Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica

Tópicos sobre Codificação Exata de Componentes de Cor em Imagens Digitais Estáticas de TV

Orientando
Abdalla Ibrahim Elusta

Orientador
João B. T. Yabu-uti

Tese apresentada na Faculdade de Engenharia Elétrica, Departamento de Comunicações, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Setembro de 1990

Agradecimentos

Antes de qualquer coisa é preciso agradecer o ótimo trabalho realizado por *Maria Lúcia C. Cardoso* em vários desenhos, bem como o trabalho minucioso e dedicado realizado por *Luiz Roberto Delphim* na editoração eletrônica desta tese.

Agradecemos também a *Ricardo R. Sofia* e *Abraão L. Queiroz*, pelo suporte oferecido no laboratório LCD/DECOM/FEE.

Os agradecimentos aos *Profs. Iuzo* e *Luiz C. Martini*, companheiros na Equipe CDI/FEE/Unicamp, pelo muito que fizeram para que este trabalho pudesse ser realizado. Aproveitamos o momento para agradecer também aos pesquisadores da Equipe PDI/CPqD-Telebrás, na pessoa do *Engº Roberto Vivaldi Rodrigues*, pelo grande apoio técnico oferecido no âmbito do contrato 387/90 - Transmissão Digital.

Finalmente, estes agradecimentos são extensivos ao *Prof. Afonso O. Alonso* pelo trabalho dedicado à orientação auxiliar.

Resumo

Neste trabalho, alguns aspectos sobre a codificação exata (estatística) de componentes de imagens coloridas de TV para transmissão através da linha telefônica convencional são abordados. Na fase de desenvolvimento deste serviço de comunicações visuais estáticas usou-se um microcomputador tipo PC-XT, com a finalidade de fornecer subsídios para viabilizar o seu emprego futuro como parte central na operação de um tal sistema.

No capítulo I é feita uma exposição da motivação, bem como da estruturação do trabalho realizado.

O capítulo II apresenta uma recordação de fundamentos teóricos necessários à sua elaboração, apresentando conceitos básicos de entropia, transformação digital de imagem (Hadamard - TDH e Cosseno - TDC), codificação híbrida Transformada + MCP Diferencial e código compacto de Huffman.

O capítulo III apresenta a implementação prática do trabalho na estrutura baseada no micro PC-XT. Os programas (BASIC e Assembler) para a obtenção de entropias bem como dos histogramas no *plotter* são discutidos. Problemas de transbordamento, arredondamento e truncamento de registros são também enfocados, quando da realização de algoritmos rápidos das transformadas pela UCP-8088 do PC.

O capítulo IV analisa e comenta os resultados obtidos para as imagens originais, bem como para aquelas transformadas (TRH e TRC) em híbrido com o MCP Diferencial.

No capítulo V as conclusões e comentários finais são realizados.

Índice

Capítulo I
Introdução Geral

Capítulo II
A Parte Teórica

Capítulo III
Implementação

Capítulo IV
Resultados Obtidos

Capítulo V
Comentários e Conclusões Finais

Capítulo I
Introdução Geral

I.1 - Introdução Geral

Com o avanço tecnológico da comunicação moderna a demanda de transmissão e armazenamento de imagem digitalizada está crescendo rapidamente. Este crescimento requer, naturalmente, eficiência e rapidez nos algoritmos usados para processar, transmitir e/ou armazenar grande quantidade de dados, aproveitando as facilidades atuais dos computadores digitais. A compressão de imagem, ou seja, a redução da quantidade de informação necessária para transmitir ou armazenar uma imagem começa, na verdade, na digitalização da mesma. Assim, o sinal analógico que é composto de um número infinito de pontos transforma-se numa sequência com um número finito e reduzido de pontos ou amostras na saída do digitalizador ou conversor analógico/digital (A/D).

O número de amostras, é claro, depende da finalidade do processo bem como da qualidade desejada da imagem reconstruída. As linhas da imagem digitalizada contém normalmente 2^m pontos, onde m é um número inteiro (7, 8, 9, 10, etc). Uma imagem típica de televisão digitalizada tem uma resolução espacial de 512 x 512 pontos que são chamados elementos de imagem (EI's). Quantizados por 8 *bits*/EI, e com 30 quadros por segundo, a taxa de transmissão necessária é de aproximadamente r = 63 *Mbits*/s. Tal taxa é muito grande sabendo-se que a taxa comercial no sistema PAL-M[1] a ser utilizada no Brasil deverá ser da ordem de 34 *Mbits*/s. Este fato é uma desvantagem da imagem digital em comparação com a imagem analógica.

Por razões claras, existe agora uma grande quantidade de trabalho visando reduzir (comprimir) a quantidade de informação necessária para transmitir e/ou armazenar a imagem digitalizada. Esta redução não é arbitrária nem ilimitada; é uma função da estatística da imagem, da propriedade da visão humana, dos métodos usados, bem como da quantidade tolerável de degradação [1]. Sendo assim, é necessário estabelecer uma medida concreta a fim de comparar o ganho obtido em tal redução contra o preço pago em termos de processamento e de uma eventual degradação da imagem.

Neste ponto da discussão, é importante verificar se é necessário degradar a imagem para reduzir a taxa de transmissão ou a capacidade de memória de armazenamento. Esta questão está intimamente ligada à finalidade do serviço de imagens desejado e, portanto, à opção de fazer codificação exata ou aproximada, como será visto na próxima seção.

I.2 - Codificação Exata

Como já frisado na seção anterior, o ganho na codificação digital da imagem implica necessariamente na redução da taxa de *bits* R em *bits*/EI ou em *bits*/símbolo, seja para transmissão ou para armazenamento. Isto explicitamente, significa redução do tempo de transmissão ou do espaço de armazenamento. Quando se procura uma taxa R' < R (*bits*/EI), sem recorrer à degradação da imagem, a codificação exata surge como opção real [2]. Como sugerido nesta referência, a recuperação da imagem do banco de memória (ou do receptor) deve ser efetuada sem degradações acumulativas. Deste modo, a menos dos inevitáveis erros de transmissão ou transferência, a imagem recuperada é igual à transmitida ou armazenada. Neste ponto da discussão, é importante frisar que o olho humano é crítico em relação às imagens paradas (*still-picture*).

1 Phase Alternating Line, padrão M.

A colocação da codificação exata, ao invés da aproximada, para o caso em questão pode parecer um pouco acadêmica, um pouco teórica demais. Entretanto, tal não ocorre. A codificação do *fac-simile* de documento – Grupo 3 do CCITT[1] – é um exemplo admirável (e prático) do emprego da codificação exata. Nesta, os vários comprimentos de correntes de zeros e uns (brancos e pretos) têm palavras-código cujos comprimentos respeitam suas probabilidades de ocorrência. Quanto mais frequentes, mais curtos [3].

Este trabalho é dedicado à codificação exata da imagem digitalizada. O objetivo é preparar então a imagem comprimida para transmissão ou armazenamento. Em termos de transmissão, busca-se uma taxa R' menor do que a taxa R (*bits*/EI) da imagem original, sem qualquer erro de processamento, evitando assim as aproximações geralmente feitas na codificação não exata, especialmente para imagens dinâmicas em altas taxas. Sendo assim, o fator de mérito aqui utilizado é a entropia[2], uma vez que na codificação exata a relação sinal/ruído é infinita e o erro quadrático médio é zero, já que a degradação é nula.

I.3 - Linha de Trabalho

Ao longo deste trabalho são estudadas também duas transformadas: a **transformada discreta de Hadamard** (TDH) e a **transformada discreta do cosseno** (TDC). As duas transformadas são aplicadas separadamente em quatro imagens-padrão de testes do SMPTE (*Society of Motion Picture and Television Engineers*). O desempenho delas é avaliado e discutido. É estudada também a modulação por código de pulsos diferencial (MCPD), aproveitando ainda as vantagens da codificação híbrida (transformada + MCPD).

Além disso, a codificação exata utilizando o código compacto de Huffman é estudada, baseando-se em parâmetros estatísticos da imagem. A probabilidade de ocorrência P_i de cada i-ésima mensagem da fonte discreta é usada para associar as palavras-código [4]. Este ponto é melhor discutido no capítulo II deste trabalho. Também no capítulo II é realçado o fato de que o conceito da entropia H fornece um limitante inferior da taxa de *bits* R' necessária para a transmissão do sinal em questão. É interessante mencionar que, felizmente, o comprimento médio $\overline{L}$ das palavras do código compacto de Huffman é aproximadamente igual à entropia H, sendo apenas ligeiramente maior do que esta nos casos de maior interesse aqui estudados.

A figura I.1 mostra o processo de codificação pretendida.

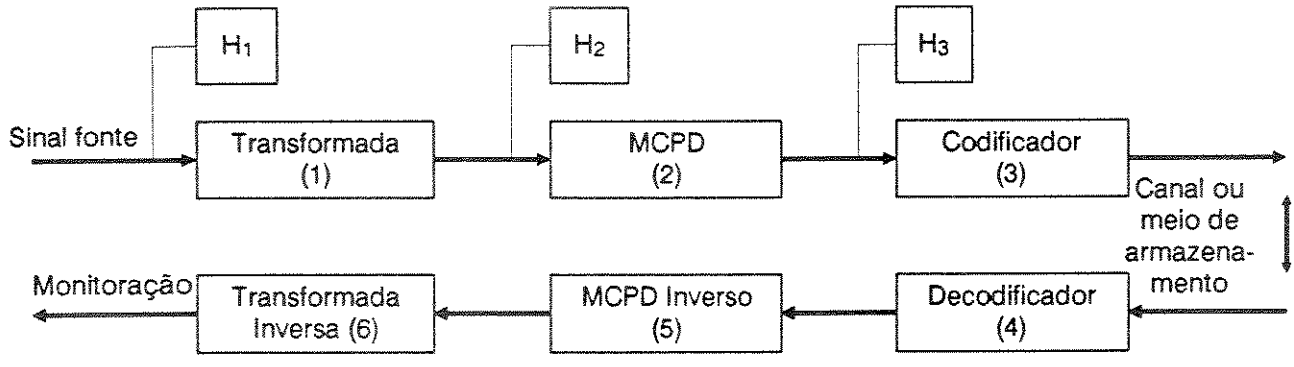


Figura I.1 - Sistema de codificação

1 Comitê Consultivo Internacional de Telegrafia e Telefonia

2 Parâmetro discutido no capítulo II deste trabalho.

Na figura I.1 o sinal fonte é um quadro de sinal de TV digitalizado e quantizado em 8 *bits*/amostra. Assim, como já mencionamos, tem-se um quadro de 512 x 512 EI's, onde cada elemento da imagem pode assumir qualquer valor de 0 a 255.

Os blocos denominados por H_i (i = 1, 2 e 3) significam apenas que a entropia do sinal naquele ponto é avaliada. Vários programas, no Assembler/PC, foram feitos para calcular a entropia do sinal nos estágios indicados. Tal avaliação é importante nos três estágios do processo de codificação a fim de comparar a eficiência de cada um. A ordem dos blocos 1 e 2 (ver figura I.1) pode também ser trocada, buscando um melhor desempenho global do sistema. No bloco 1, denominada Transformada, a imagem é dividida unidimensionalmente (2, 4, 8, 16... EI's), e posteriormente submetida aos programas que calculam os coeficientes da transformada escolhida.

Acompanhando as avaliações entrópicas, são também feitos vários histogramas das probabilidades de ocorrência dos símbolos originais e transformados, possibilitando assim visualizar os efeitos dos vários processamentos sobre as imagens.

Na etapa seguinte (bloco 2), o sistema MCPD pode efetuar as diferenças entre coeficientes correspondentes (atual menos o anterior), tanto na direção horizontal como na vertical. O sinal diferença ε resultante é então codificado de modo exato, ou seja, numa quantização 1:1. A entropia pode então ser calculada e comparada com aquelas obtidas nos outros estágios (ver figura I.1).

No bloco 3, denominado Codificador, o código compacto de Huffman é aplicado. A aplicação desse código, como é sabido, requer o cálculo das probabilidades de ocorrência dos elementos do sinal a ser codificado. isto fica por conta dos programas (micro PC/XT), que serão discutidos nos capítulos seguintes. O sinal recebido, evidentemente, sofrerá o processo inverso como mostrado na figura I.1. Um ponto a ser lembrado nessa altura é o fato de que o sinal codificado recebido deve voltar, após o processo de decodificação, ao mesmo sinal original que foi transmitido. Em outras palavras, os números inteiros que representam os EI's digitalizados reconstruidos na recepção têm que ser iguais aos EI's que foram transmitidos.

I.4 - Material de Trabalho

Antes de passar para o próximo capítulo, cabe introduzir brevemente as imagens digitalizadas que são utilizadas neste trabalho. Entre as quinze imagens padronizadas pelo SMPTE já citadas na seção anterior foram escolhidas quatro representativas, com graus diferentes de complexidade [5]. As quatro imagens originais usadas são mostradas nas figuras I.2 até I.5. As imagens já estão convertidas nos formatos de três componentes RGB (vermelho, verde e azul). O tamanho de cada arquivo (R, G ou B) é de 40000 *bytes* em hexadecimal (H), ou 262144 *bytes* em decimal (D). Cada *byte* representa um elemento de imagem, e pode variar de 0H até FFH, ou seja, de 0D até 255D. Assim, para cada uma das quatro imagens escolhidas, existem três arquivos originais de componentes **RGB**. Uma outra representação, aqui também utilizada, é o chamado arquivo de luminância (brilho) da imagem em questão. O arquivo de luminância Y é computado dos arquivos das componentes RGB pela equação:

$$Y = 0{,}299R + 0{,}587G + 0.114B \qquad (I.1)$$

Dos arquivos Y, R, G e B obtém-se então os arquivos chamados de diferença de cor **R-Y**, **G-Y** e **B-Y**, aqui também utilizados. O formato do chamado sinal de vídeo composto PAL-M [6] não é estudado neste trabalho. Assim sendo, para codificar uma imagem estática colorida deve-se fazer uso de, pelo menos, três arquivos componentes, tais como **Y**, **R-Y** e **B-Y**. Na recepção, as componentes **RGB** são normalmente recuperadas por filtros especiais, para propiciar a excitação separada dos três canhões do tubo receptor.

Resumindo, neste trabalho estão sendo utilizados os seguintes arquivos de cada imagem:

R: cor vermelha
G: cor verde
B: cor azul
Y: luminância
R-Y: vermelho menos luminância
G-Y: verde menos luminância
B-Y: azul menos luminância

Apresentadas as imagens, o próximo capítulo mostra os algoritmos teóricos aplicados em tais imagens.

Figura I.2 - Imagem SM01 (Praia)

Figura I.3 - Imagem SM02 (Sala)

figura I.4 - Imagem SM04 (Zelda)

figura I.5 - Imagem SM15 (Cozinha)

Capítulo II
A Parte Teórica

II.1 - Introdução/Motivação

Como já mencionado no capítulo anterior, a quantidade de informação original da imagem digitalizada deve ser reduzida. Desta maneira, o tempo de transmissão da imagem e/ou a capacidade do banco de memória ficam diminuidos. Algumas das muitas técnicas utilizadas para efetuar a chamada compressão de bits de imagens são revistas neste capítulo, do ponto de vista teórico. Os resultados experimentais e computacionais dessa técnicas são mostrados nos próximos capítulos.

Reduzir a taxa, o tempo de transmissão ou o espaço de armazenamento implica em aproveitar a redundância[1] existente entre os EI's na mesma linha, entre as linhas adjacentes ou até entre os quadros sucessivos. Na verdade, um dos alvos principais do processamento digital de imagens é a extração desta redundância, a fim de reduzir a quantidade de informação a ser transmitida ou armazenada. Em geral, uma cena de TV contém, em média, muita quantidade de informação que pode ser descartada sem perder a qualidade da cena. Este fato implica que uma técnica adequada que utiliza as estatísticas das imagens pode efetuar uma grande compressão sobre as mesmas. Na literatura de pesquisa do mundo inteiro, encontram-se afirmações neste sentido [2, 7, 8 e 9]. Entretanto, a grande maioria dos trabalhos está concentrada na redução de informação das imagens, às custas de uma degradação considerada tolerável. Este trabalho, como já foi mencionado anteriormente, procura investigar alguns aspectos de tal redução (compressão) mas, sem degradação. Isto, na verdade, é a chamada extração de redundância estatística [9]. Assim, o sinal recuperado do meio de armazenamento (memória semicondutora, fita, disco, etc) ou então da recepção deve voltar ao sinal original com 100% de precisão[2]. Além dessa vantagem óbvia, a codificação exata aqui estudada pode servir como medida comparativa para avaliar a eficiência das outras técnicas aproximadas, que se valem da chamada extração de redundância subjetiva, baseada no comportamento psico-físico da visão humana [9].

A codificação exata está sendo muito utilizada pelas conhecidas máquinas de fac-símile de documentos do Grupo-3 do CCITT que estão invadindo o mundo atual das telecomunicações [3]. Na verdade, este aspecto até motivou a pesquisa na direção da codificação exata.

As próximas seções deste capítulo apresentam uma revisão teórica que serve de base para o desenvolvimento do trabalho.

II.2 - A Entropia

No caso da codificação exata proposta neste trabalho, as medidas tradicionais como relação sinal/ruido, erro quadrático médio e qualidade da imagem recuperada têm os valores ∞, 0 e 100%, respectivamente, para qualquer imagem. Sendo assim, a eficiência da codificação é medida pela compressão (ou redução) do número de bits necessários para a transmissão, em comparação com o número de bits da imagem original. O número médio mínimo de bits necessários para transmitir a imagem digitalizada é dado pela sua entropia, ou quantidade de informação, H, como definido na teoria de informação [10 e 11].

1 A redundância nesse contexto é a repetição ou a semelhança de pontos próximos (no espaço ou no tempo) da imagem.

2 Supondo inexistência de erros no canal de transmissão ou no meio de armazenamento

A entropia é dada por:

$$H = -\sum_{i=1}^{N} P_i \log_2 P_i \quad [\text{bits/EI}] \qquad (II.1)$$

onde

H = entropia dada em bits/EI
P_i = probabilidade de ocorrência da mensagem i
N = o número de mensagens possíveis da fonte

No capítulo IV a entropia é calculada para algumas imagens de teste, antes e depois de se aplicar as várias técnicas de redução de redundância, para avaliar a eficiência das mesmas.

Uma observação que deve ser feita é que a expressão II.1 dá o número médio mínimo de [bits/EI], quando a possível dependência entre símbolos adjacentes é ignorada. É a chamada entropia da fonte de memória zero (sem memória). Como será visto nos próximos capítulos, cada símbolo da fonte da imagem original é um valor compreendido na faixa de 0 até 255, uma vez que o conversor A/D (analógico para digital) usado é de 8 bits.

Neste momento, é importante observar a figura II.1.

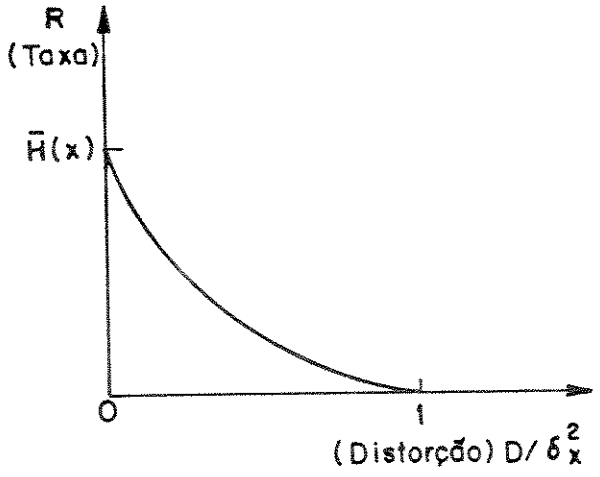


Fig. II.1 - Compromisso entre a taxa de informação R e a distorção para fontes de amplitude discreta.

O fato notável desta curva, como citado em [7], reside no fato de que há um compromisso na função R(D). Para R(0), ou seja, para distorção zero, há uma taxa finita para a fonte que produz amplitudes discretas. No caso contínuo, tal não ocorre; a taxa de transmissão de informação para distorção zero teria que ser infinita. Porém, no caso discreto aqui estudado a curva mostra que há um valor finito para R(D = 0), e ainda que tal valor é a entropia da informação $\bar{H}(x)$, ou seja, o conteúdo de informação média. Para uma fonte com N = 256 símbolos, a entropia da informação não pode ser maior do que $\log_2 N = \log_2 256 = 8$ bits/EI, que é o noso caso.

Na verdade, o valor máximo acima só é atingido quando os símbolos são equiprováveis, ou seja, $P_i = 1/2^8 = 1/256$, para $i = 0, 1, 2, ..., 255$. Entretanto, utilizando algumas técnicas de codificação pode-se explorar as dependências intersimbólicas, bem como memória de fonte (de símbolos), de tal sorte que a entropia seja menor do que o valor máximo dado por $\log_2 N$. Tal assunto é geralmente referido como codificação entrópica da fonte.

Como já foi frisado na seção II.1, o fac-símile do Grupo-3/CCITT é um caso admirável da utilização conjunta da codificação exata (D = 0) aliada à codificação entrópica. Em tal sistema, os comprimentos das correntes de "zeros" (brancos) e de "uns" (pretos) foram estatisticamente avaliados. Cada comprimento de corrente teve sua probabilidade de ocorrência P_i avaliada sobre uma série de 8 documentos de teste, e palavras-código de Huffman (ver seção II.5) foram então associadas [3]. Tal procedimento ficou conhecido como Código de Huffman Modificado (CHM), aplicado à codificação unidimensional.

O que se procura neste trabalho é tentar um paralelo ao que foi feito no serviço de fac-símile referido. A adaptação de um sistema de fac-símile preto e branco (2 tons) para o caso de imagens coloridas será melhor discutida mais adiante, neste trabalho.

Antes de finalizar esta seção, é importante colocar o conceito de redundância numa forma numericamente tratável. Neste trabalho, a redundância é obtida através da relação[1]:

$$Rd = 8 - H \quad [bist/EI] \qquad (II.2)$$

onde a entropia H é aquela obtida da equação II.1. Um outro conceito também utilizado ao longo deste trabalho é o que trata da eficiência de uma técnica de compressão. A eficiência de um certo sistema de codificação pode ser avaliada diretamente através do fator de compressão, aqui definido como sendo:

$$Fc = \frac{8}{H} \qquad (II.3)$$

onde a entropia H é dada pela equação II.1.

As relações II.1, II.2 e II.3 vistas nesta seção são empregadas exaustivamente ao longo deste trabalho, a fim de ponderar o poder de compressão das várias técnicas apresentadas.

Na próxima seção o conceito de transformada de imagens é revisto.

II.3 - Transformação de Imagem

Os elementos de imagens vizinhas no mosaico das amostras digitalizadas são naturalmente muito ricos de redundância. Assim, eles são estatisticamente dependentes. Um dos métodos de diminuir o grau de dependência entre os EI's consiste na aplicação de uma transformação que, em geral, resulta numa imagem representada pelos coeficientes de transformada menos correlacionados entre si do que os EI's originais.

1 O número 8 que aparece nas relações II.2 e II.3 é devido apenas ao fato de que a conversão A/D realizada é de 8 bits.

Algumas das transformadas utilizadas na codificação de imagens são, entre outras:

- Transformada Discreta de Fourier (TDF);
- Transformada de Karhunen-Loeve (TKL);
- Transformada Discreta do Cosseno (TDC);
- Transformada Discreta de Hadamard (TDH);
- Transformada Discreta de Slant (TDS);
- Transformada Discreta de Haar (TDHR).

Neste trabalho são utilizadas as transformadas de Hadamard e do Cosseno, que são revistas a seguir.

II.3.1 - A Transformada Discreta de Hadamard (TDH)

A transformação mais simples é, provavelmente, a de Hadamard. Ela tem sido muito referida e comparada com outras transformadas na literatura sobre o assunto [12, 13]. Em notação matricial, a matriz de Hadamard de ordem 2 pode ser vista na equação II.4:

$$H_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (II.4)$$

No caso específico da TDH um resultado recursivo conhecido [7] para a geração da transformada de ordem 2N, a partir da de ordem N, pode ser expresso pela relação II.5:

$$\vec{H}_{2N'} = \frac{1}{\sqrt{2N}}\begin{bmatrix} \vec{H}_{N'} & \vec{H}_{N'} \\ \vec{H}_{N'} & -\vec{H}_{N'} \end{bmatrix} \qquad (II.5)$$

Usando as relações II.4 e II.5, tem-se as seguintes matrizes de Hadamard:

$$H_4' = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \text{ ou } H_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \qquad (II.6)$$

$$H_8' = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \text{ ou } H_8 = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \qquad (II.7)$$

Para ordens maiores que 8 segue-se a mesma recursão.

As matrizes H_4' e H_8' são mais conhecidas na literatura por matrizes de Hadamard. Por outro lado, as matrizes denotadas por H_4 e H_8 são muitas vezes referidas como matrizes de Walsh-Hadamard. Estas últimas têm, como diferença básica, o fato de estarem numa forma mais ordenada. Em outras palavras, as linhas de H_4, por exemplo, estão em ordem crescente de sequência, ou frequência. A primeira linha de H_4 não tem nenhuma inversão de sinal; a segunda tem apenas uma; a terceira tem duas, enquanto a quarta (última) linha tem três alternâncias de sinal.

Em vista dos fatos acima citados, as transformadas de Walsh-Hadamard serão aqui referidas apenas por transformadas discretas de Hadamard.

II.3.1.a - TDH Unidimensional

Uma notação matricial sintética sobre a transformação pode ser vista nas relações abaixo:

$$\vec{\Theta} = \vec{H}\,\vec{x} \quad ; \quad \vec{x} = \vec{H}^{-1}\vec{\Theta} \tag{II.8}$$

onde $\vec{H}^{-1}$ é a matriz inversa de $\vec{H}$.

A matriz $\vec{x}$ representa o vetor-coluna de entrada, composta das amostras de uma sequência $\{x(i)\}$, com $i = 0, 1, 2, ..., N-1$, representativa de N elementos na horizontal ou vertical da imagem. Assim, o vetor $\vec{x}$ é representado por:

$$\vec{x} = \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \cdot \\ x(N-1) \end{bmatrix} \tag{II.9}$$

Como a matriz de II.9 tem dimensão (N x 1) e a matriz $\vec{H}$ tem dimensão (NxN), a matriz $\vec{\Theta}$ de II.8 tem dimensão (Nx1). Assim, Θ é também um vetor-coluna dado por:

$$\vec{\Theta} = \begin{bmatrix} \theta(0) \\ \theta(1) \\ \theta(2) \\ \cdot \\ \theta(N-1) \end{bmatrix} \tag{II.10}$$

onde $\{\theta(j)\}$ é a sequência dos coeficientes resultantes da transformação desejada, para $j = 0,1,2,...N-1$.

Uma ligeira reflexão mostra que, na verdade, as colunas da matriz $\vec{H}^{-1}$ (ou as linhas de $\vec{H}$) são os chamados vetores-base da transformação em questão. Assim, sejam os vetores base de $\vec{H}^{-1}$:

$$\vec{h_0}^{-1} = \begin{bmatrix} h_{0,0} \\ h_{0,1} \\ . \\ h_{0,N-1} \end{bmatrix} \quad \vec{h_1}^{-1} = \begin{bmatrix} h_{1,0} \\ h_{1,1} \\ . \\ h_{1,N-1} \end{bmatrix} \quad \ldots \quad \vec{h_{N-1}}^{1} = \begin{bmatrix} h_{N-1,0} \\ h_{N-1,1} \\ . \\ h_{N-1,N-1} \end{bmatrix}$$

Desta forma, a transformada inversa $\vec{x} = \vec{H}^{-1}\vec{\Theta}$ fornece um conjunto de relações que expressam combinações lineares. Em outras palavras, a amostra x(0), por exemplo, é uma combinação linear na forma:

$$x(0) = h_{0,0}.\theta(0) + h_{1,0}.\theta(1) + \ldots + h_{N-1,0}.\theta(N-1)$$

De modo matricial, mais sintético, tem-se:

$$\vec{x} = \theta(0).\vec{h_0}^{-1} + \theta(1).\vec{h_1}^{-1} + \ldots + \theta(N-1).\vec{h_{N-1}}^{1} \qquad (II.11)$$

onde $\vec{h_j}^{-1}$, com $j = 0,1,2,\ldots N\text{-}1$, são os vetores-base da transformação. Tal relação estabelece o fato de que o vetor coluna $\vec{x}$ da sequência de entrada é uma soma ponderada dos vetores-base, com pesos dados pelos coeficientes do vetor coluna $\vec{\Theta}$ da sequência transformada.

Das relações II.8 e II.11 pode-se fazer a generalização:

$$\vec{x} = \vec{H}^{-1}\vec{\Theta} = \sum_{j=0}^{n-1} \theta(j)\,\vec{h_j}^{-1} \qquad (II.12)$$

Uma observação às matrizes reais H_2, H_4 e H_8 (ver relações II.4, II.6 e II.7) mostra que é válida a relação:

$$\vec{H}^{-1} = \vec{H}^{T} \qquad (II.13)$$

que implica em

$$\vec{H}^{T}\vec{H} = \vec{H}\vec{H}^{T} = \vec{I} \qquad (II.14)$$

onde I é a matriz identidade de ordem N.

Na verdade, as relações II.13 e II.14 definem a ortogonalidade da matriz real $\vec{H}$ de ordem N = 2, 4, 8, etc. Em outras palavras, os vetores-base são as linhas (ou as colunas) de $\vec{H}_N$, e vale a relação do produto escalar:

$$\vec{h_i}^{T} \cdot \vec{h_j} = \vec{\delta}_{i,j} \qquad (II.15)$$

onde a função delta de Kroenecker é tal que:

$$\delta_{i,j} = \begin{cases} 0 \text{ se } i \neq j \\ 1 \text{ se } i = j \end{cases} \qquad (II.16)$$

Os vetores-base da TDH para ordem N = 8 podem ser vistos na figura II.2

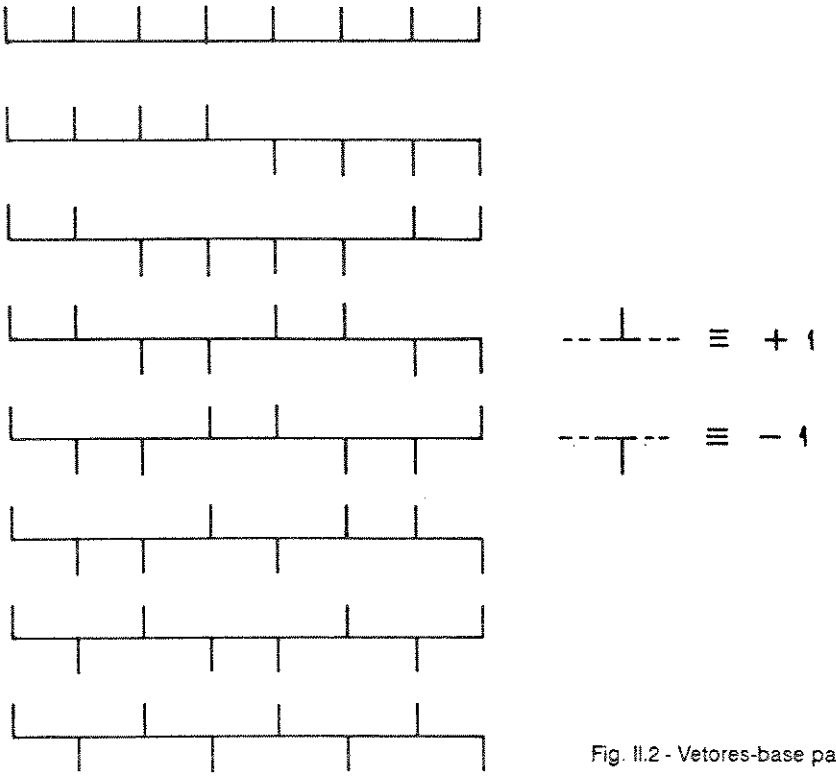


Fig. II.2 - Vetores-base para TDH de ordem N = 8

A transformada discreta de Hadamard não ordenada (natural) pode ser expressa de forma algébrica sintética pela relação:

$$\theta(j) = \sum_{i=0}^{N-1} x(i)\, h'(j,i) \quad \text{para } j=0,1,...,N-1 \qquad (II.17)$$

onde h'(j,i) é o chamado núcleo ("kernel") de transformação direta e obedece à relação:

$$h'(j,i) = \frac{1}{\sqrt{N}}(-1)^{p(j,i)} \qquad (II.18)$$

com

$$p(j,i) \equiv \sum_{l=0}^{n-1} j_l . i_l \qquad (II.19)$$

onde os termos j_l e i_l são as representações binárias de j e i, respectivamente. Por exemplo:

$$(j)_{\text{decimal}} = (j_{r-1}\, j_{r-2} \ldots j_1\, j_0)_{\text{binário}} \qquad (II.20)$$

com $j,i \in \{0,1\}$ e a somatória de II.19 sendo feita módulo de dois.

A transformação inversa daquela expressa em II.17 pode ser obtida da relação:

$$x(i) = \sum_{j=0}^{N-1} \theta(j)\, h'(j,i) \qquad \text{para } i=0,1,2,...N-1 \tag{II.21}$$

onde o núcleo inverso h'(j,i) é o mesmo dados pelas expressões II.18, II.19 e II.20.

Por outro lado, se a transformada discreta de Hadamard na forma ordenada for desejada (ver II.4, II.6 e II.7 para $\vec{H}_2$, $\vec{H}_4$, $\vec{H}_8$) as expressões algébricas para as transformações direta e inversão são aquelas já expressas nas relações II.17 e II.21, mas o núcleo h'(j,i) é agora substituído pelo "kernel" h(j,i) dado por:

$$h(j,i) = \frac{1}{\sqrt{N}} (-1)^{q(j,i)} \tag{II.22}$$

com

$$q(j,i) = \sum_{l=0}^{n-1} i_l . r_l(j) \tag{II.23}$$

e

$$\begin{aligned} r_0(j) &= j_{n-1} \\ r_1(j) &= j_{n-1} \oplus j_{n-2} \\ r_2(j) &= j_{n-2} \oplus j_{n-3} \\ &\cdots \\ r_{n-1}(j) &= j_1 \oplus j_0 \end{aligned} \tag{II.24}$$

onde $\oplus$ significa adição módulo de dois, e a somatória da expressão II.23 também é feita módulo de dois.

II.3.1.b - TDH Bidimensional

A transformada discreta de Hadamard pode ser estendida para o caso bidimensional. Desta forma, um "quadrado" (NxN) de elementos da imagem x(g,i) pode ser transformado num novo "quadrado" de elementos transformados ou coeficientes θ(j,k), também de dimensão NxN.

As expressões II.17 e II.21 acima podem então ser generalizadas para o caso bidimensional nas formas algébricas:

$$\theta(j,k) = \sum_{g=0}^{N-1} \sum_{i=0}^{N-1} x(g,i).h(j,k,g,i) \quad \text{para } j,k=0,1,...N-1 \tag{II.25}$$

$$x(g,i) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \theta(j,k).h(j,k,g,i) \quad \text{para } g,i=0,1,...,N-1 \qquad (II.26)$$

Sabendo de antemão que há vários algoritmos de transformações unidimensionais rápidos na literatura sobre o assunto, é interessante aproveitá-los neste trabalho. Isto será feito no capítulo III. Para o momento, basta lembrar o fato de que na relação II.25, por exemplo, o núcleo da transformação pode ser reescrito felizmente como:

$$h(j,k,g,i) = h_v(j,g).h_h(k,i) \qquad (II.27)$$

A propriedade expressa em II.27 é conhecida como a separabilidade do núcleo bidimensional da transformada em questão.

Sendo assim, a relação II.25 pode ser realizada em dois passos, ou seja:

$$\theta(j,k) = \sum_{g=0}^{N-1} \sum_{i=0}^{N-1} x(g,i)h_v(j,g).h_h(k,i)$$

Rearranjando, obtém-se:

$$\theta(j,k) = \sum_{g=0}^{N-1} h_v(j,g) \sum_{i=0}^{N-1} x(g,i)h_h(k,i) = \sum_{g=0}^{N-1} h_v(j,g)\theta(g,k) \qquad (II.28)$$

De II.28 nota-se então que, primeiramente, os elementos da imagem original na linha genérica g são unidimensionalmente transformados pelo "kernel" horizontal, obtendo-se $\theta(g,k)$. Posteriormente, o núcleo h_v é usado na direção vertical, transformando unidimensionalmente os elementos da coluna k.

A expressão II.28 acima pode ser posta numa forma matricial mais concisa, ou seja:

$$\vec{\Theta} = \vec{H}_V \vec{X} \vec{H}_H^T \qquad (II.29)$$

onde agora, no caso bidimensional, $\vec{X}$ e $\vec{\Theta}$ são matrizes, ou seja, arranjos de EI's e de coeficientes dispostos num quadrado com elementos x(g,i) e $\theta(j,k)$, respectivamente.

As matrizes de transformação Hadamard $\vec{H}_h$ e $\vec{H}_v$ unidimensionais são compostas dos elementos $h_h(r,s)$ e $h_v(r,s)$, ou seja:

$$\vec{H}_H = \left\{h_h(r,s)\right\}_{r,s=0,1,...,N-1}$$

$$\vec{H}_V = \left\{h_v(r,s)\right\}_{r,s=0,1,...,N-1} \qquad (II.30)$$

Para os casos dos núcleos simétricos da transformada de Hadamard unidimensional (ver seção anterior) tem-se:

$$\vec{H}_H = \vec{H}_V = \vec{H} \qquad (II.31)$$

Assim, tem-se:

$$\vec{\Theta} = \vec{H}\,\vec{X}\,\vec{H}^T \qquad (II.32)$$

Como, neste caso $H^{-1} = H^T$, obtém-se a transformação inversa:

$$\vec{X} = \vec{H}^T\,\vec{\Theta}\,\vec{H} \qquad (II.33)$$

Neste ponto é útil relembrar o que expressa a relação II.12. Nela, observou-se que o vetor coluna $\vec{x}$ é uma combinação linear das colunas da matriz $\vec{H}^{-1}$ (linhas de $\vec{H}$); é uma soma ponderada dos vetores-base multiplicados pelos coeficientes da transformada. Nesta linha de raciocínio, pode-se exprimir as relações II.33 e II.26 da seguinte forma:

$$\vec{X} = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} \theta(j,k)\vec{H}_{jk} \quad ; \quad \vec{H}_{jk} = \vec{h}_j\,\vec{h}_k^T \qquad (II.34)$$

Para o caso da matriz de Hadamard mostrada na relação II.4 deste capítulo, por exemplo, obtém-se para $\vec{H}_{jk}$ de II.34 as chamadas imagens-base, em contraposição aos chamados vetores-base da transformação unidimensional:

$$\vec{H}_{00} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \vec{H}_{01} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

$$\vec{H}_{10} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \qquad \vec{H}_{11} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad (II.35)$$

Neste caso, a relação II.34 expressa a matriz (ou arranjo) dos EI's originais $\vec{x}$ como uma combinação linear das imagens base H_{jk} dadas em II.35; uma soma ponderada onde os pesos são dados pelos coeficientes $\theta(j,k)$ da transformada.

Note-se ainda que cada imagem-base lembra um tabuleiro de xadrez, onde 1 representa o branco e -1 o preto. Apesar deste fato, imagens multi-tonais (com gradação de cinza) são comumente representáveis na TDH, uma vez que os coeficientes $\theta(j,k)$ de II.34 geralmente não são apenas binários.

II.3.2 - Transformada Discreta do Cosseno (TDC)

A transformada discreta do cosseno é relativamente nova (1974). Em contraposição à TDH que decompõe os EI's em ondas quadradas amostradas, a TDC usa ondas componentes cossenoidais amostradas de certas frequências e fases [2, 7, 8, 9 e 14].

Os conceitos já frisados na seção II.3.1 geralmente são válidos no caso da TDC. O que muda realmente é a transformação.

Assim, pode-se definir a TDC para os casos de maior interesse: unidimensional e bidimensional.

II.3.2.a - TDC Unidimensional

Na forma matricial, pode-se escrever:

$$\vec{\Theta} = \vec{C}\vec{x} \quad ; \quad \vec{x} = \vec{C}^{-1}\vec{\Theta} \tag{II.36}$$

onde $\vec{C}^{-1}$ é a inversa da matriz da transformação do cosseno $\vec{C}$, $\vec{x}$ e $\vec{\Theta}$ são os vetores-colunas já definidos nas expressões II.9 e II.10.

A matriz $\vec{C}$ é ortogonal, valendo então as relações:

$$\vec{C}^{-1} = \vec{C}^{T} \tag{II.37}$$

$$\vec{C}^{T}\vec{C} = \vec{C}\vec{C}^{T} = \vec{I} \tag{II.38}$$

onde $\vec{I}$ é a matriz identidade de ordem N. São válidas também as relações de produto escalar:

$$\vec{c_i}^{T} \cdot \vec{c_j} = \delta_{ij} \qquad \text{(delta de Kroenecker)} \tag{II.39}$$

com

$$\delta_{ij} = \begin{cases} 0 \text{ se } i \neq j \\ 1 \text{ se } i = j \end{cases}$$

As formas algébricas que sintetizam a TDC direta, bem como a inversa, são dadas respectivamente por:

$$\theta(j) = \sqrt{\frac{2}{N}}\,\alpha(j) \sum_{i=0}^{N-1} x(i) \cos\frac{(2i+1)j\pi}{2N} \tag{II.40}$$

$$x(i) = \sqrt{\frac{2}{N}} \sum_{j=0}^{N-1} \alpha(j)\,\theta(j) \cos\frac{(2i+1)j\pi}{2N} \tag{II.41}$$

onde i e j = 0,1,2,...,N-1 e

$$\alpha(0) = \frac{1}{\sqrt{2}} \quad ; \quad \alpha(j) = 1 \quad \text{para } j \neq 0$$

A título de ilustração, as matrizes de ordem N = 2, 4 e 8 são dadas a seguir:

$$\vec{C}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (II.42)$$

$$\vec{C}_4 = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ a & b & -b & -a \\ 1/2 & -1/2 & -1/2 & 1/2 \\ b & -a & a & -b \end{bmatrix} \qquad (II.43)$$

onde a = 0,653281 e b = 0,270598

$$\vec{C}_8 = \begin{bmatrix} \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ a & b & c & d & -d & -c & -b & -a \\ e & f & -f & -e & -e & -f & f & e \\ b & -d & -a & -c & c & a & d & -b \\ \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ c & -a & d & b & -b & -d & a & -c \\ f & -e & e & -f & -f & e & -e & f \\ d & -c & b & -a & a & -b & c & -d \end{bmatrix} \qquad (II.44)$$

onde

a = 0,490393
b = 0,415735
c = 0,277785
d = 0,097545
e = 0,461940
f = 0,191342

A figura II.3 ilustra os vetores-base para TDC de ordem N = 8:

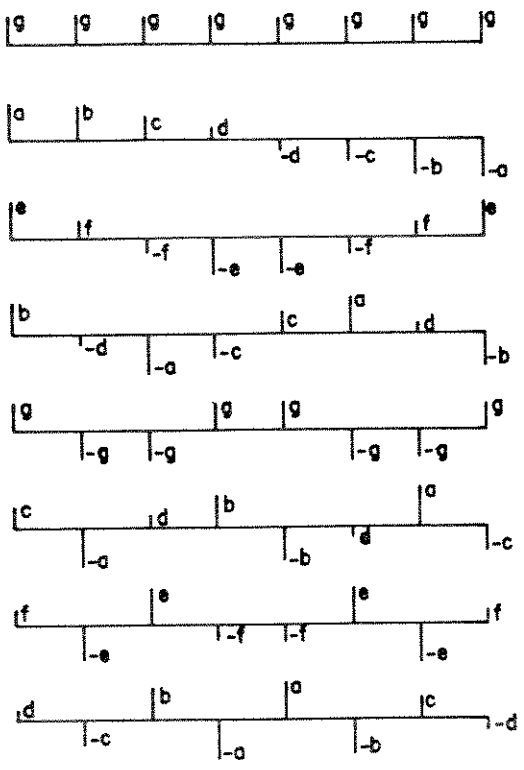


Fig. II.3 - Vetores-base para TDC de ordem N = 8

As equações II.40 e II.41 podem ser escritas também no formato de "núcleo", como feito anteriormente para o caso de TDH. Assim, tem-se:

$$\theta(j) = \sum_{i=0}^{N-1} x(j).c(j,i) \quad \text{para } j = 0,1,2,...,N-1 \qquad (II.45)$$

$$x(i) = \sum_{j=0}^{N-1} \theta(j).c(j,i) \quad \text{para } i = 0,1,2,...,N-1 \qquad (II.46)$$

onde o núcleo da transformação discreta de cosseno c(j,i) é dado por:

$$c(j,i) = \sqrt{\frac{2}{N}}\ \alpha(j) \cos\frac{(2i+1)j\pi}{2N} \qquad (II.47)$$

com

$$\alpha(0) = \frac{1}{\sqrt{2}} \quad \text{e} \quad \alpha(j) = 1 \quad \text{para } j \neq 0 \qquad (II.48)$$

II.3.2.b TDC Bidimensional

Felizmente, valem também para a TDC as mesmas considerações já feitas na seção II.3.1.b para a TDH. Neste caso, a TDC direta e a inversa são dadas respectivamente por:

$$\theta(j,k) = \sum_{g=0}^{N-1} \sum_{i=0}^{N-1} x(g,i).c(j,k,g,i) \quad \text{para } j,k = 0,1,2,...,N-1 \qquad (II.49)$$

$$x(g,i) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \theta(j,k).c(j,k,g,i) \quad \text{para } g,i = 0,1,2,...,N-1 \qquad (II.50)$$

A separabilidade do núcleo bidimensional existe, de tal sorte que:

$$c(j,k,g,i) = c_v(j,g).c_h(k,i) \qquad (II.51)$$

Desta forma, visando a relação II.47, obtém-se:

$$\theta(j,k) = \frac{2}{N} \alpha(j)\,\alpha(k) \sum_{g=0}^{N-1} \sum_{i=0}^{N-1} x(g,i).\cos\frac{(2g+1)j\pi}{2N} . \cos\frac{(2i+1)k\pi}{2N} \qquad (II.52)$$

$$x(g,i) = \frac{2}{N} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \alpha(j)\,\alpha(k)\,\theta(j,k).\cos\frac{(2g+1)j\pi}{2N} . \cos\frac{(2i+1)k\pi}{2N} \qquad (II.53)$$

onde x(g,i) geralmente é um arranjo "quadrado" de EI's, de dimensão NxN. Também as igualdades expressas em II.48 são válidas para os $\alpha(j)$ e $\alpha(k)$ das relações II.52 e II.53.

Talvez a conclusão mais importante contida nas relações obtidas acima reside no fato de que a TDC bidimensional pode também ser obtida pela aplicação sucessiva de duas transformações unidimensionais. Sendo assim, os algoritmos rápidos unidimensionais que serão aplicados no capítulo seguinte podem ser usados em dois passos sucessivos, para a obtenção rápida da TDC bidimensional.

Para finalizar esta seção, as formas matriciais da TDC bidimensional podem, se desejado, ser colocadas na forma:

$$\vec{\Theta} = \vec{C}\,\vec{X}\,\vec{C}^T \qquad (II.54)$$

$$\vec{X} = \vec{C}^T\,\vec{\Theta}\,\vec{C} \qquad (II.55)$$

onde $\vec{X}$ representa uma matriz quadrada N x N formada pelos EI's x(g,i); $\vec{\Theta}$, por sua vez, é a matriz dos elementos transformados (ou coeficientes) $\theta(j,k)$.

II.4 - Codificação Híbrida

As transformadas TDH e TDC mencionadas nas seções anteriores são aplicadas unidimensionalmente, tanto no sentido horizontal (nas linhas) como na vertical (nas colunas). O descorrelacionamento dos coeficientes proporcionado pelas transformadas depende, simultaneamente, do tamanho do bloco e da quantidade de detalhes da cena. Assim sendo, se a cena tem poucos detalhes, o tamanho do bloco que pode ser usado é maior. Naturalmente, qualquer que seja o tamanho do bloco, a semelhança entre os coeficientes correspondentes depende do grau de variação dos EI's dentro de cada bloco original. Nas cenas planas, por exemplo, os coeficientes correspodentes nos blocos adjacentes são muito parecidos. Em geral, os coeficientes vizinhos no sentido perpendicular àquele em que foi aplicada a transformada unidimensional não foram atingidos pelo descorrelacionamento. A mesma lógica é válida no caso da transformada bidimensional, no que se refere aos coeficientes correspondentes de um bloco para o outro adjacente. Desta consideração, surge então a possibilidade do emprego da codificação híbrida [15, 16], que consiste em aplicar o MCPD no sentido que ainda não foi descorrelacionado. Tal procedimento proporciona então a compressão desejada na imagem. Nos trabalhos referidos a codificação preditiva baseada no MCP Diferencial é utilizada. A figura II.4 ilustra este sistema de codificação híbrida.

Os elementos da imagem digitalizada são primeiramente transformados (TDH ou TDC unidimensional) na direção horizontal, por exemplo. Os coeficientes resultantes θ_i são então aplicados ao sistema MCPD na direção vertical. Desta forma, o sinal diferença ε é dado pela relação:

$$\varepsilon = \theta_i - \hat{\theta_i} \qquad (II.56)$$

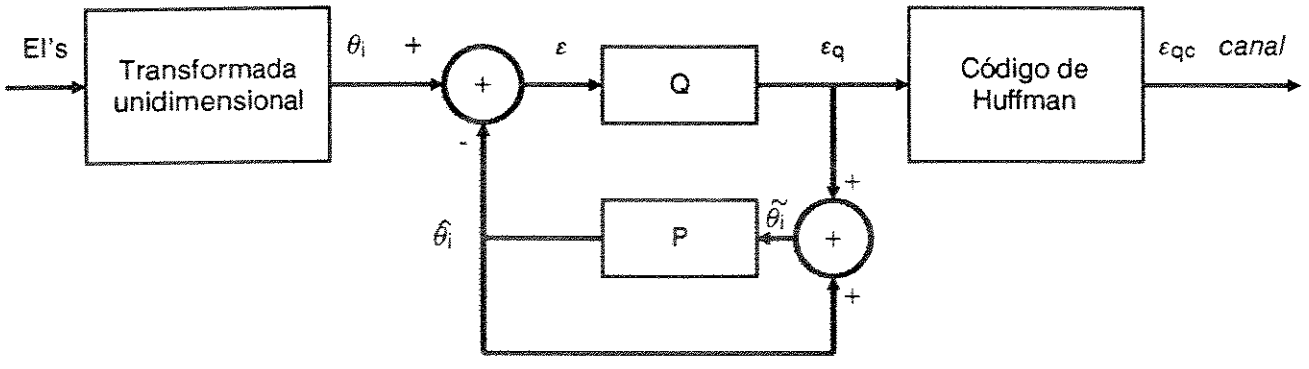


Fig. II.4 - Codificação híbrida Transformada/MCPD

O símbolo $\hat{\theta}$ é uma estimativa (predição) do i-ésimo coeficiente que vai ser codificado. O preditor (P) mais simples é dado por:

$$\hat{\theta} = \theta_{i-512} \qquad (II.57)$$

o que equivale à previsão da amostra prévia. Neste ponto, é bom lembrar o fato de que as imagens digitalizadas possuem 512 linhas, com 512 EI's/linha. Em outras palavras, θ_{i-512} na relação II.57 significa o coeficiente (i-512)correspondente na linha imediatamente anterior.

As direções de aplicação da transformada e da MCPD podem ser invertidos. Assim, pode-se transformar os EI's na direção vertical e codificar as diferenças dos coeficientes na direção horizontal. Tal procedimento exigiria uma quantidade maior de memória de atraso, numa eventual implementação prática.

O ponto mais importante a ser citado prende-se ao fato de que tal sistema tem sido investigado do ponto de vista da codificação não exata. Neste caso, o quantizador Q (ver figura II.4) não é 1:1. Além disso, as diferenças de alguns coeficientes podem ser descartadas, baseado em alguns critérios psico-físicos do olho humano. A conjunção destes dois fatores tem levado a uma redução da taxa de bits (ou compressão), às custas de uma degradação tolerável na imagem recuperada. Entretanto, o desempenho da codificação híbrida é investigado neste trabalho à luz da codificação exata, baseada apenas nas redundâncias naturalmente existentes nas imagens estudadas.

Finalizando esta seção, deve-se mencionar ainda que, na codificação exata, ε_q (ver figura II.4) significa uma sequência de diferenças inteiras, valendo a relação:

$$\varepsilon_q = \varepsilon \qquad (II.58)$$

ou seja, o quantizador Q empregado é realmente 1:1. Desta forma, a redundância ainda existente no sinal ε é extraída por meio de código compacto de Huffman, revisto na próxima seção. Tal código produz então as palavras-código da sequência ε_{qc} da figura II.4.

II.5 - Código Compacto de Huffman

Todos os processos desenvolvidos na seções anteriores desse capítulo almejam, de algum modo, a redução da taxa de bits para a transmissão de imagem. Nesse contexto, é anti-vantajoso utilizar qualquer código que não minimiza o comprimento das palavras-código $\bar{L}$ e, portanto, a taxa de transmissão R bits/EI. Felizmente, existe um código que usa a estatística das mensagens (EI's), ou seja, as probabilidades de ocorrência P_i das mensagens m_i da fonte discreta. O chamado código compacto de Huffman usa um comprimento médio das palavras-código $\bar{L}$ aproximadamente igual à entropia da mensagem H(m). Da teoria de codificação, para um canal não ruidoso, não existe qualquer outro código unicamente decodificável de comprimento médio menor que H(m) [4,17].

O código compacto de Huffman é construido da seguinte maneira:

Dadas as mensagens (EI's) m_i de uma fonte discreta de N símbolos, com probabilidades de ocorrência P_i ($i = 1,2,...,N$), os EI's são reordenados de acordo com a ordem decrescente de P_i (ver tabela II.1). Em seguida, as duas probabilidades menores são somadas e novo reordenamento decrescente é feito de acordo com as novas probabilidades, da fonte reduzida F_1. A soma das duas últimas P_i's e a reordenação são repetidas até sobrar apenas duas mensagens (ver F_2 na tabela II.1), às quais são dadas as palavras-código "0" e "1".

A partir daí, a associação das palavras-código é feita voltando para trás, dando novamente "0" e "1" para o segundo dígito das duas mensagens que foram combinadas no estágio anterior. O processo continua até associar todas as palavras do código compacto.

El's	P_I	Código	F_1		F_2	
m_1	0,45	1	0,45	1	0,55	0
m_2	0,22	01	0,33	00	0,45	1
m_3	0,20	000	0,22	01		
m_4	0,13	001				

Tabela II.1 - Código compacto de Huffman

O comprimento médio $\bar{L}$ do código compacto de Huffman é dado, neste exemplo, por:

$$\bar{L} = \sum_{i=1}^{n} P_i L_i = 0{,}45\times1+0{,}22\times2+0{,}20\times3+0{,}13\times3$$

$$\bar{L} = 1{,}88 \quad \text{bits/EI}$$

Por outro lado, a entropia teórica H(m) para o caso dos quatro elementos de imagem no exemplo dado pode ser então calculada, ou seja:

$$H(m) = \sum_{i=1}^{n} P_i \log_2 \frac{1}{P_i}$$

$$H(m) = 0{,}45\times1{,}15+0{,}22\times2{,}18+0{,}20\times2{,}32+0{,}13\times2{,}94$$

$$H(m) = 1{,}84 \quad \text{bits/EI}$$

A eficiência η do código compacto pode ser definida pela relação:

$$\eta = \frac{H(m)}{\bar{L}}$$

$$\eta = \frac{1{,}84}{1{,}88} = 0{,}978$$

Em outras palavras, a codificação compacta obtida na tabela II.1 permite operar 97,8% do código teórico ótimo.

Para concluir, é importante frisar que o código compacto de Huffman revisto nesta seção é o complemento ideal para a codificação entrópica investigada ao longo deste trabalho.

No próximo capítulo serão mostrados os aspectos experimentais da implementação e desenvolvimentos práticos das técnicas de codificação exata até então mencionadas.

Capítulo III
Implementação

III.1 - Introdução / Facilidades Disponíveis

É preciso dizer, primeiramente, que não foram usados pacotes de *software* nem qualquer outro programa pré-elaborado para realizar as implementações deste trabalho. Uma das razões desta decisão foi aquela de tentar dominar as técnicas de processamento de imagens usado a linguagem Assembler/PC, misturando-as com a linguagem de alto nível BASIC/PC, razoavelmente familiares no início do trabalho. A primeira linguagem tem o mérito da rapidez e eficiência para as contas, enquanto o BASIC possui facilidade e clareza para tratamento de tela, por exemplo.

Outra razão para desenvolver programas próprios é aprender a administração de detalhes que só a linguagem de máquina oferece. Problemas como *overflow* e *underflow* de capacidade de registradores de 8, 16 e 32 bits, também a necessidade de ponto flutuante, real de precisão simples/dupla (4 ou 8 bytes), são resolvidos com muito mais profundidade do que quando se usa pacotes que dão resultados prontos, escondendo, às vezes, detalhes importantes. Realização deste tipo passa, necessariamente, pelo domínio geral sobre o micro PC e seu sistema operacional MS-DOS ou PC-DOS, numa máquina I-7000 PC/XT da Itautec, por exemplo.

Ainda outra razão que influiu na decisão de usar programação própria o fato de que a combinação de BASIC-PC interpretado (não compilado) com o Assembler/Debug [1], ao invés do ASM [2] ou MASM [3], possibilita a rapidez necessária para mudar ou corrigir programas. Assim sendo, não é necessário passar por editores de texto, criar programas fonte, criar módulos executáveis, listáveis usado link-editores, compiladores, etc.

A máquina acima citada dispõe de 640 kbytes de memória dinâmica (DRAM), dois acionadores de disquete, com 360 kbytes de capacidade cada, unidade de processamento central UPC-8088 operando tanto em 4,77 como em 8 MHz e co-processador aritmético de dados numéricos PDN-8087. O micro possui também um cartão GPIB (*General Pourpose Interface Bus*), também chamado HP-IB (IEEE-488), para interface paralela de conversação entre instrumentos.

Para algumas situações, como a obtenção de histogramas das imagens, é necessário utilizar o plotter HP-9862A da calculadora de mesa HP-9820A, em conjunto com o PC mencionado.

Nas próximas duas seções deste capítulo são discutidos os programas mistos implementados para a obtenção das entropias e dos histogramas das imagens originais e processadas.

---

1 Montador/Depurador.

2 Assembler = (montador).

3 Macro-Assembler.

III.2 - Entropia

A entropia H, como definida na seção II.2 é dada pela relação II.1 do capítulo anterior, é uma medida do número mínimo de bits necessários para se transmitir os dados em questão. Por isso, ela dá uma idéia geral da complexidade ou da riqueza entrópica de uma dada imagem.

Para calcular a entropia das 4 imagens originais e/ou processadas utilizando as facilidades mencionadas na seção anterior, um programa misto foi elaborado. O pano de frente é um programa em linguagem BASIC-PC como mostrado em Prog. 1A que, por sua vez, chama o Prog. 1B, feito em linguagem de máquina do Assembler-PC, via depurador/montador Debug.

O Prog. 1A faz basicamente o tratamento de tela do PC. O Cálculo da entropia baseado na equação II.1 (capítulo II) é efetivamente realizado pelo programa 1B, que é carregado na memória do PC nas linhas 60 e 70 do programa-mãe em BASIC. Tal carregamento é feito numa região alta da memória para não haver superposição entre os dois programas e os dados da imagem.

Neste ponto, é importante mencionar que o MS-DOS do PC utiliza a notação "*segmento:offset*" para indicar o endereço de memória. Desta forma, a linha 60 do programa 1A define que a carga do programa 1B (em Assembler) será feita no segmento 7000: (hexadecimal), fora da área de trabalho do BASIC. Na linha 70, o carga binária (BLOAD) é feita no *offset* zero.

Uma ligeira observação no programa ENTROP.BIN (ver programa 1B) mostra que são usados menos de 100H posições de memória, a partir do endereço 7000:0000. O programa em linguagem de máquina vai até 7000:00F0 (hexadecimal), ocupando, portanto, 15 x 16 + 0 x 0 = 240 bytes, em notação decimal.

Na linha 100 do ENTROP.BAS define-se a variável A com o valor de 262.144, pelo fato de que este é o número de elementos de imagem de cada arquivo utilizado (512 x 512).

Quando o programa 1A é rodado, a linha 80 define que a variável SUBROT é igual a zero. Sendo assim, quando o PC executa a linha 120, o contador de instrução ou de programa IP da UCP-8088 pula para executar as intruções depositadas a partir de 7000:0000.

Em outras palavras, o micro passa a executar o programa 1B em linguagem de máquina. O detalhe importante a ser lembrado é o fato de que, na linha 120, a instrução CALL SUBROT(A) passa uma variável real, de precisão simples, para que seja aproveitada pelo programa 1B. Tal *passagem* é, na verdade, feita pela pilha do PC, via registro SP (*stack pointer*) que, por sua vez, transfere para o registro ponteiro de base BP (*base pointer*) [20, 21]. Tal fato pode ser observado nas primeiras instruções do programa 1B, ou seja,

```
JMP   0010
PUSH  BP
MOV   BP,SP
```

A primeira é apenas um salto (JUMP) para o endereço 7000:0010, a segunda salva (PUSH) o conteúdo do registrador BP na pilha de retorno, a terceira move (MOV) o conteúdo do registrador ponteiro de pilha (*stack pointer*) SP para o ponteiro de base (*base pointer*).

É importante frisar que a variável A real (262144) não é passada pela pilha (SP). O que se passa é apenas o endereço da variável A, para que ENTROP.BIN a encontre.

Em seguida, as instruções FINIT e FLDZ do programa 1B são executadas [22,23]. Elas significam *inicialização* (uma espécie de *reset* do PDN) e *carregue zero* no primeiro - ST (0) - dos 8 registros de armazenamento (*storage*) do PDN, respectivamente. As instruções do PDN tem sempre mnemônicos começando com a letra F, para lembrar *floating-point*. Deve-se observar ainda que estas duas instruções são precedidas pela instrução WAIT, ou seja, a UCP-8088 espera o PDN-8087 fazer as contas. Deste modo há sincronização no trabalho dos dois *chips*.

A instrução MOV SI,[BP + 06] do programa 1B traz o endereço da variável real A de precisão simples (4 bytes) para o registro SI. Note que o conteúdo de memória apontado por BP + 06 (registro da base com deslocamento + 06) aponta A. Desta maneira, SI (índice de fonte) servirá de ponteiro para trazer a variável A.

Neste ponto é importante lembrar que o deslocamento (*off-set*) de + 06 aponta o lugar correto de A. Na verdade [ref. 19], o deslocamento em relação a BP segue a fórmula abaixo:

$$\text{Deslocamento em relação a BP} = 2.(n-m) + 6 \qquad (III.1)$$

onde

n = número de variáveis a passar;
m = posição do argumento na lista (m = 1,2,...,n).

A referência acima citada dá um exemplo para o caso de variáveis inteiras de 16 bits com sinal (*signed word*). A adaptação para o caso de ponto flutuante aqui feita é imediata, pois os endereços das variáveis são sempre dados em 16 bits, independentemente das variáveis serem inteiro-palavra ou reais de ponto flutuante com 4 bytes.

Voltando então à equação III.1, obtém-se o deslocamento de + 06, pois n = 1 (uma só variável A) e m = 1 (primeiro da lista).

Seguindo, no programa 1B, a linha 7000:001E contém uma chamada de subrotina (CALL C0), que tem a finalidade de converter o formato real de precisão simples do BASIC, para o formato IEEE [24] usado pelo PDN-8087. As diferenças nos dois formatos dos números de ponto flutuante podem ser vistas na figura III.1.

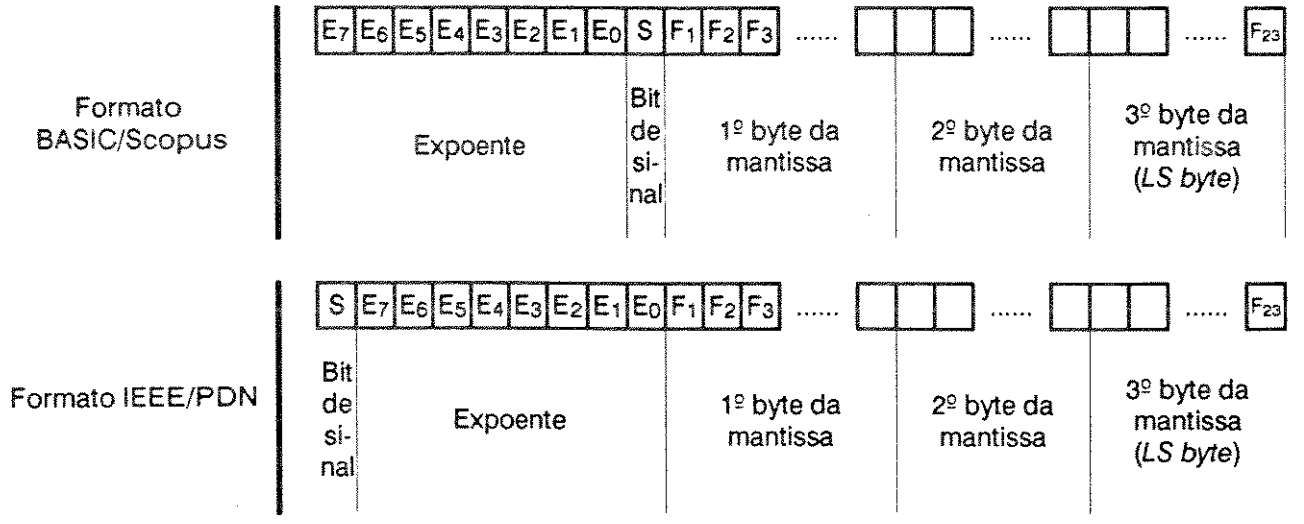


Figura III.1 - Formatos de ponto flutuante do BASIC/Scopus e PDN-8087 (IEEE), em precisão simples de 4 bytes.

Observando a figura III.1 nota-se a diferença de posição no bit de sinal; é zero se positivo (S = +1), ou é um se negativo (S = -1).

Assim, os dois formados são dados pela expressão III.2:

$$X_{BASIC} = (S)\, 2^{E-128}\ \text{Mantissa}$$

$$X_{PDN} = (S)\, 2^{E-126}\ \text{Mantissa} \qquad \textit{(III.2)}$$

Note também das expressões III.2 que a polarização do expoente é 128 para o BASIC, e 126 para o formato IEEE (PDN).

Uma outra informação importante, para ambos os formatos, é que a mantissa normalizada tem o bit mais significativo de forma implícita [25]. Em outras palavras, F_0 não aparece na representação, mas vale sempre a unidade[1].

Como exemplo, o número X=0,75 seria representado nos dois formatos por:

$$X_{BASIC} = (1).2^{128-128}\,(1.2^{-1} + 1.2^{-2} + 0.2^{-3} + 0 + \ldots + 0)$$

$$X_{PDN} = (1).2^{126-126}\,(1.2^{-1} + 1.2^{-2} + 0.2^{-3} + 0 + \ldots + 0)$$

Note que ambos resultam no mesmo valor desejado X = 0,75, mas seriam guardados na memória (4 bytes) sob formas diferentes, ou seja:

Representação de X_{BASIC} ⇒ 80 40 00 00 (Hex)

Representação de X_{PDN} ⇒ 3F 40 00 00 (Hex) *(III.3)*

1 F_0 é o coeficiente de 2^{-1}, F_1 de 2^{-2}, F_2 de 2^{-3}, etc.

Sendo desta forma, a instrução CALL C0 (ver programa 1B) chama a rotina de conversão desejada BASIC ⇒ PDN. Tal rotina faz basicamente três serviços:

a) aponta e traz para o acumulador os dois bytes superiores da palavra, usando ADD SI, + 2 e MOV AX,[SI];

b) da parte superior AH do acumulador AX subtrai-se 2, devido à diferença de polarização nos expoentes, usando-se SUB AH,2;

c) a parte inferior AL deve ser rodada uma vez para a esquerda, através do *carry*, pela instrução RCL AL,1.

Neste ponto é importante observar que o bit de sinal S foi parar no registro de *carry*. Sendo assim, tem-se:

d) uma rotação de todo o acumulador AX (AH e AL) para a direita, com *carry*, utilizando a intrução RCR AX,1.

Neste ponto, o bit de sinal S já está no lugar correto para o formato PDN. Daí para frente, o conteúdo já alterado de AX é devolvido no mesmo lugar (registro de destino e DI) via instrução MOV [DI],AX.

Após o retorno da subrotina C0, o PDN-8087 carrega - FLD DWORD PTR [SI] - a palavra dupla apontada por SI (ver linha 7000:0022). Estes 4 bytes que representam a variável A já modificada são, na verdade, carregados no primeiro - ST (0) - dos oito registros (ST (i), para i = 0,1,2,...,7) existentes no co-processador numérico. Feito isto, a instrução FST ST (2) armazena A também no registro ST(2), para usar depois.

Na linha 7000:0027, o processador principal 8088 volta a trabalhar, executando serviços que ele pode fazer eficientemente. Desta linha em diante, até 7000:0047, o 8088 procura copos estatísticos[1] que têm contagem (conteúdo) diferente de zero. Assim, se um certo copo (ou reservatório) tem ocorrência não-zero, o PDN terá que incluí-lo nas contas, devido à instrução JNZ 0050 da linha 7000:3A; Caso contrário, a UCP incrementa o ponteiro para um valor de 4 bytes acima, para que possa acessar o copo seguinte (ver as quatro instruções INC SI).

Como detalhes, as linhas 40, 42 e 45 testam se já esgotou todo o segmento, isto é, se investigou todos os copos de contagem estatística para SI de 0000 até FFFF.

A instrução OR AX,DX da linha 38 faz um "ou", bit a bit, dos conteúdos dos dois registros AX e DX de 16 bits. Se resulta tudo zero em AX, o conteúdo no copo é zero e deve-se investigar o próximo copo seguindo pela linha 7000:003C. Se, por outro lado, AX não resulta zero, o conteúdo deve ser incluído na fórmula da entropia executada pelo PCN-8087. Este direcionamento para o PDN é feito pela instrução JNZ 0050, ou seja *Jump if Not Zero* para a linha 7000:0050.

1 Estes copos são discutidos em detalhe na próxima seção deste capítulo

Quando a afirmativa acima ocorre, o co-processador numérico recomeça o seu trabalho. Como a capacidade dos reservatórios (copos) é um número inteiro dupla-palavra (32 bits) e não está no formato ponto-flutuante, a instrução FILD DWORD PTR[SI], na linha 51, carrega[1] este inteiro, com sinal, no primeiro registro ST(0).

A tabela III.1.1 ilustra melhor o que ocorre no PDN daí para a frente. Note que a instrução principal é FYL2X, que executa as parcelas (Hi) logarítmicas exigidas pela entropia. O acúmulo $\left(\sum H_i \right)$ é feito pela instrução FADD ST(1),ST que deixa o resultado em ST(1).

Estado anterior ST(0) = ST(2) = A e ST(1) = 0	
FILD DWORD PTR [SI] ST(0) = Ni, ST(1) = ST(3) = A, ST(2) = 0	
FDIV ST,ST(1) $ST(0) = P_i = \frac{N_i}{A}$	FST ST(1) ST(0) = ST(1) = P_i
FYL2X ≡ ST(1).$\log_2$ [ST(0)] ST(0) = $H_i = P_i \log_2 P_i$, ST(1) = 0, ST(2) = A	
FADD ST(1),ST ST(1) = $\sum H_i$, ST(0) = H_i e ST(2) A	
FSTP ST(5) ST(0) = $\sum H_i$, St(1) = A	
FLD ST(1) ST(0) = A, ST(1) = $\sum H_i$, ST(2) = A	

Tabela III.1.1 - Estados do PDN quando passa pela rotina da linha 7000:0050.

Após varrer todo o segmento 6000:0000 até FFFF, a entropia está calculada. Isto ocorre quando ponteiro SI atinge *zero* novamente. A instrução JZ 0070 (*Jump if Zero*) despacha o apontador de instrução IP do micro para a linha 7000:0070, quado então a devolução do resultado obtido deve ser feita ao programa-mãe em BASIC. Na linha 7000:0070 (ver programa 1B), o segmento de dados DS do BASIC deve ser recuperado pela instrução MOV DS,BX. Na linha 74, FLD ST(1) recoloca a entropia (-H) calculada (ainda negativa) no registro ST(0).

1 Note que FLD é convenientemente substituído por FILD, avisando o 8087 para carregar um número em formato inteiro

Na linha 7000:0076, MOV DI,[BP + 06] aponta o local correto para a devolução, como já discutido anteriormente. Deste modo, a instrução FST DWORD PTR [DI] devolve o valor real obtido em precisão simples de 4 bytes, no formato IEEE. A modificação adequada de formatos é feita pela sub-rotina em 00E0 (ver programa 1B). Esta rotina executa a modificação do formato PDN para o BASIC, na variável real calculada pelo PDN (ver figura III.1).

O retorno longínquo RETF 0002 é finalmente feito, na linha 7000:0081, de tal sorte que o micro/PC passa a executar novamente o BASIC interpretado. Observe que o retorno ao BASIC é feito na linha 130 (ver programa 1A), quando então o resultado H = -A da entropia calculada pelo PDN pode ser impressa na tela do PC. O programa finaliza na linha 140.

Na próxima seção, os programas HISTO.BAS e HISTO.BIN são apresentados.

```
10          CLS:PRINT SPC(14) "*****************************************************"
20          PRINT SPC (11) "ESTE PROGRAMA CALCULA A ENTROPIA GLOBAL DAS DIFEREN";CHR$(128):"AS"
30          PRINT SPC(20) "RODAR ANTES O PROGRAMA MCPDM.COM":PRINT
40          PRINT SPC(3) " A F";CHR$(162):"RMULA USADA ":CHR$(144):" h = - ":CHR$(228):"[ Pi * LOG (na base 2)
de Pi]  [bits/EI]
50          PRINT:PRINT SPC(3)":onde Pi = Probab. da i"; CHR$(130);"sima mensagem"
60          DEF SEG = &H7000
80          SUBROT = 0
90          PRINT "A = 262.144 = DIVISOR P/ PROBAB. Pi"
100         A = 262.144!
110         GOSUB 160
120         CALL SUBROT(A)
130         PRINT SPC(15)"A ENTROPIA GLOBAL DA IMAGEM DA ";P$;" COM BLOCO DE TAMANHO ";N$",COR
";M$;",VALE H = ";-A;"  [bits/EI]"
140         DEF SEG:END
160         INPUT "  TAMANHO DO BLOCO ? (8,16,32,...)";N$
170         INPUT "  IMAGEM USADA ? (VERMELHA,VERDE,AZUL,LUMA)";M$
180         INPUT "  NOME USADO P/ IMAGEM ? (ZELDA,COZINHA,PRAIA,SALA)";P$
190         RETURN
```

Programa 1A - ENTROP.BAS

```
7000:0000 EB0E      JMP   0010              ;pula para a linha 10.

7000:0010 55        PUSH  BP                ;guarda o valor corrente na pilha.
7000:0011 90        NOP
7000:0012 89E5      MOV   BP,SP             ;leva o conteúdo de SP para BP.
7000:0014 90        NOP
7000:0015 9B        WAIT                    ;UCP-8088 aguarda PDN-8087.
7000:0016 DBE3      FINIT                   ;inicializa PDN ("reset").
7000:0018 9B        WAIT
7000:0019 D9EE      FLDZ                    ;carrega zero em ST(0).
7000:001B 8B7606    MOV   SI,[BP+06]        ;traz para SI o endereço de A.
7000:001E E89F00    CALL  00C0              ;conversão de formatos BASIC-IEEE.
7000:0021 9B        WAIT
7000:0022 D904      FLD   DWORD PTR [SI]    ;carrega em ST(0) a
7000:0024 9B        WAIT                    ;palavra real A de 32 bits.
7000:0025 DDD2      FST   ST(2)             ;guarda em ST(2) para depois.
7000:0027 BE0000    MOV   SI,0000
7000:002A 8CDB      MOV   BX,DS             ;salva segmentos de dados em BX.
7000:002C 90        NOP
7000:002D B80060    MOV   AX,6000           ;área dos copos é no segmento
7000:0030 8ED8      MOV   DS,AX             ;6000:0000 até FFFF
7000:0032 90        NOP
7000:0033 8B04      MOV   AX,[SI]           ;investiga se o
7000:0035 8B5402    MOV   DX,[SI+02]        ;copo tem
7000:0038 09D0      OR    AX,DX             ;contagem.
7000:003A 7514      JNZ   0050              ;inclui e manda PDN calcular.
7000:003C 46        INC   SI                ;capacidade de cada copo é
7000:003D 46        INC   SI                ;de duas palavras (32 bits).
7000:003E 46        INC   SI
7000:003F 46        INC   SI
7000:0040 89F0      MOV   AX,SI
7000:0042 3D0000    CMP   AX,0000           ;acabou o segmento?
7000:0045 7429      JZ    0070              ;se é zero vai terminar.
7000:0047 EBEA      JMP   0033              ;continue procurando copos não vazios.

7000:00C0 83C602    ADD   SI,+02            ;aponta 2 bytes mais significativos.
7000:00C3 8B04      MOV   AX,[SI]           ;e traz p/ acumulador AX.
7000:00C5 80EC02    SUB   AH,02             ;diminue 2 no expoente.
7000:00C8 D0D0      RCL   AL,1              ;traz bit S para registro "Carry".
```

Programa 1B - ENTROP.BIN

```
7000:00CA D1D8      RCR   AX,1              ;roda tudo para direita.
7000:00CC 89F7      MOV   DI,SI
7000:00CE 8905      MOV   [DI],AX           ;devolve de onde pegou.
7000:00D0 83EE02    SUB   SI,+02            ;até não precisava.
7000:00D3 C3        RET                     ;retorne para quem chamou.

7000:0050 9B        WAIT
7000:0051 DB04      FILD  DWORD PTR [SI]    ;carrega em ST(0) os
7000:0053 9B        WAIT                    ;4 bytes apontados pelo SI.
7000:0054 D8F1      FDIV  ST,ST(1)          ;obtém Pi = Ni/A.
7000:0056 9B        WAIT
7000:0057 DDD1      FST   ST(1)             ;leva também para ST(1).
7000:0059 9B        WAIT
7000:005A D9F1      FYL2X                   ;calcula parcela Hi da entropia.
7000:005C 9B        WAIT
7000:005D DCC1      FADD  ST(1),ST          ;acumulando (som.)Hi em
7000:005F 9B        WAIT                    ;ST(1).
7000:0060 D9DD      FSTP  ST(5)             ;descarta para dar "POP".
7000:0062 9B        WAIT
7000:0063 D9C1      FLD   ST(1)             ;recoloca A em ST(0).
7000:0065 EB05      JMP   003C

7000:0070 8EDB      MOV   DS,BX             ;volta segmento de dados/BASIC.
7000:0072 90        NOP
7000:0073 9B        WAIT
7000:0074 D9C1      FLD   ST(1)             ;coloca (-H) em ST(0).
7000:0076 8B7E06    MOV   DI,[BP+06]        ;aponta área do BASIC.
7000:0079 9B        WAIT
7000:007A D915      FST   DWORD PTR [DI]    ;devolve.
7000:007C E86100    CALL  00E0              ;remodifica formato PDN-BASIC.
7000:007F 5D        POP   BP
7000:0080 90        NOP
7000:0081 CA0200    RETF  0002              ;retorna para linguagem de alto nível.
                                            ;Obs: RETF 2.n, onde n é o número de
                                            ;variáveis passadas.

7000:00E0 83C702    ADD   DI,+02            ;modifica de formato de variável
7000:00E3 89FE      MOV   SI,DI             ;real, em ponto flutuante
7000:00E5 8B04      MOV   AX,[SI]           ;(PDN-BASIC).
7000:00E7 D1E0      SHL   AX,1
7000:00E9 D0D8      RCR   AL,1
7000:00EB 80C402    ADD   AH,02
7000:00EE 8905      MOV   [DI],AX
7000:00F0 C3        RET
```

Programa 1B - ENTROP.BIN (continuação)

III.3 - Histograma

Antes de descrever as técnicas usadas nesta seção, uma ligeira recordação sobre as imagens de teste é útil. Como já foi visto na seção I.4 do capítulo I, as quatro imagens do SMPTE (de teste) são formadas de três componentes primárias R, G e B, cada uma. Destas três são obtidas outras formas úteis, tais como a luminância Y e as diferenças de cor R-Y, B-Y e G-Y, como detalhado na seção mencionada. As componentes R, G e B são introduzidas neste trabalho em três arquivos de dados originais de 262.144 bytes cada um (40000H).

Nesta seção, discute-se o programa implementado para a obtenção de histogramas de imagens originais e processadas. Os resultados obtidos para imagens originais e processadas submetidas à verificação estatística serão apresentados e discutidos no capítulo IV.

Especificamente, fazer os histogramas consiste em contar e desenhar os valores de ocorrência de cada EI (cada nível) dentro de um arquivo da imagem original, por exemplo. Tal número ajuda a identificar os *níveis de cinza* mais dominantes num arquivo de Y, por exemplo. Também, através do número de ocorrência de níveis, pode-se obter algumas informações sobre a complexidade da imagem, a existência ou não de bordas agudas e as regiões planas. Esse cálculo ajuda também na comparação das imagens originais com aquelas que foram processadas (transformadas, por exemplo). Todas as conclusões obtidas destes histogramas (capítulo IV) de imagens pode ajudar na escolha do tipo de sistema (processo, código, etc) mais eficiente a ser aplicado. Eles dão também uma visão complementar aos cálculos de entropia, como já descrito na seção anterior.

Sendo assim, um programa (micro PC) foi desenvolvido utilizando a linguagem BASIC, bem como a linguagem de máquina do Assembly - 8088/8087. Tal interação entre o BASIC e o Assembly é muito útil devido à facilidade/clareza do BASIC e à rapidez do programa de máquina.

No programa HISTO.BAS (ver programa 2A) está o programa principal (programa-mãe), que inclui várias instruções em BASIC, auto-explicativas. Além de comunicar com o *plotter* 6532A da HP-9820A (ver programa 2C), ele está programado para *plotar* as raias do histograma. Também, o programa-mãe chama a subrotina de máquina HISTO.BIN (ver programa 2B) para obter as ocorrências de cada nível, isto é, de cada símbolo discreto da fonte.

Na linha 30 do HISTO.BAS (ver programa 2A) o programa emite na tela do PC a advertência de que é preciso ter rodado antes algum programa, tal como o MCPVBYT.COM, por exemplo. Isto significa apenas que os copos (reservatórios) estatísticos já devem ter conteúdos. Como já se sabe, o segmento 6000:0000 até FFFF da área de memória do micro é reservado para tal. Recordando, são reservados 4 bytes (dupla-palavra) para cada reservatório. Deste modo, são possíveis 16.384 = 2^{14} rótulos diferentes na área. Entretanto, a capacidade do copinho é maior do que 2 bilhões, isto é, equivale a 2^{31}, considerando o chamado inteiro-curto (32 bits), com o bit mais significativo reservado para o sinal ($\pm$).

Note, da descrição acima, que o valor do nível 0 (rótulo 0), por exemplo, é guardado no endereço 6000:0000 - 0003; o rótulo + 1, por sua vez, fica no endereço 6000:0004 - 0007, etc. Um valor negativo como -1, por exemplo, é guardado no endereço 6000:FFFC - FFFF[1].

Na linha 40 de HISTO.BAS, quatro comandos de saída para porta (no formato OUT Ender.,Dado) são efetuados no cartão GPIB do PC[2], disponível no micro empregado. Os dois primeiros fazem o *reset* do cartão; o terceiro coloca o PC como falador (*talker*) no barramento GPIB, enquanto o quarto coloca o cartão em prontidão (*goto standby*).

A linha 50, por sua vez, apenas relembra o fato de que é preciso conectar, pelo cabo GPIB, o PC-IBM ao conjunto calculadora/*plotter* da HP, e rodar o programa HISTOPLOT/HP (ver programa 2C).

---

1 Valores negativos podem ocorrer quando, por exemplo, se aplica o MCP Diferencial (ou alguma Transformada) nos EI's originais.

2 O cartão é o STD-8410, da empresa Sistemas Tecnológicos Digitais S.A., Brasília/DF, seguindo o protocolo da interface GPIB, HP-IB, ou IEEE-488/1978

Nas linhas 60 e 70 do programa 2A, o programa HISTO.BIN (ver programa 2B) é carregado na memória dinâmica, a partir do endereço 7000:0000 (Hex). O programa HISTO.BAS solicita dois valores principais (ver linhas 100 e 130) para saber entre quais valores (níveis) é desejado o histograma. Tais variáveis são recebidas do teclado (console) e associadas às variáveis reais A e B no BASIC. A variável contadora C, por outro lado, é colocada inicialmente em zero (ver linha 130).

Uma ligeira reflexão mostra que o primeiro valor passado é, na verdade, a entropia *H* da linha 110. Após transformá-la em cadeia - *string* - [ver instrução A$ = STR$ (H)] a rotina 340 é chamada. Tal subrotina tem a função de pulverizar toda a cadeia de caracteres, resultante da execução da linha 360. Cada caracter ASCII[1] obtido, representado pela variável S, é mandado pela interface GPIB na subrotina 420 de HISTO.BAS. A linha 420 coloca o caracter S na porta, para que possa ser capturado (lido) pela calculadora HP-9820A, com os devidos protocolos de intercâmbio das linhas 430, 440 e 450.

Esta entropia H, calculada pelo ENTROP.BAS discutido na seção anterior deste capítulo, é lida pelo HP-9820A na linha 2 do programa HISTOPLOT (ver programa 2C), pela instrução RED 13,X em formato livre real.

Depois disso, o programa-mãe passa, de forma análoga, os níveis (ou diferenças no caso do MCPD) mínimo e máximo necessários ao histograma. O HISTO.BAS realiza estas duas passagens nas linhas 120 e 140, respectivamente. O programa 2C os recebe nas linhas 3 e 4. Note, de passagem, que HISTOPLOT espera ainda a variável contadora C em sua linha 5. Tal valor só estará pronto quando o programa 2B - HISTO.BIN for chamado pela primeira vez, na linha 150 de HISTO.BAS. Esta chamada serve para inicializar o PDN-8087 e fazê-lo obter a variável contadora C = B - A + 1, do número de raias do histograma. As linhas 7000:0 até 7000:16A de HISTO.BIN contêm comentário suficientemente explicativos. A variável contadora C é devolvida para o BASIC quando as linhas 7000:192-1A1 são executadas. Mais especificamente, na intrução WAIT FST DWORD PTR [DI] o número de raias desejadas (C = B - A + 1) é colocado pelo PDN, em formato real, na variável C anteriormente posta em zero pelo BASIC.

Em virtude do exposto acima, o ponteiro de instruções IP do 8088 volta a executar o programa HISTO.BAS na linha 160. O valor de C é posto na tela pela linha 170. Na 180, por sua vez, o valor de C é jogado na interface para ser lido então pela HP-9820A (ver programa 2C, linha 5).

Deste ponto em diante, HISTOPLOT prepara suas escalas para fazer o gráfico apropriado. Entretanto, nas suas linhas 13 e 14, o HP-9820A precisa [26,27] ler as variáveis R1 e R2, ou seja, as coordenadas horizontal (nível ou diferença) e vertical (ocorrência), respectivamente. Um detalhe importante para ser lembrado neste instante reside no fato de que as variáveis R1 e R2 citadas são passadas, par a par, pelas linhas 280 (variável A = R1) e 290 (variável B = R2) de HISTO.BAS[2].

---

1 ASCII - American Standard Code for Information Interchange

2 Neste ponto a subrotina (sem parâmetros) das linhas 190/210 do BASIC já foi executada, para calcular o endereço do 1º nível (ver linhas 7000:200 - 25B de HISTO.BIN

Para que a passagem acima seja entendida, deve-se observar, primeiramente, que a linha 250 de HISTO.BAS chama a rotina de máquina, esperando devolução nas variáveis A, B e C. O endereço de entrada é o *off-set* 300 (Hex), como estabelecido na linha 240 do programa-mãe em alto nível. Cada vez que se volta para o HISTO.BAS, na linha 260, os valores inteiros do nível A, da ocorrência B e do contador decrementado C devolvidos pelo HISTO.BIN são também colocados na tela (ver linha 270 do BASIC)[1]. Como já foi frisado, os dois valores (nível e ocorrência) são também mandados para a HP nas linhas 280 e 290, até que C chegue em zero. Enquanto há raias a fazer no histograma, com C ≠ 0, o programa-mãe fica mandando os pares (ver linha 300 de HISTO.BAS, bem como linha 17 de HISTOPLOT/HP) para gráfico e tela.

Os passos de HISTO.BIN, a partir do endereço 7000:300, são amplamente comentados (ver programa 2B). De notável, deve-se citar que a variável C (contador de raias) é também devolvida para HISTO.BAS, de tal modo que este último saiba se tem mais raias a desenhar. O programa-mãe finaliza tudo quando executa as instruções das linhas 310, 320 e 330.

Finalmente, é preciso frisar que o programa HISTOPLOT contém uma série enorme de instruções, que poderiam ser até desprezadas se não fosse desejado um certo grau de sofisticação na apresentação dos histogramas.

Na próxima seção, o algoritmo da transformada rápida de Hadamard (TRH) é apresentado, bem como um programa de máquina que o executa.

1 Uma tabela de valores no papel da impressora pode ser também obtida substituindo PRINT por LPRINT

```
10          CLS:PRINT SPC(14) "*****************************************************"
20          PRINT SPC(11) "ESTE PROGRAMA FAZ O HISTOGRAMA DAS DIFEREN";CHR$(128);"AS NO PLOTTER
DA H&P-9820"
30          PRINT SPC(20) "RODAR ANTES O PROGRAMA MCPDVBYT.COM,POR EXEMPLO":PRINT
40          OUT       701,2:OUT 701,0:OUT 700,128:OUT 701,16
50          PRINT SPC(10) "CONECTE O CABO HPIB DO PC PARA A CALC. + PLOTTER E RODE O PROG.
HISTOPLOT/H&P"
60          DEF SEG = &H7000
70          BLOAD"HISTO.BIN",0
80          SUBROT = 0
90          PRINT
100         INPUT "ENTRE COM A DIFERN. MIN.";A
110         H = 3.49223:A$ = STR$(H):GOSUB 340
120         A$ = STR$(A):GOSUB 340
130         INPUT "ENTRE COM A DIFERN. MAX.";B :C = 0
140         A$ = STR$(B):GOSUB 340
150         CALL SUBROT(A,B,C)
160         PRINT
170         PRINT "O CONTADOR P/ O PLOTTER VALE ";C
180         A$ = STR$(C):GOSUB 340
190         SUBROT = &H200
200         GOTO 210
210         CALL SUBROT
220         GOTO      230
230         PRINT SPC(10) "NIVEL";SPC(10)"OCORREN.";SPC(10)"CONT.":GOTO 240
240         SUBROT = &H300
250         CALL SUBROT(A,B,C)
260         B = INT(B):C = INT(C)
270         PRINT SPC(12) A;SPC(11) B; SPC(10) C
280         A$ = STR$(A):GOSUB 340
290         A$ = STR$(B):GOSUB 340
300         C = C-1:IF C = 0 THEN GOTO 250
310         DEF SEG
320         OUT 701,2:OUT 701,0
330         END
340         GOTO      350
350         FOR I = 1 TO LEN(A$)
360         S = ASC(MID$(A$,I,1))
370         GOSUB 420
380         NEXT I
390         S = 13:GOSUB 420
400         S = 10:GOSUB 420
410         RETURN
420         OUT 696,S
430         E = INP(697)
440         E = E AND 2
450         IF E = 0 THEN GOTO 430
460         RETURN
```

Programa 2A - HISTO.BAS

```
7000:0000 E91D01    JMP   120                 ;pula para linha 120.

7000:0120 55        PUSH  BP                  ;guarda o conteúdo de BP.
7000:0121 89E5      MOV   BP,SP               ;leva ponteiro da pilha SP para o
7000:0123 90        NOP                       ;índice da base BP.
7000:0124 9B        WAIT
7000:0125 DBE3      FINIT                     ;começa no formato de pto. flutuante.
7000:0127 8B760A    MOV   SI,[BP+0A]          ;aponta a primeira variável A.
7000:012A E89300    CALL  01C0                ;modifica formato BASIC-PDN.
7000:012D 9B        WAIT
7000:012E D904      FLD   DWORD PTR [SI]      ;A-registro ST(0).
7000:0130 8B7608    MOV   SI,[BP+08]          ;aponta segunda variável B.
7000:0133 E88A00    CALL  01C0
7000:0136 9B        WAIT
7000:0137 D904      FLD   DWORD PTR [SI] ;ST(0) = B,ST(1) = A.
7000:0139 8CDD      MOV   BX,DS               ;salva DS do BASIC.
7000:013B 90        NOP
7000:013C B80080    MOV   AX,8000
7000:013F 8ED8      MOV   DS,AX               ;área de rascunho.
7000:0141 90        NOP
7000:0142 BF0000    MOV   DI,0000
7000:0145 9B        WAIT
7000:0146 DB1D      FISTP DWORD PTR [DI]      ;descarrega B (nível
7000:0148 83C710    ADD   DI,+10              ;max.) em 8000:0000.
7000:014B 9B        WAIT
7000:014C DB1D      FISTP DWORD PTR [DI]      ;e A (nível min.) em
7000:014E 9B        WAIT
7000:014F DBE3      FINIT                     ;8000:0010 , mas
7000:0151 89FE      MOV   SI,DI
7000:0153 9B        WAIT                      ;convertendo para
7000:0154 DB04      FILD  DWORD PTR [SI]
7000:0156 83EE10    SUB   SI,+10              ;formato inteiro
7000:0159 9B        WAIT
7000:015A DB04      FILD  DWORD PTR [SI]      ;de dupla-palavra.
7000:015C 9B        WAIT
7000:015D D8E1      FSUB  ST,ST(1)            ;B-A+1 = C e o
7000:015F 9B        WAIT
7000:0160 D9E8      FLD1                      ;contador de raias
7000:0162 9B        WAIT
7000:0163 D8C1      FADD  ST,ST(1)            ;do histograma.
7000:0165 9B        WAIT
7000:0166 DB1620    FIST  DWORD PTR [0020]    ;descarrega C
7000:016A EB26      JMP   0192                ;(inteiro) em
                                              ;8000:0020.

7000:0192 90        NOP
7000:0193 8EDB      MOV   DS,BX               ;restabelece segmento DS do BASIC.
7000:0195 90        NOP
7000:0196 8B7E06    MOV   DI,[BP+06]          ;lugar da variável C.
7000:0199 9B        WAIT
7000:019A D915      FST   DWORD PTR [DI]      ;devolve.
7000:019C E84100    CALL  01E0                ;formato PDN-BASIC.
7000:019F 5D        POP   BP
7000:01A0 90        NOP
7000:01A1 CA0600    RETF  0006                ;retorno longo para 3 variaveis.

7000:01C0 83C602    ADD   SI,+02              ;Obs:
7000:01C3 8B04      MOV   AX,[SI]             ;Subrotina (ja comentada)
7000:01C5 80EC02    SUB   AH,02
7000:01C8 D0D0      RCL   AL,1                ;para modificar o n
7000:01CA D1D8      RCR   AX,1
7000:01CC 89F7      MOV   DI,SI               ;real do formato BASIC
7000:01CE 8905      MOV   [DI],AX
7000:01D0 83EE02    SUB   SI,+02              ;para o formato PDN.
7000:01D3 C3        RET

7000:01E0 83C702    ADD   DI,+02              ;Obs:
```

Programa 2B - HISTO.BIN

```
7000:01E3 89FE      MOV   SI,DI
7000:01E5 8B04      MOV   AX,[SI]          ;Subrotina para modificar
7000:01E7 D1E0      SHL   AX,1
7000:01E9 D0D8      RCR   AL,1             ;o n real no formato
7000:01EB 80C402    ADD   AH,02
7000:01EE 8905      MOV   [DI],AX          ;PDN para o formato BASIC.
7000:01F0 C3        RET

7000:0200 EB1E      JMP   0220             ;pula para a linha 220

7000:0220 55        PUSH  BP
7000:0221 89E5      MOV   BP,SP
7000:0223 90        NOP
7000:0224 9B        WAIT
7000:0225 DBE3      FINIT                  ;inicializa PDN.
7000:0227 8CDB      MOV   BX,DS
7000:0229 90        NOP
7000:022A BA0080    MOV   DX,8000
7000:022D 8EDA      MOV   DS,DX            ;área de rascunho é 8000.
7000:022F 90        NOP
7000:0230 EB08      JMP   023A             ;vai para 023A

7000:023A BE1000    MOV   SI,0010          ;aponta A (1 nível),
7000:023D 8B04      MOV   AX,[SI]          ;e traz para acumulador.
7000:023F 01C0      ADD   AX,AX
7000:0241 01C0      ADD   AX,AX            ;(x4) e o seu endereco,
7000:0243 89C6      MOV   SI,AX
7000:0245 BF1000    MOV   DI,0010
7000:0248 8905      MOV   [DI],AX          ;e guarda em 8000:0010.
7000:024A 90        NOP
7000:024B 90        NOP
7000:024C 90        NOP
7000:024D 8EDB      MOV   DS,BX            ;volta para segmentos de
7000:024F 90        NOP                    ;dados do BASIC.
7000:0250 EB07      JMP   0259


7000:0259 90        NOP
7000:025A 5D        POP   BP               ;retorna para alto nível.
7000:025B CB        RETF

7000:0300 EB1E      JMP   0320             ;pula para a linha 0320

7000:0320 55        PUSH  BP
7000:0321 89E5      MOV   BP,SP
7000:0323 90        NOP
7000:0324 9B        WAIT
7000:0325 DBE3      FINIT                  ;inicializa PDN.
7000:0327 8CDB      MOV   BX,DS
7000:0329 90        NOP
7000:032A BA0080    MOV   DX,8000          ;área de trabalho.
7000:032D 8EDA      MOV   DS,DX
7000:032F 90        NOP
7000:0330 BE1000    MOV   SI,0010          ;aponta endereço da
7000:0333 8B04      MOV   AX,[SI]          ;primeira raia.
7000:0335 89C1      MOV   CX,AX            ;salva em CX.
7000:0337 050400    ADD   AX,0004          ;incrementa para o próximo.
7000:033A 89F7      MOV   DI,SI
7000:033C 8905      MOV   [DI],AX          ;leva de volta para 8000:0010
7000:033E 1E        PUSH  DS               ;salva 8000:
7000:033F 90        NOP
7000:0340 BA0060    MOV   DX,6000
7000:0343 8EDA      MOV   DS,DX            ;vai para área de copos.
7000:0345 90        NOP
7000:0346 89CE      MOV   SI,CX            ;aponta raia atual.
7000:0348 9B        WAIT
7000:0349 DB04      FILD  DWORD PTR [SI]   ;ST(0) = B
```

Programa 2B - HISTO.BIN (continuação)

```
0:      CMD "?5";CFG 3
1:      4E4#R10
2:      FMT *;RED 13,X
3:      FMT *;RED 13,A
4:      FMT *;RED 13,B
5:      FMT *;RED 13,C
6:      FLT 9;PRT "NMIN = ",A
7:      PRT "NMAX = ",B
8:      PRT "CONT. = ",C
9:      (A + B)/2#R7
10:     A-R7#R3;B-R7#R4
11:     -R10/2#R5;R10/2#R6
12:     SCL R3,R4,R5,R6
13:     FMT *;RED 13,R1
14:     FMT *;RED 13,R2
15:     C-1#C
16:     IF R20;GTO 19
17:     IF C = 0;GTO 27
18:     GTO 13
19:     IF FLG 3 = 1;GTO 24
20:     GTO 21
21:     .85(R1-R7)#R8;.85(R2-R6)#R9
22:     PLT R8,.85R5;PLT R8,R9;PEN
23:     SFG 3;GTO 17
24:     .85(R1-R7)#R8;.85(R2-R6)#R9
25:     R8,R9;PLT R8,.85R5;PEN
26:     CFG 3;GTO 17
27:     10#R20;R10/R20#R11;CFG 3
28:     R11#R50;R11#R60
29:     0#R11
30:     .85(R11-R6)#R50
31:     IF FLG 3 = 1;GTO 36
32:     LTR R3,R50,211;FLT 1;PLT R11;FLT 9
33:     PLT .85R3,R50;PLT .85R4,R50;PEN;SFG 3
34:     R11 + R60#R11;IF R11
10;GTO 30
35:     GTO 38
36:     PLT .85R4,R50;PLT .85R3,R50;PEN
37:     LTR R3,R50,211;FLT 1;PLT R11;FLT 9;CFG
3;GTO 34
38:     R4-R3 + 1#R20
39:     IF R2017;GTO 44
40:     IF R20129;GTO 51
41:     ENT "PASSO DOS NIVEIS P/ PLOT.?",R11
42:     GTO 53
43:     GTO 62
44:     1#R11;R11#R50;R50#R60
45:     A#R11
46:     .85(R11-R7)#R50;R50 + .01R3#R50
47:     LTR R50,.94R5,211
48:     FXD 0;PLT R11;FLT 9
49:     R11 + R60#R11;IF R11;GTO 46
50:     GTO 43
51:     FLT 9
52:     R20/10#R11;INT (R11 + .5)#R11
53:     R11#R50;R50#R60
54:     A#R11
55:     .85(R11-R7)#R50;R50 + .01R3#R50
56:     LTR R50,.94R5,211
57:     FXD 0;PLT R11;FLT 9
58:     R11 + R60#R11;IF BR11;GTO 55
59:     B#R11;.85(R11-R7)#R50;R50 + .01R3#R50
60:     LTR R50,.94R5,211;FXD 0;PLT R11
61:     FLT 9;GTO 43
62:     LTR.91R4,.3R5,222
63:     PLT "OCORRENCIA"
64:     PLT .89R4,.1R6;PLT .89R4,.3R6;PLT
.87R4,.25R6;PEN
65:     PLT .89R4,.3R6;PLT .91R4,.25R6;PEN
66:     LTR .15R3,R5,231
67:     PLT "NIVEL"
68:     PLT .15R4,.985R5;PLT .3R4,.985R5;PLT
.25R4,.97R5;PEN
69:     PLT .3R4,.985R5;PLT .25R4,R5;PEN
70:     LTR .1R4,.87R6,211
71:     PLT "ENTROPIA **"
72:     LTR .35R4,.87R6,211
73:     FXD 5;PLT "H  ";PLT X;PLT " (BIT/EI)";FLT
9;A#R11
74:     PLT .38R4,.875R6;PLT .39R4,.875R6;PEN
75:     PLT .38R4,.875R6;PLT.39R4,.885R6;PEN
76:     .85(R11-R7)#R50
77:     PLT R50,.88R5;PLT R50,.865R5;PEN
78:     R11 + R60#R11;IF BR11;GTO 76
79:     B#R11;.85(R11-R7)#R50
80:     PLT R50,.88R5;PLT R50,.865R5;PEN
81:     END
```

Programa 2B - HISTO.BIN (continuação)

```
7000:034B 1F       POP  DS                ;(ocorrencia atual) e volta
7000:034C 90       NOP  ;para 8000:
7000:034D BE2000   MOV  SI,0020          ;aponta contador C
7000:0350 FF0C     DEC  WORD PTR [SI]    ;e decrementa.
7000:0352 9B       WAIT
7000:0353 DB04     FILD DWORD PTR [SI]   ;ST(0) = C(novo) e
7000:0355 89C8     MOV  AX,CX            ;ST(1) = B.
7000:0357 B102     MOV  CL,02
7000:0359 D3F8     SAR  AX,CL            ;(/4) o atual endereco
7000:035B 7813     JS   0370             ;dando o atual nível A.
7000:035D B90000   MOV  CX,0000
7000:0360 890E     MOV  [0032],CX        ;cria marca 0000 em
7000:0364 EB2A     JMP  0390             ;8000:32, se positivo.

7000:0370 B9FFFF   MOV  CX,FFFF          ;se negativo,
7000:0373 890E32   MOV  [0032],CX        ;guarda marca FFFF,
7000:0377 EB17     JMP  0390             ;em 8000:32.

7000:0390 BF3000   MOV  DI,0030          ;guarda nível a em 8000:30
7000:0393 8905     MOV  [DI],AX          ;para depois.
7000:0395 89FE     MOV  SI,DI
7000:0397 9B       WAIT
7000:0398 DB04     FILD DWORD PTR [SI]   ;ST(0) = nível atual.
7000:039A 8EDB     MOV  DS,BX
7000:039C 90       NOP
7000:039D 8B7E0A   MOV  DI,[BP+0A]       ;aponta A (nível) p/ devolver.
7000:03A0 9B       WAIT

7000:03A1 D91D     FSTP DWORD PTR [DI]
7000:03A3 E83AFE   CALL 01E0
7000:03A6 8B7E06   MOV  DI,[BP+06]       ;aponta C para devolver
7000:03A9 9B       WAIT
7000:03AA D91D     FSTP DWORD PTR [DI]
7000:03AC E831FE   CALL 01E0
7000:03AF 8B7E08   MOV  DI,[BP+08]       ;aponta B (ocorrência) p/ devolv.
7000:03B2 9B       WAIT
7000:03B3 D91D     FSTP DWORD PTR [DI]
7000:03B5 E828FE   CALL 01E0             ;formato PDN-BASIC.
7000:03B8 5D       POP  BP
7000:03B9 90       NOP
7000:03BA CA0600   RETF 0006             ;volta , com 3 variáveis.
```

Programa 2C - HISTOPLOT/HP

III.4 - Transformada Rápida de Hadamard - TRH

Das relações II.6, II.8, II.9 e II.10 do capítulo anterior, a TDH unidimensional com tamanho do bloco $N = 4$, por exemplo, pode ser escrita na seguinte forma matricial:

$$\begin{bmatrix} \theta(0) \\ \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad (III.4)$$

onde x(i) são os EI's originais e θ(i) são os coeficientes resultantes da TDH em questão, com i = 0, 1, 2 e 3. Da mesma maneira a TDH inversa é dada por:

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} \theta(0) \\ \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix} \qquad (III.5)$$

Realizando a multiplicação matricial pedida nas relações III.4 ou III.5 tem-se um número de multiplicações igual a N^2 e um número de somas igual a $N(N-1) = N^2 - N$. Se o tamanho do bloco N for grande, $N^2 >> N$, então o número de operações matemáticas (multiplicações + somas) é aproximadamente igual a $2N^2$.

Para diminuir esse número de operações matemáticas foi desenvolvido um algoritmo rápido computacional mostrado na figura III.2 [2, 12, 13 e 15].

Tal algoritmo é um resultado imediato da computação matricial da relação III.4, mas com a TDH na sua forma natural, não ordenada (ver H_4' da relação II.6 do capítulo anterior). A menos do fator ½ deixado para a TRH Inversa, os elementos da matriz de Hadamard tem apenas 1 e -1. Sendo assim, as operações de multiplicação tornam-se adições e subtrações, de tal modo que o número de computações é reduzido para $N \log_2 N$ ao invés de aproximadamente $2N^2$, como já frisado.

O cálculo dos coeficientes de Hadamard usando a TRH da figura III.2 pode então ser sistematizado. Na primeira etapa são feitas N' = N/2 somas x(0) + x(N/2), x(1) + x(N/2 + 1),... e x(N/2 − 1) + x(N−1), bem como N/2 subtrações x(0) − x(N/2), x(1) − x(N/2 + 1),... e x(N/2 − 1) − x(N − 1).

Nas seguinte etapas (total de $\log_2 N$ etapas), cada sub-bloco N' resultante da etapa anterior é dividido em N/2 somas e N/2 subtrações, até chegar às chamadas borboletas, ou seja, apenas uma soma e uma subtração. Tais borboletas darão os coeficientes θ(i) de Hadamard desejados.

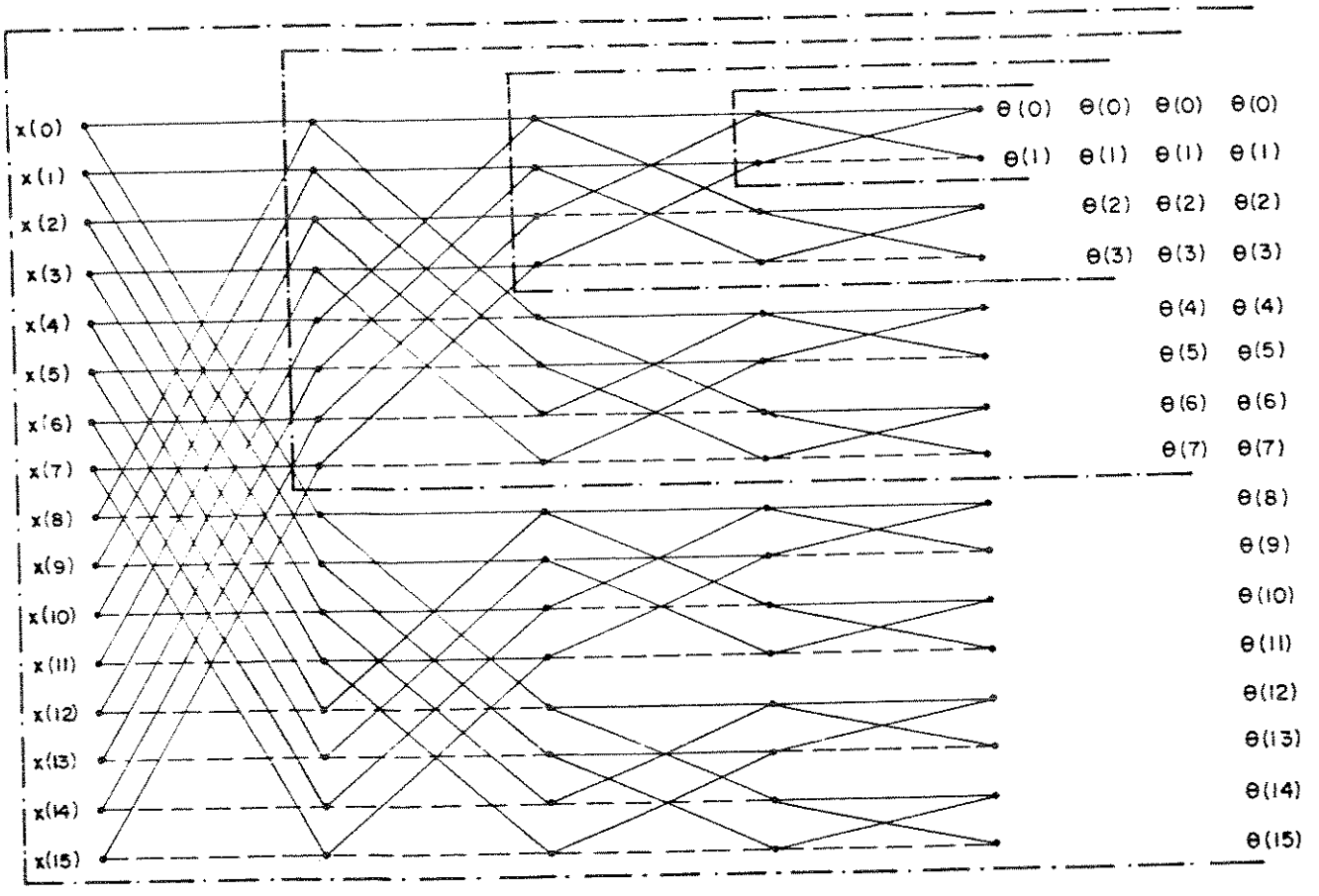


Figura III.2 - A TRH para N = 2, 4, 8 e 16

Cabe notar que o chamado termo CC (componente contínua) da transformada, $\theta(0)$, é sempre igual à soma de todos os N elementos x(i). Sendo assim, $\theta(0)$ tem o maior valor dos coeficientes. Outra observação importante é domínio dos coeficientes que pode oscilar de $-N/2 \times 255$ ao $N \times 255$ enquanto os EI's (positivos) originais variam entre 0 e 255.

O programa 3 (TRH4UH.COM) mostra o cálculo da TRH observando a sistematização já vista na figura III.2. Mais uma vez o programa foi criado dentro do utilitário DEBUG, usando instruções do Assembler do IBM-PC [28,29].

Em resumo, a primeira parte do programa TRH4UH.COM lê os bytes (EI's originais) dos segmentos 2000: e 3000:, transforma-os em palavras de dois bytes e armazena-os nos segmentos 6000:, 7000:, 8000: e 9000:, de acordo com as instruções comentadas nas linhas 100 a 142 do programa 3. Na parte seguinte, começando pela linha 14A, as palavras da parte anterior são lidas dos segmentos 6000: a 9000:, para calcular os coeficientes θ(i). Por exemplo, $\theta(0) = x(0) + x(1) + x(2) + x(3)$ é obtido nas linhas 159 até 165 e $\theta(1) = x(0) + x(1) - x(2) - x(3)$ nas linhas 16A a 179. Após o cálculo de cada θ(i) a subrotina 1C0 é chamada para armazenar os coeficientes (inteiro-palavra com sinal) nos segmentos

2000: até 5000:. Feito o cálculo de todos os coeficientes, as linhas 1B6 a 1B8 devolvem o segmento de dados ao estado original e o programa é encerrado.

Desta maneira, os coeficientes de metade de uma imagem original são computados e armazenados nos segmentos, estando disponíveis para salvar em disco ou então para efetuar qualquer outro processamento. No caso em que os histogramas ou entropias das imagens transformadas (ver seções III.2 e III.3 deste capítulo) os coeficientes $\theta(i)$ são passado para os respectivos programas. Se, por outro lado, a transformada híbrida é o alvo, os coeficientes $\theta(i)$ são utilizados para calcular as diferenças obtidas pelo MCP Diferencial, como será discutido na seção III.6 deste capítulo. Os resultados-entropias e histogramas serão mostrados e comentados no próximo capítulo.

```
09AC:0100 1E          PUSH  DS              ;salva o conteúdo do seg. de dados.
09AC:0101 90          NOP
09AC:0102 BB0060      MOV   BX,6000         ;leve BX p/ segmento 6000:
09AC:0105 BA0020      MOV   DX,2000         ;leve DX p/ segmento 2000:
09AC:0108 BE0000      MOV   SI,0000         ;zera o indicador de fonte.
09AC:010B B90000      MOV   DI,0000         ;zera o indicador de destino.
09AC:010E 8EDA        MOV   DS,DX           ;leve o seg. de dados ao indic. 2000:
09AC:0110 90          NOP
09AC:0111 B80000      MOV   AX,0000
09AC:0114 8B04        MOV   AX,[SI]         ;carrega a palavra apontada pelo SI.
09AC:0116 25FF00      AND   AX,00FF         ;mascara byte menos significativo.
09AC:0119 E81400      CALL  0130            ;chama a linha 130.
09AC:011C 90          NOP
09AC:011D 90          NOP
09AC:011E 46          INC   SI              ;incrementa para pegar a seguinte.
09AC:011F 75F3        JNZ   0114
09AC:0121 81C200      ADD   DX,1000         ;adiciona 1000 ao DX.
09AC:0125 81FA00      CMP   DX,4000         ;compare com 4000.
09AC:0129 75E3        JNZ   010E            ;pula , se nao é zero , DX Ö 4000.
09AC:012B EB1D        JMP   014A

09AC:0130 1E          PUSH  DS
09AC:0131 90          NOP
09AC:0132 8EDB        MOV   DS,BX
09AC:0134 90          NOP
09AC:0135 8905        MOV   [DI],AX         ;guarda conteudo de AX em memor. DI.
09AC:0137 47          INC   DI              ;incrementa DI.
09AC:0138 47          INC   DI
09AC:0139 7403        JZ    013E            ;pula,se DI=0,p/ linha 13E, senão segue.
09AC:013B 1F          POP   DS              ;volta ao estado anterior do segmento.
09AC:013C 90          NOP
09AC:013D C3          RET                   ;volta o fluxo do programa.
09AC:013E 81C300      ADD   BX,1000         ;vai ao segmento próximo.
09AC:0142 EBF7        JMP   013B            ;volta à linha 13B.

09AC:014A BE0000      MOV   SI,0000
09AC:014D BF0000      MOV   DI,0000
09AC:0150 BB0020      MOV   BX,2000         ;para onde vai.
09AC:0153 BA0060      MOV   DX,6000         ;de onde vem.
09AC:0156 8EDA        MOV   DS,DX           ;segmento-fonte.
09AC:0158 90          NOP
09AC:0159 8B04        MOV   AX,[SI]         ;carrega palavra apont. pelo SI em AX.
09AC:015B 46          INC   SI
09AC:015C 46          INC   SI
09AC:015D 0304        ADD   AX,[SI]         ;soma com AX a palavra apontada
09AC:015F 46          INC   SI              ;pelo novo SI.
09AC:0160 46          INC   SI
09AC:0161 0304        ADD   AX,[SI]         ;soma a terceira.
09AC:0163 46          INC   SI
09AC:0164 46          INC   SI
09AC:0165 0304        ADD   AX,[SI]         ;soma a quarta.
09AC:0167 E85600      CALL  01C0            ;guarda @(0).
09AC:016A 83EE06      SUB   SI,+06          ;sub. 6 do apontador da fonte SI.
09AC:016D 8B04        MOV   AX,[SI]
09AC:016F 46          INC   SI
09AC:0170 46          INC   SI
09AC:0171 0304        SUB   AX,[SI]
09AC:0173 46          INC   SI
09AC:0174 46          INC   SI
09AC:0175 2B04        ADD   AX,[SI]
09AC:0177 46          INC   SI
09AC:0178 46          INC   SI
09AC:0179 2B04        SUB   AX,[SI]
09AC:017B E84200      CALL  01C0            ;guarda @(1).
09AC:017E 83EE06      SUB   SI,+06
```

Programa 3 - TRH4UH.COM

```
09AC:0181 8B04      MOV   AX,[SI]
09AC:0183 46        INC   SI
09AC:0184 46        INC   SI
09AC:0185 2B04      ADD   AX,[SI]
09AC:0187 46        INC   SI
09AC:0188 46        INC   SI
09AC:0189 2B04      SUB   AX,[SI]
09AC:018B 46        INC   SI
09AC:018C 46        INC   SI
09AC:018D 0304      SUB   AX,[SI]
09AC:018F E82E00    CALL  01C0          ;guarda @(2).
09AC:0192 83EE06    SUB   SI,+06
09AC:0195 8B04      MOV   AX,[SI]
09AC:0197 46        INC   SI
09AC:0198 46        INC   SI
09AC:0199 2B04      SUB   AX,[SI]
09AC:019B 46        INC   SI
09AC:019C 46        INC   SI
09AC:019D 0304      SUB   AX,[SI]
09AC:019F 46        INC   SI
09AC:01A0 46        INC   SI
09AC:01A1 2B04      ADD   AX,[SI]
09AC:01A3 E81A00    CALL  01C0          ;guarda @(3).
09AC:01A6 90        NOP
09AC:01A7 83C602    ADD   SI,+02        ;avança 2 bytes.
09AC:01AA 75AD      JNZ   0159
09AC:01AC 81C200    ADD   DX,1000       ;pula p/ o próximo segmento.
09AC:01B0 81FA00    CMP   DX,A000
09AC:01B4 75A0      JNZ   0156
09AC:01B6 1F        POP   DS
09AC:01B7 90        NOP
09AC:01B8 CD20      INT   20            ;termina o programa.

09AC:01C0 1E        PUSH  DS
09AC:01C1 90        NOP
09AC:01C2 8EDB      MOV   DS,BX         ;vai guardar.
09AC:01C4 90        NOP
09AC:01C5 8905      MOV   [DI],AX
09AC:01C7 47        INC   DI
09AC:01C8 47        INC   DI
09AC:01C9 7403      JZ    01CE          ;acabou o seg. de destino?
09AC:01CB 1F        POP   DS
09AC:01CC 90        NOP
09AC:01CD C3        RET
09AC:01CE 81C300    ADD   BX,1000       ;incrementa o segmento-destino.
09AC:01D2 EBF7      JMP   01CB
```

Programa 3 - TRH4UH (continuação)

III.5 - Transformada Rápida do Cosseno - TRC

Na mesma linha de discussão da seção anterior, a transformada rápida do cosseno TRC pode ser derivada da TDC unidimensional dada pela relação II.40 do capítulo anterior. Dado o tamanho do bloco N = 4, por exemplo, a equação II.40 pode ser escrita desta forma:

$$\theta(j) = \sqrt{\frac{1}{2}}\,\alpha(j)\sum_{i=0}^{3} x(i)\cos\frac{(2i+1)j\pi}{8} \qquad (III.6)$$

onde $\theta(j)$ é o coeficiente resultante, x(i) é o EI, para i = 0, 1, 2 e 3. O parâmetro $\alpha(j)$ vale $1/\sqrt{2}$ para j = 0 e 1 para os outros valores de j. Calculando:

$$\theta(0) = 0{,}5[x(0) + x(1) + x(2) + x(3)]$$

$$\theta(1) = 0{,}653281.x(0) + 0{,}270598.x(1) - 0{,}270598.x(2) - 0{,}653281.x(3) \qquad (III.7)$$

$$\theta(2) = 0{,}5\left[x(0) - x(1) - x(2) + x(3)\right]$$

$$\theta(1) = 0{,}270598.x(0) - .0{,}653281x(1) + 0{,}653281.x(2) - 0{,}270598.x(3)$$

Das relações III.7, uma sistemática combinação dos coeficientes pode ser desenvolvida pelo algoritmo rápido de duas etapas mostrado na figura III.3

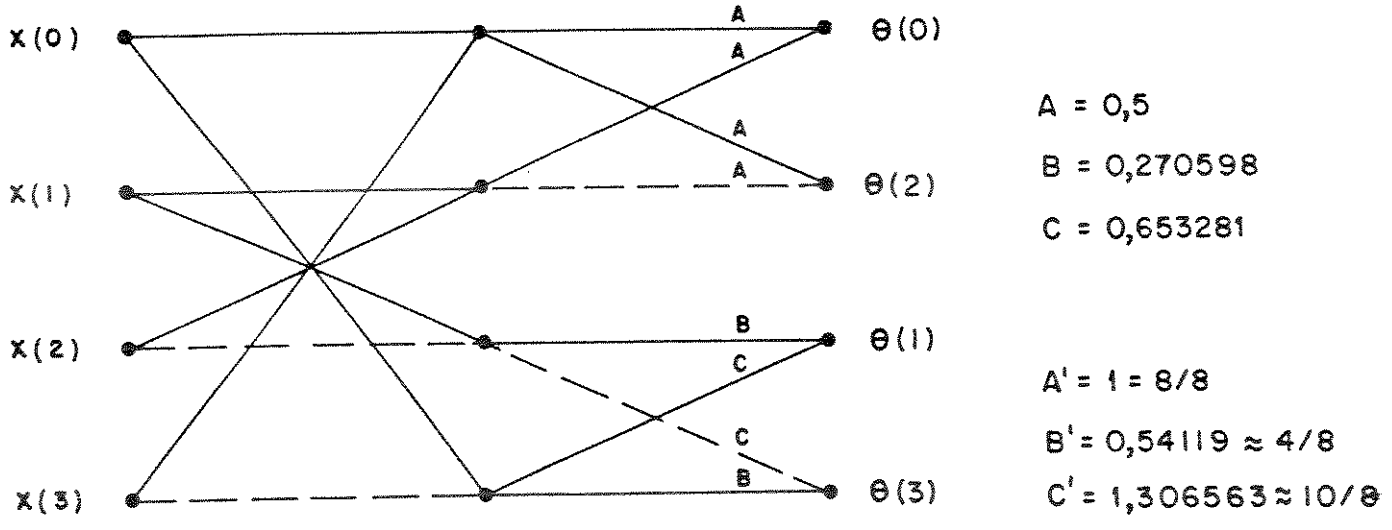


Figura III.3 - TRC (N = 4)

Considerações análogas levam ao algoritmo rápido inverso mostrado na figura III.4, para N = 4.

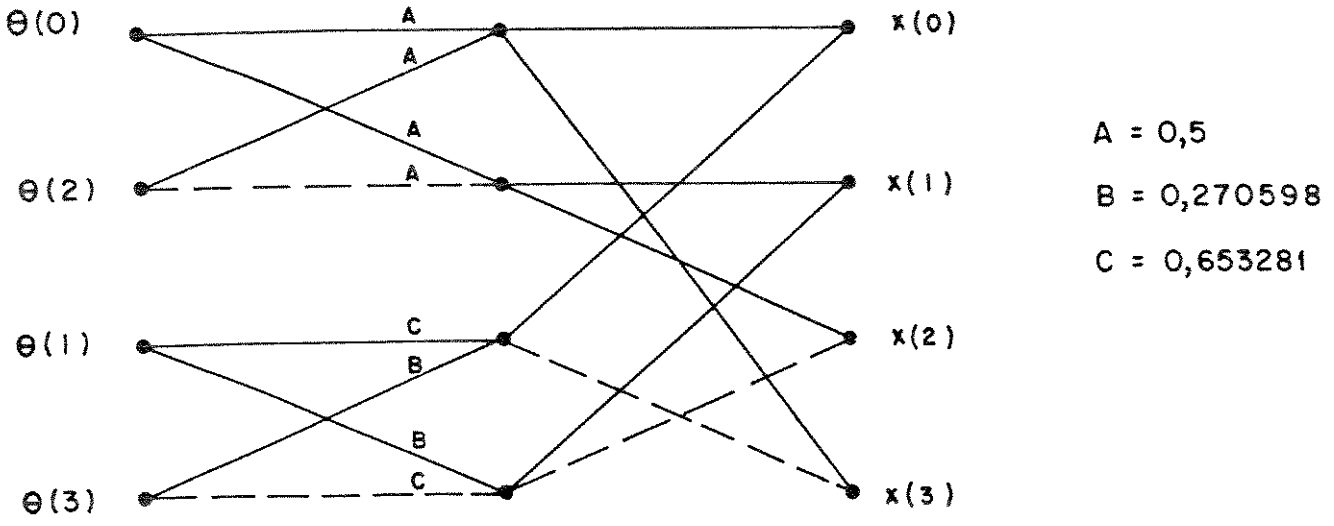


Figura III.4 O TRCI (N = 4)

Das relações II.40 e II.41 (formas algébricas), bem como da relação matricial II.44 (ver capítulo II), os algoritmos rápidos TRC e TRCI para N = 8 podem ser também sistematizados, como visto nas figuras III.5 e III.6.

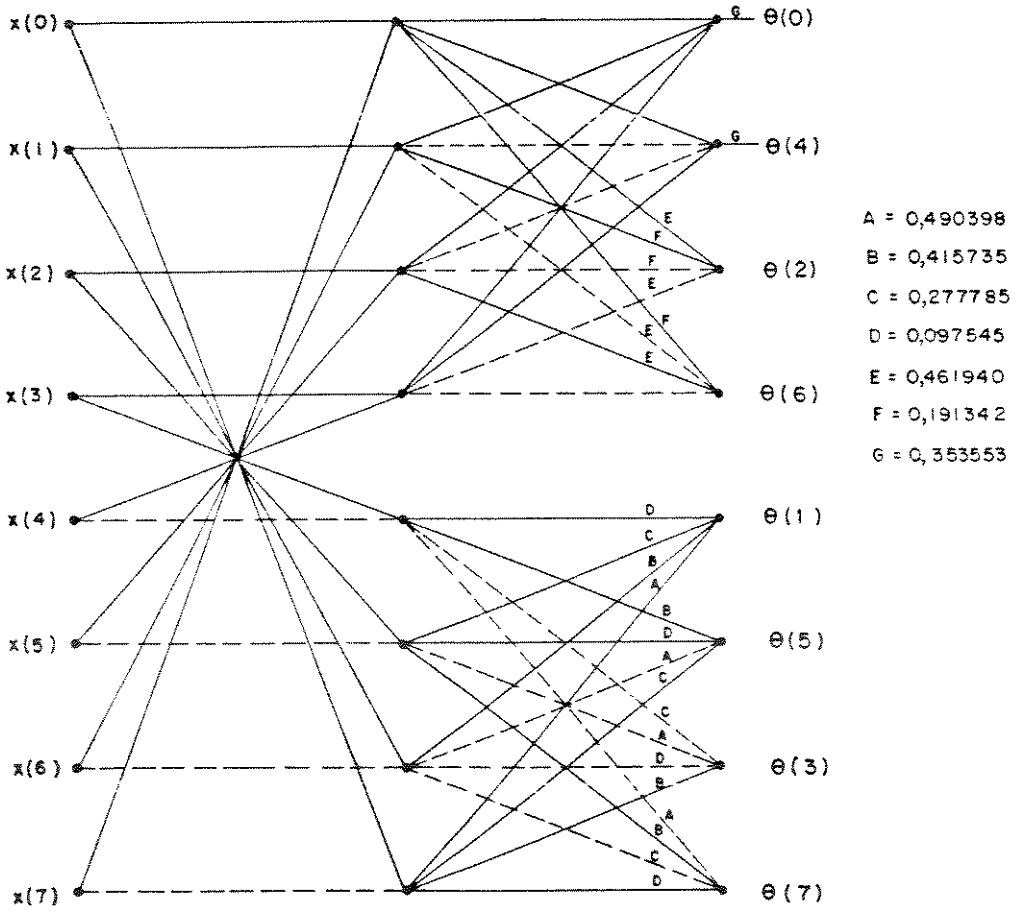


FIG. III.5 – TRC (N = 8)

Figura III.5 - TRC (N = 8)

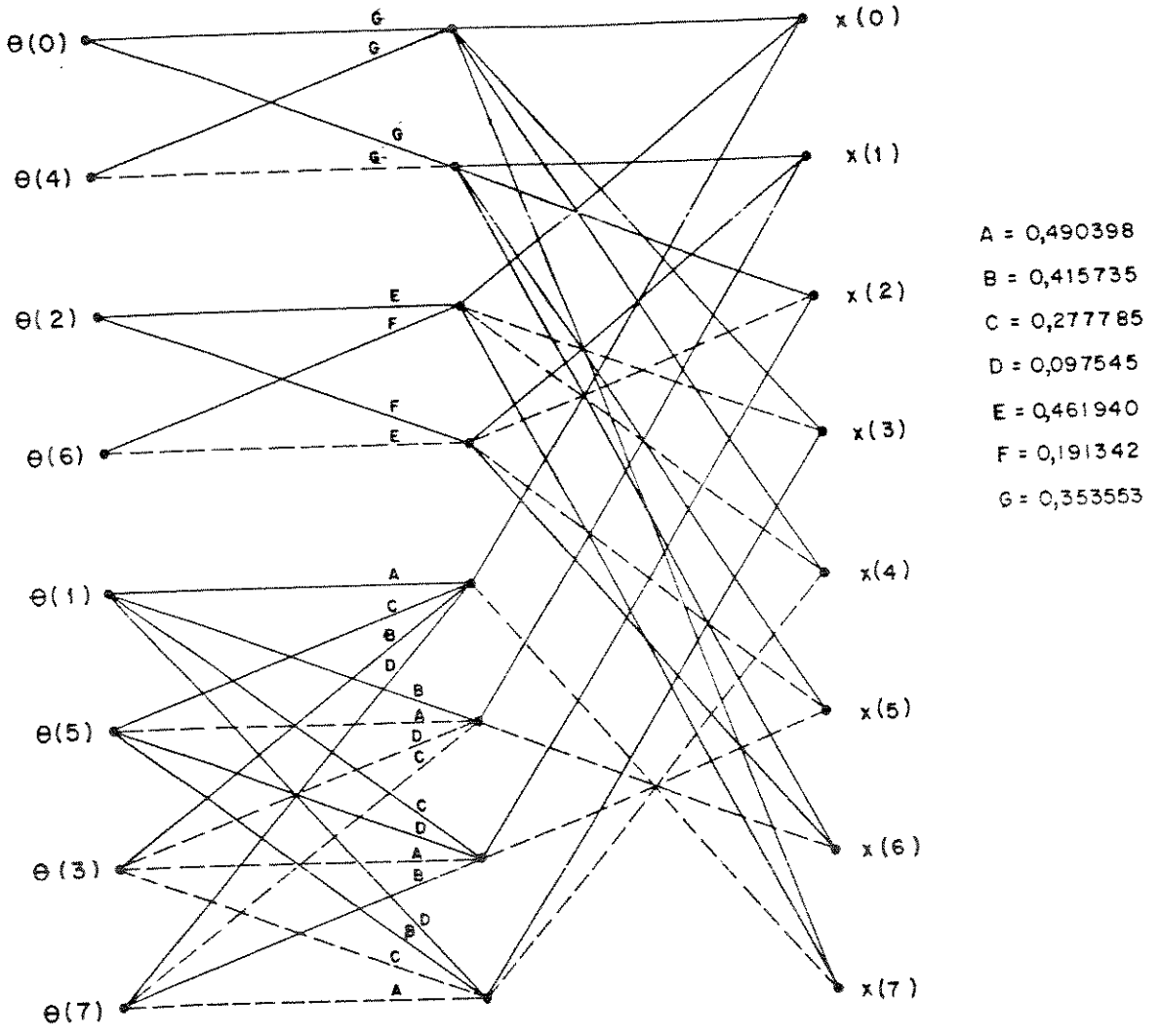


FIG. III. 6 – TRCI (N = 8)

Figura III.6 - TRCI (N = 8)

Um programa em linguagem de máquina (ver programa 4 - TRC4UH.COM) foi elaborado para calcular a TRC de tamanho de bloco N = 4). Basicamente, o programa é dividido em duas etapas. Na primeira etapa, os bytes x(0) e x(3) são lidos do segmento 2000:, transformados cada um numa palavra de 16 bits, com as somas as as diferenças sendo então armazenadas nos segmentos 6000: a 9000: (ver programa 4 nas linhas 100 a 130). Em seguida, um procedimento semelhante é feito para obter x(1) + x(2) e x(1) − x(2), completando-se assim a primeira etapa (ver linhas 134 até 16A).

As borboletas finais da TRC passam então a ser calculadas na segunda etapa, iniciando pela linha 1A3. As palavras resultantes da etapa anterior são lidas dos segmentos 6000: até 9000:, para serem efetuadas as somas e subtrações desejadas, com as devidas constantes multiplicativas (ver figura III.3). Como a UCP-8088 só executa multiplicações de inteiros, as constantes A', B' e C' foram utilizadas, empregando primeiramente apenas os numeradores (9, 4 e 10) das frações equivalentes (ver linhas 1D7, 1E6, 1F0 e 200, por exemplo). Posteriormente, a fim de que os coeficientes não tenham um ganho desnecessário, o desvio para a linha 248 divide todos os valores de θ(i) pelo denominador 8 das frações citadas.

A respeito da problemática acima, é importante notar que a aplicação da TRC produz normalmente coeficientes no formato real de ponto flututante. Tal ocorre, uma vez que durante o processo de cálculo dos coeficientes são usados os fatores multiplicativos A, B e C (ver figura III.3) que varia de -1 até 1 (ver também as funções básicas da figura II.3, do capítulo II).

Esses coeficientes no formato real podem ser calculados pelo PDN (8087) do PC. Entretanto, se os coeficientes obtidos são simplesmente arredondados ao inteiro mais próximo, após calcular a transformada inversa, ocorrerá um erro de processamento. Como esse trabalho é dedicado à codificação exata, então esse erro é inaceitável. No programa aqui discutido, o PDN não foi empregado. Optou-se pelo uso apenas do processador principal 8088, fazendo as multiplicações por números inteiros, ou seja, pelos numeradores das frações já comentadas (ver A', B' e C' da figura III.3). A divisão por 8 (denominadores) é trivial, usando a instrução SHR (*shift right*, ou deslocamento para a direita) por três vezes, pois $2^{-3} = 1/8$[1]. Os problemas de arredondamento para cima, ou apenas truncamento, são administrados pelos trechos do programa 4 existentes de 248 a 258, bem como de 260 a 26D.

Finalizado esta seção, deve-se frisar que o procedimento seguido permite então a obtenção de coeficientes discretos, e não "contínuos" em formato real de precisão simples, o que dificultaria o cálculo de entropia e histogramas pelos programs já discutidos nas seções anteriores.

1 Para transformadas com N muito grande, esta divisão pode ser vital, para inpedir estouro de capacidade de registro de 16 bits.

```
09AC:0100 1E          PUSH  DS              ;salva conteúdo do seg. de dados.
09AC:0101 90          NOP
09AC:0102 BA0020      MOV   DX,2000         ;segmento fonte.
09AC:0105 BB0060      MOV   DX,6000         ;segmento destino.
09AC:0108 BE0000      MOV   SI,0000
09AC:010B BF0000      MOV   DI,0000
09AC:010E 8EDA        MOV   DS,DX           ;os dados a serem lidos em 2000:
09AB:0110 90          NOP
09AC:0111 8B04        MOV   AX,[SI]         ;carrega a palavra apontada pelo SI.
09AC:0113 25FF00      AND   AX,00FF         ;zera byte mais sign. do acumulado .
09AC:0116 46          INC   SI
09AC:0117 46          INC   SI
09AC:0118 8B0C        MOV   CX,[SI]         ;traz x(3) para CH.
09AC:011A 81E100      AND   CX,FF00         ;leva x(3) para CL.
09AC:011E 86E9        XCHG  CH,CL
09AC:0120 50          PUSH AX               ;salva x(0).
09AC:0121 90          NOP
09AC:0122 51          PUSH  CX              ;salva x(3).
09AC:0123 90          NOP
09AC:0124 01C8        ADD   AX,CX           ;obtém soma X(0) + x(3).
09AC:0126 E84700      CALL  0170
09AC:0129 90          NOP
09AC:012A 59          POP   CX              ;recupera x(3).
09AC:012B 90          NOP
09AC:012C 58          POP   AX              ;recupera x(0).
09AC:012D 90          NOP
09AC:012E 29C8        SUB   AX,CX           ;obtém diferença x(0)-x(3).
09AC:0130 E86000      CALL  0193            ;vai guardar.
09AC:0133 90          NOP
09AC:0134 4E          DEC   SI              ;decrementa para pegar x(1).
09AC:0135 4E          DEC   SI
09AC:0136 8B04        MOV   AX,[SI]
09AC:0138 2500FF      AND   AX,FF00         ;peneira o x(1) e
09AC:013B 86E0        XCHG  AH,AL           ;leva para AL.
09AC:013D 46          INC   SI
09AC:013E 46          INC   SI
09AC:013F 8B0C        MOV   CX,[SI]         ;traz x(2).
09AC:0141 81E1FF      AND   CX,00FF         ;peneira x(2) no CL.
09AC:0145 50          PUSH  AX              ;salva x(1).
09AC:0146 90          NOP
09AC:0147 51          PUSH  CX              ;salva x(2).
09AC:0148 90          NOP
09AC:0149 01C8        ADD   AX,CX           ;x(1) + x(2).
09AC:014B E82200      CALL  0170
09AC:014E 90          NOP
09AC:014F 59          POP   CX              ;volta x(2).
09AC:0150 90          NOP
09AC:0151 58          POP   AX              ;volta x(1).
09AC:0152 90          NOP
09AC:0153 29C8        SUB   AX,CX           ;x(1)-x(2).
09AC:0155 BD0100      MOV   BP,0001         ;passa para BP o numero 1.
09AC:0158 E81500      CALL  0170
09AC:015B 90          NOP
09AC:015C 46          INC   SI
09AC:015D 46          INC   SI
09AC:015E 75B1        JNZ   0111            ;fim do segmento fonte.
09AC:0160 81C200      ADD   DX,1000
09AC:0164 81FA00      CMP   DX,4000         ;compare se acabou a fonte.
09AC:0168 75A4        JNZ   010E            ;segue avante.
09AC:016A 1F          POP   DS
09AC:016B 90          NOP
09AC:016C EB35        JMP   01A3            ;inicia a 2 etapa
09AC:0170 1E          PUSH  DS              ;salva segmento fonte.
09AC:0171 90          NOP
09AC:0172 8EDB        MOV   DS,BX           ;traz segmento destino.
09AC:0174 90          NOP
09AC:0175 6EB1        MOV   [DI],AX         ;guarda soma.
```

Programa 4 - TRC4UH.COM

```
09AC:0177 83FD01      CMP   BP,+01
09AC:017A 7410        JZ    018C          ;pula se é verdade,BP = 1,p/ linha 18C.
09AC:017C 47          INC   DI
09AC:017D 47          INC   DI
09AC:017E 83FF00      CMP   DI,+00        ;fim do segmento destino?
09AC:0181 7403        JZ    0186
09AC:0183 1F          POP   DS            ;volta segmento fonte.
09AC:0184 90          NOP
09AC:0185 73          RET                 ;retorna para quem chamou.
09AC:0186 81C300      ADD   BX,1000
09AC:018A EBF7        JMP   0183
09AC:018C 47          INC   DI
09AC:018D 47          INC   DI
09AC:018E BD0000      MOV   BP,0000
09AC:0191 EBE9        JMP   017C
09AC:0193 1E          PUSH  DS
09AC:0194 90          NOP
09AC:0195 E8DB        MOV   DS,BX
09AC:0197 90          NOP
09AC:0198 83C704      ADD   DI,+04
09AC:019B 8905        MOV   [DI],AX       ;grava na posição de memória apontada
09AC:019D 83EF04      SUB   DI,+04        ;pelo indicador de destino DI.
09AC:01A0 1F          POP   DS
09AC:01A1 90          NOP
09AC:01A2 C3          RET
09AC:01A3 1E          PUSH  DS
09AC:01A4 90          NOP
09AC:01A5 BE0000      MOV   SI,0000       ;reinicialize os ponteiros.
09AC:01A8 BF0000      MOV   DI,0000
09AC:01AB BA0060      MOV   DX,6000       ;fonte.
09AC:01AE BB0020      MOV   BX,2000       ;destino.
09AC:01B1 8EDA        MOV   DS,DX
09AC:01B3 90          NOP
09AC:01B4 52          PUSH  DX
09AC:01B5 90          NOP
09AC:01B6 90          NOP
09AC:01B7 90          NOP
09AC:01B8 8B04        MOV   AX,[SI]       ;carrega x(0) + x(3).
09AC:01BA 8B4C02      MOV   CX,[SI+02]    ;carrega x(1) + x(2).
09AC:01BD 90          NOP
09AC:01BE 50          PUSH  AX            ;salva p/ depois.
09AC:01BF 90          NOP
09AC:01C0 51          PUSH  CX            ;salva p/ depois.
09AC:01C1 90          NOP
09AC:01C2 50          PUSH  BX            ;salva destino.
09AC:01C3 90          NOP
09AC:01C4 BB0800      MOV   BX,0008       ;coloca o numero A = 8 em BX.
09AC:01C7 01C8        ADD   AX,CX         ;obtém soma.
09AC:01C9 F7EB        IMUL  BX            ;multiplica,no formato inteiro com
09AC:01CB 5B          POP   BX            ;sinal,pelo A e o result. fica em AX.
09AC:01CC 90          NOP
09AC:01CD E86000      CALL  0230          ;vai guardar.
09AC:01D0 90          NOP
09AC:01D1 59          POP   CX
09AC:01D2 90          NOP
09AC:01D3 58          POP   AX
09AC:01D4 90          NOP
09AC:01D5 53          PUSH  BX
09AC:01D6 90          NOP
09AC:01D7 BB0800      MOV   BX,0008       ;A' modificado para
09AC:01DA 29C8        SUB   AX,CX         ;obter @(2).
09AC:01DC F7EB        IMUL  BX
09AC:01DE 58          POP   BX
09AC:01DF 90          NOP
09AC:01E0 E84D00      CALL  0230          ;guarda @(2).
09AC:01E3 90          NOP
09AC:01E4 53          PUSH  BX
```

Programa 4 - TRC4UH.COM (continuação)

```
09AC:01E5 90        NOP
09AC:01E6 BB0400    MOV   BX,0004          ;fator B' modificado.
09AC:01E9 8B440A    MOV   AX,[SI+04]
09AC:01EC F7EB      IMUL  BX
09AC:01EE 89C1      MOV   CX,AX
09AC:01F0 BB0A00    MOV   BX,000A          ;fator C' modificado.
09AC:01F3 8B4406    MOV   AX,[SI+06]
09AC:01F6 F7EB      IMUL  BX
09AC:01F8 5B        POP   BX
09AC:01F9 01C8      ADD   AX,CX
09AC:01FB E83200    CALL  0230             ;guarda @(1).
09AC:01FE 53        PUSH  BX
09AC:01FF 90        NOP
09AC:0200 BBF6FF    MOV   BX,FFF6          ;fator C' negativo (-10).
09AC:0203 8B4404    MOV   AX,[SI+04]
09AC:0206 F7EB      IMUL  BX
09AC:0208 89C1      MOV   CX,AX
09AC:020A BB0400    MOV   BX,0004
09AC:020D 8B4406    MOV   AX,[SI+06]
09AC:0210 F71B      IMUL  BX
09AC:0212 01C8      ADD   AX,CX
09AC:0214 5B        POP   BX
09AC:0215 90        NOP
09AC:0216 E81700    CALL  0230             ;guarda @(3).
09AC:0219 90        NOP
09AC:021A 83C608    ADD   SI,+08
09AC:021D 7599      JNZ   01B8
09AC:021F 5A        POP   DX
09AC:0220 90        NOP
09AC:0221 81C200    ADD   DX,1000
09AC:0225 81FA00    CMP   DX,A000
09AC:0229 7586      JNZ   01B1
09AC:022B 53        POP   DS
09AC:022C 90        NOP
09AC:022D CD20      INT   20               ;fim do programa.
09AC:022F 90        NOP
09AC:0230 58        PUSH  DS               ;salva fonte.
09AC:0231 90        NOP
09AC:0232 8EDB      MOV   DS,BX            ;traz o destino.
09AC:0234 90        NOP
09AC:0235 58        PUSH  CX               ;salva x(1)+x(2).
09AC:0236 EB10      JMP   0248
09AC:0238 59        POP   CX               ;volta x(1)+x(2).
09AC:0239 8905      MOV   [DI],AX          ;guarda @(i).
09AC:023B 47        INC   DI
09AC:023C 47        INC   DI
09AC:023D 7403      JZ    0242             ;fim do segmento destino?
09AC:023F 1F        POP   DS               ;volta segmento fonte.
09AC:0240 90        NOP
09AC:0241 C3        RET
09AC:0242 81C300    ADD   BX,1000          ;atravessa segmento de destino.
09AC:0246 EBF7      JMP   023F
09AC:0248 B103      MOV   CL,03            ;o divisor e 8.
09AC:024A 050000    ADD   AX,0000          ;prepara teste de sinal.
09AC:024D 7811      JS    0260             ;pula,se o conteudo do Hx e negativo,
09AC:024F D3E8      SHR   AX,CL            ;para linha 260 e divide por 8.
09AC:0251 7202      JC    0255             ;pula se tem "carry".
09AC:0253 EBE3      JMP   0238
09AC:0255 050100    ADD   AX,0001          ;vai um.
09AC:0258 EBDE      JMP   0238

09AC:0260 F7D8      NEG   AX               ;torna positivo.
09AC:0262 D3E8      SHR   AX,CL            ;divide AX por 8.
09AC:0264 7204      JC    026A             ;pula se ha "carry" p/ linha 26A.
09AC:0266 F7D8      NEG   AX               ;restaura o sinal negativo.
09AC:0268 EBCE      JMP   0238
09AC:026A 050100    ADD   AX,0001          ;vai um.
09AC:026D EBF7      JMP   0266
```

Programa 4 - TRC4UH.COM (continuação)

III.6 - MCP Diferencial

Nesta seção, um sistema MCPD, já comentado na seção II.6 do capítulo anterior, é programado a fim de descorrelacionar os EI's ou coeficientes nas direções horizontal e vertical. Resultados para o preditor planar [30] podem ser também obtidos. Na verdade, uma série de programas foi elaborada. Eles podem ser aplicados nos arquivos originais das imagens, como também nos arquivos dos elementos transformados (coeficientes). Em alguns casos, o MCPD é utilizado também como parte da codificação híbrida já vista anteriormente.

Neste trabalho, como já frisado, há vários programas que calculam o erro de predição ε, ou seja, as diferenças, em modo horizontal, vertical ou planar. O programa 5, MCPDVPRE.COM, é um exemplo significativo do elenco mencionado. Ele se refere ao MCPD na direção vertical (amostra prévia), usando os EI's originais, com aplicação pré-transformada, ou seja, antes de uma transformada. O programa 5, basicamente, lê um byte de uma linha da imagem dos segmentos 2000: e 3000: e subtrai dele o byte correspondente da linha anterior (ver as linhas 120 a 12F). As diferenças ε são armazenadas nos segmentos 6000:, 7000: 8000: e 9000: como mostra a subrotina 300 chamada na linha 131. Feito isso, o programa é encerrado na linha 192 (INT 20), e as diferenças são utilizadas então para o cálculo da entropia, histogramas, transformadas, etc.

O ponto importante a notar no MCPDVPRE.COM, e que não apareceu nos programas anteriores é o fato de que as diferenças são feitas entre as linhas. Isto significa, precisamente, passar de um segmento para outro novo, com a cauda do MCPD ainda no antigo. Em outras palavras, a última linha de um segmento é subtraída, byte a byte, da primeira linha do segmento novo. Tal procedimento precisa de um tratamento específico, como mostrado no trecho compreendido entre as linhas 200 e 233. Resultados obtidos para este caso, bem como para outros tipos de MCPD, serão mostrados e discutidos no capítulo IV deste trabalho.

```
09AC:0100 1E        PUSH  DS              ;salva o estado do segmento de dados.
09AC:0101 90        NOP
09AC:0102 BB0000    MOV   BX,0000         ;inicialize registros.
09AC:0105 BD0000    MOV   BP,0000
09AC:0108 BE0002    MOV   SI,0200         ;aponta a 2 linha de EI'S.
09AC:010B BF0000    MOV   DI,0000
09AC:010E B90060    MOV   CX,6000
09AC:0111 BA0020    MOV   DX,2000         ;segmento-fonte, pois
09AC:0114 8EDA      MOV   DS,DX           ;os EI'S estão em 2000:
09AC:0116 90        NOP
09AC:0117 EB07      JMP   0120

09AC:0120 8B04      MOV   AX,[SI]         ;lê byte atual.
09AC:0122 25FF00    AND   AX,00FF
09AC:0125 812C00    SUB   SI,0200         ;volta à linha anterior (1).
09AC:0129 8B1C      MOV   BX,[SI]
09AC:012B 81E3FF    AND   BX,00FF         ;deixa byte em AL.
09AC:012F 29D8      SUB   AX,BX           ;sinal de erro E.
09AC:0131 E8CC01    CALL  0300            ;chama subrotina p/ armazenar.
09AC:0134 81C601    ADD   SI,0201         ;aponta o próximo.
09AC:0138 83FE00    CMP   SI,+00
09AC:013B 75E3      JNZ   0120            ;faz de novo.
09AC:013D 81C200    ADD   DX,1000         ;passa para próximo segmento (3000:).
09AC:0141 81FA00    CMP   DX,4000         ;termina segmento fonte?
09AC:0145 7449      JZ    0190
09AC:0147 E9B600    JMP   0200

09AC:0200 BD0000    MOV   BP,0000         ;zera contador.
09AC:0203 8EDA      MOV   DS,DX           ;segmento do minuendo.
09AC:0205 90        NOP
09AC:0206 1E        PUSH  DS              ;salva para depois.
09AC:0207 90        NOP
09AC:0208 8B04      MOV   AX,[SI]
09AC:020A 25FF00    AND   AX,00FF         ;minuendo.
09AC:020D 81EE00    SUB   SI,0200
09AC:0211 81EA00    SUB   DX,1000
09AC:0215 8EDA      MOV   DS,DX           ;segmento do subtraendo.
09AC:0217 90        NOP
09AC:0218 8B1C      MOV   BX,[SI]
09AC:021A 81E3FF    AND   BX,00FF         ;subtraendo
09AC:021E 29D8      SUB   AX,BX
09AC:0220 E8DD00    CALL  0300            ;vai guardar E.
09AC:0223 5A        POP   DX              ;volta segmento novo.
09AC:0224 90        NOP
09AC:0225 81C601    ADD   SI,0201
09AC:0229 45        INC   BP
09AC:022A 81FD00    CMP   BP,0200         ;fim da 1 linha de 512 bytes?
09AC:022E 75D3      JNZ   0203
09AC:0230 8EDA      MOV   DS,DX           ;a cauda também já atravessou?
09AC:0232 90        NOP
09AC:0233 E9EAFE    JMP   0120            ;vai normal.

09AC:0190 1E        POP   DS              ;volta seg. de dados ao est. anterior.
09AC:0191 90        NOP
09AC:0192 CD20      INT   20              ;termina tudo.

09AC:0300 1E        PUSH  DS
09AC:0301 90        NOP
09AC:0302 8ED9      MOV   DS,CX           ;destino é o segmento.
```

Programa 5 - MCPDVPRE.COM

```
09AC:0304 90          NOP
09AC:0305 8905        MOV   [DI],AX          ;leva E previsto ao seu lugar.
09AC:0307 47          INC   DI
09AC:0308 47          INC   DI
09AC:0309 83FF00      CMP   DI,+00           ;fim do segmento destino?
09AC:030C 7412        JZ    0320
09AC:030E 1F          POP   DS
09AC:030F 90          NOP
09AC:0310 C3          RET

09AC:0320 81C100      ADD   CX,1000          ;avança 64 Kbytes.
09AC:0324 EBE8        JMP   030E
```

Programa 5 - MCPDVPRE.COM (continuação)

Capítulo IV
Resultados Obtidos

IV.1 - Introdução

Nos capítulos anteriores foram detalhadas as técnicas utilizadas neste trabalho, bem como as ferramentas aqui empregadas no sentido de propiciar a codificação exata das imagens. Neste capítulo, os resultados são apresentados e comentados.

Como já mencionado na seção I.4 do capítulo I, cada uma das quatro imagens de teste analisadas contém três arquivos primários R, G e B. Destes, obtém-se um arquivo de luminância Y e três arquivos de diferenças de cor R-Y, G-Y e B-Y. Desta maneira, há 28 (vinte e oito) resultados diferentes para cada processo utilizado, ou seja, entropia, transformada, etc. Sendo calculadas duas transformadas (TRH e TRC) com vários tamanhos do bloco (N = 2, 4, 8,...) separadamente e também em conjunto, a quantidade de resultados (entropias, histogramas, etc) é muito extensa. Nas seções a seguir são mostrados os resultados mais importantes.

IV.2 - Entropia

Rodando o programa 1A (e consequentemente o programa 1B) da seção III.2 do capítulo anterior, obtém-se os resultados entrópicos mostrados na sub-seções seguintes.

IV.2.1 - Entropia das Imagens Originais

A tabela 4.1 mostra as entropias H [bits/EI] das imagens originais. Note que na primeira coluna da tabela 4.1 estão os nomes das quatro imagens utilizadas.

Original	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	7,18	7,28	7,32	7,33	5,59	4,41	5,79	21,78	18,71	15,79
SM02 (Sala)	6,80	6,60	6,46	6,72	4,45	3,51	5,04	19,86	16,21	13,00
SM04 (Zelda)	6,91	6,99	6,16	6,97	5,32	4,24	4,97	20,06	17,26	14,53
SM15 (Cozinha)	6,92	6,86	6,79	6,79	4,50	3,81	5,22	20,57	16,51	13,53
$\bar{H}$ [bits/EI]	6,95	6,93	6,68	6,95	4,97	3,99	5,26	20,57	17,17	14,21

Tabela 4.1 - Entropia H [bits/EI] das imagens originais

Cabe lembrar que a luminância Y é calculada dos arquivos R, G e B de acordo com a relação I.1 do capítulo I. Os arquivos de diferença de cor $C_R = R - Y$, $C_G = G - Y$ e $C_B = B - Y$ são também obtidos diretamente do arquivo Y, bem como dos respectivos primários R, G e B, sem nenhuma filtragem (ou constante) adicional.

Da tabela 4.1 é notável a tendência geral de que a imagem SM01 tem a maior entropia, para todos os arquivos. A imagem SM02, por sua vez, tem a menor entropia na maioria dos arquivos, excluindo alguns arquivos que contêm componentes azuis. As imagens SM04 e SM15 têm entropias intermediárias, alternando a segunda e a terceira posição.

Outra tendência que pode ser facilmente notada na tabela 4.1 é o fato de que a entropia da luminância Y é aproximadamente igual às entropias das componentes primárias R, G e B (ver entropia média $\overline{H}$ na tabela 4.1). É notável também observar que as entropias das componentes diferença de cor C_R, C_G e C_B são cerca de 2,1 bits (em média), menores do que as primárias R, G e B.

Neste ponto, é preciso ressaltar que é desejado transmitir (ou armazenar) uma imagem colorida. Sendo assim, é mais útil imaginar a codificação de todos os três arquivos, sejam eles primários ou derivados. Para isto, deve-se olhas as três colunas da direita da tabela 4.1. Considerando apenas os valores médios ($\overline{H}$), nota-se que uma redução de 3,40 bits/EI é conseguida, quando se usa as componentes derivadas Y + C_R + C_B ao invés das primárias R + G + B. No entanto, as derivadas C_R + C_G + C_B propiciam uma redução de 6,36 bits/EI em comparação com as mesmas primárias. Neste último caso pode-se obter o fator de compressão

$$FC1 = \frac{\overline{H}(R + G + B)}{\overline{H}(C_R + C_G + C_B)} = \frac{20,57}{14,21} = 1,45 \qquad (IV.1)$$

Desta consideração pode-ser obter a redução percentual do tempo de transmissão da imagem, aqui chamada RT. Tal parâmetro é dado então pela relação:

$$RT = (1 - 1/FC) \times 100\% \qquad (IV.2)$$

Para o caso já mencionado (ver FC1 da relação IV.1) tem-se:

$$RT1 = (1 - 1/1,45) \times 100\%$$

$$RT1 = 31,0\% \qquad (IV.3)$$

Deste cálculo conclui-se que há uma economia de 31% no tempo médio de transmissão das imagens, supondo uma mesma taxa de bits no canal. No caso de armazenamento, os arquivos utilizados (fita, disco, memória semicondutora, etc) necessitariam uma capacidade 31% menor.

Cabe notar neste ponto que as imagens no formato de diferença de cor possuem entropia menor. Esse fato é esperado pela maneira como os arquivos são calculados. A subtração de Y das componentes R, G e B resulta num descorrelacionamento parcial devido ao fato de que Y contém as três componentes nas proporções indicadas pela relação I.1, lembrando o funcionamento do MCP Diferencial operando com preditor não otimizado.

A redução RT1 obtida no tempo de transmissão (ver relação IV.3) é considerável. Ela foi conseguida analisando quatro imagens coloridas, como já mencionado. Se, por exemplo, uma imagem monocromática for colocada para esta codificação por sinais diferença de cor ($C_R + C_G + C_B$) um problema aparece. É sabido que a tríade acima é identicamente nula para esta situação. A luminância Y (níveis de cinza) existente na imagem não poderia então ser enviada. A codificação Y + C_R + C_B seria então uma solução para este caso. Entretanto, se esta alteração for feita, a transmissão de imagens coloridas, tais como as aqui investigadas será um tanto menos eficiente, ou seja, da tabela 4.1 obtém-se o fator de compressão:

$$FC2 = \frac{\bar{H}(R + G + B)}{\bar{H}(Y + C_R + C_B)} = \frac{20,57}{17,17} = 1,20 \qquad (IV.4)$$

A redução no tempo de transmissão RT é então menor, ou seja:

$$RT2 = (1 - 1/1,20) \times 100\%$$

$$RT2 = 16,7\% \qquad (IV.5)$$

No caso da imagem monocromática já discutido é importante observar que, se desejado, o codificador pode ser chaveado para transmitir apenas a luminância, para impedir que o sistema colorido se torne muito ineficiente. Para comprovar tal fato, basta referir aos valores de $\bar{H}$ (Y) da tabela 4.1.

Finalizando esta seção, e completando as observações quanto às complexidades entrópicas das quatro imagens testadas, pode-ser observar que a *praia* é realmente a mais rica em detalhes (ver tabela 4.1). Em outras palavras, o tempo de transmissão desta imagem é o maior. A imagem da *sala*, por outro lado, é a menos rica (menor H) e requer portanto o menor tempo de transmissão. Entre estes dois extremos estão as imagens Zelda e Cozinha.

IV.2.2 - Entropia da TRH

Como já mencionado anteriormente, a transformada unidimensional pode ser aplicada nas imagens originais, tanto na direção horizontal como na vertical. Aproveitando as duas aplicações mencionadas, resulta a transformação bidimensional. Os resultados mostrados nesta seção foram obtidos utilizando programas do PC, como frisado na seção III.4 do capítulo anterior. Depois da aplicação do programa adequado que calcula os coeficientes da TRH de um dado tamanho do bloco N, o programa 1A é chamado para calcular os respectivos valores da entropia H.

As tabelas 4.2, 4.3 e 4.4 mostram as entropias das imagens transformadas pela TRH, com tamanhos de blocos iguais a 2, 4 e 8, respectivamente. Estes resultados são obtidos aplicando a TRH Unidimensional Horizontal.

TRH2UH	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	7,31	7,52	7,42	7,45	5,64	4,29	5,70	22,25	18,79	15,63
SM02 (Sala)	6,92	6,76	6,75	6,76	4,72	3,66	5,47	20,43	16,95	13,85
SM04 (Zelda)	6,69	6,75	6,42	6,67	4,96	3,99	4,97	19,86	16,60	13,92
SM15 (Cozinha)	6,81	6,75	6,60	6,68	4,51	3,64	4,92	20,16	16,11	13,07
$\bar{H}$ [bits/El]	6,93	6,95	6,80	6,89	4,96	3,90	5,27	20,68	17,11	14,12

Tabela 4.2 - Entropia das imagens transformadas pela TRH Unidimensional Horizontal com N = 2

TRH4UH	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	7,16	7,42	7,26	7,30	5,39	4,47	5,87	21,84	18,56	15,73
SM02 (Sala)	6,72	6,57	6,58	6,52	4,83	3,89	5,44	19,87	16,79	14,16
SM04 (Zelda)	6,34	6,38	6,29	6,27	4,81	3,94	5,00	19,01	16,08	13,75
SM15 (Cozinha)	6,53	6,45	6,24	6,38	4,55	3,66	5,09	19,22	16,02	13,30
$\overline{H}$ [bits/El]	6,69	6,71	6,59	6,62	4,90	3,99	5,35	20,00	16,86	14,24

Tabela 4.3 - Entropia das imagens transformadas pela TRH Unidimensional Horizontal com N = 4

TRH8UH	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	7,37	7,64	7,46	7,51	5,98	4,85	6,24	22,47	19,73	17,07
SM02 (Sala)	6,82	6,68	6,83	6,61	5,12	4,34	5,85	20,33	17,58	15,31
SM04 (Zelda)	6,39	6,43	6,41	6,32	4,97	4,14	5,36	19,23	16,65	14,47
SM15 (Cozinha)	6,62	6,52	6,30	6,45	4,78	3,91	5,24	19,44	16,47	13,93
$\overline{H}$ [bits/El]	6,80	6,82	6,75	6,72	5,21	4,31	5,67	20,37	17,60	15,20

Tabela 4.4 - Entropia das imagens transformadas pela TRH Unidimensional Horizontal com N = 8

Das tabelas 4.2, 4.3 e 4.4 nota-se que a tendência geral de H é cair ligeiramente de N = 2 para N = 4 e subir para N = 8. Tal tendência indica que o melhor desempenho de TRH, na maioria da imagens, é conseguido com N = 4. Esta é uma razão suficiente para não fazer N = 16. Na verdade, comparando os valores de entropia H dessa sub-seção com aqueles obtidos na sub-seção anterior (originais) nota-se que nenhuma mudança apreciável é obtida, como já era esperado teoricamente. Entretanto, um fato notável é que, a *sala* (SM02), de maneira geral, passa a figurar em segundo lugar, no que se refere à riqueza entrópica dos coeficientes resultantes. Enquanto isso, a *praia* se mantém como a mais complexa. No caso de N = 4, a TRH obtém o melhor desempenho, devido, principalmente, aos bons resultados nas imagens mais complexas como a *praia*.

Encerrando esta sub-seção, pode calcular o FC para TRH4UH da seguinte maneira:

$$FC3 = \frac{\overline{H}(R + G + B)}{\overline{H}(C_R + C_G + C_B)} = \frac{20,57}{14,24} = 1,44 \qquad (IV.6)$$

onde $\overline{H}(R + G + B)$ é aquele da tabela 4.1.

O valor de RT para este caso é dado por:

$$RT3 = 30,6\% \qquad (IV.7)$$

No caso de utilizar a tríade Y + C_R + C_B os valores acima se alteram para:

$$FC4 = 1{,}22 \text{ e } RT4 = 18\% \qquad (IV.8)$$

IV.2.3 - Entropia do Sistema Híbrido TRH + MCPD

Nesta sub-seção os coeficientes da TRH (em palavras/16 bits) obtidos na sub-seção anterior são utilizados para aplicação de MCPD vertical. A entropia das diferenças resultante é calculada pelo programa 1A do capítulo anterior, como está mostrado nas tabelas 4.5, 4.6 e 4.7.

TRH2UH + MCPDV	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	4,91	5,16	4,88	4,95	4,29	3,28	4,48	14,95	13,72	12,05
SM02 (Sala)	4,26	4,10	4,45	3,98	3,34	2,55	4,40	12,81	11,72	10,29
SM04 (Zelda)	3,79	3,71	3,85	3,56	3,11	2,41	3,57	11,35	10,24	9,09
SM15 (Cozinha)	4,09	3,89	3,69	3,79	3,09	2,27	3,22	11,67	10,10	8,58
$\overline{H}$ [bits/El]	4,26	4,22	4,22	4,07	3,48	2,63	3,92	12,70	11,45	10,00

Tabela 4.5 - Entropia do sistema híbrido TRH + MCPD, com N = 2

TRH4UH + MCPDV	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	5,29	5,55	5,24	5,33	4,36	3,72	5,03	16,08	14,72	13,11
SM02 (Sala)	4,62	4,47	4,89	4,34	3,79	3,42	4,77	13,98	12,90	11,98
SM04 (Zelda)	4,13	4,02	4,30	3,85	3,56	2,87	4,04	12,45	11,45	10,47
SM15 (Cozinha)	4,39	4,22	4,00	4,11	3,55	2,74	3,83	12,61	11,49	10,12
$\overline{H}$ [bits/El]	4,61	4,57	4,61	4,41	3,82	3,19	4,42	13,78	12,64	11,42

Tabela 4.6 - Entropia do sistema híbrido TRH + MCPD, com N = 4

TRH8UH + MCPDV	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	5,79	6,05	5,73	5,83	5,19	4,21	5,54	17,57	16,56	14,94
SM02 (Sala)	5,09	4,95	5,51	4,80	4,28	3,58	5,26	15,54	14,34	13,12
SM04 (Zelda)	4,55	4,42	4,75	4,24	4,04	3,35	4,54	13,72	12,82	11,93
SM15 (Cozinha)	4,85	4,66	4,41	4,54	4,04	3,23	4,32	13,92	12,86	11,59
$\overline{H}$ [bits/EI]	5,07	5,02	5,10	4,85	4,39	3,59	4,92	15,19	14,15	12,90

Tabela 4.7 - Entropia do sistema híbrido TRH + MCPD, com N = 8

Das tabelas 4.5, 4.6 e 4.7 nota-se que a entropia da codificação híbrida com blocos de 2, 4 e 8, respectivamente, tem tendência crescente com o crescimento do tamanho do bloco N. Em outras palavras, o melhor desempenho híbrido, envolvendo TRH unidimensional e MCPD ocorre com menor tamanho do bloco, ou seja, N = 2. Lembrando o fato de que na sub-seção anterior o resultado de H [*bits*/EI] para N = 4 é menor do que para N = 2 (ver as tabelas 4.2 e 4.3), a conclusão mais adequada é que MCPD é mais eficientes com blocos menores de TRH.

É importante ressaltar aqui que o MCPD vertical utilizado é aquele mais simples, da amostra prévia. Desta forma, os coeficientes correspondentes de cada bloco são subtraídos para formar o sinal diferença entre as duas linhas adjacentes. Além disso, as diferenças de coeficientes obtidas não são identificadas, e a entropia global calculada é então considerada.

Para finalizar esta sub-seção, procede-se à obtenção dos parâmetros FC e RT.

$$FC5 = \frac{\overline{H}(R + G + B)}{\overline{H}(C_R + C_G + C_B)} = \frac{20,57}{10,00} = 2,06$$

$$RT5 = \left(1 - \frac{1}{FC5}\right) \times 100 = 51,5\% \qquad (IV.9)$$

Note que, mais uma vez, o valor de $\overline{H}(R + G + B)$ é aquele obtido dos arquivos originais (ver tabela 4.1).

Se, por outro lado, o resultado $\overline{H}(C_R + C_G + C_B)$ for substituído por aquele de $\overline{H}(Y + C_R + C_B)$ (ver tabela 4.5), obtém-se os novos valores abaixo:

$$FC6 = 1,80 \text{ e } RT6 = 44,4\% \qquad (IV.10)$$

Comparando-se estes resultados com aqueles obtidos na relação IV.8, nota-se uma melhoria considerável de desempenho, às custas deste sistema híbrido mais complexo.

IV.2.4 - Entropia da TRC

Num paralelo ao que foi feita para a TRH na sub-seção IV.2.2, calcula-se aqui as entropias da TRC para blocos de tamanho N = 2, 4 e 8, utilizando programas tais como o programa 4 da seção III.5 do capítulo III. Na verdade, a TRH e a TRC com N = 2 são idênticas, pois as matrizes são iguais (ver as relações II.4 e II.42 do capítulo II). Aliás, esse fato é válido para todas as transformadas mencionadas na seção II.3 do capítulo II. Sendo assim, os resultados de H [bits/EI] mostrados nas tabelas 4.2 e 4.5 são válidos também para TRC e híbrido TRC + MCPD com N = 2, respectivamente. As tabelas 4.8 e 4.9 a seguir mostram as entropias para TRC de N = 4 e N = 8, respectivamente.

TRC4UH	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	6,94	7,19	7,01	7,06	5,58	4,47	5,76	21,14	18,40	15,81
SM02 (Sala)	6,53	6,38	6,49	6,31	4,78	3,65	5,40	19,40	16,49	13,83
SM04 (Zelda)	6,13	6,13	6,16	6,00	4,77	3,91	5,11	18,42	15,88	13,79
SM15 (Cozinha)	6,34	6,24	6,03	6,15	4,52	3,65	5,63	18,61	16,30	13,80
$\overline{H}$ [bits/EI]	6,49	6,49	6,42	6,38	4,91	3,92	5,48	19,39	16,77	14,31

Tabela 4.8 - Entropia das imagens transformadas pela TRC unidimensional horizontal com N = 4

TRC8UH	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	6,95	7,22	6,99	7,05	5,76	4,68	6,02	21,16	18,83	16,46
SM02 (Sala)	6,46	6,59	6,46	6,22	4,98	4,14	5,76	19,51	16,96	14,88
SM04 (Zelda)	5,96	5,93	5,94	5,79	4,81	4,03	5,29	17,83	15,89	14,13
SM15 (Cozinha)	6,24	6,11	5,87	6,62	4,67	3,82	5,10	18,22	16,39	13,59
$\overline{H}$ [bits/EI]	6,40	6,46	6,32	6,42	5,06	4,17	5,54	19,18	17,02	14,77

Tabela 4.9 - Entropia das imagens transformadas pela TRC unidimensional horizontal com N = 8

Comparando as tabelas 4.2, 4.8 e 4.9, observa-se que $\overline{H}$ para componentes R, G e B é menor para N = 8, embora com desempenho próxima de N = 4. Para sinais de diferença de cor, em geral, o melhor resultado é obtido para N = 2, confirmando a mesma tendência verificada para Hadamard. Para casos que incluem a luminância Y, por outro lado, o desempenho melhor é obtido para N = 4, em média.

Em modo geral, o mérito de aumentar o tamanho do bloco de N = 4 para N = 8 é discutível, lembrando o aumento de complexidade de processamento requerido.

Da tabela 4.8, e considerando a codificação de Y + C_R + C_B o fator de compressão pode ser calculado:

$$FC7 = \frac{\overline{H}(R + G + B)}{\overline{H}(Y + C_R + C_B)} = \frac{20,57}{16,77} = 1,23$$

$$RT7 = (1 - 1/1,23) \times 100 = 18,7\% \qquad (IV.11)$$

Se a codificação de $C_R + C_G + C_B$ é considerada, então:

$$FC8 = \frac{20,57}{14,31} = 1,44 \text{ e } RT8 = 30,6\% \qquad (IV.12)$$

Entretanto, sobre este último resultado é preciso salientar que TRC ou TRH, para N = 2, produz um resultado ligeiramente melhor (ver tabela 4.2), ou seja:

$$FC9 = \frac{20,57}{14,12} = 1,46 \text{ e } RT9 = 31,5\% \qquad (IV.13)$$

IV.2.5 - Entropia do Sistema Híbrido TRC + MCPD

As tabelas 4.10 e 4.11 mostram os valores da entropia H para N = 4 e N = 8, respectivamente. Mais uma vez, o MCPD de amostra prévia já comentado anteriormente é utilizado.

TRC4UH + MCPDV	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	5,19	5,44	5,14	5,21	4,68	3,40	4,97	15,77	14,86	13,05
SM02 (Sala)	4,55	4,49	4,87	4,27	3,78	2,66	4,75	13,91	12,80	11,19
SM04 (Zelda)	4,10	3,99	4,29	3,82	3,56	2,89	4,13	12,38	11,51	10,58
SM15 (Cozinha)	4,33	4,17	3,97	4,05	3,54	2,76	3,82	12,47	11,41	10,12
$\overline{H}$ [bits/El]	4,54	4,52	4,57	4,34	3,89	2,93	4,42	13,63	12,65	11,24

Tabela 4.10 - Entropia do Sistema Híbrido TRC + MCPD, com N = 4

TRC8UH + MCPDV	R	G	B	Y	C_R	C_G	C_B	R+G+B	Y+C_R+C_B	C_R+C_G+C_B
SM01 (Praia)	5,60	5,85	5,53	5,61	5,12	4,16	5,42	16,98	16,15	14,70
SM02 (Sala)	4,95	4,91	5,33	4,66	4,23	3,50	5,22	15,19	14,11	12,95
SM04 (Zelda)	4,48	4,34	4,67	4,16	4,00	3,35	4,63	13,49	12,79	11,98
SM15 (Cozinha)	4,72	4,54	4,32	4,40	4,00	3,23	4,27	13,58	12,67	11,50
$\overline{H}$ [bits/El]	4,94	4,91	4,96	4,71	4,34	3,56	4,89	14,81	13,93	12,78

Tabela 4.11 - Entropia do Sistema Híbrido TRC + MCPD, com N = 8

Comparando esses resultados com aqueles obtidos no sistema híbrido TRH + MCPD (ver também tabelas 4.6 e 4.7) nota-se uma ligeira melhoria de desempenho do híbrido TRC + MCPD, para o mesmo N (4 ou 8). Tal melhoria parece mais acentuada com as imagens mais complexas, tal como SM01.

Quanto ao desempenho apenas do híbrido TRC + MCPD, o fato notável é que o aumento de N não traz vantagens. Sendo assim, o sistema de codificação híbrido para $N = 2$ mostra-se como sendo mais vantajoso, tanto do ponto de vista de entropia como de complexidade de implementação.

Encerrando essa seção de entropias, deve-se frisar que os melhores valores para FC e RT calculados para os híbridos TRC + MCPD e TRH + MCPD são aqueles mesmos já obtidos nas relações IV.9 e IV.10.

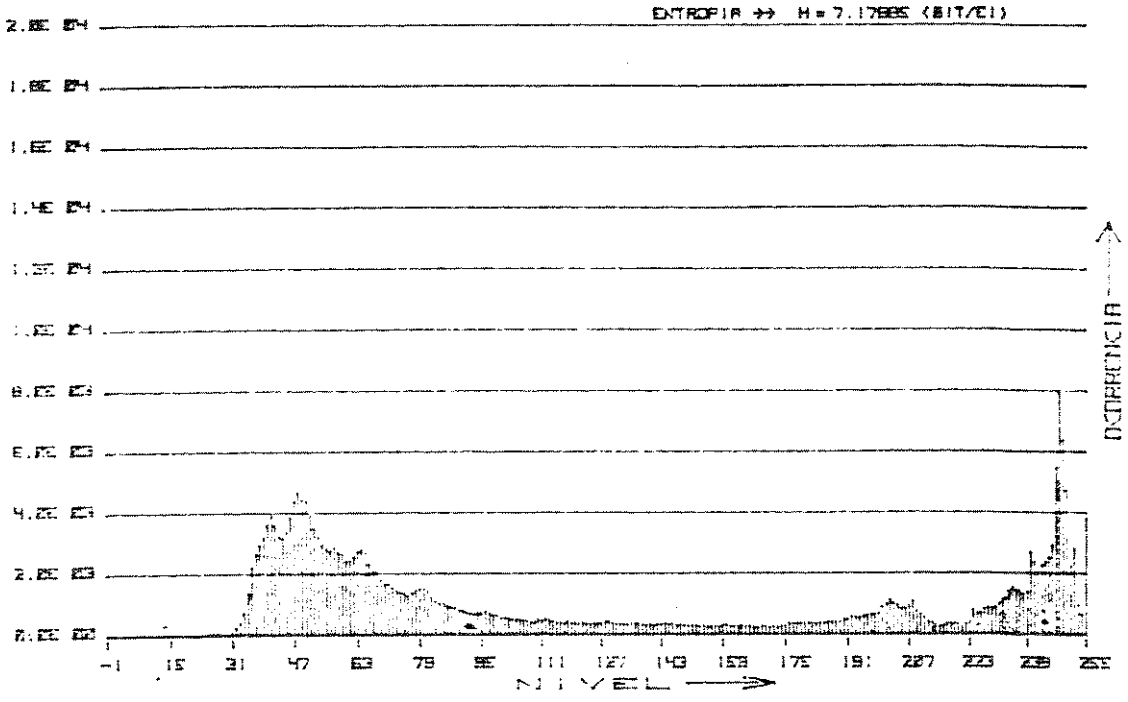
IV.3 - Histogramas

Como já visto na seção III.3 do capítulo anterior, o histograma pode dar uma idéia geral sobre os dados em questão. Quanto mais uniformemente espalhados estiverem os EI's, entre os níveis mínimo e máximo, mais complexo e lento será o processamento necessário. A falta de riqueza entrópica, ou seja, a existência de grandes áreas planas no histograma, por exemplo, indica a simplicidade da imagem e portanto do processamento de codificação.

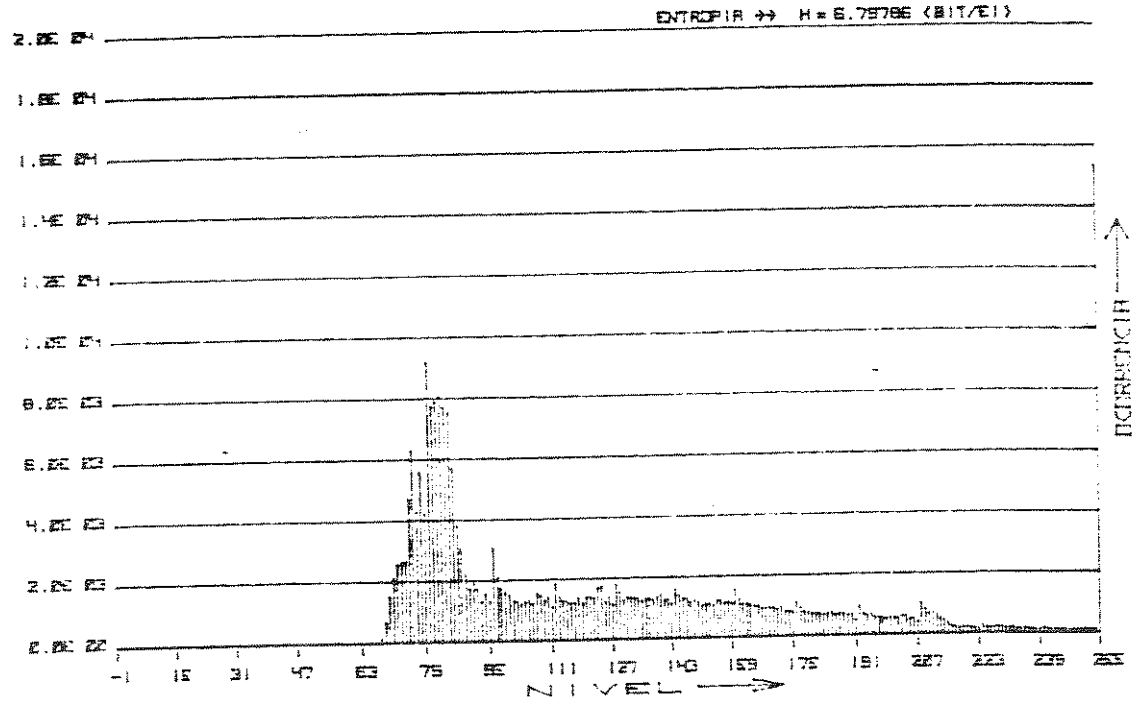
Nesta seção os histogramas são apresentados e analisados para os vários processos utilizados neste trabalho. Na verdade, apenas alguns histogramas representativos das imagens são mostrados, pois a quantidade obtida das várias combinações é muito grande.

IV.3.1 - Histogramas das Imagens Originais

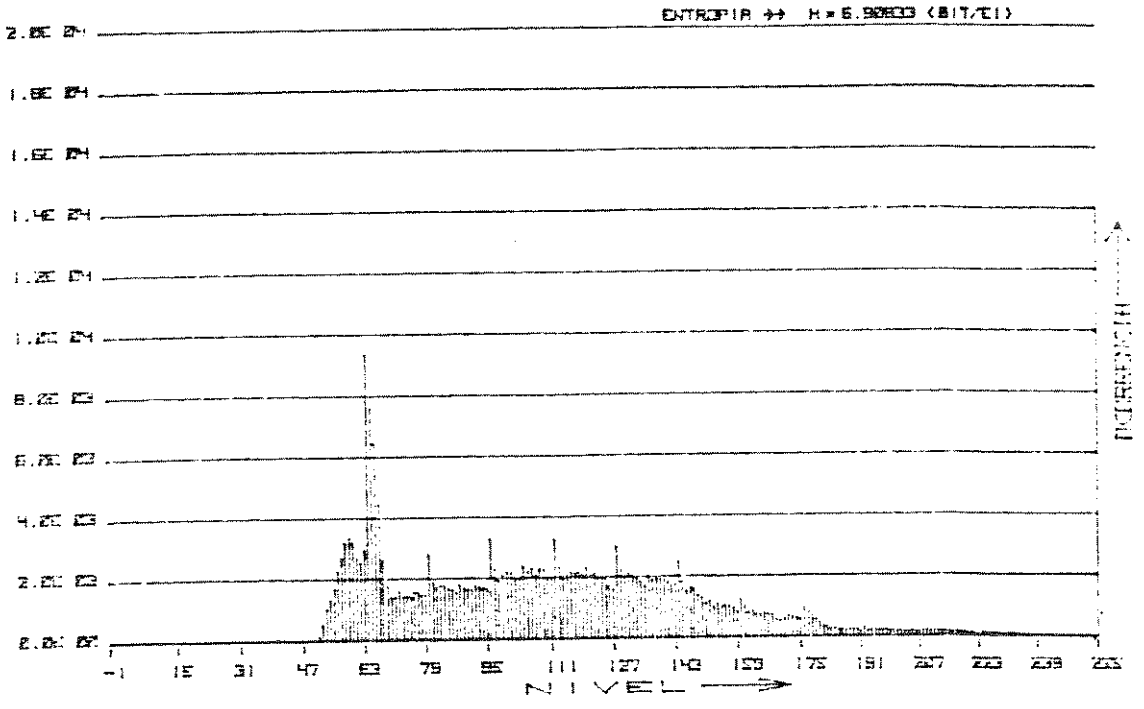
As figuras denominadas Histo IV.1, IV.2, IV.3 e IV.4 apresentam os histogramas das imagens originais de teste, SM01, SM02, SM04 e SM15, respectivamente, para a componente vermelha.



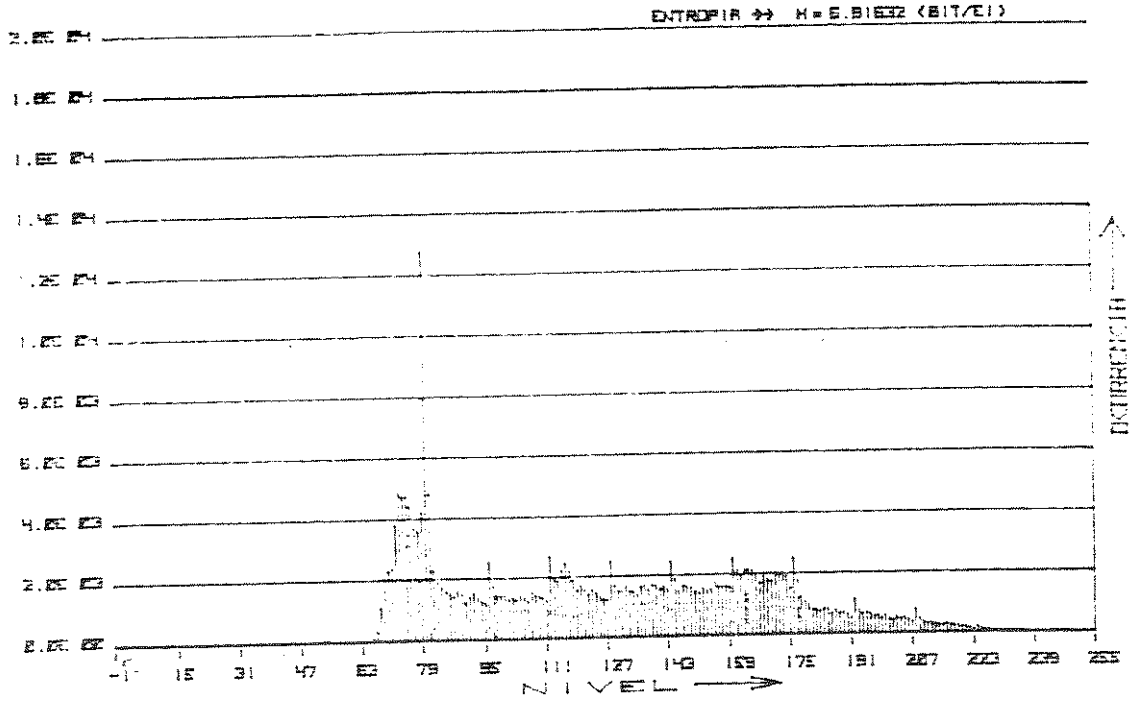
Histo IV.1 - Histograma da SM01, original, vermelha



Histo IV.2 - Histograma da SM02, original, vermelha



Histo IV.3 - Histograma da SM04, original, vermelha



Histo IV.4 - Histograma da SM15, original, vermelha

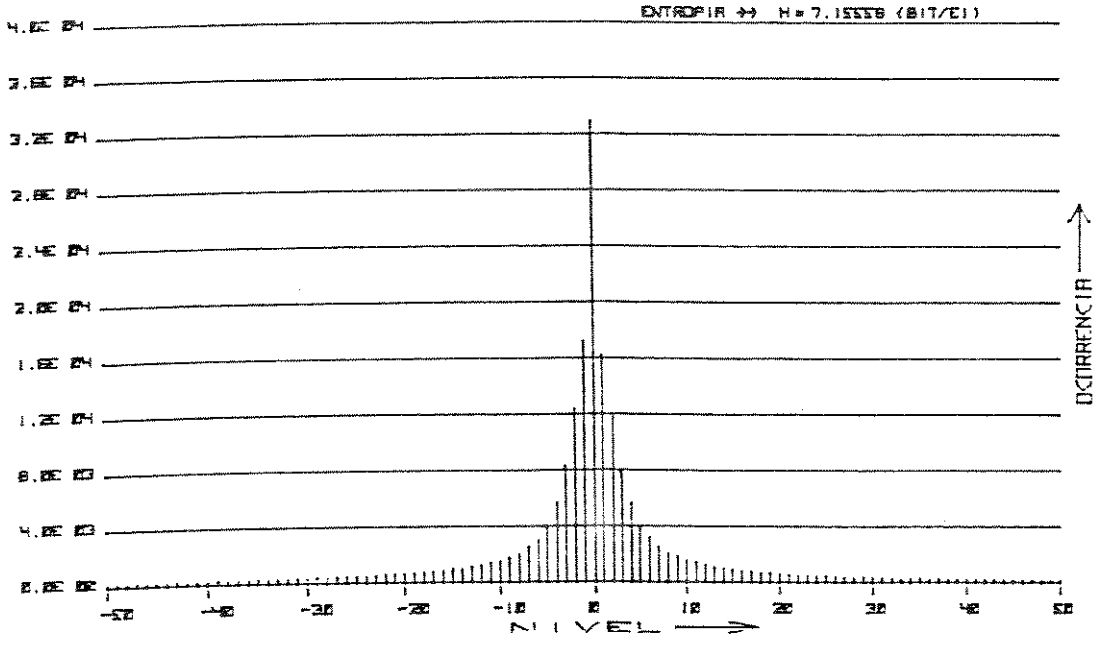
A aplicação dos programas da seção III.3 do capítulo III sobre as imagens originais SM01, SM02, SM04 e SM15 resulta nos quatro histogramas já referidos, respectivamente. Baseado somente nestes histogramas os seguintes pontos podem ser observados:

- Todas as imagens digitalizadas têm valores não negativos;
- Os níveis (horizontal em Histo IV.1 a IV.4) têm valores que varia de 0 a 255. Este fato vem da quantização das amostras por 8 bits ($2^8 = 256$), após a digitalização;
- Nas imagens SM02 e SM15, as ocorrências diferentes de zero começam aproximadamente no nível 66; na imagem SM04, começam perto do nível 50; na SM01, em 30;
- Enquanto nas três imagens SM02, SM04 e SM15 ocorre uma queda muito acentuada após, aproximadamente, o nível 200, na imagem SM01 ocorre o contrário. Para esta última existe uma área plana que se estende entre os níveis 100 e 200; a ocorrência começa a aumentar e chega a estabelecer um pico perto do nível 248;
- Ainda sobre a imagem SM01, comparando-a com as outras três imagens, ela tem quase todos os níveis possíveis, exceto aqueles entre 0 e 30. Baseando-se neste fato, pode-se concluir que esta imagem é mais complexa do que as outras três;
- As três imagens SM02, SM04 e SM15 embora tenham histogramas parecidos, possuem também alguns detalhes distintos. Na SM02, após os picos principais entre os níveis 70 e 85 (ver Histo IV.2), os valores das ocorrências variam suavemente, mas sempre abaixo do 2000. Esta variação é menos suave e, às vezes, maior de 2000 nas imagens SM04 e SM15. Essa observação, junto com a anterior, indica que SM02 (*sala*) é a imagem menos complexa.

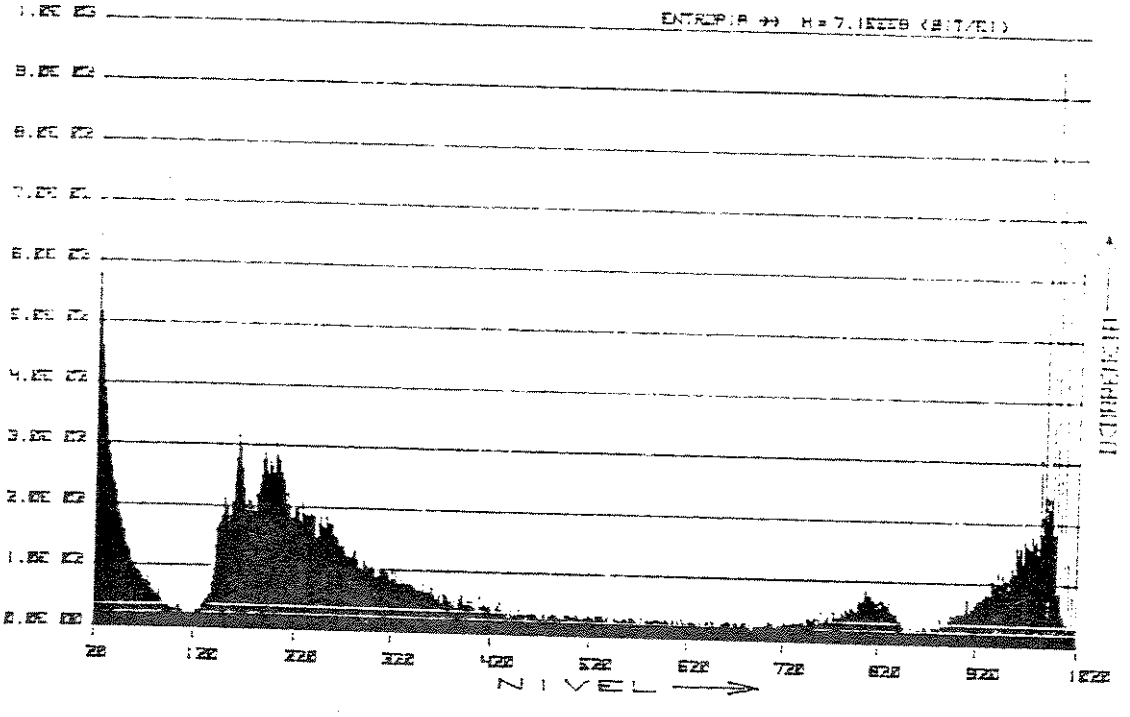
IV.3.2 - Histogramas das Imagens Processadas

Nesta sub-seção uma amostra dos vários histogramas das imagens processadas é apresentada. Eles são aqui denominados Histo IV.5 até Histo IV.10.

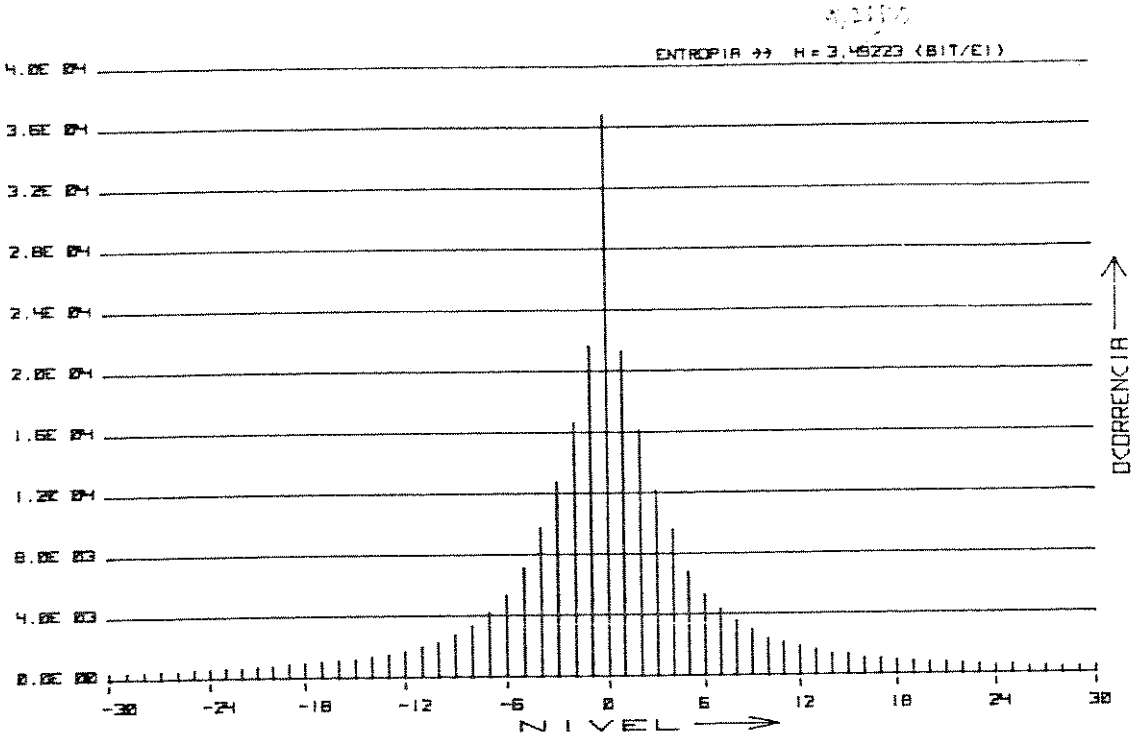
Histo IV.5 mostra o histograma da imagem SM01R (*R* para vermelha) transformada pela TRH unidimensional horizontal com N = 4 (detalhe na origem). Histo IV.6, por sua vez, mostra a mesma imagem com maiores detalhes (panorâmica) processada também pela TRH4UH. Histo IV.7 ilustra o histograma da *praia* (SM01R) processada pelo algoritmo híbrido TRH4UH + MCPDV. Os Histos IV.8, IV.9 e IV.10 mostram histogramas dos três casos anteriores, respectivamente, com TRC no lugar de TRH.



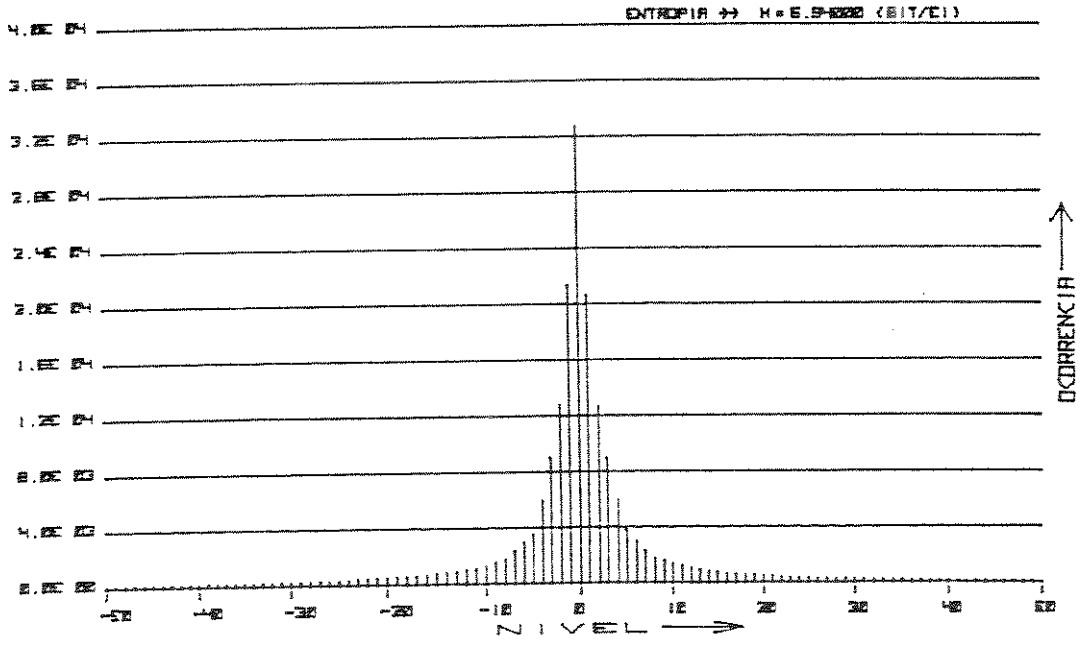
Histo IV.5 - Histograma da SM01R transformada pela TRH4UH (detalhe da origem)



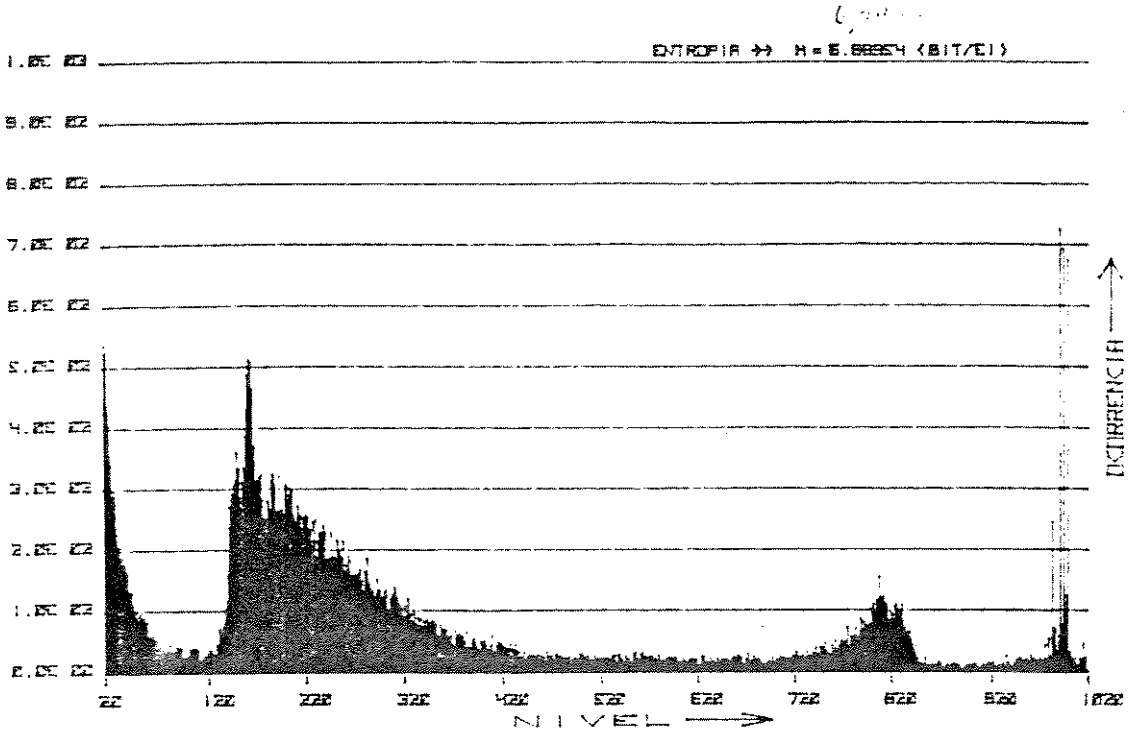
Histo IV.6 - Histograma da SM01R transformada pela TRH4UH (panorâmica)



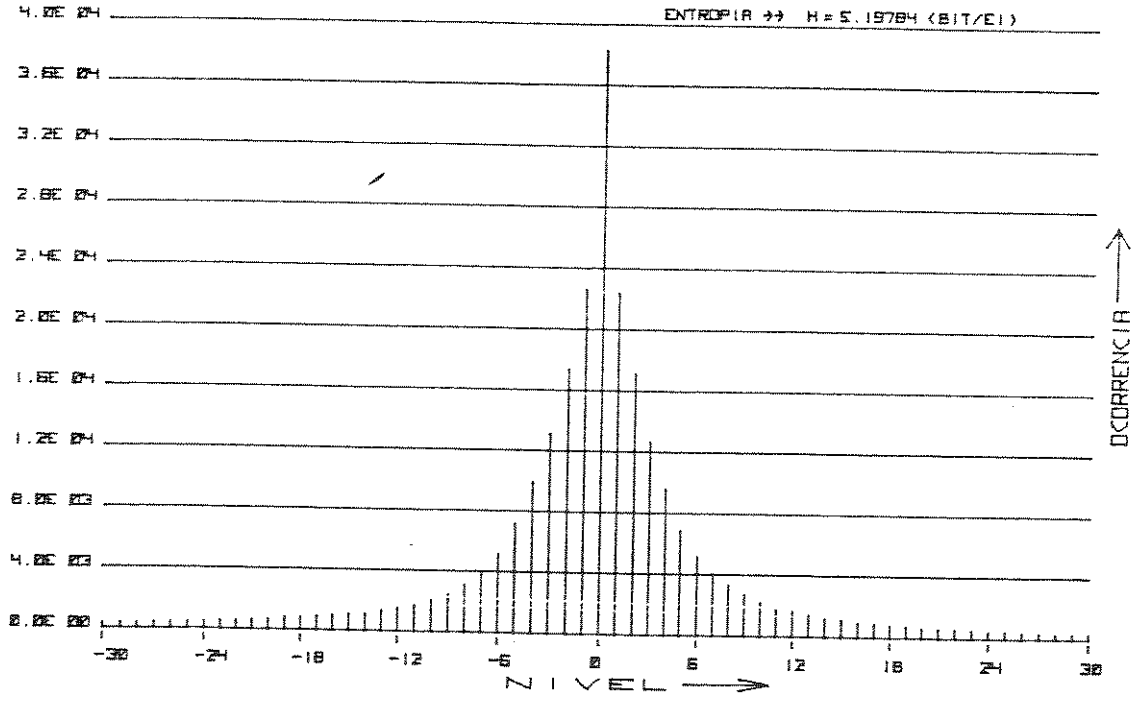
Histo IV.7 - Histograma da SM01R transformada pelo híbrido TRH4UH + MCPDV



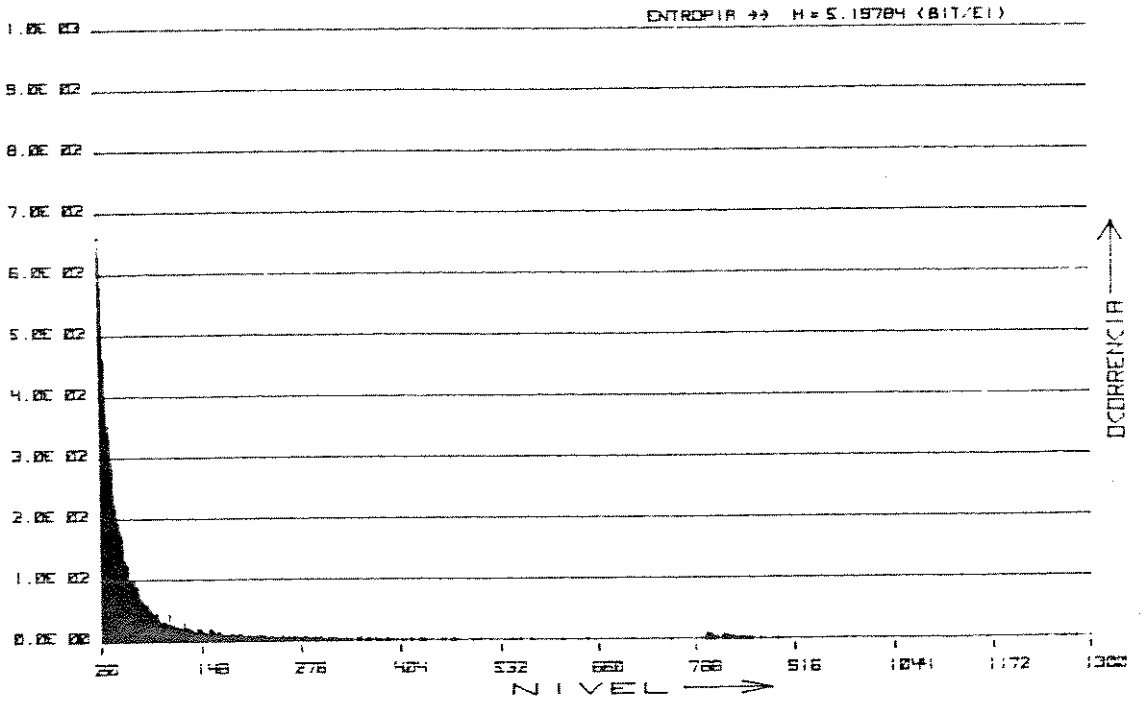
Histo IV.8 - Histograma da SM01R transformada pela TRC4UH (detalhe na origem)



Histo IV.9 - Histograma da SM01R transformada pela TRC4UH (panorâmica)



Histo IV.10 - Histograma da SM01R transformada pelo híbrido TRC4UH + MCPDV



Histo IV.11 - Histograma da SM01R transformada pelo híbrido TRC4UH + MCPDV (panorâmica)

Dos histogramas IV.5 até IV.10 desta sub-seção nota-se os seguintes fatos:

- Diferentemente dos histogramas das imagens originais, os histogramas das imagens processadas contêm níveis menores que zero. Na verdade, na maioria dos casos, há uma forte simetria especialmente nas proximidades do zero. Uma razão de tal simetria nos histogramas (ver Histo IV.5 e IV.8) para TRH e TRC é consequência da estrutura das matrizes de transformação (ver relações II.6 e II.43, por exemplo). Nestas duas transformadas, o primeiro termo (CC) e a soma dos N EI's; os N-1 termos restantes incluem sempre $N/2$ fatores positivos, bem como os mesmos $N/2$ fatores negativos. Tal estrutura produz, nas áreas planas das imagens, a distribuição simétrico com forte ocorrência dos níveis 0, ±1, ±2, etc;

- Os histogramas de TRH e TRC (puras) para a mesma imagem são muito semelhantes (ver também Histo IV.6 e Histo IV.9), ilustrando o fato de as entropias práticas serem muito próximas, como esperado teoricamente;

- Para os algoritmos híbridos (ver Histo IV.7 e Histo IV.10) o que se percebe rapidamente é que a simetria em torno do zero é reforçada. Observando apenas o nível zero, nota-se que a ocorrência é cerca de 37×10^3, para ambos os híbridos (ver histogramas IV.7 e IV.8). Nos cados de apenas TRH e TRC (ver histogramas IV.5 e IV.8) o nível zero ocorre 32×10^3 vezes, aproximadamente. Pode-se fazer considerações análogas para os níveis ±1, ±2, etc. Nota-se, por exemplo, que para o primeiro caso (híbridos), obtém-se cerca de 22×10^3 (para ±1), enquanto que com TRH e TRC obtém-se 10×10^3, em média. Esta reaglutinação adicional em torno de zero é propiciada pela ação do MCP Diferencial sobre os coeficientes da

TDH e da TDC. O histograma Histo IV.11 ilustra bem este fato, uma vez que os lóbulos laterais (concentrações laterais) longe da origem desaparecem, para compensar a aglutinação mencionada.

No próximo capítulo, os comentários finais sobre o trabalho são apresentados.

Capítulo V
Comentários e Conclusões Finais

V.1 - Introdução

Neste capítulo, vários aspectos do trabalho realizado são abordados. Para efeito de organização, divide-se este capítulo em duas seções, de tal modo que haja uma separação entre os aspectos do ferramental utilizado e os resultados obtidos do ponto de vista da codificação propriamente dita.

V.2 - Ferramental Utilizado

Como se nota após as exposições feitas no capítulo III, a implementação desta pesquisa foi estruturada em cima de um micro tipo PC-XT, também auxiliado por um *plotter* acoplado à calculadora de mesa HP-9820A. O uso desta estrutura permitiu um aprendizado muito interessante sobre as possibilidades que um microcomputador pode oferecer a problemas modernos de telecomunicações.

Na verdade, o que se imagina é que o micro, cada vez mais potente, pode sem empregado em comunicações visuais de imagens estáticas, através da linha telefônica. A linha telefônica convencional (atual) pode naturalmente trafegar sinais de voz e/ou de vídeo. Hoje já são fabricados no Brasil modems do tipo cartão-PC para taxas de até 2400 bps; no exterior, velocidades maiores já são disponíveis. Desta forma, quando o micro não está trabalhando em problemas genericamente mais comuns, ele se presta ao serviço de comunicações visuais. No futuro, a RDSI (Rede Digital de Serviços Integrados) permitirá o uso, em larga escala, de taxas de bits maiores do que aquelas acima citadas. Assim, taxas com p x 64 [kbits/s], para p = 1, 2, 3,..., ficarão disponíveis para o tráfego tanto de imagens estáticas como dinâmicas.

Na verdade, a transmissão de imagens de TV com faixa integral requer o uso de, pelo menos, cerca de 34 Mbits/s [6,9], em enlaces de fibra ótica ou rádio digital terrestre satélite. Tal taxa, como é sabido, só é conseguida às custas de uma tolerável degradação, escudada também em aspectos psico-físicos da visão humana [1,9]. A transmissão de imagens estáticas, por sua vez, traz uma condição singela, pelo fato de que o olho humano tem tempo disponível para se deter nos detalhes da cena em questão. Em outras palavras, ele tem tempo para se tornar mais crítico. Tais considerações levaram à investigação de alguns aspectos sobre a codificação exata feita nesta pesquisa.

A necessidade da codificação para redução de redundância de imangens, como já frisado neste trabalho, visa também o diminuição de meios de armazenamento tais como fita, disco ou RAM (*Random Access Memory*) semicondutora. Sobre este tópico, como bem ressaltado nas referências [2,7], o uso de uma codificação exata, como a entrópica por exemplo, é benvindo.

O emprego de um microcomputador na transmissão composta de imagens coloridas de TV PAL-M tem sido investigado [31,32]. Aqui, por outro lado, o enfoque central é dado na codificação de componentes, sejam elas primárias (R, G e B) ou derivadas (Y, C_R e C_B) e (C_R, C_G e C_B).

É interessante realçar o fato de que a comunicação visual onde o micro é o gerente, transmissor e/ou receptor, é factível agora; os PC's estão por aí, os modems e os cabos telefônicos também. Num futuro próximo, com o advento da RDSI, o canal será mais rápido. Sendo assim, pode-se imaginar um crescimento paralelo dos micros, em termos tanto de

velocidade de processamento, como de capacidade de memória. Este provável estado de coisas traz, por consequência, uma ligeira meditação; se o canal fica mais rápido, talvez não haja necessidade de codificações tão poderosas e complexas. Entretanto, numa situação como esta, a RDSI poderia ser utilizada por tais micros também na codificação de imagens com algum movimento (dinâmicas).

O uso das instruções do Assembler/PC aqui realizado traz uma possibilidade muito grande, quanto à quantidade de algoritmos que podem ser experimentados. Deste modo, a facilidade disponível é flexível, bastando para isto a troca de um programa ou disquete. Se, por outro lado, uma "máquina dedicada" (e não um micro) for desejada, o *hardware* pode ser amplamente desenvolvido e depurado na "estação de imagens" empregada; usando o largo conjunto de instruções visto no capítulo III, tais como SHR/SHL (*shift's* à direita e à esquerda), ADD/SUB (adiciona, subtrai), INC/DEC (incrementa, decrementa), etc, etc.

Aliás, sobre o exposto acima, é inegável a "proximidade" entre o *hardware* digital e os *op-codes* do Assembler/PC.

Finalizando, se a transmissão com um modem de 2400 bps for pensada, com transmissão assíncrona (1 *start bit* e 1 *stop bit*), palavras código de 8 bits poderiam ser enviadas a 240 Hz. Isto significa colocar um byte (ou palavra-código) na porta serial (National INS-8250) a cada 4 milisegundos, aproximadamente. Tal período, no micro em questão, possibilita executar cerca de 2000 instruções na UCP 8088. Neste caso, uma codificação bem complexa e poderosa com MCPD, transformadas, codificação compacta de Huffman, bem como codificação para combater error no canal poderia estar então sendo utilizada.

Concluindo esta seção, e a título de ilustração, deve-se frisar que a maioria dos programas aqui implementados (no Assembler/PC) tem tempo de execução variando entre 5 e 15 segundos.

V.3 - Processamento da Imagem

Encerrando este trabalho é importante relembrar que a codificação exata é uma técnica que depende primeiramente das estatísticas dos dados em questão; as imagens neste trabalho. Na literatura, a ênfase na codificação digital da imagem, é geralmente concentrada nas propriedades da visão humana, como já citado anteriormente. Em outras palavras, o processamento descarta ou deixa de considerar (de transmitir, por exemplo) aqueles elementos cuja falta não é percebida pela visão humana (redundância subjetiva [9]). Desta maneira, a taxa e o tempo de transmissão são reduzidos ou comprimidos. Neste trabalho, por outro lado, onde a codificação exata é o alvo principal, todos os elementos que fazem parte da informação primária contendo redundância estatística [9] são utilizados na compressão da imagem. Desta maneira, as imagens recebidas e/ou recuperadas são reconstruídas integralmente (100%) no processo inverso feito na decodificação.

Um bom exemplo da codificação exata (estatística) de imagens é a codificação híbrida (Transformada + MCPD) aqui investigada. Na verdade, há trabalhos baseados neste tópico [16,17], embora na linha de codificação não exata (subjetiva). Infelizmente, uma comparação entre os resultados não pode ser realizada.

Um resumo do que está feito nos quatro capítulos deste trabalho pode ser agora recordado.

Nos sete arquivos de cada uma das quatro imagens foram aplicadas duas transformadas distintas, TDH e TDC, utilizando algoritmos rápidos. Dividindo cada arquivo (total de 28) de tamanho 262144 em blocos de 2, 4 e 8 EI's e aplicando TRH e TRC horizontalmente, tem-se uma descorrelação de *pixels* próximos na direção mencionada. Complementando este procedimento híbrido, o MCPD é aplicado sobre os arquivos transformados na direção perpendicular, ou seja vertical.

As entropias e os histogramas foram então obtidos para os vários casos de arquivos originais e processados, investigados no capítulo IV.

Da relação IV.9 do capítulo IV o fator de compressão FC5 = 2,06 é válido para os sistema híbridos TRH + MCPDV e TRC + MCPDV para N = 2. A redução de tempo RT5 = 51,5% é também obtida para este caso de componentes ($C_R + C_G + C_B$). No caso de ($Y + G_R + C_B$), como visto na relação IV.10, o desempenho é ligeiramente inferior.

Os valores da relação IV.9 mencionada obtidos para FC e RT são os melhores resultados obtidos ao longo do trabalho. Eles significam, em resumo, que a quantidade de informação a ser transmitida ou armazenada é reduzida à metade. Essa redução é calculada tomando como referência uma média sobre as quatro imagens originais.

Com relação às duas transformadas TRH e TRC investigadas, uma informação importante é o fato de que TRH fornece entropias crescentes com o aumento de N. Tal acontece devido ao aumento do intervalo de variação dos níveis da fonte discreta original, espalhando assim o histograma original de ocorrência. Para TRC, por outro lado, com o mesmo N, o desempenho é ligeiramente melhor, em geral. Atribui-se esta melhoria ao fato de que os vetores-base cossenoidais se adequam melhor do que as bases quadradas, na decomposição das imagens aqui testadas.

Finalizando, este trabalho pode ser visto apenas como uma pequena contribuição a este vasto campo da codificação entrópica. É importante mencionar que vários outros testes são necessários para uma conclusão mais definitiva.

Bibliografia

[1] JAIN, A. K.; *Image Data Compression: A Review*. **Proc. IEEE**, vol. 69, nº3, mar. 1981.

[2] GONZALEZ, R. C. & WINTZ, P.; **Digital Image Processing**. Addison-Wesley Co., 2ª edição, 1987.

[3] ALONSO, A. O., TOLOSA, H. J. G., ALENS, N. & YANO, Y. **Tópicos sobre FAC-SIMILE**. Publ. FEC 18/80, RT/73 do contr. Telebrás - Unicamp, 139/76, jun. 1980.

[4] LATHI, B. P.; **Random Signals and Communication Theory**. International Textbook Company, 1968.

[5] IANO, Y. **Digitalização de Sinais de TV Através de um Sistema MCPD com Predição e Quantização.** Tese de doutorado, jan. 1986.

[6] ALONSO, O. A., YABU-UTI, J. B., ALENS, N. & YANO, Y.; **Digitalização de Sinais de TV**. Publ. FEC 44/80, RT/80 do contr. Telebrás - Unicamp, dez. 1980.

[7] JAYANT, N. S. & NOLL, P.; **Digital Coding of Waveform**. Prentice-Hall Inc., 1984.

[8] STAFFORD, R. H.; **Digital Television: Bandwidth Reduction and Communication Aspects**. John Wiley & Sons Inc. 1980.

[9] NETRAVALI, A. N. & HASKELL, B. G.; **Digital Pictures Representation and Compression**. Plenum Press, 1989.

[10] ABRAMSON, N.; **Information Theory and Coding**. McGraw-Hill, 1963.

[11] HAMMING, R. W.; **Coding and Information Theory**. 2ª edição, Prentice-Hall, 1986.

[12] PRATT, W. K., KANE, J. & ANDREWS, H. C.; *Hadamard Transform Image Coding*. **Proc. IEEE**, vol 57, nº1, jan. 1969.

[13] WALKER, R. & CLARKE, C. K. P.; **Walsh-Hadamard Transformation of Television Picture**. BBC Report, mar. 1974.

[14] CHEN, W., SMITH, C. H. & FRALICK, S. C.; *A Fast Computation Algorithm for the Discrete Cosine Transform*. **IEEE Trans.on Communication**, vol com-25, nº9, set. 1977.

[15] WALKER, R.; Hadamard Transformation: **A Real-time Transformer for Broadcast Standard P.C.M. Television**. BBC Report, fev. 1974.

[16] HABIBI, A.; *Hibrid Enconding of Pictorial Data*. **IEEE Trans. on Communication**, vol. com-22, nº 5, maio 1974.

[17] MURA, J. C.; **Compressão de Imagens de Satélite Meteorológico por Codificação Híbrida - Transformada Unidimensional do Cosseno e DPCM**. Tese de mestrado, ITA, fev. 1985.

[18] HUFFMAN, D. A.; *A Method for the Construction of Minimun Redundancy Codes*. **Proc. IRE**, vol. 40, nº 10, set. 1952.

[19] **Manual de Referência do BASIC/Nexus 2600**, Apêndice B, B.3, Scopus Tecnologia S/A.

[20] KELLEY, J. E. JR; **IBM PC e Seus Compatíveis**; Trad. e rev. de Rezende, J. D. e Perez, C. M., McGraw-Hill, Ltda, 1987.

[21] **Desenvolvimento sob SIM/DOS**; Itaú Tecnologia S/A, maio 1986.

[22] MORGAN, C. L. & WAITE, M.; **8086/8088 - Manual do Microprocessador de 16 bits**. Trad. e rev. de Ungnius, L. G. E. e Silva Neto, U. R., McGraw-Hill, 1988.

[23] **iAPX 86/20 - iAPX 88/20 - Numeric Data Processor, Preliminar**. pp 8-66, Component Data Catalog, Intel Corporation, 1981/1982.

[24] **Digital Signal Processing Applications with the TMS320 Family - Theory, Algorithms and Implementations**. pp 213-245, Texas Instruments Inc., 1986.

[25] BASIC - **Manual de Referência I-7901**. Itaú Tecnologia S/A, 1985.

[26] **Hewlett-Packard 9820A Calculator 11224A Peripheral Control II, Operating Manual.** Hewlett-Packard Company, 1972.

[27] **Hewlett-Packard 9862A Calculator Plotter, Peripheral Manual**. Hewlett-Packard Company, 1972.

[28] ROLLINS, D.; **IBM-PC 8088 Macro Assembler Programming**. Macmillan Publishing Company, 1985.

[29] ABEL, P.; **Assembler for the IBM-PC and PC-XT**. Reston Publishing Company, 1984.

[30] O'NEAL Jr, J. B. *Predictive Quantizing Systems (Differential Pulse Code Modulation) for the Transmission of Television Signals*. **B.S.T.J.**, vol 45, mai.-jun. 1966, pp. 689.

[31] ALONSO, A., IANO, Y., YABU-UTI, J. B. & MARTINI, L. C. **Viabilidade de Codificação e Transmissão Digital de Imagens Paradas de TV PAL-M na Rede Telefônica**. 2º Simpósio Brasileiro de Telecomunicações - Campinas/SP, set. 1984.

[32] ALONSO, A., IANO, Y., YABU-UTI, J. B., MARTINI, L. C. & ALENS, N. **Imagens Paradas de TV PAL-M na Linha Telefônica / Definição de um Sistema e Algoritmos de Geração**. 5º Simpósio Brasileiro de Telecomunicações - Campinas/SP, set. 1987.