MINISTÉRIO DO EXÉRCITO
SECRETARIA DE CIÊNCIA E TECNOLOGIA
(Real Academia de Artilharia, Fortificação e Desenho, 1792)

TESE DE MESTRADO

MÓDULO DE COMPACTAÇÃO DE IMAGENS DISCRETAS

Marco Antonio Lemos Parra

IME

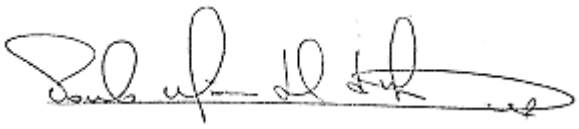
INSTITUTO MILITAR DE ENGENHARIA

MÓDULO DE COMPACTAÇÃO DE IMAGENS DISCRETAS

POR
MARCO ANTONIO LEMOS PERNA

TESE SUBMETIDA
COMO REQUISITO PARCIAL
PARA OBTENÇÃO DO GRAU DE
MESTRE EM CIÊNCIAS
EM SISTEMAS E COMPUTAÇÃO

Assinatura do Orientador da Tese



Paulo Márcio Leal de Menezes - M.C./IME

Rio de Janeiro - RJ

Fevereiro 1994

A Deus em primeiro lugar,

a meus pais e a meu irmão.

AGRADECIMENTOS

Ao INSTITUTO MILITAR DE ENGENHARIA, pela oportunidade de realização deste curso e por toda a experiência e ensinamentos adquiridos.

À SEÇÃO DE CARTOGRAFIA (SE/6), pela acolhida e apoio durante todo o período do curso.

Ao Professor PAULO MÁRCIO LEAL DE MENEZES, pela orientação perfeita e profundos conhecimentos transmitidos e pelo apoio, compreensão e amizade demonstrados no decorrer deste trabalho.

Ao Professor ANTONIO JOSÉ MACHADO E SILVA, coorientador, cuja contribuição foi de grande valia para a finalização deste trabalho.

Ao Professor JORGE XAVIER-DA-SILVA, pela gentileza, interesse e contribuições ao participar da Banca Examinadora.

Ao Professor e amigo LEONARDO CASTRO DE OLIVEIRA, pelas contribuições a este trabalho.

À amiga FLÁVIA MANDARINO, pela solicitude e colaboração tão valiosas para a realização deste trabalho.

RESUMO

Este trabalho de dissertação de mestrado se propõe a estudar algoritmos de compactação visando a redução do volume de dados para armazenamento de imagens multiespectrais e imagens obtidas através de digitalização de mapas por varredura. Para tanto, é feito um estudo das características particulares desse tipo de dado e dos procedimentos necessários para a realização da compactação. Este trabalho tem como objetivo fim, a escolha de algoritmos que atendam as necessidades de compactação dessas imagens através de testes e a proposição de melhorias nesses algoritmos visando uma melhora da compactação. Por último é elaborado um Módulo de Compactação que atenda essas necessidades.

ABSTRACT

The primary purpose of this work is to study various error-free data-compression algorithms. This work seeks to reduce the number of bits used to store multispectral and scanned images. For this, this work presents a close look for this type of data and pre-processing procedures. The final purpose is the choice, by performance tests, of the algorithms that fall in the image compression requirements. In these algorithms are proposal changes with the focus in better performance. At last, an error-free compression module is developed, which applies the techniques used in this work.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

O início dos estudos para compactação de dados datam da década de 40. A compactação de dados consiste basicamente em reduzir o volume dos mesmos, podendo-se retornar ao estado anterior.

Porém, no contexto de imagens multiespectrais provenientes de satélites de sensoriamento remoto ou imagens obtidas pela digitalização matricial de mapas e cartas, torna-se necessário um estudo para avaliar a eficácia dos diversos algoritmos conhecidos ou recém desenvolvidos. Neste estudo, devem ser levadas em consideração as características particulares desse tipo de dado.

As imagens de satélite possuem uma utilização relevante há muitos anos. Por outro lado, as imagens digitalizadas de mapas, são utilizadas há relativamente pouco tempo, e não possuem um estudo muito profundo a respeito, visto não se tratar de um processo de obtenção de dados dos mais precisos, apesar de mapas serem documentos cartográficos cujas informações seguem normas de precisão. Essa imprecisão ocorre em função dessas imagens não serem cópias fiéis de seus mapas originais.

Entretanto, seu uso tem crescido em decorrência do aumento do emprego de Sistemas de Informação Geográfica (SIG) para os mais diversos fins onde a precisão pode não ser relevante,

sendo o aspecto estatístico do mapa mais importante, na maioria das vezes.

Com o uso crescente de imagens digitalizadas de mapas e cartas, a necessidade do estudo em relação a sua compactação pode ser exemplificada tomando-se um fragmento em tamanho A-4 (21.0 x 29.7 cm) de um mapa topográfico digitalizado por varredura a 300 dpi (pontos por polegada), ou seja, 300 pontos horizontalmente e 300 verticalmente por polegada, que necessita de 12.429.474 bits em uma digitalização em preto e branco (dois níveis de cinza) para seu armazenamento.

Isto equivale a uma necessidade de armazenamento de 1.553.684 bytes (pouco mais que 1 Mbyte), já que há 8 bits por byte. Este número de bits e bytes independe do tipo de imagem digitalizada, seja texto, mapa ou figura. A Tabela 1.1 e 1.2 contém o volume de dados de telas gráficas de computador.

TABELA 1.1: Volume de dados de telas gráficas de computador.

Dispositivo	Resol Espacial (em *pixels*)	Profundidade (em *bits*)	Volume (em kbytes)
EGA	320x200	4	31,25
VGA	320x200	8	62,50
SVGA	640x400	8	250,00
SUN/R6000	1280x1024	24	3840,00

TABELA 1.2: Volume de dados de uma folha de 8,5 por 11,0 pol.

Resolução Espacial (em dpi)	Bits/*Pixel*			
	1	4	8	24
75	64,2 kb	256,8 kb	513,6 kb	1540,8 kb
150	256,8 kb	1027,2 kb	2054,4 kb	6163,3 kb
300	1027,2 kb	4108,9 kb	8217,8 kb	24653,3 kb
600	4108,9 kb	16435,5 kb	32871,1 kb	98613,3 kb

O tamanho do arquivo gerado é demasiadamente grande para as condições de armazenamento atuais e até para futuras tecnologias, pois normalmente um projeto ou trabalho contém mais de uma imagem. A tendência dos sistemas de armazenamento (do tipo disco rígido) é de que quanto mais há o aumento do seu volume de dados, maior é a necessidade de armazenamento, pois normalmante passa-se a manter mais imagens armazenadas em disco rígido, para um acesso mais fácil do que se teria em uma fita magnética. O que mostra que é de extrema relevância o estudo para compactação desses tipos de dados em particular.

Com o avanço da velocidade de processamento, o tempo de processamento passa a segundo plano, já que o que hoje é inviável, amanhã pode ser plenamente aceitável, simplesmente trocando-se o equipamento computacional. O *hardware* não será fator determinante, porém, processadores com endereçamento direto, sem segmentação de memória, são a ferramenta ideal para a compactação e o processamento de imagens, pois demandam menor trabalho para o programador, que passa a se preocupar mais com a

compactação do que com técnicas para suplantar problemas de tecnologia de *hardware* e *software*.

O arquivo compactado poderá ser facilmente manipulado por qualquer sistema de processamento de imagens ou SIG, bastando a inclusão deste módulo de compactação-descompactação no sistema como uma função que leia a imagem descompactando-a para a memória principal.

Este trabalho tem como objetivos a elaboração de um estudo em codificação de dados no que tange reduzir o volume de dados de forma reversível, ou seja que se possa retornar aos dados originais, e a elaboração de um módulo de compactação e descompactação de imagens digitais.

No Capítulo 2 encontra-se uma exposição básica a cerca de imagens digitais e suas características, dos processos de aquisição e da preparação para compactação.

O Capítulo 3 aborda codificação de dados e os métodos de compactação.

No Capítulo 4 são feitas propostas para o Módulo de Compactação.

O Capítulo 5 expõe a linguagem a ser utilizada, o sistema operacional e os cuidados necessários para implementação.

O Capítulo 6 contém os testes realizados envolvendo imagens e os resultados utilizando vários algoritmos compactadores.

No Capítulo 7 estão as conclusões e recomendações.

CAPÍTULO 2

IMAGENS DIGITAIS

2.1 - DEFINIÇÃO

Imagem digital ou discreta é uma matriz bidimensional MxN (linha x coluna). Cada elemento da imagem digital é chamado de ***pixel*** ou ***pel*** (*picture element*), que contém valores inteiros, ou seja, discretos (os níveis de cinza ou atributos de cor, que contém uma informação discreta sobre um determinado ponto) variando de 0 a K-1.

A imagem digital, Figura 2.1a, difere de sua correspondente analógica (contínua), Figura 2.1b, no que diz respeito à sua qualidade visual e a forma com que é armazenada e manipulada. Salvo em casos em que não existam suas correspondentes analógicas (por exemplo, imagens digitais para visualização de abstrações matemáticas, médicas, geológicas ou artísticas), a imagem analógica possui qualidade visual superior à da digital (antes de se realizar qualquer processamento para melhoria da qualidade visual da imagem digital), pois carrega mais informação em sua área de definição, ou seja, no espaço visual onde se desenvolve. Normalmente, a imagem analógica possui uma gradação de cores maior que a digital, bem como nitidez também maior, não se notando falhas em linhas contínuas. Isso é motivado porque a imagem digital necessita de uma grande quantidade de pixels, em sua área de definição, com uma grande quantidade de

cores para que o olho humano não perceba a diferença entre ela e sua correspondente analógica.

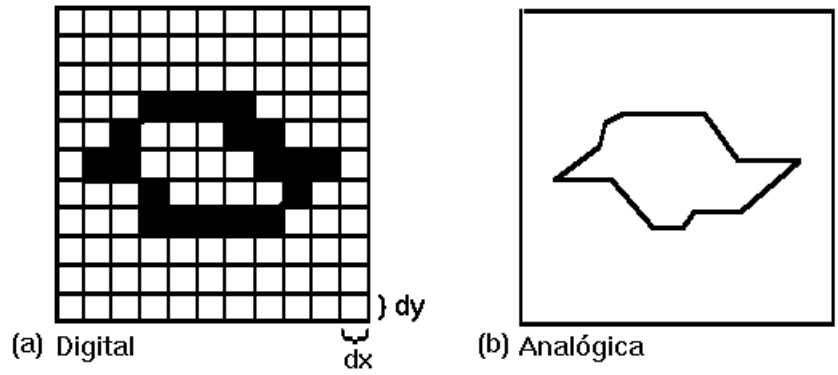


FIGURA 2.1: Representação Digital e Analógica.

Uma resolução (densidade) típica de 300 dpi (pontos por polegada, pontos = pixels), permite ao olho humano perceber uma pequena diferença, na qualidade visual, da imagem analógica. Porém, apesar de ter uma qualidade visual apenas aceitável, para uma imagem no tamanho A-4, possui um volume de 12.429.474 bytes (caso use-se um pixel com 8 bits, o que proporciona uma quantidade de 256 cores ou níveis de cinza), ou aproximadamente 12 Mbytes, que é um tamanho muito grande para armazenamento de uma imagem para os padrões atuais de armazenamento, levando-se em conta também, que em qualquer trabalho não deve ser usada apenas uma imagem. Esses valores dificultam a aquisição e manipulação de imagens com essa qualidade.

Obtem-se uma imagem digital a partir de uma imagem analógica através de um processo que envolve dois passos:

(i) amostragem, que consiste em discretizar o domínio de definição da imagem, ou seja, escolher os valores dx e dy da grade, Figura 2.1a;

(ii) quantização, que consiste em escolher um valor inteiro relacionando a quantidade de informação que cada pixel pode discriminar, com um intervalo de valores inteiro e finito com espaçamento múltiplo de uma constante.

Uma imagem digital é definida matematicamente da seguinte forma :

$$I(i, j): \mathrm{N}^2 \rightarrow \mathrm{N}$$

$$I(i, j) \in [o, \mathrm{N}]$$

$$I(i, j) = z, \qquad i \in [0, M-1]$$

$$j \in [0, N-1]$$

$$z \in [0, Z-1]$$

onde: i = índice de linha;

j = índice de coluna;

z = nível de cinza;

M = número máximo de linhas;

N = número máximo de colunas.

Z = número de níveis de cinza;

Os coeficientes M, N e K geralmente são potências de 2, sendo que M e N normalmente possuem os valores 256, 512 ou 1024 dentre outros, e K normalmente possui os valores 2 (pixel com 1 bit), 32 (pixel com 5 bits), 64 (6 bits), 128 (7 bits) ou 256 (8 bits ou um byte). Quando K é igual a 2, ou seja, pixel com 1 bit, assumindo os valores 0 e 1, a imagem é denominada **binária**.

Cada pixel de uma imagem digital possui um valor distinto que pode ser considerado uma amostra de uma imagem analógica, sendo a imagem digital no seu todo, o espaço amostral

da sua correspondente analógica. Logo, a imagem como um conjunto de amostras pode ser sujeita a processamentos estatísticos e probabilísticos. O **histograma** de uma imagem representa a distribuição das amostras pelo espaço amostral, como mostra a Figura 2.2. O histograma tem utilização na análise de uma imagem em conjunto com os resultados obtidos pela compactação. Porém, sua utilização com amostras estatísticamente dependentes não permite uma análise completa, necessitando-se a análise visual da própria imagem digital, entre outras análises possíveis.

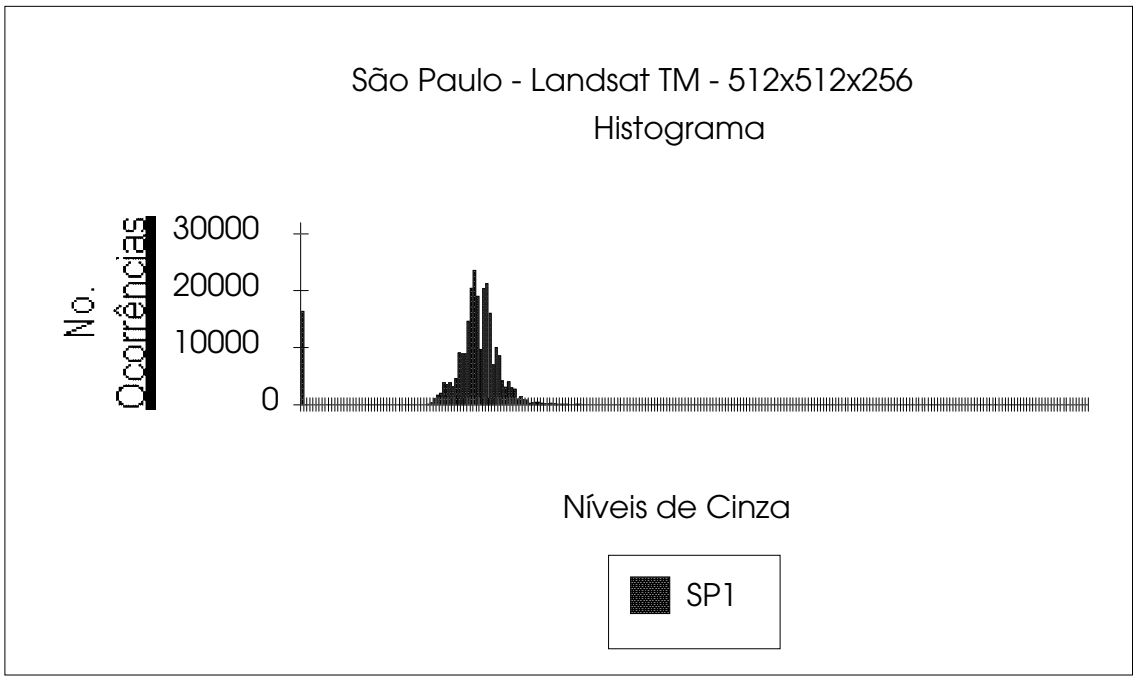


FIGURA 2.2: Histograma de uma Imagem Digital.

2.2 - ESPECTRO ELETROMAGNÉTICO

No conjunto do espectro eletromagnético (Figura 2.3) as radiações visíveis, isto é, aquelas sensíveis ao olho humano, têm

comprimentos de onda que vão desde 380 até 760 nanômetros. Cada faixa dessas radiações corresponde a emissão de determinada cor (Tabela 2.1).

As cores visíveis correspondem apenas a uma pequena parte do espectro eletromagnético. As faixas do espectro não visíveis ao olho humano são captadas por sensores especiais a fim de obter imagens que informem características não visíveis do elemento observado.

TABELA 2.1: Comprimento de onda das cores visíveis.

Violeta	380 - 450 nm
Azul	450 - 500
Verde	500 - 570
Amarelo	570 - 590
Laranja	590 - 610
Vermelho	610 - 760

Essas características determinam a sua ***assinatura espectral*** (Oliveira, 1988), que é a maneira particular segundo a qual a matéria emite ou reflete energia, o que torna possível a identificação dessa matéria.

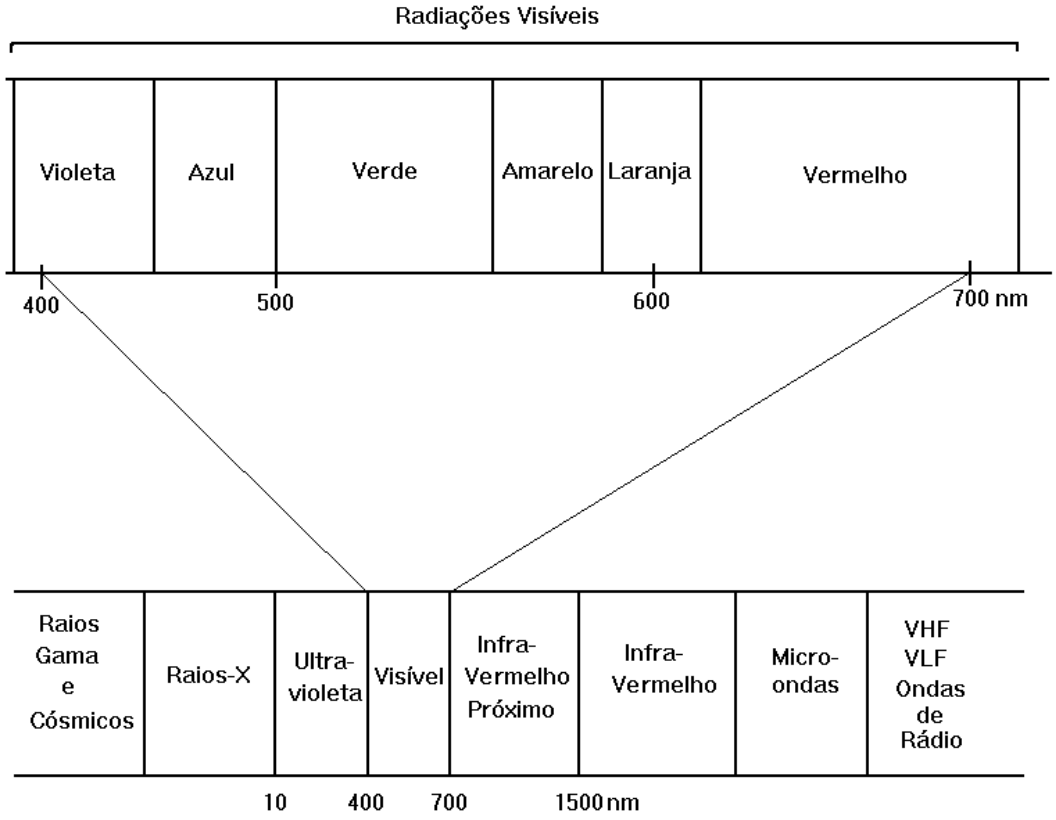


FIGURA 2.3: Espectro eletromagnético.

2.3 - IMAGENS MULTIESPECTRAIS E COLORIDAS

Uma imagem **multiespectral** é uma coleção de imagens de uma mesma cena, em um mesmo instante, obtida por um ou mais sensores com respostas espectrais diferentes. Cada faixa do espectro corresponde a uma banda. Sendo assim, ao se referir a uma determinada posição do ponto I em uma imagem multiespectral usa-se o terno ordenado (i,j,k), onde i é a linha, j a coluna e k a banda. Sendo I(i,j,k) = z, z = nível de cinza do ponto I na linha i, coluna j e banda k.

Quando os sensores apresentam sensibilidade nas faixas do visível com bandas correspondentes ao azul, verde e vermelho, a composição colorida é dita em **cor natural**. Isto porque o

sistema sensor reproduz o sistema da retina do olho humano. Caso contrário, diz-se que a imagem está em **falsa cor**.

É possível, através de combinações lineares das diversas bandas, produzir uma imagem colorida que se aproxime da imagem em cor natural, mesmo com sensores que operam fora do espectro do visível. Neste caso, essa imagem é dita em **cor pseudo-natural**.

2.4 - SISTEMAS DE CORES

Muitas teorias científicas têm procurado explicar a percepção da cor. Segundo Young-Helmholtz (LOVELL, 1992), o olho humano obtém a sensação da cor mediante a excitação de três tipos de cones retinianos sensíveis às três principais regiões da porção visível do espectro de radiações eletromagnéticas: as regiões do azul-violeta, do verde e do vermelho.

Quando há excitação de um cone apenas, formam-se alternadamente as cores **primárias luminosas**: azul-violeta, verde e vermelho-alaranjado. Quando há excitação simultânea de dois cones formam-se, alternadamente, as cores **primárias pigmento** : ciano, magenta e amarelo.

Prova-se esta composição através da síntese aditiva, superpondo-se parcialmente três feixes luminosos em uma câmara escura. Na superposição total das três luzes coloridas forma-se o branco. Na superposição parcial do verde com o vermelho-alaranjado forma-se o amarelo. Na superposição parcial do azul-violeta com o vermelho-alaranjado aparece o magenta. Finalmente, na superposição parcial do verde com o azul-violeta aparece o

azul-esverdeado (ciano). Os tubos de raios catódicos (monitores de vídeo, televisão) baseiam-se neste processo.

No raciocínio inverso, isto é, bloqueando-se parcialmente a luz branca com filtros teremos, sobre uma tela branca, a síntese subtrativa. Na superposição total dos três filtros (ciano, magenta e amarelo) não há passagem de luz, aparecendo o preto. Na superposição parcial do filtro amarelo com o filtro magenta obtém-se o vermelho-alaranjado. Na superposição parcial do filtro amarelo com o filtro ciano obtém-se o verde. E na superposição parcial do filtro ciano com o filtro magenta obtém-se o azul-violeta. As artes gráficas (e.g., impressão de mapas) baseiam-se neste processo.

Com as cores primárias luminosas é montado um modelo aditivo denominado Sistema **RGB** (*Red*-vermelho, *Green*-verde e *Blue*-azul) que pode ser visualizado por um sistema de coordenadas cartesianas, como mostra a Figura 2.4.

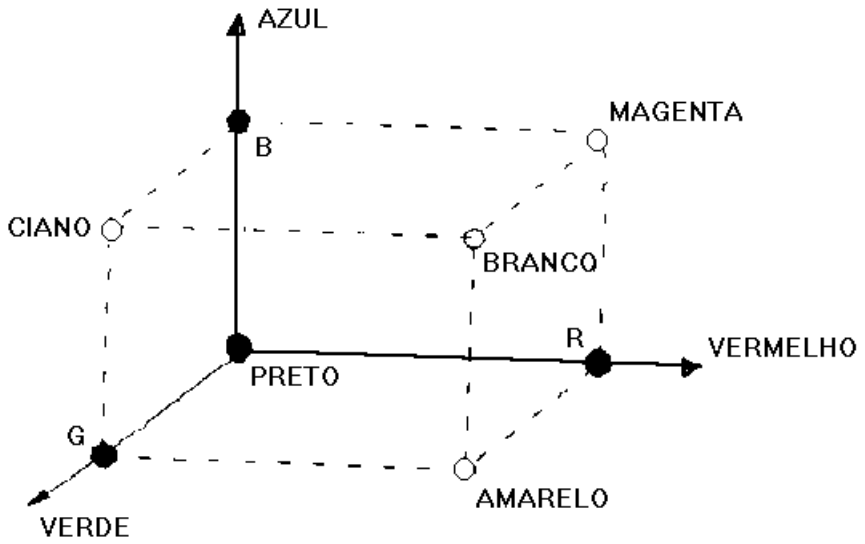


FIGURA 2.4: Sistema RGB e CMY.

Com as cores primárias pigmento é montado um modelo subtrativo denominado Sistema **CMY** (*Cian*-ciano, *Magenta* e *Yellow*-amarelo) que também pode ser visualizado em um sistema de coordenadas cartesianas, porém complementar ao sistema RGB como mostra a Figura 2.4.

2.5 - AQUISIÇÃO DE IMAGENS

Os sistemas físicos destinados a produzir ou captar imagens digitais são chamados de **Sistemas de Imageamento**. São divididos em duas categorias:

(i) aquisição indireta : onde a imagem digital é captada ou registrada indiretamente, ou seja, a partir de uma representação gráfica já existente. É o caso da obtenção de imagens digitais a partir de analógicas. Um exemplo de sistema físico é o *scanner*. As câmeras fotográficas digitais podem ser utilizadas.

(ii) aquisição direta : onde a imagem digital é captada ou registrada diretamente do que se quer transformar em imagem digital. Alguns exemplos a destacar são os sensores de satélites e as câmeras fotográficas digitais;

Na aquisição direta de uma imagem de satélite a imagem de uma cena é registrada em um plano através de um sistema ótico-eletrônico. Neste plano, são colocados sensores que medem a intensidade da luz incidente, como também a sua freqüência. Os dois tipos de sensores mais usados na obtenção de imagens digitais são a câmera de TV e os dispositivos fotossensíveis.

Por outro lado, a aquisição indireta de imagens a partir de mapas é efetuada por digitalização por varredura, que é efetuada por aparelhos denominados ***scanners***, dispositivos exclusivos para entrada de dados. Todos os *scanners* possuem um princípio de funcionamento semelhante, independentemente da imagem a ser digitalizada estar em preto e branco, em tons de cinza ou em cores diversas.

Na digitalização através de *scanners*, a luz que incide sobre o papel é refletida e através de um jogo de espelhos e filtros atinge um dispositivo denominado ***Charge-Coupled Device*** (CCD) (Figura 2.5). O CCD é um dispositivo fixo com aproximadamente uma polegada quadrada e alguns milhares de células fotossensíveis.

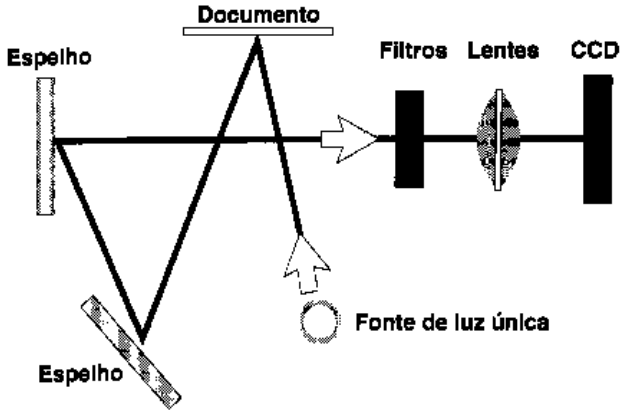


FIGURA 2.5: *Scanner* monocromático.

Diferentes tipos de CCD fazem diferentes captações da imagem digitalizada. Existem CCDs que só reconhecem dois níveis de intensidade de luz. Estes equipam os *scanners* preto e branco. Os coloridos e os de tons de cinza são bem semelhantes, e podem possuir CCDs que reconheçam até 256 intensidades diferentes de

luz, ou seja, gera um *pixel* com 256 níveis de cinza que é representado em oito bits.

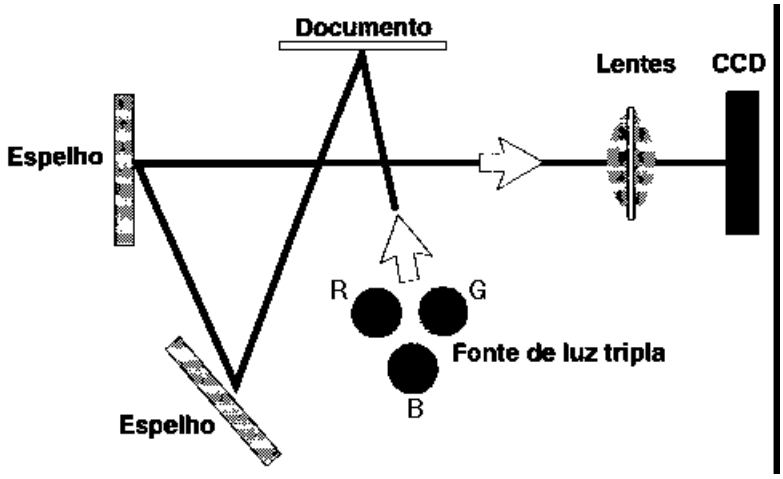


FIGURA 2.6: Aquisição de imagem colorida.

As imagens coloridas são compostas das três cores primárias luminosas (*Red*-vermelho, *Green*-verde e *Blue*-azul). Para captá-las existem três métodos diferentes.

(i) O mais comum utiliza o mesmo método dos *scanners* que digitalizam tons de cinza, porém são feitas três passagens, uma para cada componente (cor). Assim, tem-se um total de 24 bits para cada célula, o que resulta em 16.777.216 opções de cores para cada ponto.

(ii) Outra alternativa é a realização de uma única varredura com as lâmpadas das três cores piscando alternadamente (Figura 2.6).

(iii) O último método faz com que a luz refletida pela imagem passe através de filtros, que separam as três cores básicas atingindo, respectivamente, três CCDs independentes (Figura 2.7).

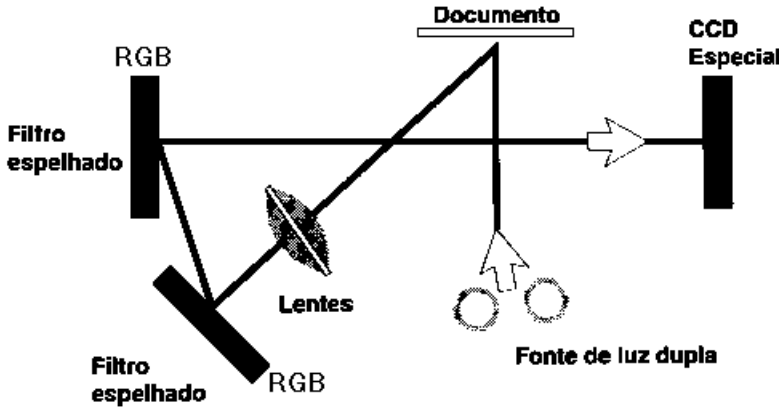


FIGURA 2.7: Reflexão por filtros.

2.6 - RESOLUÇÃO NECESSÁRIA PARA AQUISIÇÃO DE DADOS PARA USO CARTOGRÁFICO

A **resolução espacial** é grandeza que indica quantidade de *pixels* por unidade de área. Em um *scanner*, geralmente pode variar entre 100 dpi e 2000 dpi (pontos por polegada). A maior grandeza gráfica apreciável a olho nu é de 0,2 mm, o que equivale a 1/127 da polegada ou a uma resolução de 127 dpi. Logo, para digitalização por varredura de mapas a resolução deve ser sempre maior ou igual que 127 dpi.

Em (Cartensen, 1991) foram realizados estudos sobre resolução espacial, sendo mostrado que 150 dpi e 300 dpi são resoluções aceitáveis em relação à precisão cartográfica.

De acordo com (Francisco, 1993), verifica-se que a necessidade de resoluções maiores é decorrência da precisão exigida pela aplicação a ser feita do mapa digitalizado por varredura.

A **resolução radiométrica** é o intervalo de valores que cada pixel pode discriminar. Normalmente, essa resolução pode

abranger de 1 bit (imagem binária) a 24 bits (imagem RGB). É também função do uso que será feito do mapa.

Cabe ressaltar que quanto maiores forem essas resoluções, maior será o arquivo gerado.

2.7 - DIGITALIZAÇÃO DE MAPAS POR VARREDURA

A digitalização de mapas através de varredura é um processo que visa aproveitar a base cartográfica existente para utilização em áreas como, por exemplo, Sistemas de Informação Geográfica. Porém, esse procedimento é custoso devido à dificuldade da separação das cores e níveis de informação na imagem obtida em situação real de aquisição.

Os mapas possuem poucas cores, com valores teoricamente precisos após a digitalização por varredura. Esses valores são precisos apenas na teoria, devido a falhas durante a digitalização. Dentre estas falhas podem ser citadas: mapa antigo com cores desbotadas, sujeira entre o mapa e o *scanner*, etc..

Entretanto, de acordo com o pré-conhecimento dos valores que devem ser obtidos para cada cor, é possível efetuar uma correção radiométrica (QUINTANILHA, 1991) pelo processo de quantização, discretizando os valores das cores. Por exemplo, sabendo que determinada cor possui o valor 32, e que o mapa não possui nenhuma outra cor na vizinhança de 10, ou seja entre 22 e 42, pode-se assumir que pelo menos qualquer valor entre 27 e 37 corresponde àquela determinada cor, devendo portanto, assumir o valor 32.

Para que essa metodologia possa ser utilizada, deve-se utilizar a maior resolução radiométrica possível, de preferência de 24 bits (RGB), para facilitar a separação de cores.

Para digitalizar mapas por varredura de forma mais eficiente e precisa, efetua-se a digitalização dos mapas por níveis de informação, ou *overlays*, no caso da terminologia da cartografia convencional. Assim, obtém-se uma imagem digital por *overlay*.

Os *overlays* são fotoplásticos copiados, por contato, do original cartográfico (para a fase de gravação), tantos quantos forem as cores em que haja elementos representados a traço, ver (Oliveira, 1988) e (Oliveira, 1983) para maiores detalhes.

Para uma folha topográfica, é necessário que o original seja exposto cinco vezes, pois impõem-se cinco tipos de gravação, isto é, uma para cada tipo de representação correspondente a estas cores: preto, azul, sépia (castanho), vermelho e verde. Deste modo, têm-se cinco fotoplásticos, cada um com uma dessas cores, o que torna a digitalização perfeita no que diz respeito à separação de cores e níveis de informação. Uma descrição das etapas de um procedimento de digitalização por varredura é feita por (Francisco, 1993).

As imagens obtidas a partir dos *overlays* são binárias e são armazenadas em arquivos distintos. A imagem digitalizada por varredura de mapa colorido é compostas dos componentes RGB, podendo ser armazenada em arquivos diferentes para cada componente, como as bandas das imagens multiespectrais, ou em qualquer outro formato de armazenamento.

CAPÍTULO 3

CODIFICAÇÃO

3.1 - DEFINIÇÃO

Codificar uma **fonte** (arquivo, origem dos dados) de **símbolos** (bytes, caracteres, pixels, etc.) significa modificar um sistema de representação da informação do símbolo ou grupos de símbolos emitidos por essa fonte, para outro sistema, de forma que se possa retornar ao sistema original ou próximo dele.

Entre alguns dos objetivos da codificação pode-se citar :

(i) a redução do volume de dados da fonte;

(ii) criptografia;

(iii) como meio auxiliar na transmissão de dados, tanto para correção de erros, quanto para envio de símbolos de controle da transmissão.

A codificação, no escopo desse trabalho, tem como objetivo reduzir o volume de informação necessária para representar uma imagem, gerando códigos para cada símbolo com menos bits, em média, que o símbolo original.

A codificação, neste caso, aplica-se na transmissão de imagens em redes de distribuição de TV, em sistemas de teleconferência, em transmissão de documentos por fac-símile, em armazenamento de documentos, em armazenamento de imagens médicas, de satélites, etc.

Os métodos para codificação de imagens baseiam-se na eliminação da redundância das imagens e nas limitações do sistema visual humano. Eles podem ser divididos em:

(i) Métodos Reversíveis: a imagem é codificada e decodificada sem perda de informações. São chamados também de métodos Sem Perda (*loss-less*, *error-free*). A codificação é denominada de **compactação** e a decodificação de descompactação de acordo com recomendação em (BLAHUT, 1990).

(ii) Métodos Irreversíveis: a imagem é codificada e decodificada com perda de informações, porém alcançando taxas maiores. Esse método geralmente envolve três estágios: uma transformação reversível (eliminação de redundância), uma codificação irreversível e uma codificação reversível utilizando algum método reversível. Existem diversos métodos, como por exemplo o **JPEG** (*Joint Photographic Experts Group*) (KAY, 1993). A codificação é denominada de **compressão** e a decodificação de descompressão de acordo com recomendação em (BLAHUT, 1990).

Da Teoria da Informação (ABRAMSOM, 1963) vem que a **entropia** (quantidade de informação) gerada por uma fonte de símbolos é definida pela Equação 3.1.

$$H(S) = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} \quad \textit{bits / símbolo} \qquad (3.1)$$

onde : n = número de símbolos a ser codificado;

p_i = a probabilidade de ocorrência do i-ésimo símbolo na fonte;

H(S) = entropia da fonte de símbolos. Define um número teórico mínimo de bits por símbolos necessários para codificar mensagens geradas por essa fonte, assumindo que os sucessivos símbolos emitidos da fonte são estatísticamente independentes.

E o número de bits necessários para codificar um símbolo (entropia do símbolo) é dado pela Equação 3.2.

$$H_i = \log_2 \frac{1}{p_i} \quad \textit{bits / símbolo} \qquad (3.2)$$

onde: p_i = a probabilidade de ocorrência do símbolo i na fonte;

H_i = define um número teórico mínimo de bits por símbolo, assumindo que os sucessivos símbolos emitidos da fonte são estatísticamente independentes.

Em geral, a compactação de dados é efetuada obtendo símbolos de uma fonte, processando-os, e colocando os códigos em um arquivo (compactado), Figura 3.1.

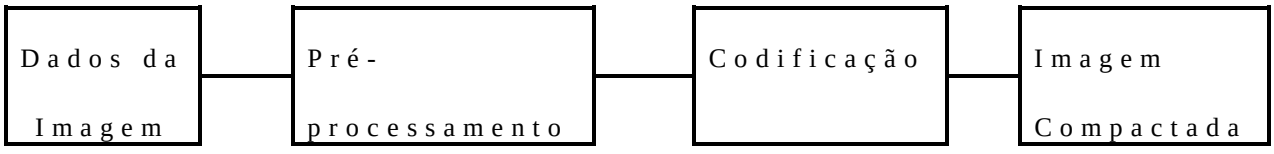


FIGURA 3.1: Processo de Compactação.

Para o desenvolvimento de um método de compactação de dados que atinja seu objetivo, no caso da utilização de 8 bits para

codificar cada símbolo, é preciso que o número médio de bits por símbolo da codificação gerado pela fonte seja inferior a 8 para que o resultado da compactação tenha um tamanho menor que a fonte de símbolos. Na quase totalidade dos casos práticos este número é inferior a 8, e, em geral, não é inteiro. Para se tentar aproximar deste valor mínimo é preciso que se utilize alguma técnica de codificação.

A taxa de compactação (TC), em percentagem, é a relação entre o volume original dos dados a serem compactados e o volume dos dados após a compactação. Ela é definida pela Equação 3.3.

$$TC = 100 * \frac{\text{VO - VC}}{\text{VO}} \qquad (3.3)$$

onde : VO = Volume Original;

VC = Volume Compactado.

Quanto maior a TC (i.e., mais próxima de 100%), menor o volume dos dados compactados.

3.2 - MÉTODOS DE COMPACTAÇÃO

Grande parte dos métodos de compactação de uso comum hoje em dia recaem em três metodologias:

(i) método Baseado em Dicionário;

(ii) métodos estatísticos;

(iii) métodos simples.

Na prática, o limite entre os métodos não é de fácil distinção, pois alguns métodos não podem ser perfeitamente caracterizados por uma ou outra metodologia, existindo processos híbridos.

Neste trabalho é dado ênfase nos métodos estatísticos, por possuirem maior quantidade de algoritmos baseados neles. Além de que, seus conceitos podem ser utilizados em outros métodos ou em métodos híbridos. São apresentados um algoritmo simples e um baseado em dicionário.

3.3- MÉTODOS ESTATÍSTICOS

A necessidade de se predizer corretamente a probabilidade de ocorrência dos símbolos (e.g., níveis de cinza, ou valores dos pixels) obtidos da fonte é inerente da natureza dos métodos estatísticos.

O princípio desse tipo de codificação é reduzir o número de bits necessários para codificar um símbolo a medida que sua probabilidade de ocorrência aumenta (primeira condição para compactação).

Por exemplo: em uma imagem com pixel de 8 bits, o nível de cinza 35 representa 25%, ou seja, possui uma probabilidade de 1/4 dos pixels de entrada, ele será codificado com apenas 2 bits, pois sendo $H = \log_2 (1/(1/4))$, $H = 2$. Se o nível 56 possui uma probabilidade de 1/1024 que é aproximadamente 0,1% dos pixels de entrada, ele será codificado com 10 bits, pois, sendo $H = \log_2 (1/(1/1024))$, $H = 10$. Se o modelo não estiver gerando probabilidades corretamente, ele poderá trocar as

probabilidades e usar 10 bits para representar o 35 e 2 bits para representar o 56, causando expansão da imagem ao invés de compactação, já que 25% da imagem aumentará de tamanho em 2 bits por pixel original e apenas 0,1% da imagem sofrerá redução de 6 bits por pixel original.

A segunda condição para compactação estabelece que o modelo probabilístico deva fazer predições que divirjam de uma distribuição uniforme. Tanto melhor o modelo faça essas predições, melhor será a taxa de compactação. Por exemplo, um modelo poderia ser criado para atribuir a todos os 256 níveis de cinza, de uma imagem com pixel de 8 bits, uma probabilidade uniforme de 1/256, ou seja, uma distribuição uniforme. Esse modelo cria um arquivo de saída que é exatamente do mesmo tamanho do arquivo origem, porque cada nível de cinza é codificado com sua entropia exatamente igual a 8 bits.

Somente encontrando corretamente probabilidades que divirjam de uma distribuição uniforme o número de bits pode ser reduzido, levando à compactação. O aumento das probabilidades tem que refletir corretamente a realidade, como prescrito na primeira condição.

A probabilidade de ocorrência de um dado símbolo emitido pela fonte não é fixo. Dependendo do modelo probabilístico utilizado, a probabilidade de ocorrência de um símbolo pode mudar totalmente.

Por exemplo: quando compacta-se um texto, como o penúltimo parágrafo, a probabilidade de um caracter de controle para finalização de parágrafo no texto é 1/280, ou seja uma ocorrência (no final do parágrafo) em 280 caracteres (incluindo

espaços). Essa probabilidade é determinada para todo o texto, dividindo o número de ocorrências do caracter pelo número total de caracteres. Mas utilizando-se de uma técnica de modelagem que visualize o caracter anterior, a probabilidade mudará. Nesse caso, se o caracter anterior for um ".", a probabilidade de ocorrência de um caracter de controle para finalização de parágrafo passa para 1/2, pois só existem dois caracteres '.' e é apenas após o último '.' que termina o parágrafo.

Essa técnica melhorada de modelagem leva a uma melhor taxa de compactação, mesmo que ambos os modelos estejam gerando corretamente as probabilidades.

3.3.1 - MODELAGEM DE CONTEXTO-FINITO

O tipo de modelagem denominada modelagem de "contexto-finito", (NELSON, 1991) baseia-se em uma idéia bem simples : A probabilidade de cada símbolo é calculada em relação ao contexto no qual aparece. Nos exemplos já mostrados, o contexto consiste nos símbolos emitidos anteriormente pela fonte que são visualizados para o cálculo da probabilidade de ocorrência, o que é chamado **ordem** do modelo. O termo "finito" é empregado por levar em consideração um número finito de símbolos anteriores ao símbolo corrente.

O modelo mais simples de contexto-finito é o modelo ordem-0, no qual a probabilidade de cada símbolo independe de qualquer símbolo anterior. Para implementar esse modelo, necessita-se somente de uma tabela simples contendo a freqüência que cada símbolo possue na fonte, no caso de uma imagem, o seu

histograma. Para um modelo ordem-1, serão geradas 256 diferentes tabelas de freqüências, porque será necessário guardar em separado o conjunto de freqüências para cada possível contexto. Da mesma forma, um modelo ordem-2 necessita gerenciar 65536 diferentes tabelas de contextos. Modelos com ordem acima de zero são chamados de modelos de Markov (ABRAMSOM, 1963) (*Markov Information Source*).

A Figura 3.2 representa o modelo ordem 1. A tabela da esquerda representa o símbolo corrente, sendo que está se considerando símbolos com 8 bits e conseqüentemente 256 valores diferentes. Cada índice corresponde a um símbolo e armazena sua freqüência. As tabelas apontadas por cada símbolo representam o símbolo anterior ao corrente e suas freqüências são dependentes do símbolo corrente.

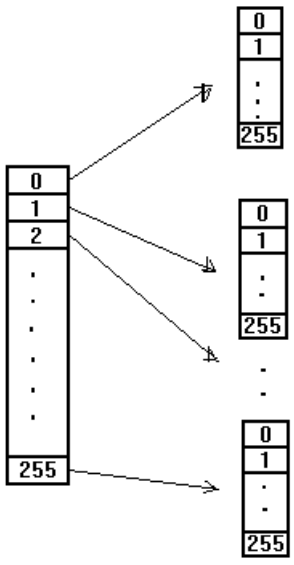


FIGURA 3.2 : Tabelas de freqüência do modelo ordem 1.

3.3.2 - MODELO ADAPTÁVEL

De acordo com o aumento da ordem do modelo, a taxa de compactação melhora, caso haja uma grande quantidade de símbolos estatísticamente dependentes, pois são necessários menos bits para a codificação. Por exemplo: a probabilidade do caracter "u" aparecer num texto pode ser de cinco por cento, mas se o caracter anterior do contexto for o "q", a probabilidade pode subir para noventa e cinco por cento. Ser capaz de predizer caracteres com alta probabilidade diminui o número de bits necessários, e grandes contextos permitem efetuar predições melhores.

Infelizmente, ao mesmo tempo que a ordem do modelo aumenta linearmente, a memória por ele consumida aumenta exponencialmente. Com um modelo ordem-0, o espaço consumido pelas estatísticas pode ser tão pequeno quanto 256 bytes. Mas, uma vez que a ordem do modelo passe para 1 ou 2, mesmo o modelo melhor idealizado e implementado consumirá milhares de bytes.

Uma forma genérica de compactação de dados é desenvolvida em dois passos:

(i) ler os dados e gerar as estatísticas para o modelo

(ii) codificar os dados, compactando-os.

Os métodos que utilizam este processo são conhecidos como **estáticos**. Pois, as estatísticas das probabilidades de ocorrência são armazenadas junto com os dados compactados para posterior descompactação, ou seja, as estatísticas são fixas ou

estáticas. Isso acarreta problemas se as estatísticas do modelo consumirem mais espaço do que os dados a serem compactados.

A solução para o problema é a utilização de uma compactação **adaptável** (ou adaptativa em um passo, ou dinâmica), na qual ambos, o compactador e o descompactador, começam com o mesmo modelo e a compactação ou descompactação é efetuada em apenas um passo, ajustando o modelo probabilístico automaticamente à medida que recebe os símbolos da fonte.

O compactador codifica um símbolo utilizando um modelo existente (*default*), atualizando-o para responder pelo novo símbolo. O descompactador, da mesma forma, decodifica o símbolo utilizando o modelo existente, atualizando-o em seguida. Como o algoritmo para atualizar o modelo funciona de forma idêntica para o compactador e para o descompactador, o processo pode perfeitamente ser efetuado sem a passagem da tabela de estatísticas do compactador para o descompactador. A atualização do modelo é o ponto fraco do algoritmo onde a compactação adaptável perde desempenho.

Para o modelo ordem-0, os mesmos símbolos podem estar armazenados em uma disposição (forma) diferente, pode-se até misturá-los aleatoriamente que a taxa de compactação permanecerá a mesma, pois a probabilidade de ocorrência de cada símbolo não é alterada. Mas para modelos de ordem diferente de zero, a disposição de armazenamento modifica as estatísticas, pois o contexto é alterado. Como as imagens possuem uma forma conhecida de armazenamento, por exemplo em linha x coluna, pode-se modificar esse formato para melhorar a taxa de compactação, por exemplo mudando-o para coluna x linha. Deve ser escolhida a

melhor forma, que pode variar de imagem para imagem, e armazena-la no arquivo compactado para sua posterior restauração.

Porém, a modificação da forma de armazenamento pode gerar uma perda de desempenho no processamento, pois em alguns casos necessita-se de operações aritméticas para efetuar a alteração e a decodificação da imagem, pois caso a imagem esteja armazenada por linha x coluna e necessite-se inverte-la para coluna x linha, será efetuada a operação aritmética da Equação 3.4, para obter a posição seqüencial (*offset*) do pixel dentro do novo arquivo de imagem, o que implica em perda de desempenho em velocidade.

$$pos\_cl\_pel = c * max\_l + l \quad (3.4)$$

onde:

pos_cl_pel = *offset* do pixel no arquivo de coluna x linha,

c = coluna do pixel no arquivo original,

max_l = número máximo de linhas do arquivo original,

l = linha do pixel no arquivo original.

OBS: O domínio dos valores para linha, coluna e *offset* inicia em zero.

Essa perda existirá também na exibição da imagem em monitor de vídeo, pois normalmente a memória de vídeo armazena linha após linha contigüamente, necessitando, caso a imagem não esteja armazenada por linha x coluna, que haja acessos a posições de memória não subseqüêntes, degradando o desempenho.

Atualmente, a maioria dos métodos de compactação utilizados em imagens são de uso geral, ou seja, não foram

desenvolvidos para esse tipo específico de dado. Exemplo disso são os formatos GIF (Graphics Interchange Format) e PCX, entre outros, que implementam os métodos LZW (Lempel, Ziv e Welch) e RLE (Run Length Encoding) respectivamente.

3.4 - ALGORITMOS

3.4.1 - RUN LENGTH ENCODING

O *Run Length Encoding* (RLE) é um método de compactação simples que codifica símbolos repetidos dispostos em seqüência. Uma seqüência é uma série de símbolos iguais, que são trocados no arquivo compactado por um código com o número de redundâncias seguido do símbolo original. Para imagens, em geral esse algoritmo não é eficaz, havendo a necessidade de modificar o algoritmo básico para que compense a sua utilização. Porém, para imagens de mapas digitalizados é um bom método, uma vez que o número de cores ou níveis de cinza necessários a quantizar na imagem é relativamente pequeno, ocasionando seqüências de redundância relativamente grandes. Algumas variações do algoritmo, para imagens, foram estudadas por Almeida (ALMEIDA, 1989).

Uma das variações mais utilizadas atualmente é o formato PCX (KAY, 1993), concebido para o PaintBrush da ZSoft, utilizado em microcomputadores da linha IBM-PC. É um formato de implementação simples, com um cabeçalho com informações

sobre a imagem, e consta do Módulo de Compactação de Imagens, pela sua importância para intercâmbio de dados entre softwares.

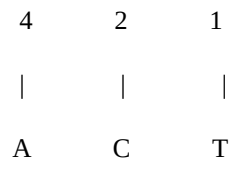
Algoritmo PCX

```
Ler caracter do arquivo
Se os dois bits de mais alta ordem do caracter lido estão com o valor 1
Então
    Número de bytes repetidos = Número correspondente aos
                                seis bits restantes
    Ler caracter do arquivo
    Caracter repetido = Caracter lido
Senão
    Número de bytes repetido = 1
    Caracter repetido = Caracter lido
Fim Se
Fim Algoritmo
```

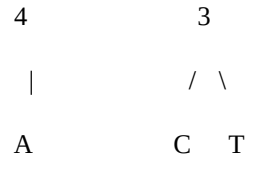
3.4.2 - CODIFICAÇÃO DE HUFFMAN

É um método estatístico, que associa códigos de tamanho fixo a símbolos, cujo algoritmo clássico (HUFFMAN, 1951) pode ser exemplificado da seguinte forma:

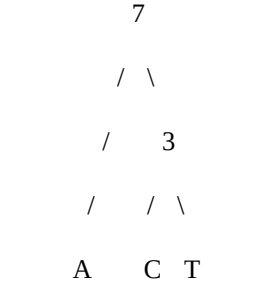
Suponha que a fonte a ser codificada é "ACATACA", com quatro "A", dois "C" e um "T". Essa situação é representada como se segue:



Combina-se os dois menos freqüentes símbolos em um, resultando na nova freqüência 2 + 1 = 3:



Repete-se o passo acima até que todos os símbolos estejam combinados em uma árvore:



Inicia-se no topo (raiz) dessa árvore de codificação, e navega-se para o símbolo que se deseje codificar. Se for para a esquerda, envie um bit '0'; de outra forma envie um bit '1'. Assim, "A" é codificado por '0', "C" por '10', "T" por '11'. Logo, "ACATACA" será codificado em dez bits, '0100110100'.

Para decodificar é necessário conhecer a árvore de codificação (estatísticas) original, que deverá estar no arquivo

compactado. Esse método é denominado de Huffman estático, utilizado pelos utilitários *pack* e *unpack* do UNIX System V da AT&T.

Uma modificação desse algoritmo clássico é o modelo adaptável, proposto por (FALLER, 1973). Independentemente, (GALLAGER, 1978) também propôs o mesmo método, e o algoritmo proposto por ambos foi melhorado por (KNUTH, 1985). Assim o algoritmo que implementa o Huffman adaptável ou dinâmico é conhecido por algoritmo FGK (Faller, Gallager, Knuth).

3.4.3 - CÓDIGO ARITMÉTICO

É um método estatístico, cuja concepção original foi proposta por (ELIAS, 1963). As primeiras implementações práticas datam de 1976 (RISSANEM, 1976) e (PASCO, 1976). Uma implementação em linguagem C é descrita por (WITTEN, 1987). É um método patenteado pela IBM.

A codificação de Huffman codifica um símbolo com sua entropia aproximada para um número inteiro de bits. A codificação aritmética tem desempenho melhor em taxa de compactação por essa restrição não existir.

A codificação aritmética não utiliza a metodologia de codificar um símbolo em um código específico, transformando a seqüência completa de símbolos da fonte em um único número real. Isso só é possível em computadores através de artifícios de programação, pois, a aritmética de computador utiliza um número finito e pequeno de bits para implementar os tipos de dados.

O processamento da codificação aritmética resulta em um número menor que 1 e maior ou igual a 0, que pode ser decodificado para a seqüência de símbolos original.

Para gerar esse número, são atribuídos aos símbolos um conjunto de probabilidades. A seqüência "MARCOPERNA", por exemplo, tem uma distribuição de probabilidade mostrada na Tabela 3.1. A cada símbolo é, então, atribuída uma subdivisão da faixa de valores entre 0 e 1, correspondendo, em tamanho, a sua probabilidade de ocorrência, como mostra a Tabela 3.1.

TABELA 3.1 : Probabilidade e faixa dos símbolos.

Símbolo	Probabilidade	Faixa
A	2/10	0.0 <= f < 0.2
C	1/10	0.2 <= f < 0.3
E	1/10	0.3 <= f < 0.4
M	1/10	0.4 <= f < 0.5
N	1/10	0.5 <= f < 0.6
O	1/10	0.6 <= f < 0.7
P	1/10	0.7 <= f < 0.8
R	2/10	0.8 <= f < 1.0

Com a tabela de freqüências pronta inicia-se a codificação construindo duas tabelas, que possuem o número gerado pelo menor valor e pelo maior valor respectivamente. O primeiro símbolo, a letra "M" está contida na faixa 0.4 <= f < 0.5, o valor mínimo e o máximo são colocados nas tabelas de menor e

maior valor respectivamente, como mostra a Tabela 3.2 A tabela é finalizada de acordo com o seguinte algoritmo :

```
menor = 0.0;
maior = 1.0;
enquanto não terminar a seqüência de símbolos faça
{
  faixa = maior - menor;
  maior = menor + faixa * maior_da_faixa(símbolo);
  menor = menor + faixa * menor_da_faixa(símbolo);
}
```

TABELA 3.2: Codificação Aritmética.

Símbolo	Menor valor	Maior valor
	0.0	1.0
M	0.4	0.5
A	0.40	0.42
R	0.496	0.520
C	0.5008	0.5032
O	0.50224	0.50248
P	0.502408	0.502432
E	0.5024152	0.5024176
R	0.50241712	0.50241760
N	0.502417360	0.502417408
A	0.5024173600	0.5024173690

Onde "maior_da_faixa(símbolo)" é uma função que retorna o maior valor da faixa do símbolo, passado como

parâmetro, de acordo com a Tabela 3.1, e "menor_valor_faixa(símbolo)" retorna o menor valor. A cada iteração da estrututa "enquanto", os valores calculados de maior e menor são colocados na Tabela 3.2, em seus símbolos correspondentes.

O menor valor calculado para o último símbolo é a codificação final da seqüência, ou seja, 0.5024173600.

Uma prova matemática formal de que o processo converge para o limite da entropia é dada por (PASCO, 1976).

Uma implementação em linguagem C é descrita por (NELSON, 1991), utilizando modelos de Markov superiores a zero, o que torna esse método bastante apreciável, considerando que memória e velocidade não são problemas.

3.4.4 - CODIFICAÇÃO LZW

Esse algoritmo foi proposto por Lempel e Ziv, em 1977; melhorado em 1978 (LEMPEL, 1978); e modificado por (WELCH, 1984) pesquisador da Sperry, cuja fusão com a Burroughs gerou a UNISYS. É um método adaptativo de um passo que associa seqüências de tamanho variável de símbolos a códigos de tamanho fixo ou variável. Essas características levam o método ser chamado de **Método Baseado em Dicionário**.

É também um método dependente do contexto da ocorrência do símbolo na fonte. É patenteado pela UNISYS, tendo a IBM também patenteado sua versão.

O algoritmo consiste em preparar uma tabela que pode conter milhares de itens. Inicialmente coloca-se da posição zero

até a posição 255 os usuais 256 caracteres. Lê-se vários símbolos da fonte, e pesquisa-se na tabela pela mais longa correspondência. Supondo que a mais longa correspondência é dada pela seqüência "ABC". Coloca-se a posição de "ABC" na tabela. Le-se o próximo símbolo da fonte. Se ele for "D", então coloca-se uma nova seqüência "ABCD" na tabela, e reinicia-se o processo com o símbolo "D". Se houver *overflow* da tabela, descarta-se o mais antigo ítem ou, preferencialmente, o menos usado. O Figura 3.3 mostra o fluxo do algoritmo.

As variações básicas de implementação consistem em usar códigos de tamanho fixo (LZW estático) ou variável (LZW dinâmico). O conceito de estático e dinâmico aqui é diferente do conceito dos métodos estatísticos. Os códigos de tamanho fixo geralmente não fornecem um bom desempenho comparados com os de tamanho variável. Normalmente começa-se a codificação com códigos de 9 bits e a medida que o algoritmo necessita de mais bits, aumenta-se o código em um bit, passando para 10 bits e assim sucessivamente.

O formato GIF (*Graphics Interchange Format*) (COMPUSERVE, 1987), da CompuServe, usa esse método com uma variação de 3 a 12 bits para o tamanho do código. Outra melhoria ao método é a limpeza da tabela de itens sempre que necessário, reiniciando-se a tabela como estava no início da codificação. Uma implementação em linguagem C foi descrita por (NELSON, 1989) e melhorada por (REGAN, 1990).

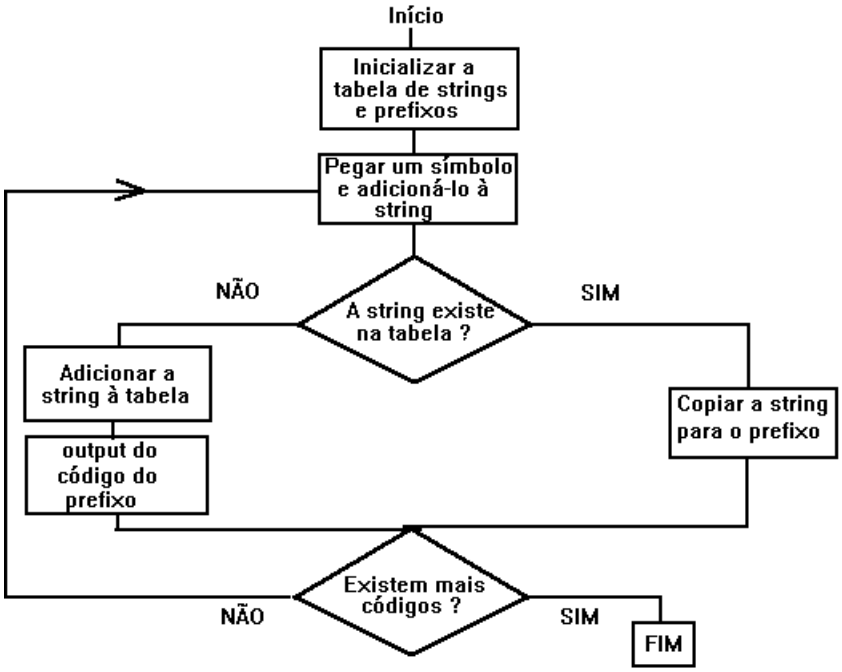


FIGURA 3.3: Fluxograma de compactação LZW.

CAPÍTULO 4

MÓDULO DE COMPACTAÇÃO

4.1 - CONSIDERAÇÕES

O Módulo de Compactação contém dois métodos de compactação, que foram selecionados a partir da análise dos testes preliminares dos algoritmos do Capítulo 3, mostrados no Capítulo 6, sendo um deles escolhido pelo desempenho de taxa de compactação e o outro por questões de portabilidade. Os dois métodos de compactação selecionados foram:

(i) Código aritmético com modelo de Markov com ordem 1: escolhido pelo desempenho nos testes preliminares;

(ii) Algoritmo PCX: escolhido por ser um formato de compactação simples de implementar, como apresentado no Capítulo 3, e utilizado amplamente em *softwares* de tratamento ou manipulação de imagens.

Nestes dois métodos, foram realizadas modificações visando melhorar a taxa de compactação de imagens. Essas modificações estão descritas nos itens seguintes.

4.2 - CÓDIGO ARITMÉTICO MODIFICADO

O método utilizado é o código aritmético com modelo de Markov com ordem 1. Esse método realiza um controle da compactação através do monitoramento da taxa de compactação.

Esse controle permite que, caso a taxa de compactação se encontre abaixo de um valor pré-determinado, seja efetuada uma limpeza na tabela de estatísticas para possibilitar uma melhoria da taxa de compactação, se possível.

Outra característica do monitoramento é o momento de checagem da taxa de compactação. Normalmente é efetuada cada vez que uma determinada quantidade de símbolos é lida, com intervalo constante.

A rotina, em linguagem C, apresentada a seguir (NELSON, 1991), realiza um monitoramento normal, sendo chamada a um intervalo de 256 símbolos. Essa rotina efetua a modificação quando a taxa de compactação se encontra abaixo de 10%, ou seja, se o tamanho do arquivo gerado estiver com 90% da quantidade de símbolos já lidos. Nesse caso, a taxa de compactação está aumentando, o que indica que não está mais sendo realizada compactação. Desta forma, a rotina efetua a reinicialização das estatísticas no modelo, visando permitir que estatísticas mais atualizadas possam influir melhor na compactação.

ROTINA DE MONITORAMENTO

```
booleano checa_compactação ( arq_entrada, arq_saida )
{
    static long local_input_marker = 0L;
    static long local_output_marker = 0L;
    long total_input_bytes;
    long total_output_bytes;
    int local_ratio;
```

```
    total_input_bytes = posicao ( arq_entrada ) -
                        local_input_marker;
    total_output_bytes = posicao ( arq_saida );
    total_output_bytes -= local_output_marker;
    if ( total_output_bytes == 0 )
        total_output_bytes = 1;
    local_ratio = (int)( ( total_output_bytes * 100 ) /
                         total_input_bytes );
    local_input_marker = posicao ( arq_entrada );
    local_output_marker = posicao ( arq_saida );
    return( local_ratio > 90 );
}
```

A modificação proposta reside na forma em que é feito o controle da taxa de compactação. Leva-se em consideração a variação da taxa de compactação, não esperando que a mesma caia abaixo de um limite pré-determinado, ou seja: caso a taxa de compactação esteja diminuindo, a nova rotina efetua a modificação, não tendo que esperar um limite pré-determinado, como o de 10% da rotina anterior.

ROTINA DE MONITORAMENTO MODIFICADA

```
booleano checa_compactacao( FILE *input, BIT_FILE *output )
{
    static long local_input_marker = 0L;
    static long local_output_marker = 0L;
```

```
    static long ratio_avg = -1l;
    int ok;
    long total_input_bytes, total_output_bytes, local_ratio;

    total_input_bytes = f_tell_in( input ) -
                            local_input_marker;
    total_output_bytes = f_tell_out( output->file );
    total_output_bytes -= local_output_marker;
    if ( total_output_bytes == 0l )
      total_output_bytes = 1l;
    local_ratio = ( ( total_output_bytes * 100l ) /
                        total_input_bytes );
    local_input_marker = f_tell_in( input );
    local_output_marker = f_tell_out( output->file );
    if (ratio_avg == -1l)
       ratio_avg = local_ratio;
  ok = ((long)((double)ratio_avg * 1.50) > local_ratio &&
           local_ratio <= 90);
  ratio_avg = (ratio_avg + local_ratio) / 2l;
  return (!ok);
}
```

Com relação ao intervalo de símbolos lidos para ativação da checagem, é proposto testes para sua aferição. Pois, é necessário verificar qual intervalo resulta em uma melhor taxa de compactação. Só é possível afirmar que: quanto maior o intervalo, mais rápida é a compactação, pois implica em menos processamento.

4.3 - ALGORITMO PCX MODIFICADO

A utilização do algoritmo de compactação PCX, uma variação do RLE com uma estrutura de cabeçalho com informações a respeito da imagem, é bastante utilizado, apesar de seu fraco desempenho em relação a taxa de compactação, como formato de armazenamento compactado de imagens, devido a sua facilidade de implementação. No caso de imagens de satélite pode também ser utilizado, pois desde os primórdios do processamento de imagens de satélite vem-se utilizando o RLE com algumas modificações, como em Almeida (ALMEIDA, 1989).

O algoritmo PCX é simples, ver item 3.4.1, e neste trabalho é utilizado o tipo 5 com 8 bits para cada *pixel* e um plano de bits. Porém, só compacta símbolos de 8 bits de tamanho que não tenham os dois bits de ordem mais alta com o valor 1, estes são utilizados para controle da compactação. Desta forma conclui-se que, como esses dois bits correspondem a valores entre 192 e 255 (11000000_2 e 11111111_2), imagens que possuam símbolos isolados, entre esses valores, em quantidade pelo menos maior ou igual a metade do arquivo, não serão compactadas. Isto porque, para símbolos com valores nessa faixa, o algoritmo coloca no arquivo compactado um código de controle e, em seguida, coloca o símbolo emitido pela fonte, sem qualquer alteração, dobrando a quantidade de bits necessária para representar este símbolo.

As imagens de satélite, de um modo geral, possuem uma variação pequena de valores, como mostra o histograma da Figura 4.2. Exceções são feitas em imagens não-convencionais ou imagens

que já tenham sido submetidas a algum tipo de processamento. Porém, se essa faixa estiver situada justamente entre os valores 192 e 255, a imagem compactada será com certeza maior que a original. Em vista disto, é proposto neste trabalho, uma variação do algoritmo PCX em que imagens com histograma situado nessa faixa de valores (Figura 4.3), são pré-processadas utilizando o operador binário *not* para cada pixel, gerando uma imagem cujo histograma não está situado entre os valores 192 e 255 (Figura 4.4). Por exemplo, o valor 192 (11000000_2) passa a ser representado, após a operação binária, pelo valor 63 (00111111_2) que não prejudicará o desempenho do algoritmo.

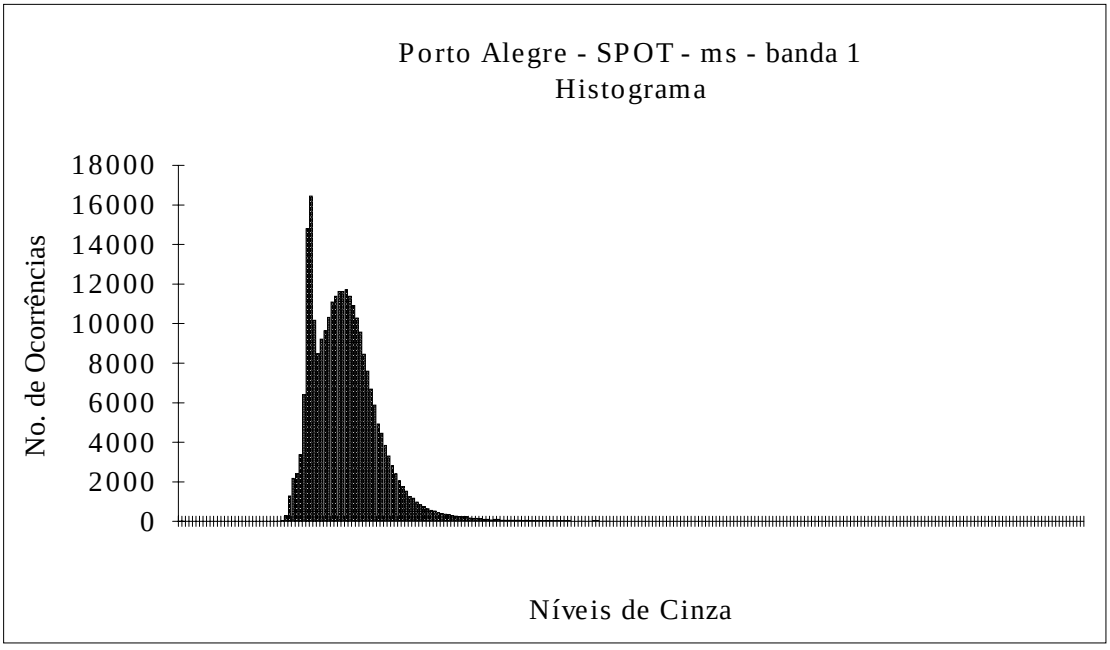


FIGURA 4.2 : Histograma da imagem PA1.I.

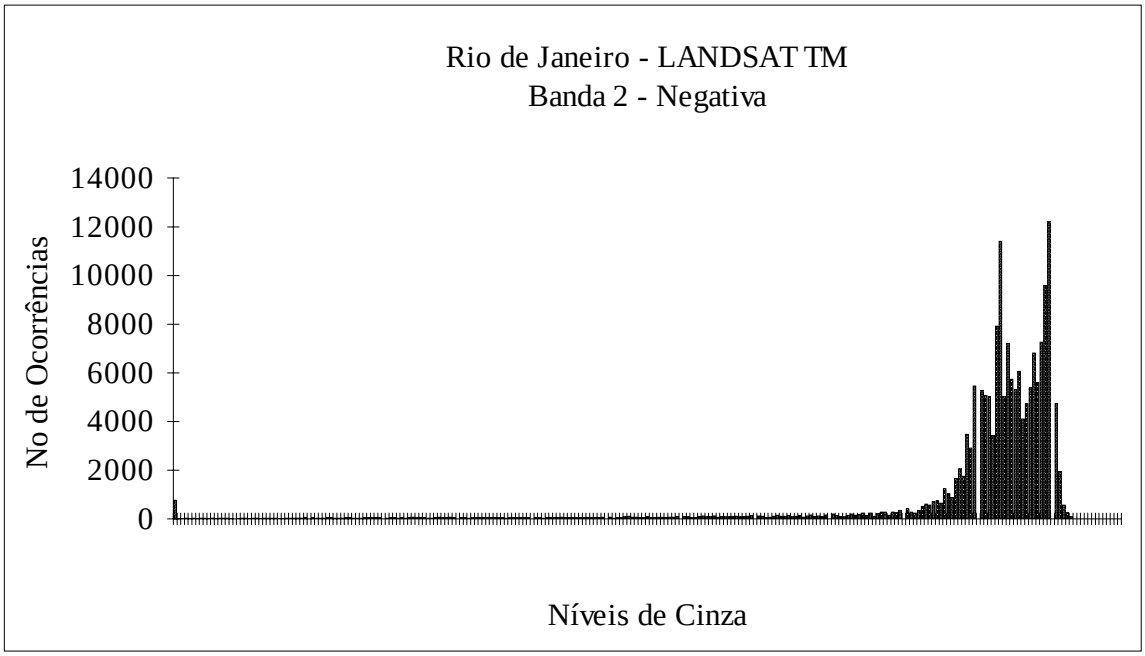


FIGURA 4.3 : Histograma da imagem negativa RJ2_NEG.I.

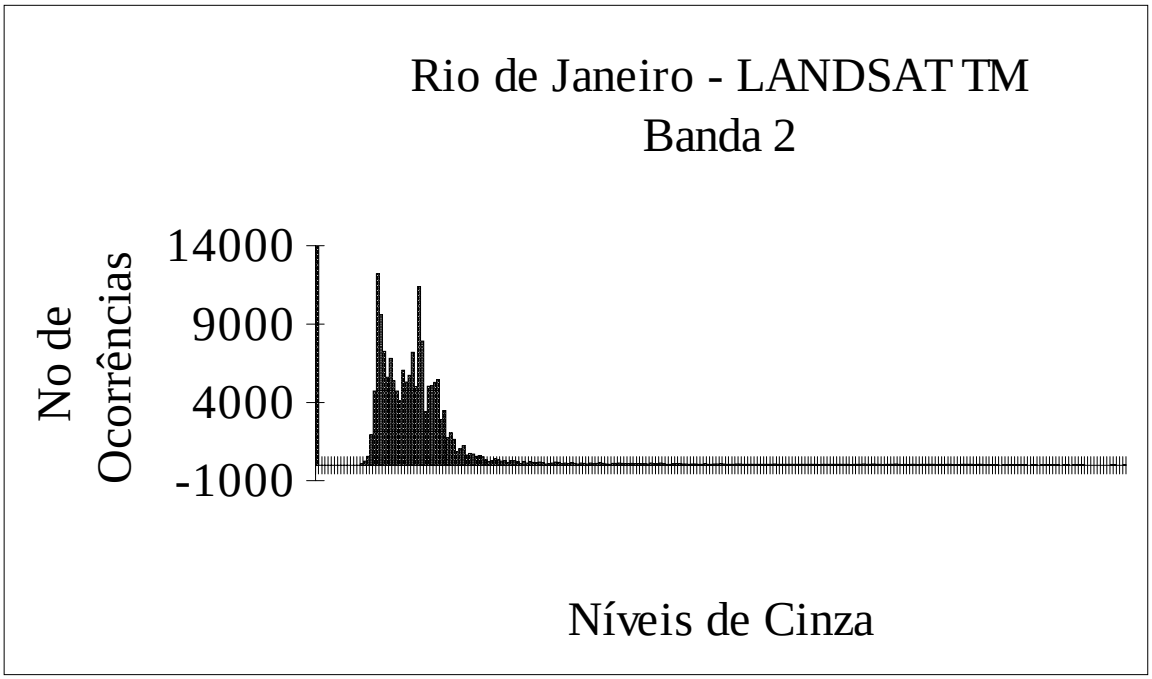


FIGURA 4.4 : Histograma da imagem RJ2.I.

4.4 IMPLEMENTAÇÃO

O módulo implementa o código aritmético com ordem 1 e o formato PCX. O programa de compactação pelo código aritmético, leva o nome de MAP e deve ser compilado, em MS-DOS no modelo de memória LARGE, com todas as otimizações possíveis para velocidade, ou em UNIX apenas com otimizações de velocidade. O programa que gera o formato PCX tem as mesmas necessidades de compilação e leva o nome de IMG2PCX.

O arquivo compactado gerado pelo Módulo, utilizando código aritmético, possui um cabeçalho onde estão descritas informações a respeito da imagem, como por exemplo: a região geográfica da imagem, o processo de obtenção, o número de linhas, colunas, número de níveis de cinza, a data e a hora. No caso do formato PCX, o cabeçalho segue sua padronização. A Tabela 4.1 detalha a estrutura do cabeçalho do Módulo utilizando o código aritmético.

A linha de comando a seguir deve ser digitada no *prompt* do interpretador de comandos do sistema operacional utilizado, para utilização do Módulo de Compactação.

```
MAP opção arqmap [arqimagem i j k região origem d m a h m]

  onde :   opção : u = compactação, e = descompactação;
           arqmap = arquivo compactado
           arqimagem = arquivo com a imagem original
           i, j, k = número de linhas, colunas e níveis de cinza;
           região = região geográfica da imagem
```

origem = processo de obtenção da imagem

d, m, a, h, m = dia, mês, ano, hora, minuto de obtenção da imagem.

TABELA 4.1 : Cabeçalho do arquivo de imagem compactado.

BYTE	TIPO	DESCRIÇÃO
0..12	ASCII terminado com 0	id da imagem
13..14	inteiro com MSB primeiro	número de linhas
15..16	inteiro com MSB primeiro	número de colunas
17..18	inteiro com MSB primeiro	n. de níveis de cinza
19..38	ASCII terminado com 0	região geográfica
39..40	inteiro com MSB primeiro	lat. - graus - canto NW
41..42	inteiro com MSB primeiro	latitude - minutos
43..44	inteiro com MSB primeiro	latitude - segundos
45	ASCII	N = norte, S = sul
46..47	inteiro com MSB primeiro	longitude - graus
48..49	inteiro com MSB primeiro	longitude - minutos
50..51	inteiro com MSB primeiro	longitude - segundos
57	inteiro	dia - data da imagem
57	inteiro	dia - data da imagem
58	inteiro	mês
59..60	inteiro com MSB primeiro	ano
61	inteiro	hora
62	inteiro	minuto
63..82	ASCII terminado com 0	aparelho de origem
83..99	reservado	reservado

CAPÍTULO 5

IMPLEMENTAÇÃO

5.1 - INTRODUÇÃO

Neste Capítulo estão descritas algumas características necessárias do ambiente de *software* e *hardware* para o desenvolvimento e utilização do módulo de compactação implementado neste trabalho.

5.2 - LINGUAGEM UTILIZADA

A codificação foi feita em linguagem C, escolhida por ser uma linguagem estruturada de bom desempenho na execução dos programas gerados, voltados para *software* básico ou científico, salvo *software* científico com muitos cálculos matemáticos, de ponto flutuante, onde o FORTRAN continua sendo muito utilizado. O conhecimento de arquitetura de computadores é um ponto fundamental para implementar corretamente operações orientadas a bit, tais como o *shift*, o *and* e o *or*, entre outros procedimentos de programação necessários à elaboração de um programa nos moldes do apresentado neste trabalho. A escolha da linguagem C para implementar os algoritmos foi feita levando-se em conta alguns fatores em particular :

(i) C tornou-se uma linguagem padrão entre cientistas e profissionais de software básico;

(ii) os compiladores de linguagem C estão disponíveis para várias classes de computadores, desde microcontroladores até supercomputadores;

(iii) É uma linguagem com poucas construções usadas como elementos básicos de linguagem, o que torna fácil a tradução do código para outra linguagem, como Pascal.

(iv) A eficiência do código gerado e sua portabilidade.

A linguagem C, embora tida como uma linguagem portátil, pode nem sempre apresentar esta característica. Um programa que compila e executa em uma determinada máquina pode não compilar ou ser executado em outra. De forma análoga, é possível que um programa funcione com um determinado compilador de um fabricante e não funcione com o de outro em uma mesma máquina. É importante lembrar que C não é portátil, e sim que pode ser portátil. O código encontrado neste trabalho foi desenvolvido visando a portabilidade.

As funções da biblioteca utilizada têm que ser portáteis. Embora a linguagem C seja muito bem definida em (KERNIGHAN, 1978), que foi utilizado como padrão da linguagem até o final da década de 80, as bibliotecas (*libraries*) a serem utilizadas não foram definidas.

Somente em 1990, o **American National Standards Institute** publicou uma padronização das bibliotecas e extensões da linguagem, o padrão ANSI XJ11.34.

Para compilar os programas de compactação é necessário um compilador que siga o padrão ANSI. Embora muitas máquinas possam ter somente o padrão K&R, isso não chega a ser um

problema, pois é possível fazer pequenas alterações no programa para que ele seja compilável.

Com a padronização das bibliotecas, os itens restantes para portabilidade são três:

(i) tamanho dos tipos básicos de dados;

(ii) compiladores compatíveis;

(iii) forma de armazenamento dos tipos básicos de dados na memória.

(i) A maioria dos conflitos de tipos de dados surgem quando há a mudança entre uma máquina de 16 bits e uma de 32 bits. Porém é fácil controlar esta mudança. Apesar do tamanho do tipo básico inteiro mudar de 16 para 32 bits, ambas as máquinas possuem o tipo de dados "short int", de 16 bits, assim como o tipo de dados "long int", de 32 bits. Logo, nos casos em que o tamanho do tipo de dados inteiro fizer diferença, é possível contornar o problema através do uso desses dois outros tipos de dados de tamanho fixo. Entretanto, caso não haja necessidade de se ter esse controle, deve-se usar o tipo inteiro comum "int", por permitir sempre o acesso mais rápido ao dado contido em sua posição de memória. Esse acesso é mais eficiente quando se busca dados que possuam o tamanho da palavra utilizada na máquina. Por exemplo, numa máquina de 16 bits (palavra de 16 bits) o tipo "int" possui 16 bits. No caso de "short int" e "long int" só há eficiência quando utilizados em máquinas de 16 e 32 bits respectivamente.

Na grande maioria das máquinas utilizadas atualmente, os compiladores C implementam o tipo de dados "char" com o

tamanho de 8 bits, portanto não há necessidade de maiores controles. Entretanto, deve-se utilizar a declaração "unsigned char" para evitar problemas com o sinal desse tipo de dado, pois a maioria dos compiladores C implementam o tipo "char" com sinal, num intervalo de -128 a +127.

(ii) O segundo problema a tratar é a incompatibilidade entre compiladores. Vai desde o uso de padrões diferentes, como o ANSI e o K&R, até diferenças entre versões de um mesmo fabricante, passando por formas de solucionar problemas, inerentes a uma determinada máquina, encontradas por fabricantes diferentes. Por exemplo, a forma de implementar ponteiros nos diversos modelos de memória, para compilação, necessários para implementações em MS-DOS, adotada por diferentes fabricantes de compiladores.

(iii) A forma de armazenamento dos tipos básicos de dados muda de acordo com o processador utilizado pelo computador. No caso da linha INTEL 80x86, o tipo "short int" é armazenado com o byte menos significativo (LSB) primeiro, enquanto na linha MOTOROLA é feito justamente o inverso.

No que tange ao algoritmo de compactação e descompactação essa diferença não importa. Como o Módulo de Compactação gera no arquivo de imagem compactada um cabeçalho contendo informações a seu respeito, que deverão ser interpretadas corretamente qualquer que seja o processador, é necessário padronizar a posição do byte menos significativo ou não usar os tipos de dados "short int" e "long int", passando a usar somente os tipos "float" e "double" que são padronizados pelo IEEE, e "char",

que possui somente um byte (na maioria considerável de computadores).

No Módulo de Compactação foi padronizado o uso do byte mais significativo (MSB) como o primeiro a ser armazenado na memória, no caso do uso do "short int". Não foi necessário usar o "long int". O método utilizado para implementar essa padronização foi o do uso de um vetor de dois elementos do tamanho de um byte cada, no qual foi armazenado no índice 0 o byte mais significativo e no índice 1 o menos significativo.

Existem várias formas de colocar esses bytes na posição correta, uma delas é o uso de operadores orientados a bit, que foi descartado por problemas de portabilidade. O método escolhido segue a Equação 5.1.

```
v = (unsigned short) (v / 256 + 256 * (v % 256));  (5.1)
```

onde : v = valor do byte.

Outra característica importante a ser ressaltada é a forma de armazenamento de estruturas de dados do tipo registro, que é representada em C pelo *token* ***struct*** e em Pascal por ***Record***. Os compiladores C possuem duas formas de armazenamento de registro: (i) por alinhamento de *byte* e, (ii) por alinhamento de *word* (palavra do processador).

Quando é feita opção por alinhamento de *byte*, o registro passa a possuir todo o seu tamanho preenchido por dados sem espaços livres entre si. No caso de alinhamento por *word*, o registro passa a alinhar seus campos por múltiplos da palavra do

processador, ou seja, um computador de 16 bits possui uma palavra de 16 bits, logo um registro com dois campos do tipo *char*, que possui um *byte* cada, vai consumir 32 bits ou 4 *bytes* de armazenamento porque cada campo vai ser alinhado em 16 bits.

Essa característica é relevante quando se utiliza registros de informação em arquivos que vão ser lidos em ambientes diferentes dos que foram gerados. Pois, caso se utilize alinhamento de *byte* e tente-se ler o registro utilizando alinhamento de *word*, os dados vão ser lidos errados. No caso de utilizar alinhamento de *word* em um computador com palavra de 16 bits e depois utilizar em um com palavra de 32 bits, também haverá problema. O correto é sempre utilizar alinhamento de *byte* para gerar o arquivo.

5.3 - SISTEMA OPERACIONAL UTILIZADO

De acordo com a característica de portabilidade descrita no ítem anterior, o Módulo de Compactação pode ser gerado para pelo menos dois dos mais importantes sistemas operacionais: o **MS-DOS** (*Microsoft Disk Operation System*) (MICROSOFT, 1992); o **UNIX**; além, é claro dos sistemas operacionais de outros fabricantes que sejam compatíveis com os mencionados e possuam compiladores C.

Porém, problemas com a troca de sistema operacional ocorrem devido a forma como eles gerenciam o computador. Existe um grave problema de gerenciamento de memória no MS-DOS, que é a barreira de 640 kbytes. Alguns artifícios são utilizados por gerenciadores de memória e por alguns compiladores C. No

entanto, esses artifícios até o momento só resolveram o problema do tamanho do código executável, que já pode romper a barreira dos 640 kbytes. No caso dos algoritmos de compactação/compressão, alguns necessitam manipular grandes massas de dados em memória principal, e como o DOS não permite, é necessário utilizar o artifício de gerenciadores de memória.

Entretanto, nem sempre estes gerenciadores estão disponíveis, nem tampouco um compilador C com essas características, o que faz com que os algoritmos que precisem romper essa barreira só possam ser utilizados, no momento, em um sistema operacional como o UNIX, que não possui essa limitação.

O problema ocorre devido ao processador 8088, para o qual foi desenvolvido o MS-DOS, possuir um barramento de 8 bits para endereços. Isto não permitia gerenciamento de muita memória, podendo chegar aos 640 kbytes através de artifícios, mas não além. Com a evolução dos processadores, essa característica de gerenciamento de memória foi melhorada, sendo que em máquinas de 32 bits a quantidade de memória gerenciável diretamente é enorme. O MS-DOS, porém, não pôde evoluir devido ao compromisso de manter compatibilidade com os processadores antigos nos quais ele já era executado. O UNIX, por sua vez, não possuí esse compromisso por características da filosofia com a qual foi desenvolvido.

A estrutura de endereços segmentados da série de microprocessadores 80x86 é uma das mais complexas, e desnecessária, característica que diferencia computadores que a utilizam de arquiteturas simplificadas de outros computadores. No estágio atual do desenvolvimento do PC essa característica não

deverá ser mudada por muitos anos ainda. Os processadores 80386 e 80486, que a princípio permitem uma estrutura de programas mais simples e confiável, e os ambientes Windows e OS/2, que utilizam largamente o espaço de endereçamento virtual provido por esses processadores, tem complicado mais do que simplificado o trabalho do programador, que é obrigado a suportar não somente a nova arquitetura, mas muitos conhecimentos da antiga também.

Outra característica importante do sistema operacional necessária para que não ocorra o problema de limitação de memória é o conceito de memória virtual, em relação ao uso de mais memória principal do que a existente realmente, através da troca (*swap*) de blocos de memória principal com a memória secundária. Essa característica existe no UNIX sem necessidade de se utilizar qualquer artifício.

CAPÍTULO 6

TESTES E AVALIAÇÃO DOS RESULTADOS

6.1 - CONSIDERAÇÕES

Utilizando os métodos e as modificações expostos no Capítulo 4, foram realizados testes para validação e aferição, pois o comportamento da compactação varia de acordo com a natureza da imagem.

Foram utilizadas cinco imagens (ver Apêndice A) de três tipos diferentes para os testes : (i) imagem de satélite; (ii) imagem fotográfica e (iii) imagem digitalizada por varredura de mapa. A saber:

(i) RJ2.I : imagem multiespectral do satélite LANDSAT TM banda 2, do Rio de Janeiro com 361 linhas por 512 colunas e 256 níveis de cinza, com data de aquisição de 09 de setembro de 1986, cujo histograma é apresentado na Figura 4.4;

(ii) LENA.I : imagem fotográfica de 256 linhas por 256 colunas e 256 níveis de cinza, obtida por *scanner*, cujo histograma é apresentado na Figura 6.2;

XMAS67.I : imagem fotográfica colorida de 712 linhas por 488 colunas e 24 bits para cor (8 bits para vermelho, 8 para verde e 8 para azul), obtida por *scanner*, cujos histogramas de cada componente de cor estão nas Figuras 6.4, 6.5 e 6.6 .

(iii) PV.I : imagem colorida de mapa topográfico da Praia Vermelha no Rio de janeiro, com 450 linhas por 464 colunas

e 256 cores (quantizadas em 8 cores), obtida por *scanner*, cujo histograma é apresentado na Figura 6.3.

EQ.I : imagem colorida de mapa topográfico do Equador, com 337 linhas por 322 colunas e 256 cores (quantizadas em 16 cores), obtida por *scanner*, cujo histograma é apresentado na Figura 6.1.

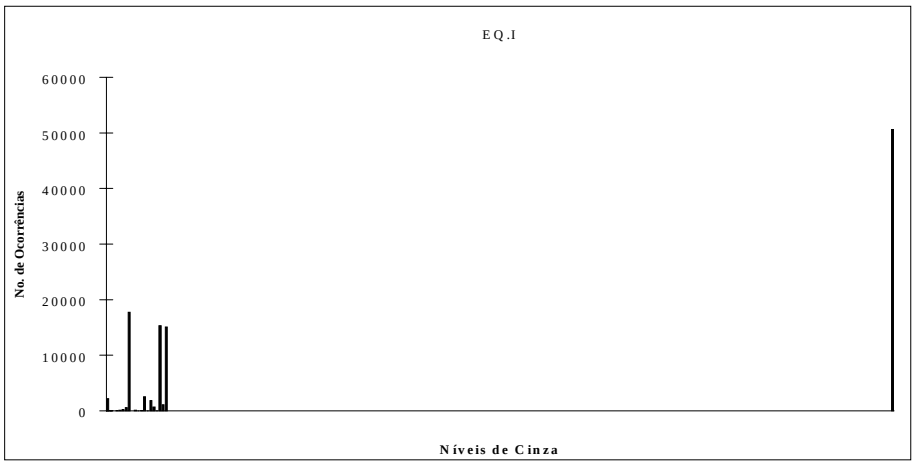


FIGURA 6.1 : Histograma da imagem EQ.I.

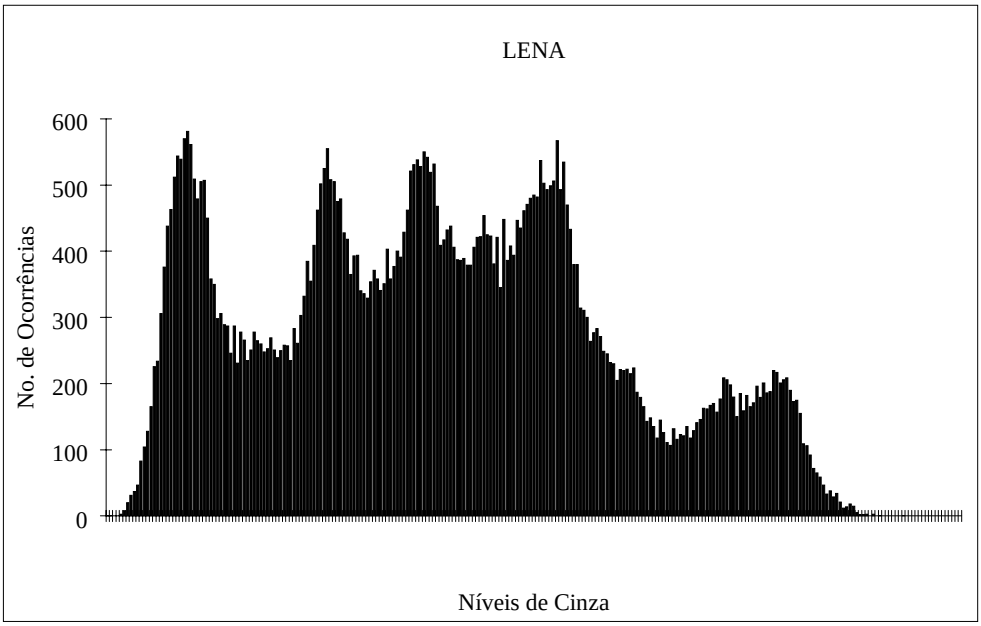


FIGURA 6.2 : Histograma da imagem LENA.I.

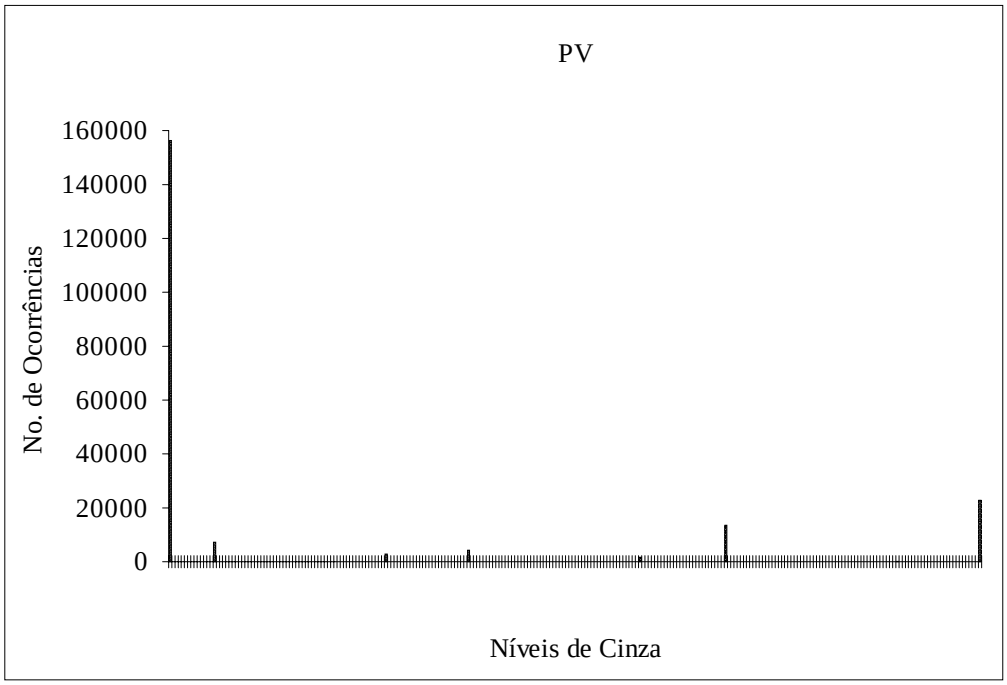


FIGURA 6.3 : Histograma da imagem PV.I.

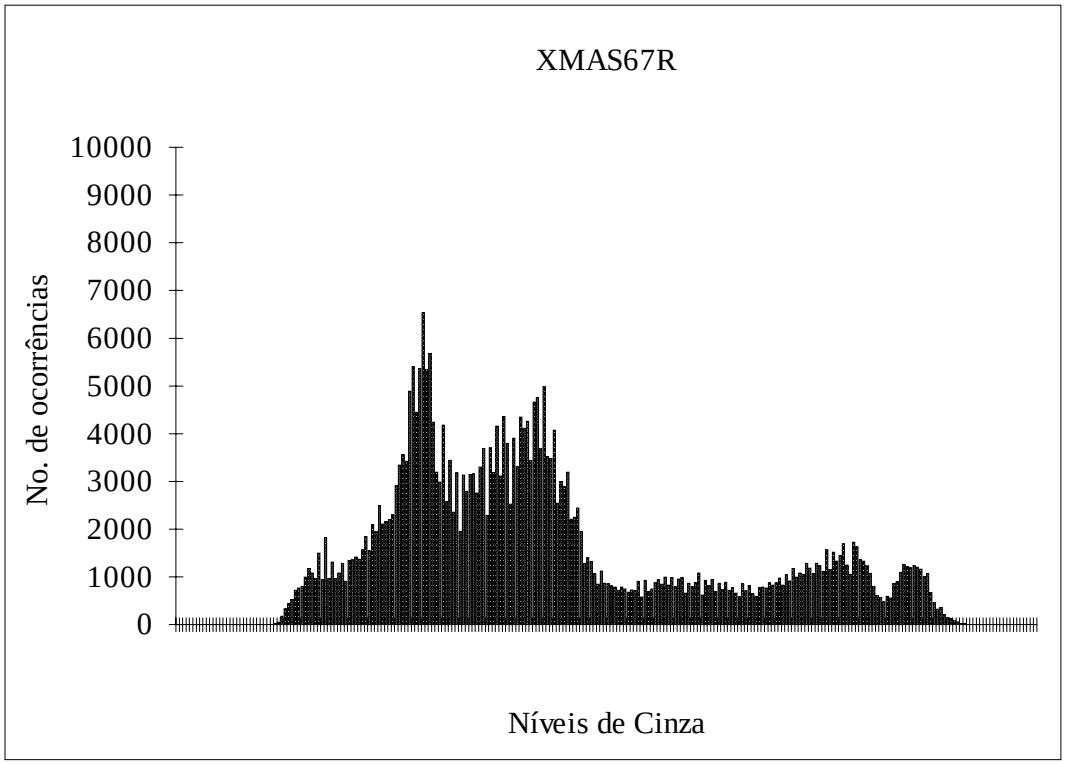


FIGURA 6.4 : Histograma da imagem XMAS67R.I.

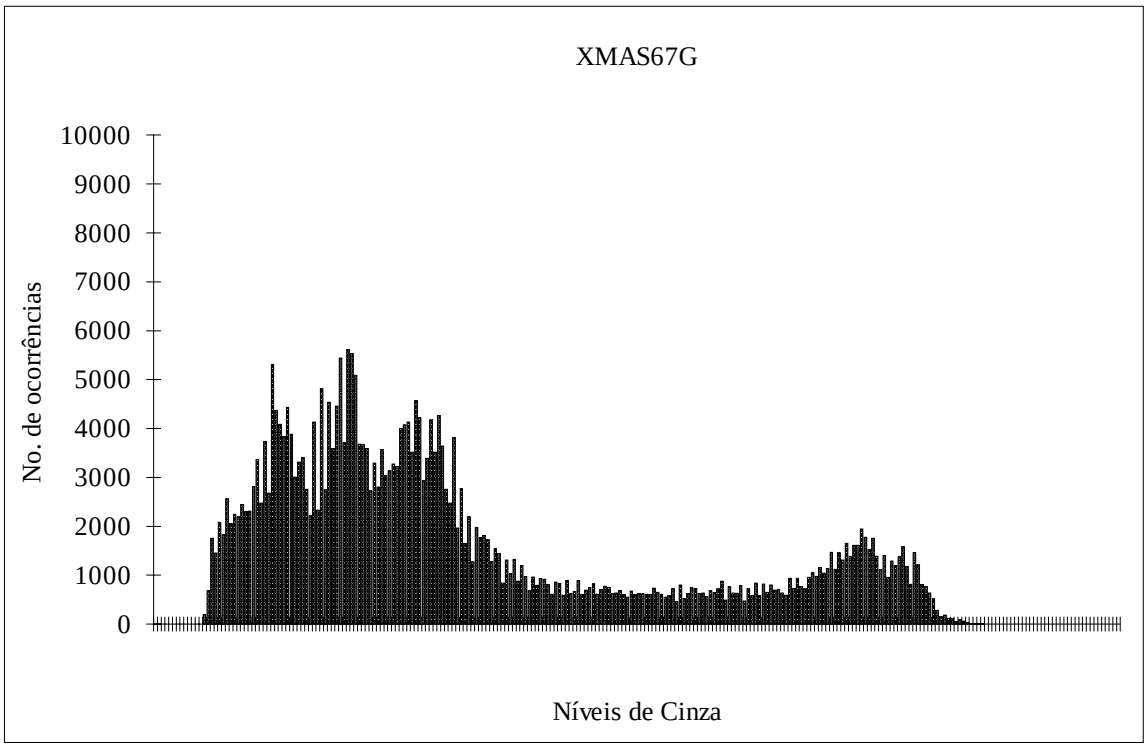


FIGURA 6.5 : Histograma da imagem XMAS67G.I.

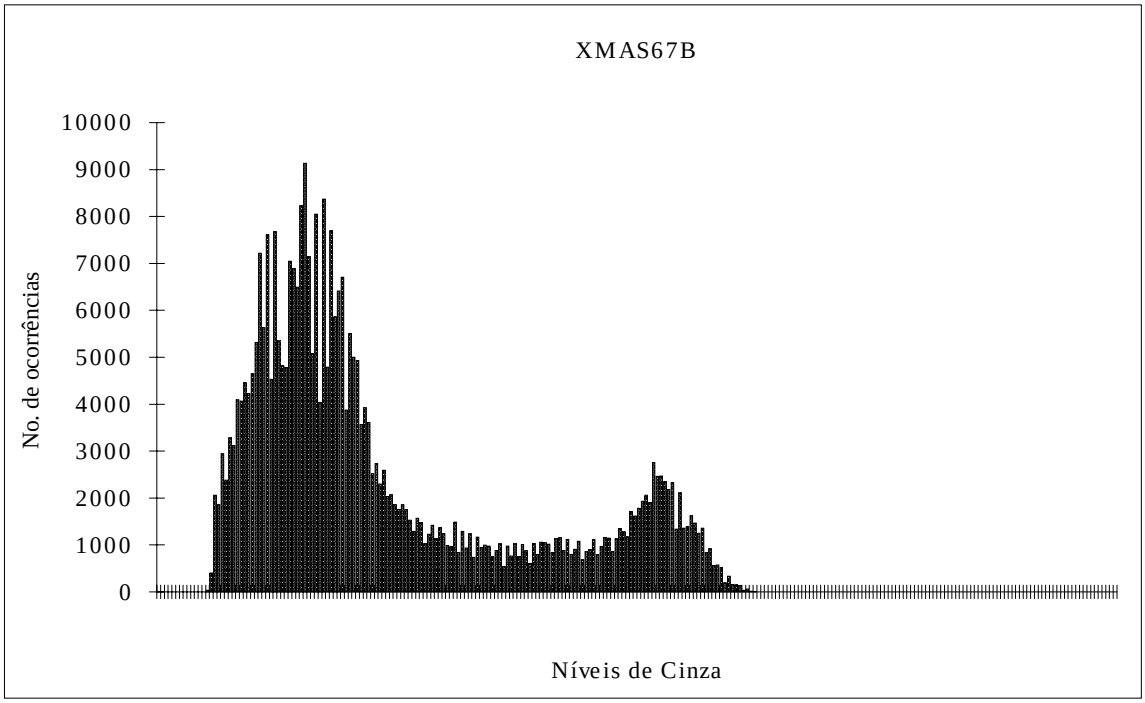


FIGURA 6.6 : Histograma da imagem XMAS67B.I.

Foram também efetuados testes utilizando além dos dois métodos expostos no Capítulo 4, mais seis métodos conhecidos, expostos no Capítulo 3 e o compactador proprietário PKZIP 2.04 da PKWare, que utiliza uma variação de (LEMPEL, 1977), visando demonstrar a escolha dos métodos PCX e Código Aritmético adaptativo com modelo de Markov ordem 1. Os métodos são os seguintes:

(i) RLE com 8 bits;

(ii) Huffman estático;

(iii) Huffman dinâmico (adaptativo);

(iv) LZW com limpeza de tabela e código variando de 9 a 14 bits;

(v) código aritmético adaptativo com modelo de Markov ordem 0, 2 e 3;

(vi) formato GIF, que utiliza o método LZW;

(vii) compactador PKZIP 2.04 da PKWare.

6.2 - TESTES DOS ALGORITMOS

O objetivo destes testes é a verificação do desempenho de taxa de compactação dos algoritmos e o seu comportamento em relação ao tipos de imagem analizados.

Com os três tipos de imagens foram efetuados testes com os algoritmos de compactação. A Tabela 6.1 mostra o tamanho dos arquivos compactados e as Figuras 6.7, 6.8, 6.9 e 6.10 mostram graficamente a tabela para análise.

Verifica-se que a imagem de satélite tem boa compactação pelos algoritmos estatísticos (Huffman e Aritmético) e baseados em dicionário (LZW e GIF), por ser um tipo de imagem com poucas seqüências de símbolos iguais, o que prejudica o RLE e o PCX. O desempenho do código aritmético com modelo ordem 1, mostra que um contexto ordem 1 tem desempenho melhor que contextos maiores ou mesmo seqüências (palavras) grandes de símbolos dos métodos baseados em dicionário, como o LZW, pois este tipo de imagem não possui uma grande quantidade de seqüências iguais com mais de dois símbolos (diferentes ou iguais). O modelo ordem 1 tem desempenho também melhor que os modelos ordem 0 aritmético e Huffman.

A imagem fotográfica LENA.I possui uma análise análoga a imagem de satélite, salvo em relação ao algoritmo baseado em dicionário LZW e GIF (que é uma versão do LZW), que teve um desempenho fraco, perdendo até para o RLE e o PCX, por características inerentes à imagem, o que faz juz a uma análise mais profunda em outro trabalho de pesquisa.

Os modelos ordem 2 e 3 tiveram bom desempenho apenas para as imagens PV.I e EQ.I, por serem imagens com poucos símbolos (cores) diferentes, com uma quantidade suficiente de símbolos estatísticamente dependentes para se conseguir esse resultado. Esses modelos só puderam ser executados sob o sistema operacional UNIX, pelo consumo de memória, o que torna ainda inviável o seu uso, fazendo juz a um estudo mais profundo em outro trabalho, que busque sua utilização de forma adaptável, ou seja, que o algoritmo consiga mudar de ordem do modelo e

reinicializa-lo de acordo com as necessidades da imagem que estiver sendo codificada.

TABELA 6.1 : Tamanho dos arquivos compactados de LENA, PV, RJ2 e EQ.

	LENA	PV	RJ2	EQ
RLE8	67768	98970	177794	110867
IMG	65536	208800	184832	108514
PCX	68946	83238	159277	102084
GIF	70980	34446	118938	33324
HUF	62447	40348	126552	31757
ARIT0	61119	34326	122115	31004
LZW	66804	31435	110397	30675
AHUF	62370	40313	123177	31462
ZIP2.04	57820	34811	105903	32928
ARIT1	51986	28593	87352	27466
ARIT2	54125	28041	90309	26854
ARIT3	55036	27463	96031	26689

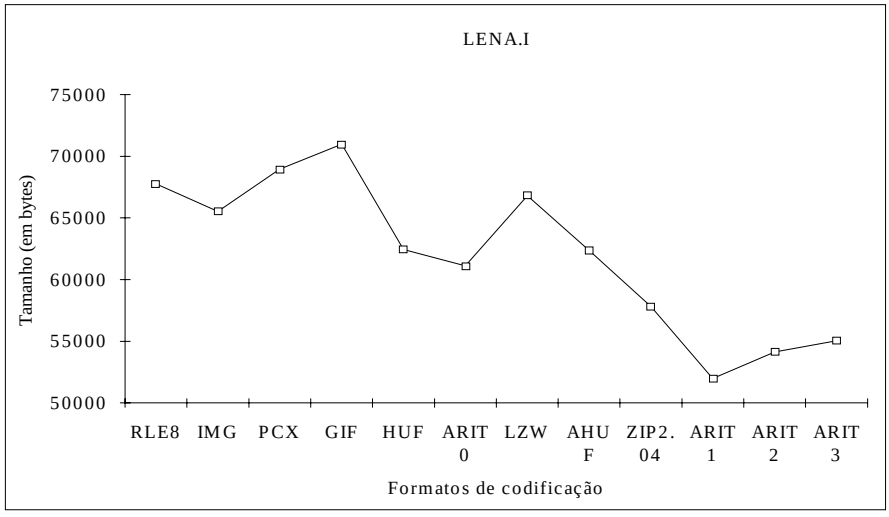


FIGURA 6.7: Desempenho dos compactadores sobre a imagem LENA.I.

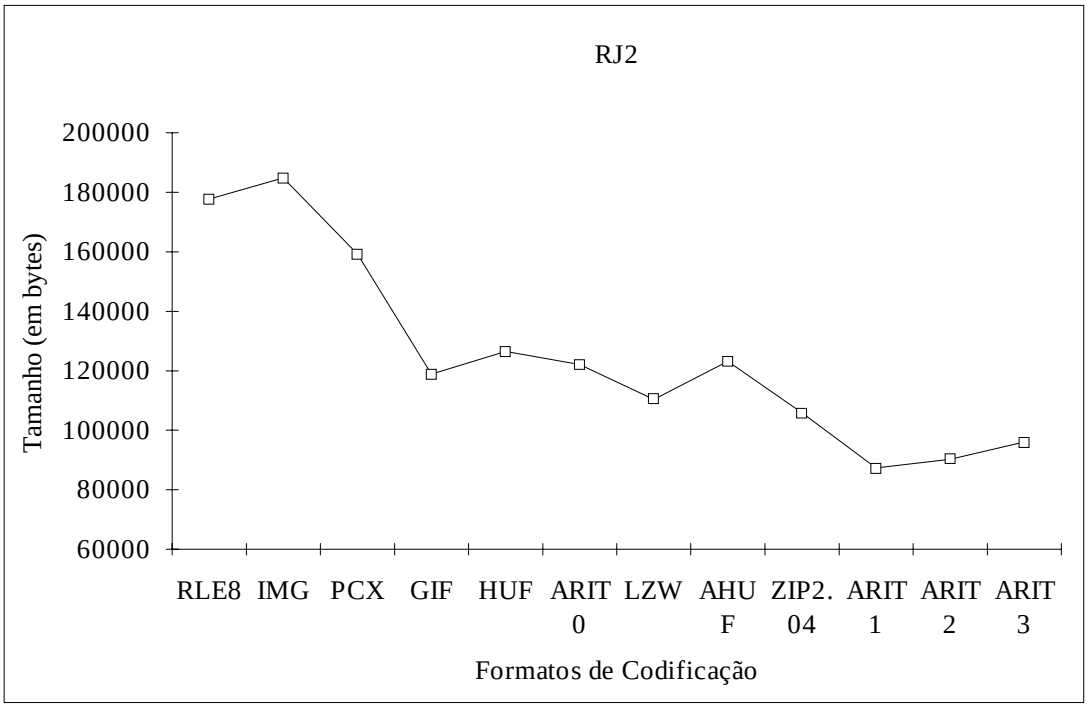


FIGURA 6.8 : Desempenho dos compactadores sobre a imagem RJ2.I.

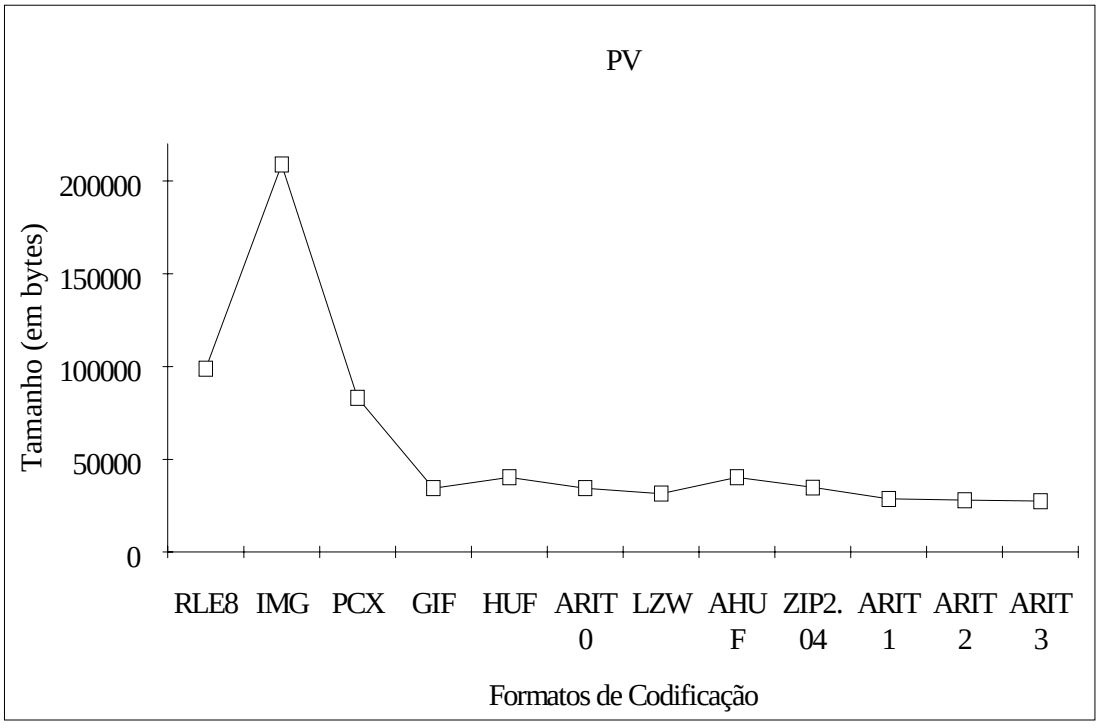


FIGURA 6.9 : Desempenho dos compactadores sobre a imagem PV.I.

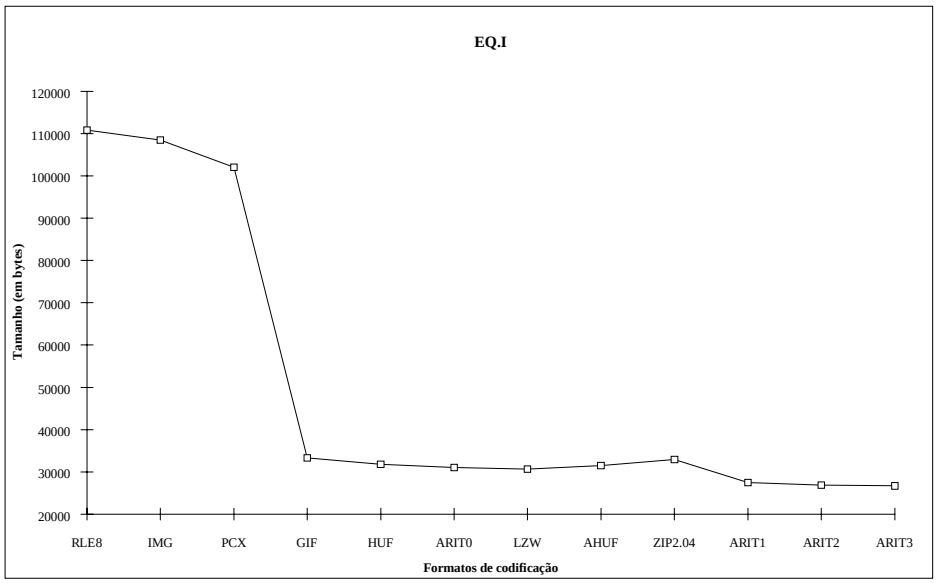


FIGURA 6.10 : Desempenho dos compactadores sobre a imagem EQ.I.

Os resultados expostos nas Figuras 6.11, 6.12 e 6.13 correspondentes a Tabela 6.2, com relação aos componentes RGB separados da imagem XMAS67, certificam que imagens fotográficas tem um comportamento semelhante diante dos vários compactadores e que, assemelham-se à imagem de satélite.

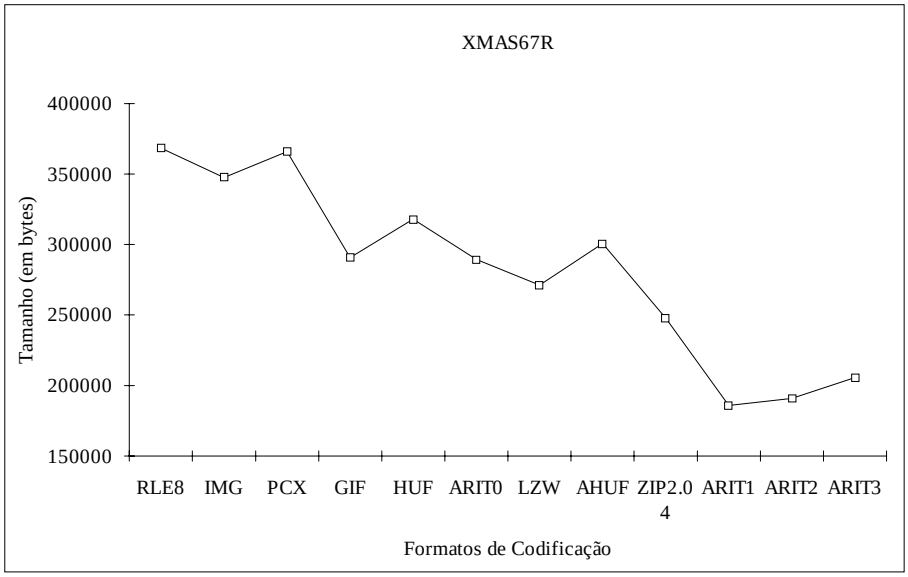


FIGURA 6.11 : Desempenho dos compactadores sobre a imagem XMAS67R.I.

TABELA 6.2 : Tamanho dos arquivos compactados de XMAS67.

	XMAS67R	XMAS67G	XMAS67B
RLE8	368467	367473	368540
IMG	347456	347456	347456
PCX	365803	345927	324817
GIF	290678	286980	251464
HUF	317905	316464	291605
ARIT0	289467	288407	262498
LZW	271261	270960	231569
AHUF	300630	300101	274781
ZIP2.04	247718	244658	217737
ARIT1	185900	184947	162705
ARIT2	190848	189662	165214
ARIT3	205560	203427	176417

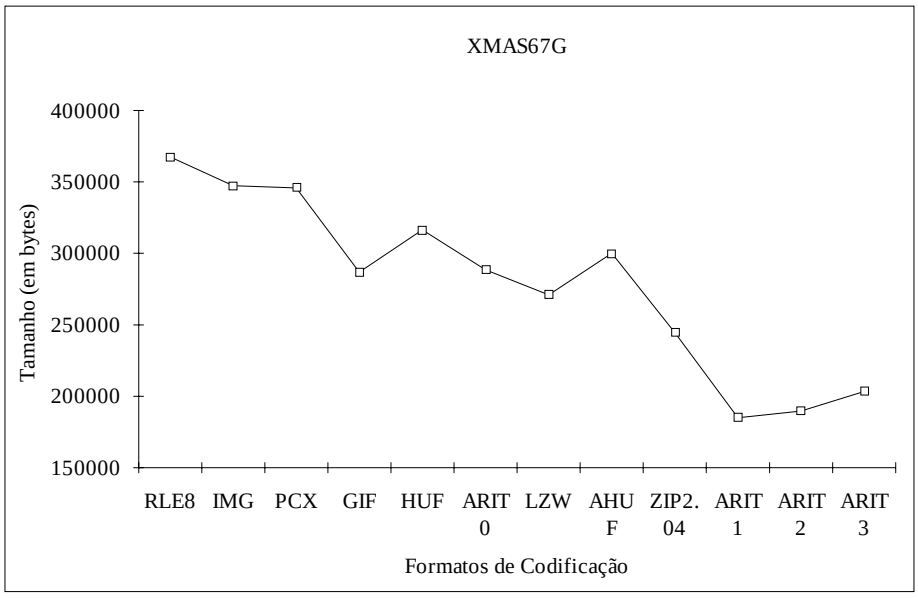


FIGURA 6.12 : Desempenho dos compactadores sobre a imagem XMAS67G.I.

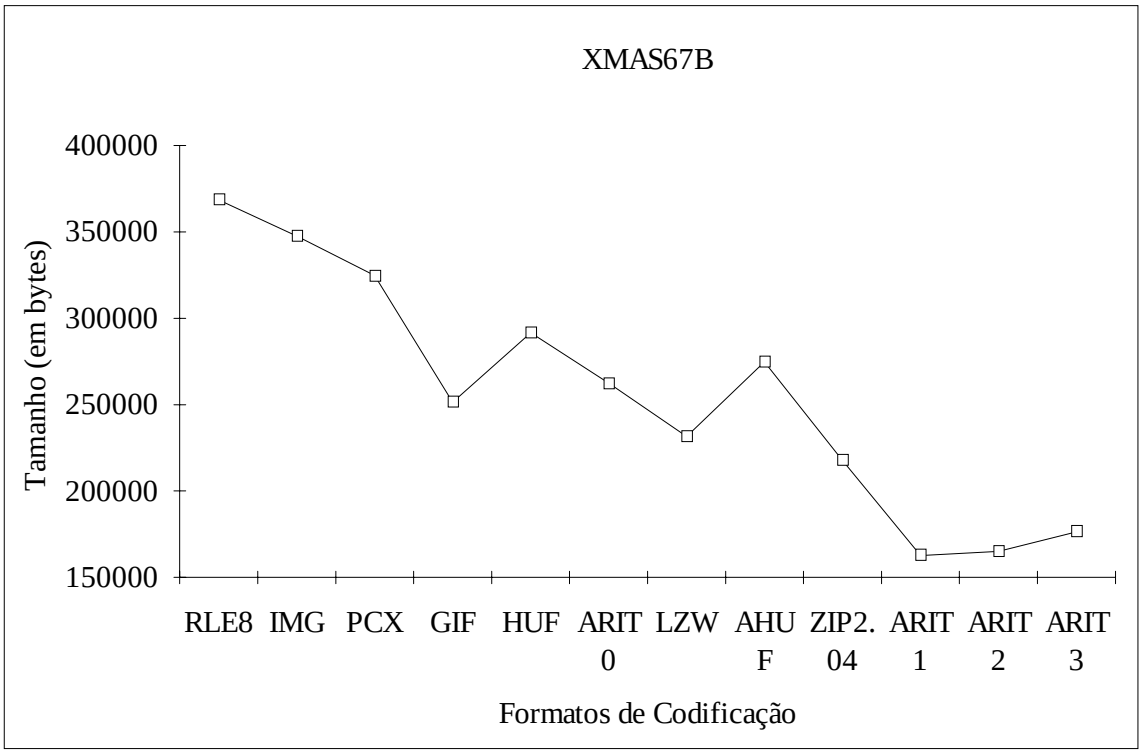


FIGURA 6.13 : Desempenho dos compactadores sobre a imagem XMAS67B.I.

6.3 - CÓDIGO ARITMÉTICO MODIFICADO

Buscou-se, inicialmente aferir qual intervalo de símbolos é mais significativo para a compactação.

Os testes foram realizados, com os três tipos de imagem, utilizando o intervalo de 256 símbolos como parâmetro inicial, pois (NELSON, 1991) utilizou-o como padrão. Em seguida foi aumentado o intervalo para 512, 1024, 2048 e 3072 símbolos, sucessivamente. Foi fixado o limite de 10% para ativação da limpeza da tabela de estatísticas. Os resultados estão na Tabela 6.3.

TABELA 6.3 : Compactação de RJ2.I utilizando intervalos diferentes..

Imagem RJ2.I (184.832 bytes)	
Arq. compactado em bytes	TAM.DO INTERVALO
87352	sem checagem
88396	256
87352	512
87352	1024
87352	2048
87352	3072

TABELA 6.4 : Compactação de PV.I utilizando intervalos diferentes.

Imagem PV.I (208.800 bytes)	
Arq. compactado em bytes	TAM.DO INTERVALO
28593	sem checagem
28593	256
28593	512
28593	1024
28593	2048
28593	3072

TABELA 6.5 : Compactação de LENA.I utilizando intervalos diferentes.

Imagem LENA.I (65.536 bytes)	
Arq. compactado em bytes	TAM.DO INTERVALO
51986	sem checagem
52057	256
51986	512
51986	1024
51986	2048
51986	3072

TABELA 6.6: Compactação de EQ.I utilizando intervalos diferentes.

Imagem EQ.I (108.514 bytes)	
Arq. compactado em bytes	TAM.DO INTERVALO
27466	sem checagem
27466	256
27466	512
27466	1024
27466	2048
27466	3072

As Tabelas 6.3, 6.4, 6.5 e 6.6 mostram que a partir de um intervalo de 512 símbolos já obtem-se uma melhora na taxa de compactação. Um intervalo grande é sempre melhor em termos de velocidade de compactação.

As imagens PV e EQ, tiveram um resultado único devido a alta taxa de compactação que essas imagens possuem, não permitindo que o algoritmo realize alguma tentativa de melhora da taxa de compactação a partir do monitoramento feito dentro dos intervalos testados.

As imagens LENA e RJ2, tiveram o pior resultado no intervalo original, porém esse resultado foi motivado porque justamente nesse intervalo foram feitas tentativas de melhoria da taxa de compactação pelo monitoramento, o que mostra que o controle desse monitoramento não produz bons resultados na versão original da implementação do algoritmo do código aritmético ordem 1.

Os testes seguintes, expostos na Tabela 6.7 e nas Figuras 6.15, 6.16, 6.17 e 6.18, foram realizados para verificação da utilização do monitoramento da variação da taxa de compactação proposto no Capítulo 4, visando melhorar a implementação original.

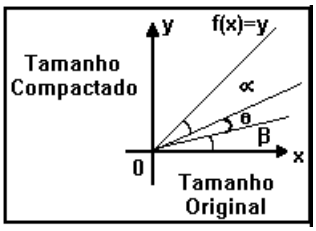
A Figura 6.14 demonstra o que é a variação da taxa de compactação. O gráfico relaciona o tamanho original da imagem com o tamanho compactado. A reta f(x) = y, indica uma taxa de compactação nula e possui um ângulo de 45 graus com o eixo x. Retas com ângulos maiores que 45 graus com o eixo x, representam expansão em lugar de compactação e retas com ângulos menores que 45 graus representam compactação da imagem.

Se, em determinado instante da compactação, o comportamento da taxa de compactação indica uma reta com ângulo Beta com o eixo x, e a seguir, após um intervalo de símbolos lidos, esse comportamento é representado pela reta que possui o ângulo Teta+Beta com o eixo x, pode-se afirmar que a imagem está perdendo taxa de compactação, ou seja, a taxa de compactação está variando de forma prejudicial. Essa perda não indica necessariamente que a imagem não será compactada, mas indica que em determinados instantes há uma perda na taxa de compactação devido as características momentâneas da imagem.

O objetivo do monitoramento da variação da taxa de compactação é justamente evitar uma possível perda de taxa.

TABELA 6.7 : Resultados da modificação de monitoramento.

	LENA	EQ	PV	RJ2
ORIGINAL	65536	108514	208800	184832
<=90	51986	27466	28593	87352
<=90; 80%	52964	27446	28603	87633
<=90; 50%	52964	27449	28609	87633
<=90; 20%	53206	27470	28676	87813
<=90; 10%	53632	27484	28763	88369
<=90; 5%	53741	27542	28828	89032
<=90; 2%	54429	27614	28828	89595



FIIGURA 6.14: Critério do Controle da Compactação.

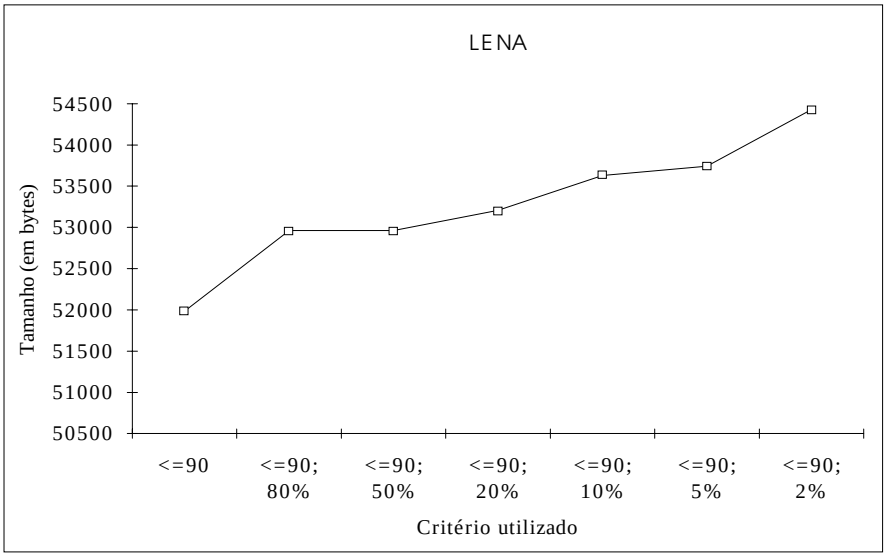


FIGURA 6.15:Resultado de LENA com monitoramento modificado.

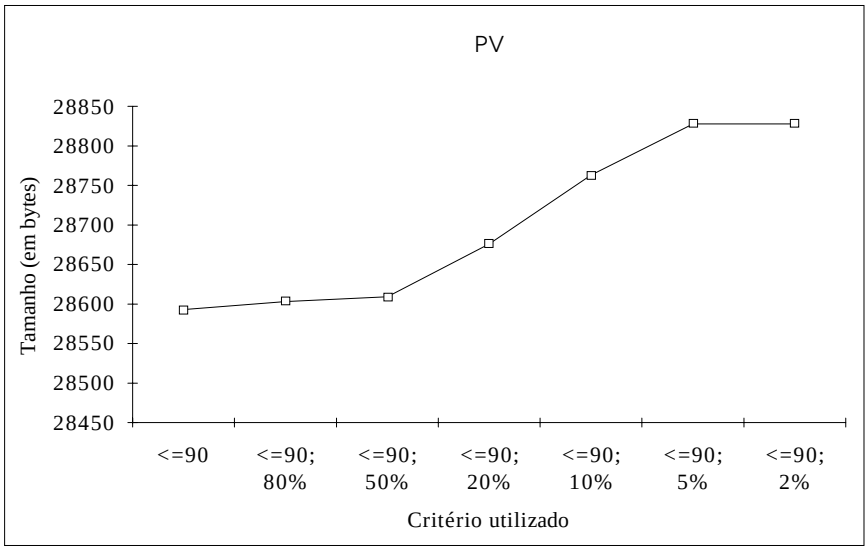


FIGURA 6.16: Resultado de PV com monitoramento modificado.

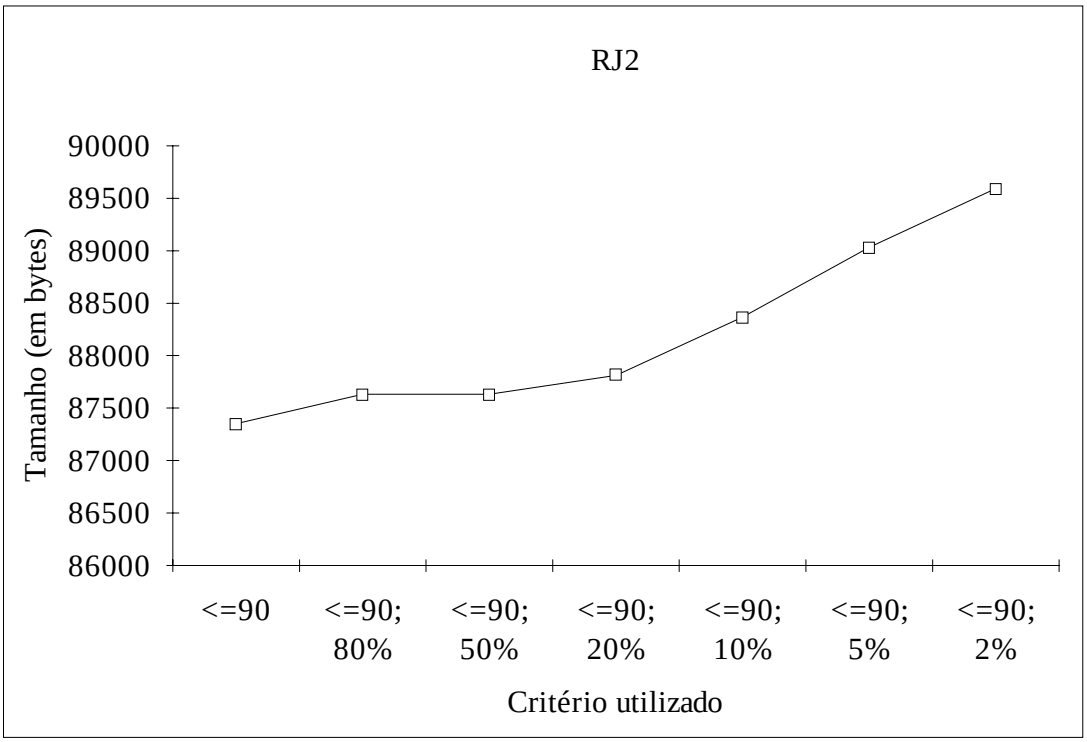


FIGURA 6.17: Resultado de RJ2 com monitoramento modificado.

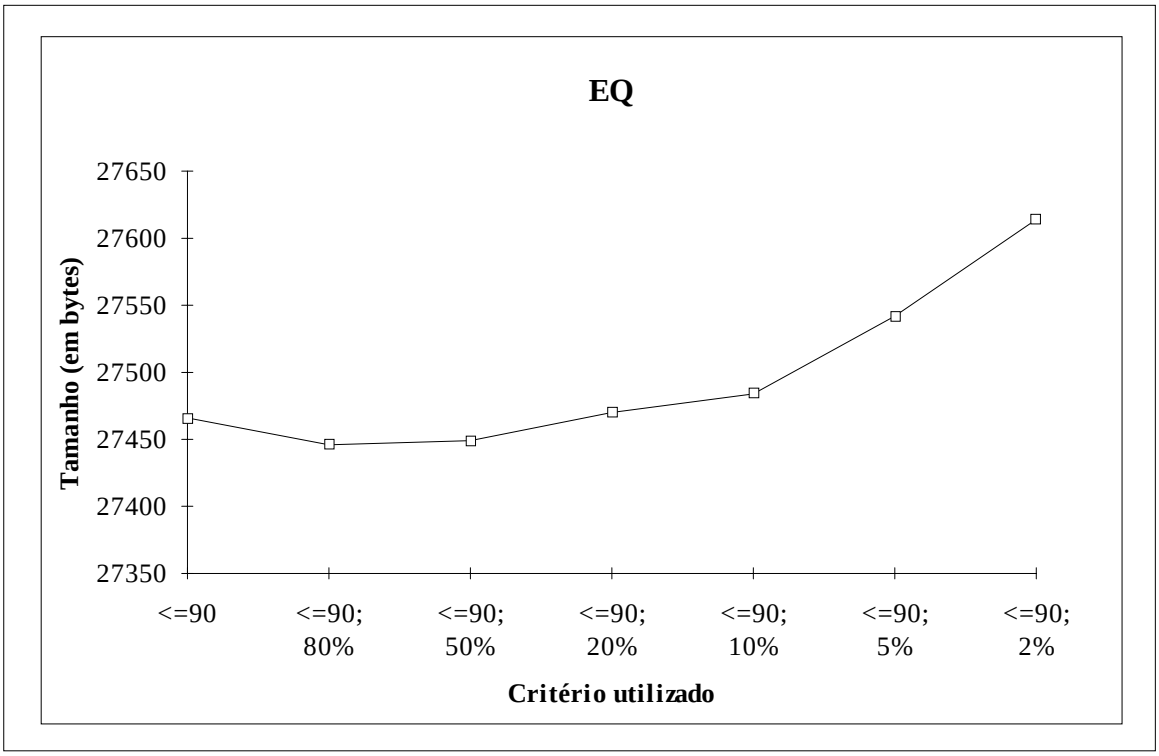


FIGURA 6.18: Resultado de EQ com monitoramento modificado.

Esses resultados mostram que a forma que foi realizado o monitoramento não atendeu as necessidades. Pois, em praticamente todos os casos houve perda na taxa de compactação em relação a compactação sem utilização do monitoramento proposto. O que mostra que é necessário um estudo mais aprofundado em outros trabalhos.

6.4 - ALGORITMO PCX MODIFICADO

Foi utilizada, para os testes, a imagem RJ2.I. A Figura 4.4 mostra o histograma dessa imagem, e a Figura 4.3 mostra a simulação de uma imagem com histograma com distribuição de valores concentrada acima do nível de cinza 192, como explanado no Item 4.3. Imagem essa, obtida a partir do operador binário *not*, aplicado a todos os pixels que compõem a imagem.

Os testes demonstraram que imagens que possuem poucas seqüências de símbolos iguais (que não são indicadas para métodos baseados no RLE por ele compactar justamente seqüências de símbolos iguais) e que possuem uma distribuição com concentração de símbolos acima do valor 192, compactam pouco, podendo até mesmo expandir no formato PCX, e que a mesma imagem, operada pelo *not* binário, não possui o mesmo fraco desempenho.

Para essa imagem foram obtidos os valores de tamanho do arquivo PCX mostrados na Tabela 6.8:

Que demonstra a coerência da modificação proposta. Pois, as imagens com concentração acima de 192, ver histogramas nas Figuras 4.3, 4.4, 6.2, 6.3, 6.19 e 6.20, além de ficarem

maiores que suas correspondentes negativas, ficaram maiores até que as imagens originais, no caso de LENA e RJ2.

TABELA 6.8: Resultados do pré-processamento do formato PCX.

	RJ2	LENA	PV
IMG	184832	65536	208800
PCX	159277	68946	83238
PCX/NEG	256312	80954	86569

Os testes realizados com as imagens LENA.I e PV.I foram efetuados para demonstrar que esse pré-processamento é válido para todos os tipos de imagem, e que somente em imagens com distribuição uniforme no histograma, é que não faria diferença este pré-processamento. Porém, é muito difícil existir uma imagem com essa característica.

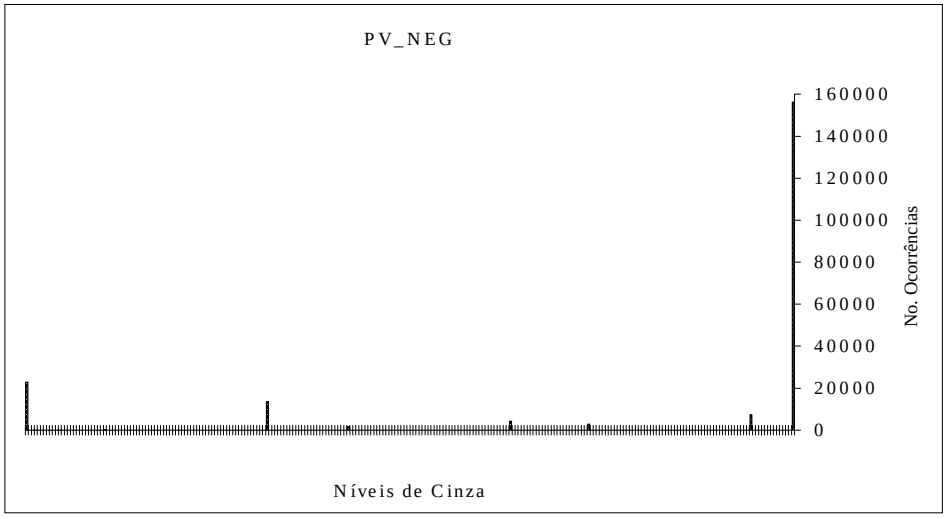


FIGURA 6.19 : Histograma da imagem PV_NEG.

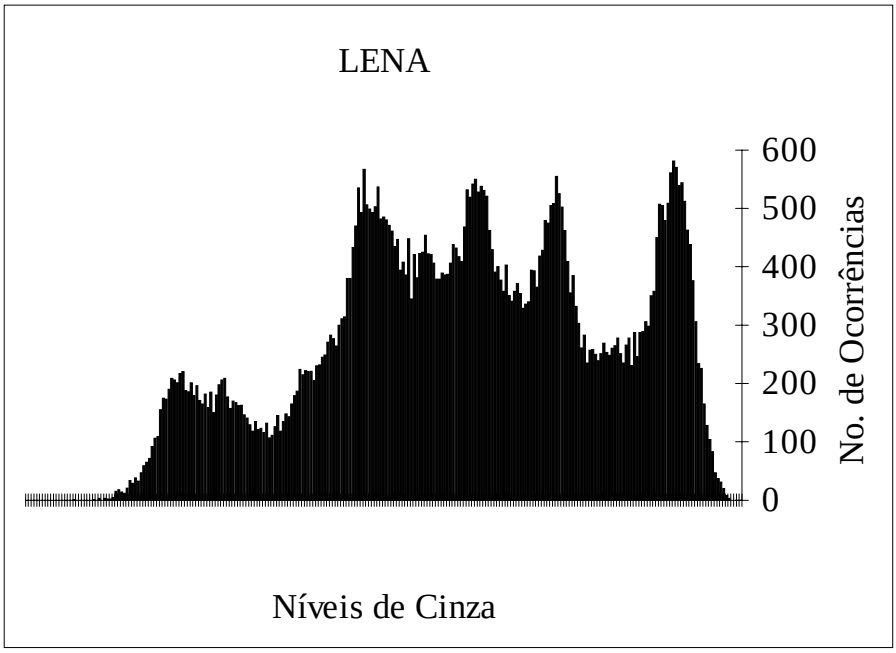


FIGURA 6.20 : Histograma da imagem LENA_NEG.

CAPÍTULO 7

CONCLUSÕES

Verificou-se que, imagens de satélite e fotográficas têm desempenho (em taxa de compactação) semelhante em relação aos algoritmos de compactação. E que o código aritmético ordem 1 se mostrou como o de melhor desempenho. O Formato PCX, mostrou que realmente tem um desempenho fraco, porém, é um bom formato para implementação e intercâmbio de imagens.

As imagens de mapas mostraram as maiores taxas de compactação por possuirem poucas cores, com desempenho ruim apenas nas variações do RLE. Porém, durante a realização deste trabalho, verificou-se a dificuldade de separação de cores em mapas digitalizados por varredura. Essa dificuldade aponta para uma linha de pesquisa de grande relevância, visto que muitas vezes não é possível digitalizar mapas por *overlay*, como exposto no Capítulo 2.

O estudo do controle do monitoramento da taxa de compactação é uma linha de pesquisa que pode gerar bons trabalhos, pois os resultados apesar de não terem sido bons, mostram que é necessário um estudo mais aprofundado.

É importante observar também que toda a modelagem estatística deve ser melhor estudada, visando a criação de algoritmos com modelos de Markov adaptáveis as condições de cada imagem. Pois, apesar do fraco desempenho do modelo ordem 2 e 3,

eles podem ser úteis em pequenas regiões da imagem, em que os símbolos estejam estatísticamente dependentes.

No caso do algoritmo LZW, verificou-se que ele não teve um bom desempenho na imagem LENA. Isto é motivado pelo fato da imagem LENA ter uma quantidade muito pequena de símbolos estatísticamente dependentes. E, como o LZW monta tabelas com a seqüência de símbolos obtidos da fonte, formando um "dicionário" com seqüências de tamanho variado, tudo indica que se esse dicionário contiver seqüências de no máximo dois símbolos a taxa de compactação deve modificar. Essa é outra linha de pesquisa relevante.

7.1 - RECOMENDAÇÕES

7.1.1 - MODIFICAÇÕES DE LEIAUTE *(LAYOUT)*

São operações de pré-processamento que visam melhorar o desempenho dos algoritmos de compactação através da modificação do leiaute de armazenamento da imagem, de forma reversível.

Essas modificações são feitas através de programa, que lê uma imagem em seu leiaute original e gera uma segunda com leiaute modificado. Pode ser realizada, também, através da modificação da forma de leitura dos símbolos durante a compactação, para que se leia a imagem simulando o leiaute modificado.

Em uma leitura normal, os símbolos são obtidos da primeira coluna à esquerda, até a última, à direita, linha a linha,

até o final da imagem. Em outras palavras, lê-se a imagem linha a linha. Existem muitas variações para a forma de leitura, como por exemplo a inversão da leitura normal, passando-se a ler coluna a coluna. Entretanto, esta técnica não vai influenciar a taxa de compactação de algoritmos estatísticos baseados no modelo de Markov ordem 0, por visualizarem a probabilidade de ocorrência do símbolo no arquivo de imagem, que não vai ser alterada.

A técnica apresentada é aplicável em imagens com mais de uma banda, ou componente de cor, o que na prática faz com que seja mais de uma imagem, porém com a mesma área representada. A leitura é feita obtendo-se o primeiro símbolo da primeira imagem, em seguida o primeiro da próxima e assim sucessivamente até a última, quando então retorna-se à primeira imagem e obtém-se o segundo símbolo, repetindo-se o procedimento anterior até o final da imagem. A obtenção do símbolo de cada imagem segue a forma normal, linha a linha (Figura 7.1).

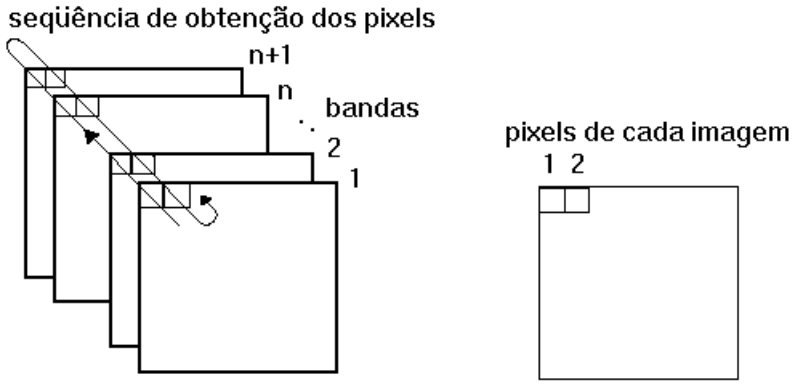


FIGURA 7.1 : Seqüência de Obtenção dos Símbolos das Imagens.

7.1.2 - ORDENAÇÃO DA TABELA DE CORES DO FORMATO PCX

O Formato PCX possui uma tabela de cores cujos índices são os valores dos *pixels* da imagem. Logo os valores dos pixels da imagem podem ser modificados, bastando que se modifique a tabela de cores para manter a mesma cor para o determinado *pixel*, modificando apenas sua posição na tabela, e não na imagem. De acordo com os testes realizados no Capítulo 6, para melhoria do algoritmo PCX, verificou-se que imagens com histograma sem concentração acima do valor 192 possui uma melhor taxa de compactação.

Porém, o critério utilizado nos testes, consiste em uma modificação na imagem para habilitá-la para a compactação, necessitando-se realizar a operação inversa após a descompactação.

Uma melhoria nesta metodologia consiste em ordenar a tabela de cores, colocando os pixels com maior ocorrência na imagem, com índices próximos a zero e pixels com menor ocorrência com os índices maiores. Obviamente modificando o valor dos índices dos pixels na imagem, para que ela permaneça idêntica visualmente.

Essa melhoria proporcionará uma taxa de compactação melhor que a metodologia apresentada no Capítulo 6 e deixará a imagem intacta, sem a necessidade de uma operação inversa na descompactação.

7.2 - SUGESTÕES PARA TRABALHOS FUTUROS

(a) Estudar a separação de cores na aquisição de imagens de mapas coloridos, visando a separação de níveis de informação;

(b) Estudar o monitoramento da taxa de compactação;

(c) Estudar a modelagem estatística, com modelos de Markov adaptáveis as condições de cada imagem;

(d) Estudar o algoritmo LZW com o tamanho das seqüências de símbolos adaptado ao uso de imagens;

(e) Estudar modificações de leiaute, principalmente em imagens multiespectrais;

(f) Estudar a ordenação da tabela de cores do formato PCX.

APÊNDICE A

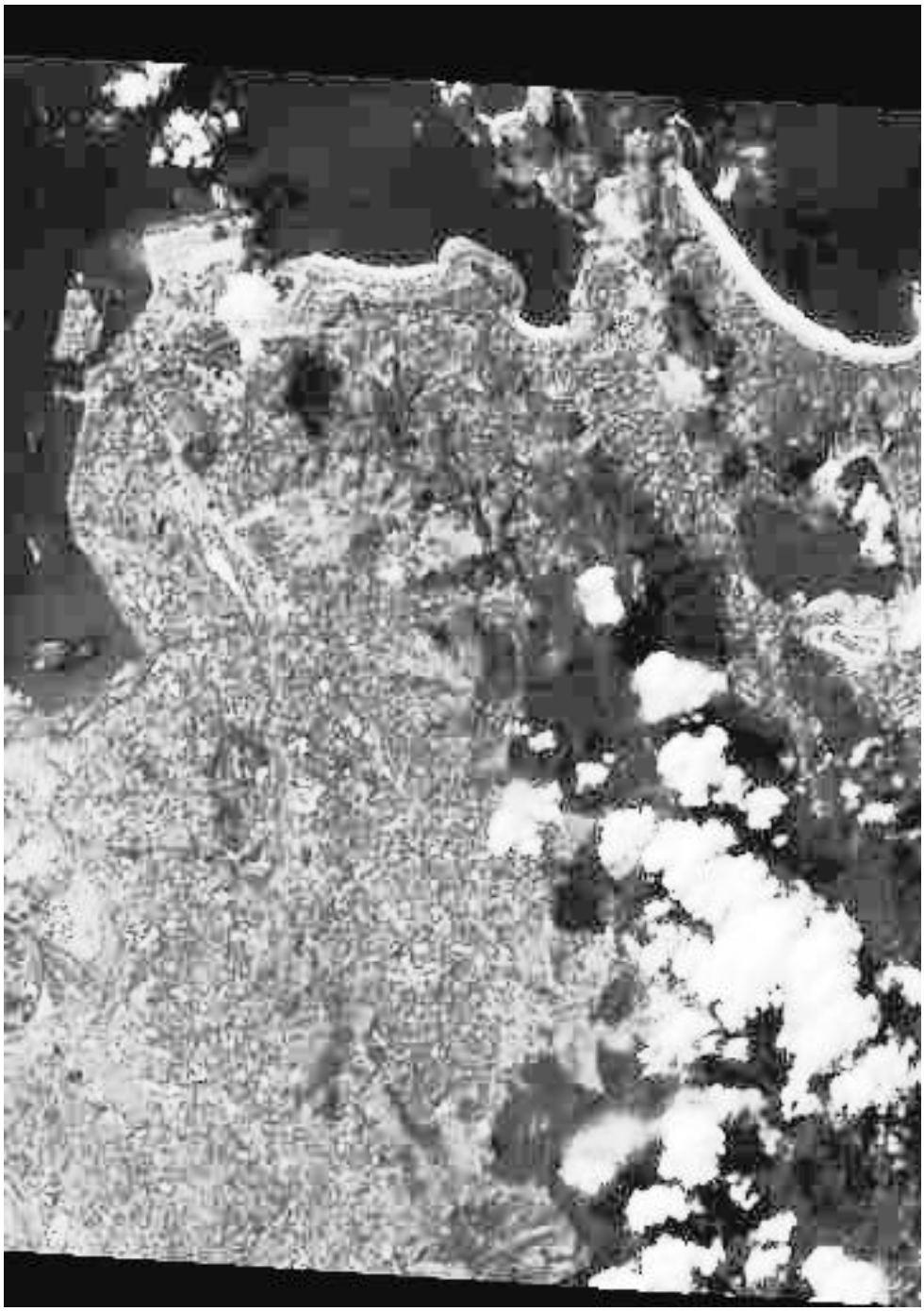


FIGURA A.1: RJ2.I.

FIGURA A.2: LENA.I.

FIGURA A.3: XMAS67.I

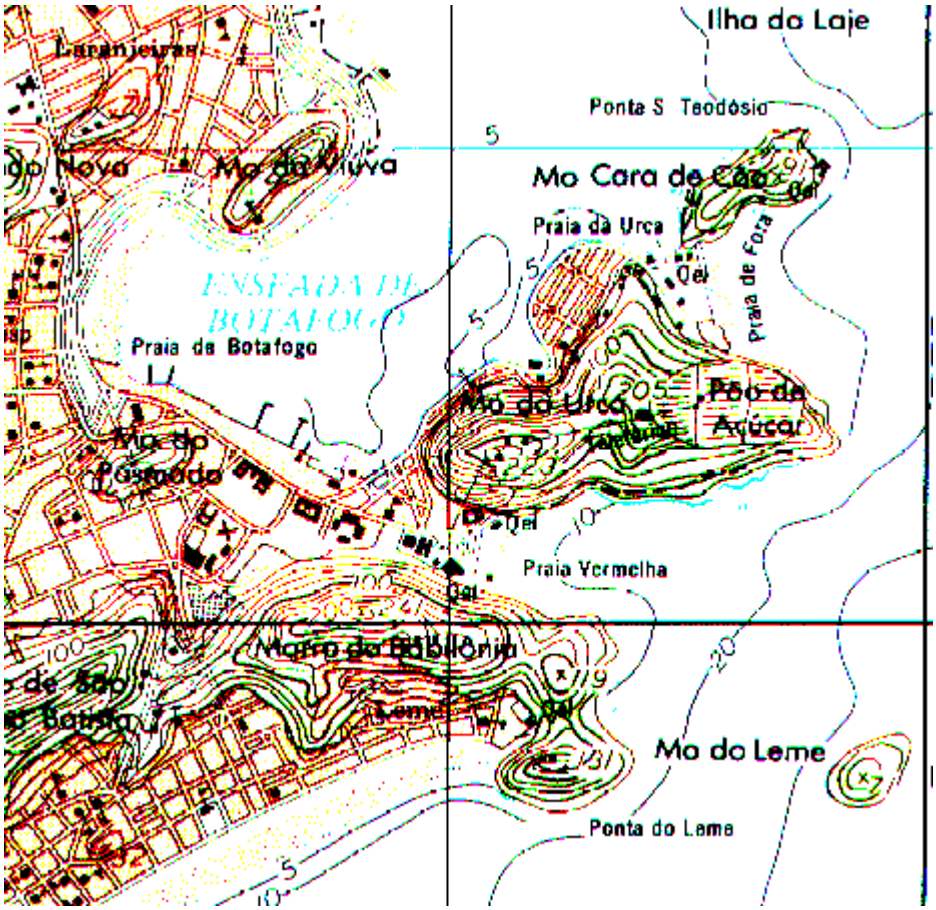


FIGURA A.4: PV.I.

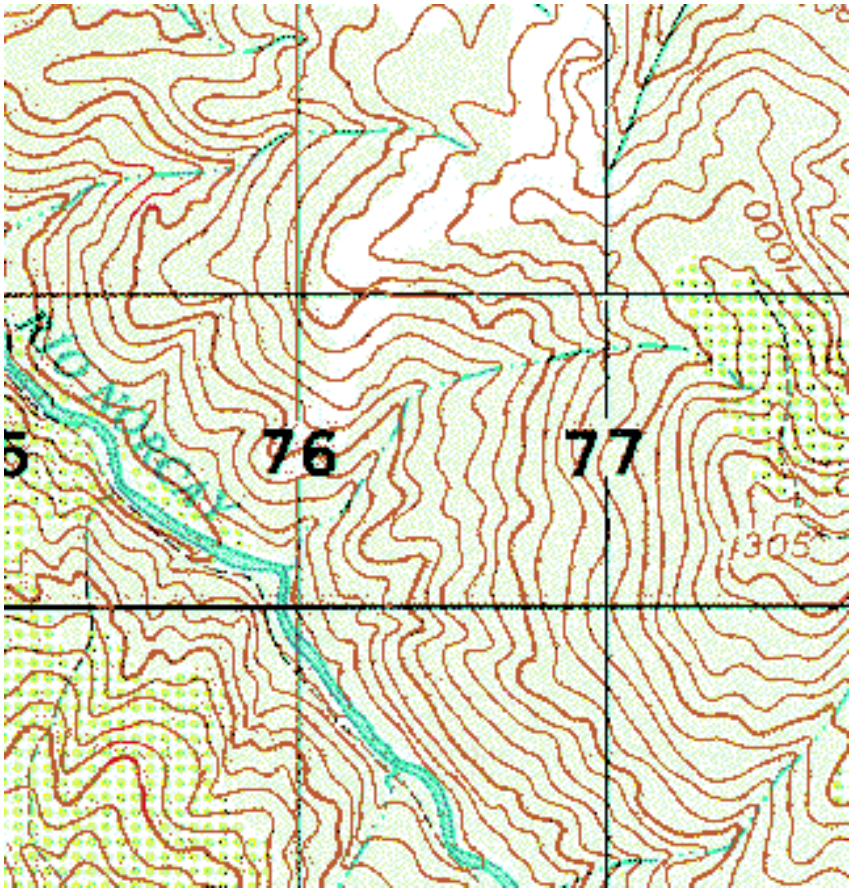


FIGURA A.5: EQ.I

REFERÊNCIAS BIBLIOGRÁFICAS

ABRAMSOM, Norman. **Information Theory and Coding**. McGraw-Hill Eletronic Science Series, 1963.

ALMEIDA, Marcos de. **Alternativas para Armazenamento de Imagens Discretas**, tese de mestrado IME, Janeiro de 1989.

APIKI, S. **Lossless Data Compression**. Byte, Março 1991.

BELL, T.C. IEEE Transactions COM-34, 1176-1182 (1986).

BELL, Timothy C.; CLEARY, J.G.; WITTEN, I.H. **Text Compression**. Englewood Cliffs, N.J.: Prentice Hall, 1990.

BLAHUT, Richard. **Principles Practices For Information Theory**. Addison-Wesley Publishing Company, 1990.

BUTRON, Cesar A. Gonzales. **Advances in Image Compression Techiniques**. IBM Research Division. T.J.Watson Research Center. Yorktown Heights, NY 10598.

CARTENSEN, L.W. & CAMPBELL, J.B.. **Desktop Scanning for Cartographic Digitazion and Spatial Analysis**. Photogrametric Enginneering and Remote Sensing. 57(11): 1437-1446, 1991.

CASE, Shaun. **Run Length Encoding compressor program**, atman%ecst.csuchico.edu@RELAY.CS.NET, (internet), 1991.

CHUI, Paul. **A C++ PCX File Viewer for Windows 3**. Dr.Dobb's Journal, Julho 1991.

COMPUSERVE INCORPORATED. **GIF - Graphics Interchange Format**. 15 de Junho de 1987.

COMPUSERVE INCORPORATED. **GIF - Graphics Interchange Format**. 1989.

ELIAS, P. I**nformation Theory and Coding**. N.Abramson, McGraw-Hill Book Co., Inc, New York, 1963.

FALLER, Newton & SALENBAUCH, Pedro. **Compressão de Dados no Unix**. Boletim do Plurix nº 15, NCE-UFRJ, 1991.

FALLER, Newton. **An Adaptative System for Data Compression**. Records of Seventh Asilomar Conference on Circuits, Systems and Computers, Pacific Grove, CA, E.U.A., 1973, 593-597.

FALLER, Newton. **Sistema adaptativo para compressão de dados**, Tese de Mestrado, COPPE-UFRJ, 1973.

FRANCISCO, C.N. & XAVIER-DA-SILVA, J.. **O Uso de Scanners na Digitalização de Mapas Destinados a Sistemas de Informações Geográficas**. Anais do XVI Congresso Brasileiro de Cartografia. 807-815, Outubro 1993.

GALLAGER, R.G. **Variations on a Theme by Huffman**. IEEE Transactions on Information Theory IT-24,6 (Nov.1978), 668-674.

GONZALEZ, Rafael C.. **Digital Image Processing**. 2ª edição, Addison-Wesley Publishing Company, 1977.

GRAEF, Gerald L. **GRAPHICS FORMATS**, A close look at GIF, TIFF and other attempts at a universal image format. Byte, Setembro 1989.

HELD, G. **Compressão de Dados**. Editora Érica, 1992.

HUFFMAN, David A.. **A Method for the Construction of Minimum-Redundancy Codes**. 6 de dezembro de 1951. I.R.E..

JAIN, Anil K. **Fundamentals of Digital Image Processing**. Prentice Hall Information and System Sciences Series.

KATZ, Phil. **PKZIP.EXE 2.04 & PKARC.EXE 3.6**. PKWare(Glendale, Wisc., USA).

KAY, David C. & LEVINE, John R.. **Graphics File Formats**. Windcrest/McGraw-Hill, 1993.

KERNIGHAN, Brian W. & RITCHIE, Dennis M., **The C Programming Language**. Englewood Cliffs, N.J.: Prentice-Hall, 1978.

KNUTH, D.E. **Dynamic Huffman Coding**. Journal of Algorithms 6 (1985), 163-180.

LEMPEL, A. & ZIV, J.. **A Compression of Individual Sequences via Variable-Rate Coding**. IEEE Transactions on Information Theory IT-24, 5 , Setembro 1978.

LEMPEL, A. & ZIV, J.. **A Universal Algorithm for Sequencial Data Compression**. IEEE Transactions on Information Theory IT-23, 3 , Maio 1977.

LOVELL, D.J.. **Color**. The Software Toolworks Multimedia Encyclopedia/Grolier. Edição 1992.

MEADOW, Anthony; OFFNER, Rocky; BUDIANSKY, Michael. **Handling Image Files with TIFF**. Dr.Dobb's Journal, Maio 1988.

MICROSOFT. **MS-DOS 5.0 User's Guide**. Microsoft, 1992.

NELSON, Mark R.. **Arithmetic Coding and Statistical Modeling**. Dr.Dobb's Journal, Fevereiro 1991.

NELSON, Mark R.. **LZW Data Compression**. Dr.Dobb's Journal, Outubro 1989.

OKUMURA, Haruhiko. **Data Compression Algorithms of LARC and LHarc**, COMPRESS.TXT, Science SIG (forum) of PC-VAN Japan.

OLIVEIRA, Cêurio de. **Curso de Cartografia Moderna**. FIBGE, Rio de Janeiro, 1988.

OLIVEIRA, Cêurio de. **Dicionário Cartográfico**. FIBGE, 2a edição, Rio de Janeiro, 1983.

PRATT, William K. **Digital Image Processing**. Wiley-Interscience, 1978.

QUINTANILHA, José Alberto. **Processamento de Imagens Digitais**. Simpósio Brasileiro de Geoprocessamento. Anais EPUSP, São Paulo - SP, P-37-52, Maio 1991.

QUIRK, Kent. **Translating PCX Files**. Dr.Dobb's Journal, Agosto 1989.

REGAN, Shawn M.. **LZW Revisited**. Dr.Dobb's Journal, Junho 1990.

SEDGEWICK, R. **Algorithms**, 2ª edição (Addison-Wesley, 1988).

STORER, J.A. & SZYMANSKY, T.G. **LZSS**. J. ACM, 29, 928-951 (1982).

STORER, J.A. **Data Compression Methods and Theory**. Computer Science Press (1988).

THOMAS, Kas. **ENTROPY The Key to Data Compression**. Dr.Dobb's Journal, Fevereiro 1991.

VITTER, J.S. **Design and Analysis of Dynamic Huffman Codes**. Journal of the Association for Computing Machinary, Outubro 1987.

WELCH, Terry. **A Technique for High-Performance Data Compression**. Computer, IEEE, Vol.17, Nº 6, Junho 1984.

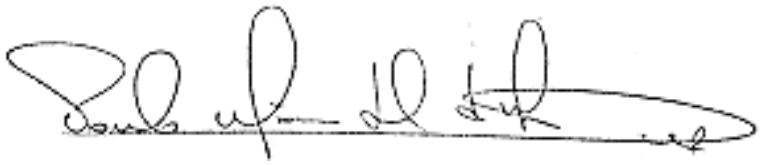
WITTEN, Ian H. et alli. **Arithmetic Coding for Data Compression**. Communications of the ACM, Junho 1987.

Tese apresentada por

_ _

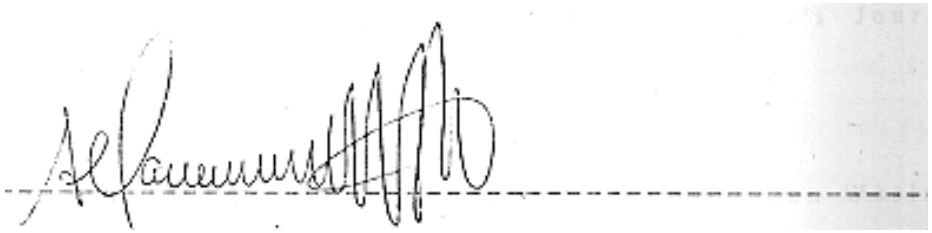
MARCO ANTONIO LEMOS PERNA

e aprovada pelos Srs.:



_ _

Cel PAULO MÁRCIO LEAL DE MENEZES - M.C. IME



ANTONIO JOSÉ FERREIRA MACHADO E SILVA - M.C. INPE

IME, RIO DE JANEIRO- RJ, 10 de fevereiro de 1994.