JPG

P.Kantek
UFPR
2018

JPEG stands for Joint Photographic Experts Group who created the standard.

A digitalização do nosso mundo

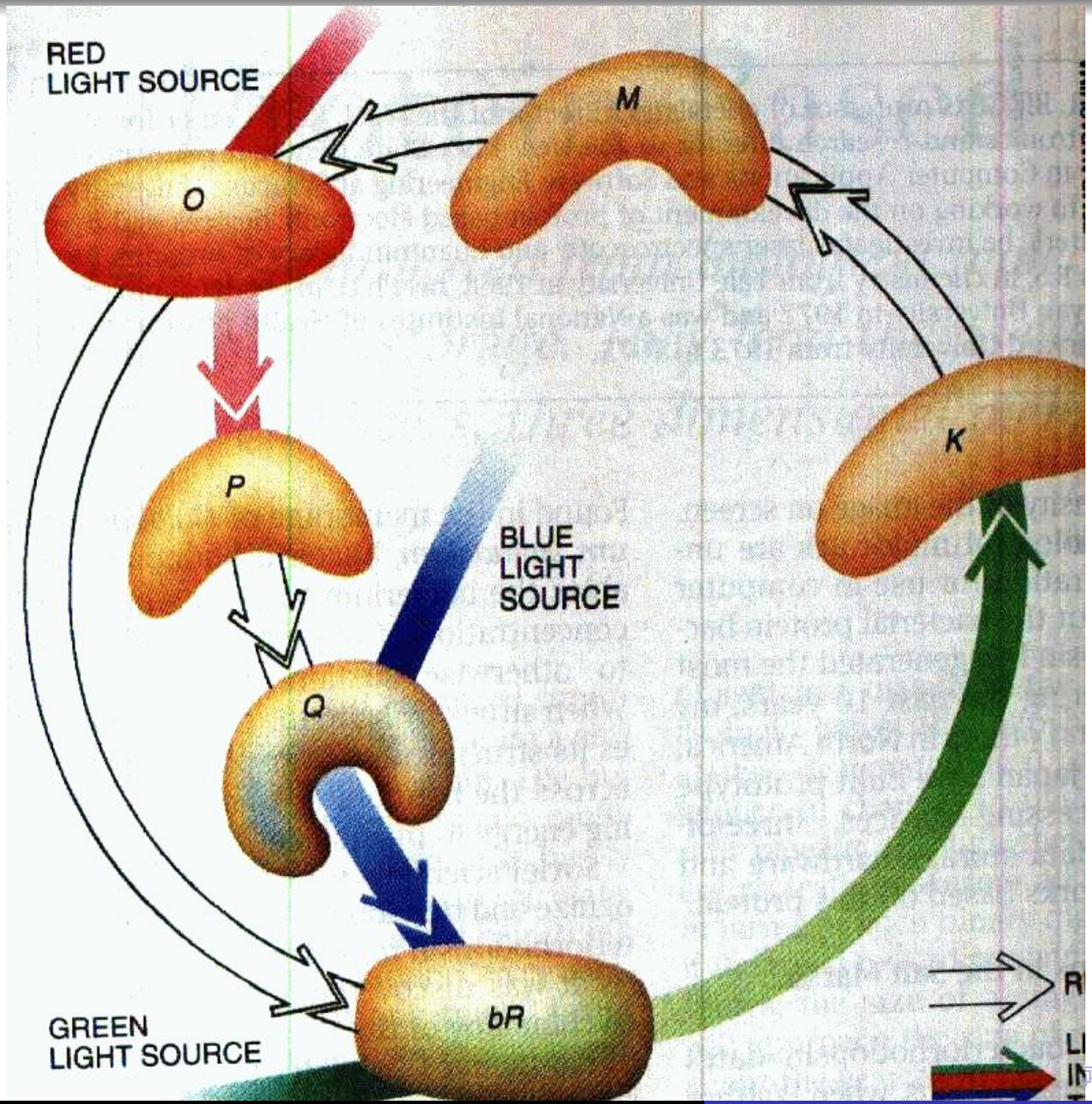
papel	PDF (Boeing 747)
música	MP3
dados científicos	WEB (Tim Berners Lee)
vídeo	Net Flix
taxi	Uber
hotel	AirBNB
dinheiro	bitcoin
inscrição no ...	www.algumacoisa.com
carta	e-mail
telegrama	whats-app
compras	www.algumacoisa...
votação	urna eletrônica
???	???

A pergunta que não quer calar: somos feitos de átomos ou somos feitos de bits ?

Imagens digitais

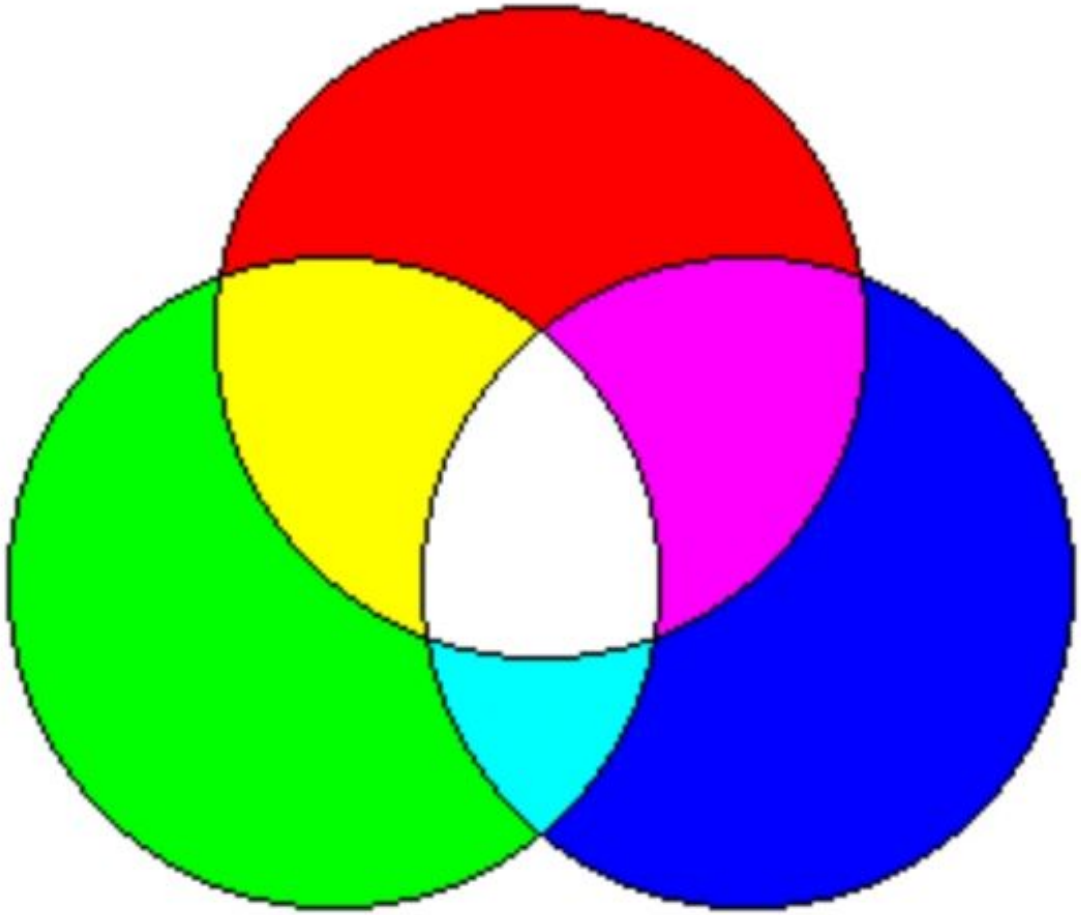
Pixel: picture element (elemento da imagem) $\rightarrow$ ponto
Na fotografia original: grão de sais de prata (iodeto)
Na fotografia digital: um número (monocromática) ou 3 números (colorida)

Rodopsina


RED
LIGHT SOURCE
O
M
K
P
BLUE
LIGHT
SOURCE
Q
bR
GREEN
LIGHT SOURCE

RGB

Foto: http://andredesign8.blogspot.com



Alguns números

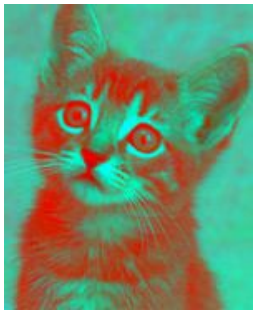
Uma imagem de TV HD: $1080 \times 1920 = 2073600$ pixels.
Se cada um ocupar 1 byte, serão 2025 Kbytes ou 1.97 Mbytes
Se cada um ocupar 3 bytes (RGB) serão 6075 Kbytes ou 5.93 Mbytes POR IMAGEM
a cada 30 segundos: serão 177.97 MB por segundo
1CD é capaz de armazenar 619 MB, logo sem compressão, uma TV precisaria ler um CD a cada 3.4 segundos...
Padrões:
RAW = padrão das câmeras de fotografia
BMP = origem do OS2 em 1984...
GIF = padrão na internet, permite compressão LZW e animação. Copyright
PNG = padrão GIF sem Copyright
JPG = compressão com perda (Localidade de referência espacial)

Antes de estudar o padrão JPG, algumas manipulações...

```
       blu[ı5;ı20]
182 183 184 183 173 141  99  65 52 38 32 43 60 65 49 32 46 61 68 59
185 181 177 175 172 152 124  98 49 39 29 26 31 40 49 53 44 55 61 52
178 178 174 166 157 140 124 109 55 50 40 32 28 32 40 47 38 48 60 59
181 181 176 164 146 122  97  77 45 41 45 57 70 67 46 26 25 28 42 54
166 162 156 155 153 138 106  76 32 36 41 40 33 33 39 45 39 27 29 43
       red[ı5;ı20]
197 198 203 204 197 170 130 100 90 78 73 84 101 106 90 73 87 102 107  98
200 198 196 196 196 181 155 132 85 79 70 67  72  81 90 94 85  96 100  91
195 195 193 187 181 169 155 143 91 86 80 72  68  72 80 88 79  91 101 100
198 198 195 183 170 148 128 110 79 77 83 97 110 107 86 67 68  71  83  95
183 179 175 174 177 164 137 109 66 71 77 76  73  73 79 86 82  71  71  85
       gre[ı5;ı20]
205 206 207 208 197 169 126  95 81 69 62 73  90 95 79 62 76 91 96 87
208 203 200 200 196 180 151 127 78 70 59 56  61 70 79 83 74 85 89 80
200 200 197 191 181 168 151 138 84 79 71 63  59 63 71 77 68 77 87 86
203 203 199 187 170 148 124 106 74 70 74 88 101 98 77 56 54 57 69 81
188 184 179 178 177 164 133 105 61 66 70 69  64 64 70 75 68 56 55 69
```

Negativa o componente vermelho

```
      red[⍳5;⍳20]
197 198 203 204 197 170 130 100 90 78 73 84 101 106 90 73 87 102 107  98
200 198 196 196 196 181 155 132 85 79 70 67  72  81 90 94 85  96 100  91
195 195 193 187 181 169 155 143 91 86 80 72  68  72 80 88 79  91 101 100
198 198 195 183 170 148 128 110 79 77 83 97 110 107 86 67 68  71  83  95
183 179 175 174 177 164 137 109 66 71 77 76  73  73 79 86 82  71  71  85
      red←255-red
      red[⍳5;⍳20]
58 57 52 51 58  85 125 155 165 177 182 171 154 149 165 182 168 153 148 157
55 57 59 59 59  74 100 123 170 176 185 188 183 174 165 161 170 159 155 164
60 60 62 68 74  86 100 112 164 169 175 183 187 183 175 167 176 164 154 155
57 57 60 72 85 107 127 145 176 178 172 158 145 148 169 188 187 184 172 160
72 76 80 81 78  91 118 146 189 184 178 179 182 182 176 169 173 184 184 170
      grave 'c:\apl2\lixo11.bmp'
feito... Tempo gasto =      .5
```



Multiplica o componente azul por 2

```
        leia 'c:\apl2\im11.bmp'
criados "blu" (matriz azul), "gre" (verde) e "red" (vermelho)
      blu[⍳5;⍳20]
182 183 184 183 173 141  99  65 52 38 32 43 60 65 49 32 46 61 68 59
185 181 177 175 172 152 124  98 49 39 29 26 31 40 49 53 44 55 61 52
178 178 174 166 157 140 124 109 55 50 40 32 28 32 40 47 38 48 60 59
181 181 176 164 146 122  97  77 45 41 45 57 70 67 46 26 25 28 42 54
166 162 156 155 153 138 106  76 32 36 41 40 33 33 39 45 39 27 29 43
       blu ← 255⌊blu×2
       blu[⍳5;⍳20]
255 255 255 255 255 255 198 130 104  76 64  86 120 130 98  64 92 122 136 118
255 255 255 255 255 255 248 196  98  78 58  52  62  80 98 106 88 110 122 104
255 255 255 255 255 255 248 218 110 100 80  64  56  64 80  94 76  96 120 118
255 255 255 255 255 244 194 154  90  82 90 114 140 134 92  52 50  56  84 108
255 255 255 255 255 255 212 152  64  72 82  80  66  66 78  90 78  54  58  86
       grave 'c:\apl2\lixo11.bmp'
feito... Tempo gasto =      .5
```

Zera o verde

```
          leia 'c:\apl2\im11.bmp'
criados "blu" (matriz azul), "gre" (verde) e "red" (vermelho)
      gre[⍳5;⍳20]
205 206 207 208 197 169 126  95 81 69 62 73  90 95 79 62 76 91 96 87
208 203 200 200 196 180 151 127 78 70 59 56  61 70 79 83 74 85 89 80
200 200 197 191 181 168 151 138 84 79 71 63  59 63 71 77 68 77 87 86
203 203 199 187 170 148 124 106 74 70 74 88 101 98 77 56 54 57 69 81
188 184 179 178 177 164 133 105 61 66 70 69  64 64 70 75 68 56 55 69
       gre←(⍴gre)⍴0
       gre[⍳5;⍳20]
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 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 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
        grave 'c:\apl2\lixo11.bmp'
feito... Tempo gasto =      .5
```

Degradê central


```
       leia 'c:\apl2\im11.bmp'
criados "blu" (matriz azul), "gre" (verde) e "red" (vermelho)
       oba←degradecentral red
       red←255|oba×2
       grave 'c:\p\n\690\dcen.bmp'
feito... Tempo gasto =      .5
      ∇degradecentral[□]∇
    ∇
[0]   r←degradecentral x;meio;dist;lin;col;d
[1]   ⍝ pkantek - maio/2002 - v.1.1
[2]   ⍝ acha o meio da imagem e a distancia maxima. A partir daqui,
[3]   ⍝       faz o degrade do meio para fora
[4]   ⍝ ou seja, cada pixel e multiplicado pelo fator (dist ao centro)÷dist do c
entro ao pixel 0,0
[5]   r←(⍴x)⍴0
[6]   meio←⌊(⍴x)÷2
[7]   dist←⌊(+/meio*2)*0.5
[8]   lin←1
[9]  t1:→(lin>1↑⍴x)/fim
[10]  col←1
[11] t2:→(col>1↓⍴x)/ufa
[12]  d←⌊(+/((lin,col)-meio)*2)*0.5
[13]  r[lin;col]←⌊x[lin;col]×(dist-d)÷dist
[14]  col←col+1
[15]  →t2
[16] ufa:
[17]  lin←lin+1
[18]  →t1
```

Degradê no vermelho

```
      leia 'c:\apl2\im11.bmp'
criados "blu" (matriz azul), "gre" (verde) e "red" (vermelho)
     red[⍳5;⍳20]
197 198 203 204 197 170 130 100 90 78 73 84 101 106 90 73 87 102 107  98
200 198 196 196 196 181 155 132 85 79 70 67  72  81 90 94 85  96 100  91
195 195 193 187 181 169 155 143 91 86 80 72  68  72 80 88 79  91 101 100
198 198 195 183 170 148 128 110 79 77 83 97 110 107 86 67 68  71  83  95
183 179 175 174 177 164 137 109 66 71 77 76  73  73 79 86 82  71  71  85
      pt←×/⍴red
      p←⍳pt
      p←256|p
      p←p[⍋p]
      p←(⍴red)⍴p
      red←p
     red[⍳5;⍳20]
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
      grave'c:/p/n/690/degver.bmp'
feito... Tempo gasto =     1.0
```

Seno Coxabranca

```
      leia 'c:\apl2\im11.bmp'
criados "blu" (matriz azul), "gre" (verde) e "red" (ver
      red[⍳4;⍳20]
197 198 203 204 197 170 130 100 90 78 73 84 101 106 90
200 198 196 196 196 181 155 132 85 79 70 67  72  81 90
195 195 193 187 181 169 155 143 91 86 80 72  68  72 80
198 198 195 183 170 148 128 110 79 77 83 97 110 107 86
      red←senocoxabranca red
      blu←senocoxabranca blu
      red[⍳4;⍳20]
197 198 203 204 197 170 130 100  0  0  0 0 0 0 0 0 0 0
200 198 196 196 196 181 155 132 85  0  0 0 0 0 0 0 0 0
195 195 193 187 181 169 155 143 91 86  0 0 0 0 0 0 0 0
198 198 195 183 170 148 128 110 79 77 83 0 0 0 0 0 0 0
      grave 'c:\apl2\senocox.bmp'
feito... Tempo gasto =      .5
```

APL2 1001 - Object Editor - senocoxabranca

Object Edit Breakpoints Signals Options Windows Help

```
[0]  r←senocoxabranca x;meio;dist;lin;col;d;matcoef
[1]  ⍝ pkantek - janeiro/2004 - v1.3
[2]   r←matcoef←(⍴x)⍴0
[3]   meio←⌊(⍴x)÷2  ⍝ calcula as coordenadas do meio da fig
[4]   dist←⌊(+/meio*2)*0.5 ⍝ acha a distancia do meio ate 0
[5]   lin←1
[6]  t1:→(lin>1↑⍴x)/fim
[7]   col←1
[8]  t2:→(col>1↓⍴x)/ufa
[9]   d←⌊(+/((lin,col)-meio)*2)*0.5 ⍝ acha a distancia de x
[10]  →(0=col-meio[2])/ezero  ⍝ se esta na vertical, caia f
[11]  d←(¯3○(lin-meio[1])÷col-meio[2])÷1.570796327 ⍝ calcul
[12]  →soma
[13] ezero:d←((○1)÷2)÷1.570796327  ⍝ valor = pi/2
[14] soma:
[15]  matcoef[lin;col]←2|⌊10×(d+1)÷2  ⍝ d varia entre ¯1 e
[16] ⍝ entre 0 e 1 dai multiplica por 10 e toma o resto da
[17] ⍝ 10, varia-se o numero de raios. O resultado final aq
[18]  r[lin;col]←⌈x[lin;col]×matcoef[lin;col]
[19]  col←col+1
[20]  →t2
[21] ufa:
[22]  lin←lin+1
[23]  →t1
[24] fim:MATCOEF←matcoef
```

Usando a média de 4 pixels

- neste caso desconsiderando quem não têm os 4 vizinhos
- Só para ver o que o algoritmo faz

continuando

comparando o primeiro com o último

JPG

- JPG significa JPEG = Grupo Unificado de Especialistas em Fotografia
- É um padrão de armazenamento de imagens com perdas
- Adequado a fotografias
- Estabelecido em 1991
- Mínimo de 16 níveis de cinza
- Imagens até 65536 $\times$ 65536 pixels
- Suportado por todos os browsers
- Padrão de fato nas máquinas fotográficas
- Sob este título há mais de 40 padrões
- vide em http://www.jpeg.org/

Porque comprimir

- Uma imagem monocromática de 256 niveis de 512 $\times$ 512 ocupa 262KB
- a mesma imagem true color ocupa 786KB
- uma imagem 3000 $\times$ 2000 true color ocupa 17MB
- o olho não percebe tanto detalhe
- JPG colorido com 20:1 não apresenta distorção visível
- com 50:1 distorções pequenas a moderadas
- 100:1 adequado para imagens tipo *preview*
- JPG monocomático, distorção a partir de 5:1

perdas

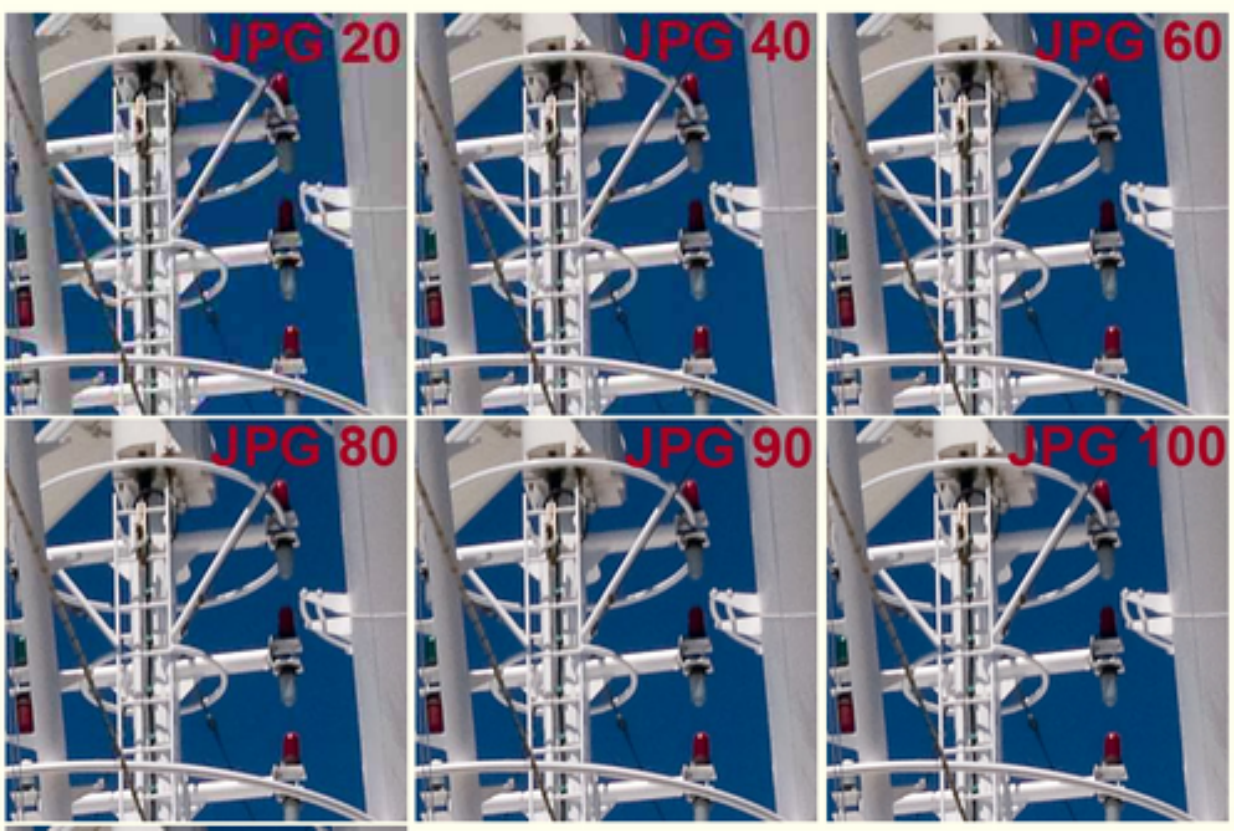
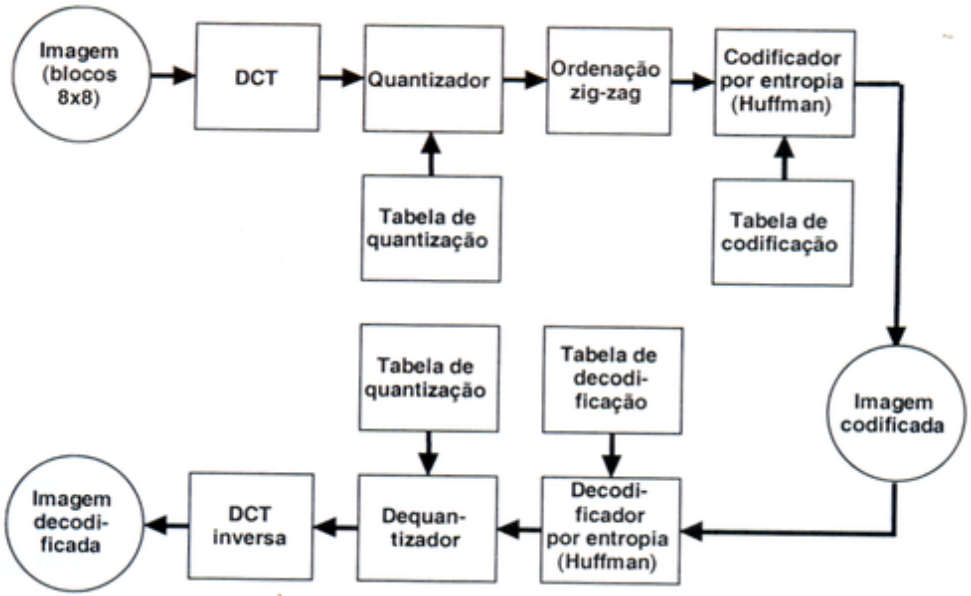


Imagem	características	tamanho
300 x 300	3 RGB	264KB
JPG 100	29%	77KB
JPG 90	11%	29KB
JPG 80	8%	21KB
JPG 60	5%	14KB
JPG 40	4%	11KB
JPG 20	3%	8KB

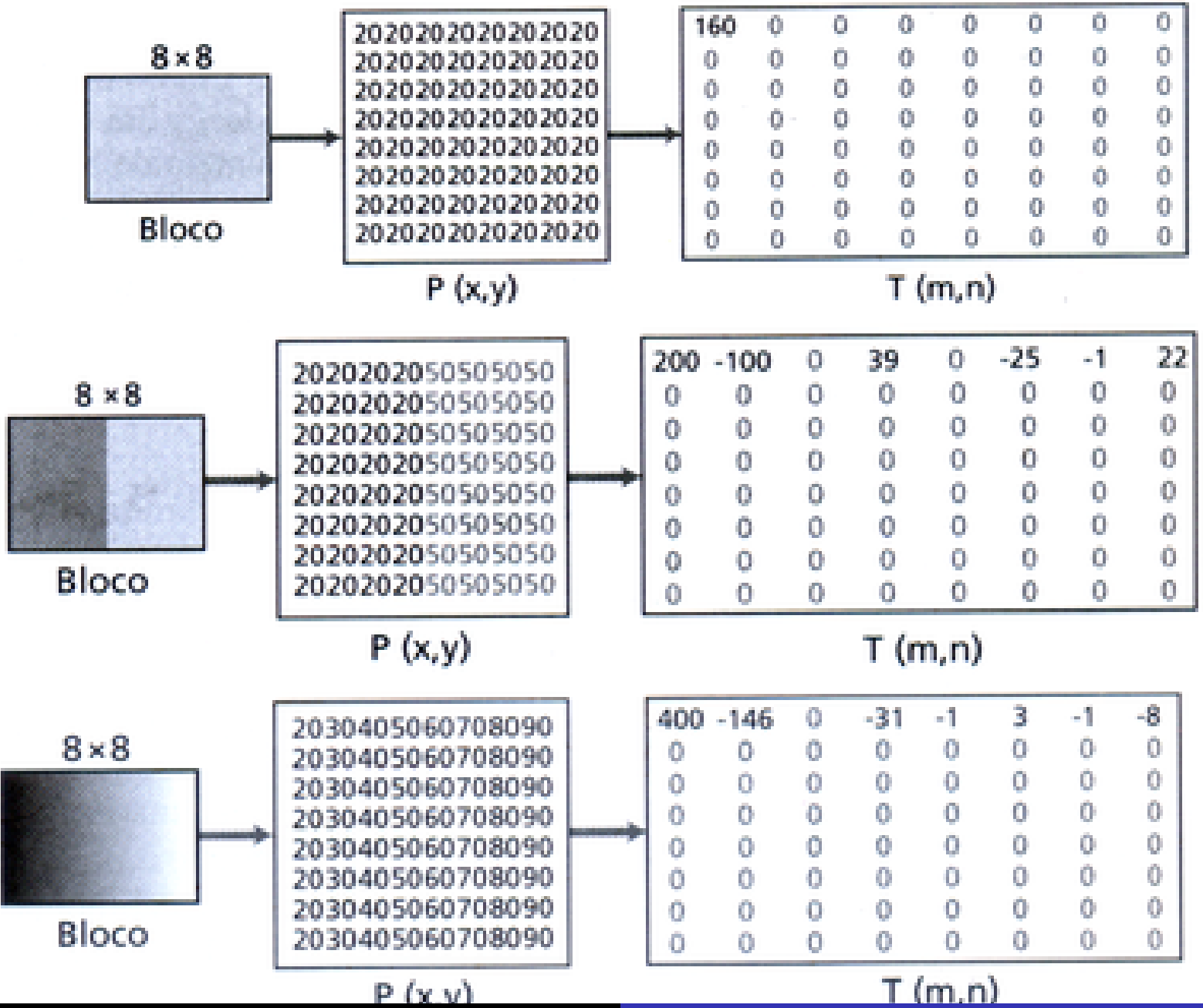
esquema do processo



Processo

- A imagem é dividida em blocos de 8×8 pixels
- Aplica-se a transformada discreta do cosseno (1974)
- o elemento [1,1] contém o valor médio do bloco (chamado DC)
- Na linha 1 e coluna 1 aparecem as variações (chamadas AC)
- A transformada é inversível

DCT


8 × 8
Bloco
202020202020202020
202020202020202020
202020202020202020
202020202020202020
202020202020202020
202020202020202020
202020202020202020
202020202020202020
P (x,y)
160 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 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
T (m,n)
8 × 8
Bloco
202020205050505050
202020205050505050
202020205050505050
202020205050505050
202020205050505050
202020205050505050
202020205050505050
202020205050505050
P (x,y)
200 -100 0 39 0 -25 -1 22
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 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
T (m,n)
8 × 8
Bloco
203040506070809 0
203040506070809 0
203040506070809 0
203040506070809 0
203040506070809 0
203040506070809 0
203040506070809 0
203040506070809 0
P (x,y)
400 -146 0 -31 -1 3 -1 -8
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 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
T (m,n)

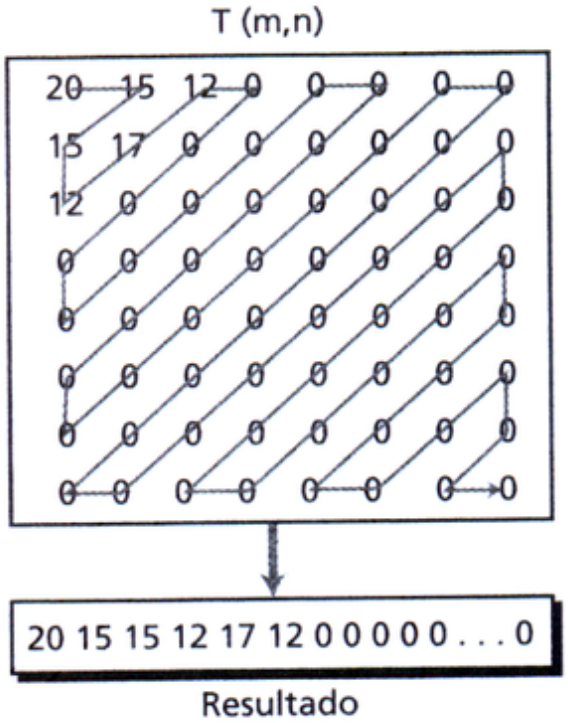
Quantização

- Após a DCT a imagem é discretizada
- Cada número é dividido por uma constante
- A fração decimal é desprezada
- Esta é a fonte de perdas no processo
- É a única etapa irreversível

Compressão

- Agora a matriz é lida em zig-zag
- Finalmente este resultado é comprimido
- usando Huffman ou codificação aritmética
- Para ler uma imagem o processo é seguido em ordem inversa

zig zag


T (m,n)
20 15 12 0 0 0 0 0
15 17 0 0 0 0 0 0
12 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 0 0 0
0 0 0 0 0 0 0 0
20 15 15 12 17 12 0 0 0 0 0 . . . 0
Resultado

Arquivo JPG

- JPG não é um formato de arquivo
- Um padrão (coleção de algoritmos) de tratamento de imagens
- Os arquivos JPG são JFIF = *JPG file interchange format*