressão. É claro que para cada região diferente deve-se inserir a nova tabela de

codificação o que gera um *overhead* e portanto, deve haver um compromisso entre a exploração da redundância e o número de sub-regiões.

Codificação Run-Length

A codificação Run-Length (RL), como compressão, explora o fato de que, em várias oportunidades, o *bitstream* (seqüência de bits) na saída do transmissor (ou na entrada do armazenador) é composto de grandes trechos com símbolos ou *bytes* repetidos, definidos como *Runs*.

A filosofia do RL é agrupar todos os símbolos de um *Run* em apenas dois campos: um contendo o valor do símbolo (*symbol*) e o outro contendo o número de repetições (*run-length*), de forma que o número total de bits do *bitstream* fica bem reduzido.

Porém, de uma forma geral, a codificação RL não responde bem a um único erro nos dados comprimidos. Quando um erro de bits ocorre dentro de um *bitstream* compactado o resultado pode ser muito ruim. Imagine, por exemplo, uma seqüência comprimida como a mostrada na Fig.3.5.

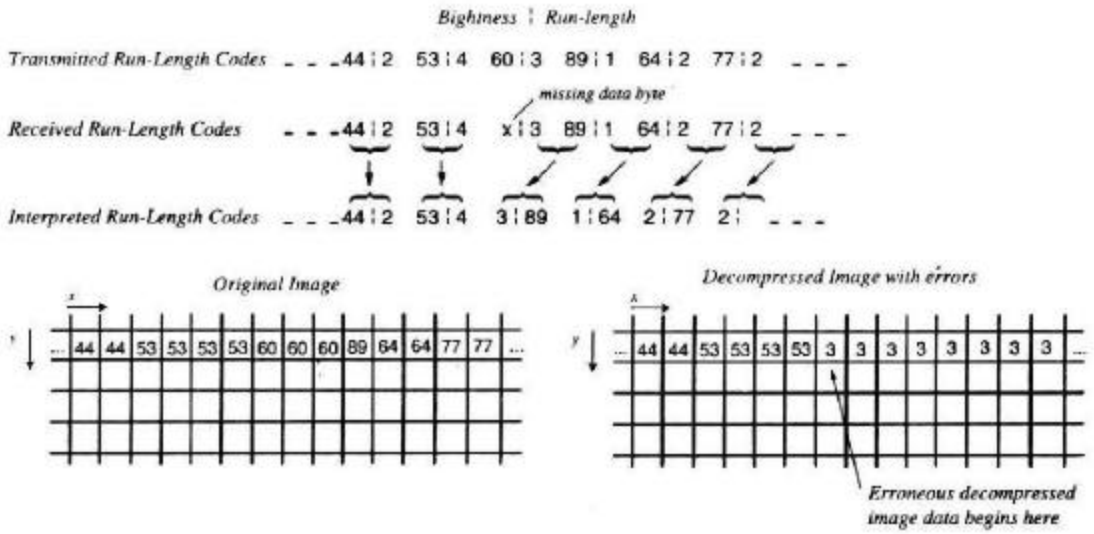


Fig. 3.5 – Exemplo de erro de dados com codificação RL. [12]

Se ocorrer uma perda de símbolo na posição indicada, o resultado final na imagem recuperada é simplesmente desastroso. Pode-se notar que a imagem será completamente deteriorada.

Para contornar esses problemas, algumas técnicas foram desenvolvidas para se modificar a operação básica do codificador tornando-o um pouco mais robusto nesses casos. Uma modificação

simples que ajuda significativamente é reinicializar o codificador no início de cada linha. Dessa forma, consegue-se evitar que um erro se propague para além da linha onde ele ocorreu.

3.4 – Efeito de Bloco

O Processamento de Imagens em Blocos

Foi visto anteriormente que a codificação por transformada é uma técnica de compressão muito eficiente na remoção da redundância de imagens e que seu ponto fundamental é a utilização de uma transformada ortonormal que descorrelaciona os dados de entrada compactando a energia do sinal em poucos coeficientes.

Um ponto não abordado anteriormente diz respeito à velocidade de execução das transformadas. A aplicação de uma transformada em uma imagem MxN requer, se calculada pela definição, da ordem de O{ M^2N^2 } operações (multiplicações e somas), o que é um número muito alto.

Para contornar esse problema surgiram os algoritmos rápidos que conseguem reduzir consideravelmente essa complexidade para O{$MN.\log_2(MN)$}. Mas ainda, em se tratando de imagens esse número é muito alto.

Considere, por exemplo, que se tem uma seqüência de vídeo monocromático, 640x480 pixels por *frame* e 30 *frames* por segundo. Usando codificação por transformada com um algoritmo rápido, para codificar essa imagem seria preciso aproximadamente $168x10^6$ operações por segundo somente para se executar a transformação (sem contar os procedimentos de quantização e codificação) o que é um número ainda bem alto.

Para reduzir ainda mais o número de operações, a solução encontrada foi dividir a imagem em blocos menores e processá-los separadamente. Para se ter uma idéia da redução no número de operações, calculando-se novamente para aquela imagem agora dividida em bloco de 8x8 pixels esse número cai para aproximadamente $55x10^6$ operações por segundo, o que representa uma redução de 3 vezes no número total de operações por segundo necessárias para a codificação o que justifica a sua adoção na maioria dos esquemas de compactação de vídeo.

Efeito de Bloco em Imagens

A divisão de imagens em blocos, no entanto, não trouxe apenas vantagens para a codificação por transformada. Um dos maiores, senão o maior, inconveniente proporcionado pela codificação por blocos é o Efeito de Bloco.

Como visto anteriormente, a imagem que vai ser codificada passa por uma transformação que concentra a energia em poucos coeficientes, então determina-se por algum critério o número de bits que cada coeficiente irá receber do quantizador e quantiza-se os coeficientes.

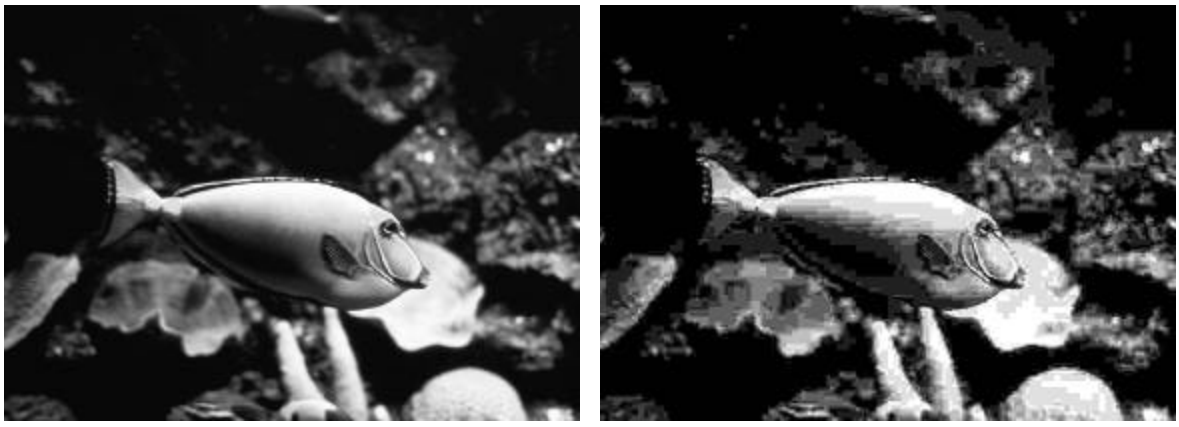
Como se sabe, a quantização é um processo que acarreta em perda de informação para o sinal e uma conseqüente distorção na imagem decodificada. O codificador então deve ser projetado para manter essas distorções dentro de um limite aceitável para a o sistema que será usuário desta imagem. No caso de imagens dedicadas à visão humana, como por exemplo as imagens de televisão, a tolerância a distorções é muito alta, pois a percepção visual humana não é muito apurada com detalhes. Assim poderíamos aplicar uma taxa de compressão bem grande nessas imagens.

Contudo, como o MPEG-2, por exemplo, adota codificação por transformada baseada em blocos e a quantização de cada um deles é feita separadamente dos adjacentes; não leva em conta as disparidades que aparecerão eventualmente na reconstrução entre esses blocos adjacentes. Dessa forma, as perdas de dois blocos sozinhos podem não ser notadas isoladamente, porém lado a lado, essas distorções nas proximidades da junção podem ficar contrapostas, realçando sua fronteira.

Essas disparidades nas fronteiras de blocos adjacentes (evidenciamento das junções) nas imagens reconstruídas são chamadas Efeito de Bloco, conhecido como o maior artefato de reconstrução no esquema de codificação MPEG.

Esse artefato se torna muito sério quando os coeficientes são “grosseiramente “ quantizados em decorrência de uma distribuição insuficiente de bits [13] e se torna altamente visível quando isso ocorre nos termos DC ou os AC de freqüência mais baixa. Daí, usualmente mais bits são usualmente necessários para o DC e ACs de baixa freqüência do que para os demais.

A Fig. 3.6 abaixo mostra uma imagem severamente afetada pelo Efeito de Bloco.



(a) Imagem original (b) Imagem recuperada

Fig. 3.6 – Exemplo do efeito de bloco em imagem compactada por codificação por transformada.

Redução do Efeito de Bloco

Para tentar resolver o problema da "blocagem" sem elevar a taxa de bits, várias técnicas têm sido propostas [14]. Filtragens Passa-Baixas nas fronteiras dos blocos [15][16], Aproximação DPCM-DC [17], Predição AC utilizando coeficientes DC de blocos vizinhos [17] são algumas delas.

Como o Efeito de Bloco se deve primariamente à inabilidade por parte da DCT em explorar correlações inter-blocos, a maioria das técnicas citadas explora correlações de valores de intensidade em blocos vizinhos.

As Filtragens Passa-Baixas, que reduzem os componentes de alta freqüência próximos à fronteira do bloco apresenta a vantagem de não necessitar transmitir qualquer informação adicional ou qualquer operação adicional no codificador; porém gera desnecessário borramento da imagem.

Aplicar DPCM (*Differential Pulse Code Modulation*) no componente DC dos coeficientes, técnica utilizada pelo JPEG e pelo MPEG2, reduz a blocagem causada pela quantização "grosseira" do coeficiente DC através da exploração da correlação dos coeficientes DCs de blocos vizinhos. Entretanto, requer estrita sincronização (típica em esquemas DPCM) entre codificador e decodificador e não possibilita redução do efeito de bloco em virtude dos termos ACs.

Já no método de predição AC (utilizado pelo JPEG), os ACs dos coeficientes DCT são preditos através dos DCs dos blocos vizinhos como forma de redução da blocagem, porém seu desempenho não se revela muito bom em virtude de que somente poucos coeficientes ACs quantizados que se tornam zero após a quantização são substituídos pela predição dos DCs.

Há outros algoritmos propostos para a redução do efeito de bloco tanto em imagens estáticas como em seqüências de vídeo. As aproximações para esse problema são feitas através dos mais variados pontos de vista. Há trabalhos desde os mais clássicos utilizando filtragens passa-baixas com vários formatos de filtros até métodos adaptativos baseados em Redes Neurais [18] e aplicações de modelamentos em domínio transformado [19].

Existem ainda trabalhos no campo de novas transformadas intencionando a substituição da DCT com taxas de compressão compatíveis e que apresentam menor efeito de bloco como a DSTr (*Discrete Sine Transform with Axis Rotation*) [8] e a *Wavelet* [7]. Nesse campo se destacam as transformadas *Lapped* [9].

Contudo, um problema levantado principalmente nos artigos mais recentes com relação à essas técnicas é a não efetividade da aplicação de critérios objetivos como o SNR (*Signal-to-Noise Ratio*) e da conseqüente necessidade se desenvolvimento de critérios que levem em conta mais fielmente as características da visualização humana.

Nesta pesquisa procura-se enfocar o problema da Redução do Efeito de Bloco pela perspectiva da melhoria da qualidade subjetiva das imagens baseando-se num procedimento de localização do efeito de bloco e na posterior correção do mesmo.

3.5 – Comentários

Neste capítulo finaliza-se os princípios de funcionamento do processo de codificação por transformada e a sua principal desvantagem: o efeito de bloco. Também descreve-se as técnicas mais utilizadas atualmente para a redução desse artefato e sua relação com este trabalho.

Antes de se detalhar a nossa proposta de redução de efeito de bloco, faz-se no próximo capítulo uma descrição dos sistema de quantização e codificação do padrão de codificação de imagens JPEG e do padrão de codificação de vídeo MPEG-2. Apesar do algoritmo MPEG-2 ter sido projetado especificamente para vídeo, a sua apresentação é devido ao fato de termos submetido à revista *IEEE Transactions on Multimedia* uma artigo a respeito deste trabalho utilizando a codificação e a quantização do MPEG-2 (*Test Model 5*).

Capítulo 4

Padrões de Compressão

Nos capítulos anteriores foram apresentados os princípios básicos do processo de codificação por transformada, que constitui a base de todos os padrões de compressão de imagens utilizados atualmente [6][17]. Outro ponto comum entre os padrões de compressão é a utilização da transformada DCT, embora já estejam em estágios avançados propostas baseadas em transformada *Wavelet* [7].

Já os processos de Quantização e de Codificação variam mais entre os diversos padrões dependendo do nível de complexidade e da eficiência de compactação de cada um deles. Apresentar-se-ão aqui os detalhes acerca dos dois padrões com os quais trabalha-se nessa pesquisa: o JPEG e o MPEG-2 (*Test Model 5*).

4.1 – O Padrão JPEG

O JPEG é, atualmente, um dos padrões mais utilizados para compressão de imagens digitais. Ele foi proposto em 1992 pelo comitê JPEG (*Joint Photographic Experts Group*) [17] baseado em estudos que vinham sendo realizados desde 1986.

O objetivo básico do JPEG foi criar um método geral para compressão de imagens digitais que atendesse à um conjunto de especificações técnicas que possibilitasse a aplicação desse padrão à uma ampla gama de aplicações. Algumas dessas especificações são [6]:

- Estar o mais próximo possível do "estado da arte" da compressão de imagens;
- Permitir à aplicação (ou ao usuário) poder facilmente balancear a relação taxa de compressão/qualidade da imagem de forma a atender suas necessidades específicas;
- Trabalhar independente do tipo de imagem, ou seja, não possuir restrições com relação ao tipo de fonte de imagem, conteúdo da imagem, espaço de cores, dimensões, resolução de pixels, etc.;
- Ter uma complexidade computacional pequena o suficiente que possibilite implementações em software sem necessitar de hardware especial mesmo em computadores muito simples;
- Permitir varreduras seqüencial ou progressiva de acordo com a necessidade;

- Oferecer uma opção de codificação hierárquica no qual uma versão de baixa resolução da imagem possa ser acessada sem a necessidade de descomprimir a imagem à resolução plena.

Basicamente, o JPEG trabalha com dois métodos de compressão, com e sem perdas. O processo sem perdas é baseado em codificação preditiva e o processo com perdas é baseado na codificação por transformada usando DCT. Há ainda 3 classes de codificação com perdas: Básico (*Baseline* ou Seqüencial), Progressivo e Hierárquico [6]. Como a presente pesquisa está relacionada com a redução do efeito de bloco decorrente da codificação por transformada, trabalha-se apenas com o modo de compressão seqüencial do padrão JPEG devido a sua menor complexidade.

JPEG Básico (*Baseline*)

A Fig. 4.1 mostra o diagrama em blocos do codificador JPEG Básico com perdas para uma imagem com apenas uma componente de cor (monocromática).

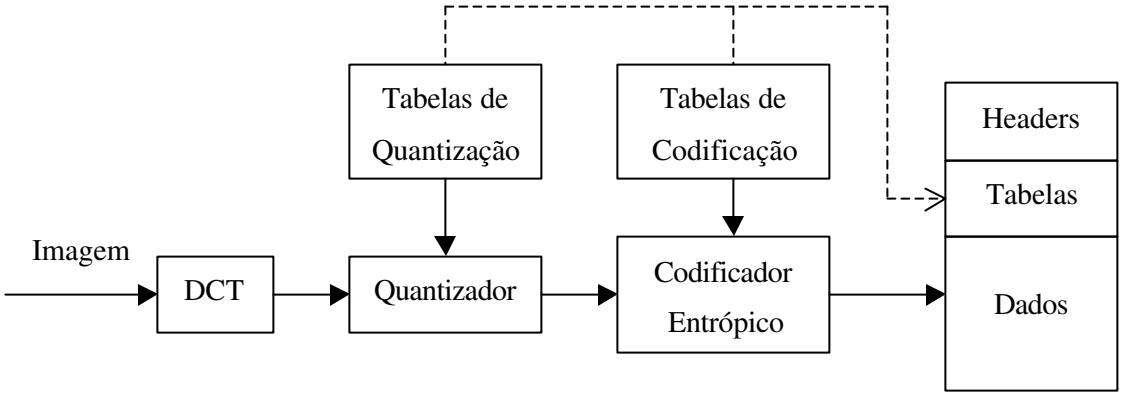


Fig. 4.1 – Diagrama em blocos do codificador JPEG Básico. [6]

Nesse processo de codificação, a imagem de entrada é dividida em blocos de tamanho 8x8. Cada bloco é submetido à DCT (que concentra a energia em alguns coeficientes) depois quantizado e preparado para o codificador entrópico utilizando a ordenação Zigzag mostrada na Fig. 4.2.

Antes da codificação entrópica, o coeficiente DC é subtraído do valor do coeficiente DC do bloco anterior. Essa diferença (DPCM) é a que será codificada entropicamente. Já os outros coeficientes (AC) passam por uma codificação *Run-Length* de forma a explorar a característica de empacotamento de energia da transformada que faz com que vários coeficientes quantizados sejam nulos. Como as imagens reais tem característica predominantemente passa-baixas, a ordenação Zigzag tende a agrupar os coeficientes nulos relativos às altas freqüências em longos *Runs*, aumentando a eficiência do algoritmo.

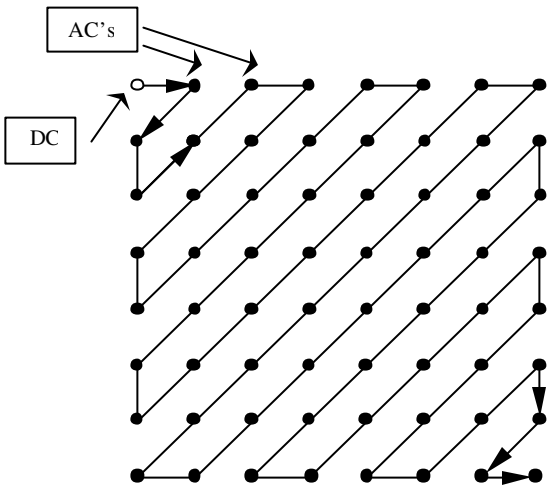


Fig. 4.2 – Ordenação Zigzag para um bloco 8x8.

Logo após o *Run-Length* é executada uma codificação entrópica que, para o JPEG Básico, é uma codificação de Huffmann. Nos modos Progressivo e Hierárquico pode-se alternativamente utilizar-se codificação Aritmética, mas a maioria das implementações prefere o código de Huffmann que é muito mais simples e apresenta um desempenho apenas 2% a 10% pior do que a codificação Aritmética [6].

O diagrama em blocos do decodificador JPEG, mostrado na Fig. 4.3, consiste basicamente na realização dos processos inversos equivalentes do codificador.

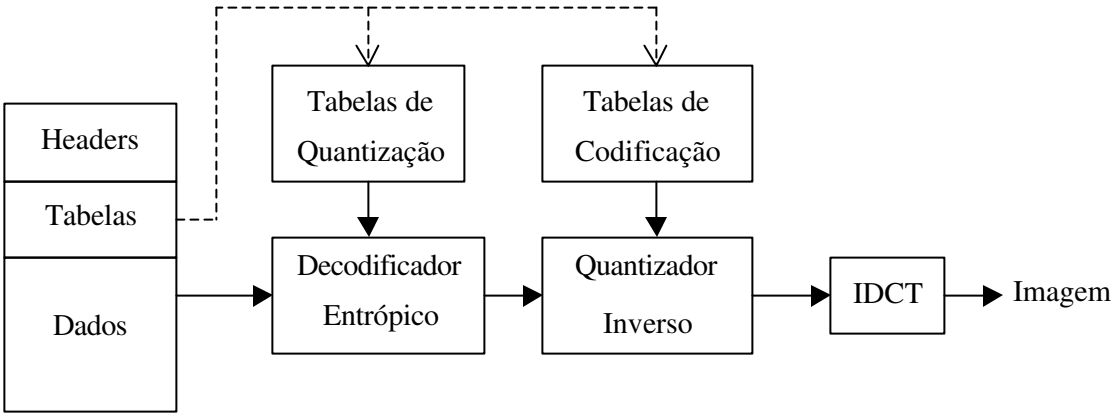


Fig. 4.3 – Diagrama em blocos simplificado do decodificador JPEG.

É importante observar-se que, como o processo de quantização não é reversível, a etapa de "Quantização Inversa" indicada na Fig. 4.3 se refere à multiplicação do valor decodificado pelo passo de quantização de forma a se obter o valor aproximado do coeficiente original, como será visto no item "Quantização".

Descreve-se agora alguns detalhes do processo de codificação/decodificação do JPEG básico:

DCT:

O JPEG utiliza a DCT par (também referenciada como DCT-II). As expressões para essa DCT foram apresentadas nas equações (2.53) e (2.54) e são:

$$C_{pq} = \alpha_p \cdot \alpha_q \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} F_{jk} \cos\left[\frac{\pi (2j+1) p}{2N}\right] \cos\left[\frac{\pi (2k+1) q}{2N}\right], \quad 0 \le p,q \le N-1 \tag{4.1}$$

$$F_{jk} = \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} \alpha_p \cdot \alpha_q \cdot C_{pq} \cos\left[\frac{\pi (2j+1) p}{2N}\right] \cos\left[\frac{\pi (2k+1) q}{2N}\right], \quad 0 \le j,k \le N-1 \tag{4.2}$$

onde:

$$\alpha_p = \begin{cases} \sqrt{\frac{1}{N}}, & para\ p = 0; \\ \sqrt{\frac{2}{N}}, & para\ 1 \le p \le N-1 \end{cases} \tag{4.3}$$

Quantização:

Cada coeficiente de saída da etapa DCT é quantizado pela divisão do seu valor original pelo valor correspondente da Matriz de Quantização e arredondado (inteiro mais próximo). Dessa forma, os elementos da matriz de quantização servem como valores de passo de quantização para os respectivos coeficientes. Sendo $\tilde{C}_{pq}$ o valor quantizado de C_{pq} e Q_{pq} o valor do passo de quantização respectivo, então pode-se escrever:

$$\tilde{C}_{pq} = inteiro\ mais\ próximo\ de \left[\frac{C_{pq}}{Q_{pq}}\right], \quad 0 \le p,q \le 7 \tag{4.4}$$

Os valores da matriz de quantização estão restritos a valores inteiros na faixa $1 \le Q_{pq} \le 255$ e são determinados pelo codificador. No caso de imagens coloridas, pode haver uma matriz para cada componente, mas é comum que duas ou mais componentes utilizem a mesma matriz. O Anexo K do padrão JPEG estipula uma matriz *default* para a componente de Luminância e outra para as de Crominância. Essas matrizes *default* foram definidas através de uma série de experimentos psicovisuais para determinar o limiar de visibilidade para as funções base da DCT em imagens de tamanho 760x576, com as componentes de cor subamostradas por um fator de 2 na direção horizontal e observada à uma distância de seis vezes a largura da tela [7]. Essas matrizes estão reproduzidas na Fig. 4.4:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a)

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(b)

Fig. 4.4 – Matrizes de quantização *default* do JPEG para: (a) Luminância; (b) Crominância. [20]

No decodificador, a "Quantização Inversa" é exatamente a multiplicação do valor decodificado pelo correspondente passo de quantização para a obtenção da aproximação $\hat{C}_{pq}$ do coeficiente original C_{pq}. Matematicamente:

$$\hat{C}_{pq} = Q_{pq} \cdot \tilde{C}_{pq}\ ,\ \ 0 \le p,q \le 7 \tag{4.5}$$

A utilização dessas matrizes de quantização *default* apresenta, em geral, uma boa relação entre taxa de compressão e qualidade subjetiva apesar de não representar o ponto ótimo para cada imagem. Mas é necessário prover alguma forma de ajustar essa relação de acordo com as necessidades de compressão de cada sistema. Na prática, é muito comum a utilização de versões escalonadas dessas matrizes *default*. Uma implementação muito popular é a que utiliza um *Fator de Qualidade Q* para determinar a escala que é aplicada a cada elemento da matriz de quantização. O valor de *Q* é um inteiro limitado à faixa de 1 a 100 e o fator de escala *FE* é calculado a partir de *Q* pela expressão [7]:

$$FE = \begin{cases} \dfrac{50}{Q}\ , & para\ 1 \le Q \le 49 \\ 2 - \dfrac{Q}{50}\ , & para\ 50 \le Q \le 99 \\ 0{,}01\ , & para\ Q = 100 \end{cases} \tag{4.6}$$

Assim, a nova matriz de quantização é conseguida através da multiplicação de cada um dos elementos da matriz *default* pelo fator de escala, arredondamento para o inteiro mais próximo e limitação à faixa permitida para cada valor da matriz: de 1 a 255. É interessante notar que para *Q=50* a matriz de quantização é a matriz *default.*

Codificação:

A codificação do JPEG é basicamente dividida em duas etapas: o Mapeamento Coeficiente-Símbolo (Pré-codificação) e a Codificação Entrópica. Essas duas etapas tratam diferenciadamente os coeficientes DC e os coeficientes AC dos blocos, isso porque existe, em geral, uma forte correlação entre os coeficientes DC de blocos vizinhos. Assim, o coeficiente DC de cada bloco é codificado diferencialmente com relação ao coeficiente DC do bloco anterior (DPCM). Para uma imagem original com precisão de 8 bits, o coeficiente DC pode assumir valores na faixa *[0,2047]*, assim, a diferença entre dois DCs pode assumir qualquer valor na faixa *[-2047,2047]*. Para realizar a codificação de Huffmann subsequente, tratando-se o problema dessa forma, seria necessária a inclusão no cabeçalho de uma tabela de codificação com 4095 valores, o que ocupariaq muito espaço e prejudicaria a compressão. Para contornar esse problema, os valores de diferença são classificados em 12 categorias de acordo com a sua amplitude, como mostrado na Tabela IV.1:

Categoria	**Módulo da Amplitude**
0	0
1	1
2	2,3
3	4 a 7
4	8 a 15
5	16 a 31
6	32 a 63
7	64 a 127
8	128 a 255
9	256 a 511
10	512 a 1023
11	1024 a 2047

TABELA IV.1 – Categorias de amplitude para coeficiente DC. [20]

Dessa forma, a amplitude pode ser representada por uma categoria *K* seguida dos *K* bits menos significativos da amplitude (usando complemento de um para os número negativos). Assim, as categorias são codificadas entropicamente requerendo a inserção de uma tabela com apenas 12 códigos. Por exemplo, suponha que a diferença entre o coeficiente DC do bloco atual e o DC do bloco anterior seja igual a –10. Essa diferença é mapeada para o símbolo: (4,-10). Se o código de Huffmann para a categoria 4 for ‘101’, então o valor codificado fica ‘101 0101’.

Já os coeficientes AC apresentam, devido à quantização e à ordenação em Zigzag [20], uma grande quantidade de termos nulos consecutivos, o que favorece a aplicação da codificação *Run-Length*. Como os valores dos coeficientes AC variam na faixa de *[-1023,1023]*, também aqui utiliza-se a classificação em categorias da Tabela IV.1 só que limitada à 11 categorias (de 0 a 10).

Assim, os coeficientes AC não-nulos são mapeados em uma tripla *[(R/K)A]*: *R=Run*: indica o número de coeficientes AC nulos desde o último coeficiente transmitido (não-nulo); *K=Categoria*: da amplitude segundo a Tabela IV.1; e *A=Amplitude*: os *K* bits menos significativos da amplitude (em complemento de um para os negativos) semelhante ao caso DC [20].

Para os coeficientes AC, a tabela de Huffmann é feita para o par (*Run, Categoria*) (referenciado aqui por *R/K*) sendo que o *Run* é limitado em 15, resultando em uma tabela de 150 valores. Dentro dessa tabela há dois símbolos de especial interesse (a): quando o *Run* entre dois coeficientes não-nulos é maior do que 15, esse *Run* é dividido em partes inteiras de 15 (quantas forem necessárias), que são codificadas com o símbolo (*15,0*), e uma contendo o restante do *Run* codificada normalmente; (b) o símbolo (*0,0*) que indica que todos os coeficientes restantes do bloco são nulos e, por isso, é denominado EOB (*End of Block*).

O padrão JPEG permite a especificação de até 8 tabelas de Huffmann para a codificação de todas as componentes de cor de uma imagem. No Anexo K do padrão são apresentadas quatro tabelas recomendadas para imagens em geral: duas para a Luminância (DC e AC) e duas para Crominância (também DC e AC).

Exemplo de Codificação de um Bloco com JPEG:

Na Fig. 4.5 é mostrada a imagem Zelda destacando-se o bloco que foi selecionado para esse exemplo:

112	114	114	107	102	96	101	95
102	104	102	108	98	99	95	95
103	101	107	101	102	100	98	102
115	103	99	107	103	95	92	95
113	117	109	100	97	88	92	92
116	115	109	93	96	96	97	91
116	110	102	95	100	98	95	93
112	107	101	100	102	98	103	96

(a) (b)

Fig. 4.5 – (a) Imagem Zelda 256x256 com bloco em destaque; (b) Valores numéricos do bloco.

Aplicando-se nesse bloco a DCT-II direta, equação (4.1), e usando-se a matriz de quantização *default* para Luminância (Fig. 4.4a), obtém-se os coeficientes:

814,5	44,49	8,14	1,99	0,25	-0,08	-2,56	3,68
3,06	-5,83	-10,76	-9,66	2,32	2,82	1,54	-3,81
6,01	-6,06	-1,77	1,74	-4,29	2,73	-1,89	0,25
3,17	17,25	5,79	-9,03	-2,61	-2,30	-2,25	2,77
4,75	6,47	-0,30	-1,81	4,50	2,36	-5,29	1,06
3,38	0,57	1,11	5,21	4,26	0,11	0,77	2,14
5,36	-2,77	2,61	-2,12	-5,80	-0,01	1,77	5,66
1,32	0,69	1,42	-5,21	-3,43	-3,49	0,31	3,33

(a)

51	4	1	0	0	0	0	0
0	0	-1	-1	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(b)

Fig. 4.6 – Coeficientes do bloco exemplo : (a) Após a DCT-II ; (b) Após a quantização.

O passo seguinte é utilizar a ordenação Zigzag, mostrada na Fig. 4.2, para formar a seqüência de dados que será codificada. A representação desse bloco então fica:

$$51, 4, 0, 0, 0, 1, 0, -1, 0, 0, 0, 1, 0, -1, 0, 0, ...,0 \tag{4.7}$$

Supondo que o coeficiente DC do bloco anterior fosse igual a 56 para a aplicação do DPCM-DC, aplicando o *Run-Length* para os coeficientes AC e utilizando a Tabela IV.1, pode-se pré-codificar essa seqüência de dados no conjunto de símbolos:

$$[(3, -5)]; [(0/3,) 4]; [(3/1), 1]; [(1/1), -1]; [(3/1), 1]; [(1/1),-1]; (EOB) \tag{4.8}$$

Como a imagem original nesse exemplo é monocromática, para realizar a sua codificação entrópica são utilizadas as tabelas-código recomendadas pelo padrão para a componente de Luminância [17]. Dessa forma, o *bitstream* final para esse bloco fica:

$$\underbrace{\overline{100}\,\overline{010}}_{6bits};\ \underbrace{\overline{100}\,\overline{100}}_{6bits};\ \underbrace{\overline{111010}\,\overline{1}}_{7bits};\ \underbrace{\overline{1100}\,\overline{0}}_{5bits};\ \underbrace{\overline{111010}\,\overline{1}}_{7bits};\ \underbrace{\overline{1100}\,\overline{0}}_{5bits};\ \underbrace{\overline{1010}}_{4bits} \Rightarrow 40\ bits \tag{4.9}$$

onde *K, A* e *R* representam, respectivamente, a categoria, a amplitude do coeficiente e o número de coeficientes nulos desde o último não nulo da seqüência (*Run*).

Dessa forma, os 64 coeficientes do bloco que, originalmente, ocupariam $64 \cdot 8 = 512\ bits$ são codificados em apenas *40 bits*, resultando em uma taxa de compressão de aproximadamente 13:1.

4.2 – O padrão MPEG-2 Test Model 5

Introdução

O MPEG-2 é um algoritmo de compactação de vídeo com perdas, resultado da segunda fase dos trabalhos do MPEG (*Moving Picture Experts Group*). Esse algoritmo foi projetado para suportar uma ampla gama de aplicações de vídeo desde HDTV (com taxas de 80 a 100 Mbps) até aplicações menores com taxas abaixo de 1 Mpbs.

De forma geral, o algoritmo MPEG-2 apresenta várias sintaxes possíveis para os vários níveis de aplicações suportadas. Como o interesse desse trabalho nesse sistema é servir de base de comparação para os resultados obtidos com o algoritmo de redução de Efeito de Bloco proposto, será descrita apenas a parte de Codificação do MPEG-2 com ênfase na etapa de compressão espacial (visto que se trabalha apenas sem a compressão temporal).

Dentro do padrão MPEG-2 foi escolhido o *Test Model 5 Document* (TM5) [21] para as simulações por fornecer uma aproximação para o controle da taxa de bits, uma vez que o mecanismo específico pelo qual o fator de escala do quantizador é calculado não é especificado pelo padrão MPEG-2 [22]. O código fonte do codificador MPEG-2 TM5 utilizado nessa pesquisa foi encontrado na *Home Page* do *MPEG Software Simulation Group* [23].

A seguir, serão descritos maiores detalhes dos procedimentos de Compressão Espacial utilizados por esse ambiente, relacionados à geração do Efeito de Bloco.

Codificação do MPEG-2

Um sinal de vídeo digital é composto por uma seqüência de imagens que representam uma amostragem temporal da projeção de uma cena (3D) sobre a superfície sensora (2D) da câmera de vídeo. Cada imagem monocromática também é amostrada espacialmente em Pixels (*Picture Elements*) e dividida em Macroblocos (MBs) de 16x16 pixels (usados para a compressão temporal) contendo cada um quatro blocos de 8x8 pixels (usados para a compressão espacial) [24].

O MPEG classifica cada uma das imagens da seqüência em um dos três tipos: *I*-, *P*- ou *B*-*pictures* (detalhadas mais adiante) segundo o tipo de compressão temporal (estimação/compensação de movimento) utilizada em cada uma delas. Como apenas foram utilizadas imagens estáticas, é descrito aqui apenas o processo de compressão espacial do MPEG-2.

Dentro de cada um dos tipos de imagem, os MBs podem ser codificados de forma diferente. As árvores de decisão e comentários para a codificação deles encontram-se na Fig.4.7 [6].

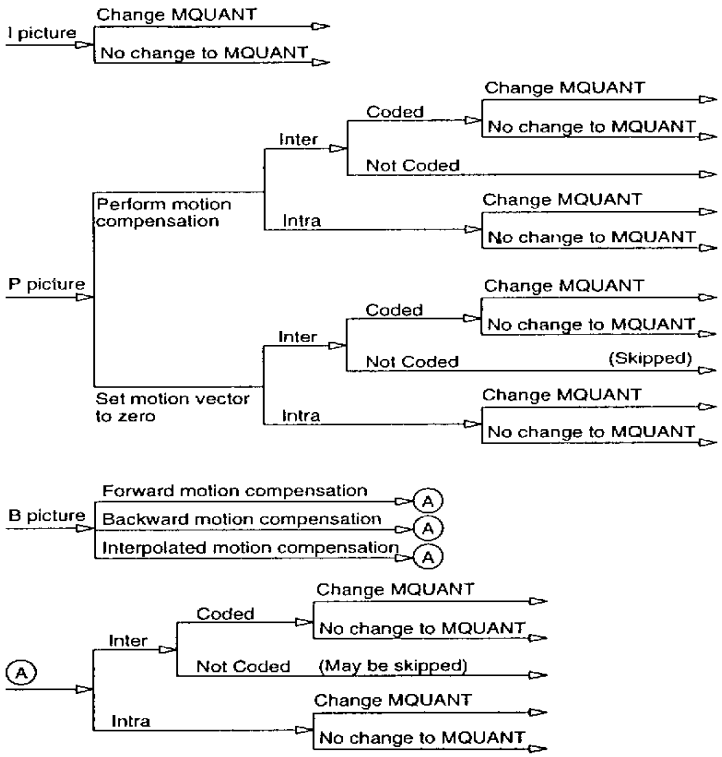


Fig. 4.7 – Árvores de decisão para a codificação dos MB's em *I-, P-,* e *B- pictures*. [6]

I-Picture

As *I-Pictures* são codificadas sem compensação de movimento [6], ou seja, nenhum dos seus MBs sofre compressão temporal e, porisso, são denominados macroblocos *Intra* ou MB *Intra*.

P-Pictures

Nas *P-Pictures*, os MBs são codificados podendo usar compensação de movimento *forward* [6], ou seja, diferencialmente com relação a um *frame* referência já codificado. Assim, o codificador tem uma gama maior de escolhas para cada MB, como pode ser visto na Fig. 4.7.

Primeiramente, em alguns casos, o erro de predição usando um vetor de movimento não-nulo pode estar próximo ao erro de predição para esse MB assumindo um vetor de movimento nulo. Como vetores de movimento não-nulos requerem bits adicionais, se torna mais eficiente codificar esse MB utilizando a segunda opção.

A seguir, vem a decisão de se codificar o MB como tipo *Intra* ou *Inter*, pois em muitos casos pode requerer menos bits codificar um MB como se fosse *Intra* (sem compensação de movimento), mesmo que ele pertença a uma *P-Picture*. Isso pode ocorrer se a estimação de movimento falhar muito devido a um alto nível de atividade temporal.

O próximo ramo é a decisão se o MB-*Inter* precisa ou não ser codificado: se, após a quantização, há pelo menos um coeficiente no MB que seja não-nulo, o MB é codificado como do

tipo *Non-Intra*, se todos os coeficientes forem nulos e os vetores de movimento forem não-nulos então o bloco é do tipo *Not Coded* e se além dos coeficientes, os vetores de movimento também se anulem, então o MB é *Skipped*.

A última escolha é se o fator de escala de quantização *mquant* (vide subseção "Quantização do MPEG-2") precisa ou não ser alterado. O codificador pode decidir alterá-lo se por exemplo detectar possível *buffer overflow* ou *buffer underflow*.

B-Pictures

A escolha do tipo de codificação do MB é muito similar ao caso das *P-Pictures*. Resumidamente:

Decisão sobre usar compensação de movimento *forward*, *backward* ou interpolada.

Codificar o MB como tipo *Intra* ou *Inter*.

Decisão se o MB necessita ou não ser codificado. O MB pode ser *Non-Intra* se a escolha anterior tiver sido tipo *Inter* e os seus coeficientes quantizados não forem todos nulos. Também como na *P-Picture*, mesmo quando o MB é codificado, alguns de seus blocos podem ser *Not Coded* ou *Skipped.*

Decisão se a escala do quantizador (*mquant*) precisa ou não ser modificado.

Quantização do MPEG-2

Enquanto a compensação de movimento é baseada na estrutura de MB (16x16), a transformação e quantização são baseadas em blocos de 8x8 pixels. As etapas abaixo estão descritas nos capítulos 7 e 10 disponíveis em [21] .

Para efeito da quantização, os MBs são classificados em *Intra* ou *Inter.* O tipo *Inter* engloba os outros três tipos de MB: *Coded, Not Coded* e *Skipped*

A) Quantização de *Intra-Blocks*

Após a DCT, cada coeficiente é quantizado sendo dividido por seu elemento correspondente da matriz de quantização *Intra.*

Para o coeficiente DC o passo de quantização é fixo, enquanto para os ACs depende não somente do valor do elemento correspondente na matriz de quantização, mas também do fator de escala de quantização (a ser visto mais à frente).

Coeficiente DC

O tamanho do passo de quantização para o coeficiente DC dos componentes luminância e crominância é 8,4,2 ou 1 e é determinado pelo codificador separadamente do restante do bloco. Assim, sendo *dc* o valor pós-DCT não quantizado com precisão de 11 bits; o valor quantizado do

DC, *QDC* é [21]:

$$QDC = dc // 8 \tag{4.10}$$

$$QDC = dc // 4 \tag{4.11}$$

$$QDC = dc // 2 \tag{4.12}$$

Sendo que a simbologia 'A//B' indica a parte inteira da divisão de A por B.

Coeficientes AC

Os coeficientes AC, *ac(i,j),* são primeiro quantizados por fatores de quantização individuais [21]:

$$ac\sim(i,j) = 16 \cdot \frac{ac(i,j)}{QI(i,j)} \tag{4.13}$$

onde *QI(i,j)* é o (i,j)-ésimo elemento da matriz de quantização *Intra* da Fig. 4.8. O valor *ac~(i,j)* é limitado ao range [-2048, 2047].

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Fig. 4.8 – Matriz de Quantização *Intra* do MPEG-2. [21]

A seguir, o passo de quantização para esses "coeficientes DCT escalonados" (*ac~(i,j)*) é obtido através do parâmetro de quantização, *quantizer_scale* (também referenciado como *mquant*), que é calculado à nível de MB na seção "Controle de Taxa de Bits e Controle de Quantização".

Desse modo [21]:

$$QAC(i,j) = \frac{ac\sim(i,j) + sign(ac\sim(i,j)) \cdot (p \cdot quantizer\_scale // q)}{2 \cdot quantizer\_scale} \tag{4.14}$$

Para o *Test Model 5*, *p=3* e *q=4.*

Cada coeficiente quantizado é então limitado (*clipped*) no intervalo (*range*) *[-255, 255] ou [-2047, 2047]*, dependendo da *flag* de compatibilidade com o MPEG-1, que "seta" o número máximo de bits que poderá ser usado na quantização do coeficiente.

B) Quantização de *Inter-Blocks*

O tamanho do passo para quantizar tanto o DC como os ACs é obtido através do parâmetro de quantização, *quantizer_scale*, que é calculado à nível de macrobloco segundo a seção "Controle de Taxa de Bits e Controle de Quantização". Assim [21]:

$$ac\sim(i,j)=(16\cdot ac(i,j))//QN(i,j) \tag{4.15}$$

onde $QN(i,j)$ é o elemento da matriz de quantização da Fig. 4.9.

16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24
18	19	20	21	22	23	24	25
19	20	21	22	23	24	26	27
20	21	22	23	25	26	27	28
21	22	23	24	26	27	28	30
22	23	24	26	27	28	30	31
23	24	25	27	28	30	31	33

Fig. 4.9– Matriz de Quantização *Inter* do MPEG-2. [21]

Desse modo [21]:

$$QAC(i,j)=\frac{ac\sim(i,j)}{2\cdot quantizer\_scale} \tag{4.16}$$

C) Comentários extras

O MPEG-2 permite que as matrizes de quantização *Intra* e *Non-Intra* (também chamada de *Inter*) sejam alteradas à nível de *frame* e para os formatos 4:2:2 e 4:4:4 suporta o uso de diferentes matrizes de quantização para os componentes de luminância e de crominância. Logo, se for de interesse do usuário, pode-se usar duas matrizes para blocos de luminância e duas matrizes para blocos de crominância (*Intra* e *Non-Intra*).

Outro ponto importante é que no MPEG-2, o fator de escala do quantizador pode ser mudado à nível de MB objetivando obter-se uma taxa de bits constante na saída do codificador. O critério para calculá-lo não é parte do padrão MPEG-2. Esse procedimento é abordado no *Test*

Model 5 [21] e será descrito a seguir por ter sido utilizado nessa pesquisa.

Controle de Taxa de Bits e Controle de Quantização

O TM5 [21] descreve procedimentos para controlar a taxa de bits através da adaptação do parâmetro de quantização, *quantizer_scale* (ou *mquant*) em três passos:

Número almejado de bits na alocação: nesse passo (realizado antes da codificação da imagem), estima-se o número de bits disponível para codificá-la.

Controle da taxa de bits: tem o significado de um *buffer* virtual e seta o valor referência do parâmetro de quantização para cada MB.

Quantização Adaptativa: modula o valor referência do parâmetro de quantização de acordo com a atividade espacial no MB para derivar o parâmetro de quantização, *mquant*, usado para quantizar o MB.

Passo 1) Alocação de Bits

Estimação da Complexidade

Após uma imagem de determinado tipo ser codificada (*I, P, B*), a correspondente medida de complexidade global (X_i, X_p, X_b) é atualizada da seguinte forma:

$$X_i = S_i Q_i \tag{4.17}$$

$$X_p = S_p Q_p \tag{4.18}$$

$$X_b = S_b Q_b \tag{4.19}$$

Onde S_i, S_p, S_b são o número de bits gerados com a codificação dessa imagem e Q_i, Q_p, Q_b são os parâmetros de quantização médios calculados através da média dos valores de quantização reais usados na codificação de todos os MBs incluindo os *skipped*. Sendo *bit_rate* a taxa de bits (em bits/s); os valores de inicialização são:

$$X_i = (160 \cdot bit\_rate) / 115 \tag{4.20}$$

$$X_p = (60 \cdot bit\_rate) / 115 \tag{4.21}$$

$$X_b = (42 \cdot bit\_rate) / 115 \tag{4.22}$$

O número almejado de bits para a próxima imagem no GOP (*Group of Pictures*) T_i, T_b ou T_p é calculado da seguinte forma:

$$T_i = max\left\{ \frac{R}{\left(1 + \frac{N_p X_p}{X_i K_p} + \frac{N_b X_b}{X_i K_b}\right)}, \frac{bit\_rate}{8 \cdot picture\_rate} \right\} \tag{4.23}$$

$$T_p = max\left\{ \frac{R}{\left(N_p + \frac{N_b K_p X_b}{K_b X_p} \right)}, \frac{bit\_rate}{8 \cdot picture\_rate} \right\} \quad (4.24)$$

$$T_b = max\left\{ \frac{R}{\left(N_b + \frac{N_p K_b X_p}{K_p X_b} \right)}, \frac{bit\_rate}{8 \cdot picture\_rate} \right\} \quad (4.25)$$

Nessas fórmulas :

- K_p e K_b são constantes "universais" que dependem das matrizes de quantização. Para as matrizes das Figs. 4.8 e 4.9, K_p =1,0 e K_b = 1,4.
- $S_{i,p,b}$ é o número de bits gerado na imagem já codificada.
- R é o número restante de bits associado ao GOP e é atualizado via: $R = R - S_{i,p,b}$ após codificar a *picture.*
- Antes da codificação da primeira *picture* do GOP (a *I-Picture*), a inicialização é dada por G = *bit_rate*N / picture_rate* e $R = G+R$, onde N é o número de *pictures* no GOP. No início da seqüência $R=0$.
- N_p e N_b são o número de *P-Pictures* e *B-Pictures* restantes no GOP corrente na ordem do codificador.

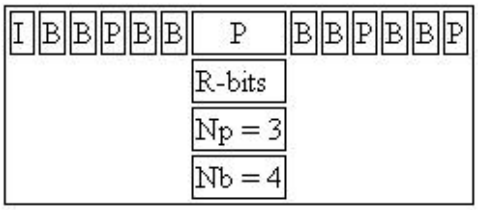


Fig. 4.10 – Exemplo de *pictures* remanescentes num GOP no frame 7.[21]

Passo 2) Controle da Taxa de Bits

Antes de codificar o **MBj** (j >=1), é calculado a ocupação do *buffer* virtual apropriado da seguinte forma de acordo com o tipo de *picture*:

$$d_j^i = d_0^i + B_{j-1} - \left(\frac{T_i \times (j-1)}{MB\_cnt} \right) \quad (4.26)$$

$$d_j^p = d_0^p + B_{j-1} - \left(\frac{T_p \times (j-1)}{MB\_cnt} \right) \tag{4.27}$$

$$d_j^b = d_0^b + B_{j-1} - \left(\frac{T_b \times (j-1)}{MB\_cnt} \right) \tag{4.28}$$

Nas fórmulas acima:

- d_0^i, d_0^p, d_0^b são a ocupação do *buffer* virtual iniciais, um para cada tipo de *picture.*
- B_j é o número de bits gerado pela codificação dessa *picture* para todos os MBs até o *MBj* inclusive.
- *MB_cnt* é o número de MBs na *picture*.
- d_j^i, d_j^p, d_j^b são a ocupação do *buffer* virtual no *MBj*, um para cada tipo de *picture*.

A cada *picture*, o valor final de ocupação do *buffer* virtual ($d_0^i, d_0^p, d_0^b : j=MB\_cnt$) será utilizado como parâmetros iniciais d_0^i, d_0^p, d_0^b na codificação da próxima *picture* do mesmo tipo.

A seguir, sendo *r* definido com o "parâmetro de reação":

$$r = 2 \times \frac{bit\_rate}{picture\_rate} \tag{4.29}$$

e d_j a ocupação do *buffer* virtual do tipo certo de *picture*, o parâmetro de quantização referência *Qj* para o MB_j é calculado por:

$$Q_j = \left(\frac{d_j \times 31}{r} \right) \tag{4.30}$$

E a inicialização do *buffer* virtual fica:

$$d_0^i = 10 \times \frac{r}{31}; \quad d_0^p = K_p \times 10 \times \frac{r}{31}; \quad d_0^b = K_b \times 10 \times \frac{r}{31} \tag{4.31}$$

Passo 3) Quantização Adaptativa

Modula o valor referência do parâmetro de quantização de acordo com a atividade espacial no MB para derivar o parâmetro de quantização, *mquant*, usado para quantizar o MB efetivamente.

Assim, primeiro há o cálculo da medida de atividade espacial para o *MBj* a partir dos 4 blocos (*n = 1..4*) de luminância organizados em *frame* e dos 4 blocos (*n = 5..8*) de luminância organizados em *field* usando valores originais (Intra) dos pixels. Sendo P_k os valores das amostras no *n*-ésimo bloco original 8x8:

$$P\_mean_n = \frac{1}{64} \times \sum_{k=1}^{64} P_k^n \tag{4.32}$$

$$vblk_n = \frac{1}{64} \times \sum_{k=1}^{64} \left(P_k^n - P\_mean_n\right)^2 \tag{4.33}$$

$$act_j = 1 + min\left(vblk_1, vblk_2, ..., vblk_8\right) \tag{4.34}$$

Sendo *avg_act* o valor médio de *actj* da última imagem codificada, através da normalização, *N_actj* fica:

$$N\_act_j = \frac{\left(2 \times act_j\right) + avg\_act}{act_j + \left(2 \times avg\_act\right)} \tag{4.35}$$

Na primeira imagem, *avg_act = 400.*

Por fim, sendo *Qj* o parâmetro de quantização referência obtido no passo 2; *mquant* é obtido através de:

$$mquant_j = Q_j \times N\_act_j \tag{4.36}$$

O valor final do *mquant* é limitado no intervalo (*range*) [1..31].

As limitações desse procedimento segundo a própria fonte [21] é que o passo 1 não manuseia eficientemente mudanças de cena; um valor errado de *avg_act* é usado no passo 3 após uma mudança de cena; e a obediência ao VBV (*Virtual Buffer Verifier*) não é garantida, podendo ocasionar um *overflow* ou um *underflow.*

Codificação do MPEG-2

Após a etapa de quantização, os blocos sofrem, por fim, uma codificação VLC (*Variable Length Coding*) sem perdas baseada nos algoritmos *Run-Length* e código de Huffmann semelhantes aos utilizados no algoritmo JPEG.

Essa codificação é dependente do tipo do macrobloco que vai ser codificado. Como descrito no item "Quantização do MPEG-2", há basicamente quatro tipos de macroblocos possíveis: *Intra*, *Non-Intra*, *Not-Coded* e *Skipped*.

I) MBs-*Intra*

Os macroblocos do tipo *Intra* são todos os macroblocos de uma *I-Picture* e de uma *P- ou B-Picture* nas quais o MPEG-2 constata como mais vantajoso em termos de bits utilizar uma codificação *Intra.* Isso ocorre porque a Estimação de Movimento gera um sinal erro cuja atividade é

muito grande (maior do que um determinado limiar) ou maior do que a atividade do próprio macrobloco.

Num Macrobloco Intra todos os blocos tem que ser codificados da mesma forma [22]:

A) O componente DC é codificado em DPCM usando como predição o valor armazenado no preditor. O preditor armazena o componente DC do último bloco Intra codificado e é "resetado" para um valor *default* quando: a) Inicia um novo *Slice*; b) Um bloco *Non-Intra* é codificado; c) Um macrobloco é *Skipped.* O valor *default* depende somente da *flag: intra_dc_precision* de acordo com a Tabela IV.2:

Intra_dc_precision	**Bits of precision**	**Reset value**
0	8	128
1	9	256
2	10	512
3	11	1024

Tabela IV.2 – Relação entre *intra_dc_precision* e o valor *default* do preditor DC.[22]

A seguir, o valor DC é chamado de *dc_dct_differential* e é codificado num par (L/A), onde L é a *Level* (equivalente à Categoria do JPEG) e A é o Deslocamento de Amplitude que é formado representando o módulo da amplitude do coeficiente em binário com L bits e substituindo-se o bit mais significativo pelo bit de sinal *s* (0=positivo e 1=negativo).

O bit mais significativo é recuperado no decodificador através da informação do *Level* (L). A determinação do valor L é dada em função da amplitude de acordo com a Tabela IV.3, que é uma extensão da Tabela IV.1, pois o MPEG-2 [22]trabalha com 40 *Levels* ao invés das 12 do JPEG.

B) Os demais componentes (AC) são codificados utilizando o procedimento para os componentes de um bloco Non-Intra (semelhante ao usado no JPEG) descritos a seguir.

II) MBs-*Non-Intra*

O fato de um macrobloco ser *Non-Intra* não assegura que todos os seus blocos serão codificados. Se qualquer um seus dos blocos for inteiramente nulo, esse bloco não é codificado e isso é notificado no cabeçalho do macrobloco através do campo *cbp* (*coded block pattern*).

Os blocos *Non-Intra* que serão codificados são, primeiramente, transformados em vetor segundo a ordem *Zig-Zag* mostrada na Fig.4.2 [22].

Posteriormente, os componentes do bloco são pré-codificados utilizando-se o *sistema Run-Level* (R/L) semelhante ao do JPEG. A principal diferença aqui é que, no caso dos blocos Non-

Intra, o valor DC também entra no cálculo do *Run*, ou seja, todos os coeficientes são tratados igualmente. Depois do último valor não-nulo do bloco, é enviado o código *End of Block (EOB)* para indicar que todos os outros valores daquele bloco são nulos.

Level	**Amplitude (em módulo)**
0	0
1	1
2	2,3
3	4 a 7
4	8 a 15
5	16 a 31
6	32 a 63
7	64 a 127
8	128 a 255
9	256 a 511
10	512 a 1023
11	1024 a 2047
12	2048 a 4095
.	.
.	.
.	.
40	2^{39} a (2^{40}-1)

TABELA IV.3 – Relação entre nível e amplitude do coeficiente. [6]

O par R/L é então codificado em VLC por uma tabela fixa (Tabelas B.14, B.15 e B.16 do documento [22]) e é seguido pela Amplitude A que é formada representando-se módulo da amplitude do coeficiente em binário com L bits e substituindo-se o bit mais significativo pelo bit de sinal *s* (0=positivo e 1=negativo). As principais diferenças desse sistema com o sistema do JPEG é que no neste último é possível especificar-se uma tabela VLC diferente para cada imagem e o valor do Deslocamento de Amplitude *A* é representado de forma diferente.

III) MBs-*Not-Coded*

Um macrobloco é do tipo *Not-Coded* quando todos os seus blocos quantizados são inteiramente nulos e a Estimação de Movimento resultou em um Vetor de Movimento não-nulo.

Assim, para esses MBs o codificador envia apenas as informações de cabeçalho e os vetores de movimento. O decodificador identifica um MB-*Not-Coded* através do campo *cbp* (*Coded Block Pattern*) presente no cabeçalho do MB.

IV) MBs-*Skipped*

Quando todos os blocos de um MB forem nulos, assim como os Vetores de Movimento, o MB é denominado *Skipped*.

O codificador quando encontra um macrobloco *Skipped* simplesmente o ignora e acrescenta 1 ao contador de deslocamento de endereço de macrobloco (*MBAinc*). Como esse contador é parte integrante do cabeçalho de todos os macroblocos, quando o codificador encontrar um bloco não-*Skipped*, envia esse número e "reseta" o contador.

Dessa forma, o decodificador sempre lê no início do macrobloco quantos macroblocos *Skipped* existem entre o macrobloco que vai ser decodificado no momento e o que acabou de ser decodificado.

Exemplo de codificação de um bloco com MPEG-2 TM5

Para exemplificar a codificação de um bloco *Intra*, considere o bloco transformado mostrado na Fig. 4.6a. Fazendo com que o fator de escala de quantização (*mquant*) seja arbitrariamente igual a 4 e o DC tenha precisão de 8 bits, o bloco quantizado é mostrado na Fig. 4.11:

102	5	1	0	0	0	0	0
0	0	-1	-1	0	0	0	0
1	-1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Fig. 4.11 – Bloco destacado quantizado da imagem Zelda.

Seguindo a ordem Zig-Zag (Fig. 4.2) o bloco seria transformado no seguinte vetor:

$$102, 5, 0, 1, 0, 1, 0, -1, -1, 0, 0, -1, 0, 1, 0, 0, \ldots, 0. \tag{4.37}$$

Utilizando agora o Código Run-Level, a predição DC=128 (para *intra_dc_precision=0*, Tabela IV.2) e utilizando a tabela IV.3 nesse vetor a seqüência seria codificada em:

$$[5/-26]; [(0/3),5]; [(1/1),1]; [(1/1),1]; [(1/1),-1]; [(0/1),-1]; [(2/1),-1]; [(1/1),1]; (EOB) \tag{4.38}$$

Dessa forma, esse bloco seria codificado (utilizando-se as tabelas B.12 e B.14 de [22]) como:

$$\underbrace{1110}_{L}\underbrace{\mathbf{1}1010}_{\mathbf{s}A\ 9bits};\ \underbrace{00101}_{R/L}\underbrace{\mathbf{0}001}_{\mathbf{s}A\ 8bits};\ \underbrace{011}_{R/L}\underbrace{\mathbf{0}}_{\mathbf{s}A\ 4bits};\ \underbrace{011}_{R/L}\underbrace{\mathbf{0}}_{\mathbf{s}A\ 4bits};\ \underbrace{011}_{R/L}\underbrace{\mathbf{1}}_{\mathbf{s}A\ 4bits};\ \underbrace{11}_{R/L}\underbrace{\mathbf{1}}_{\mathbf{s}A\ 3bits};\ \underbrace{0101}_{R/L}\underbrace{\mathbf{1}}_{\mathbf{s}A\ 5bits};\ \underbrace{011}_{R/L}\underbrace{\mathbf{0}}_{\mathbf{s}A\ 4bits};\ \underbrace{10}_{EOB\ 2\ bits} \Rightarrow 43\ bits \tag{4.39}$$

onde:

L = Nível Codificado VLC através da tabela B.12 de [22];

R/L = *Run-Level* Codificado VLC através da tabela B.14 de [22];

A = Deslocamento de Amplitude do módulo do Componente;

s= bit de sinal (em negrito): 0 para positivo e 1 para negativo;

Assim, esse bloco utiliza, para ser completamente codificado, apenas 43 bits, o que representa uma taxa de compressão da aproximadamente 12:1. É interessante notar que quando o L=1 só é necessário a codificação do bit de sinal.

4.3 – Comentários

Nesse capítulo foram apresentados os dois padrões utilizados no desenvolvimento dessa pesquisa, o padrão de compressão de imagens JPEG (*Joint Photographic Experts Group*) e o padrão de compressão de vídeo MPEG-2 TM5 (*Moving Picture Expert Group – Test Model 5*). Nesse último demos ênfase à etapa de compressão espacial, por se relacionar com o objetivo da presente pesquisa.

O objetivo desse primeiros quatro capítulos foi caracterizar o ambiente de trabalho de forma a situar a pesquisa dentro da Compressão Digital de Imagens. No próximo capítulo serão apresentados a descrição do procedimento de Redução de Efeito de Bloco desenvolvido, seu princípio de funcionamento, características e métodos de codificação.

Capítulo 5

Proposta de Redução de Efeito de Bloco

5.1 - Introdução

A proposta desta pesquisa é a criação de um algoritmo de redução de Efeito de Bloco em imagens compactadas por esquemas de codificação por transformada baseada em blocos com altas taxas de compressão. Com esse objetivo foi desenvolvida uma medida de borda localizada, o MBFB – Medida de Borda na Fronteira do Bloco (BBEM – *Block Boundary Edge Measure*), que se inserida no cabeçalho da imagem codificada, possibilita um procedimento de correção melhorando a qualidade subjetiva da imagem recuperada sem causar uma perda excessiva (borramento) dos seus detalhes.

Esse capítulo apresenta o princípio de funcionamento dos procedimentos propostos de medição e redução de efeito de bloco, bem como detalhes de suas implementações para dois sistemas de codificação: o padrão de compressão de imagens JPEG Básico e um sistema baseado na etapa de quantização e codificação do padrão de compressão de vídeo MPEG-2 *Test Model* 5.

Dessa forma, são apresentadas aqui as contribuições principais desse trabalho: o desenvolvimento do critério subjetivo de detecção de borda MBFB; um esquema de compressão de imagens por transformada utilizando o procedimento interpolativo de redução de efeito de bloco; e a medida de qualidade subjetiva de imagens com relação ao efeito de bloco TEB – Taxa de Erro de Bits no VBFB.

5.2 – Princípio de Funcionamento dos Algoritmos

Conforme foi colocado no Capítulo 3, o efeito de bloco é o evidenciamento das fronteiras entre blocos vizinhos das imagens codificadas por transformada, especialmente nas taxas de compressão maiores. Também foi comentado que as medidas mais usadas para a avaliação da qualidade de imagem são baseadas em critérios objetivos (matemáticos), como o SNR (Relação Sinal Ruído), e que esses métodos, em geral, nem sempre representam bem a qualidade subjetiva das imagens [4] especialmente quanto ao efeito de bloco. A partir dessa observação, surgiu a idéia de se definir uma medida de efeito de bloco que propiciasse uma melhor aproximação da qualidade subjetiva da imagem que sofre esse evidenciamento, levando-se em conta as relações entre pixels vizinhos à fronteira entre os blocos.

Suponha a fronteira entre 2 blocos de 8x8 pixels em destaque na Fig. 5.1. O objetivo é, então, determinar um critério que forneça uma medida numérica da visibilidade dessa borda e que esse critério seja o mais compatível possível com a impressão subjetiva do olho humano.

A medida mais simples que pode ser implementada se baseia na diferença entre os pixels vizinhos à fronteira linha a linha, como por exemplo os *pixels* $x_{m,n-1}$ e $x_{m,n}$ mostrado na Fig. 5.1, armazenando cada uma das diferenças entre *pixels* vizinhos na borda em uma posição de um vetor. Esse vetor de diferenças foi denominado Função de Borda (FB) e será utilizado no Capítulo 6.

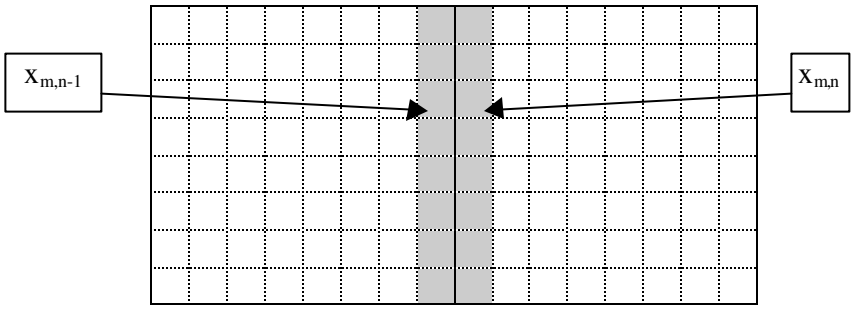
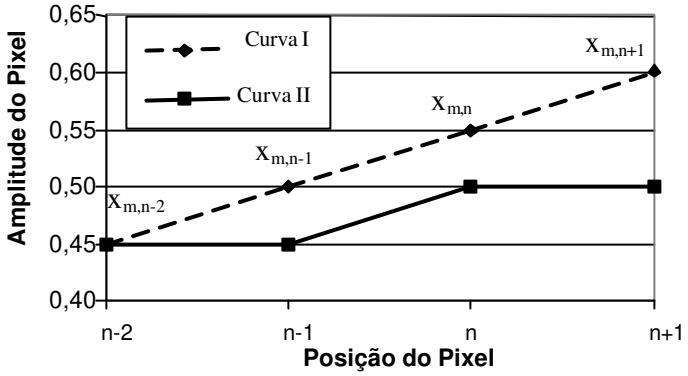


Fig. 5.1 – Dois blocos da imagem original.

O pixel $x_{m,n}$ representa o pixel da linha *m* e da coluna *n* da imagem.

No entanto, analisando-se esse critério baseado na diferença entre pixels e considerando-se a sensibilidade do olho humano, pode-se observar que nem sempre uma alta diferença entre pixels representa a presença de uma borda. Um exemplo dessa situação peculiar é mostrado na Fig. 5.2. Essa figura apresenta duas curvas possíveis para uma linha genérica *m* entre as colunas *n-2* e *n+1* que é a região de fronteira entre dois blocos da imagem. Para esse exemplo usou-se blocos de 8x8.

a)



b)

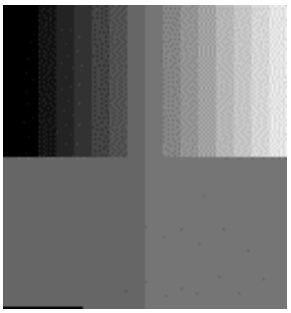


Fig. 5.2 – Situação peculiar para a análise do Critério de Diferença.

Nessa Fig. 5.2a pode-se observar que a curva I e a curva II apresentam entre $x_{m,n}$ e $x_{m,n-1}$ a mesma diferença absoluta, mas a curva I é plana e a curva II apresenta um degrau nessa transição.

Assim, na imagem proporcionada pelas duas curvas (Fig. 5.2b), percebe-se claramente na imagem correspondente à curva II (parte inferior) que a divisão entre os blocos está evidenciada, o que não ocorre na parte correspondente à curva I (parte superior).

Assim, um critério que melhor aproxime a impressão subjetiva da presença de borda será baseado na linearidade (suavidade) dos pixels nas proximidades da fronteira entre blocos. Essa observação aproxima, então, a definição da presença de uma borda visível, da presença de uma função degrau (*edge function*).

Dessa forma, para melhor representar uma borda visível, a solução encontrada foi a utilização de um critério que levasse em conta a linearidade da imagem em vez da sua diferença absoluta. Para tanto, o caminho encontrado foi a utilização de uma medida mais elaborada à qual denominou-se Medida de Borda na Fronteira do Bloco (MBFB).

Medida de Borda na Fronteira do Bloco (MBFB)

Denominou-se os quatro pontos da vizinhança da borda, como por exemplo, $x_{m,n-2}$, $x_{m,n-1}$, $x_{m,n}$ e $x_{m,n+1}$ mostrados na Fig. 5.2a, como Região Entreblocos (REB). O novo critério associa a presença de borda à uma variação entre duas derivadas consecutivas dentro da REB que seja maior do que um determinado limiar, chamado Limiar de Percepção (LP). Esse critério pode ser expresso matematicamente como:

$$MBFB = \begin{cases} 1, & se \quad \left|\Delta^m_{n-2,n-1} - \Delta^m_{n-1,n}\right| > LP \quad \underline{\boldsymbol{OU}} \quad \left|\Delta^m_{n-1,n} - \Delta^m_{n,n+1}\right| > LP \\ 0, & caso\ contrário \end{cases} \tag{5.1}$$

onde:

$$\Delta^m_{u,v} = x_{m,u} - x_{m,v},\ (u,v) \in \{(n-2,n-1),(n-1,n),(n,n+1)\} \tag{5.2}$$

A determinação do valor desse limiar foi realizada experimentalmente utilizando-se a imagem teste das Figs. 5.3 e 5.4 levada a apreciação de 20 pessoas aleatoriamente na FEEC/UNICAMP (Faculdade de Engenharia Elétrica e de Computação).

As imagens foram organizadas de forma que o nível branco é representado pelo valor 1 e o nível preto pelo valor 0. Usou-se a Fig. 5.3, onde tem-se um quadrado preto no canto superior esquerdo. O nível A tem intensidade 1%, o nível B 2%, e assim por diante. Os níveis são intercalados por faixas pretas. Na Fig. 5.4 foi feito um procedimento similar partindo-se do branco.

À essas pessoas usadas no teste, foi pedido que observassem as imagens a aproximadamente 50 cm de distância (segurando a folha com o braço esticado) e relatassem a letra do primeiro quadrado claramente distinguível do restante da imagem.

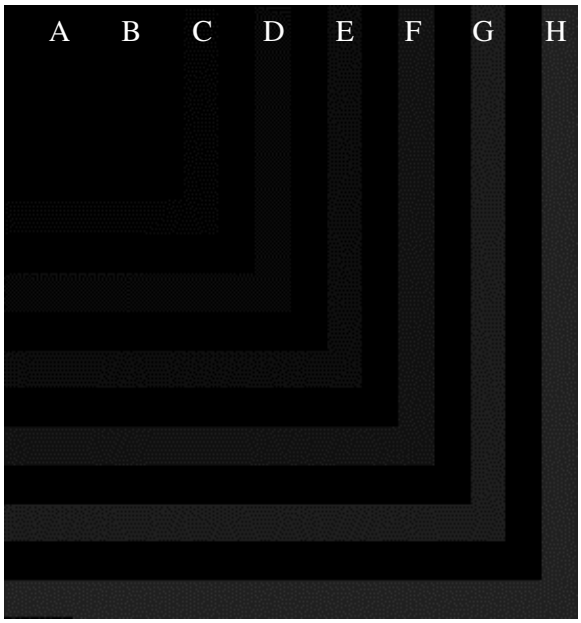


Fig. 5.3 – Figura auxiliar I para a determinação do Limiar de Percepção.

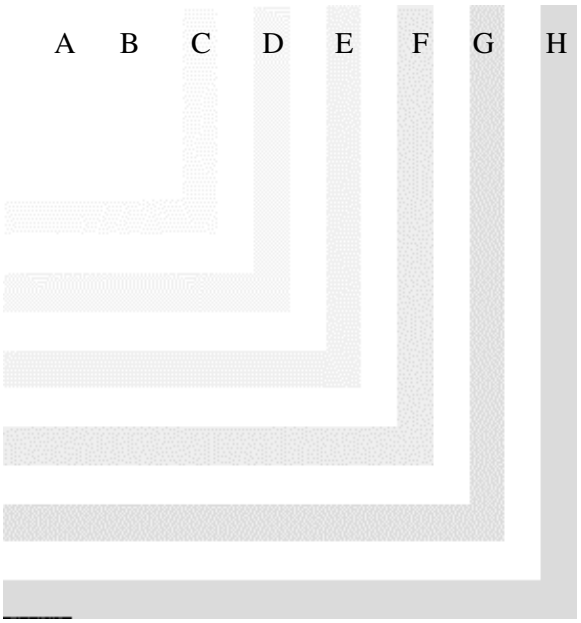


Fig. 5.4 – Figura auxiliar II para a determinação do Limiar de Percepção.

O resultado foi: Fig. 5.3: {C=3, D=8, E=8, F=1}, Fig. 5.4: {D=1, E=15, F=4}. A partir desse resultado, foram realizados alguns testes com limiares de 0,03 a 0,06 de forma a avaliar o desempenho em termos de qualidade visual e taxa de compressão da função de borda digital. Nesses testes pôde-se determinar que o valor de Limiar de Percepção que apresentava uma boa qualidade de imagem e uma taxa de compressão aceitável foi de 0,05 (Valor E). Foi considerada uma taxa de compressão aceitável nessa etapa, a taxa que levasse a um *overhead* máximo (usando compressão sem perdas para a FBD) de 0,1 bpp.

Assim, definido o critério de determinação de presença e ausência de bordas em cada REB da imagem, para que se tenha uma função que represente a imagem como um todo, é necessário, então um Critério de Varredura que passe por todas as Regiões Entreblocos da imagem e armazene cada um dos MBFBs em uma posição correspondente de um vetor. Esse vetor foi denominado VBFB – Vetor de Borda na Fronteira dos Blocos.

O critério de Varredura utilizado foi percorrer a imagem primeiramente nas colunas entre blocos e posteriormente nas linhas como ilustrado na Fig. 5.5.

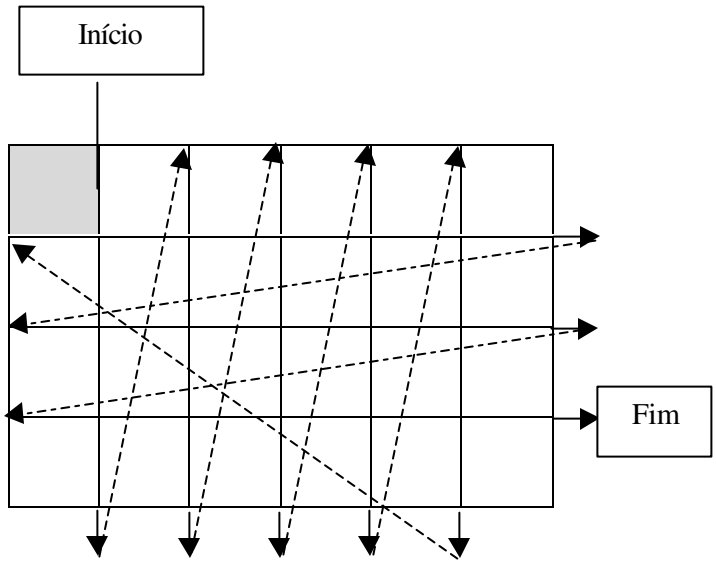


Figura 5.5 – Critério de Varredura das Regiões Entreblocos.

É importante salientar que nessa figura, cada quadrado como aquele quadrado em destaque no canto superior esquerdo é um bloco de 8x8 pixels da imagem. Com esse Critério de Varredura, o VBFB é representado, para uma imagem de dimensões MxN, por um vetor de comprimento L:

$$L = \left(\frac{M}{8} - 1\right) * N + \left(\frac{N}{8} - 1\right) * M = \frac{M * N}{4} - N - M \qquad (5.3)$$

Para uma imagem de 256x256, por exemplo, o comprimento do vetor é de 15872. Se esse vetor fosse inserido no cabeçalho da imagem codificada sem qualquer tratamento, ele representaria um acréscimo de 0,24bpp ao bitstream codificado. Esse número é grande considerando-se que é comum a utilização de taxas de compressão abaixo de 1bpp. Dessa forma, é necessária a definição de um procedimento de compressão desse vetor, que será mostrado na seção 5.4.

Apresenta-se agora uma análise das características do VBFB, do processo de deteção da geração de borda e do procedimento de redução de efeito de bloco proposto.

5.3 – Análise do VBFB e Deteção da Geração de Borda

O vetor VBFB, como foi definido na seção passada, possibilita a determinação exata das fronteiras entre blocos da imagem original nas quais existe uma borda (segundo o critério MBFB). Dessa forma, estando a informação presente no decodificador, calculando-se o VBFB para a imagem recuperada após a decodificação por transformada e comparando-se com o VBFB original, pode-se determinar os pontos na imagem reconstruída onde foi gerado o efeito de bloco. Este efeito ocorre nos pontos onde não havia borda na imagem original (bit 0 em determinada posição do VBFB) e passou-se a ter borda na imagem recuperada (bit 1 na posição correspondente do VBFB).

Na Fig. 5.6 mostra-se o diagrama de mudança de estados entre as VBFBs da imagem *Zelda* original e a compactada pelo JPEG a 0,6 bpp.

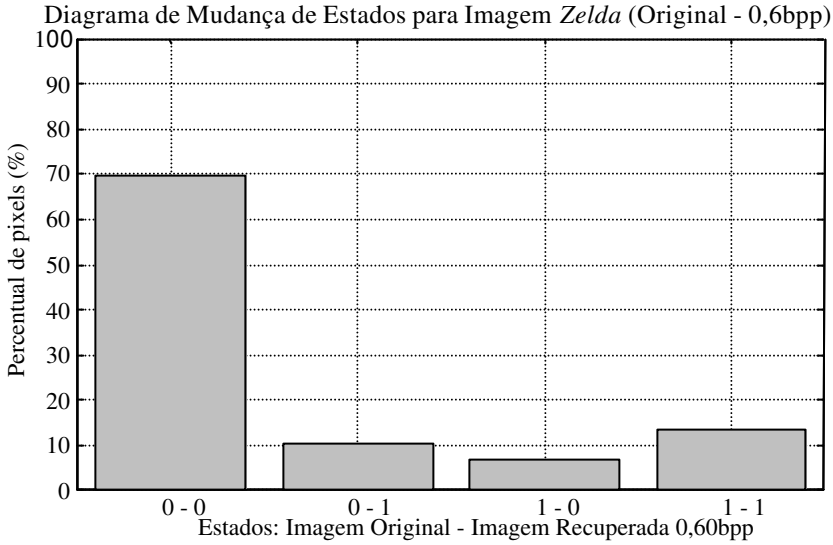


Fig. 5.6 – Diagrama de transição de estado da Função de Borda.

Nesse gráfico (Fig. 5.6) as transições ‘0-1’ e ‘1-0’ representam respectivamente a geração e a eliminação de uma borda original, ou seja, essas transições indicam o erro devido à codificação.

Tratando-se de imagens, na prática a transição ‘1-0’ representa que houve um borramento da imagem original, ou seja, uma borda que havia na imagem original foi suavizada e perdida na imagem recuperada. Já a transição ‘0-1’, como foi dito, representa a geração de uma borda, o que provoca o efeito de bloco.

É interessante notar que, de posse dos dois vetores no decodificador, é possível identificar não somente a geração de uma borda na REB, mas também, a eliminação de uma borda original em uma dessas regiões. No entanto, nota-se na Fig. 5.6 que a quantidade de transições ‘0-1’ (bordas geradas) é bem maior do que as ‘1-0’ (borramento). Essa tendência foi observada em todas as imagens testadas. Isso pode ser explicado pelo fato de que, quantizando-se independentemente dois blocos vizinhos, é menos provável que o valor dos pixels da fronteira entre eles convirjam para um mesmo ponto do que para pontos diferentes.

De posse das informações da Fig. 5.6, pode-se então definir a taxa de bits errados no VBFB devido à codificação, que representa em termos visuais, uma medida da distorção visual das fronteiras da imagem. Essa medida foi designada Taxa de Erro de Bits do VBFB (ou TEB) e será definida matematicamente no Capítulo 6.

Localizados os pontos de geração do efeito de bloco, para atingir o objetivo final desse algoritmo (a redução do efeito de bloco), utilizou-se uma interpolação linear dos valores dos pixels da REB que apresentaram a geração de borda. Essa interpolação utilizou como base os dois pixels vizinhos externos na direção em que ocorreu a borda. Considerando os pontos $x_{m,n-2}$, $x_{m,n-1}$, $x_{m,n}$ e $x_{m,n+1}$ como os valores dos pixels em uma linha da REB de uma imagem como mostrado na Fig. 5.2a, a interpolação linear dos pontos $x_{m,n-1}$ e $x_{m,n}$ em função de $x_{m,n-2}$ e $x_{m,n+1}$ pode ser expressa matematicamente por:

$$x_{m,n} = \frac{2 \cdot x_{m,n-2} + x_{m,n+1}}{3} \quad e \quad x_{m,n-1} = \frac{x_{m,n-2} + 2 \cdot x_{m,n+1}}{3} \tag{5.4}$$

A escolha da interpolação como o procedimento adotado para a redução do efeito de bloco foi realizada em função do critério de deteção de borda. Analisando o procedimento de interpolação através do MBFB, pode-se ver que a interpolação, descrita na equação (5.4), iguala todas as derivadas na REB. Dessa forma, pode-se observar que a interpolação elimina as bordas da REB segundo o critério MBFB, independente do valor dos pixels.

Definidas todas essas etapas, pode-se então montar um sistema de compressão por transformada genérico utilizando esses procedimentos e visando uma redução do efeito de bloco bem como uma melhora da qualidade subjetiva da imagem. Esse esquema é mostrado na Fig. 5.7.

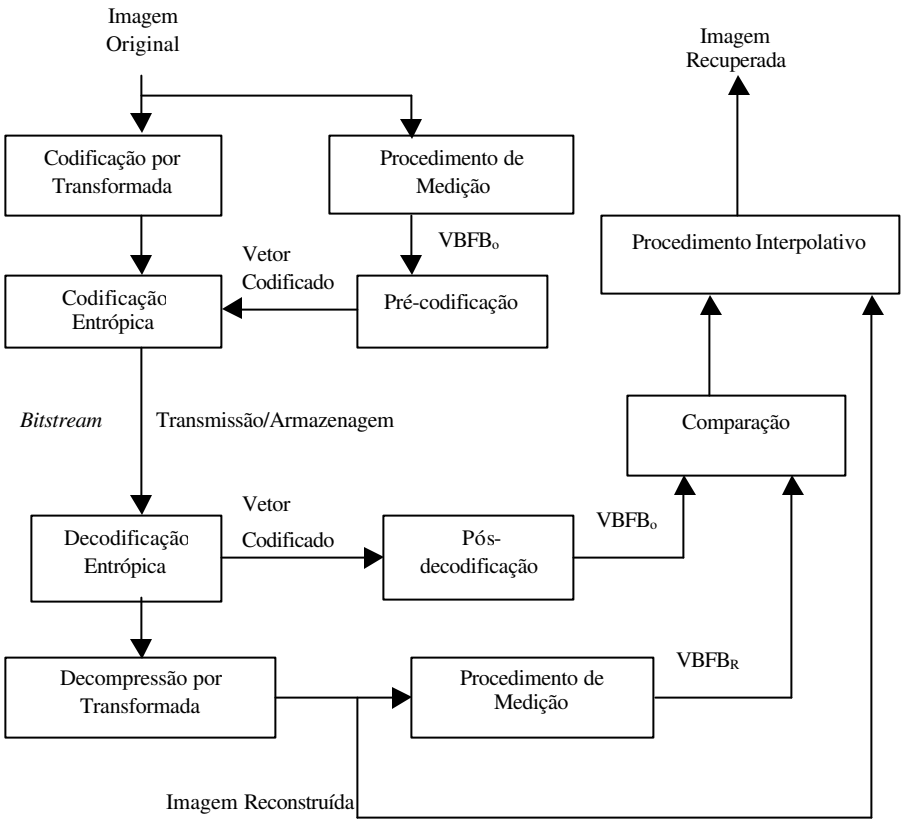


Fig. 5.7 – Esquema genérico de codificação por transformada com redução de Efeito de Bloco.

5.4 – Codificação do Vetor de Borda na Fronteira dos Blocos (VBFB)

A VBFB, em sua forma original, é um vetor de comprimento *L*, dado na equação (5.3), cujos componentes são binários. Uma característica bem pronunciada dos elementos desse vetor é a alta incidência de *0*'s (zeros) consecutivos. Isso ocorre porque as imagens naturais são, normalmente, muito planas e suaves fazendo com que não haja muitas bordas naturais. Além disso, para que uma borda da imagem original tenha algum reflexo na VBFB ela deve estar dentro da Região Entreblocos da imagem, o que explica a baixa ocorrência de *1*'s (uns) no VBFB.

Essa característica do VBFB facilita a aplicação da técnica de codificação sem perdas do tipo *Run-Length* (Capítulo 4). Como o VBFB é binário, pode-se utilizar uma forma simplificada chamada *Run-Length Binário* (RLB) [25].

No RLB, como só existem dois símbolos possíveis: 0 e 1, dado o bit inicial, não é necessária a informação de símbolo e portanto, a codificação é feita apenas contando-se os *Runs* entre duas mudanças de símbolo consecutivas, como no exemplo a seguir:

$$\begin{array}{ll} \textit{Bitstream} & \overbrace{0000000}\overbrace{1111}\overbrace{00000}\overbrace{1}\overbrace{00}\overbrace{1111110}\cdots \\ \textit{Runs} & \quad 7 \quad\; 4 \quad\; 5 \;\; 1 \; 2 \quad 6 \end{array} \tag{5.5}$$

A compressão RLB é muito efetiva nas regiões de alta correlação, mas nas regiões de maior detalhamento das imagens, tende a gerar alguns trechos de trocas sucessivas de bits, ou seja, vários códigos 1 seguidos. Como o menor número da contagem é 1 (não pode haver menos de um bit diferente), a seqüência RLB é subtraída de 1 em cada um dos seus termos e então submetida à uma codificação do tipo *Run-Level.* Na Imagem *Zelda*, mostrada na Fig. 4.5, uma das áreas que apresentam essa característica é a região correspondente ao cabelo.

Na Fig. 5.8, é mostrada a freqüência de ocorrência dos *Runs* para o VBFB da imagem *Zelda*. Apesar de haver *Runs* de até 474 bits para essa imagem, foi representada apenas a faixa de 1 a 40 por facilitar a visualização e também porque os *Runs* acima de 40 bits representam apenas 2,93% (69 ocorrências em 2356) do total e estão muito espalhados.

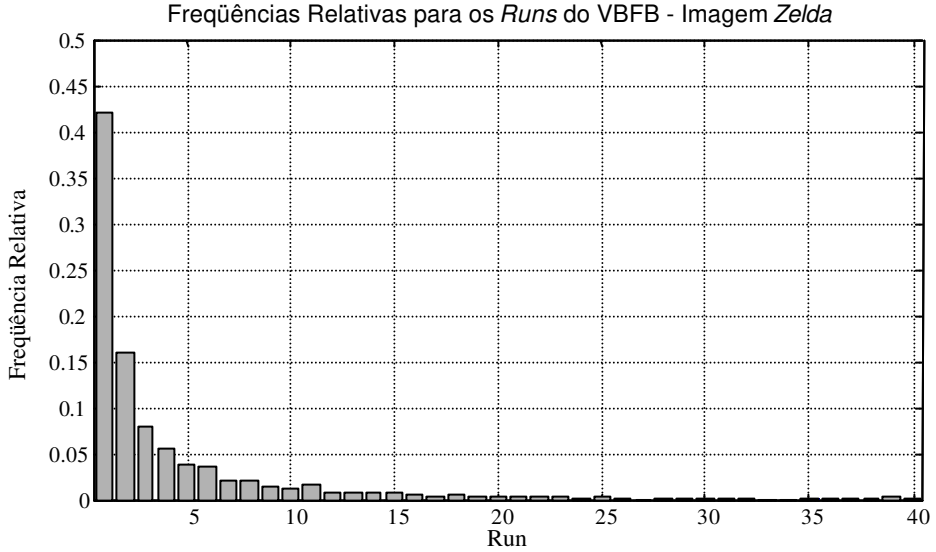


Fig. 5.8. Distribuição parcial de *Runs* para a imagem *Zelda.*

Como o vetor de *Runs* é composto somente por valores positivos, além de subtrair 1 dos valores do vetor, a representação do *Level* pode dispensar o bit de sinal, o que na prática, é equivalente a se determinar o *Level* através da Tabela IV.3 e diminuí-lo em 1. Para exemplificar esse procedimento, consideremos alguns elementos do vetor de *Runs* da imagem *Zelda*:

$$94, 4, 4, 1, 1, 1, 4, 2, 1, 1, 14, 1, 214, 3, 1, 1, 1, 1, 9, \ldots \tag{5.6}$$

Subtraindo-se 1 de cada um desses valores e aplicando-se o *Run-Level* modificado como descrito anteriormente, pode-se mapear essa seqüência pelos símbolos:

(0/6,93), (0/2,3), (0/2,3), (3/2,3), (0/1,2), (2/3,13), (1/7,213), (0/1,2), (4/3,8), ... (5.7)

De acordo com o padrão de compressão utilizado, pode-se aplicar à essa seqüência uma codificação VLC diferente de modo a balancear a eficiência de compressão com a facilidade de implementação, a capacidade de processamento e as restrições de tempo de codificação.

Para os resultados baseados no padrão MPEG-2, foi utilizada a tabela código B.14 especificada em [22] para se codificar o par R/L sendo que o valor de Amplitude foi codificado de forma semelhante à descrita no Capítulo 4 removendo-se o bit de sinal. Como o padrão JPEG estipula que a tabela código deve ser incluída no cabeçalho, foi implementado para o par R/L um codificador de Huffmann e a tabela no vetor codificado foi incluída, enquanto que o valor da Amplitude foi codificado normalmente e excluindo-se, novamente, o bit mais significativo.

Utilizando-se essa codificação alcançou-se uma taxa de compressão sem perdas de aproximadamente 2:1, o que resulta num *overhead* menor que 20% (em torno de 0,2bpp) pela utilização do procedimento para as taxas de compressão menores (menores que 8:1). No entanto para taxas de compressão maiores, o *overhead* se torna muito significativo inviabilizando o processo. Para minorar esse problema foi utilizado um sistema de codificação com perdas que consiste em subamostrar o VBFB antes da codificação sem perdas.

Essa subamostragem é feita com base na média entre os pixels que serão subamostrados. Na subamostragem de fator 2, usou-se a média entre os pixels 2 a 2 formando um novo vetor com a metade do tamanho do vetor original. Passou-se cada média por um critério de limiar sendo que se a média for menor ou igual a 0,5, então o valor equivalente é 0, se for maior, o valor é 1. Na prática, o valor do pixel subamostrado só é 1 se os dois pixels originais forem também 1. Na subamostragem de fator 4 faz-se a média de 4 em 4 pixels e associa-se o valor 1 à médias maiores que 2 e 0 às menores ou iguais a 2.

Utilizando essa subamostragem a taxa de compressão do VBFB chega a aproximadamente 8:1 resultando num *overhead* em torno de 0,05bpp o que possibilita a aplicação desse procedimento em imagens codificadas à taxas altas de compressão (maiores que 8:1). No quadro a seguir exemplificam-se as subamostragens de fator 2 e de fator 4 para o vetor mostrado na equação (5.5):

Original	0 0 0 0	0 0 0 1	1 1 1 0	0 0 0 0	1 0 0 1	1 1 1 1	1 0 0 0	**...**
Sub. Fator 2	0 0	0 0	1 0	0 0	0 0	1 1	0 0	**...**
Sub. Fator 4	0	0	1	0	0	1	0	**...**

(5.8)

5.5 – Comentários:

Foram apresentadas nesse capítulo as contribuições principais desse trabalho: o desenvolvimento do critério subjetivo de deteção de borda MBFB; um esquema de compressão de imagens por transformada utilizando o procedimento de redução de efeito de bloco; e a medida de qualidade subjetiva de imagens com relação ao efeito de bloco TEB – Taxa de Erro de Bits no VBFB.

Um resumo da evolução da proposta de medição de efeito de bloco desenvolvida durante essa pesquisa até o MBFB final também foi apresentado, bem como o processo de interpolação utilizado para a redução do efeito de bloco, o critério de varredura usado na formação do VBFB e detalhes da subamostragem e codificação sem perdas aplicadas em cada um dos padrões utilizados: JPEG e MPEG-2.

Dessa forma, foi realizada a especificação completa do sistema proposto, restando agora a etapa de Apresentação e Análise dos Resultados obtidos com a aplicação desse procedimento, que será o tema fundamental do próximo capítulo.

Capítulo 6

Simulações e Resultados

6.1 – Introdução

Nesse capítulo são apresentados os resultados obtidos pela aplicação do procedimento proposto para redução de efeito de bloco nos ambientes de compressão: JPEG (padrão de compressão de imagens) e um sistema baseado na quantização e codificação do MPEG-2 TM5. O objetivo de realizar testes com o MPEG-2 é possibilitar a implementação futura de uma versão desse procedimento para vídeo.

Os resultados obtidos são analisados com base na qualidade visual das imagens recuperadas, na taxa de compressão (medida em *bits por pixel–bpp*), na Taxa de Erro de Bits da VBFB (TEB - medida em pontos percentuais) e no *PSNR* (*Peak Signal-to-Noise Ratio*). Também foram analisados os histogramas de brilho das imagens (apresentados no Anexo B). A escolha do PSNR é devido à sua extensa utilização tanto nos artigos quanto na bibliografia de PDI como medida de qualidade objetiva das imagens. O PSNR entre duas imagens X e Y de tamanho MxN com pixels representados em 8 bits pode ser expresso como função do MSE (*Mean Square Error)*:

$$PSNR(X,Y) = 10 \cdot log_{10}\left(\frac{255^2}{MSE(X,Y)}\right), \text{ onde:} \tag{6.1}$$

$$MSE(X,Y) = \frac{1}{M \cdot N}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}\left(x_{jk} - y_{jk}\right)^2 \tag{6.2}$$

É importante lembrar que um valor de PSNR alto não implica necessariamente em uma boa qualidade de imagem reconstruída [6], pois não leva em consideração o conjunto geral da imagem, mas apenas o valor de seus pixels isoladamente. Esse fato é de fundamental relevância para esse trabalho por constituir um dos motivos da realização dessa pesquisa. Uma medida que revela um pouco melhor a qualidade subjetiva da imagem é a Taxa de Erro de Bit do VBFB. Sendo V_o o VBFB da imagem original e V_r o VBFB da imagem recuperada, a expressão da TEB fica:

$$TEB = \frac{100\%}{L} \cdot \sum_{k=1}^{L}\left| V_o(k) - V_r(k) \right| \tag{6.3}$$

Para o texto a seguir, usa-se a expressão “Imagem Detalhada” para indicar imagens que possuem muitos objetos pequenos, pontos e linhas (detalhes) distribuídos na imagem. Uma “Imagem Suave”, ao contrário, é uma imagem que não possui muitos detalhes. Divide-se a apresentação dos

Resultados e Comentários em duas seções para cada padrão de compressão usado.

6.2 – Resultados para o JPEG

Nos experimentos com o padrão JPEG foram utilizadas com 5 imagens retiradas da base de dados do Laboratório de Comunicações Visuais (LCV/DECOM/FEEC/UNICAMP). As imagens utilizadas são conhecidas da literatura de Processamento Digital de Imagens e são de domínio público podendo ser encontradas em *sites* especializados da Internet.

Para os resultados apresentados aqui utilizou-se como ambiente de programação o *Matlab 5.0* principalmente as *Toolboxes* de processamento de sinais e de imagens além de outras funções que foram desenvolvidas em *Matlab script* para as simulações. As imagens foram codificadas em 3 taxas de compressão diferentes (0,8 bpp, 0,6 bpp e 0,4 bpp) utilizando-se blocos de tamanho 8x8.

Na etapa de codificação, os dados das imagens foram codificados utilizando-se uma tabela de Huffmann calculada especificamente para cada imagem pelo algoritmo JPEG Baseline e o VBFB foi subamostrado por um fator de 4 e codificado em *Run-Level* usando-se uma tabela de Huffmann própria (como descrito no Capítulo 5). Essa tabela foi anexada ao vetor codificado no cabeçalho da imagem codificada.

Tendo em vista que esse trabalho visa obter uma melhora da qualidade visual das imagens, são apresentadas ampliações de algumas regiões das imagens de forma a possibilitar a visualização detalhada do efeito de bloco e das diferenças na qualidade subjetiva das imagens. Na Fig. 6.1 são mostradas todas as imagens originais com as quais se trabalhou com os correspondentes nome e dimensões em *pixels*. Essas imagens são monocromáticas e estão apresentadas aqui em 8 bits por pixel (256 níveis de cinza).

Serão apresentadas, a partir da Fig. 6.2, duas Ampliações (I e II) para cada imagem recuperada seguindo a ordem da Fig. 6.1, para cada uma das taxas de compressão utilizadas. Cada região ampliada tem, na imagem original, as dimensões 48x64 pixels. Juntamente com as imagens recuperadas são colocadas as informações de PSNR e Taxa de Erro de Bits para facilitar a avaliação da qualidade da imagem.

Para cada imagem recuperada apresenta-se um gráfico da Função de Borda e um gráfico do VBFB para uma das regiões mostradas nas Ampliações. Por uma questão de facilidade de visualização, todos os gráficos apresentam duas curvas (uma para o JPEG e outra para o Procedimento Proposto), e serão baseados em uma das ampliações das imagens com taxas de compressão de 0,6 bpp. Também são apresentados alguns comentários relevantes a respeito de cada ampliação. A análise completa dos resultados obtidos e a tabela contendo o resumo de todos os

resultados numéricos são apresentadas após as imagens, na subseção 6.3.

(a) Imagem *Flowers* – 512 x 384 *pixels*

(b) Imagem *Foto Aérea* – 512 x 512 *pixels*

(d) Imagem *Pirate* – 1024 x 1024 *pixels*

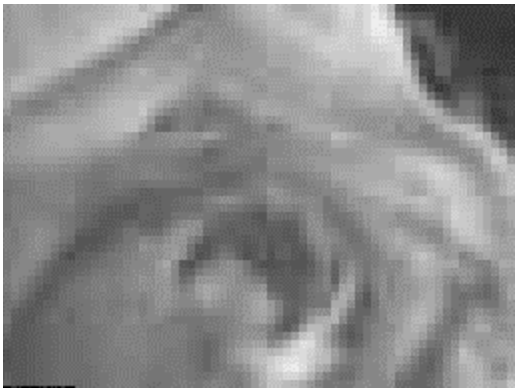
(c) Imagem *Lena* – 512 x 512 *pixels*

(e) Imagem *Zelda* – 256 x 256 *pixels*

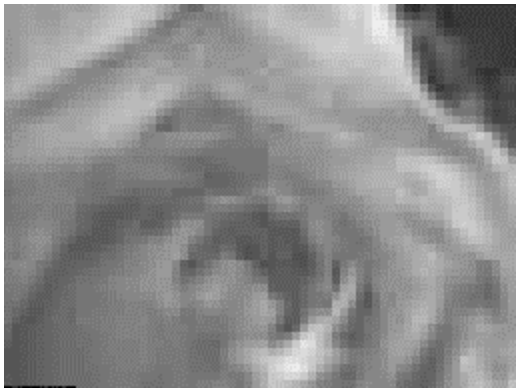
Fig. 6.1 – Imagens originais utilizadas para os testes com o algoritmo JPEG.

Imagem *Flowers*

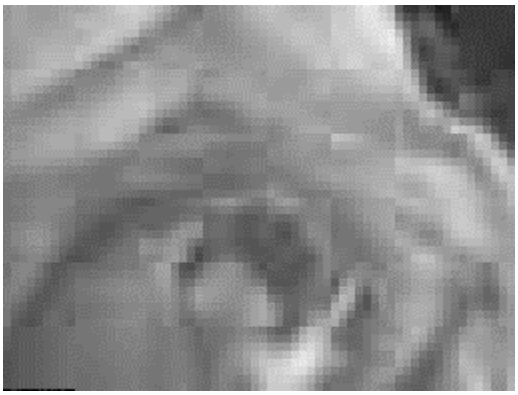
Essa imagem tem características interessantes no que se refere ao estudo do efeito de bloco por apresentar áreas completamente planas, como o pano de fundo da foto; áreas de poucos detalhes como as superfícies das pétalas; e áreas de grande detalhamento como as folhas dos ramos. A Ampliação I se refere à rosa do canto superior direito da imagem original. Nas legendas subsequentes, "Prop" indica os resultados para o sistema proposto.



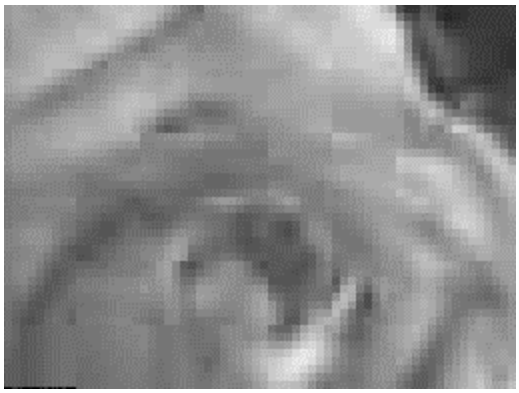
(a) JPEG, 0,8 bpp PSNR=34,3dB TEB=24,0%



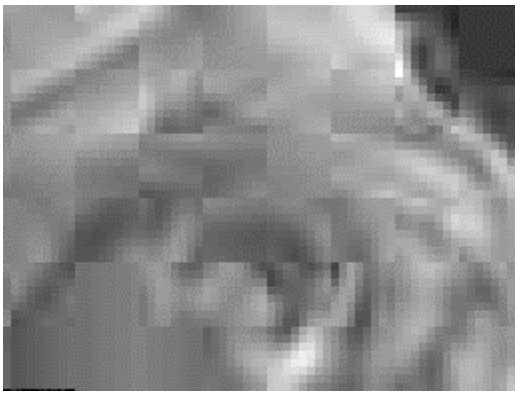
(b) Prop., 0,8 bpp PSNR=33,5dB TEB=17,7%



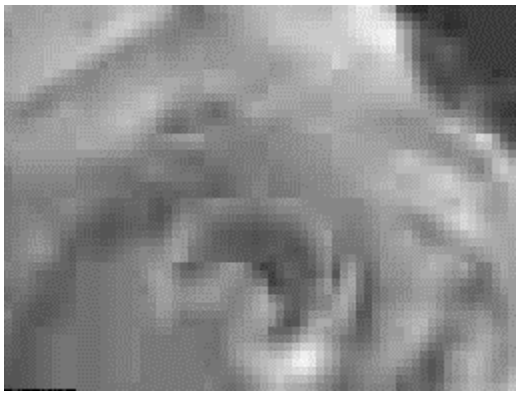
(c) JPEG, 0,6 bpp PSNR=32,5dB TEB=27,4%



(d) Prop, 0,6 bpp PSNR=32,0dB TEB=17,9%



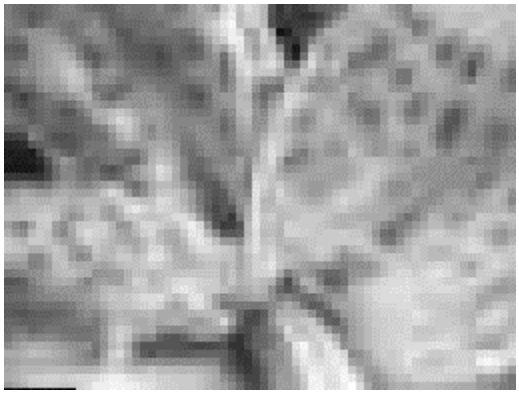
(e) JPEG, 0,4 bpp PSNR=30,0dB TEB=31,6%



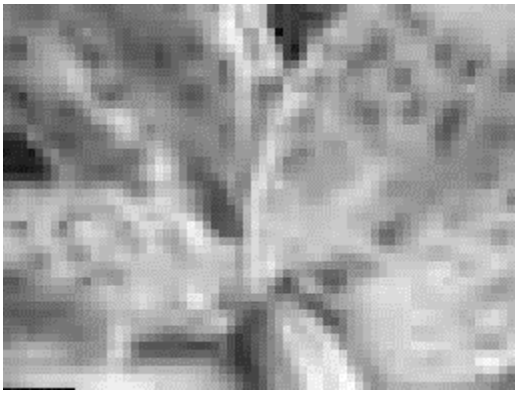
(f) Prop, 0,4 bpp PSNR=29,8dB TEB=18,0%

Fig. 6.2 – Ampliação I da imagem *Flowers* para o Procedimento Proposto e para o JPEG.

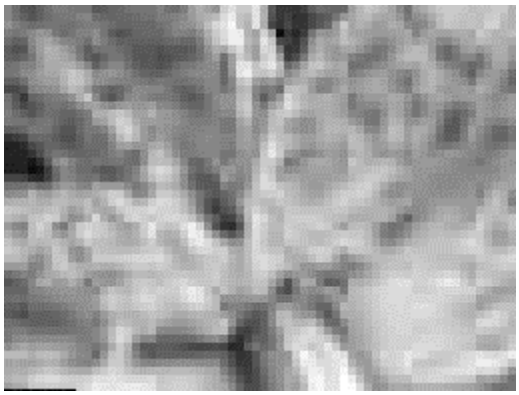
A Ampliação II é a orquídea na parte central da imagem.



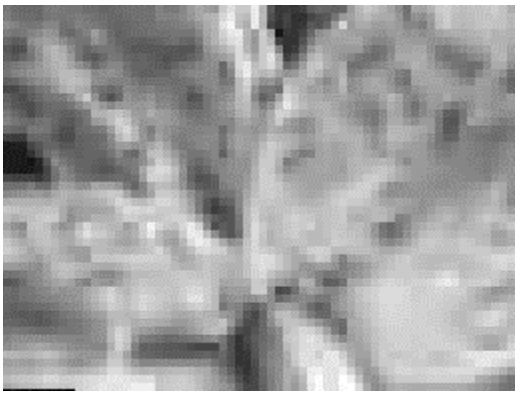
(a) JPEG, 0,8 bpp PSNR=34,3dB TEB=24,0%



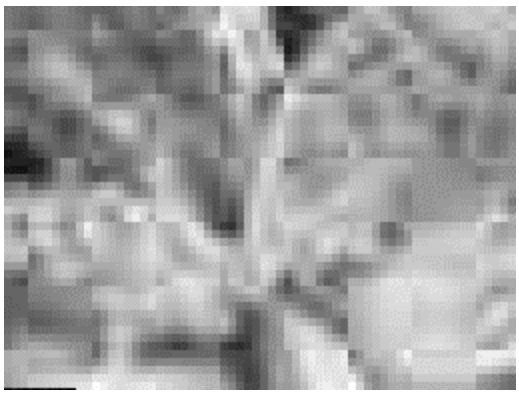
(b) Prop., 0,8 bpp PSNR=33,5dB TEB=17,7%



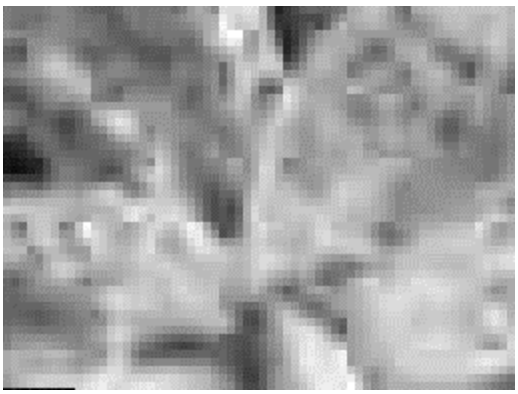
(c) JPEG, 0,6 bpp PSNR=32,5dB TEB=27,4%



(d) Prop, 0,6 bpp PSNR=32,0dB TEB=17,9%



(e) JPEG, 0,4 bpp PSNR=30,0dB TEB=31,6%



(f) Prop, 0,4 bpp PSNR=29,8dB TEB=18,0%

Fig. 6.3 – Ampliação II da imagem *Flowers* para o Procedimento Proposto e para o JPEG.

A escolha dessas duas regiões para a ampliação se deu por apresentarem características diferenciadas com relação ao nível de detalhamento. A primeira ampliação possui poucos detalhes sendo composta em sua maior parte por áreas com transições suaves o que faz com que o efeito de bloco seja mais visível mesmo nas taxas mais altas. Essa região foi escolhida para demonstrar a

eficiência do algoritmo proposto na eliminação de muitas das bordas indesejáveis e a conseqüente redução do efeito de bloco.

Já na Ampliação II tem-se uma região rica em detalhes especialmente devido aos pontos escuros das pétalas da orquídea. Essa região é interessante visto que pode-se observar a redução do efeito de bloco com o algoritmo proposto sem ocorrer uma excessiva perda de detalhes, especialmente nas taxas de compressão maiores. Na Fig. 6.4, mostra-se os Gráfico da Função de Borda e do VBFB da imagem *Flowers* (codificada a 0,6bpp) de uma das REB que atravessam a Ampliação I para as imagens recuperadas pelo JPEG e pelo procedimento Proposto. É importante lembrar aqui que, para o VBFB, o bit 1 significa presença de borda e o bit 0, ausência.

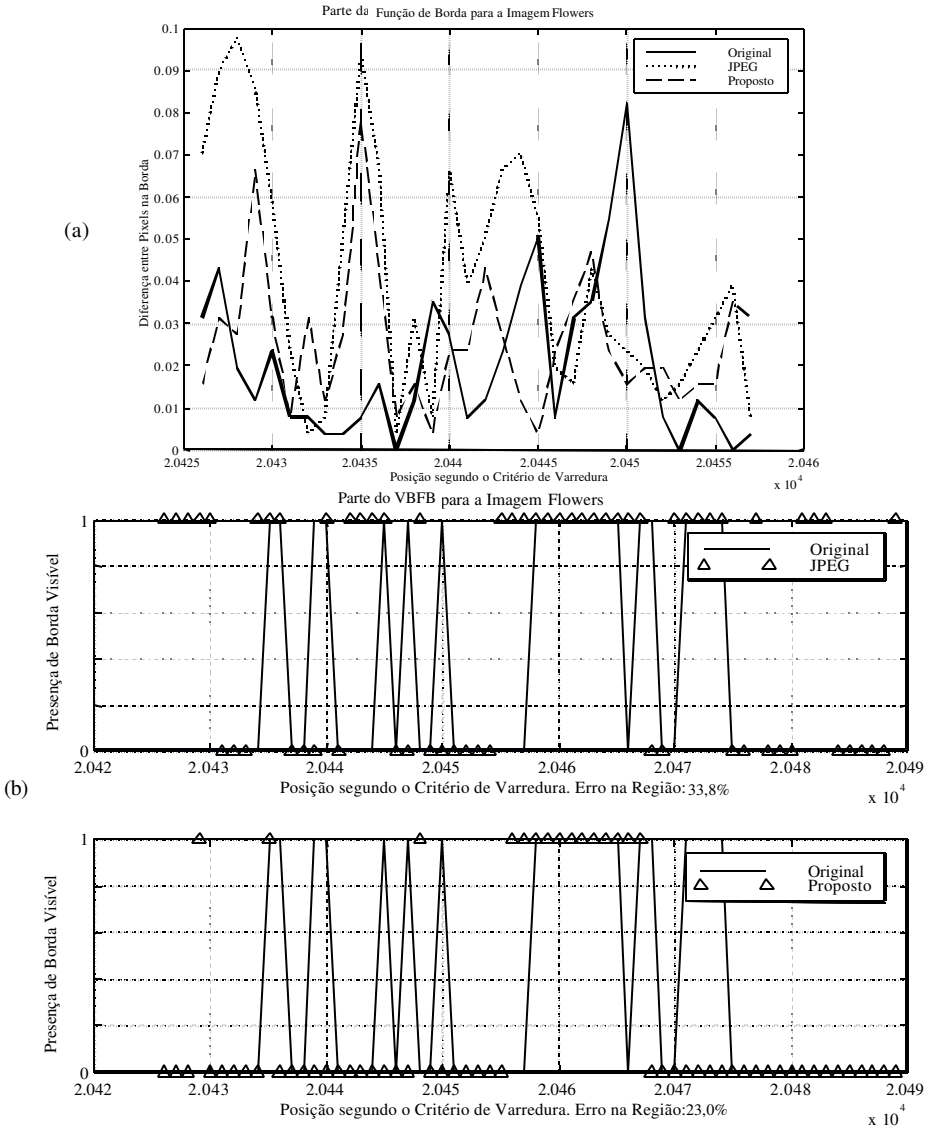


Fig. 6.4 – (a) Função de Borda; (b) VBFB para uma região da Ampliação I da imagem *Flowers.*

Como pode-se observar na Fig. 6.4a, o gráfico da Função de Borda para o procedimento proposto é, de forma geral, mais próximo à FB da imagem Original que o Gráfico do JPEG. Isso implica que as diferenças entre pixels das fronteiras dos blocos na imagem recuperada pelo sistema proposto aproximam melhor as diferenças existentes na imagem original, desfavorecendo, assim, a geração de bordas que não existiam previamente.

Já o gráfico do VBFB (Fig.6.4b) mostra como o procedimento proposto ajuda a eliminar grande parte das bordas geradas (reduzindo o TEB da região de 33,8% para 23,0%) sem, no entanto, atenuar a borda original (situada entre 20460 e 20470), levando à uma qualidade visual bem melhor do que a do sistema JPEG original. Também pode-se notar que o procedimento não é totalmente livre de perdas, e isso ocorre devido à codificação *lossy* do VBFB (subamostragem). É interessante notar aqui que o fator de subamostragem do VBFB determina o maior comprimento de borda que pode ser perdido, como será comentado mais adiante no item 6.3.

A seguir são apresentados os resultados obtidos para a imagem *Foto Aérea*, que é muito utilizada na literatura, e apresenta características diferentes da *Flowers*.

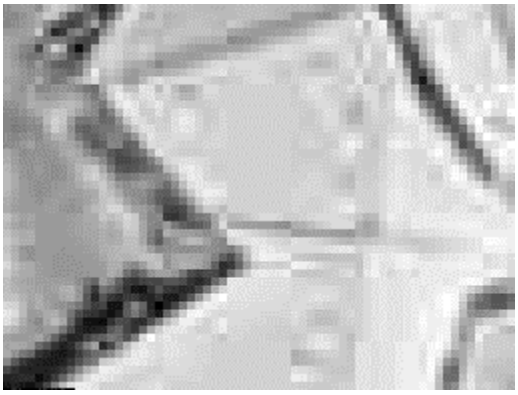
Imagem *Foto Aérea*

Essa imagem, ao contrário da anterior, tem por característica a grande presença de detalhes espalhados por toda a imagem e também muitas formas geométricas irregulares como o desenho quase triangular formado pelas três estradas. Essas formas geométricas são interessantes para o estudo do efeito de bloco pois as perdas da quantização tendem a entrecortar as linhas inclinadas com relação à bordas da imagem ressaltando suas fronteiras (gerando o Efeito de Bloco).

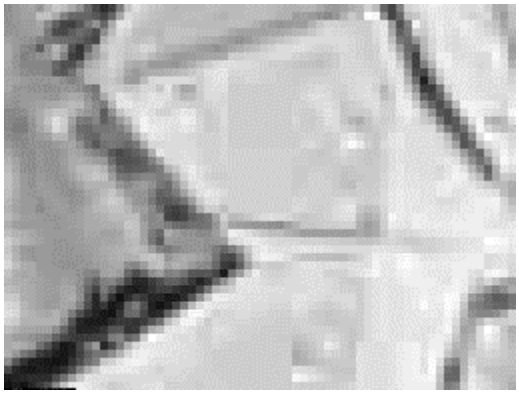
Devido à essa característica geométrica dessa imagem, os detalhes escolhidos para serem ampliados foram de duas regiões que apresentam linhas inclinadas. A primeira ampliação mostra a casa com um corte em "V" na parte centro-superior da foto. Essa região foi escolhida por apresentar uma grande quantidade de linhas com várias inclinações correspondentes aos cortes do telhado da casa.

A Ampliação II foi feita a partir da parte de cima do galpão retangular localizado na parte central-esquerda da imagem. Esse galpão, além das linhas, apresenta alguns detalhes pontuais (árvores) em suas proximidades proporcionando a possibilidade de se analisar a conservação dos detalhes da imagem.

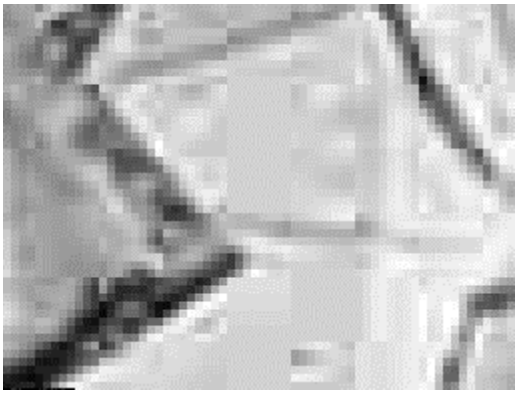
As Ampliações I e II são mostradas nas Figs 6.5 e 6.6, respectivamente.



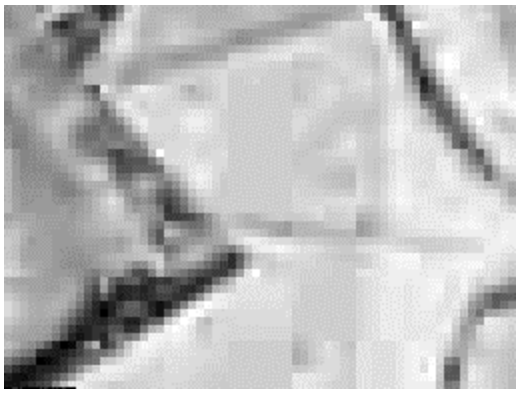
(a) JPEG, 0,8 bpp PSNR=29,2dB TEB=31,2%



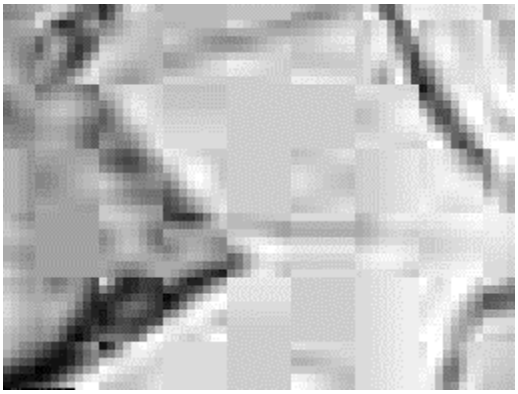
(b) Prop., 0,8 bpp PSNR=28,7dB TEB=21,4%



(c) JPEG, 0,6 bpp PSNR=27,6dB TEB=33,7%



(d) Prop, 0,6 bpp PSNR=27,3dB TEB=22,2%



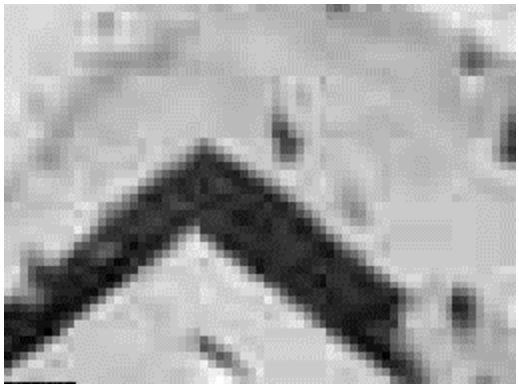
(e) JPEG, 0,4 bpp PSNR=25,5dB TEB=35,7%

(f) Prop, 0,4 bpp PSNR=24,9dB TEB=22,4%

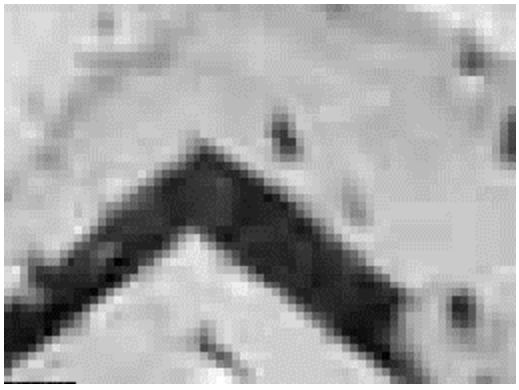
Fig. 6.5 – Ampliação I da imagem *Foto Aérea* para o Procedimento Proposto e para o JPEG.

Comparando-se as imagens recuperadas pelos dois sistemas apresentados na Fig. 6.5 pode-se observar que o procedimento proposto apresenta efeito de bloco reduzido com relação ao JPEG Básico, especialmente nas regiões próximas aos contornos da casa, restaurando a qualidade visual da imagem mantendo os detalhes originais da imagem sem grandes alterações.

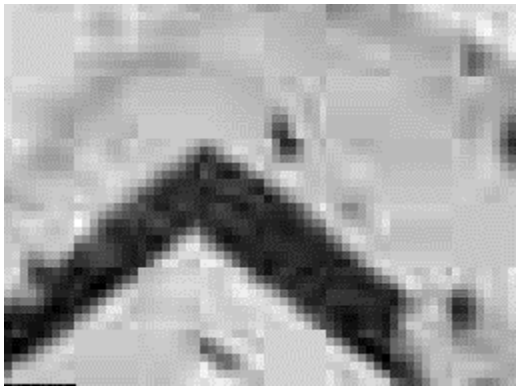
Para a Ampliação II, mostrada na Fig. 6.6, também pode-se observar a restauração dos contornos originais do galpão sem, no entanto, prejudicar a qualidade subjetiva da imagem, como as árvores do pátio e o detalhe no telhado do galpão. Note que mesmo com a taxa de compressão de 0,4 bpp, a imagem ainda conserva uma qualidade visual aceitável.



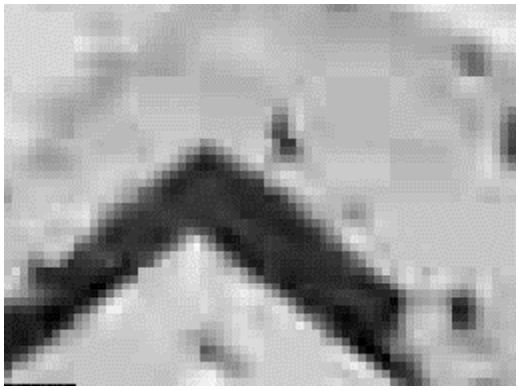
(a) JPEG, 0,8 bpp PSNR=29,2dB TEB=31,2%



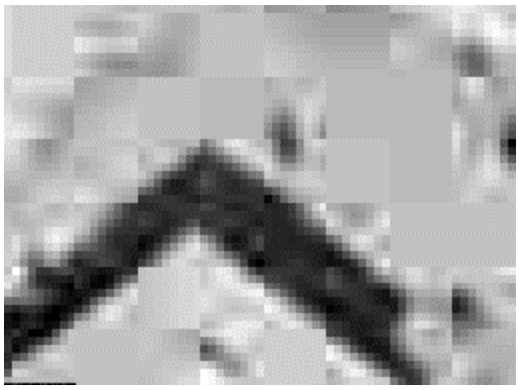
(b) Prop., 0,8 bpp PSNR=28,7dB TEB=21,4%



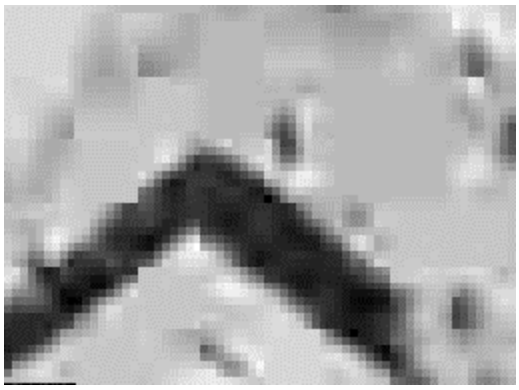
(c) JPEG, 0,6 bpp PSNR=27,6dB TEB=33,7%



(d) Prop, 0,6 bpp PSNR=27,3dB TEB=22,2%



(e) JPEG, 0,4 bpp PSNR=25,5dB TEB=35,7%



(f) Prop, 0,4 bpp PSNR=24,9dB TEB=22,4%

Fig. 6.6 – Ampliação II da imagem *Foto Aérea* para o Procedimento Proposto e para o JPEG.

Os gráficos das FBs e dos VBFBs para uma região da Ampliação I são mostrados na Fig. 6.7. Note que, para essa imagem, a FB da imagem recuperada pelo procedimento proposto é, novamente, mais semelhante à FB original do que a FB da imagem do JPEG, como observado nas ampliações. Para o gráfico do VBFB também pode-se notar aqui a eliminação de grande parte das borda geradas sem o borramento das bordas originais (redução no TEB da região de 36,0% para 15,6%), o que levou a uma melhoria considerável da qualidade da imagem como mostrado nas Ampliações I e II.

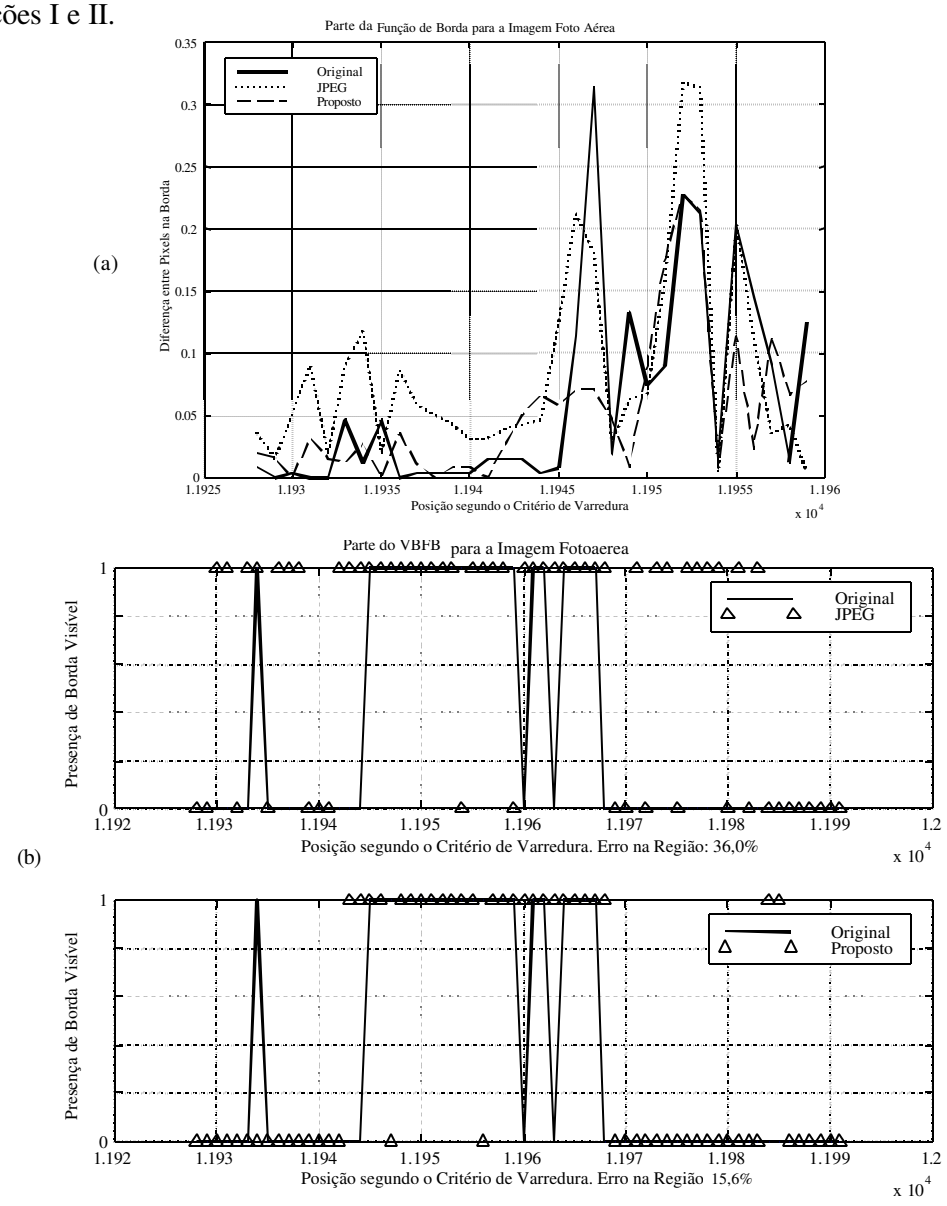
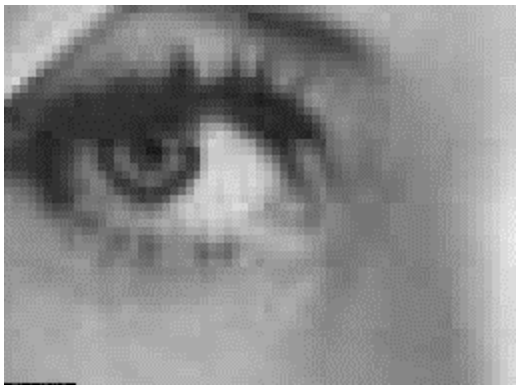


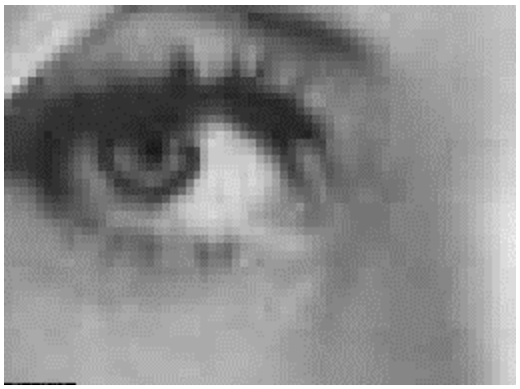
Fig. 6.7 – (a) Função de Borda; (b) VBFB para uma região da Ampliação I da imagem *Foto Aérea.*

Imagem *Lena*

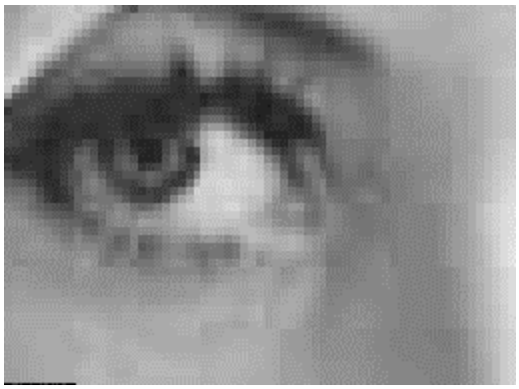
A imagem *Lena* é certamente uma das imagens mais utilizadas nos artigos e em livros de Processamento Digital de Imagens atualmente, o que justifica plenamente a sua utilização nesse trabalho. Os detalhes escolhidos para a ampliação foram a região do olho direito (região central da imagem) e a aba do chapéu.



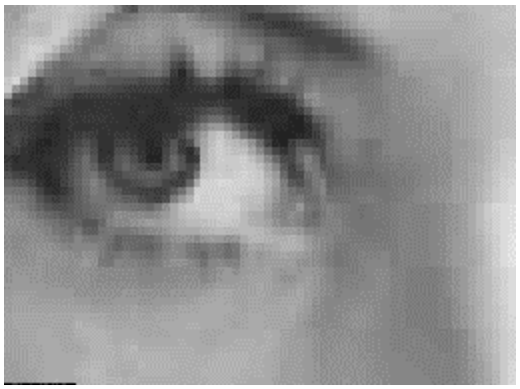
(a) JPEG, 0,8 bpp PSNR=36,8dB TEB=18,8%



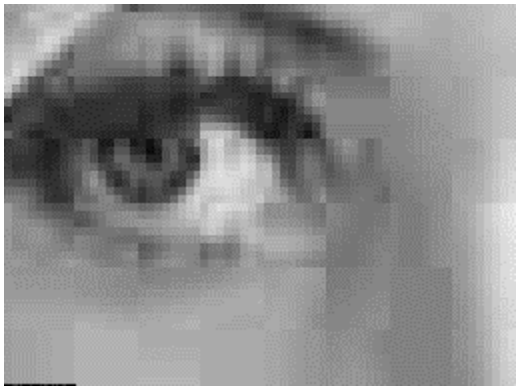
(b) Prop., 0,8 bpp PSNR=36,3dB TEB=15,9%



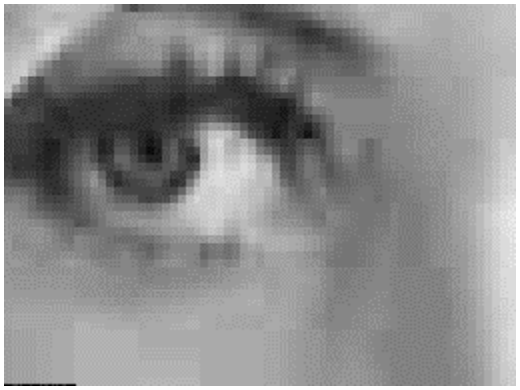
(c) JPEG, 0,6 bpp PSNR=35,6dB TEB=20,0%



(d) Prop, 0,6 bpp PSNR=35,1dB TEB=16,1%

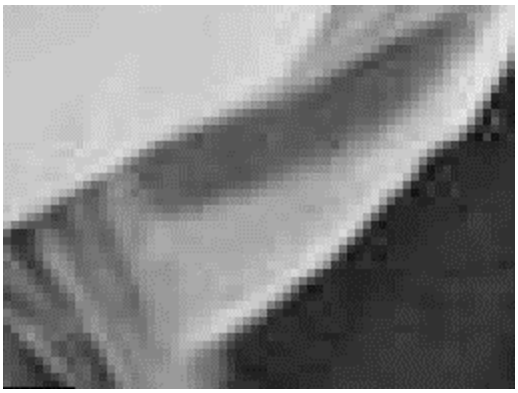


(e) JPEG, 0,4 bpp PSNR=33,5dB TEB=22,9%

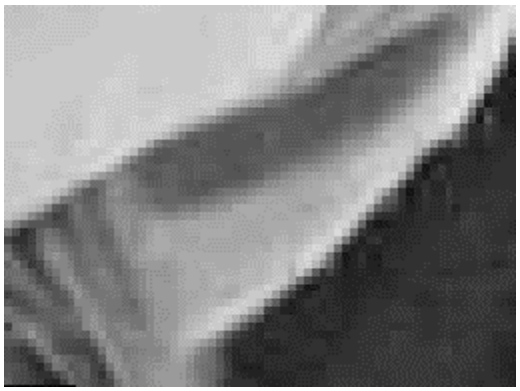


(f) Prop, 0,4 bpp PSNR=33,2dB TEB=16,6%

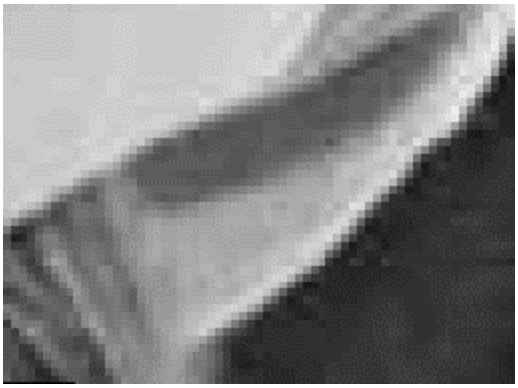
Fig. 6.8 – Ampliação I da imagem *Lena* para o Procedimento Proposto e para o JPEG.



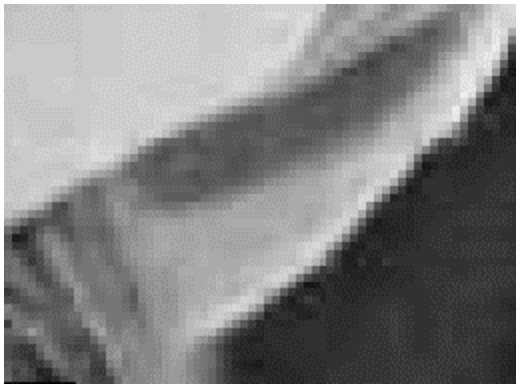
(a) JPEG, 0,8 bpp PSNR=36,8dB TEB=18,8%



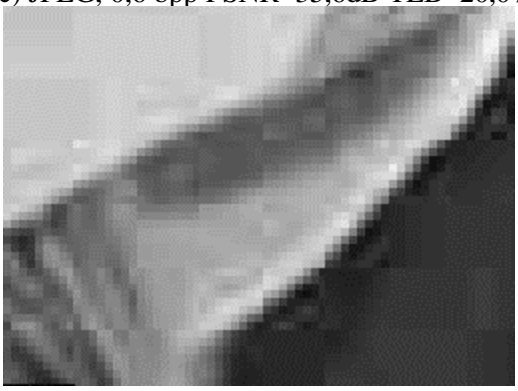
(b) Prop., 0,8 bpp PSNR=36,3dB TEB=15,9%



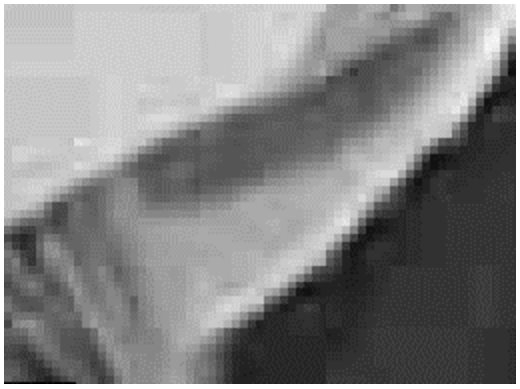
(c) JPEG, 0,6 bpp PSNR=35,6dB TEB=20,0%



(d) Prop, 0,6 bpp PSNR=35,1dB TEB=16,1%



(e) JPEG, 0,4 bpp PSNR=33,5dB TEB=22,9%



(f) Prop, 0,4 bpp PSNR=33,2dB TEB=16,6%

Fig. 6.9 – Ampliação II da imagem *Lena* para o Procedimento Proposto e para o JPEG.

Novamente pode-se observar nas ampliações a redução da blocagem da imagem, especialmente nas regiões planas e, no caso da Ampliação II, também para a linha inclinada (da aba do chapéu), como aconteceu para a imagem *Foto Aérea*. Em ambas as ampliações, a maior redução do efeito de bloco ocorre na região central das ampliações e é mais visível para a taxa de 0,4 bpp. Isso ocorre devido a característica plana e suave da imagem que são características desejáveis para a aplicação da codificação por transformada.

Os gráficos da Fig. 6.10 mostram novamente o melhor nível de aproximação da característica da imagem original (especialmente pelo gráfico das FBs). No gráfico dos VBFBs pode-se ver a baixa presença de bordas da imagem original na região selecionada. Essa característica é devida ao fato de grande parte da imagem ter, de forma geral, variações graduais e suaves de tonalidade. Para esta região, o TEB do procedimento proposto foi bem reduzido (17,2%) com relação ao do JPEG Básico (40,6%).

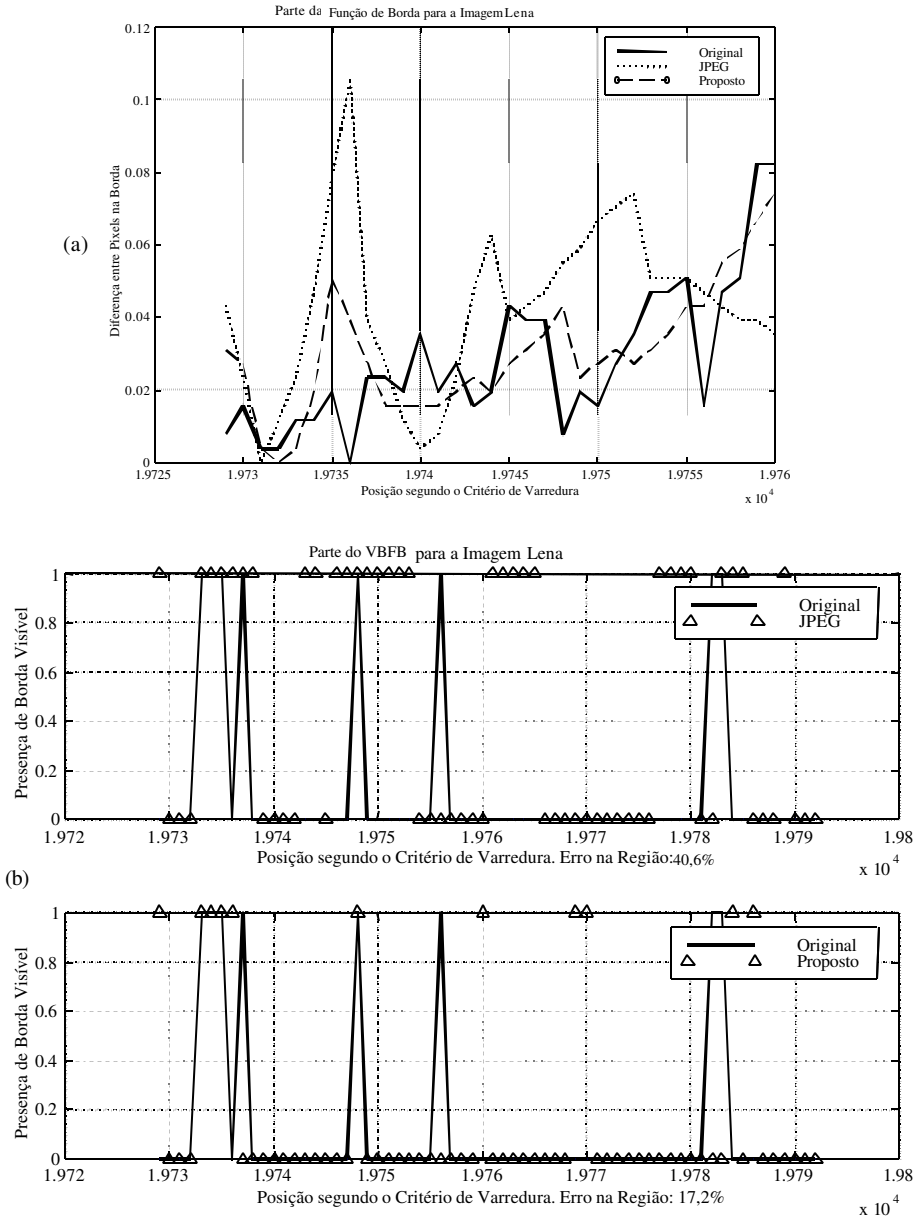
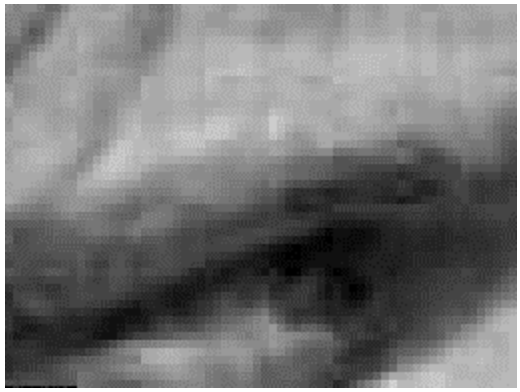


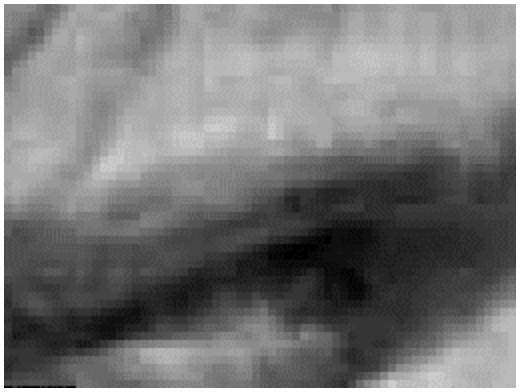
Fig. 6.10 – (a) Função de Borda; (b) VBFB para uma região da Ampliação I da imagem *Lena*.

Imagem *Pirate*

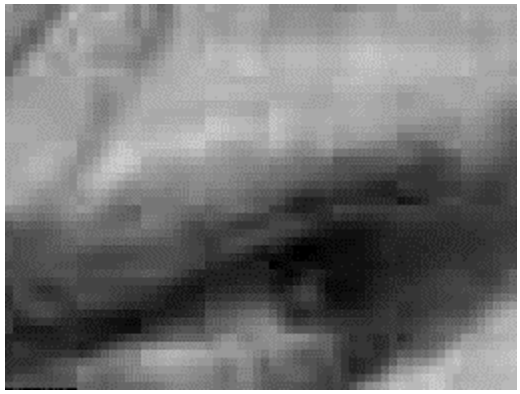
O motivo principal que levou à escolha da imagem *Pirate* foi o de verificar o comportamento do procedimento proposto para imagens de dimensões grandes. Para essa imagem selecionou-se as regiões de ampliação como sendo a região do olho (Ampliação I – Fig. 6.11) e a região da ponta do chapéu do pirata (Ampliação II – Fig. 6.12).



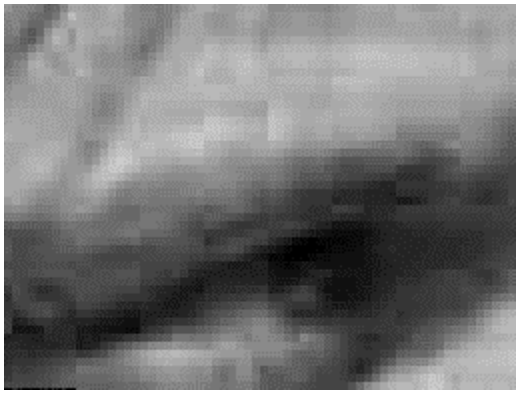
(a) JPEG, 0,8 bpp PSNR=34,2dB TEB=28,3%



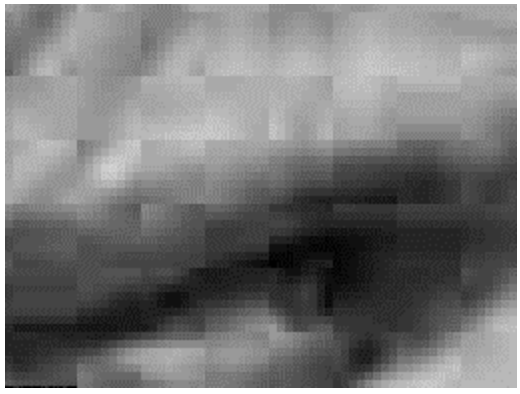
(b) Prop., 0,8 bpp PSNR=33,7dB TEB=21,6%



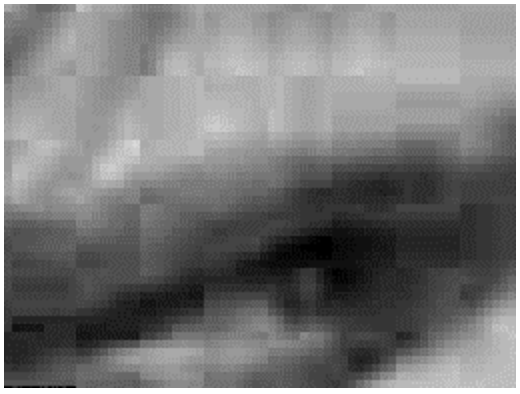
(c) JPEG, 0,6 bpp PSNR=33,0dB TEB=29,9%



(d) Prop, 0,6 bpp PSNR=32,6dB TEB=22,1%

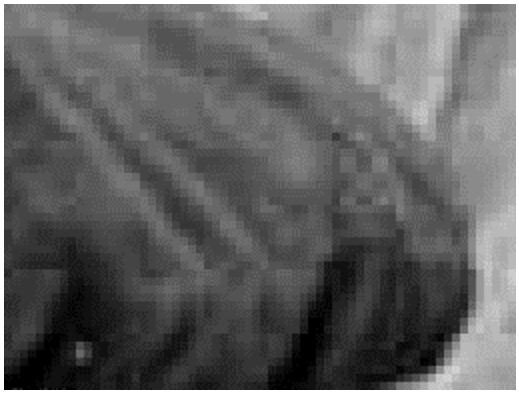


(e) JPEG, 0,4 bpp PSNR=31,1dB TEB=31,9%

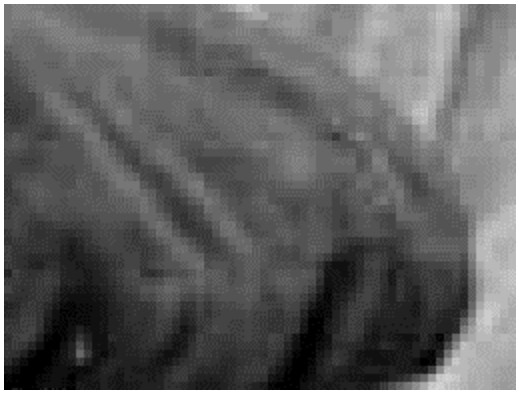


(f) Prop, 0,4 bpp PSNR=30,6dB TEB=22,3%

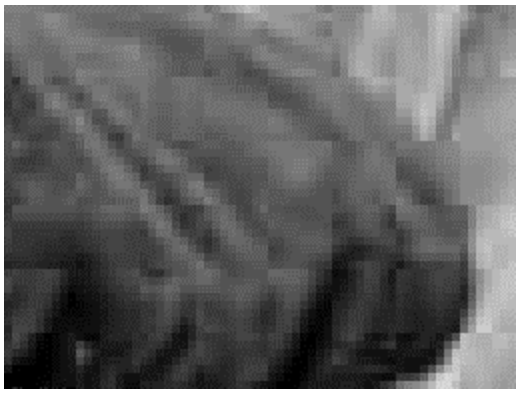
Fig. 6.11 – Ampliação I da imagem *Pirate* para o Procedimento Proposto e para o JPEG.



(a) JPEG, 0,8 bpp PSNR=34,2dB TEB=28,3%

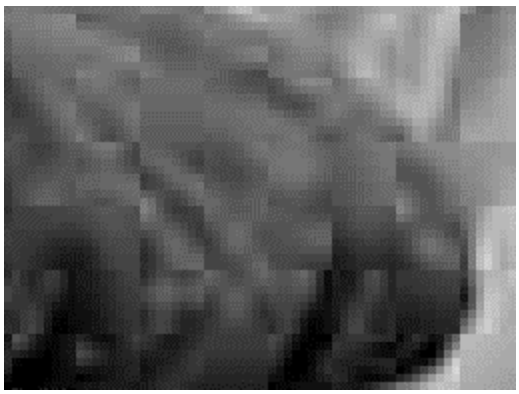


(b) Prop., 0,8 bpp PSNR=33,7dB TEB=21,6%

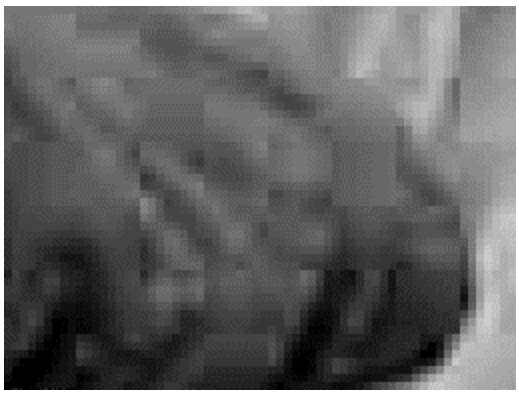


(c) JPEG, 0,6 bpp PSNR=33,0dB TEB=29,9%

(d) Prop, 0,6 bpp PSNR=32,6dB TEB=22,1%



(e) JPEG, 0,4 bpp PSNR=31,1dB TEB=31,9%



(f) Prop, 0,4 bpp PSNR=30,6dB TEB=22,3%

Fig. 6.12 – Ampliação II da imagem *Pirate* para o Procedimento Proposto e para o JPEG.

A primeira ampliação mostra uma região relativamente plana e suave, onde o procedimento proposto reduz significativamente a blocagem, repetindo o comportamento já observado para as outras imagens. Já na Ampliação II tem-se uma região de grande detalhamento devido às dobras do chapéu e, porisso, o desempenho do Procedimento proposto se assemelha ao do JPEG básico. Isso se explica pelo fato de haver originalmente muitas bordas, fazendo com que o algoritmo não atue na maioria das REBs da ampliação, evitando, assim, o borramento de detalhes da imagem.

A Fig. 6.13 mostra os gráficos de FB e VBFB para a imagem com 0,6bpp da Ampliação I imagem *Pirate*.

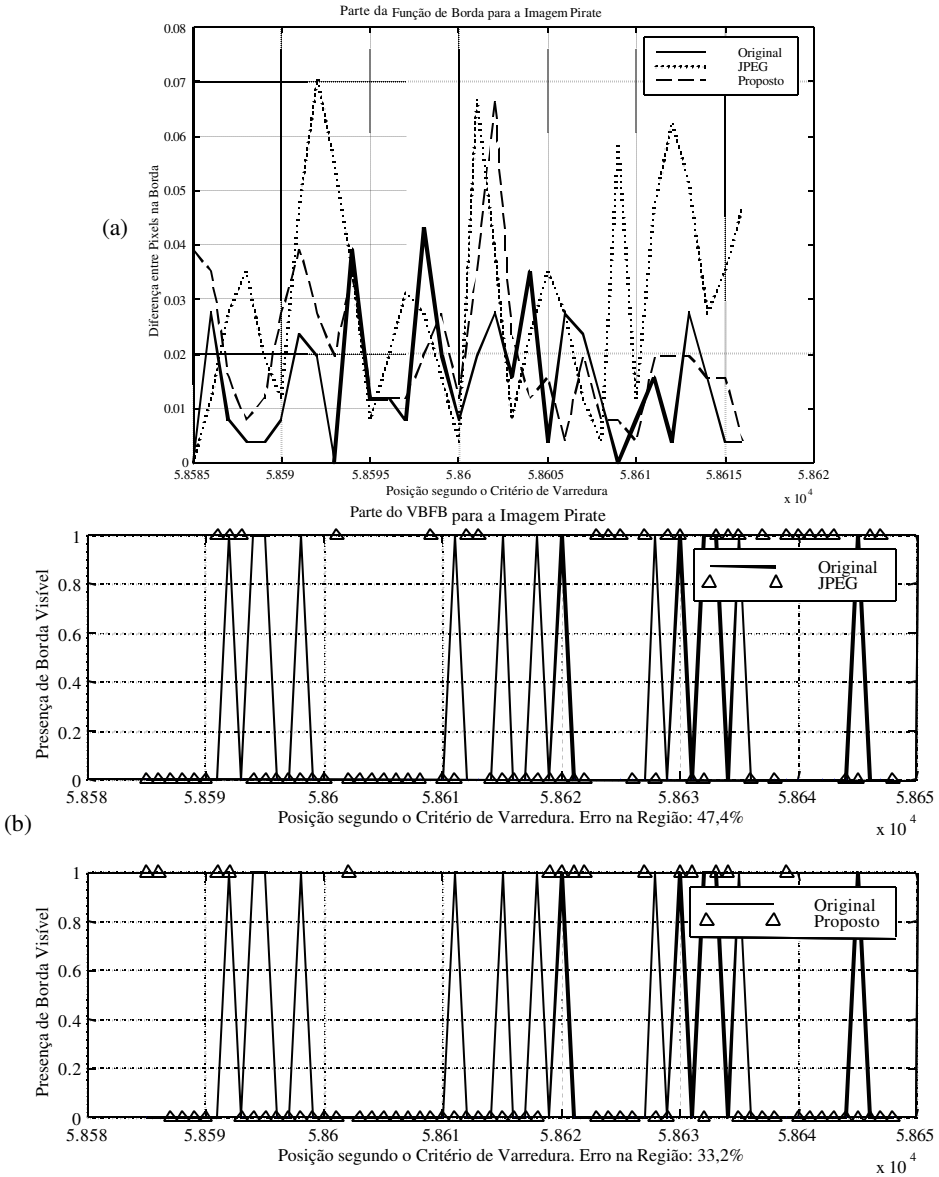


Fig. 6.13 – (a) Função de Borda; (b) VBFB para uma região da Ampliação I da imagem *Pirate*.

Nesses dois gráficos pode-se ver que o desempenho do procedimento proposto é praticamente igual ao das imagens anteriores, apresentando uma redução das bordas geradas, especialmente das regiões mais planas.

Imagem *Zelda*

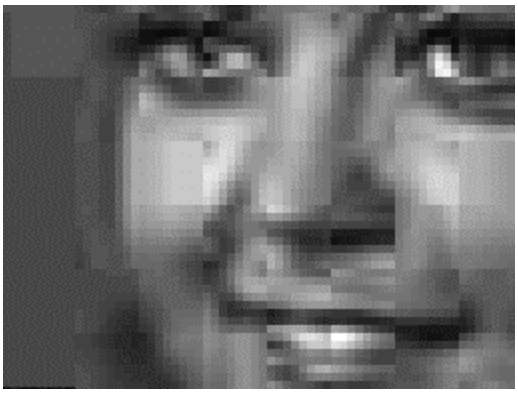
Para avaliar o comportamento do procedimento proposto em imagens menores, escolhemos a imagem *Zelda* (Fig. 6.1e) por ser essa imagem muito utilizada na bibliografia especializada. Para essa imagem, as ampliações escolhidas foram o rosto (Ampliação I) e o ombro esquerdo (Ampliação II).

(a) JPEG, 0,8 bpp PSNR=36,5dB TEB=16,6%

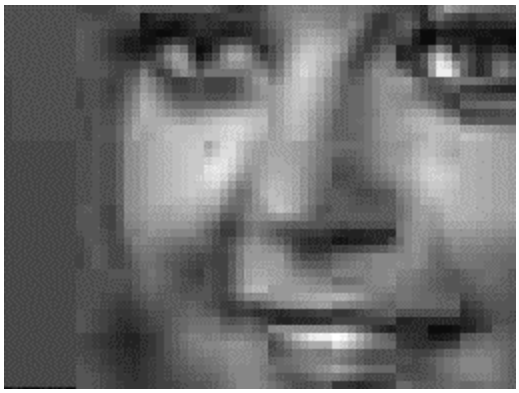
(b) Prop., 0,8 bpp PSNR=35,8dB TEB=14,0%

(c) JPEG, 0,6 bpp PSNR=35,0dB TEB=18,0%

(d) Prop, 0,6 bpp PSNR=34,4dB TEB=14,7%

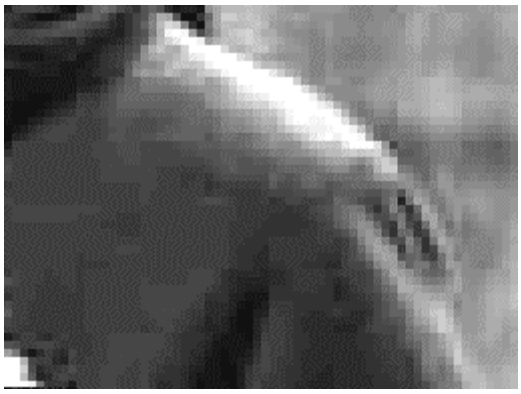


(e) JPEG, 0,4 bpp PSNR=32,8dB TEB=21,3%

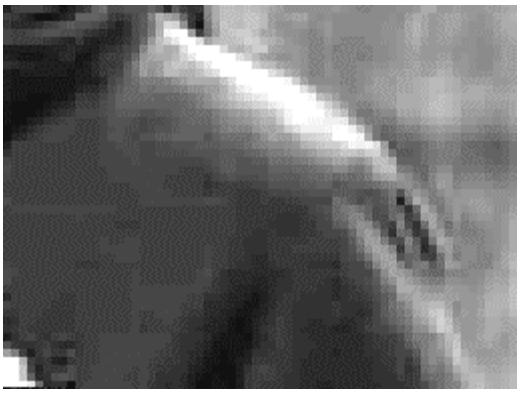


(f) Prop, 0,4 bpp PSNR=32,2dB TEB=16,3%

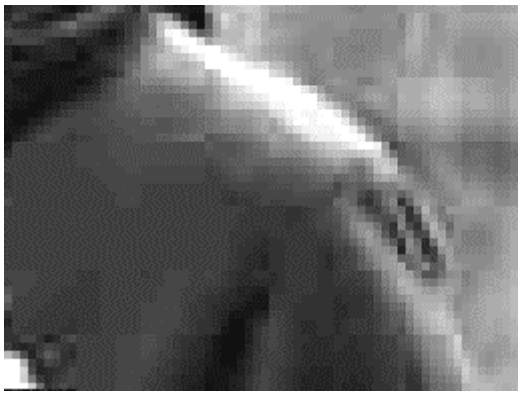
Fig. 6.14 – Ampliação I da imagem *Zelda* para o Procedimento Proposto e para o JPEG.



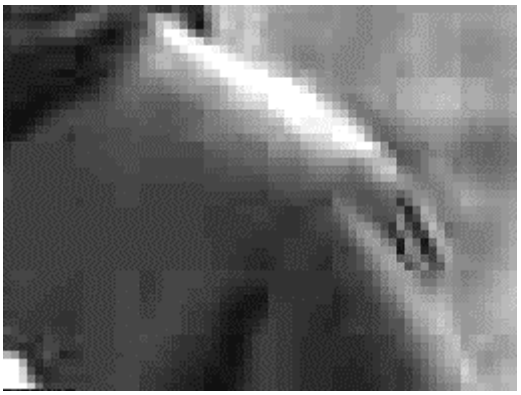
(a) JPEG, 0,8 bpp PSNR=36,5dB TEB=16,6%



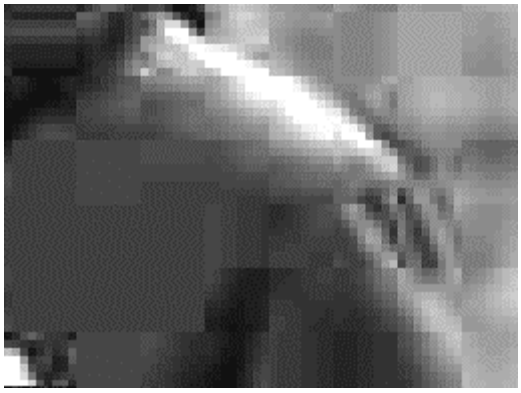
(b) Prop., 0,8 bpp PSNR=35,8dB TEB=14,0%



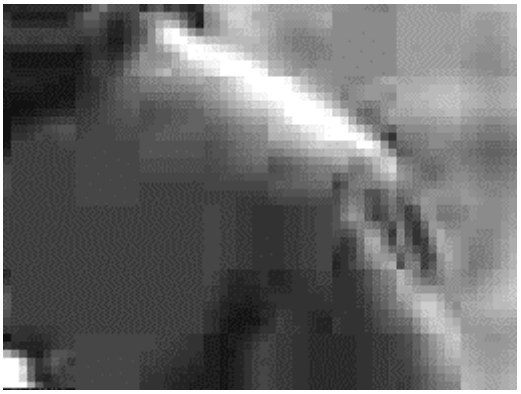
(c) JPEG, 0,6 bpp PSNR=35,0dB TEB=18,0%



(d) Prop, 0,6 bpp PSNR=34,4dB TEB=14,7%



(e) JPEG, 0,4 bpp PSNR=32,8dB TEB=21,3%



(f) Prop, 0,4 bpp PSNR=32,2dB TEB=16,3%

Fig. 6.15 – Ampliação II da imagem *Zelda* para o Procedimento Proposto e para o JPEG.

De forma geral, o resultado da aplicação do procedimento proposto para todas as imagens foram semelhantes, o que também pode ser observado para essa imagem. As regiões mais atingidas pela melhoria visual foram as regiões planas ou suaves como o nariz e a maçã do rosto na Ampliação I e a parte interna do ombro na Ampliação II. Isso ocorre pois nessas regiões o VBFB original é quase sempre nulo (indicando ausência de borda) e as bordas geradas pelas perdas de codificação sofrerão, em sua maioria, o processo de redução de efeito de bloco proposto.

O gráfico das Funções de Borda e dos VBFBs para a ampliação I são apresentados na Fig. 6.16 abaixo.

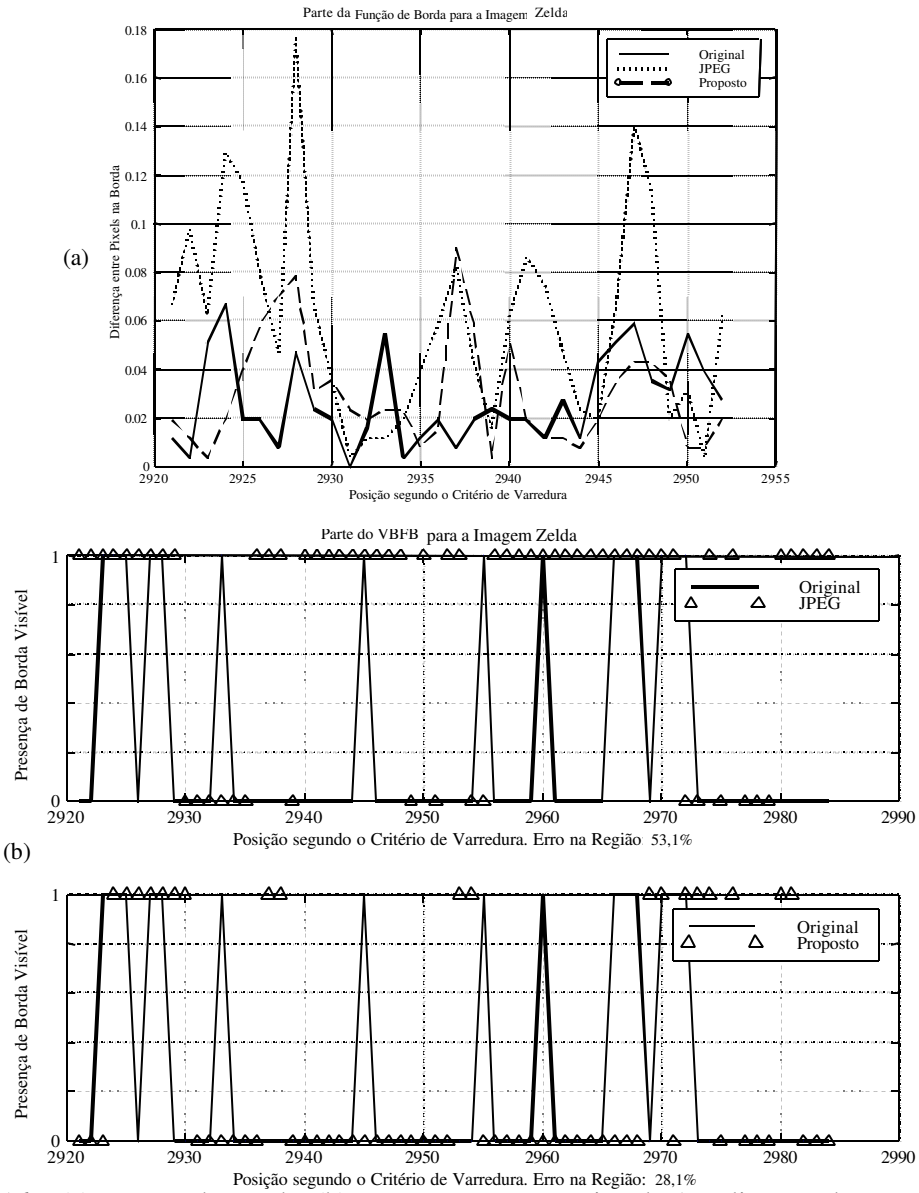


Fig. 6.16 – (a) Função de Borda; (b) VBFB para uma região da Ampliação I da Imagem *Zelda*.

Analisando-se esses dois gráficos pode-se ver claramente que o procedimento proposto realmente reduz a presença de bordas aproximando a curva da imagem recuperada da curva da imagem original. No Gráfico do VBFB pode-se ver que a região central, onde havia poucas bordas originalmente, sofreu uma intensa geração de borda utilizando apenas o JPEG básico. Já para o procedimento proposto a geração de bordas foi muito reduzida, atenuando, segundo o critério utilizado, o efeito de bloco na região.

6.3 – Análises e Comentários

Pode-se observar, através dos resultados apresentados, que para todas as imagens utilizadas, houve uma considerável melhora na qualidade visual especialmente nas regiões mais planas. Ocorreram, no entanto, algumas áreas em que o procedimento proposto suavizou algumas bordas, especialmente as menores, existentes na imagem original, prejudicando um pouco os detalhes da imagem.

Com base na análise do procedimento adotado, pode-se creditar essa melhora em grande parte ao critério de medida MBFB, que baseia sua medida na linearidade (suavidade) dos contornos da imagem. Essa medida de linearidade proporciona uma melhor aproximação numérica para a qualidade subjetiva da imagem provendo, assim, uma referência muito boa tanto para a localização de distorções quanto para a correção das bordas geradas.

Já as distorções são causadas pela subamostragem da função VBFB que acarreta em perda da informação da imagem original e também na taxa de compressão um pouco maior dos dados da imagem para que o VBFB codificado possa ser inserido no cabeçalho sem que ocorra redução na taxa de compressão total da imagem.

Analisando as causas dessas perdas de detalhes, pôde-se determinar que se o tamanho de uma borda for menor ou igual ao fator de subamostragem do VBFB, essa borda pode ser totalmente perdida. Um bom exemplo é a borda apresentada na Fig. 6.4b (VBFB para a imagem *Flowers*), entre os pontos 20471 e 20474 existe um borda de 4 pontos que na imagem reconstruída pelo sistema proposto é totalmente eliminada. Isso ocorreu porque na subamostragem do VBFB original essa borda foi dividida em duas partes iguais com dois 0's e dois 1's tendo, assim, uma média de 0,5, indicando a ausência de borda e a conseqüente atenuação da borda correspondente na imagem reconstruída. Se de outra forma, essa borda estivesse deslocada em 1 posição (para a esquerda ou para a direita), a divisão preservaria 3 pontos da borda atenuando apenas 1, o que manteria, pelo menos parcialmente, a caraterística da imagem original.

Essa análise é interessante na medida em que possibilita a determinação de quais as conseqüências da subamostragem do VBFB e abre o caminho para que se possa futuramente procurar por soluções quanto a este tópico.

Outra análise que pode ser feita é a dos resultados numéricos (medidas de qualidade) apresentados nos itens e resumidos na Tabela VI.1. Além das informações de PSNR e de TEB de todas as simulações, a tabela também mostra o tamanho em bits por *pixel* (bpp) do *overhead* gerado pela inserção do VBFB codificado e da sua tabela de codificação no cabeçalho da imagem.

Imagem	Dimensões	Overhead (bpp)	bpp Total	PSNR (dB)		Taxa de Erro de Bit (%)	
				JPEG	Proposto	JPEG	Proposto
Flowers	**512**	**0,044**	**0,80**	34,369	33,539	24,01	17,71
	x		**0,60**	32,539	32,013	27,39	17,89
	384		**0,40**	30,033	29,849	31,57	18,03
Foto Aérea	**512**	**0,056**	**0,80**	29,202	28,747	31,24	21,41
	x		**0,60**	27,563	27,250	33,68	22,21
	512		**0,40**	25,502	24,939	35,74	22,44
Lena	**512**	**0,031**	**0,80**	36,811	36,251	18,82	15,90
	x		**0,60**	35,552	35,083	19,95	16,05
	512		**0,40**	33,547	33,180	22,90	16,61
Pirate	**1024**	**0,043**	**0,80**	34,212	33,731	28,26	21,57
	x		**0,60**	33,017	32,640	29,85	22,13
	1024		**0,40**	31,094	30,642	31,87	22,31
Zelda	**256**	**0,043**	**0,80**	36,479	35,767	16,64	14,04
	x		**0,60**	34,969	34,435	17,97	14,71
	256		**0,40**	32,803	32,163	21,32	16,29

TABELA VI.1 – Resumo dos resultados numéricos das simulações para o JPEG.

Observando a tabela acima, talvez o primeiro detalhe que se ressalta é o fato de o *Overhead* gerado no sistema proposto é praticamente constante com a variação do tamanho da imagem sendo um pouco maior para a imagem *Foto Aérea* e um pouco menor para a imagem *Lena.* A explicação para essas diferenças são as características de suavidade e o tamanho dos blocos com relação ao da imagem, pois esses fatores influenciam na correlação dos elementos do VBFB. No caso da imagem *Lena*, a maior parte dos blocos é suave proporcionando uma grande incidência de 0's consecutivos no VBFB aumentando a eficiência da codificação *Run-Length* e portanto, reduzindo o tamanho do *overhead* gerado. O contrário ocorre na imagem *Foto Aérea*, que apresenta uma enorme quantidade de detalhes espalhados praticamente por toda a imagem, o que proporciona muitas trocas de bits na seqüência do VBFB, diminuindo os *Runs* e reduzindo a compressão do codificador.

Em cada uma das outras imagens utilizadas há um certo equilíbrio entre os fatores que colaboram para a compressão do VBFB: na imagem *Flowers*, o centro da imagem é bastante detalhado, mas o fundo é praticamente constante. A imagem *Pirate* o grande número de detalhes é compensado pelo maior número de blocos, fazendo com que cada bloco seja semelhante ao seu

vizinho, aumentando a linearidade entre eles e fazendo com que a correlação entre elementos vizinhos do VBFB seja maior, aumentando a eficiência da compressão . O efeito inverso ocorre na imagem *Zelda*, que é bastante suave, mas os blocos são relativamente maiores, proporcionando uma maior alternância de valor nos elementos do VBFB levando, portanto, a uma menor compactação do mesmo.

Com relação à medida de qualidade objetiva PSNR, pode-se ver que o procedimento proposto sempre apresenta um resultado um pouco abaixo do resultado para o JPEG. Esse fato era esperado porque, para se inserir o VBFB (com sua tabela de códigos) na imagem, reduziu-se o espaço reservado para os dados da imagem em si, ou seja, provoca-se uma quantização mais grosseira dos coeficientes para permitir uma maior compressão dos dados e manter o número de bits final igual. Como a quantização afeta todos os coeficientes do bloco e a interpolação afeta apenas a região de fronteira, é natural que haja uma pequena queda no PSNR da imagem.

No entanto, ao se observarmos a Taxa de Erro de Bit (TEB), pode-se notar que o procedimento proposto tem, em todos os casos, uma TEB menor que o JPEG básico, o que significa que as fronteiras entreblocos da imagem recuperada pelo sistema proposto apresenta um número muito menor de bordas geradas (medidas pelo critério MBFB). Outro ponto de interesse é que o TEB cresce à medida que a compressão aumenta, fazendo com que o efeito de bloco presente nas imagens seja compatível com o resultado numérico obtido, levando, assim, a concluir que o valor da TEB como foi definido é uma boa medida de blocagem para a imagem recuperada.

No entanto, como o TEB só leva em conta os pixels das regiões fronteiriças entre os blocos, utilizá-lo como medida única de qualidade de imagem não é recomendado visto que podem haver outras distorções nas imagens recuperadas que não geram efeito de bloco tais como as filtragens e o ruído aditivo.

Com relação à análise dos histogramas, pode-se observar nas figuras B.1 a B.5 (mostradas no apêndice B), que a procedimento proposto provoca muito poucas mudanças no histograma da imagem recuperada. Isto se deve ao fato de que o procedimento proposto altera apenas o valor dos 2 pixels internos de uma REB na qual foi caracterizada a geração de uma borda.

6.4 – Resultados para o MPEG-2 TM5

Como foi dito anteriormente, o padrão MPEG-2 foi concebido e otimizado originalmente para a compressão de seqüências de vídeo. No entanto, a intenção desse trabalho ao utilizar esse sistema de codificação para imagens estáticas foi a de analisar o comportamento do procedimento

proposto para um sistema de codificação diferente do JPEG e a de realizar um estudo inicial para a implementação de uma versão do sistema proposto para a codificação de vídeo.

Na codificação de vídeo MPEG-2 TM5, realizar a compressão de uma imagem estática implica em acertar os parâmetros de entrada de forma que o codificador não utilize compressão temporal nem outros recursos de compactação baseados em entrelaçamento (*interlace* – divisão da imagem em dois campos: de linhas pares e de linhas ímpares). Dessa forma, o codificador tratará a imagem de entrada como uma *I-picture* sem entrelaçamento, fazendo com que o seu processo de codificação seja semelhante ao do JPEG básico.

Outra preocupação relativa à utilização do codificador MPEG-2 foi a respeito do espaço ocupado pelo cabeçalho da imagem, mas pôde-se observar que isso não representa um problema pois além de praticamente todas as informações constantes no cabeçalho serem codificadas em VLC, a tabela de codificação entrópica dos dados é fixa e não precisa ser incluída no *bitstream*, o que acaba compensando a informação extra no cabeçalho.

Para os resultados que serão apresentados aqui, foram utilizadas as duas imagens de entrada monocromáticas (*Lena* e *Foto Aérea*), com precisão de 8 bits por pixel, duas taxas de compressão (0,8bpp e 0,6bpp), precisão de codificação do DC de 8 bits e as matrizes de quantização *default* para luminância. Também foi utilizado o controle de taxa de bits e de quantização especificados no *Test Model 5*, controlando-se a taxa de compressão apenas pela modificação do *bit_rate* da seqüência.

A codificação do VBFB foi feita de forma semelhante à descrita do item 6.2 para o JPEG com a única diferença de que não foi calculada uma tabela de Huffmann e sim utilizada a tabela de codificação entrópica fixa do padrão MPEG-2 (no caso, a tabela B.14 [22]).

Como na seção anterior, as imagens resultantes para os dois sistemas serão apresentados através das duas ampliações de cada imagem recuperada, juntamente com as medidas de PSNR e Taxa de Erro de Bit (TEB) e os gráficos da Função de Borda e do VBFB. De modo a facilitar a comparação entre os resultados, foram utilizados aqui as mesmas regiões para ampliação e os mesmos pontos das curvas de FB e VBFB.

Como as duas imagens com as quais se trabalhou foram imagem *Lena* e a imagem *Foto Aérea* que já foram apresentadas nas Figs. 6.1b e 6.1c, foram colocadas aqui somente as Ampliações.

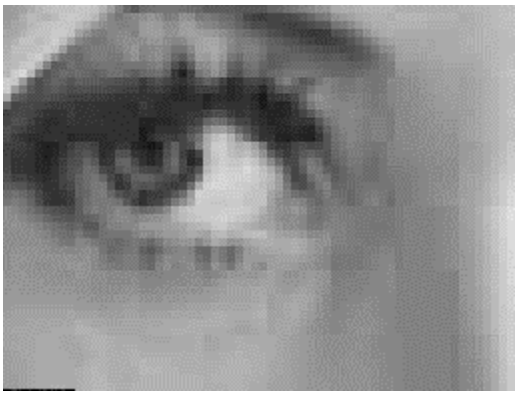
Resultados para a Imagem *Lena*

Como uma das intenções iniciais da implementação desse sistema é a comparação dos resultados com os já obtidos para o JPEG, foram escolhidas as ampliações dos mesmos detalhes, tanto para a Imagem *Lena* como para a imagem *Foto Aérea* (apresentada no próximo sub-item). Para

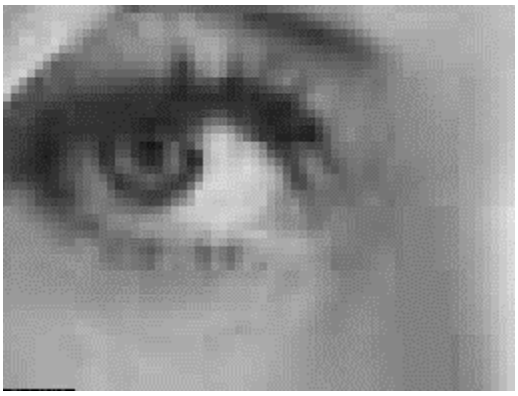
os gráficos, também foi escolhido a mesma região tanto da FB quanto do VBFB. Assim, pode-se ver na Fig. 6.17 as Ampliações da região do olho direito da modelo e na Fig. 6.18, as Ampliações da aba do chapéu.

As razões da escolha dessas duas imagens a serem submetidas à essa codificação são diversificadas. Entre elas pode-se citar o fato de que essas imagens propiciam o menor (*Lena*) e o maior (*Foto Aérea*) *overhead* gerado pela inserção do VBFB no cabeçalho da imagem. Além disso, as duas imagens são do mesmo tamanho, proporcionando um melhor ambiente de comparação. Por fim, ambas apresentam níveis de detalhamento muito diferentes, propiciando uma melhor avaliação do sistema baseado na codificação do MPEG-2.

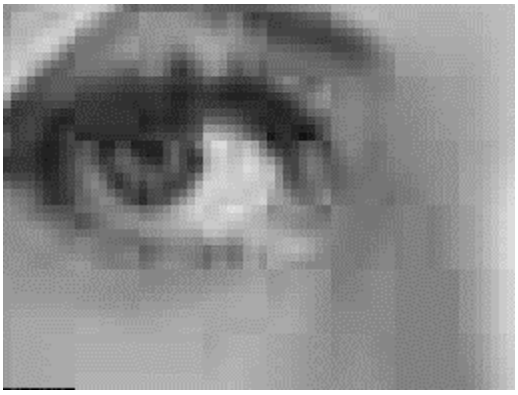
Tomando-se com base de comparação os mesmos detalhes para o JPEG, pode-se ver que esse sistema causa mais efeito de bloco nas imagens para a mesma taxa de compressão. Isso ocorre porque o processo de quantização e codificação do MPEG-2 foi projetado para trabalhar com sinais de vídeo que utilizam compressão temporal. Assim, esses sinais apresentam características de distribuição de energia nos coeficientes transformados diferentes das imagens estáticas.



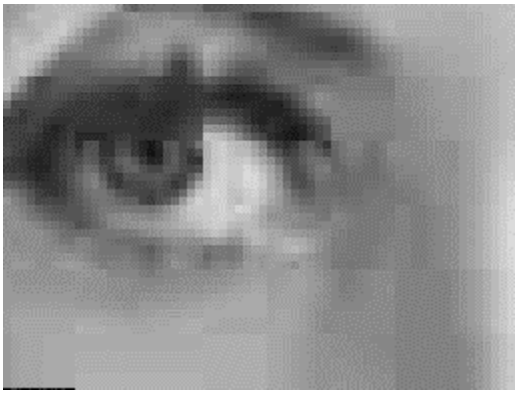
(a) MPEG2 0,8bpp PSNR=35,8dB TEB=19,0%



(b) Prop. 0,8bpp PSNR=35,3dB TEB=16,3%



(c) MPEG2 0,6bpp PSNR=34,0dB TEB=20,6%

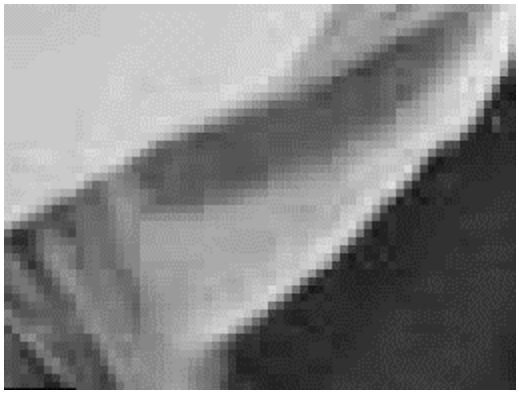


(d) Prop 0,6bpp PSNR=33,6dB TEB=16,8%

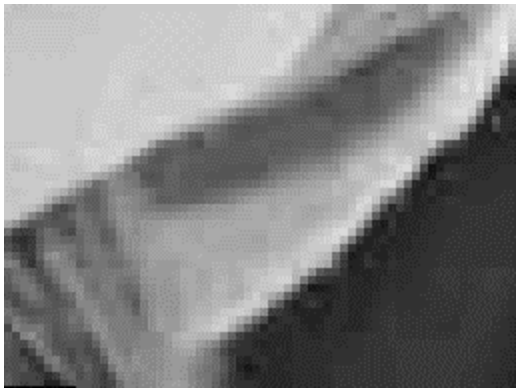
Fig. 6.17 – Ampliação I da imagem *Lena* para o Procedimento Proposto e para o MPEG-2.

É interessante observar que, assim como os resultados para o JPEG, o PSNR cai e a TEB sobe com o aumento da degradação da imagem. Com relação ao comportamento dessa medida entre o MPEG-2 e o procedimento proposto, o PSNR ainda apresenta um pequena queda enquanto a TEB apresenta melhora. Comparando os valores obtidos aqui com os mostrados na Tabela VI.1 (resultados do JPEG), tem-se que as medidas de qualidade são piores (PSNR menor e TEB maior) para essas imagens do que para as do item 6.3, como era esperado pela análise visual dos resultados.

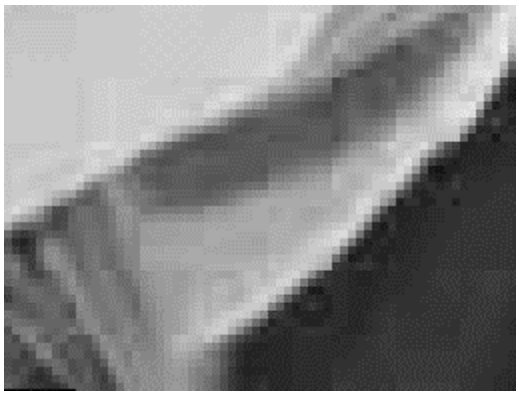
A Fig. 6.18 mostra a Ampliação II para as imagens recuperadas, onde pode-se observar que a redução do efeito de bloco pela aplicação do VBFB e da interpolação continua bastante notável, inclusive para a taxa de compressão de 0,8 bpp.



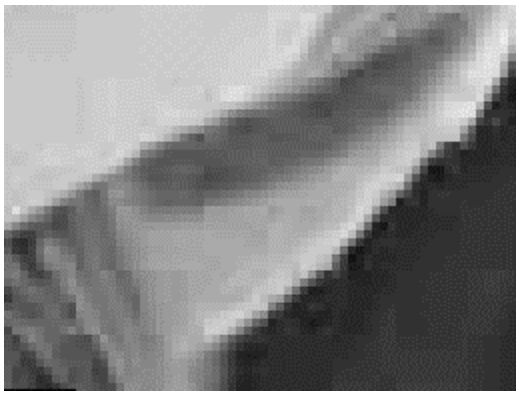
(a) MPEG 0,8bpp PSNR=35,8dB TEB=19,0%



(b) Prop. 0,8bpp PSNR=35,3dB TEB=16,3%



(c) MPEG 0,6bpp PSNR=34,0dB TEB=20,6%



(d) Prop 0,6bpp PSNR=33,6dB TEB=16,8%

Fig. 6.18 – Ampliação II da imagem *Lena* para o Procedimento Proposto e para o MPEG-2.

A seguir são mostrados os gráficos de Função de Borda e dos VBFBs para as imagens original e recuperadas pelos dois sistemas. Com relação à Fig. 6.19, deve-se notar que as curvas da FB original e do VBFB original são rigorosamente iguais aos apresentados na Fig.6.10. Isso ocorre

porque a Função de Borda e o VBFB são levantadas a partir da imagem original antes de sofrer qualquer procedimento de codificação ou transformação. Assim, as curvas de Função de Borda e do VBFB são independentes do processo de codificação ao qual a imagem será submetida.

Essa característica é uma vantagem porque permite a portabilidade do procedimento de redução entre diversos esquemas de compressão com perdas que sejam baseados em blocos.

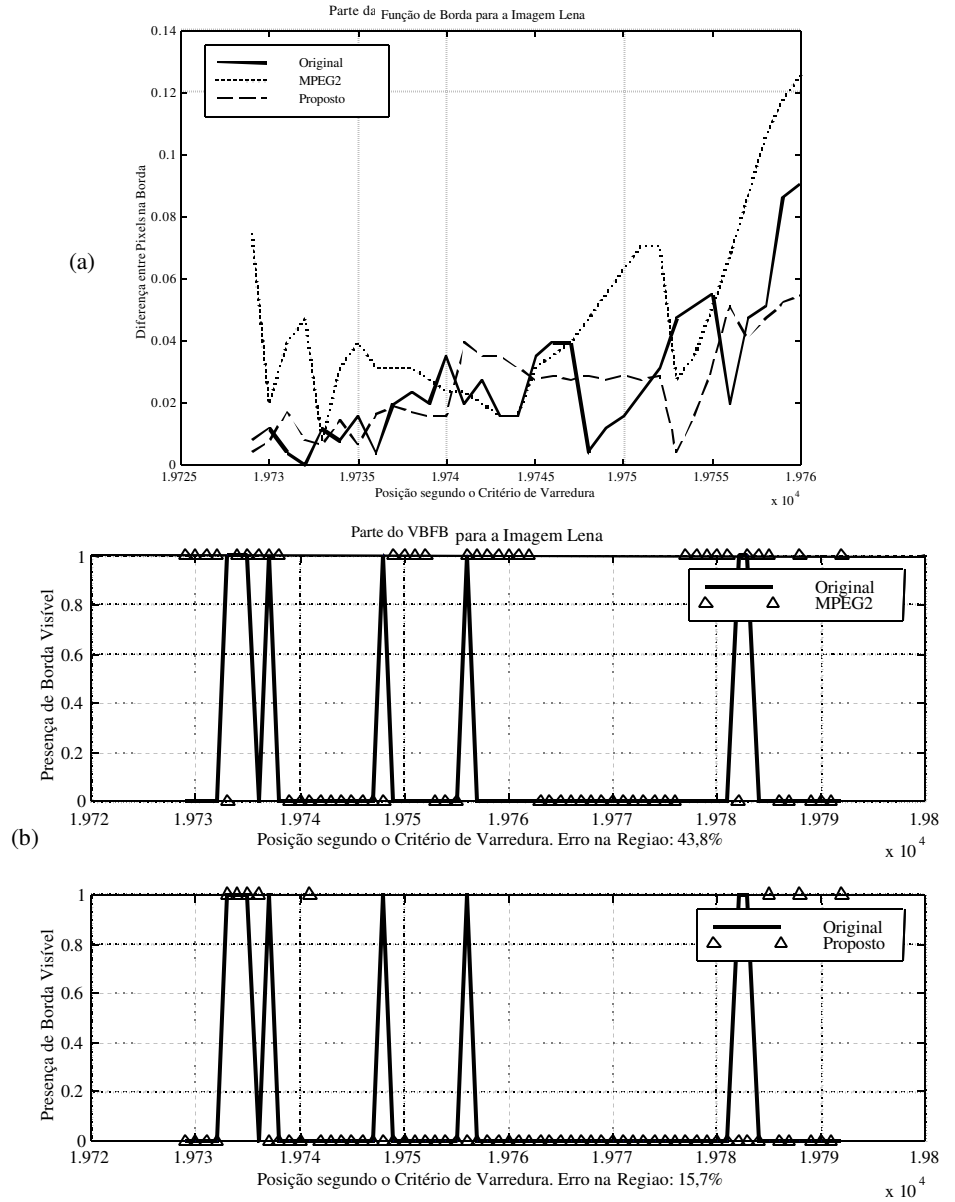


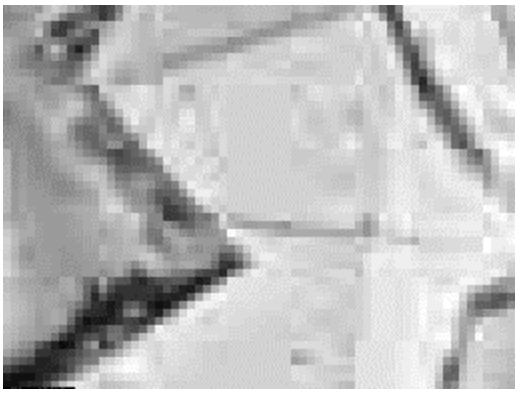
Fig. 6.19 – (a) Função de Borda; (b) VBFB para um região da Ampliação I da imagem *Lena.*

Na Fig. 6.19 pode-se observar que a característica de aproximação da FB da imagem recuperada pelo Procedimento Proposto da imagem original se mantém apesar da mudança de sistema de codificação, o que também ocorre para o gráfico do VBFB.

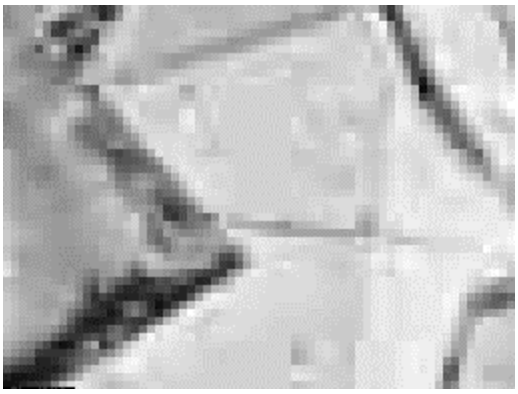
Esses resultados são um reflexo (e um exemplo) da portabilidade citada anteriormente, como uma das características do sistema proposto.

Imagem *Foto Aérea*

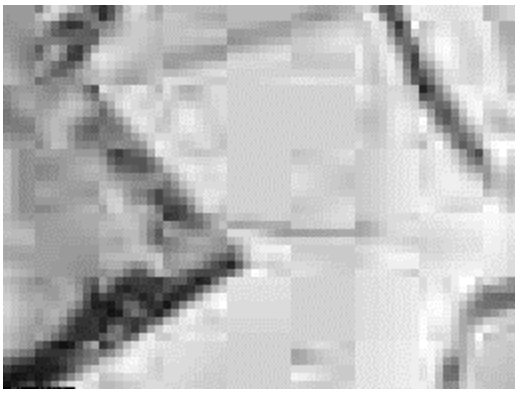
Como comentado anteriormente, a imagem *Foto Aérea* apresenta um grande número de pequenos detalhes, fazendo com que a sua compressão por um codificador por transformada tenha um desempenho reduzido. Esse fato torna interessante a sua utilização exatamente para avaliar o comportamento do processo de redução de efeito de bloco proposto sob condições mais adversas.



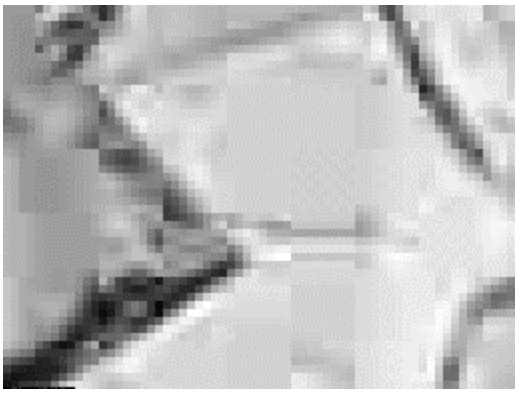
(a) MPEG 0,8bpp PSNR=28,0dB TEB=31,6%



(b) Prop. 0,8bpp PSNR=27,7dB TEB=22,3%

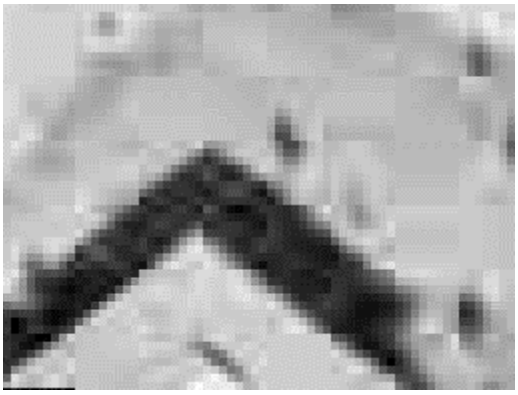


(c) MPEG 0,6bpp PSNR=26,6dB TEB=33,9%

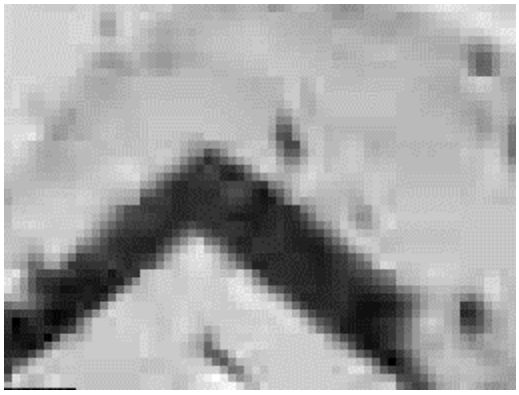


(d) Prop 0,6bpp PSNR=26,4dB TEB=22,5%

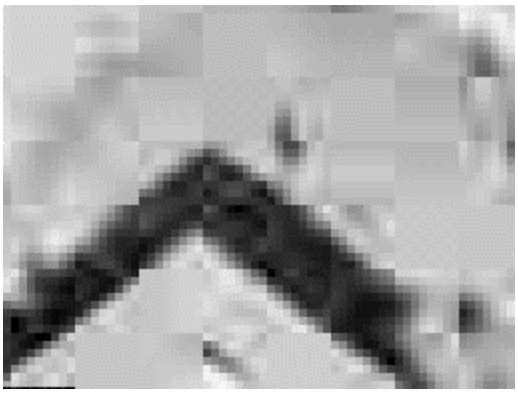
Fig. 6.20 – Ampliação I da imagem *Foto Aérea* para o Procedimento Proposto e para o MPEG-2.



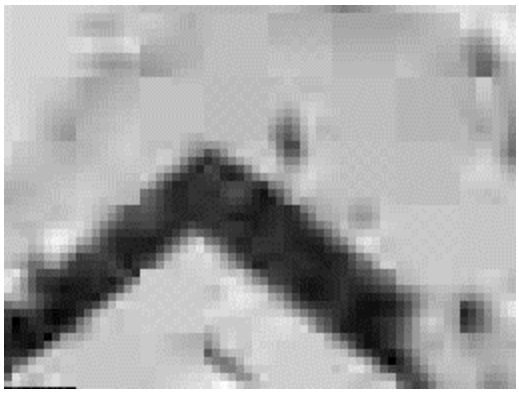
(a) MPEG 0,8bpp PSNR=28,0dB TEB=31,6%



(b) Prop. 0,8bpp PSNR=27,7dB TEB=22,3%



(c) MPEG 0,6bpp PSNR=26,6dB TEB=33,9%



(d) Prop 0,6bpp PSNR=26,4dB TEB=22,5%

Fig. 6.21 – Ampliação II da imagem *Foto Aérea* para o Procedimento Proposto e para o MPEG-2.

As Figs. 6.20 e 6.21 mostram as duas ampliações da Imagem *Foto Aérea* exatamente nas mesmas regiões mostradas para o JPEG.

Para essa imagem, como para a imagem anterior, tem-se a conservação das características de perda de qualidade subjetiva do MPEG-2 com relação ao JPEG e também de piora dos resultados numéricos de PSNR e TEB. Nas ampliações é evidente a melhora visual causada pela aplicação do processo de redução, seguindo o comportamento geral da processo.

Na Fig. 6.22 mostram-se os gráficos das FBs e dos VBFBs para a Ampliação I na mesma região mostrada na Fig. 6.7. No gráfico da FB percebe-se a presença de um pico muito pronunciado na imagem recuperada pelo procedimento proposto (entre 11950 a 11955) não existente na imagem original. A existência desse pico pode ser considerada normal porque, como pode-se observar no gráfico do VBFB, ele se localiza numa região de uma borda relativamente grande da imagem original, não sendo, portanto, alvo do procedimento de interpolação proposto.

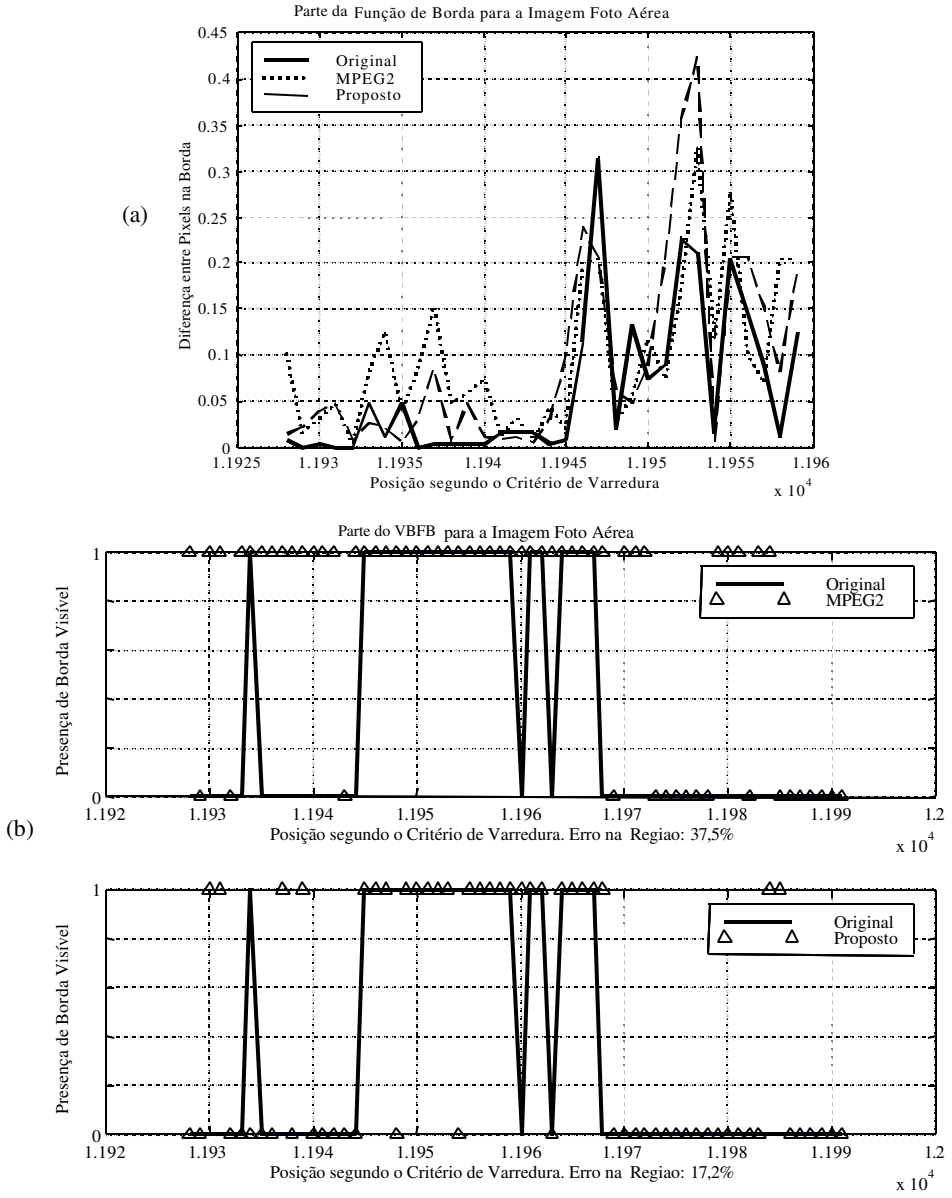


Fig. 6.22 – (a) Função de Borda; (b) VBFB para um região da Ampliação I da imagem *Foto Aérea.*

6.5 – Análises e comentários

Nessa seção foi dado um passo inicial importante no sentido da introdução do sistema de redução dentro de um sistema de compactação de vídeo, o que num futuro próximo pode possibilitar a implementação do sistema para uma seqüência de imagens e não somente para imagens estáticas.

Nesse sentido, os estudos iniciais realizados indicam que há uma boa possibilidade de que o desempenho possa ainda ser melhorado com trabalhos posteriores visando a redução *overhead*

gerado e melhorando também a qualidade da imagem como um todo.

Quanto à qualidade das imagens recuperadas, foram obtidos para esse sistema resultados compatíveis com os resultados do JPEG, tanto visualmente quanto numericamente, o que é um bom indicativo da versatilidade do sistema proposto com relação a modificação do esquema de compressão. No entanto, os sistemas de compressão JPEG e MPEG-2 (da forma como foi utilizado) são muito parecidos, o que impede que essa conclusão seja definitiva.

Um ponto que pode ser analisado mais profundamente foi a relação entre a medida numérica da Taxa de Erro de Bit como uma medida de blocagem da imagem. Como se pode ver comparando a qualidade visual e a TEB para as imagens apresentadas, essa medida representou um bom indicativo para a presença do efeito de bloco nas imagens, tanto para o JPEG quanto para o MPEG-2, sendo, assim, uma das contribuições apresentadas nesse trabalho. Na Tabela VI.2 mostra-se um resumo dos resultados numéricos apresentados na seção anterior.

Imagem	**Dimensões**	**Overhead (bpp)**	**Bpp Total**	**PSNR (dB)**		**Taxa de Erro de Bit (%)**	
				MPEG	**Prop**	**MPEG**	**Prop**
Foto Aérea	**512 x 512**	**0,060**	**0,80**	28,042	27,716	31,55	22,34
			0,60	26,632	26,424	33,90	22,52
Lena	**512 x 512**	**0,033**	**0,80**	35,825	35,315	19,03	15,32
			0,60	33,984	33,565	20,67	15,78

TABELA VI.2 – Resumo dos resultados numéricos das simulações para o MPEG-2.

Uma última análise interessante a respeito dos resultados obtidos é a dos valores dos *overheads* gerados para o MPEG-2 e para o JPEG. Como para o sistema baseado no MPEG, utilizou-se uma tabela de codificação entrópica fixa (a tabela B.14 especificada em [22]), a taxa de compressão do VBFB codificado aqui seria bem menor pela perda de otimização, mas o fato de a tabela código não precisar ser inserida no cabeçalho gerou uma boa compensação fazendo com que os *overheads* fossem praticamente iguais.

Como a tabela do MPEG-2 foi especificamente projetada para se trabalhar com os dados de imagem, esse resultado nos indica que a possibilidade da realização de um estudo estatístico das propriedades do VBFB de forma a tentar a implementação de uma tabela código fixa para o VBFB é extremamente válida e pode resultar em taxas de compressão mais significativas, colaborando para a implementação do procedimento proposto para seqüências de vídeo.

Capítulo 7

Conclusões e Sugestões para Trabalhos Futuros

7.1 – Comentários e Conclusões

Atualmente, o Processamento Digital de Imagens (PDI) possui uma ampla gama de aplicações que vai desde aplicações médicas, processamento de imagens adquiridas por satélites de sensoreamento remoto, teleconferência, tratamento de imagens geológicas, radares e sonares; até cinema digital e televisão digital de alta definição (HDTV – *High Definition Television*).

A representação digital da informação visual, em sua forma "crua" (não processada), gera, por característica inerente, um número significativo de grandes arquivos. Objetivando a melhoria e economia no armazenamento e transmissão em largura de banda, surgiram as técnicas de compressão.

O esquema de compressão espacial digital mais utilizado pelos codificadores padrão de imagem e vídeo é a Codificação por Transformada baseada em blocos, que compreende três estágios: a aplicação da transformação ortonormal em si, a quantização e a codificação entrópica.

Quanto ao primeiro destes estágios, as diversas transformadas espaciais têm por objetivo descorrelacionar os dados da imagem e portanto, em seu domínio, especialmente para imagens típicas, a energia dos coeficientes (embora constante) tende a agrupar-se em torno de uma determinada região. Como resultado há geralmente grandes áreas onde os componentes da imagem transformada possuem valores muito pequenos ou próximos a zero. Esta descorrelação aliada à propriedade de compactação de energia em poucos coeficientes, permite recuperar a imagem através de uma pequena fração dos coeficientes transformados.

Como a informação do sinal é então concentrada e não reduzida no domínio transformado, o que possibilita a compressão são os processos subsequentes à transformação: quantização e codificação em comprimento variável. A transformada é o meio que torna estes procedimentos possíveis com perda mínima de informação para subsequente armazenamento e transmissão. A qualidade da imagem reconstruída na recepção esta relacionada ao erro de quantização dos componentes após a transformação.

Na codificação por transformada prática, o processamento da imagem em blocos, reduz a necessidade de armazenamento e carga computacional durante todo o processamento. No entanto, especialmente quando se trabalha a altas taxas de compressão, o processamento por blocos pode dar

origem na reconstrução a efeitos de bloqueamento, que são descontinuidades geradas devido aos erros de quantização, que contribuem para que um bloco não fique perfeitamente casado com os blocos adjacentes. Este Efeito de Bloco é conhecido como o maior artefato de compressão dos padrões JPEG e MPEG-2, e se torna particularmente sério quanto os coeficientes DC e ACs de baixa frequência são grosseiramente quantizados, podendo causar considerável desconforto visual em termos de qualidade subjetiva da imagem.

Neste contexto, este trabalho apresentou uma nova aproximação para o problema de medição e redução do Efeito de Bloco em codificação por transformada baseada em blocos, objetivando a melhoria da qualidade subjetiva das imagens, especialmente para altas taxas de compressão. Uma nova medida de blocagem chamada Medida de Borda na Fronteira do Bloco (MBFB) foi desenvolvida, baseada nas diferenças das derivadas adjacentes próximas à fronteira do bloco, visando obter uma medida mais precisa da impressão subjetiva do olho humano à respeito desse artefato.

O MBFB é calculado para todas as fronteiras dos blocos da imagem e então armazenado num vetor: VBFB (Vetor de Borda na Fronteira do Bloco), que é codificado e anexado ao cabeçalho da imagem para ser transmitido/armazenado. No esquema proposto, de acordo com as mudanças nos elementos dos VBFBs das imagens original e reconstruída, o decodificador é capaz de determinar as áreas onde o Efeito de Bloco ocorre e então, um procedimento interpolativo localizado para redução da blocagem é executado.

Dessa forma, são sumarizadas aqui as contribuições principais desse trabalho como sendo: (a) o desenvolvimento do critério subjetivo de detecção de borda MBFB; (b) um esquema de compressão de imagens por transformada utilizando o procedimento interpolativo de redução de efeito de bloco, cujo diagrama encontra-se na Fig. 5.7; (c) uma medida de qualidade subjetiva de imagens com relação ao efeito de bloco: Taxa de Erro de Bits (TEB) no vetor de Bordas na Fronteira dos Blocos (VBFB).

O esquema proposto apresentou uma melhora significativa da qualidade subjetiva das imagens, especialmente para baixos *bit rates* sem causar significativo "borramento". Essa técnica pode ser aplicada a qualquer codificação por transformada baseada em blocos convencional, tal como os padrões MPEG ou JPEG, através de uma simples rotina adicional sem introduzir modificações em seus códigos originais.

Este trabalho apresentou duas macrodivisões: a primeira correspondendo ao estudo dos tópicos estritamente relacionados à proposta e a segunda, a apresentação da proposta propriamente dita, com resultados e avaliações pertinentes.

Primeiramente, um estudo sobre os três estágios da codificação por transformada foi feito. Assim, no capítulo 2 foram apresentadas: as transformações lineares discretas unidimensional e bidimensional, dando enfoque às características mais relevantes para a compressão de imagens; a transformada discreta DFT (*Discrete Fourier Transform*) e a transformada discreta DCT (*Discrete Cosine Transform*) 1D e 2D, sendo que esta última constitui a base de todos os padrões de compressão utilizados atualmente. As bases dos estágios de quantização e codificação por comprimento variável foram apresentados no capítulo 3, bem como o processo de geração de Efeito de Bloco. Foram também citadas algumas técnicas tradicionais para a redução deste artefato.

Para caracterizar o ambiente de trabalho, apresentou-se no capítulo 4 resumidamente o padrão de compressão de imagens JPEG Básico (*Joint Photographic Experts Group*) e a etapa de compressão espacial do padrão de compressão de vídeo MPEG-2 TM5 (*Moving Picture Experts Group – Test Model 5*), que foram utilizados como base de comparação para os resultados das simulações realizadas neste trabalho.

Foi apresentado a seguir no capítulo 5, o procedimento proposto de medição e redução de efeito de bloco em codificadores por transformada baseado em blocos. Dentro desta proposta, foram detalhados o critério de detecção de bordas proposto, o processo de localização e medição do efeito de bloco gerado e o procedimento interpolativo para a redução de efeito de bloco. Também descreveu-se a medida proposta de qualidade visual referente ao efeito de bloco e o processo de codificação da informação relativa às bordas da imagem original para os dois sistemas utilizados durante o trabalho.

Visando obter uma boa integração e a redução da quantidade de rotinas adicionadas ao sistema original, procurou-se utilizar para a codificação do VBFB uma codificação semelhante à já especificada no padrão. A utilização dessa codificação foi muito proveitosa porque proporcionou taxas de compressão para o VBFB próximas a 7:1, o que representa um acréscimo em torno de somente 0,05 bpp no *bitstream* final.

Assim, para o sistema JPEG, os dados das imagens foram codificados utilizando-se uma tabela de Huffmann calculada especificamente para cada imagem pelo algoritmo JPEG Baseline e o VBFB foi subamostrado por um fator de 4 e codificado em *Run-Level* usando-se uma tabela de Huffmann própria (como descrito no Capítulo 5). Essa tabela foi anexada ao vetor codificado no cabeçalho da imagem codificada. Para o sistema baseado no MPEG-2, foi utilizado fator de subamostragem igual a 4 e a tabela de codificação VLC fixa especificada no anexo B padrão. A vantagem de se utilizar esta tabela fixa é que não há necessidade de sua inclusão no cabeçalho da imagem, reduzindo, assim, o *overhead* gerado.

No capítulo 6 foram apresentados os resultados obtidos pela aplicação do procedimento proposto para redução de efeito de bloco em dois ambientes de compressão distintos: o padrão de compressão de imagens JPEG e um sistema baseado na quantização e codificação do MPEG-2 TM5. O objetivo de realizar testes com o MPEG-2 é possibilitar a implementação futura de uma versão desse procedimento para vídeo.

Para os resultados apresentados, utilizou-se como ambiente de programação o *Matlab 5.0* principalmente as *Toolboxes* de processamento de sinais e de imagens além de outras funções que foram desenvolvidas em *Matlab script* para as simulações constantes no anexo A. As imagens foram codificadas em diferentes taxas de compressão utilizando-se blocos de tamanho 8x8.

Foram utilizadas como imagens-teste no padrão JPEG as imagens: *Flowers* (512x384), *Foto Aérea* (512x512), *Lena* (512x512), *Pirate* (1024x1024) e *Zelda* (256x256); variando-se as taxas de compressão em três valores distintos: 0,8bpp, 0,6bpp e 0,4bpp. Os resultados para o padrão MPEG-2 foram obtidos a partir da compressão das imagens *Foto* Aérea (512x512) e *Lena* (512x512) a duas taxas de compressão distintas, 0,8bpp e 0,6bpp. Procurou-se selecionar para este trabalho várias imagens com características diversificadas, de forma que se pudesse avaliar o comportamento do procedimento proposto sob várias condições diferentes.

A imagem *Flowers* tem características interessantes no que se refere ao estudo do efeito de bloco por apresentar áreas completamente planas, como o pano de fundo da foto; áreas de poucos detalhes como as superfícies das pétalas; e áreas de grande detalhamento como as folhas dos ramos. A imagem *Foto Aérea*, ao contrário da anterior, tem por característica a grande presença de detalhes espalhados por toda a imagem e também muitas formas geométricas irregulares como o desenho quase triangular formado pelas três estradas. Essas formas geométricas são interessantes para o estudo do efeito de bloco pois as perdas da quantização tendem a entrecortar as linhas inclinadas com relação à bordas da imagem, ressaltando suas fronteiras (gerando a blocagem).

A imagem *Lena* é certamente uma das imagens mais utilizadas nos artigos e em livros de Processamento Digital de Imagens atualmente, e tem, como característica principal, a presença de grandes áreas planas e suaves na maior parte da imagem, proporcionando uma alta eficiência de compactação de energia pela transformada e portanto, uma boa qualidade visual mesmo à altas taxas de compressão. O motivo principal que levou à utilização das imagens *Pirate* e *Zelda* foi o de verificar o comportamento do procedimento proposto para imagens de várias dimensões. Além disso, nessas duas imagens ocorre uma contraposição interessante: a imagem *Pirate* tem dimensões grandes e apresenta muitos detalhes e a imagem *Zelda* é pequena e relativamente suave.

A análise dos resultados obtidos foi baseada tanto na qualidade subjetiva quanto nas medidas de qualidade objetivas PSNR (*Peak Signal-to-Noise Ratio*) e TEB (Taxa de Erro de Bits no VBFB), sendo esta última uma das contribuições desta pesquisa. A escolha do PSNR é devido à sua extensa utilização tanto nos artigos quanto na bibliografia de PDI como medida de qualidade objetiva das imagens. É importante lembrar que um valor de PSNR alto não implica necessariamente em uma boa qualidade de imagem reconstruída, pois não leva em consideração o conjunto geral da imagem, mas apenas o valor de seus *pixels* isoladamente. Esse fato é de fundamental relevância para esse trabalho por constituir um dos motivos da realização dessa pesquisa. Uma medida que revela um pouco melhor a qualidade subjetiva da imagem com relação ao efeito de bloco é a TEB.

Tendo em vista que esse trabalho visou obter uma melhora da qualidade visual das imagens, foram realizadas ampliações de duas regiões de dimensões 48x64 pixels para cada imagem recuperada às taxas de 0,8bpp, 0,6bpp e 0,4bpp, de forma a possibilitar a visualização detalhada do efeito de bloco e das diferenças na qualidade subjetiva das imagens. Essas imagens são monocromáticas e foram apresentadas originalmente em 8 bits por pixel (256 níveis de cinza). Juntamente com as imagens recuperadas foram colocadas as informações de PSNR e Taxa de Erro de Bits para facilitar a avaliação da qualidade da imagem.

Para cada imagem recuperada com taxas de compressão de 0,6 bpp, apresentou-se um gráfico da Função de Borda e um gráfico do VBFB para uma região pertencente à Ampliação I. Para facilitar a visualização, todos os gráficos apresentaram duas curvas, uma para o JPEG (ou MPEG-2) e outra para o Procedimento Proposto. Também foram colocados alguns comentários relevantes a respeito de cada ampliação e seus gráficos correspondentes. A análise completa (subjetiva e objetiva) dos resultados obtidos e a tabela contendo o resumo de todos os resultados numéricos foram apresentadas após as imagens, na subseção 6.3.

Analisando os resultados obtidos, pôde-se observar que, para todas as imagens utilizadas, houve uma considerável melhora na qualidade visual especialmente nas regiões mais planas como as mostradas na Figs. 6.2(c,d), 6.8 (e,f) e 6.14(c,d). Com base na análise do procedimento adotado, pôde-se creditar essa melhora em grande parte ao critério de medida MBFB, que baseia sua medida na linearidade (suavidade) dos contornos da imagem. Essa medida de linearidade proporcionou uma melhor aproximação numérica para a qualidade subjetiva da imagem provendo, assim, uma referência muito boa tanto para a localização de distorções, quanto para a correção das bordas geradas. Notou-se também que para imagens que apresentavam originalmente pouco efeito de

bloco, como as Fig, 6.8(a,b), 6,9(a,b) e 6.15(a,b), o procedimento proposto praticamente preservou a imagem original, evitando, assim, um indesejável borramento dos detalhes da imagem original.

Ocorreram, no entanto, algumas áreas em que o procedimento proposto suavizou algumas pequenas bordas existentes na imagem original, prejudicando um pouco os detalhes da imagem. Analisando-se os gráficos de FB, VBFB e o processo de codificação, concluiu-se que as distorções causadas pelo procedimento proposto foram devido à dois fatores: a subamostragem da função VBFB que acarretou em perda da informação da imagem original; e a taxa de compressão um pouco maior dos dados da imagem para que o VBFB codificado pudesse ser inserido no cabeçalho, sem que ocorresse redução na taxa de compressão total da imagem.

Outra análise realizada foi a dos resultados numéricos (medidas de qualidade) resumidos nas Tabelas VI.1 e VI.2. Observando essas tabelas, pôde-se notar que à medida em que cresce o nível de blocagem da imagem recuperada, o TEB é aumentado (tanto para o JPEG/MPEG-2 quanto para o Procedimento Proposto). Esse comportamento foi um bom indicativo com relação à correspondência numérica da medida TEB com o nível de blocagem da imagem. O TEB (Taxa de Erro de Bit) do procedimento proposto foi sempre numericamente menor do que o TEB do padrão utilizado para comparação (JPEG/MPEG-2). Isso era de se esperar, pois o procedimento interpolativo proposto elimina uma grande quantidade das bordas geradas pela codificação sem causar um significativo borramento das bordas originais, aproximando assim, o VBFB da imagem recuperada do VBFB da imagem original. Apesar de ainda não apresentar uma relação totalmente linear entre a degradação visual e a medida numérica, o TEB apresentou uma correspondência à qualidade subjetiva da imagem, estimulando um estudo mais profundo que poderá ter resultados positivos.

Com relação ao PSNR, pôde-se ver que o procedimento proposto sempre apresenta um resultado um pouco abaixo do resultado para o JPEG. Esse fato era esperado porque, para se inserir o VBFB (com sua tabela de códigos) na imagem, reduziu-se o espaço reservado para os dados da imagem em si, ou seja, provoca-se uma quantização mais grosseira dos coeficientes para permitir uma maior compressão dos dados e manter o número de bits final igual. Como a quantização afeta todos os coeficientes do bloco e a interpolação afeta apenas a região de fronteira, é natural que haja uma pequena queda no PSNR da imagem. Este diferença tende a diminuir à medida em que forem desenvolvidos métodos de codificação mais eficientes para o VBFB, o que constitui-se como uma das possíveis continuações deste trabalho.

Outro detalhe interessante nos resultados numéricos foi o fato de que o *overhead* gerado no sistema proposto foi praticamente constante com a variação do tamanho da imagem, tendo sido um

pouco maior para a imagem *Foto Aérea* e um pouco menor para a imagem *Lena*. A explicação para essas diferenças foi o nível de detalhamento dos blocos da imagem. No caso da imagem *Lena*, a maior parte dos blocos é suave proporcionando uma grande incidência de 0's consecutivos no VBFB, aumentando a eficiência da codificação *Run-Length* e portanto, reduzindo o tamanho do *overhead* gerado. O contrário ocorre na imagem *Foto Aérea*, que apresenta uma enorme quantidade de detalhes espalhados praticamente por toda a imagem, o que proporciona muitas trocas de bits na seqüência do VBFB, diminuindo os *Runs* e reduzindo a compressão do codificador.

Nas outras imagens utilizadas houve um equilíbrio entre os fatores que colaboram para a compressão do VBFB: na imagem *Flowers*, o centro da imagem é bastante detalhado, mas o fundo é praticamente constante. A imagem *Pirate* o grande número de detalhes é compensado pelo maior número de blocos, fazendo com que cada bloco seja semelhante ao seu vizinho, aumentando a linearidade entre eles. O efeito inverso ocorre na imagem *Zelda*, que é bastante suave, mas os blocos são relativamente maiores, proporcionando uma menor compactação.

Comparando-se os resultados obtidos sobre os *overheads* gerados para o MPEG-2 e para o JPEG, observou-se que houve apenas uma pequena variação do valor final inserido no cabeçalho (um pouco menor no JPEG). Entretanto, para o sistema baseado no MPEG, utilizou-se uma tabela de codificação entrópica fixa (a tabela B.14 especificada em [22]) que foi especificamente projetada para se trabalhar com dados de imagem. Assim, a taxa de compressão esperada do VBFB codificado para o MPEG-2 seria bem menor pela perda de otimização, mas o fato de a tabela código não precisar ser inserida no cabeçalho gerou uma boa compensação fazendo com que os *overheads* fossem praticamente iguais.

Esse resultado nos indica que a realização de um estudo estatístico das propriedades do VBFB no intuito de implementar uma tabela código fixa para ele é extremamente válido, podendo resultar em taxas de compressão mais significativas. Este estudo pode colaborar para a implementação do procedimento proposto para seqüências de vídeo.

Fazendo uma avaliação geral, este trabalho apresentou contribuições dentro da proposta do aumento da qualidade subjetiva de imagens codificadas por transformada, como, por exemplo, a medida de presença de borda MBFB, a medida de blocagem TEB e a proposta de um sistema básico para redução do efeito de bloco para o padrão JPEG. Esse sistema básico apresenta muitos pontos que sempre podem (e devem) ser melhorados. Algumas idéias para esses melhoramentos encontram-se no item "Sugestões para Trabalhos Futuros". Outro ponto positivo deste trabalho foi a realização de estudos iniciais para implementação do mesmo sistema para o codificador de vídeo

MPEG-2. Nesse caso, o estudo pode ser complementado acrescentando-se as imagens com movimento. Foi realizada apenas a parte estática.

Este trabalho será finalizado apresentando-se, à luz dos resultados obtidos, as perspectivas de continuações e avanços que poderão ser implementados. No Apêndice A são mostrados os códigos-fonte dos principais programas desenvolvidos para a realização deste trabalho.

7.2 – Sugestões para Trabalhos Futuros

Este trabalho apresenta, como uma das características, uma ampla gama de estudos para contribuições adicionais dentro de vários de seus aspectos visando melhorar a eficiência do sistema proposto ou a adaptação desta proposta a outros sistemas de compressão. Algumas sugestões interessantes que foram visualizadas durante a realização desta pesquisa são:

- A implementação do sistema proposto para imagens coloridas e para outros padrões de compressão de imagem;
- Com relação à melhoria de qualidade visual da imagem recuperada, pode-se realizar um estudo para a determinação de um método de correção das bordas perdidas pela codificação, visto que o método proposto já tem como identificar esses pontos de perda;
- Um estudo estatístico do VBFB mudando as características da sua codificação de forma a torná-lo mais eficiente e proporcionar uma redução da subamostragem melhorando a qualidade subjetiva da imagem recuperada;
- A alteração do Critério de Varredura na tentativa de formar um VBFB mais correlato, aumentando a eficiência da sua codificação;
- Estudar mais profundamente a relação de sensibilidade do olho humano à variações de brilho, matiz e saturação levando em conta a distância de observação, o tamanho da imagem e outros parâmetros que possam ser relevantes;
- Realizar a adaptação do sistema para um codificador de vídeo tentando explorar as estatísticas temporais na codificação do VBFB completando-se a aplicação em sistema de vídeo que usem o MPEG-2.

Assim, foram listadas as principais contribuições e resultados do presente trabalho em conjunto com algumas sugestões que espera-se, possam auxiliar o desenvolvimento de outras pesquisas relacionadas com o processamento e a compressão digital de imagens e vídeo.

Bibliografia Fundamental

[1] – John Watkinson – "MPEG-2" – Ed. Focal Press, 1999.

[2] – Kenneth R. Castleman – "Digital Image Processing" – Ed. Prentice Hall, 1989.

[3] – R. J. Clarke – "Transform Coding of Images" – Academic Press, 1985.

[4] – Anil K. Jain – "Fundamentals of Digital Image Processing" – Ed. Prentice Hall, 1989.

[5] – T. Natarajan, N. Ahmed and K.R. Rao – "Discrete Cosine Transform" – IEEE Trans. Comput., 1974.

[6] – Vasudev Baskaran and Konstantinos Konstantinides – "Image and Video Compression Standards" – Ed. Kluwer Academic Publisher, 1996.

[7] – "Handbook of Image and Video Processing" – Editor Al Bovik, Ed. Academic Press, 2000.

[8] – Evaldo G. Pelaes – "Transformada Seno Discreta com Rotação de Eixos Bidimensional (DSTr-2D): Aplicações na Codificação e Interpolação de Imagens para a Redução de Efeito de Blocos" – Dissertação de Doutorado apresentada à FEEC da UNICAMP, Campinas - SP, Brasil, 1998.

[9] – Henrique S. Malvar – "Signal Processing with Lapped Transforms" – Ed. Artech House, 1992.

[10] – Michael P. Ekstrom – "Digital Image Processing Techniques" – Ed. Academic Press, Inc., 1984.

[11] – H. C. Andrews and B. R. Hunt – "Digital Image Restoration" – Ed. Englewood Cliffs, N. J.: Prentice-Hall, 1977.

[12] – Gregory A. Baxes – "Digital Image Processing, Principles and Applications" – Ed. John Wiley & Sons, Inc., 1994.

[13] – Michael Yuen and H.R. Wu – "Reconstruction Artifacts in Digital Video Compression" – Proceedings of VCIP, vol 2419, pp 455-465, 1995.

[14] – Rama Chellappa and Alexander Sawchuk – "Digital Image Processing and Analysis" – Ed. Prentice Hall, 1989.

[15] – H. Reeve and J.S.Lim – "Reduction of Blocking Effects in Image Coding"- in *Opt. Eng*. Vol.23, no.1, pp 34-37, 1984.

[16] – K.-H. Tzou – "Post Filtering of Transform-Coded Images"- *in Applications of Digital Image Process*, XI Proceedings SPIE, vol 974 pp 121-126, 1988.

[17] – N.B. Pannebaker and J. L. Mitchell – "JPEG Still Image Compression Standard" – New York: Van Nostrand Reinhold, 1992.

[18] – S.-W. Hong, Y.-H. Chan and W.-C. Siu – "The Neural Network Modeled POCS Method for Removing Blocking Effect " – in *IEEE Int. Conf. Neural Networks (ICNN´95)*, vol III pp 1422-1525, Nov 1995.

[19] – B.L.Hinmam, J.G.Bernstein and D.H.Staelin – "Short-space Fourier Transform Image Processing" – in *IEEE Intl Conf. Acoust., Speech, Signal Processing*; San Diego, CA, pp 481-484, March, 1984

[20] – Fernando C. Joo – 'Simulações para a Codificação de Imagens usando o padrão JPEG Básico" – Dissertação de Mestrado apresentada à FEEC da UNICAMP, Campinas - SP, Brasil, 1995.

[21] – ISO/IEC JTC1/SC9/WG11, Coded Representation of Pictures and Audio Information, Test Model 5, April 1993.
http://www.mpeg.org/MPEG/MSSG/tm5.

[22] – ISO/IEC 13818-2, Recommendation ITU-T H262, 1996.

[23] – http://www.mpeg.org/MPEG/MSSG – MPEG Software Simulation Group Home Page.

[24] – Peter D. Symes – "Video Compression" – Ed. McGraw-Hill, New York, 1998.

[25] – Jerry D. Gibson, Toby Berger, Tom Lookabaugh, Dave Lindberg and Richard L. Baker – "Digital Compression for Multimedia" – Ed. Academic Press, London-UK, 1998.

[26] – Maria Petrou and Panagiota Bosdogianni – "Image Processing, the Fundamentals" – Ed. John Wiley & Sons Ltd, New York, 1999.

Apêndice A

Listagem dos programas desenvolvidos

Para a realização das simulações foram implementadas várias funções em MATLAB *script*, utilizando-se principalmente a *Toolbox* de processamento de imagem, além de outras funções auxiliares desenvolvidas para se realizar as simulações.

As imagens originais que foram utilizadas estão em formato *bitmap* (bmp) com precisão de 8 bits por *pixel*. Para obter-se as imagens com taxas de 0,8bpp, 0,6bpp e 0,4bpp compactadas com o algoritmo JPEG básico, foi utilizada a função *imwrite* do Matlab, variando o fator de qualidade através do parâmetro *'Quality'* de forma a se obter o valor mais próximo ao desejado.

Como o código fonte da função *imwrite* não está disponível, para implementar o procedimento proposto foi feita uma adaptação: ao invés de anexar o *overhead* ao *bitstream* codificado, criou-se um arquivo auxiliar no qual foi gravado o *overhead*. Dessa forma, o resultado em termos de número de bits e qualidade de imagem é mantido possibilitando a utilização de funções pré-implementadas e uma maior dedicação ao procedimento proposto.

O MPEG-2, no entanto, está implementado em C, de forma que a sua interface com o Matlab teve que ser feita manualmente, ou seja, foram usadas as funções de levantamento de VBFB e de redução de efeito de bloco do JPEG (pois são independentes do padrão de compressão) e foram desenvolvidas algumas funções para a interface entre os arquivos de imagens. A obtenção das imagens compactadas a 0,8bpp e 0,6bpp foi realizada manualmente variando o parâmetro *bit_rate* do codificador. Dessa forma, para o MPEG-2 não foi montado o arquivo auxiliar com o *overhead* codificado, mas apenas calculado o número total de bits que efetivamente seria necessário para a sua criação.

Assim, são apresentadas a seguir, as funções principais para ambos os sistemas de compressão propostos na forma em que foram utilizadas para a obtenção dos resultados mostrados no Capítulo 6. Na primeira seção são mostrados os programas para o codificador JPEG e na segunda seção, os programas suplementares de conversão de arquivos para o codificador MPEG-2. Cabe ressaltar que inúmeros outros programas foram desenvolvidos para a obtenção dos resultados, mas não são apresentados porque não são essenciais para a compreensão do trabalho proposto.

A.1 – Programas para o JPEG

Programa em MATLAB para a criação das imagens comprimidas pelo JPEG básico nas taxas de 0,8bpp, 0,6bpp e 0,4bpp para as imagens *Flowers*, Foto Aérea (fotoaerea), *Lena*, *Pirate* e *Zelda*. As imagens originais estão armazenadas no subdiretório 'jpeg' e as imagens comprimidas são armazenadas no mesmo diretório com o nome original seguido da taxa de compressão. Exemplo: 'Lena_080.jpg'.

```
function jpeg_basico()

% Definição dos nomes dos arquivos e dos bpp's que vão ser utilizados
nomes={'flowers'; 'fotoaerea'; 'lena'; 'pirate'; 'zelda'};
bpps=[ 0,80; 0,60; 0,40 ]

% Cálculo do número de imagens e do numero de bpp's
no_im=length(nomes);
no_bpp = length(bpp);

% Loop para todas as imagens
for i=1:no_im
    basename=char(nomes(i));
    narq=sprintf('jpeg\\%s.bmp',basename);

    % Leitura da Imagem Original - yo
    yo=imread(narq);

    % Loop para todos os bpp's
    for j=1:no_bpp
        bpp=bpps(j);
        grava_jpeg(yo,basename, bpp)
    end
end
```

```
function bpp_final=grava_jpeg(Y, name, bpp)

% Inicialização das variáveis
OK=0;
qual_a =100;
narq=sprintf('jpeg\\%s_%03d.jpg', name, bpp);
imwrite(Y, narq, 'quality', qual_a);
info=imfinfo(narq);
bpp_a=info.FileSize*8/prod(size(yo));
qual_b=qual_a - ceil(10*(bpp_a-bpp));
```

```
% Loop de procura do bpp mais próximo
while (OK==0)
      imwrite(yo,narq,'quality',qual_b);
      info=imfinfo(narq);
      bpp_b=info.FileSize*8/prod(size(yo));
      if (abs(bpp_a-bpp)<= abs(bpp_b-bpp))
            OK==1;
            imwrite(yo,narq,'quality',qual_a);
            bpp_final=bpp_a;
      else
            qual_a=qual_b; bpp_a=bpp_b;
            qual_b=qual_b - ceil(10*(bpp_b-bpp));
      end
end
```

Programa em MATLAB para o cálculo do VBFB com um fator de subamostragem *sbfac* e um limiar de percepção *LP* da imagem *yo* de dimensões múltiplas de 8 e representada em 8 bits por pixels (níveis de 0 a 255).

```
function VBFB = calc_BBEV(Imagem_orig, sbfac, LP);

% Testes de formato de imagem
Tam_im = size (Imagem_orig);
if sum(rem(Tam_im,8))~=0
   error ('A imagem deve ter dimensões em pixels multiplas de 8');
end
if  ~strcmp(class(Imagem_orig),'uint8')
       error ('A imagem deve ter precisão de 8 bits ( tipo uint8)');
end

% Inicialização de variáveis e conversão da imagem
Imagem_orig=double(Imagem_orig)/255;
Num_lin = (Tam_im(1) / 8)-1;
Num_col = (Tam_im(2) / 8)-1;

% Teste de validade para o Fator de Subamostragem e alocação do VBFB
comp = Num_col*Tam_im(1) + Num_lin*Tam_im(2);
comp_sb=comp/sbfac;
if (mod(comp_sb,1))
      error('Fator de Subamostragem inválido');
else
      BBEV =zeros(comp,1);
end

% Percorrendo as Linhas Verticais para calcular o VBFB
for j=1:Num_col;
      for i=1:Tam_im(1);
            D1 = ( Imagem_orig(i,8*j-1) - Imagem_orig(i,8*j) );
```

```
            D2 = ( Imagem_orig(i,8*j) - Imagem_orig(i,8*j+1) );
            D3 = ( Imagem_orig(i,8*j+1) - Imagem_orig(i,8*j+2) );

            if ( (abs(D1-D2)<LP) & (abs(D2-D3)<LP) )
                  BBEV ( Tam_im(1)*(j-1)+i )= 0;
            else
                  BBEV ( Tam_im(1)*(j-1)+i )= 1;
            end
      end
end

% Percorrendo as Linhas Horizontais para gerar Overhead %
for i=1:Num_lin;
      for j=1:Tam_im(2);
            D1 = ( Imagem_orig(8*i-1,j) - Imagem_orig(8*i,j) );
            D2 = ( Imagem_orig(8*i,j) - Imagem_orig(8*i+1,j) );
            D3 = ( Imagem_orig(8*i+1,j) - Imagem_orig(8*i+2,j) );
            if ( (abs(D1-D2)<LP) & (abs(D2-D3)<LP) )
                  BBEV ( Tam_im(2)*(i-1)+j + Tam_im(1)*Num_col )= 0;
            else
                  BBEV ( Tam_im(2)*(i-1)+j + Tam_im(1)*Num_col )= 1;
            end
      end
end

% Execução da Subamostragem e teste de multiplicidade entre VBFB e sbfac
aux=zeros(comp_sb,1);
for j=1:sbfac
      aux= aux + BBEV (j:sbfac:comp);
end
      BBEV =aux>(sbfac/2);
end
```

Programa em MATLAB para o tratamento do *VBFB* da imagem *name*: geração do *overhead* através da codificação de Huffmann e o seu armazenamento no arquivo *name_over.dat.*

```
function nbits=calc_overhead(BBEV, name)

% Cálculo do vetor Run-Length Binário (já subtraído de 1) e determinação do init_bit
RLB_vector=rlbcode(BBEV)-1;
if RLB_vector(1)==-1
      initbit=1;
      RLB_vector=RLB_vector(2:length(RLB_vector));
else
      initbit=0;
end

% Mapeamento do vetor RLB para Run-Level
```

```
runs=calc_run(RLB_vector);
ampl=zeros(size(runs));
comp=length(runs);
k=0;
for i=1:comp
      k=k+runs(i)+1;
      ampl(i)=RLB_vector(k);
end
levels=floor(log2(ampl+1));

% Cálculo do vetor de freqüência de ocorrência (probabilidades) para os pares R/L (até o máximo
%    de 14/15)
prob=zeros(15,15);
for i=1:comp
      if () (runs(i)<15 & levels(i)<16)
            prob(runs(i)+1,levels(i))=prob(runs(i)+1,levels(i))+1;
      else
            buf=sprintf('ERRO: i= %d; run= %d; level= %d', i, runs(i), levels(i));
            disp(buf);
      end
end
probv=im2col(prob,size(prob),'distinct')/sum(sum(prob));

% Cálculo e armazenagem do Código de Huffmann no arquivo de overhead
narq=sprintf('jpeg\\%s_over.dat',name);
fpr=fopen(narq,'wb');
code=calc_huff(probv);
codelen=zeros(size(prob));

for i=1:length(code)
      codelen(mod(i-1,15)+1,ceil(i/15))= length(char(code(i)));
      putbits(fpr, length(char(code(i))),4);
      putbits(fpr, bin2dec(char(code(i))), length(char(code(i))));
end
cmed=mean2(codelen);

%Cálculo do número de bits total usado pelo overhead.

% Tabela de Huffmann
nbits= 4*225 + sum(sum(codelen));

% Armazenagem do overhead e cálculo do número de bits total
nbits= nbits+1;
putbits(fpr, init_bit, 1);
 for i=1:comp
      nbits=nbits+codelen(runs(i)+1,levels(i))+levels(i);
      putbits(fpr,  bin2dec(char(code(runs(i)*15+levels(i)))), codelen(runs(i)+1,levels(i)));
      putbits(fpr, ampl(i), leves(i));
end
```

```
fclose(fpr);
```

Programas em MATLAB auxiliares para a codificação do VBFB

```
function funcaorlb=rlbcode(funcao)

% Cálculo dos runs binários
comp=length(funcao);
funcaoaux=zeros(size(funcao));
c=0;last=0;k=1;
for i=1:comp
      if ~~funcao(i)==last
            c=c+1;
      else
            funcaoaux(k)=c;
            c=1;last=~last; k=k+1;
      end
end
% Ajuste de tamanho do vetor
naonulos=nnz(funcaoaux)+not(funcaoaux(1));
funcaorlb=funcaoaux(1:naonulos);


function runs=calc_run(funcao)

% Cálculo do valor dos Runs
comp=length(funcao);
funcaoaux=-ones(size(funcao));
c=0; k=1;
for i=1:comp
      if funcao(i)==0
            c=c+1;
      else
            funcaoaux(k)=c;
            c=0; k=k+1;
      end
end

% Ajuste de Tamanho do vestor de Runs
naonulos=comp;
while (funcaoaux(naonulos)==-1)
      naonulos=naonulos-1;
end
runs=funcaoaux(1:naonulos);


function codfinal=calc_huff(probv)

% Inicialização de variáveis
```

```
no_est=nnz(probv)-1;
aux=probv(find(probv~=0));
probv2=zeros(2*no_est+1,1);probv2(1:no_est+1)=aux;
estagio=zeros(no_est,3);
codigo=cellstr(char(32*ones(2*no_est+1)));
probv2(find(probv2==0))=1;

% Determinação da Arvore de Huffmann
for i=1:no_est
    [m1,p1]=min(probv2);
    probv2(p1)=probv2(p1)+1;
    [m2,p2]=min(probv2);
    probv2(p2)=probv2(p2)+1;
    estagio(i,:)=[p1,p2,no_est+i+1];
    probv2(no_est+i+1)=m1+m2;
end

% Associação dos códigos na Árvore
for i=no_est:-1:1
    codbase=codigo(estagio(i,3));
    cod=strcat(codbase,'0');
    codigo(estagio(i,1))=cod;
    cod=strcat(codbase,'1');
    codigo(estagio(i,2))=cod;
end

% Ajuste de tamanho do array de códigos
codfinal=cellstr(char(32*ones(length(probv))));
codfinal(find(probv>0))=codigo(1:no_est+1);


function putbits(fpr, valor, bits)

formato = sprintf('bit%d',bits);
fwrite(fpr, valor, formato);
```

Programa em MATLAB para a redução do efeito de bloco da imagem *im_proc* de dimensões múltiplas de 8 e representada em 8 bits por pixels (níveis de 0 a 255), dados o VBFB da imagem original *funcao* e o fator de subamostragem *sbfac.*

```
function im_rest = restblock(im_proc, funcao, sbfac);

% Testes de formato da imagem
Tam_im = size (im_proc);
if sum(rem(Tam_im,8))~=0
    error ('A imagem deve ter dimensões em pixels multiplas de 8');
```

```
end
if  ~strcmp(class(Imagem_orig),'uint8')
      error ('A imagem deve ter precisão de 8 bits ( tipo uint8)');
end
% Inicialização de variáveis e conversão da imagem
im_proc=double(im_proc)/255;
Num_lin = (Tam_im(1) / 8)-1;
Num_col = (Tam_im(2) / 8)-1;
p=1;
im_rest=im_proc;

% Cálculo do VBFB para a imagem processada
funcao_rest=medblock3(im_proc,1);

% Interpolação (upsampling) dos vetores VBFB
comp=length(funcao)*sbfac;
aux=ones(comp,1);
for j=1:sbfac
      aux1(j:sbfac:comp)=funcao;
end
funcao=aux;

% Percorrendo as Linhas Verticais corrigindo e efeito de bloco
for j=1:Num_col;
      for i=1:Tam_im(1);
            if ( (funcao(p)==0) & (funcao_rest(p)==1) )
                  D = (im_rest(i,8*j-1) - im_rest(i,8*j+2))/3;
                  im_rest(i,8*j)   = im_rest(i,8*j-1) - D;
                  im_rest(i,8*j+1) = im_rest(i,8*j+2) + D;
            end
            p=p+1;
      end
end

% Percorrendo as Linhas Horizontais para corrigir o efeito de bloco %
for i=1:Num_lin;
      for j=1:Tam_im(2);
            if ( funcao(p)==0 & (funcao_rest(p)==1) )
                  D = (im_rest(8*i-1,j) - im_rest(8*i+2,j))/3;
                  im_rest(8*i,j)   = im_rest(8*i-1,j) - D;
                  im_rest(8*i+1,j) = im_rest(8*i+2,j) + D;
            end
            p=p+1;
      end
end

% Conversão da imagem final
im_rest=uint8(im_rest/255);
```

A.2 – Programas para o MPEG-2

Como foi comentado no início do apêndice, nesta seção são apresentadas as rotinas de interface de arquivos de imagem e a rotina de cálculo do número de bits do *overhead* usando a tabela de codificação B.14 do MPEG-2 [22]. Programa em MATLAB para o cálculo do número de bits do *overhead* gerado usando a tabela B.14 do padrão MPEG-2 [9].

```
function nbits=calc_overhead_mpeg(funcao)

% Cálculo do vetor RLB
RLB_vector=runlcode(funcao)-1;
if stream(1)==-1
      initbit=1;
      RLB_vector = RLB_vector (2:length(RLB_vector));
else
      initbit=0;
end

% Mapeamento do vetor RLB em RUN-Level
runs=calcrun(RLB_vector);
ampl=zeros(size(runs));
comp=length(runs);
k=0;
for i=1:comp
      k=k+runs(i)+1;
      ampl(i)=stream(k);
end
levels=floor(log2(ampl+1));

% Inicialização de variáveis
codelen=[
   1  4  5  7  8  8 10 12 12 12 12 13 13 13 13 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14;
   3  6  8 10 12 13 13 15 15 15 15 15 15 15 16 16 16 16 24 24 24 24 24 24 24 24 24 24 24 24;
   4  7 10 12 13 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   5  8 12 13 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   6 10 13 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   6 12 16 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   6 12 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   7 12 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   7 13 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   8 13 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   8 16 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   8 16 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
   8 16 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
  10 16 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
  10 16 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
  10 16 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
  11 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
```

```
  11 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
  11 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
  11 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24;
  11 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24];

% Cálculo do número de bits do overhead
nbits=1;  % Do init_bit
for i=1:comp
      if (levels(i)<14)&(runs(i)<16)
            nbits=nbits+matriz(runs(i)+1,levels(i))+levels(i);
      else
            nbits=nbits+24+levels(i);
      end
end
```

Programa em MATLAB para adquirir uma imagem *Y* (somente luminância) de dimensões *nlin x ncol* a partir de um arquivo gerado pelo codificador MPEG-2 TM5 utilizado nos experimentos e armazenada no subdiretório 'mpeg'.

```
function Y=leframe2(nomearq, nlin, ncol);

% Abertura do arquivo no subdiretório mpeg
narq=sprintf('mpeg\\%s.y',nomearq);
fpr = fopen(narq, 'rb');

% Inicialização de variáveis
Y=zeros(nlin,ncol);
for i=1:nlin
      a=fread(fpr,ncol,'uint8');
      Y(i,:)=double(a')/255;
end
fclose(fpr);
```

Programa em MATLAB para gravar uma imagem *Y* (somente luminância) de dimensões *nlin x ncol* no arquivo *name.y* no formato de entrada do codificador MPEG-2 TM5 utilizado nos experimentos.

```
function gravamatriz (Y, name);

narq=sprintf('mpeg\\%s.y', name);
fpr = fopen(narq, 'wb');
tam=size(Y);
nlin=tam(1);
ncol=tam(2);
for i=1:nlin
      fwrite(fpr,Y(i,:),'uint8');
end
fclose(fpr);
```

Apêndice B

Histogramas das Imagens Originais e Reconstruídas

Nesse apêndice são apresentados os histogramas das imagens originais *Flowers*, *Foto Aérea*, *Lena*, *Pirate* e *Zelda*. Também são mostrados os histogramas das imagens recuperadas pelo JPEG básico, para o sistema de compressão baseado na compressão espacial do MPEG-2 TM5 e pelo procedimento proposto baseado nesses dois sistemas.

Analisando os gráficos obtidos, pode-se observar que a aplicação do procedimento proposto praticamente não altera o histograma da imagem. Isso ocorre porque o procedimento proposto altera apenas 2 pixels dentro de cada REB na qual é caracterizada a geração do efeito de bloco.

Nas Figs. B.1 a B.5 estão os resultados para o JPEG e nas Figs. B.6 e B.7 estão os resultados para sistema baseado no MPEG-2 TM5.

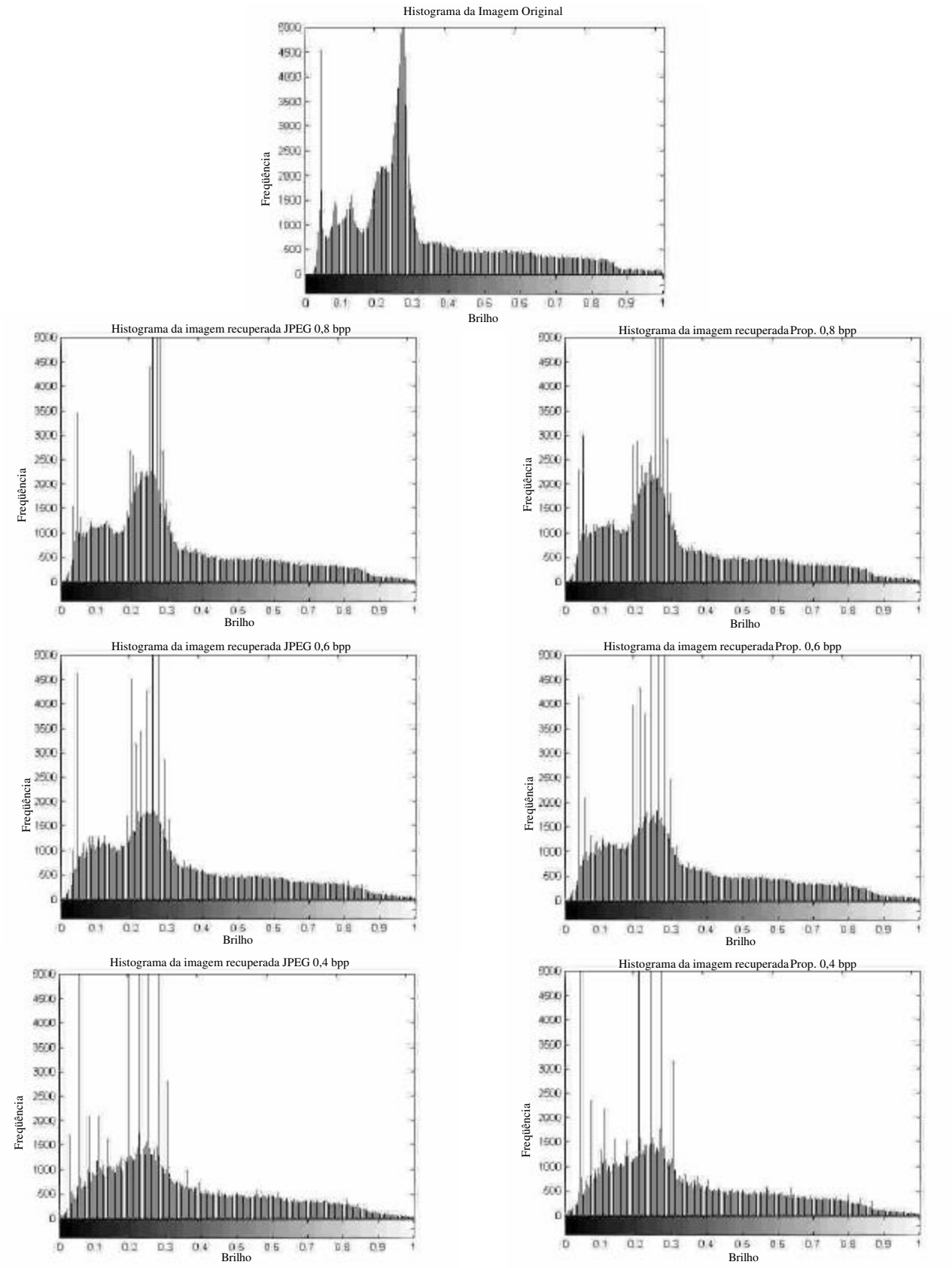


Fig. B.1 – Histogramas para a Imagem *Flowers*

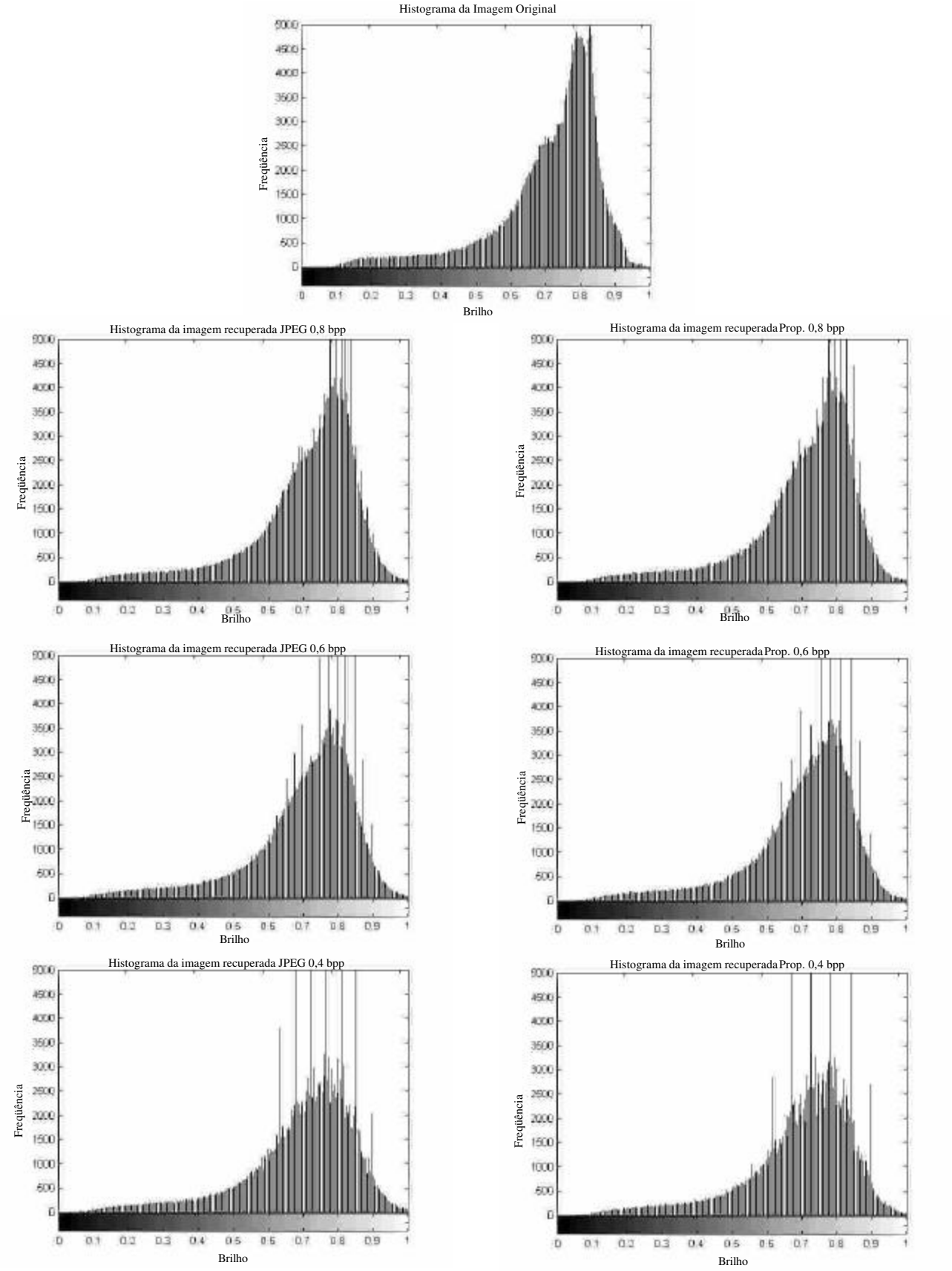


Fig. B.2 – Histogramas para a Imagem *Foto Aérea*

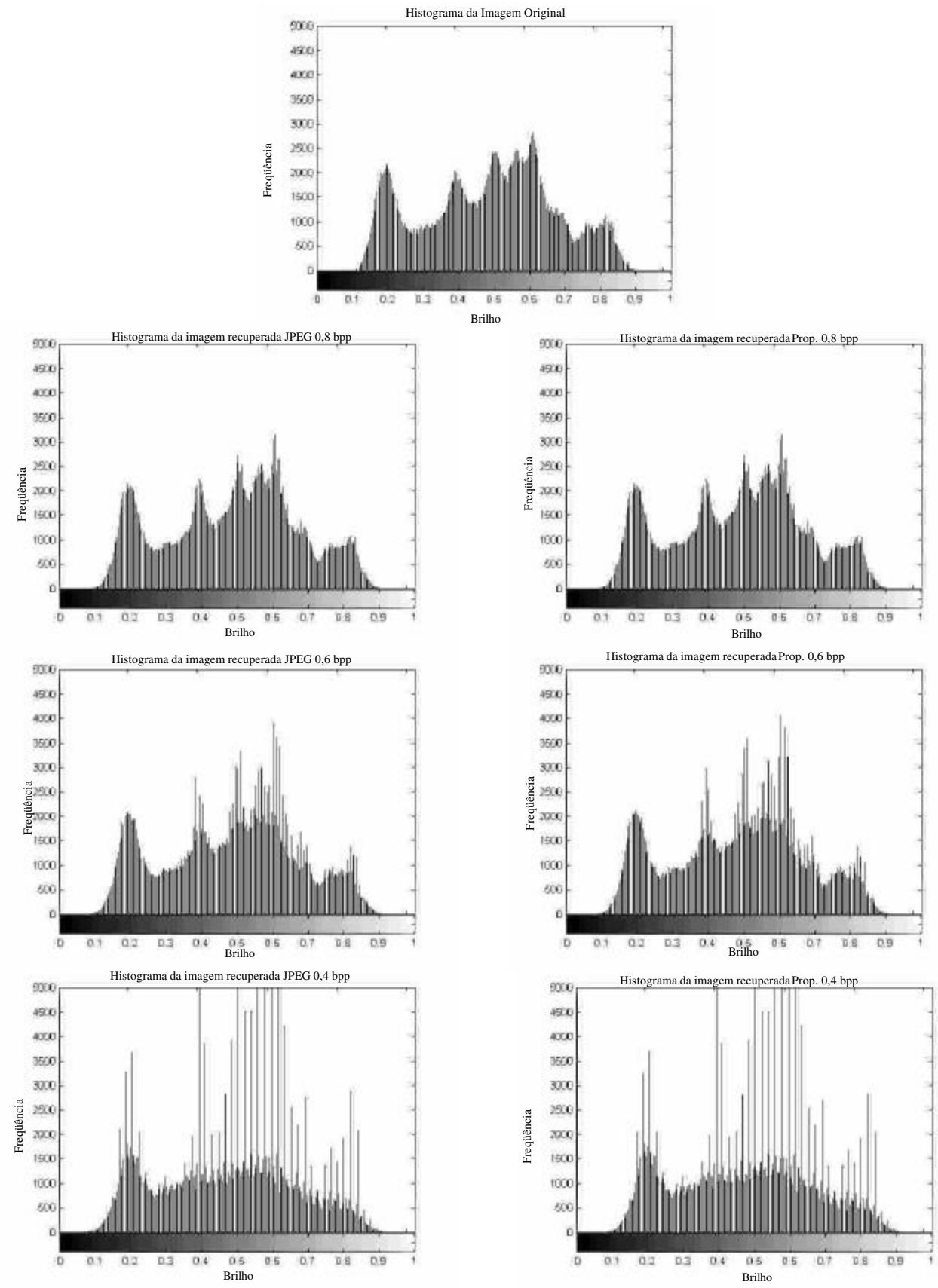


Fig. B.3 – Histogramas para a Imagem *Lena*

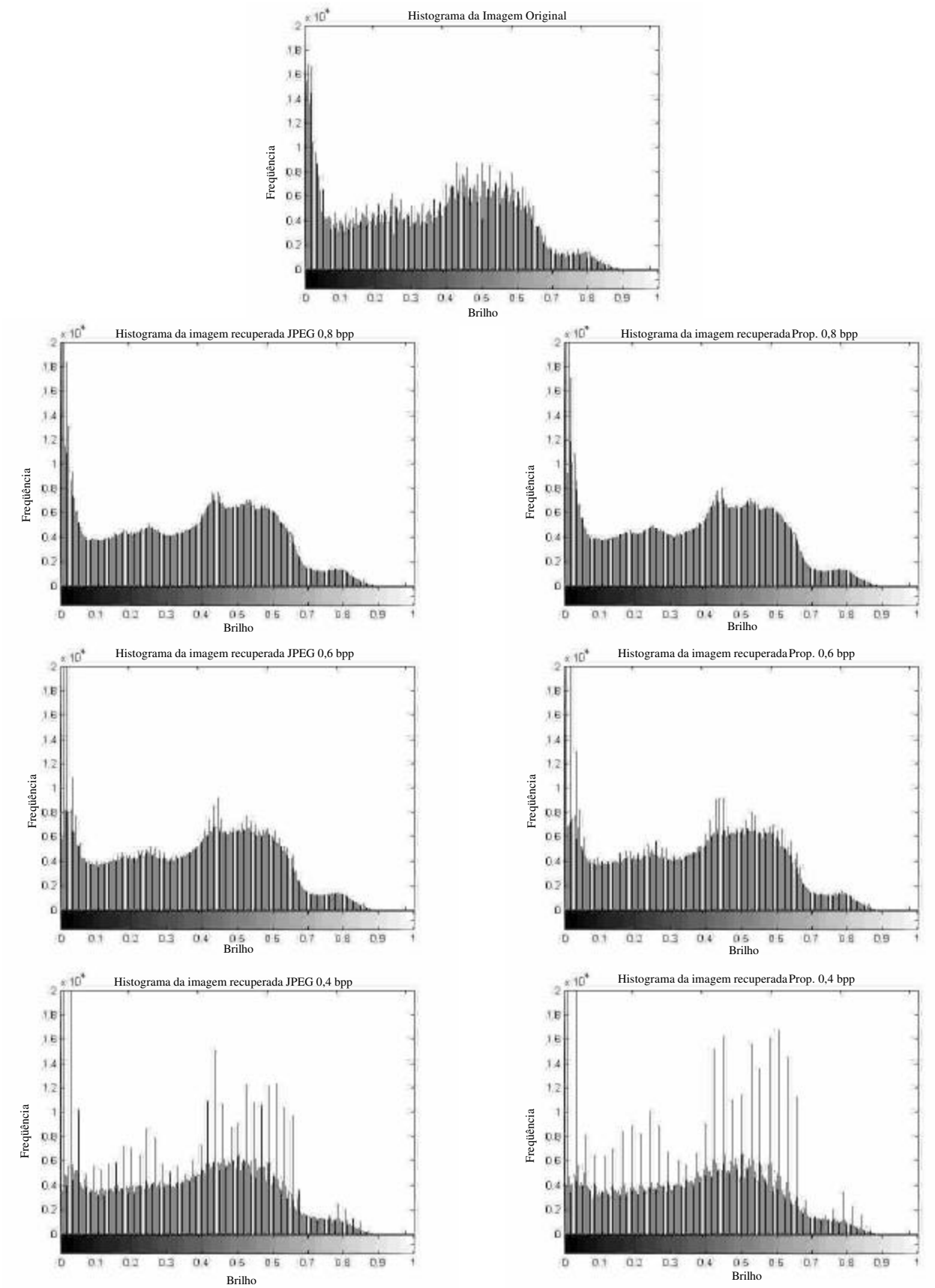


Fig. B.4 – Histogramas para a Imagem *Pirate*

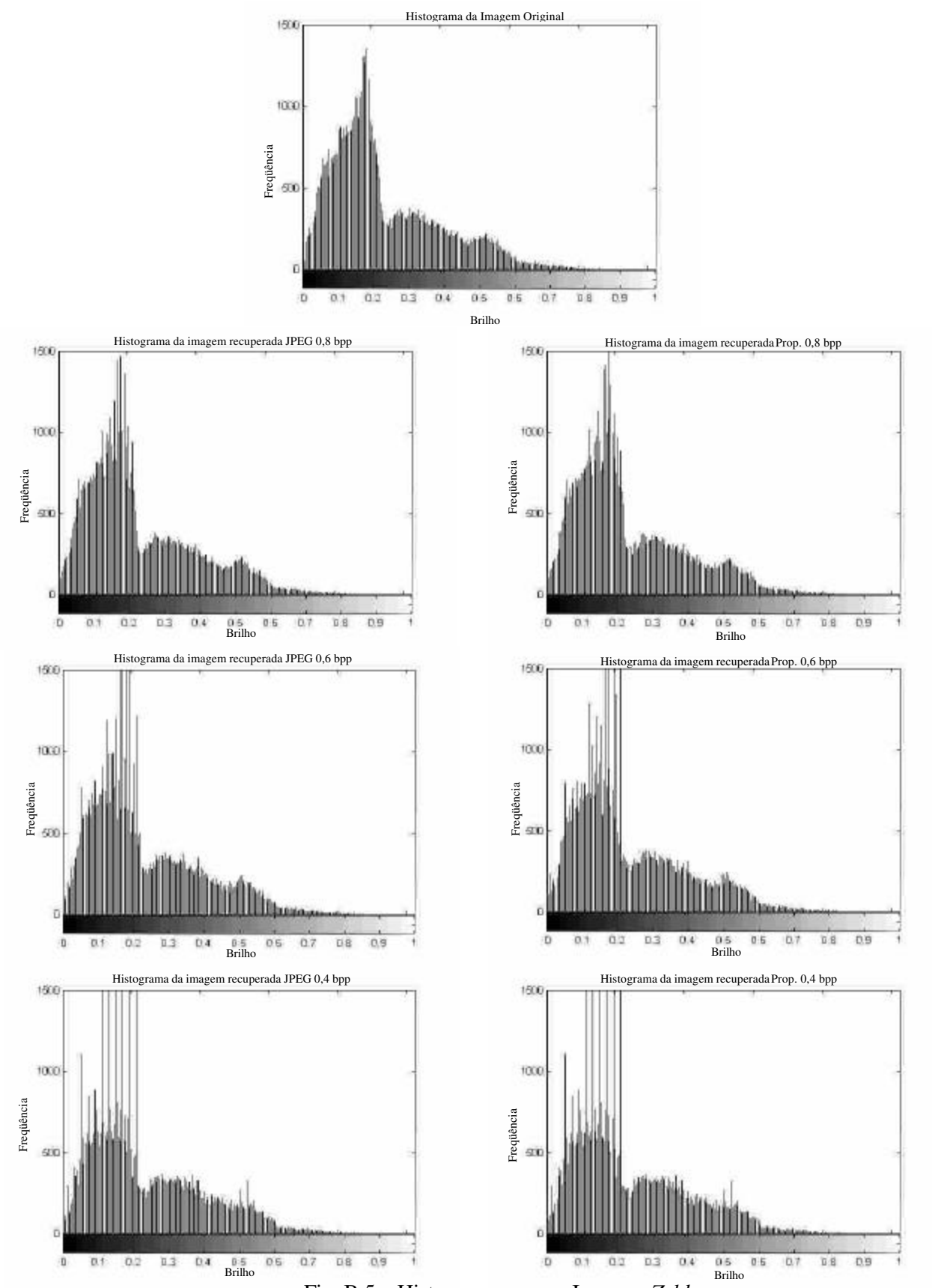


Fig. B.5 – Histogramas para a Imagem *Zelda*

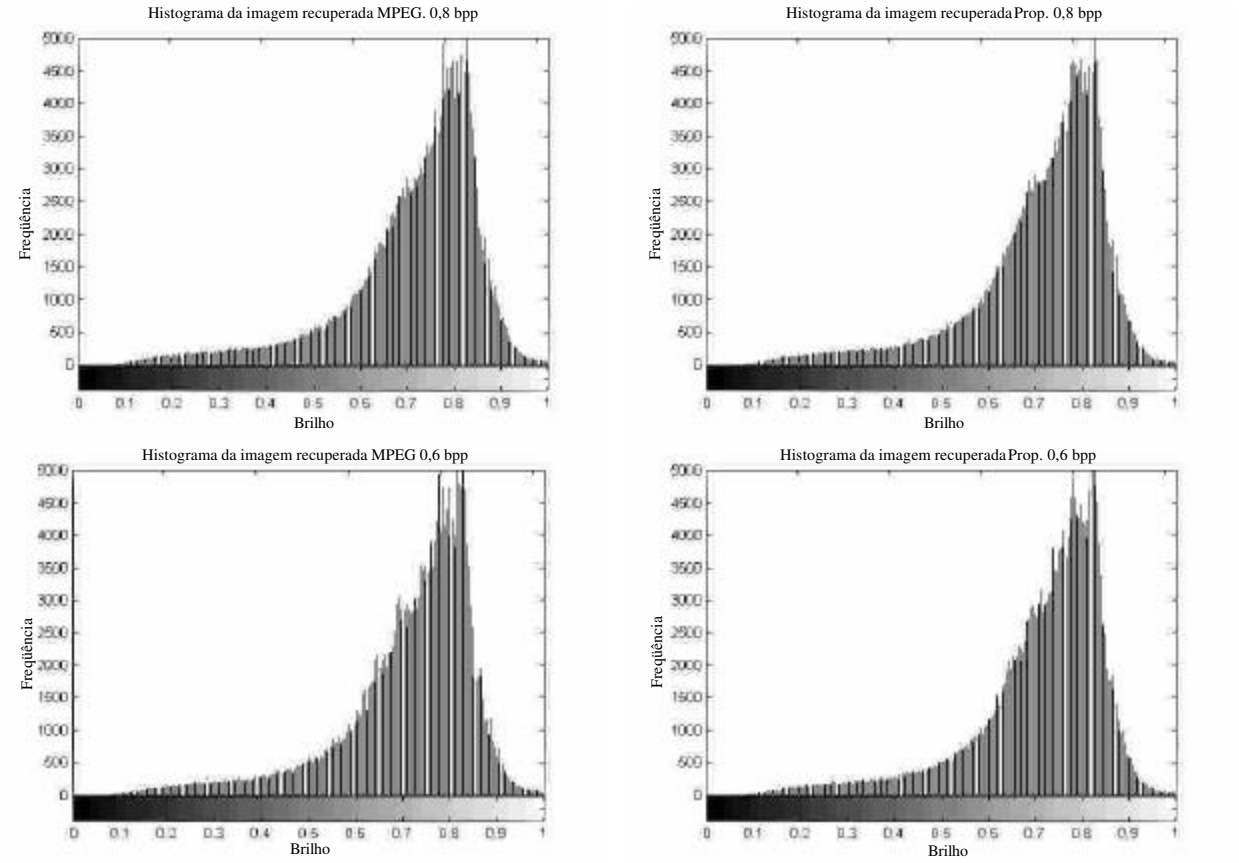


Fig. B.6 – Histogramas para a Imagem *Foto Aérea*

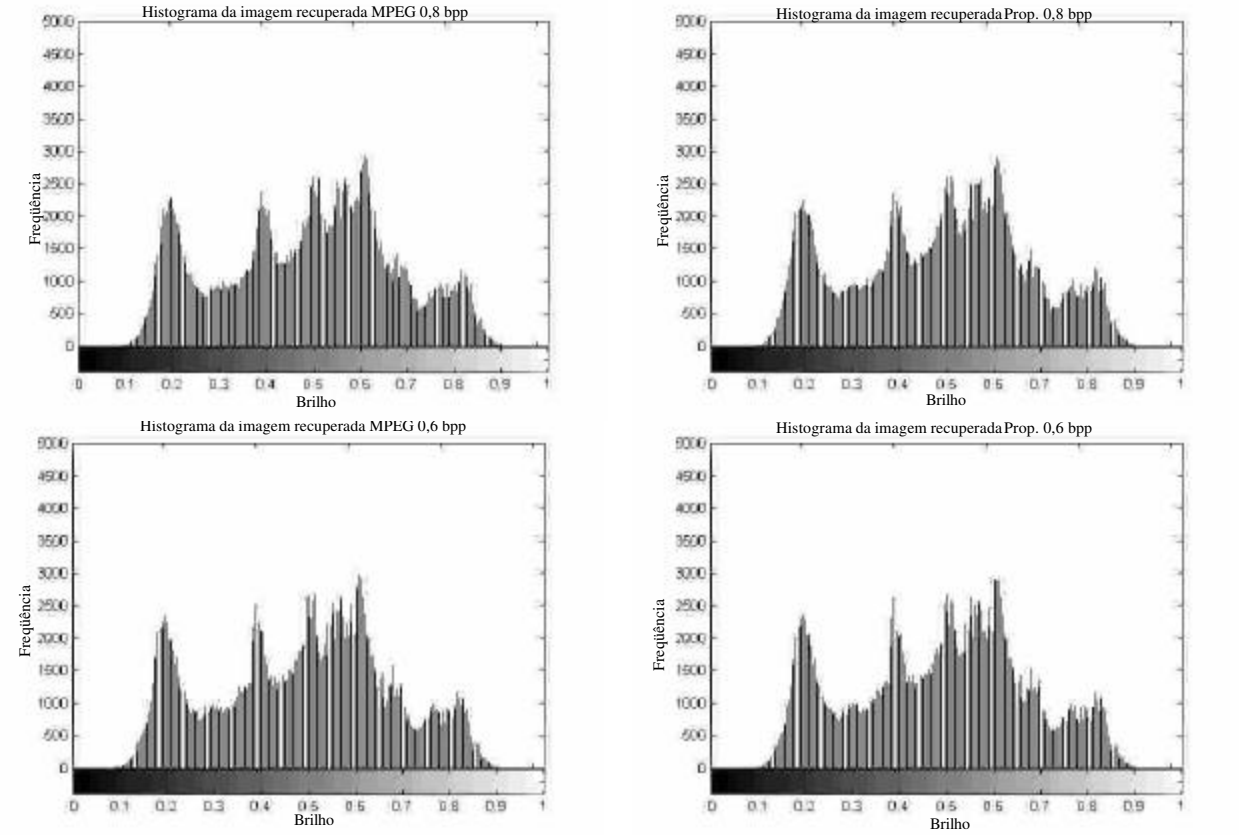


Fig. B.7 – Histogramas para a Imagem *Lena*

Publicações Submetidas

[1] – Fernando Silvestre da Silva, Ana Lúcia Mendes Cruz e Yuzo Iano – *"A Simple Local Method to Reduce Blocking Effect*" – Submetida a revista *IEEE Transactions on Multimedia* em Novembro de 2000.