1. Publicação nº	2. Versão	3. Data	5. Distribuição
INPE-2274-PRE/054		*Nov., 1981*	☐ Interna ☒ Externa ☐ Restrita
4. Origem *DSE*	Programa *DIN*		

6. Palavras chaves - selecionadas pelo(s) autor(es)
COMPRESSÃO DE IMAGENS
BANCO DE DADOS
TEORIA DA INFORMAÇÃO

7. C.D.U.: *681.3.016*

8. Título *INPE-2274-PRE/054*

MÉTODOS DE COMPRESSÃO DE IMAGENS SEM PERDA DE INFORMAÇÃO

10. Páginas: *21*

11. Última página: *20*

12. Revisada por

Flávio R.D. Velasco

9. Autoria *Orion de Oliveira Silva*

13. Autorizada por

Nelson de Jesus Parada
Diretor

Assinatura responsável Orion de Oliveira Silva

14. Resumo/Notas

Com a evolução e popularização da tecnologia de computadores, grandes programas, grandes blocos de texto e bancos de dados dos mais diversos tipos passaram a ser armazenados e acessados por computador. Embora o custo da armazenagem venha caindo continuamente, graças a avanços tecnológicos que barateiam os componentes e aumentam a densidade da informação por unidade de área, continua haver necessidade de técnicas de compactação que economizem espaço de memória (primária, secundária e terciária). Neste trabalho, apresentam-se vários métodos de compactação de imagens digitalizadas sem perda de informação, isto é, a imagem descompactada é idêntica à imagem original. Fornecem-se detalhes sobre a implementação dos algoritmos e apresentam-se os resultados comparativos, obtidos de sua aplicação a imagens de satélites meteorológicos e de sensoriamento remoto.

15. Observações *Submetido para publicação na Revista da Sociedade Brasileira de Matemática Aplicada e Computacional; apresentado no 4º Congresso Nacional de Matemática Aplicada e Computacional, 8-11 Set. 1981, Rio de Janeiro, RJ.*

MÉTODOS DE COMPRESSÃO DE IMAGENS SEM PERDA DE INFORMAÇÃO

Orion de Oliveira Silva

Instituto de Pesquisas Espaciais
Conselho Nacional de Desenvolvimento Científico e Tecnológico
Caixa Postal 515 - 12200 - São José dos Campos, SP

RESUMO

Com a evolução e popularização da tecnologia de computadores, grandes programas, grandes blocos de texto e bancos de dados dos mais diversos tipos passaram a ser armazenados e acessados por computador. Embora o custo da armazenagem venha caindo continuamente, graças a avanços tecnológicos que barateiam os componentes e aumentam a densidade da informação por unidade de área, continua haver necessidade de técnicas de compactação que economizem espaço de memória (primária, secundária e terciária). Neste trabalho, apresentam-se vários métodos de compactação de imagens digitalizadas sem perda de informação, isto é, a imagem descompactada é idêntica à imagem original. Fornecem-se detalhes sobre a implementação dos algoritmos e apresentam-se os resultados comparativos, obtidos de sua aplicação a imagens de satélites meteorológicos e de sensoriamento remoto.

ABSTRACT

With the evolution and popularization of computer technology, large programs, large blocks of text, and a wide variety of data bases have been stored and retrieved by computers. While the storage cost has been dropping continuously, thanks to technological breakthroughs which produce cheap compo

nents and increase the information density per unit area, the need for compression techniques for saving memory space (primary, secondary and terciary) still exists. Several digital image compression methods, without information loss are presented, i. e., in them the decompressed image is identical to the original one. Details about the algorithm implementation will be given, as well as comparative results derived from their application to remote sensing and meteorological satellite imagery.

1. INTRODUÇÃO

Dada a larga tendência de armazenamento de banco de imagens, estão sendo desenvolvidos neste trabalho alguns métodos de compressão de imagens sem perda de informação.

Um aspecto da compressão de imagens é a sua grande utilidade para transmissão de dados, pois reduz o tempo da transmissão.

O trabalho tem por objetivo desenvolver a implementação e a comparação de vários métodos de compressão de imagens.

2. MÉTODOS DE COMPRESSÃO DE UMA MANEIRA GERAL

Os métodos de compressão podem ser divididos em duas categorias.

2.1 - Primeira Categoria: Métodos Dependentes do Conteúdo

Nesta categoria, a compressão é direcionada para uma dada aplicação, dependendo do conteúdo dos dados, tais como: supressão de caracteres, supressão de itens repetidos em banco de dados, conversão da notação humana para uma mais compactada, substituição de itens de dados usados comumente, evitação de espaços vazios nos arquivos, substituição de textos em uma lingua

gem natural, e compressão de dados ordenados. Todos estes itens são tratados em Silva (1980).

2.2 - Segunda Categoria: Métodos Independentes do Conteúdo

Os métodos desta categoria podem ser usados para muitas aplicações e implementados em nível de "software", "hardware" ou em microprogramação.

A eficiência das técnicas definidas na primeira categoria é altamente dependente da natureza dos dados. Algumas das técnicas são inteiramente dependentes da aplicação, tais como as técnicas que serão apresentadas mais adiante, para compressão de imagens.

Um caminho, geralmente aplicável à compressão de dados armazenados ou à transmissão de dados, é o uso da forma mais eficiente de codificação de caracteres.

2.2.1 - Código de Huffman

Um dos métodos de compressão mais utilizado, independente do conteúdo, foi proposto por Huffman (1955), e é conhecido como Código de Huffman.

O código de Huffman baseia-se no fato de que os diferentes caracteres ocorrem com frequências acentuadamente diferentes em um arquivo. Então, Huffman sugeriu que fossem atribuídos códigos:

i) de menor comprimento, aos caracteres mais repetidos (de maior frequência);

ii) de maior comprimento, aos caracteres menos repetidos (de menor frequência).

Com estas duas sugestões, obtém-se um comprimento médio menor do que o necessário para representar os caracteres por um código de tamanho fixo.

Exemplo 1. Suponha-se que se queira codificar um conjunto de oito caracteres: A, B, C, D, E, F, G e H. Com um código fixo tem-se um código possível:

A - 000	E - 100
B - 001	F - 101
C - 010	G - 110
D - 011	H - 111

Seja, agora, a distribuição de probabilidades das ocorrências destes caracteres no texto que deve ser armazenado:

P(A) = 0,50	P(E) = 0,03
P(B) = 0,25	P(F) = 0,02
P(C) = 0,12	P(G) = 0,01
P(D) = 0,06	P(H) = 0,01

Pode-se codificar os caracteres A, B, C, D, E, F, G, H da maneira apresentada na Tabela 1.

O comprimento, para o código de tamanho fixo, é de 3 (três) bits por caractere, enquanto que o comprimento médio ponderado é de 2,01 bits por caractere.

O código compactado para uma fonte S é o código que tem o tamanho médio menor possível, quando se codifica os símbolos da fonte S. Sendo N o número de mensagens de S e P(i) a probabilidade da i-ésima mensagem, então, $\sum_{i=1}^{N} P(i) = 1$. O tamanho de uma mensagem, L(i), é o número de dígitos que codifica i. O tamanho médio é

$$L_{av} = \sum_{i=1}^{N} P(i)\ L(i)$$

Portanto, o código ótimo, que é o código de redundância mínima, será o que apresente:

$$\min L_{av} = \sum_{i=1}^{N} P(i)\ L(i)$$

O termo "redundante" foi definido por Shannon (1964) como uma propriedade dos códigos.

TABELA 1

CÓDIGO DE COMPRIMENTO VARIÁVEL

CARACTERE	CÓDIGO	COMPRIMENTO	PROBABILIDADE	PROBABILIDADE x COMPRIMENTO
A	0	1	0,50	0,50
B	10	2	0,25	0,50
C	110	3	0,12	0,36
D	1110	4	0,06	0,24
E	11110	5	0,03	0,15
F	111110	6	0,02	0,12
G	1111110	7	0,01	0,07
H	1111111	7	0,01	0,07
				2,01

Um código de redundância mínima será definido, aqui, como um grupo de palavras tal que, para um grupo de mensagens consistindo num número finito de elementos N, e para um dado número de dígitos, produz o menor tamanho médio de mensagem. Será entendido como "código ótimo" o "código de redundância mínima".

As seguintes restrições foram impostas por Huffman (1955) para o grupo de palavras do código:

a) Nenhuma das mensagens será constituída de arranjos idênticos de dígitos do código.

b) Os códigos-mensagens serão construídos de tal maneira que nenhuma identificação adicional seja necessária para especificar onde o código-mensagem começa e onde termina.

Exemplo 2. Sejam 01, 102, 111 e 202 códigos-mensagens válidos para um grupo de quatro dígitos. A sequência: 1111022020101111102 pode ser particionada nas mensagens individuais: 111-102-202-01-01-111-102, tendo o receptor apenas o grupo de códigos-mensagens.

c) $L(1) \leq L(2) \leq \ldots \leq L(N-1) = L(N)$.

d) Pelo menos duas e não mais que duas mensagens, com tamanho L(N), têm códigos que diferem apenas pelos seus dígitos finais.

e) Cada sequência possível de L(N)-1 dígitos deve ser usada como um código-mensagem, ou deve usar um dos seus prefixos.

A obtenção do código de Huffman é feita considerando-se, inicialmente, o conjunto de probabilidades associadas aos n caracteres a codificar. As duas probabilidades de menor valor são substituídas, no conjunto, por uma soma, fazendo-se isto n-2 vezes. Durante o processo seria atribuído o valor 0 (zero) para a maior probabilidade e o valor 1 (um) para a menor, repetir-se-ia o processo e concaternar-se-ia os valores anteriores.

Um exemplo é apresentado na Tabela 2.

TABELA 2

PROCEDIMENTO DO CÓDIGO DE HUFFMAN

SÍMBOLO	PROBABILIDADE	CÓDIGO
S_1	0,4	1 — 0,4 1 — 0,4 1 — 0,4 1 — 0,6 0
S_2	0,3	00 — 0,3 00 — 0,3 00 — 0,3 00 — 0,4 1
S_3	0,1	011 — 0,1 011 — 0,2 010 — 0,3 00
S_4	0,1	0100 — 0,1 0100 — 0,1 011
S_5	0,06	01010 — 0,1 0101
S_6	0,04	01011

Pode-se complementar o j-ésimo dígito de todas as palavras do código, obtendo-se outro código comprimido, como no Exemplo 3.

Exemplo 3. Código Comprimido 1 Código Comprimido 2

1		0
00		10
011	=>	111
010		1100
01010		11011
01011		11010

Pode-se, ainda, construir outro código, como mostra a Tabela 3.

TABELA 3

PROCEDIMENTO DE HUFFMAN

SÍMBOLO	PROBABILIDADE	CÓDIGO
S_1	0,4	1 — 0,4 — 0,4 1 — 0,4 — 0,6 0
S_2	0,3	00 — 0,3 — 0,3 00 — 0,3 00 — 0,4 1
S_3	0,1	0100 — 0,1 — 0,2 010 — 0,3 01
S_4	0,1	0101 — 0,1 0100 — 0,1 011
S_5	0,06	0110 — 0,1 0101
S_6	0,04	0111

A demonstração de que o código de Huffman é o código ótimo não é imediata, e pode ser encontrada em Abramson

(1963) e Huffman (1955). Para os códigos do Exemplo 3 tem-se:

$L_1 = 1(0,4)+2(0,3)+3(0,1)+4(0,1)+5(0,06)+5(0,04) = 2,2$ bits/ símbolo

$L_2 = 1(0,4)+2(0,3)+4(0,1)+4(0,1)+4(0,06)+4(0,04) = 2,2$ bits/ símbolo

Então, $L_1 = L_2$, de onde se pode concluir que existe mais do que um código ótimo.

2.2.2 - Método de Bruce-Hans

Este método de compressão de dados foi publicado por Hans (1974), tendo sido testado e aperfeiçoado pelo autor deste trabalho.

O método consiste no seguinte:

i) Frequentemente, a gravação de textos, contendo muitas listas de símbolos repetidos, ocupa muito espaço na memória. O resultado é que se desperdiça muito espaço por métodos comuns de armazenamento. Uma das maneiras de solucionar este problema é armazenar somente os símbolos não-repetidos. A técnica aqui utilizada, além de armazenar os símbolos não-repetidos, reconhece listas de símbolos repetidos e as trata de modo especial.

Exemplo 4. | x | y |, onde "x" corresponde ao código do símbolo (que pode ser o símbolo em EBCDIC negativo), e "y", o número de vezes que o símbolo é repetido.

ii) Os símbolos não-repetidos são codificados em grupos de tamanho fixo, como o número inteiro correspondente à maior palavra da máquina.

A técnica de compressão utilizada consiste em:

a) ler a lista de símbolos de entrada (cartão, fita, disco);

b) construir o dicionário de símbolos (DS);

c) codificar os símbolos usando o DS;

d) comprimir os dados de entrada.

A construção do dicionário de símbolos é referente a cada computador.

Seja B o comprimento do dicionário primário e P_i a posição do i-ésimo símbolo de um dado grupo, onde $(1 \leq i \leq N)$ e $(1 \leq P_i \leq B-1)$ no DS. A B-ésima posição é reservada para um caractere de fuga, que permite a extensão do dicionário. O grupo de posições $P_1, P_2, \ldots, P_N$ é codificado como um número de ponto fixo: $P_1 * B^{N-1} + P_2 * B^{N-2} + \ldots + P_{N-1} * B^1 + P_N * B^0$.

Se a B-ésima posição não for reservada, o valor $B*B^i$ pode aparecer, causando ambiguidade na codificação.

O caractere de fuga significa o símbolo seguinte à posição na faixa B+1 até 2B-1. Mais do que um caractere de fuga pode ser usado para a extensão do comprimento do dicionário.

Seja p a posição de um dado símbolo no DS; então, o símbolo é codificado com $\lfloor p/B \rfloor$ caracteres de fuga, seguidos de mod(p,B), onde $\lfloor y \rfloor$ significa a parte inteira de y. O caractere de fuga é codificado com o zero.

Exemplo 5. Seja B = 8 o comprimento do dicionário primário e N=4 (os símbolos são codificados em grupos de 4).

Considere-se os símbolos a serem codificados como tendo as posições 5,7,10,15,25 e 9 no DS. Estes símbolos são codificados em 3 números de ponto fixo:

- o primeiro tem o valor $5*8^3+7*8^2+0*8+2*8^0 = 3010$;

- o segundo tem o valor $0*8^3+7*8^2+0*8+0*8^0 = 448$;

- o terceiro tem o valor $0*8^3+1*8^2+0*8+1*8^0 = 65$.

Os símbolos 10,15,25 e 9 são codificados como (0,2), (0,7), (0,0,0,1) e (0,1), respectivamente.

Neste método, o usuário do sistema tem controle sobre os valores de B e N. Estes valores são selecionados para minimizar a quantidade média do número de bits/informação, necessários à codificação do texto de entrada.

Para cada valor de N, existe um valor ótimo para B. Este valor ótimo é dado pelo maior inteiro positivo, tal que $B^N-1 \leq L$, onde L é o maior inteiro que pode ser armazenado numa palavra do computador, isto porque o maior inteiro a ser produzido é: $(B-1) * B^{N-1} + \ldots + (B-1) = B^{n-1}$.

O valor de B é obtido pela equação $B = \lfloor (L-1)^{1/N} \rfloor$.

A Tabela 4 mostra os valores de B e N para computadores com palavra de 16 bits, 32 bits e 48 bits.

TABELA 4

VALORES ÓTIMOS DE B E N

L = -16 BITS		L = -32 BITS		L = -48 BITS	
VALOR DE N	VALOR DE B	VALOR DE N	VALOR DE B	VALOR DE N	VALOR DE B
5	8	5	74	5	776
6	6	6	36	6	256
7	4	7	22	7	116
8	4	8	15	8	64
9	3	9	11	9	40
10	3	10	9	10	28
11	3	11	7	11	21
12	2	12	6	12	16
13	2	13	5	13	13
14	2	14	5	14	11
15	2	15	4	15	9
16	2	16	4	16	8
17	2	17	4	17	7
18	2	18	3	18	6
19	2	19	3	19	6
20	2	20	3	20	5

Pode-se otimizar B e N com relação a um determinado texto, usando-se apenas estatísticas do texto, e tomando-se o cuidado de contar os caracteres repetidos como sendo uma unidade.

Sejam K_1, K_2, ..., K_n os números de caracteres codificados com 1,2,...,n dígitos, respectivamente. Então, o problema é encontrar um B e um N para um determinado texto, tal que minimizem T, onde $T = K_1 + 2*K_2 + \ldots + n K_n$.

Não foi encontrada, ainda, uma maneira eficiente de achar o valor de T mínimo. O que se fez, aqui, foi postular um valor de B no meio da tabela e aumentar ou diminuir este valor, até encontrar o valor ótimo.

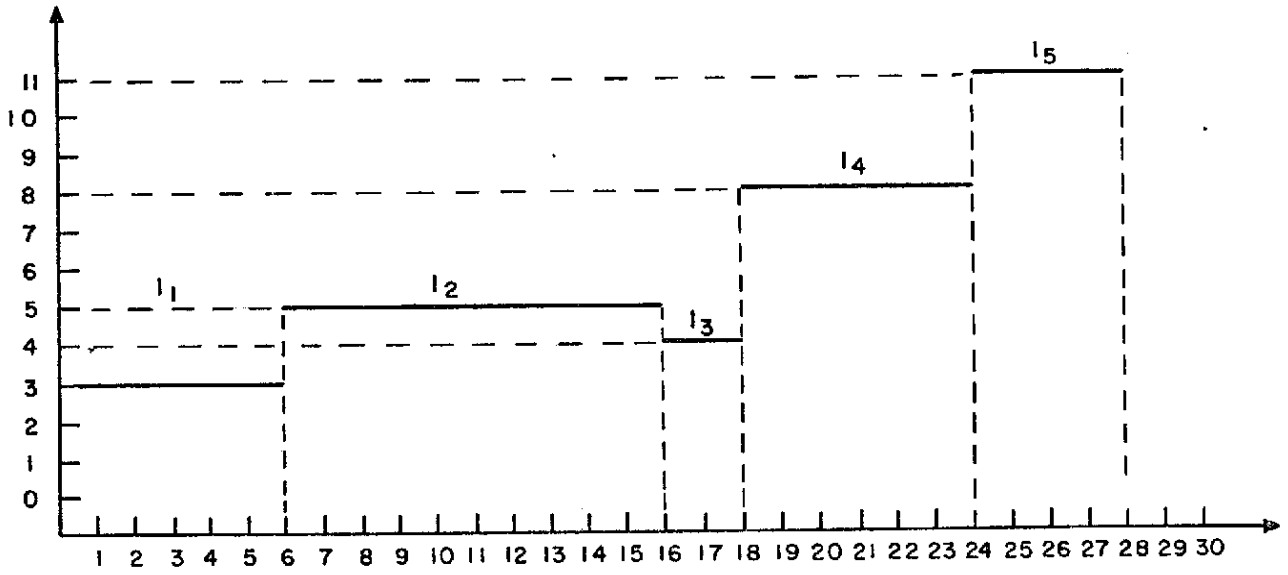
3. MÉTODOS DE COMPRESSÃO DE IMAGENS

Os métodos de compressão de imagens podem ser: sem perda de informação e com perda de informação (onde alguns critérios de fidelidade podem ser adotados). Aqui, serão tratados alguns métodos de compressão de imagens sem perda de informação.

3.1 - Método do Mapeamento

A operação de mapeamento (Gonzalez, 1977) consiste em mapear um conjunto de entradas de números ("pixels") em outro conjunto de números, como mostra a Figura 1.

Os g_i's são os valores dos "pixel's" e os ℓ_i's os números de vezes que os mesmos se repetem (contiguamente), que são armazenados, como na Figura 1.



i	g_i	ℓ_i
1	3	6
2	5	10
3	4	2
4	8	6
5	11	4

Fig. 1 - Operação de mapeamento

3.2 - Método da Distância Entre os "Pixel's"

Sejam $x_1, x_2, \ldots, x_n$ "pixel's" consecutivos, cujos valores pertencem ao conjunto $A = \{0,1,\ldots,255\}$; então, mapeia-se o conjunto de n números inteiros em um novo conjunto: $x_1, x_2 - x_1, \ldots, x_n - x_{n-1}$ (Gonzalez, 1977). Como, nas imagens, a diferença de um "pixel" para o seu sucessor ou o seu antecessor se repete bastante, pode-se mapear a sequência $x_1\ x_2 \ldots x_n$ na sequência $d_1 d_2 \ldots d_m$ $(m>n)$. Embora o dicionário de símbolos, para codificar os d_i $(1 \leq i \leq m)$, seja maior do que o dicionário de símbolos para codificar os x_j $(1 \leq j \leq n)$, existe uma grande quantidade de repetições em alguns d_i's, possibilitando assim uma média menor

de "pixel's" por caractere, do que se se codificasse os x_j (Tabelas 5 e 6 e Exemplo 6).

Exemplo 6. Seja a seguinte sequência

$1,2,4,5,6,7,8,10,9 \rightarrow x_i$ (não existe repetição)

$1,-2,-1,-1,-1,-1,-2,1 \rightarrow d_i$ (-1 se repete 4 vezes).

Caractere-código	Caractere-código
1 → 01	-1 → 01
2 → 001	-2 → 001
3 → 011	1 → 011
4 → 0001	
5 → 0011	bits por caractere=2,22
6 → 0111	
7 → 00001	
8 → 00011	
9 → 00111	

bits por caractere=4,33

Uma idéia interessante seria utilizar este método de mapeamento (explicado anteriormente), aplicando-o na Figura 1.

3.3 - Codificação de Contorno

O algoritmo de codificação de contornos, apresentado nesta seção, reduz a imagem a uma lista de contornos. Cada contorno é univocamente determinado ao especificar:

1) seu nível de cinza;
2) a localização (linha e coluna) de um "pixel" em sua fronteira, denominada "ponto inicial" (PI);
3) uma sequência de direções, que dá as direções locais do caminho que se deve traçar em torno do nível de cinza.

O algoritmo apresentado a seguir foi sugerido pelo autor deste trabalho.

3.3.1 - Algoritmo ATC (Algoritmo Traçador de Contorno)

Como mostra a Figura 2, traçar um contorno implica em definir a direção de um caminho entre elementos adjacentes, tal que nenhum elemento externo ao contorno, e adjacente a ele, tenha o mesmo nível de cinza que os elementos do contorno.

A regra para decidir a direção do caminho em cada elemento foi sugerida pelo autor e baseada na conhecida regra para encontrar o caminho de saída de um labirinto, seguindo-se sempre para esquerda "Leftmost-Looking Rule" (LML) (Gonzalez, 1977).

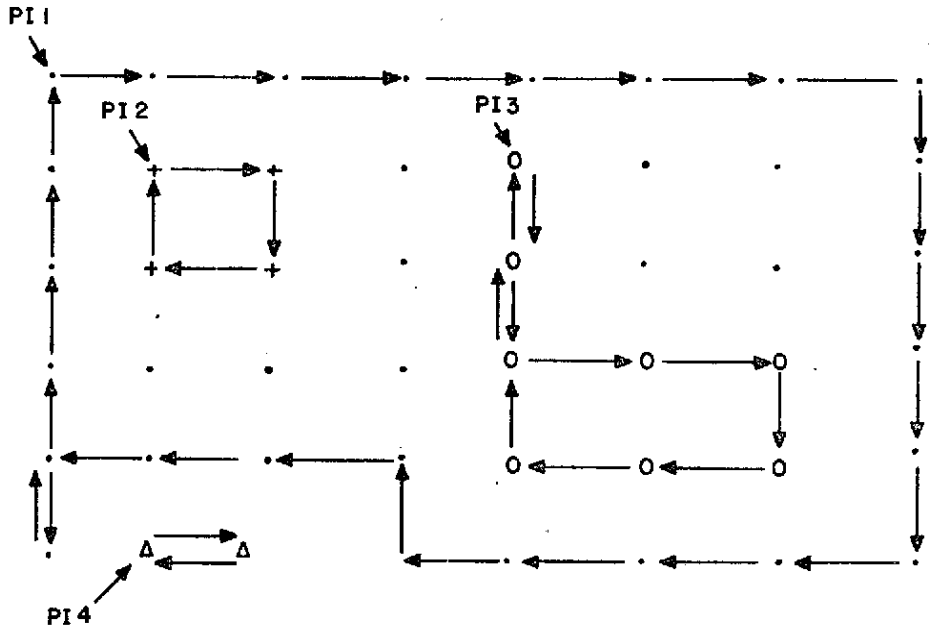


Fig. 2 - Traçado de contorno

Regra D.M.E. - Direção Mais à Esquerda

1) Estando no ponto inicial, examine o elemento acima dele (relativo ao ponto inicial). Se este elemento tem o mesmo valor, vá para este ponto acima, anote esta direção e vá para 4. Se não, vá para 2.

2) Mova o ponteiro para a próxima direção (seguindo os ponteiros do relógio) e examine o elemento nesta direção. Se este elemento tem o mesmo valor do anterior, vá para este ponto, anote esta direção e vá para 4. Caso contrá

rio, repita este passo até esgotar o número de direções diferentes; anote o valor do "pixel" e vá para 4.

3) Mova o ponteiro na direção perpendicular, para a esquerda, à direção anterior e examine o elemento correspondente a esta direção. Se nesta direção o "pixel" tiver o mesmo valor, vá para este ponto, anote a direção e vá para 4. Caso contrário, vá para o passo 2.
4) Este ponto é o inicial? Caso seja, termine o processo; caso contrário, vá para 3.

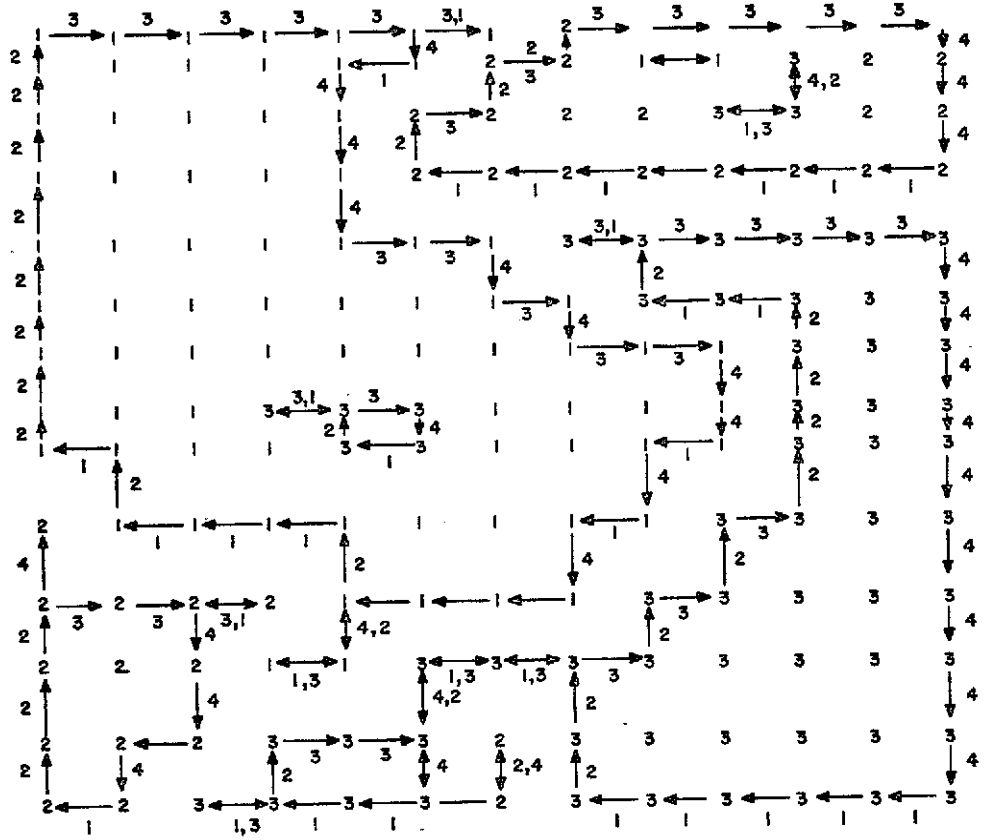
Definição 1 - Ponto Inicial (PI)

Ponto inicial é um ponto mais à esquerda em um dado contorno. O algoritmo para encontrar o ponto inicial é bastante simples. Uma imagem tem pelo menos um PI, uma vez que ela é finita.

Exemplos de pontos iniciais (PI) são vistos na Figura 2. Dois exemplos, executados em computador, são vistos nas Figuras 3 e 4.

Quando se codifica contornos, não é necessário codificar o contorno maior. A idéia aqui sugerida é codificar os contornos menores e, no contorno maior, guardar apenas o valor do "pixel" correspondente ao mesmo. Na decodificação, preenche-se o quadro correspondente à imagem com os contornos menores, e o que "sobrar" é preenchido com o valor do "pixel" que foi guardado para o contorno maior. A Tabela 5 apresenta a economia de memória.

A seguir, serão apresentadas as Tabelas 5 e 6, cujos resultados foram obtidos pelo autor no computador B6800, em linguagens ALGOL e FORTRAN, para imagens reais de satélites.



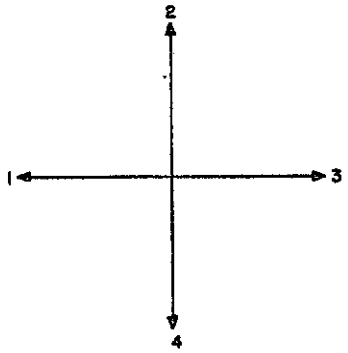


Fig. 3 - Contorno usando quatro direções.

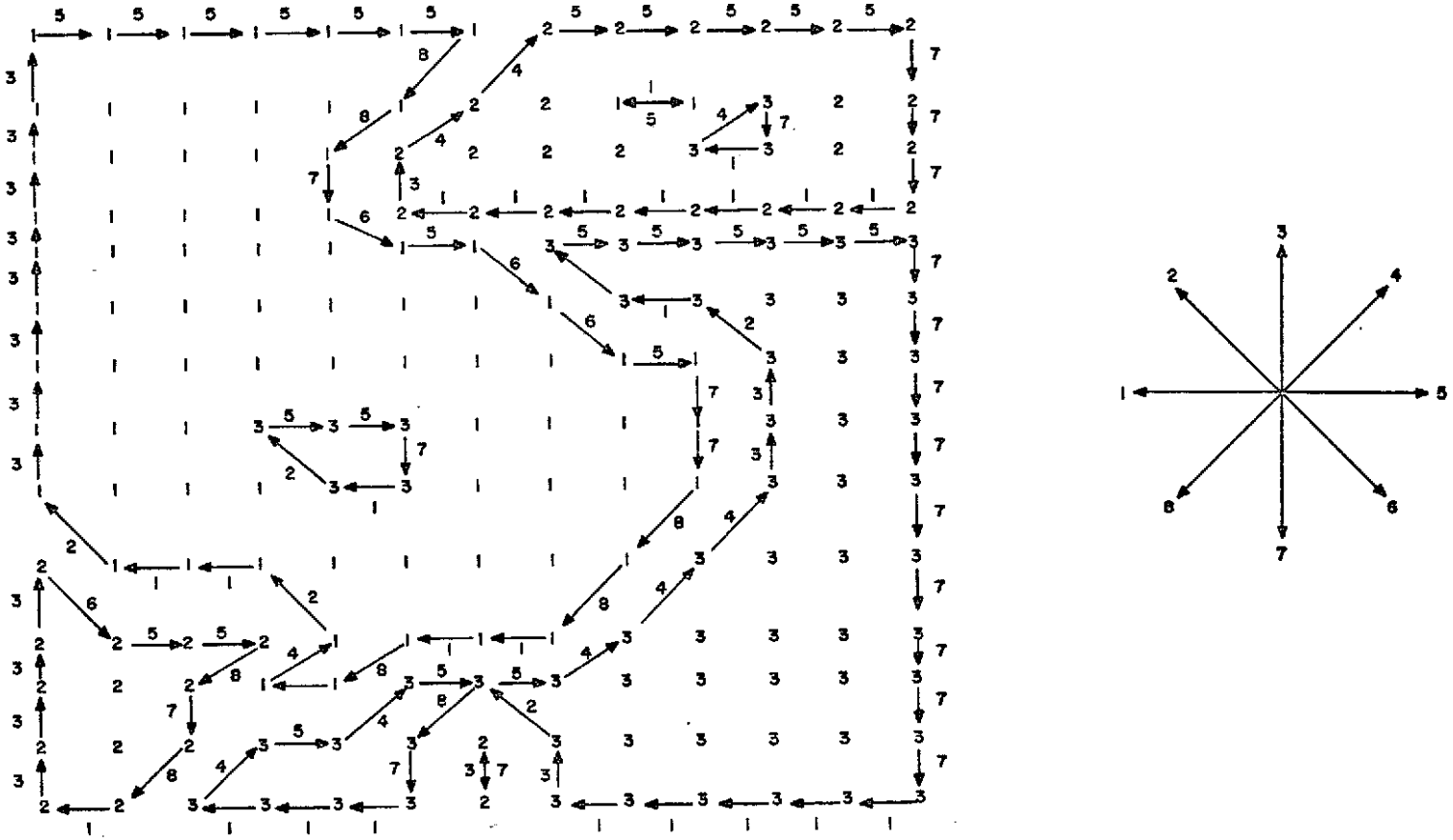


Fig. 4 - Contorno usando quatro direções.

TABELA 5

CODIFICAÇÃO DE IMAGENS CLASSIFICADAS (HOMOGÊNEAS)

BRUCE-HANS - SEM CODIFICAR CONTORNOS		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
16.384 "pixels"	2,182 bits/"pixel"	48 bits
HUFFMAN - SEM CODIFICAR CONTORNOS		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
16.384 "pixels"	2,215 bits/"pixel"	39 bits
BRUCE-HANS - CODIFICANDO CONTORNOS		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
16.384 "pixels"	1,647 bits/"pixel"	39 bits
HUFFMAN - CODIFICANDO CONTORNOS		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
16.384 "pixels"	0,439 bits/"pixel"	39 bits
HUFFMAN - SEM CODIFICAR O CONTORNO MAIOR		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
16.384 "pixels"	0,29 bits/"pixel"	39 bits
HUFFMAN - CODIFICANDO AS DIFERENÇAS		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
16.384 "pixels"	1,219 bits/"pixel"	39 bits

TABELA 6

CODIFICAÇÃO DE IMAGENS NÃO-CLASSIFICADAS[1]

BRUCE-HANS		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
65.160 "pixels"	5,270 bits/"pixel"	48 bits
65.160 "pixels"	5,680 bits/"pixel"	39 bits
BRUCE-HANS - USANDO DIFERENÇAS		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
65.160 "pixels"	5,387 bits/"pixel"	39 bits
HUFFMAN		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
65.160 "pixels"	5,043 bits/"pixel"	39 bits
HUFFMAN USANDO DIFERENÇAS		
IMAGEM ORIGINAL	BITS POR "PIXEL"	TAMANHO DA PALAVRA
65.160 "pixels"	3,758 bits/"pixel"	39 bits

[1] *Uma imagem utiliza normalmente 8 bits por valor do "pixel".*

Neste trabalho, geraram-se rotinas separadas em ALGOL e FORTRAN, que forneceram os resultados das Tabelas 5 e 6. Pretende-se melhorar estas rotinas e juntá-las num só "pacote" de "software", para que o mesmo seja usado pelos pesquisadores do Instituto de Pesquisas Espaciais (INPE).

4. CONCLUSÕES

Dada a necessidade de eficientes métodos de compressão de imagens, sugere-se a utilização dos métodos aqui apresentados, uma vez que estes forneceram bons resultados, quando testados com imagens reais.

Pretende-se fazer ainda análises em um grande número de imagens gravadas, já existentes na fitoteca do INPE.

Feito este estudo, escolher-se-ão métodos mais adequados para a compressão de imagens não-classificadas e para imagens classificadas, levando-se em consideração o tempo de processamento e a memória utilizada.

Concluiu-se, neste trabalho, que a codificação que usa contornos é mais apropriada a imagens com grandes regiões homogêneas (mesmo nível de cinza), como é o caso de imagens já classificadas. Concluiu-se também que, para imagens não-classificadas, o melhor método foi o de Huffman, que codifica as diferenças entre os valores dos "pixels", isto sem levar em conta o tempo de processamento, mas sim, a memória utilizada.

REFERÊNCIAS BIBLIOGRÁFICAS

SILVA, O.O. Algoritmos de compressão de dados. São José dos Campos, INPE, ago. 1980. (INPE-1878-RPE/223).

HUFFMAN, D.A. A method for the construction of minimum redundancy codes. Proceeding of the I.R.E., 40(10):1098--1101, Sept. 1955.

ABRAMSON, N. Information theory and coding. New York, McGraw-Hill, c1963. (McGraw-Hill Electronics Sciences Series).

HANS, B. A new technique for compression and storage of data. Communications of the ACM, 17(8): 434-436, Aug. 1974.

GONZALEZ, R.C. Digital image processing. Massachusetts, Addison Wesley, 1977.

SHANNON, C.E.; WEAVER, W. The mathematical theory of communication. Urbana, The University of Illinois Press, 1964.