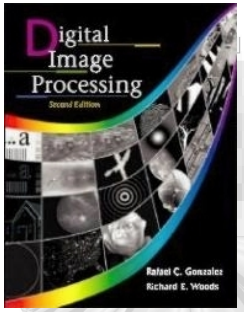


Aula 9.2

Compressão de Imagens

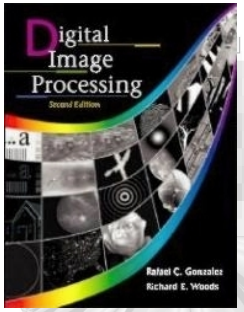


Image Compression

Estimativa de **segunda ordem**

Melhores resultados de estimativas da entropia podem ser gerados com um exame de frequência relativa dos blocos de pixels, que são agrupamentos de pixels adjacentes

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

1 2 3 4 5 6 7 8

Nível de cinza	Contagem	probabilidade
(21,21)	8	1/4
(21,95)	4	1/8
(95,169)	4	1/8
(169,243)	4	1/8
(243,243)	8	1/4
(243,21)	4	1/8

Na medida que o tamanho do bloco se aproxima do infinito, a estimativa se aproxima da verdadeira entropia da fonte

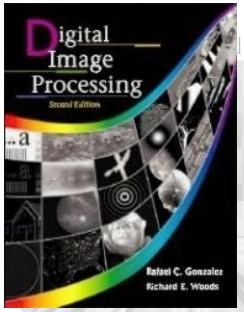


Image Compression

Estimativa de **segunda ordem**

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

Nível de cinza	Contagem	probabilidade
(21,21)	8	1/4
(21,95)	4	1/8
(95,169)	4	1/8
(169,243)	4	1/8
(243,243)	8	1/4
(243,21)	4	1/8

$$H(z) = -\sum_{j=1}^{J} P(a_j) \log P(a_j)$$

$$\ln(1/4)*1/4/\ln(2)+$$

$$\ln(1/8)*1/8/\ln(2)+$$

$$\ln(1/8)*1/8/\ln(2)+$$

$$\ln(1/8)*1/8/\ln(2)+$$

$$\ln(1/4)*1/4/\ln(2)+$$

$$\ln(1/8)*1/8/\ln(2) = 2.5/2 = 1.25$$

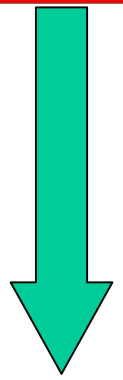
O 2 no denominador é porque usou blocos de tamanho 2

Image Compression

Resultados com a entropia

As diferenças entre a entropia de primeira ordem e as entropias de ordem mais elevadas significam haver redundância interpixels

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32



21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

Image Compression

Resultados com a entropia

A estimativa de primeira ordem é um limite inferior da compressão, que pode ser atingida por unicamente codificação de tamanho variado

Se as entropias de ordem superior forem iguais a de primeira ordem, não existe redundância interpixel e a codificação por tamanho variável fornece a compressão ótima

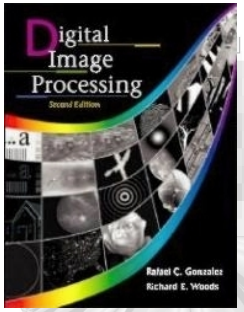


Image Compression

Resultados com a entropia

Embora as estimativas de terceira, quarta ou ordens superiores possam fornecer aproximações ainda melhores da entropia da fonte, a convergência destas estimativas para a verdadeira entropia é lenta e computacionalmente cara

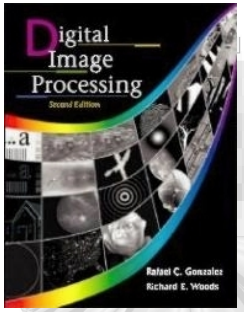


Image Compression

Resultados com a entropia

Embora o cálculo da verdadeira entropia de uma imagem seja difícil de obter, as estimativas superiores dão uma idéia melhor da compressibilidade da imagem

A estimativa de primeira ordem é apenas um limite inferior de compressão, que pode ser obtida por unicamente codificação de tamanho variável, tratada adiante com **Huffman**

Image Compression

Resultados com a entropia

As diferenças entre as estimativas de ordens mais elevadas da entropia e a de primeira ordem indicam a presença ou ausência de redundância interpixel

Ou seja, revelam se os pixels de uma imagem são estatisticamente independentes

No exemplo anterior, as estimativas 1.81 e 1.25, possuem um diferença de 0.56 bit/pixel indicam que é possível criar um mapeamento que elimine 0.56 bit/pixel

Image Compression

Resultados com a entropia

Exemplo: Usando um mapeamento baseado em diferenças

Considere o mapeamento dos pixels abaixo

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

Aqui usa-se uma matriz, mantendo-se a primeira coluna e, em seguida, são inseridas as diferenças da coluna atual para a anterior

Usando diferenças entre as colunas tem-se

21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0

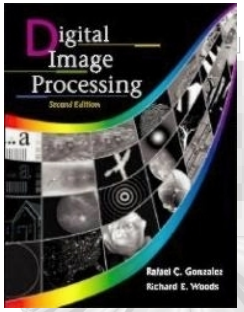


Image Compression

Resultados com a entropia

21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0

Assim, tem-se:

Nível de cinza	Contagem	Probabilidade
0	16	1/2
21	4	1/8
74	12	3/8

$$\ln(1/2)*1/2/\ln(2)+$$
$$\ln(1/8)*1/8/\ln(2)+$$
$$\ln(3/8)*3/8/\ln(2)=1.41$$

Logo, mesmo usando apenas um pixel de cada vez, porém, trabalhando com as diferenças, já se obtém uma entropia menor que antes (1.8112)

Image Compression

Resultados com a entropia

Como 1.41 > 1.25 (de segunda ordem), conclui-se que é possível obter um mapeamento ainda melhor

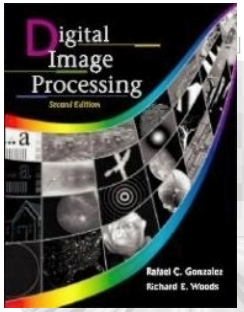


Image Compression

Resultados com a entropia

Suponha um mapeamento que faz também a diferença entre as diferenças nas linhas(*)

21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0

0	28/32
21	1/32
74	3/32

21	0	0	74	74	74	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0,64497

* o importante é que os mapeamentos sejam reversíveis

Image Compression

Compressão livre de erro

Em muitas situações, a compressão livre de erro é a única aceitável

- Winzip, WinRar (compactação de documentos e programas)
- Imagens médicas
- etc.

É comum usar os termos compactação e compressão como sinônimos, mas em PDI, não são

As principais técnicas nesta categoria são:

- Codificação por tamanho variável (Codificação de huffman)
- Codificação por planos de bits
- Codificação previsora sem perdas

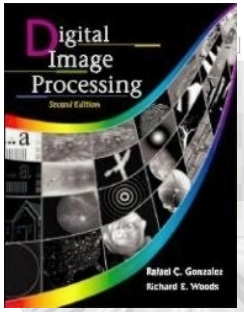


Image Compression

Compressão livre de erro

Codificação por tamanho variável

Atua de modo a minimizar a equação $$L_{médio} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

Codificação de Huffman

Esta que é a técnica mais comum, no seu primeiro passo, cria uma série de **reduções** de fonte

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

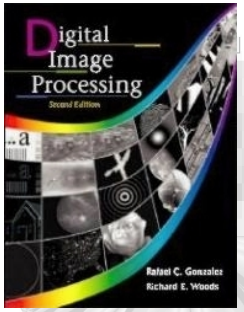


Image Compression

Compressão livre de erro

Codificação de Huffman

No segundo passo, é feita a codificação, começando com a menor fonte continuando para trás, até a fonte original

Original source			Source reduction							
Sym.	Prob.	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

Mantendo os maiores em cima

O tamanho médio deste código é $L_{médio} = \sum_{k=0}^{L-1} l_2(r_k)\, p_r(r_k) = 0{,}4(1) + 0{,}3(2) + 0{,}1(3) +$

$0{,}1(4) + 0{,}06(5) + 0{,}04(5) = 2{,}2 bits / simbolo$ e, como a entropia da fonte é 2,14 bits/símbolo, a eficiência do código de Huffman é 0,973

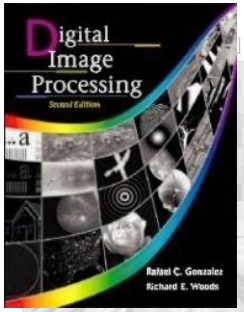
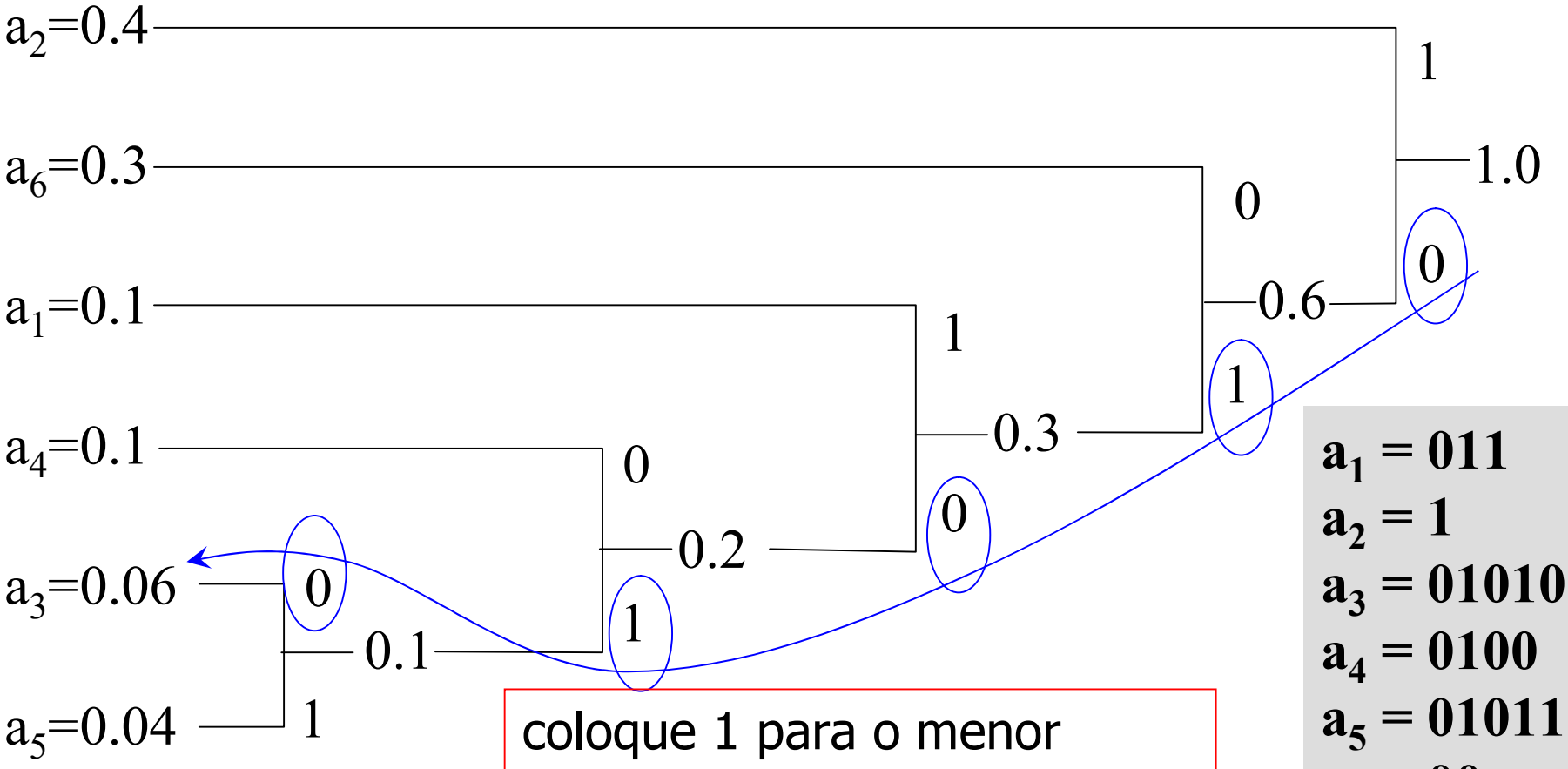


Image Compression

Compressão livre de erro

Codificação de Huffman

Fonte original	
Símbolo	Probabilidade
a_2	0,4
a_6	0,3
a_1	0,1
a_4	0,1
a_3	0,06
a_5	0,04



$a_1 = 011$
$a_2 = 1$
$a_3 = 01010$
$a_4 = 0100$
$a_5 = 01011$
$a_6 = 00$

Image Compression

Compressão livre de erro

Exercício 2: codificar "ARARAQUARA"

Total de 4 símbolos ➔ precisam de 2 bits cada

(A=00, R=01, Q=10 e U=11), como são 10 letras,

temos 2 bits x 10 letras = **20 bits para representar**

entregar agora

Resposta na aula 9a

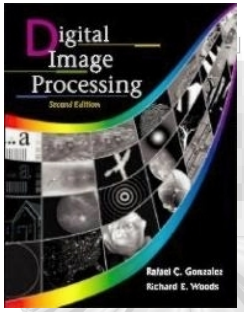


Image Compression

Compressão livre de erro

A = 0
R = 10
Q = 110
U = 111

Exemplo: codificar “ARARAQUARA”

Observe que, a partir dos código guardados, é preciso conhecer a tabela de símbolos, para a sua leitura (insere-se a tabela de símbolos no próprio arquivo da imagem)

exemplo: **0 1 0 0 1 0 0 1 1 0 1 1 1 0 1 0 0**

lê o primeiro 0 (zero), como tem símbolo A=0, vai ler “A”
lê o próximo 1 (um), como não tem um símbolo 1, pega o próximo 0 (zero), formando 10, como tem um símbolo 10, vai ler “R”

. . .

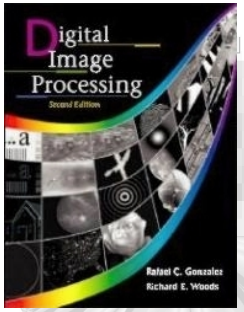


Image Compression

Compressão livre de erro

Quando se aplica Huffman em uma imagem, as cores ou tons de cinza dos pixels são os símbolos

Aqui usamos uma palavra para representar uma imagem e as letras para representar os pixels

Image Compression

Compressão livre de erro

Codificação por planos de bits

Esta técnica propõe decompor uma imagem com vários tons de cinza em várias imagens binárias e, em seguida, comprimí-la, usando uma técnica de compressão para dados binários

Os níveis de cinza de uma imagem de m bits podem ser decodificados na forma de um polinômio de base 2

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + ... + a_1 2^1 a_0 2^0$$

É interessante usar o **código gray** para evitar o problema de grandes mudanças no comportamento dos bits entre valores próximo, como por exemplo 127 e 128

01111111 e 10000000 (binário)

11000000 e 01000000 (gray)

Image Compression

Compressão livre de erro

Código binário para 4 bits

0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001
10 = 1010
11 = 1011
12 = 1100
13 = 1101
14 = 1110
15 = 1111

Código Gray para 4 bits

0 = 0000
1 = 0001
2 = 0011
3 = 0010
4 = 0110
5 = 0111
6 = 0101
7 = 0100
8 = 1100
9 = 1101
10 = 1111
11 = 1110
12 = 1010
13 = 1011
14 = 1001
15 = 1000

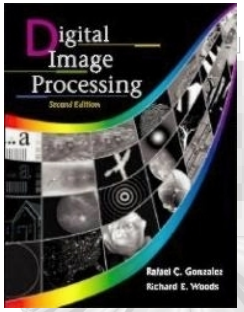


Image Compression

Compressão livre de erro

Gray é mais simples

Codificação por planos de bits

Bit 7

Bit 6

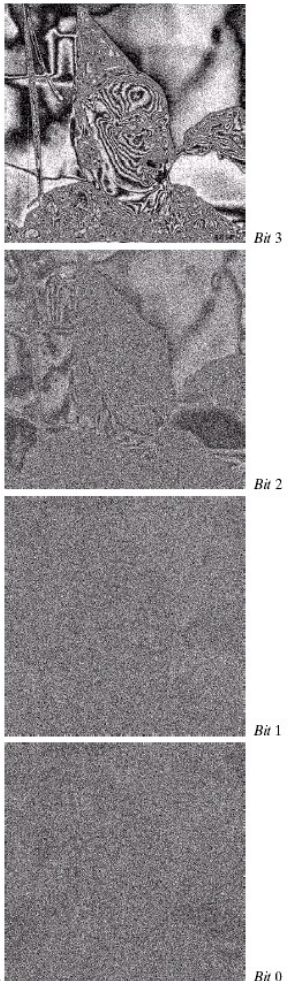
Bit 5

Bit 4

Binário

Gray

Binário



Gray

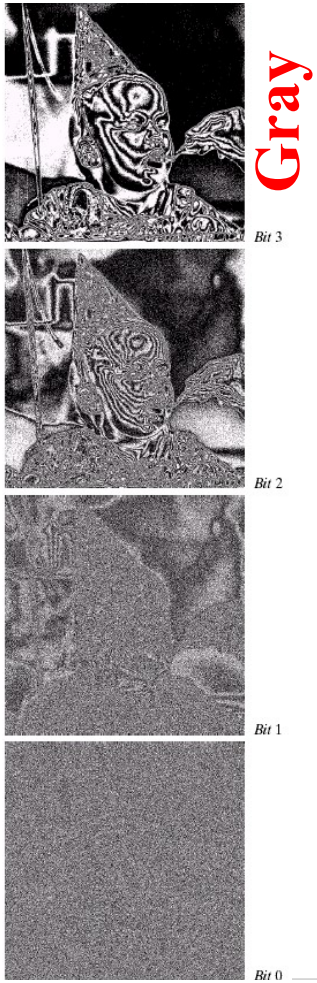


Image Compression

Compressão livre de erro

Codificação por planos de bits

Uma imagem com 2^8 tons de cinza (8 bits por pixel) vai gerar 8 imagens binárias, que deverão ser comprimidas

Compressão livre de erro

Codificação binária

Codificação de área constante (CAC)

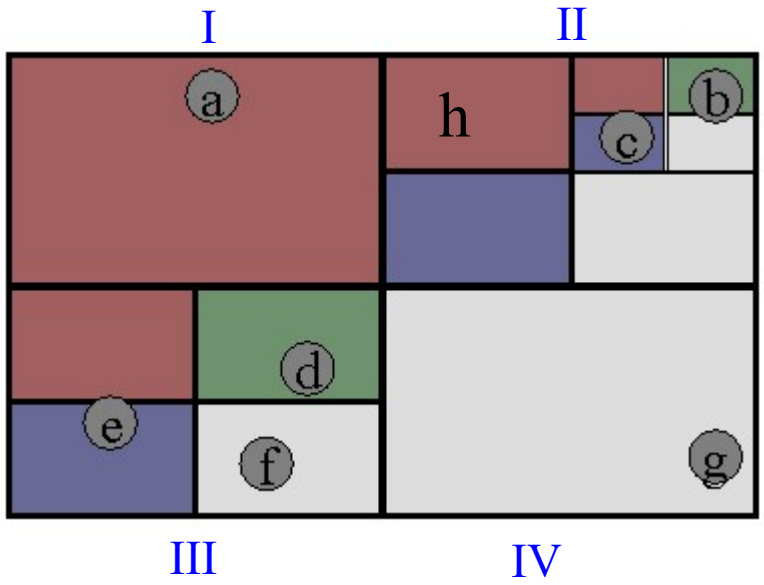
Usar palavras (códigos) para a identificação de grandes áreas de 1's ou 0's

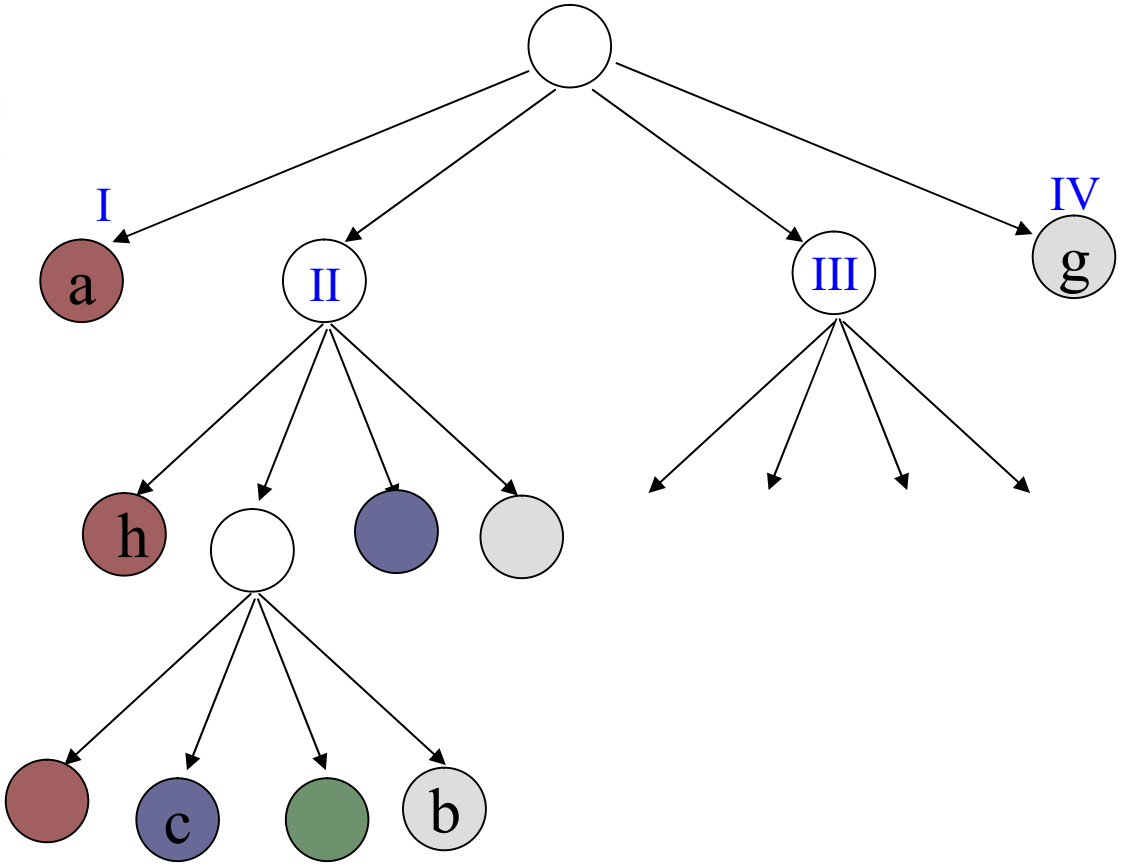
A imagem deve ser dividida até que os blocos contenham apenas um tipo de valor

Image Compression

Compressão livre de erro

Codificação de área constante (CAC)





Usando uma Quad Tree

Image Compression

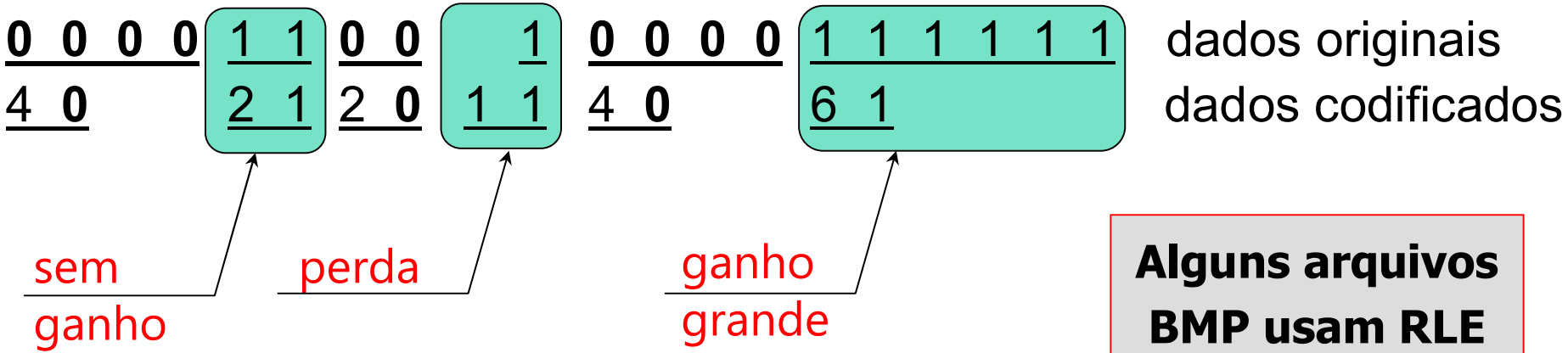
Compressão livre de erro

Codificação de corrida unidimensional (RLE – run-length encoding)

No lugar de usar regiões, usa as linhas da imagem

Anota-se a sequência de 0's e 1's em uma linha da imagem

Usar palavras (códigos) para a identificação de grandes linhas de 1's ou 0's



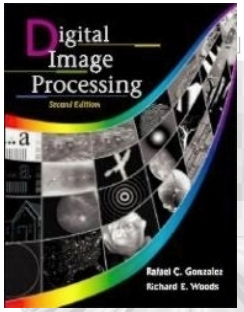


Image Compression

Compressão com ERRO ou PERDA

Em muitas situações, pequenas perdas, visualmente imperceptíveis, são aceitas, principalmente, quando se consegue altas taxas de compressão, como 30:1

A técnica mais conhecida nesta categoria é a codificação por transformada, em que uma operação de transformação (reversível) é aplicada aos dados

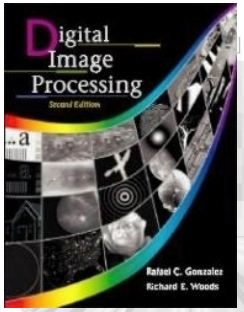


Image Compression

Compressão com ERRO ou PERDA

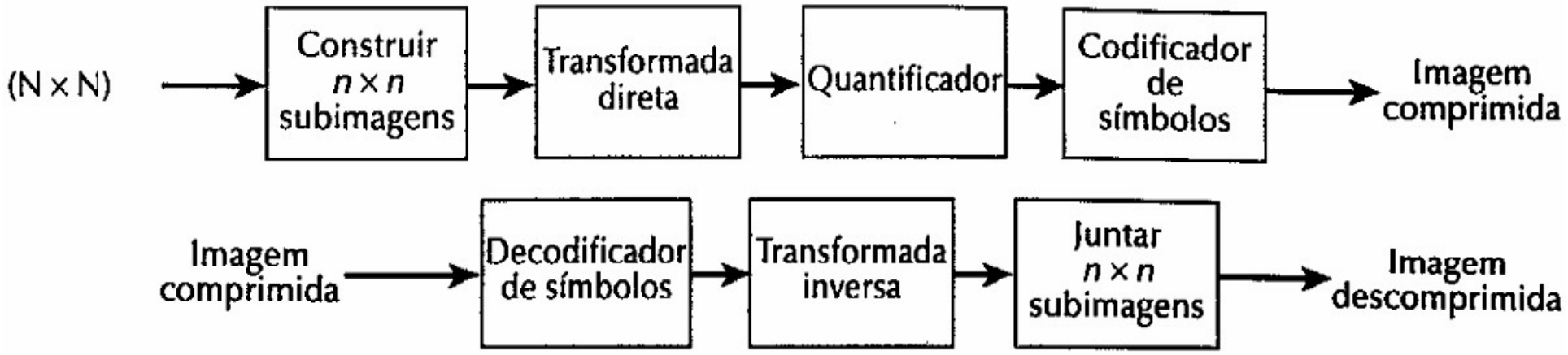
Em seguida os valores são **quantizados**, ou seja, divididos e truncados para um valor inteiro. Em seguida os valores obtidos são codificados e armazenados

No processo de decodificação, os dados são recuperados e decodificados, em seguida, a operação de **desquantização** é aplicada, sendo que agora os valores são multiplicados e, por fim é aplicada a transformada inversa, para obter os valores originais+erro

Image Compression

Compressão com erro

Codificação por transformada



As transformadas podem ser:
KLT – Karhunen-Loève
DFT – Discreta de Fourier
DCT – Discreta do Cosseno (JPG)
WHT – Walsh-Hadamard
etc.

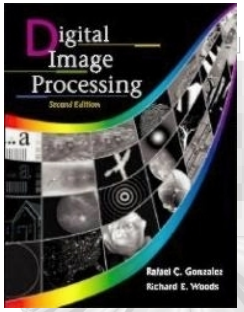
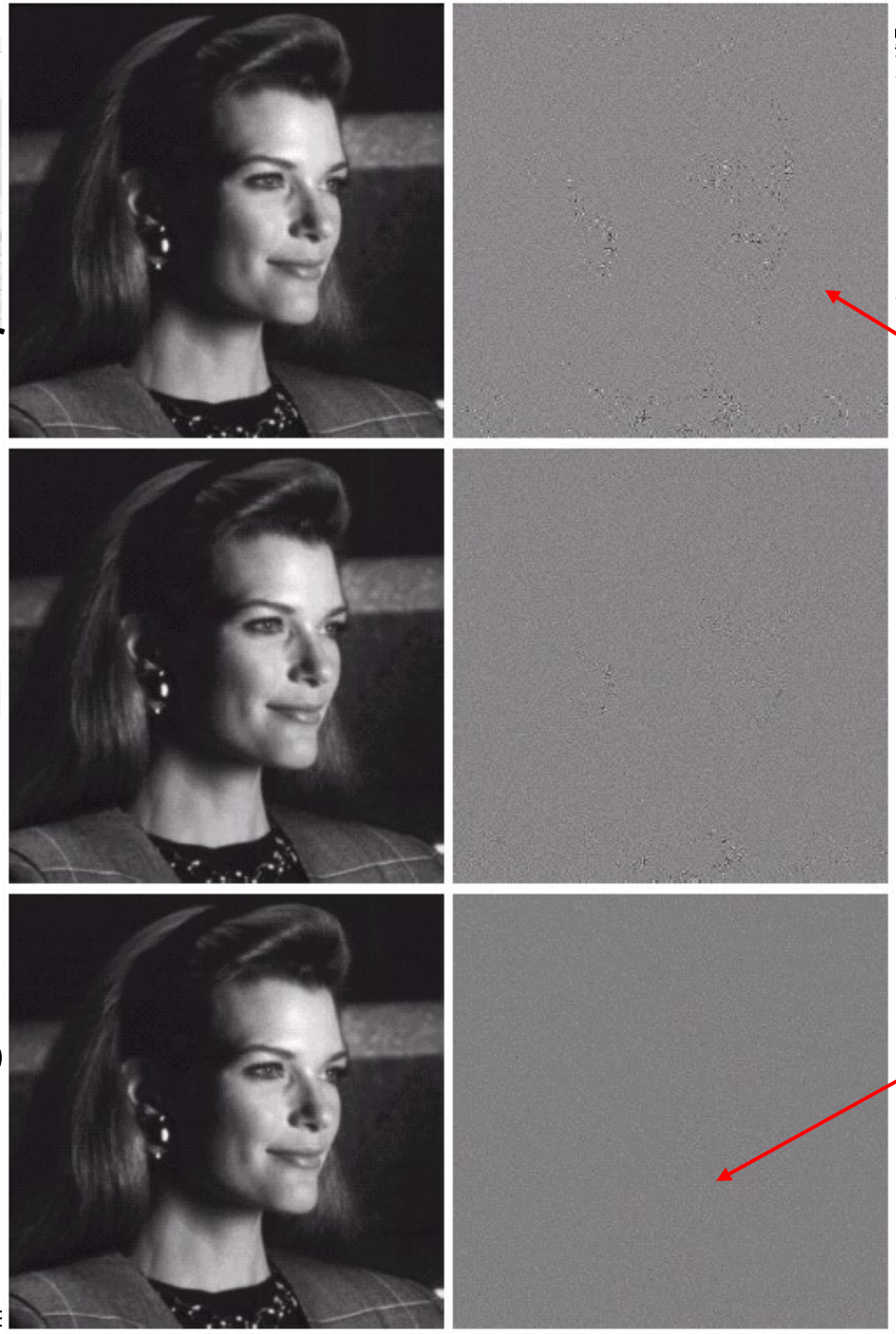


Image Compression

Fourier

Hadamard

Cosseno



Erros obtidos

Erros Maiores

Erros Menores

A maioria dos sistemas práticos usa a DCT