Um Sistema de Compressão de Imagens Digitais

Carlos Fernando Assis Paniago
NMA - Núcleo de Monitoramento Ambiental - EMBRAPA
Av. Dr. Júlio Soares de Arruda, 803
13001-970 Campinas, SP
pan@nma.embrapa.br

Agma Juci Machado Traina
ICMSC - Instituto de Ciências Matemáticas de São Carlos - USP
Av. Dr. Carlos Botelho, 1465
13560-950 São Carlos, SP
agma@icmsc.sc.usp.br

ABSTRACT

This work describes the development of a compact and modular system for compression of digital images. The system was conceived to support medical and remote sensing images. The problem of the diversity of images is discussed, and the solution used by the system is presented.

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema modular e compacto de compressão de imagens digitais. O sistema foi projetado para suportar imagens médicas e de sensoreamento remoto. O problema da diversidade de formatos de imagens é discutido, e a solução encontrada por esse sistema é apresentado.

1. Introdução

1.1. Considerações Iniciais

Este trabalho apresenta técnicas de compactação de dados, principalmente as voltadas à compressão de imagens de satélites (sensoreamento remoto), com aplicações para a agricultura; e também para imagens médicas, com ênfase em imagens tomográficas obtidas a partir do tomógrafo que utiliza a técnica de Ressonância Magnética (RM) desenvolvido no Instituto de Física e Química de São Carlos - IFQSC [TAN_87].

Matematicamente uma imagem digital pode ser definida como um função bidimensional A(x,y) definida em uma certa região do plano:

Assim a imagem é definida num retângulo [0,r]x[0,s], e os valores tomados estão contidos no intervalo [0,t]. Ao valor A(x,y) no ponto (x,y) dá-se o nome de Nível de cinza [MAS_89].

Existem dois tipos básicos de compactação de informações:

1) compactação sem perda ("lossless") - que é usada para comprimir arquivos de dados, programas executáveis, banco de dados, etc. Nessa técnica, não se pode perder um único bit da informação previamente comprimida, pois essa perda causaria problemas enormes (por exemplo um programa executável ficaria errado ou um texto correto teria letras trocadas).

2) compactação com perda ("lossy") - que é usada para comprimir imagens e sons, onde a imagem reconstruída após a compactação não é exatamente a mesma de antes dessa, mas a perda de informações é mantida a uma quantidade que permita que o som reconstruído seja inteligível, bem como as imagens sejam (para o olho humano) extremamente parecidas com a imagem original, ou seja o erro imposto deve estar dentro de limites toleráveis pela aplicação.

O enfoque desse trabalho [PAN_94] é em técnicas onde as perdas sejam mínimas, pois a perda de informação numa tomografia, pode esconder alguma anormalidade que poderia prejudicar um diagnóstico ou mesmo mascarar anomalias que porventura existam. Mas para o armazenamento destas informações, pode-se perder algumas características, desde que sejam pequenas e imperceptíveis a olho nu. No caso de imagens de satélite, a perda de definição pode resultar em cálculos errôneos de áreas, já que as imagens dos satélites comerciais tem problemas com definições (que são na ordem de 30 x 30 metros ou 10 x 10 metros nos satélites mais modernos), mas para o armazenamento e posterior apresentação destas imagens, a perda controlada pode ser útil, já que as imagens ficam armazenadas em espaço muito reduzido.

Usualmente imagens digitais são apresentadas através de uma grande quantidade de dados (geralmente acomodadas em estruturas tipo matriz). Inúmeras aplicações que produzem imagens como resultado de seu processamento (sensoreamento remoto, tomografia, sistemas de inspeção, etc), geram sequências de imagens.

Imagem do tomógrafo do IFQSC
c81t5.pgm - 256 * 256

2. O Problema do Formato de Imagem

Normalmente as imagens tomográficas geradas no tomógrafo do IFQSC, tem o seguinte formato de arquivo (**.pac**):

os quatro primeiros bytes contém o tamanho do arquivo, definido como sendo dois bytes para o número de colunas e dois bytes para o número de linhas, colocados no arquivo na ordem de byte do computador tipo PC. Os bytes seguintes armazenam a imagem propriamente dita linha por linha. Já as imagens de satélite, tem um formato mais complicado, por exemplo no SITIM (Sistema de Tratamento de Imagens - desenvolvido pelo Instituto de Nacional de Pesquisas Espaciais - INPE), a imagem é dividida em um arquivo de cabeçalho (header) (que contém informações como data da passagem, coordenadas de latitude e longitude, hora da passagem, etc) e outro arquivo que contém somente a imagem, em ordem de linha.

Para ter um formato comum e podermos usar de visualizadores de domínio público, bem como ter a possibilidade de transformar o formato para qualquer um dos aceitos (Sun Icon file, X10 e X11 bitmap, MacPaint, CMU window manager, MGR, Group 3 FAX, GEM .img, face, Degas .pi3, andrew toolkit, FITS, Usenix FAceSaver, Lisp machine bit-array-file, GIF, IFF ILBM, PICT, XPM, PCX, TARGA, HP Paintjet, YUV, SUN raster file, ,TIFF além de outros formatos só para leitura ou gravação), resolvemos usar a família de formatos definidos por Jeff Poskanzer, chamado de PBM (Portable BitMap format), PGM (Portable GrayMap format) e PPM (Portable PixMap format) [POS91]. Esse formato é simples, e é definido para ser portátil através de equipamentos e sistemas operacionais diferentes. No nosso caso o formato escolhido é o **.PGM** (Portable GrayMap format) binário que pode representar arquivos preto e branco com até 256 niveis de cinza (o caso dos dois tipos de imagens trabalhadas).

Definimos então programas para converter do formato inicial para o formato PGM. O formato adotado usa a seguinte definição:

1. Os primeiros bytes tem de ser a cadeia "P5", depois de espaços (podem ser brancos, tabs ou novas-linhas - \n);
2. Pode-se ter linhas de comentário começado por '#' no primeiro caracter da linha (até o caracter nova-linha no fim);
3. Tem-se a dimensão da imagem como dois inteiros ASCII separados por espaços. O primeiro é a largura, ou o número de colunas, (y) e o segundo a quantidade de linhas (x);
4. Finalmente tem-se o máximo nível de cinza que aparece na imagem (de 1 a 255). Esse valor tem depois dele um espaço, que é normalmente o caracter nova-linha (\n);
5. Após a informação do máximo nível de cinza seguem-se os dados (em bytes) na ordem das linhas, ou seja, a primeira linha (todas as colunas) segue-se a segunda linha (todas as colunas), etc. até o fim.

```
P5
# imagem adquirida no tomografo do IFQSC
256 256
255
(bytes da imagem linha por linha)
```

Exemplo de um arquivo **.pgm**

Usando-se esse formato, permite-se a conversão automática para as dezenas de formatos suportados pela família de arquivos .PGM, bem como permite a visualização desses arquivos por programas de domínio público como o **xv**.

Dessa forma, para que o trabalho pudesse adequar-se mais amplamente a sistemas de imageamento para aplicações distintas, desenvolveu-se um módulo de conversão entre os diversos tipos de imagens utilizados transformando imagens **.pac** para **.pgm** e também arquivos **.i** (arquivos de imagem gerados no SITIM do INPE).

3. Codificação de Huffman

A codificação de Huffman é uma técnica de compressão estatística na qual há uma redução do tamanho médio do código usado para representar o alfabeto. O código de Huffman é uma solução ótima, quando todas as probabilidades de ocorrência de uma determinada letra for uma potência de 1/2 [HUF_52].

A codificação de Huffman cria código de tamanho variável para cada símbolo de entrada, sendo que os símbolos de entrada com maior probabilidade tem um código menor e os de maior probabilidade tem código maior.

Para fazer a árvore binária de codificação e decodificação do código de Huffman, usa-se uma técnica de baixo para cima ("bottom-up"), começando pelas folhas até atingir a raiz da árvore. Primeiramente ordena-se os códigos de entrada

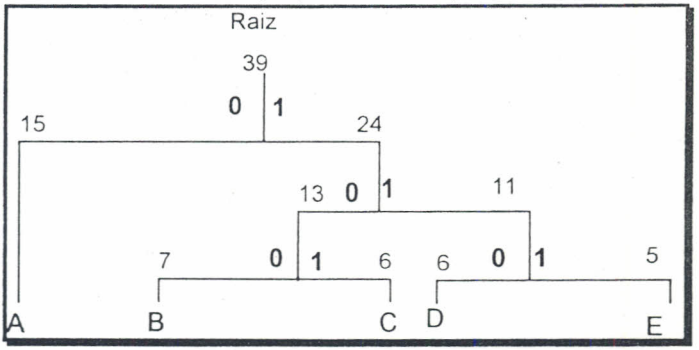
usando como peso a probabilidade de cada símbolo (conhecida a priori). Usa-se então o seguinte algoritmo:

- Ordena-se os símbolos por ordem do peso e coloca-os numa lista de nós livres;
- Os dois nós com menor peso são encontrados;
- Um nó pai desses dois nós é criado. O peso dele é a soma dos pesos dos nós filhos;
- O nó pai é adicionado na lista dos nós livres. Os 2 filhos são removidos da mesma lista;
- Um dos nós filhos é designado pelo caminho vindo do nó pai a ter o valor de decodificação arbitrado em 0, o outro filho fica com o valor 1;
- Execute os 4 últimos passos, repita-os até que só exista um nó livre. Esse é designado como a raiz da árvore binária.

Se tivéssemos a seguinte cadeia de símbolos com suas probabilidades.

A	B	C	D	E
15	7	6	6	5

A árvore binária usada nesse exemplo para a codificação de Huffman seria:



Percorrendo-se a árvore acima para cada símbolo na tabela, teria-se a seguinte codificação:

Símbolo	Huffman	Freq.	Info.	Bits Info.	Bits Huf.
A	0	15	1,38	20,68	15
B	100	7	2,48	17,35	21
C	101	6	2,70	16,20	18
D	110	6	2,70	16,20	18
E	111	5	2,96	14,82	15
TOTAL		39		82,25	87

Com a codificação de Huffman conseguiu-se 87 bits, numa mensagem de 82,25 bits (calculado pela fórmula de entropia) dois bits a menos que a codificação de Shannon-Fano, vista na seção anterior.

$$Número\_de\_bits = -\log_2(probabilidade)$$

Fórmula da Entropia

Como pode-se notar, o código de Huffman tem a propriedade de ter um único prefixo, então ele pode ser decodificado sem problemas de ambigüidade no momento da chegada dos dados no decodificador.

Um dos problemas da codificação de Huffman é que para decodificar, a tabela de entrada necessariamente precisa estar disponível, e para isto é necessário transmiti-la junto com os dados, o que aumenta o tamanho do arquivo compactado. Diversas técnicas tem sido usadas para manter essa informação mínima, usando por exemplo, escalonamento para colocar todos os códigos dentro de uma unidade de informação do tamanho de um caracter.

Outro problema é que o arquivo de dados deve ser lido pelo menos duas vezes: uma para calcular as probabilidades; e outra para fazer a codificação. Nesse caso outras técnicas podem ser usadas na codificação de Huffman para evitar esse problema, como por exemplo, construir a tabela interativamente tanto na compressão como na descompressão dos dados, mas lembrando-se que neste caso, usando uma tabela de Huffman adaptativa, o resultado não é igual ao anterior.

4. Compressão de Truncagem de Bloco - BTC

Esta técnica chamada de compressão de truncagem de bloco (BTC - Block Truncation Compression) é baseado em [KRU_92], e comprime imagens na proporção de 4:1 (exceto o cabeçalho) com perdas na imagem. A idéia central deste método é usar blocos de 4 por 4 pixels e manter o primeiro e o segundo momento do bloco. O primeiro momento é a média e o segundo momento é a média dos quadrados deste mesmo bloco.

Deve-se calcular para cada bloco as seguintes equações:

$$m_1 = \frac{1}{n}\sum_{i=0}^{n-1} p_i$$

$$m_2 = \frac{1}{n}\sum_{i=0}^{n-1} p_i^2$$

$$d = \sqrt{\frac{q}{n-q}}$$

$$\sigma = \sqrt{m_2 - m_1^2}$$

$$a = m_1 - d\sigma$$

$$b = m_1 + \frac{\sigma}{d}$$

Onde $\mathbf{m_1}$ é a média, $\mathbf{m_2}$ é a média dos quadrados, $\mathbf{p_i}$ são os elementos do bloco a serem analisados, **n** é o número de elementos no bloco (16 = 4 * 4) e **q** é o número de pixels no bloco que é maior que a média.

Após o cálculo destes elementos, calcula-se **a** e **b**, que serão colocados como resultado no arquivo de saída, seguido de 16 bits (2 bytes) dizendo que elemento é maior ou igual a média (com valor 1) ou menor que a média (com valor 0). Estes quatro bytes são gravados no arquivo de saída para cada bloco, o que perfaz a compressão de 16:4 ou seja 4:1.

Na descompressão lê-se os 4 bytes do bloco e analisa-se o mapa de bits. Para cada bit 1 coloca-se o valor de **b** e para os bits 0 o valor de **a**.

Apesar de ser uma compressão com perdas, a imagem gerada após a descompressão é bastante similar à original dependendo da imagem. Em alguns casos as bordas da imagem ficam serrilhadas, quando comparadas as imagens originais.

O formato desta imagem é definida a seguir:

Na primeira linha o cabeçalho **tP1**; copia então as duas linhas seguintes do formato .pgm (os comentários e a definição de colunas e linhas); a seguir vem os dados (4 bytes para cada bloco de 4*4 bytes).

```
tP1
# imagem adquirida no tomografo do IFQSC
256 256
(bytes da imagem compactadas por BTC, bloco a
bloco)
```

5. Joint Photographic Experts Group - JPEG

Um grupo criado pela CCITT, e pela ISO para padronizar compressão de imagens terminou uma proposta em 1991, e definiu o padrão de compressão de

imagens fotográficas [WAL_91]. O JPEG é um padrão para compressão de imagens coloridas ou em níveis de cinza, mas não comprime imagens de dois níveis (preto e branco) que é tratado por outro grupo chamado JBIG - Joint Bi-level Image Experts Group [GAI_93].

Existem diversos parâmetros no processo de compressão do JPEG. Ajustando esses parâmetros, podemos trocar qualidade de imagem reconstruída por tamanho, ou seja, se perdemos qualidade da imagem original, ganhamos mais espaço de armazenamento e se mantivermos qualidade, o ganho de armazenamento não é tão grande.

O JPEG define um algoritmo básico com perdas, mais extensões opcionais para codificação progressiva e hierárquica. Há também um modo de compressão sem perdas que tipicamente consegue razão de compressão de 2:1. A maioria dos softwares e hardwares disponíveis para a proposta do JPEG, usam somente o modelo básico.

Vamos apresentar (sem entrar em detalhes técnicos) a proposta do JPEG básica [GAI_93]:

1 - Transforme a imagem num espaço de cores adequado. Isto é uma operação nula em imagens em níveis de cinza o que é de interesse deste trabalho, mas para imagens coloridas é bom mudar da codificação RGB (vermelho, verde, azul) para um espaço YUV de luminância (Y) e crominância (UV). O componente luminância está agora em níveis de cinza e as duas outras dimensões, relativas a crominância, nos dão informações das cores.
2 - (Opcional) Deixe o nível de luminância inalterado e reduza a dimensão dos outros dois componentes na proporção 2:1 horizontalmente e 2:1 ou 1:1 verticalmente. Nos termos de JPEG isto é definido como amostragem 2h2v ou 2h1v. Essa compressão diminui o tamanho dos arquivos para arquivos coloridos.
3 - Agrupe os valores dos pixels para cada componente em blocos de 8x8. Transforme cada bloco de 8x8 através de uma transformada de cosenos discreta (DCT).
4 - Em cada bloco, divida cada uma das 64 componentes de frequências por um "coeficiente de quantização" (CQ) separado, e arredonde os valores para inteiro. Esse é o passo fundamental na perda de informação. Um CQ de 1 perde um mínimo de informação, CQ's maiores perdem sucessivamente mais informações.
5 - Codifique os coeficientes reduzidos usando codificação de Huffman ou codificação aritmética. Na prática usa-se a codificação de Huffman, pois a codificação aritmética usada no JPEG é patenteada e é necessário pagar taxas de licença.
6 - Coloque as informações iniciais (cabeçalhos) e grave os resultados. Em um arquivo universal JPEG, todos os parâmetros de compressão estão armazenados nos cabeçalhos. Para aplicações especializadas, é possível a omissão desses parâmetros, mas isto significa que o descompressor tem de conhecer os parâmetros do compressor.

5.1. Compressão Usando DCT

Uma técnica muito usada atualmente é a de usar o DCT (Discrete Cosine Transform), ou seja, transformada discreta dos cosenos. Isto deve-se ao fato que uma imagem transformada por DCT consegue ótima compressão e a transformada inversa, leva a uma imagem bastante próxima à original.

O método definido em JPEG usa DCT para conseguir comprimir uma imagem.

Vamos definir um método aproximado do JPEG, para conseguir imagens compactadas com fator de 4:1 fixo. Esta ideia baseia-se em usar blocos (4 por 4 ou 8 por 8) para comprimir a imagem. Para calcular DCT, usamos as seguintes fórmulas:

$$F[u,v]= \frac{1}{N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f[m,n]\cos(\frac{(2m+1)u\pi}{2N})\cos(\frac{(2n+1)v\pi}{2N})$$

$$f[m,n]= \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} c[u]c[v]\cos(\frac{(2m+1)u\pi}{2N})\cos(\frac{(2n+1)v\pi}{2N})$$

onde:

u, **v** são variáveis discretas de frequência (0, 1, 2, ..., N-1).
f[m,n] imagem N*N pixels (0, 1, 2, ..., N-1).
F[u,v] é o resultado do DCT.
Na função inversa **f chapéu**, as variáveis são:
m, **n** índices dos pixels resultantes (0, 1, 2, ..., N-1).
F[u,v] é a DCT (N*N) para obter o resultado inverso.
c[k] é definido como 1 se k=0, 2 se k=1, 2, 3, ..., N-1.

Para cada bloco na imagem (4*4 ou 8*8), calcula-se a matriz DCT. Ela tem uma propriedade interessante, que é possuir muitos zeros nos elementos inferiores da diagonal secundária. Portanto precisa-se armazenar apenas os elementos superiores a diagonal secundária da matriz. Além disto os elementos na diagonal superior são mais relevantes na reconstrução da imagem do que os elementos diagonais inferiores [EMB_91].

No caso 4*4 calcula-se a matriz cosenos (note-se que esta matriz pode ser calculada a priori). Usa-se o fato de que as equações acima podem ser representadas por:

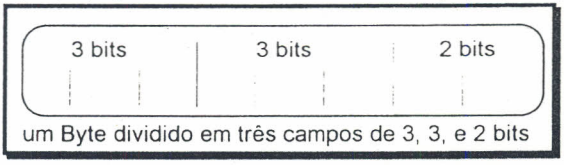
$$DCT\{A\} = \frac{CAC^T}{N^2}$$

$$c[u,n]= \cos(\frac{(2n+1)u\pi}{2N})$$

Após os cálculos a matriz DCT 4*4 é armazenada com menos bits para cada componente. Veja o mapa de bits usado nesta matriz:

8	3	3	2
3	3	2	0
3	3	0	0
2	0	0	0

Ou seja guarda-se o primeiro byte intacto [0,0] e depois grava-se três bytes compostos de 3, 3, e 2 bits cada um (ou 8 bits), usando os seguintes índices: [0,1], [0,2] e [0,3] no segundo byte, [1,0], [2,0] e [3,0] no terceiro byte, e [1,1], [2,1] e [1,2] no último byte. O restante da matriz é considerada zero. Para usar 3 bits os resultados são normalizados para o intervalo [-4, -3, ..., 3] e os de 2 bits para o intervalo [-2, -1, 0, 1].



um Byte dividido em três campos de 3, 3, e 2 bits

Para o decodificador 4*4 usa-se o processo inverso: ler a matriz de DCT armazenada, efetuando o contrário em relação ao armazenamento, e multiplicando pela matriz de cosenos transposta e cosenos, para obter o bloco, próximo ao original.

Apesar deste método conseguir compressão na ordem de 4:1 o resultado das imagens de satélite e de tomografia compactados, visualmente (ao nível de olho nu) não difere significativamente). Mas ao ampliarmos a imagem verifica-se efeitos de serrilhamento em alguns pontos.

No caso do bloco 8*8 a matriz de bits muda para:

8	8	4	4	4	4	4	4
8	8	4	4	4	4	0	0
4	4	4	4	0	0	0	0
4	4	4	4	0	0	0	0
4	4	0	0	0	0	0	0
4	4	0	0	0	0	0	0
4	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

Ou seja armazena-se ou bytes inteiros ou meio bytes. Mas alguns valores são escalados (divididos por um valor) antes de ser convertido para o arquivo de saída. A matriz de escalonamento é:

1	1	3	3	2	2	1	1
1	1	2	1	1	1	0	0
3	2	2	1	0	0	0	0
3	1	1	1	0	0	0	0
2	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0

Onde divide-se cada elemento da matriz original por esta matriz definida acima, exceto no caso dos zeros, aonde o resultado é zerado na matriz original. Na descompressão ao ler os dados baseados na matriz de bits, faz-se a multiplicação elemento a elemento com a matriz acima, antes de multiplicar pela matriz de cosenos transposta e cosenos.

O arquivo de saída é similar ao do BTC, só que na primeira linha a cadeia para o arquivo de saída usando DCT 4*4 é **tP2** e a cadeia para o arquivo de saída do DCT 8*8 é **tP3**.

```
tP2
# imagem adquirida no tomografo do IFQSC
256 256
(bytes da imagem compactadas por c4c, bloco a bloco)
```

Formato do arquivo gerado pela DCT 4*4

```
tP3
# imagem adquirida no tomografo do IFQSC
128 128
(bytes da imagem compactadas por c8c, bloco a bloco)
```

Formato do arquivo gerado pela DCT 8*8

6. Conclusão

O sistema de compressão de imagens foi desenvolvido modularmente de forma a poder ser integrado aos sistemas de sensoreamento remoto e imagens de tomografia.

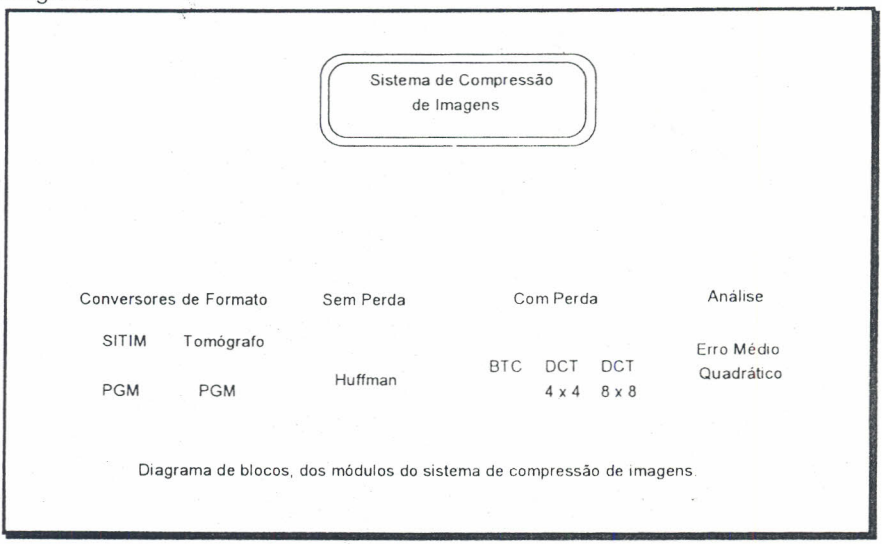


Diagrama de blocos, dos módulos do sistema de compressão de imagens.

Para obter altas taxas de compressão (de 4:1 ou mais) é necessário recorrer-se a técnicas de compressão com perdas na imagem. Apesar destas imagens reconstruídas não serem exatamente iguais às imagens originais, normalmente para o olho humano esta diferença é imperceptível. Há diversas pesquisas na área de compressão de dados a fim de diminuir a quantidade de operações necessárias para cálculo de DCT, que é a base de um dos principais métodos de compressão atual.

Um padrão que emerge como de "facto" nessa área é o JPEG, que nas suas implementações permite compressões com perda controlada de resolução (um número de 100% a 0%), onde 100% são as perdas decorrentes apenas de arredondamento de valores e 0% perda total. 75% é muito usado, onde perde-se 25% de qualidade, mas ganha-se em torno de 4:1 em compactação. Taxas como 50% já começam a apresentar mesmo a olho nu diferenças da imagem original, mas todavia consegue-se taxas de mais de 6:1. Essa perda pode ser tolerada em algumas aplicações, como por exemplo sequências de animação.

Este Trabalho apresentou uma técnica de compressão sem perda (Huffman) e duas técnicas com perda baseadas em BTC e DCT. No caso do DCT, estudou-se o uso de matrizes 4*4 e 8*8 para conseguir taxa de compressão fixa de 4:1.

7. Bibliografia

[BAR_88] - Barnsley, M. F,; Sloan, A. D.: *A Better Way to Compress Images*, Byte, Janeiro de 1988, pp. 215-223.

[BEN_91] - Bender, Paul E.; Wolf, Jack K.: *New Asymptotic Bounds and Improvements on the Lempel-Ziv Data Compression Alghorithm*, IEEE Transactions on Inf. Theory, Vol. 37, No. 3, Maio de 1991, pp. 721-729.

[BLA_91] - Black, Uyless: *The V Series Recommendations*, McGraw-Hill Inc, 1991.

[BRI_91] - Bridges, John: *Differencial Image Compression*, Dr. Dobbs Journal, Fevereiro de 1991, pp. 38-51.

[DEE_94] - Deel, Ernie F.: *Adaptatve Block Coding*, Dr. Dobbs Journal, Março de 1994, pp. 127-130.

[EMB_91] - Embree, Paul M. & Kimble, Bruce: *C Language Algoritmhms for Digital Signal Processing*, Prentice Hall, 1991

[GAI_93] - Gailly, Jean-loup: *comp.compression Frequently Asked Questions*, Internet compression-faq/part[1-3], 9 Fevereiro de 1993.

[HEL_91] - Held, Gilbert: *Data Compression*, John Wiley & Sons LTD, 1991, 3a. Ed.

[HUF_52] - Huffman, David: *A Method for the Construction of Minimum Redundancy Codes*, Proceedings of the I.R.E., Vol. 40, No. 9, Setembro de 1952,pp. 1098-1101

[JOH_93] - Johnson, Eric F & Reichard, Kevin: *Professional Graphics Programming in the X Window System*, Mis:Press, 1993

[LAN_93] - Lane, Tom: *JPEG image compression: Frequently Asked Questions*, Internet jpeg-faq, 6 Fevereiro de 1993.

[LEL_87] - Lelewer, D. A.; Hirschberg, D. S. *Data Compression*, ACM Computing Surveys,Vol. 19, No. 3, Setembro de 1987, pp. 261-296.

[KRU_92] - Kruger, Anton: *Block truncation compression*,Dr. Dobbs Journal, Abril de 1992, pp. 48-52.

[MAS_89] - Mascarenhas, N.D.A.; Velasco, R.D.: *Processamento Digital de Imagens*, IV Escola Brasileiro Argentina de Informática, Janeiro de 1989.

[MIC_93] - *Microsoft MS-DOS 6 User's guide*, 1993.

[NEL_89] - Nelson, Mark: *LZW Data Compression*, Dr. Dobbs Journal, Vol. 14, No. 10, Outubro de 1989, pp. 29-37.

[NEL_91] - Nelson, Mark: DDJ *Data Compression Contest Results*, Dr. Dobb's Journal, Novembro de 1991, pp. 62-64.

[NEL_92] - Nelson, Mark: *The Data Compression Book*, M & T Books, 1992.

[PAN_94] - Paniago, C. F. A.: *Um sistema de compressão de imagens Digitais*, dissertação de mestrado apresentada ao ICMSC-USP, maio de 1994

[PEC_82] - Pechura, Michael: *File Archival Techniques Using Data Compression, Communications of ACM*, Vol. 25, Numero 9, Setembro de 1982, pp. 605-609.

[POS91] - Poskanzer, Jeff: *Extended Portable Bitmap Toolkit*, Programas e manuais na fita do X11R5 (diretório: contrib/clients/pbmplus, 05 outubro de 1991.

[RUB_76] - Rubin, Frank: *Experiments in Text File Compression*, Communications of ACM, Vol. 19, no. 11, Novembro de 1976, pp. 617-623.

[THO_91] - Thomas, Kas: *Entropy*, Dr. Dobb's Journal, Fevereiro de 1991, pp. 32-34, 110.

[TRA_92] - Traina, Agma J. M.: *Processamento digital de imagens*, notas didádicas do ICMSC/USP - Computação Gráfica, 1992.

[WAL_91] - Wallace, Gregory K.: *The JPEG still Picture Compression Standard*, Vol. 34, No. 4, abril de 1991, pp.31-44.

[WEL_84] - Welch, Terry: *A technique for High Performance Data Compression*, IEEE Computer, Vol. 17, No. 6, Junho de 1984, pp. 8-19.

[WEG_92] - Wegner, Tim: *Image Lab*, Waite Group Press, 1992

[ZIV_77] - Ziv, J.; Lempel, A.: *A Universal Algorithm for sequential Data Compression*, IEEE Transactions on Information Theory, Vol. 23, No. 3, Maio de 1977, pp. 337-343.

[ZIV_78] - Ziv, J.; Lempel, A.: *A Compression of Individual Sequences via Variable-Rate Coding*, IEEE Transactions on Information Theory, Vol. 24, No. 5, Setembro de 1978, pp.530-536.