Processamento digital de imagens

Agostinho Brito

Departamento de Engenharia da Computação e Automação
Universidade Federal do Rio Grande do Norte

23 de novembro de 2016

Compressão de imagens

- Compressão de imagens engloba técnicas de reduzir a quantidade de bytes necessária para armazenar imagens com base na informação redundante presente nestas imagens.
- Considere um filme em padrão FULL HD (1920×1080 pontos). A quantidade de bytes/segundo que precisa ser acessada para que os quadros possam ser reproduzidos é de

$$30\frac{frames}{s} \times (1920 \times 1080) \times 3\frac{bytes}{pixel} = 186.624.000 bytes/s$$

- Para um filme com duração média de 2 horas, seriam necessários

$$186.624.000\frac{bytes}{s} \times 3600 segundos \times 2 \simeq 1.34 \times 10^{12} bytes$$

  ou aproximadamente 1.34 TB (Terabytes) de dados. Para isto, seriam necessários aproximadamente 285 dvds de camada simples para guardar o filme completo.
- Única solução: reduzir a redundância presente entre as images usando técnicas de compressão de dados

Compressão de Imagens

Possíveis abordagens

- **Não destrutiva**: a técnica possibilita reconstruir **EXATAMENTE** a imagem (ou vídeo) original que foi submetida à compressão.
- **Destrutiva**: parte das características são perdidas, mas permitem obter níveis elevados de compressão de dados.

Fundamentos

- Sejam n e n' duas representações de uma mesma informação (ex: quantidade de bits). A taxa de compressão é dada por

$$C = \frac{n}{n'}$$

- Se $C = 15$, por exemplo, diz-se que a representação n precisa de 15 bits de dados para cada bit da representação n'.
- A redundância relativa R pode ser definida como

$$R = 1 - \frac{1}{C}$$

- Se $R > 0,8$, diz-se que 80% dos dados são redundantes.
- Tipos de redundância
  1. **codificação**: o modo como a imagem é codificada introduz redundância.
  2. **inter-pixel**: a imagem mostra repetições nos padrões de pixels.
  3. **psico-visual**: a imagem inclui informação que não é visualmente relevante.

Redundância na codificação

- Se os tons de uma imagem não ocorrem com a mesma frequência, aqueles com maior probabilidade de ocorrência podem ser codificados com menos bits.
- O número médio de bits necessários para codificar uma imagem é dado pela soma dos números de bits usados para codificar cada tom r (denotado por $l(r_k)$) multiplicado pela probabilidade de ocorrência desse tom $p(l_k)$:

$$L_{med} = \sum_{k=0}^{L-1} l(r_k)p(r_k)$$

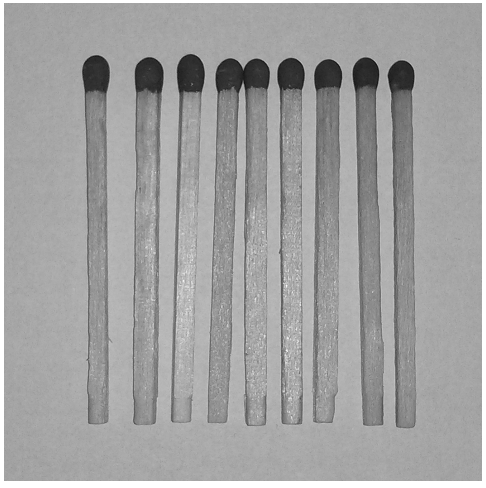
onde L é o número de tons na imagem.

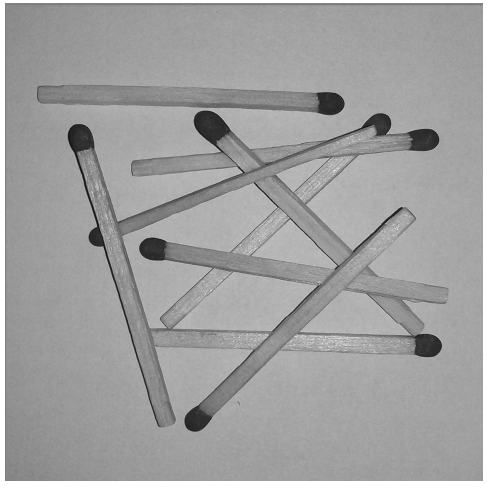
Exemplo

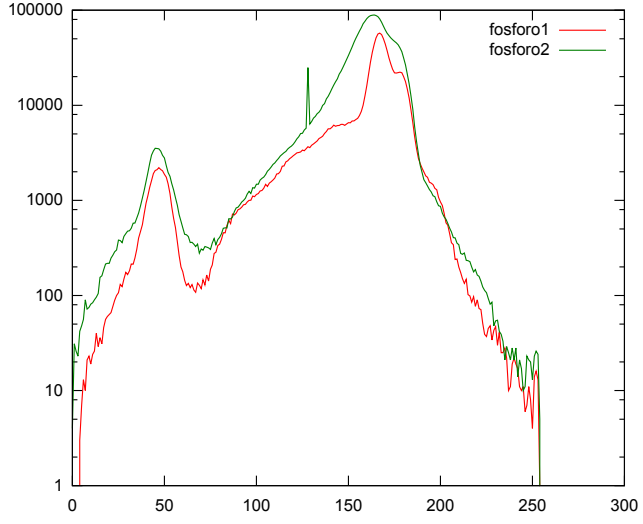
r_k	$p(r_k)$	Código 1	$l_1(r_k)$	Código2	$l_2(r_k)$
$r0 = 0$	0.19	000	3	11	2
$r1 = 1$	0.25	001	3	01	2
$r2 = 2$	0.21	010	3	10	2
$r3 = 3$	0.16	011	3	001	3
$r4 = 4$	0.08	100	3	0001	4
$r5 = 5$	0.06	101	3	00001	5
$r6 = 6$	0.03	110	3	000001	6
$r7 = 7$	0.02	111	3	000000	6

- O **Código 1** requer uma média de 3 bits/pixel, ao passo que o código 2 requer uma média de 2,7 bits/pixel.
- A taxa de compressão no processo para uma imagem de 8 bits é $C = \frac{8bits}{2,7bits/pixel} = 2,96$, indicando que $R = 1 - \frac{1}{2,96} = 0,663$, ou seja, cerca de 66,3% dos dados é redundante.
- O processo de codificação usa cadeias com comprimento variável (cada tom é codificado com um número diferente de bits).

Redundância inter-pixels

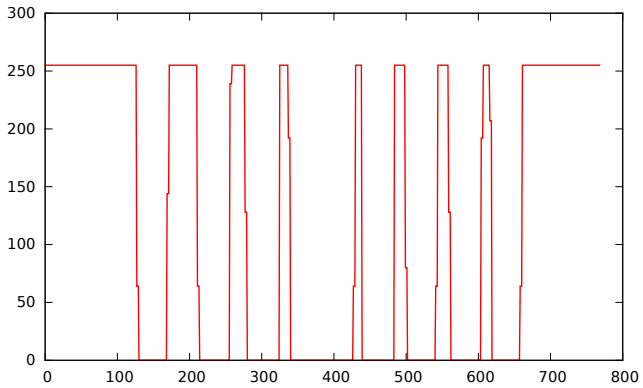






histogramas

Redundância inter-pixels



- Geralmente reduzida considerando a diferença entre pixels adjacentes na imagem (linha tomada nas cabeças dos fósforos).
- Ex: codificação por comprimento de corrida. (255,127) (64,3) (0,40) ...

Redundância psico-visual

- Critério subjetivo é envolvido na determinação da qualidade da compressão. Ex: redução do número de tons (8 bits $\rightarrow$ 4 bits).

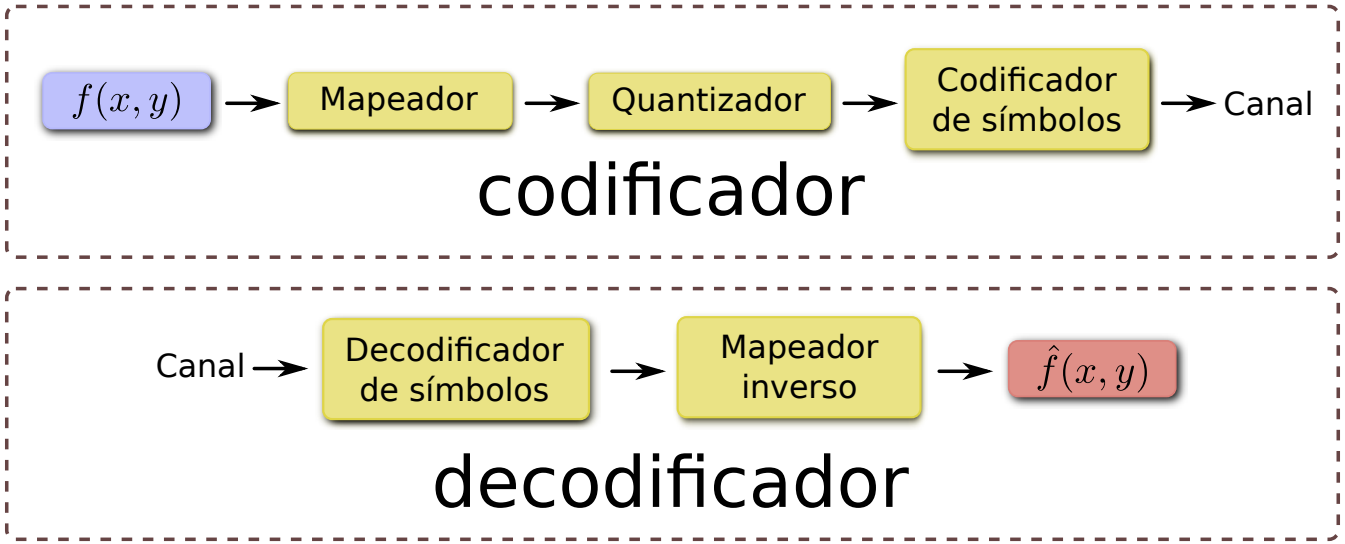


Critério de fidelidade

- O erro médio quadrático pode ser usado para avaliar a qualidade da compressão.

$$e_{rms} = \sqrt{\frac{1}{MN}\sum_{x=0}^{M}\sum_{y=0}^{N}[\hat{f}(x,y) - f(x,y)]^2}$$

Modelos de compressão



- **Mapeador**: converte a imagem para uma representação alternativa, visando reduzir redundância inter-pixel.
- **Quantizador**: reduz a qualidade do resultado do mapeador, de acordo com um critério de fidelidade.
- **Codificador de símbolos**: codifica os símbolos gerados pelo quantizador visando minimizar redundâncias.
- Em técnicas de compressão sem perdas, o quantizador não é usado.
- Apenas as operações realizadas pelo codificador de símbolos e pelo mapeador são reversíveis.

Medindo informações

- Mínima quantidade de dados para representar uma dada informação?
- A quantidade de informação (entropia) por elemento pode ser calculada pela probabilidade de ocorrência dos símbolos que a compõem, $P(a_j)$.
- Para ocorrência independente de símbolos, a entropia do elemento pode ser dada por

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

Medindo informações - Exemplo

- Qual a quantidade de informação presente na seguinte imagem?

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

- Sem redundância... 8 bits/pixel
- Estimativa de primeira ordem de entropia (símbolos independentes)

Cinza	Contagem	Probabilidade
21	12	$3/8$
95	4	$1/8$
169	4	$1/8$
243	12	$3/8$

$$H = -3/8\log(3/8) - 1/8\log(1/8) - 1/8\log(1/8) - 3/8\log(3/8) = 1,81 bits/pixel$$

Medindo informações - Exemplo (cont)

- Estimativa de segunda ordem (símbolos consecutivos)

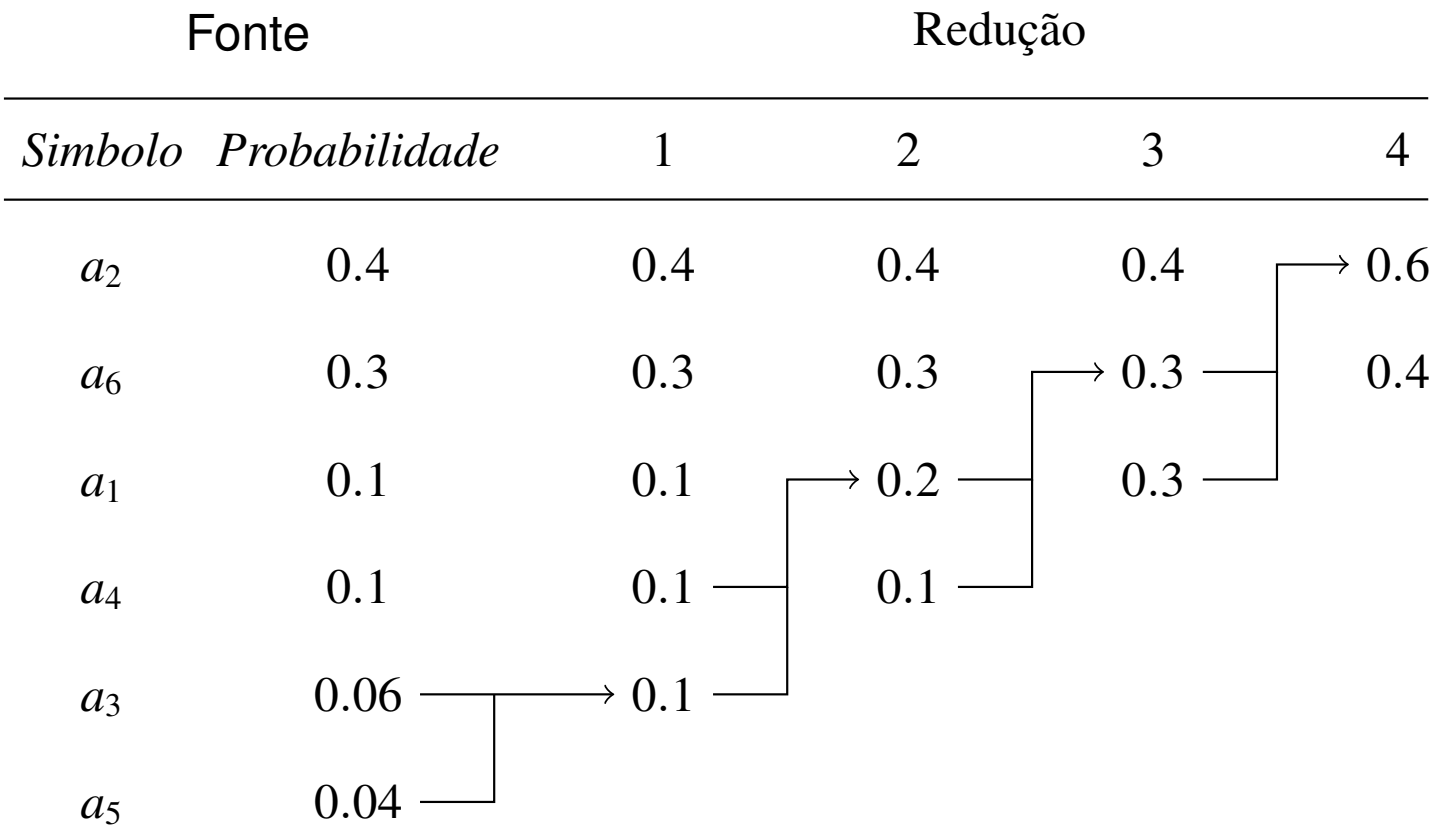
Cinza	Contagem	Probabilidade
(21,21)	8	$1/4$
(21,95)	4	$1/8$
(95,169)	4	$1/8$
(169,243)	4	$1/8$
(243,243)	8	$1/4$
(243,21)	4	$1/8$

$$H = 1,24 bits/pixel$$

- Estimativas de ordem superior são mais complicadas, pois geram números excessivos de combinações.
- Estimativa de primeira ordem: indicam possibilidade de compressão com comprimento variável.
- Diferença entre segunda e primeira ordem: indica redundância inter-pixels.

Codificação sem perdas (código de Huffman)

- Codificação com códigos de comprimento variável, proporcionando o menor número médio de bits por símbolo quando não existe redundância entre pixels.
- **1:** Símbolos são ordenados com probabilidade decrescente, sendo somados os símbolos de probabilidade menor (aglomerando símbolos) até que restem apenas dois elementos na redução.



Código de Huffman (cont.)

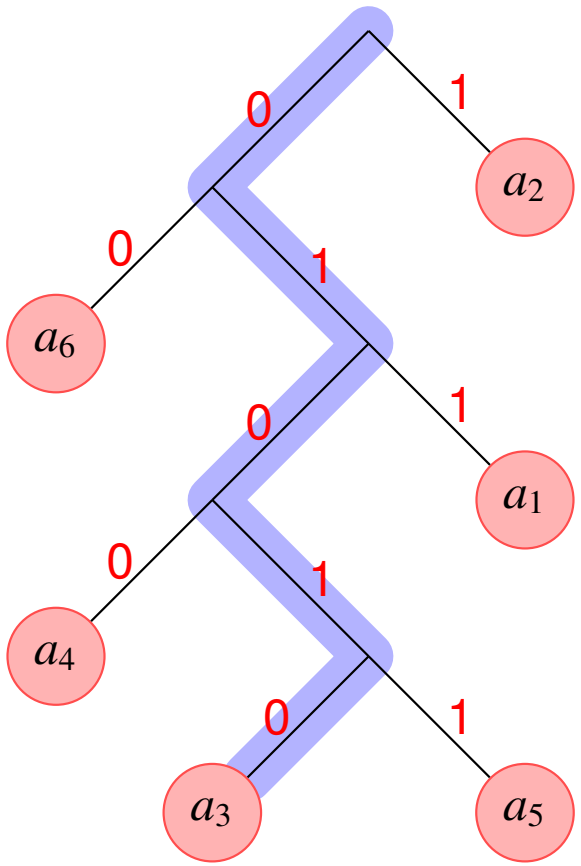
- **2:** Os símbolos são codificados adicionando novos bits a cada soma inversa que é realizada.

	Fonte		Redução							
Simbolo	*Probab.*	*Codigo*	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								

- Para este exemplo, a codificação de Huffman produziu média de 2,2 bits/pixel.
- Para cada símbolo existe um código.
- Inadequada quando existem muitos símbolos

Decodificação

- Percorre a árvore binária criada pelo código até encontrar uma folha.
- Ex: $01010 \rightarrow a_3$.



Compressão sem perdas (LZW - Lempel-ZivWelch)

- Atribui códigos de comprimento fixo a palavras de comprimento variável.
- Usado para compressão de informação que não se conhece a priori.
- Palavras são inseridas em um dicionário construído dinamicamente.
- 0-255: já pertencentes ao dicionário.
- Escolha de palavras e tamanho do dicionário são parâmetros livres.

Algoritmo 1 Codificação LZW

1: No início o dicionário contém todos os tons de cinza possíveis e a sequência S é vazia
2: **while** Sequência de entrada contiver tons de cinza **do**
3: Leia c, o próximo tom de cinza da sequência de entrada
4: **if** sequência S+c existe no dicionário **then**
5: S = S+c
6: **else**
7: Insira a sequência S na saída codificada
8: Adicione a sequência S+c ao dicionário
9: S = c
10: **end if**
11: **end while**
12: insira o código S na saída codificada

Compressão sem perdas (LZW - Lempel-ZivWelch) - exemplo

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

Sequência Reconhecida (**S**)	Pixel (**c**)	Saída	Dicionário (índice)	Dicionário (entrada)
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

Compressão sem perdas

- Codificação de planos de bits: imagem é decomposta em planos de bits, sendo estes comprimidos individualmente
  - Codificação em tom-duração (comprimento de corrida) ou pela divisão da imagem em blocos
  - Usando códigos de gray (conversão): tons com valores adjacentes diferem em apenas um bit. Ex: para $127 \rightarrow 128$ adjacentes apenas o plano de mais alta ordem sofrerá alteração.
- Codificação com previsão: função prevê os tons dos pixels seguintes com base nos anteriores, armazenando apenas a diferença entre o valor previsto e o valor efetivo.
  - A função de previsão geralmente é linear e usa os pixels presentes na linha

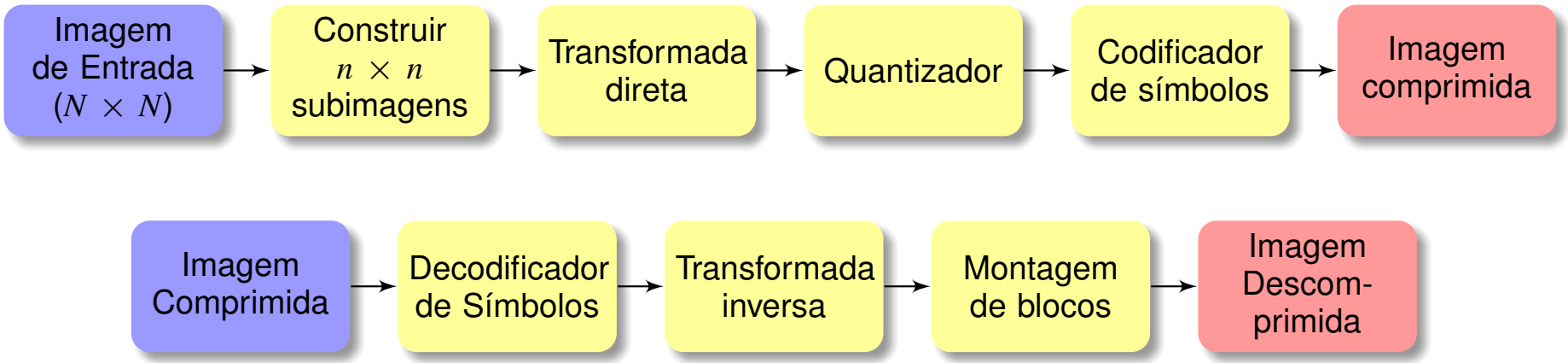
$$\hat{f}_n(x,y) = round\left[\sum_{i=1} \alpha_i f(x,y-1)\right]$$

  - Exemplo:

$$\hat{f}_n(x,y) = round[f(x,y-1)]$$

Compressão com perdas

- Na codificação com o uso de transformadas, obtém-se uma nova representação para a imagem (Ex: FFT) e realiza-se a quantização da representação.

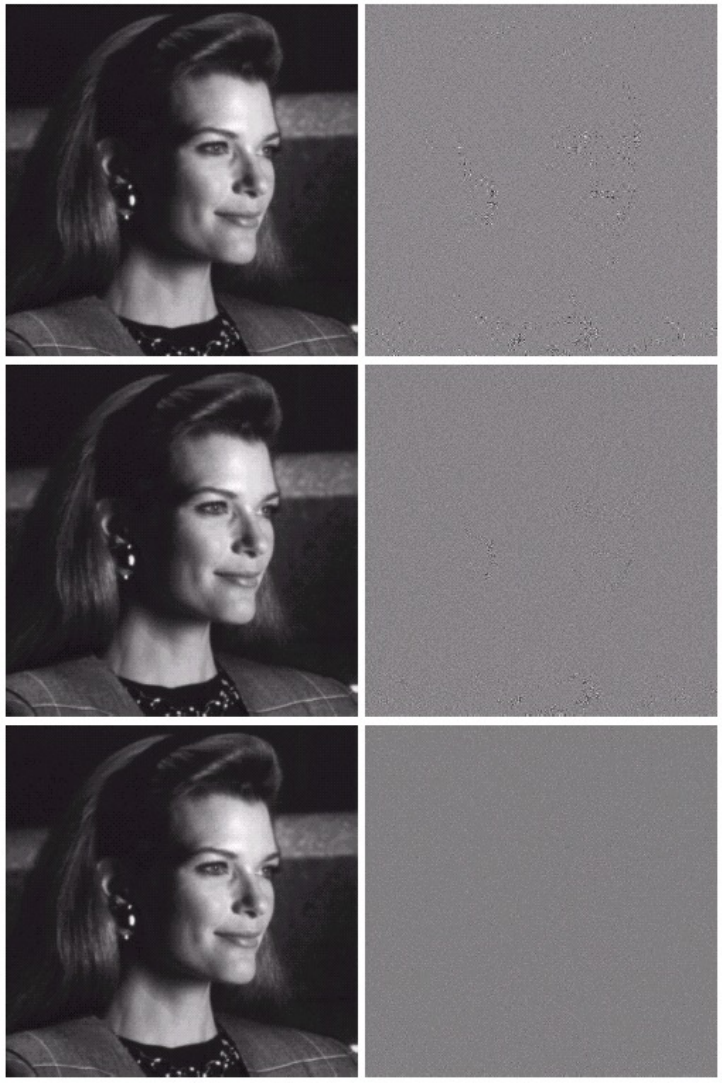


- Imagem é inicialmente dividida em blocos (inclusive de tamanho irregular).
- Quantizador elimina coeficientes com baixa resposta na qualidade visual.
- Codificador codifica os símbolos que não foram eliminados.

Compressão de imagens

Compressão com perdas

- Comparação FFT, WHT e DCT, desprezando 50% dos coeficientes.



Compressão com perdas - Transformada discreta de cosseno

$$T(u,v) = \sum_{x=0}^{n-1}\sum_{y=0}^{n-1} f(x,y)\alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2y+1)v\pi}{2n}\right]$$

$$f(x,y) = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2y+1)v\pi}{2n}\right]$$

onde

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{n}} & \text{para } u = 0 \\ \sqrt{\frac{2}{n}} & \text{para } u = 1, 2, \cdots, n-1 \end{cases}$$

- A imagem é dividida em blocos de tamanho menor 8×8, 16×16, e então codificada.

Compressão com perdas - Algoritmo JPEG

- Subdividir imagem em blocos de 8×8 pixels
- Deslocar os valores dos pixels do bloco em -128 níveis.
- Calcular a DCT direta da matriz.
- Realizar a normalização dos dados usando uma matriz especial Z.

$$\hat{T}(i,j) = arred\left[\frac{T(i,j)}{Z(i,j)}\right]$$

- Recuperar a sequência em zig-zag e codificá-la usando Huffman.
- A diferença entre os valores DC da subimagem atual e a da previamente codificada é usada para a escolha dos códigos.

Algoritmo JPEG - Exemplo

55	52	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

Bloco 8×8

-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-38	-19	-43	-59	-56
-66	-69	-60	-15	16	-24	-62	-55
-65	-70	-57	-6	26	-22	-58	-59
-61	-67	-60	-24	-2	-40	-60	-58
-49	-63	-68	-58	-51	-65	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	-60	-63	-52	-50	-34

Deslocamento de -128

-415	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

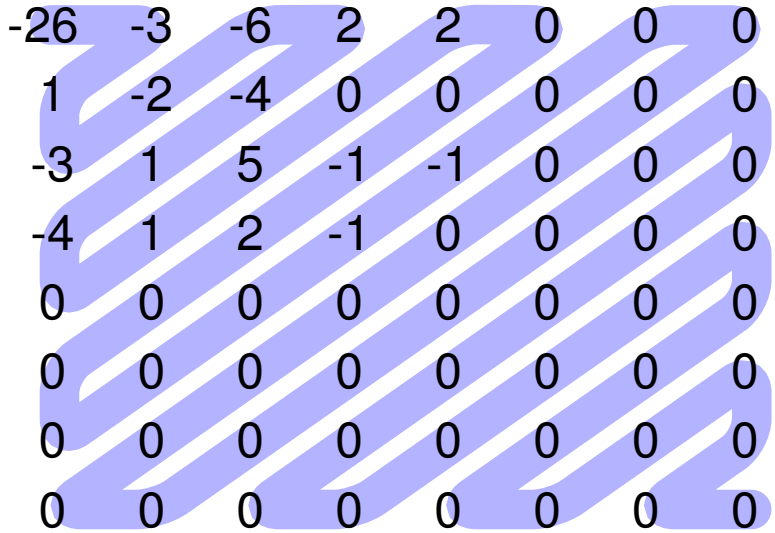
DCT

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	8	104	113	92
49	64	78	87	103	121	120	101
72	92	95	95	98	100	103	99

Matriz de Normalização

Compressão de imagens

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	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	0
0	0	0	0	0	0	0	0



- Sequência comprimida: -26 -3 1 -3 -2 -6 2 -4 1 -4 1 1 5 0 2 0 -1 2 0 0 0 0 -1 -1 EOB
- Deve-se codificar os valores DC e AC do sinal transformado.
- O valor da componente DC é o primeiro coeficiente da sequência (-26). Codifica-se a diferença entre o valor DC do bloco e o valor DC do bloco anterior.
- Ex: se valor DC do bloco anterior for -17, $(-26 - (-17)) = -9$.

Codificação dos coeficientes DC

- Para que quantizar? Por causa da semelhança entre quadros vizinhos.
- Como codificar -9? O padrão define faixas de valores e respectivas quantidades de bytes para cada faixa. Para as faixas $[-15, -8]$ ou $[8, 15]$, são previstos 3 bytes para codificar a categoria da diferença (101), mais 4 bytes restantes para codificar a própria diferença.
- Toma-se os K dígitos menos significantes da diferença positiva ou os K dígitos da diferença negativa menos 1.
- Considerando apenas os 8 últimos bits, $(-9)_{10} = (11110110)_2 + (1)_2 = (11110111)_2$
- Para os 4 bytes restantes, toma-se os 4 dígitos menos significativos, subtraindo-se 1 do valor: $(0111)_2 - 1 = (0110)$
- O valor total da palavra codificada é 1010110

codificação AC

- Para cada componente AC, são previstas três informações:
  - A quantidade de zeros que precede um valor diferente de zero
  - O número de bits necessários para codificar a quantidade diferente de zero.
  - A quantidade a ser codificada.
- Ex: Codificar o coeficiente -3.
  - Considerando apenas os 8 últimos bits, $(-3)_{10} = (11111101)_2$.
  - Realizando a subtração de 1: $(11111101)_2 - (1)_2 = (11111100)_2$
  - O valor -3 prevê codificação base 01 para nenhum ZERO anterior a este e um total de 4 bits para o comprimento do código.
  - Adicionando-se os 2bits $(= 4bits - 2bits)$ menos significativos do valor codificado $(11111100)_2$, compõe-se o código completo a ser inserido na cadeia codificada.
  - Código gerado: 0100
- Para cadeias longas de mais de 15 zeros seguidos por um zero, adiciona-se um código especial (111111110111). Ele é necessário pois os códigos de Huffmann não devem exceder 16 bits para o JPEG.
- Para a sequência acima, o código gerado seria:
  1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001 001 100101 11100110 110110 0110 11110100 000 1010